Solaris Performance Management

GRIGALE

www.grigale.com

By Haim Tzadok

haim.tzadok@grigale.com

Grigale Ltd

Table of Contents

Introducing Performance Management

Objectives

- Principles of memory tuning
- Performance tuning process
- Terms and definitions

Principles of performance tuning

Aspects of Performance tuning

Managers – Consistent, High system throughput.

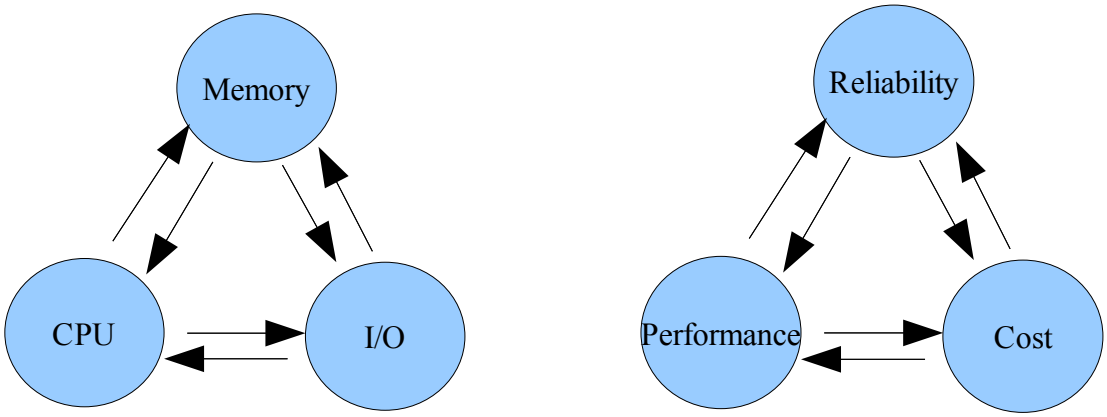
End Users – Low application response time

Programmers – fast access and sufficient system resources.

Relationships in Performance tuning

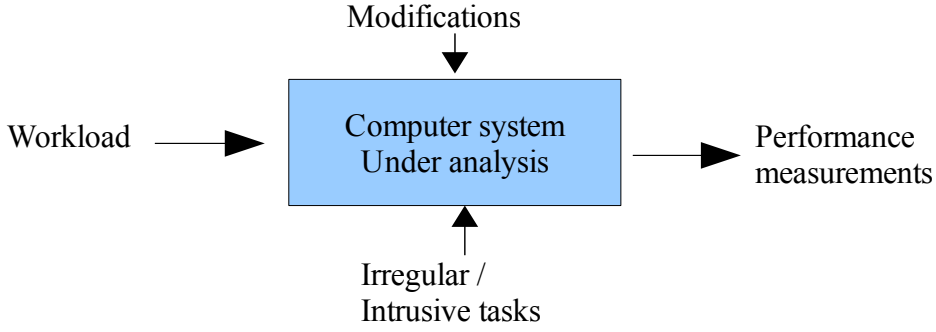
Certain adjustments can improve efficiency in one aspect at the expense of another.

For example, improving response time of an I/O-intensive application by allocating more system memory will probably increase page-in/out activity.



Conceptual model of performance

The conceptual performance model describes the system and the factors that make demands on its resources. Factors that influence system performance:



Workload – the processes that are normally scheduled to run on the system.

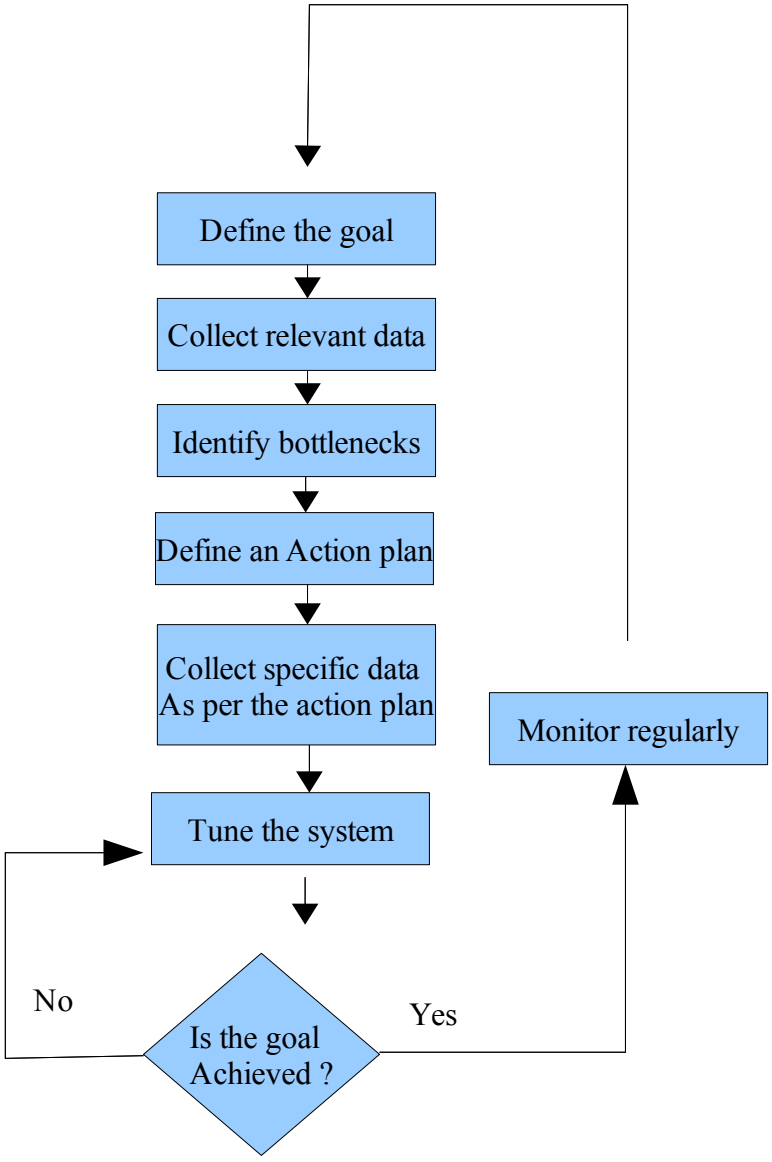
These processes have:

- CPU activity
- Memory consumption
- Network contention
- Changes in usage throughput the day
- Intense database or file-system interaction

Irregular/ intrusive tasks – Some processes may be poorly designed to efficiently use the system.

Modifications – Changes in system resources, including device driver, firmware or kernel updates and changes in application behavior are all a potential source of performance changes.

General Tuning procedure



Tunable areas of a system

Area	Example	Ideal	Practical
Kernel parameters	ufs_LW, ufs_HW	4	3
DB	IPC parameters	2	2
Application SW	Source code, Shell environment, resource limits	1	1
HW resources	NIC link speed Disk seek times Memory service time	3	4

1 – highest priority … 4 – lowest priority

Basic Terms and definitions

Response time

Response time = Wait time + Service time

Service time

The time to process a request

Wait time

The time it take to wait till we get a service time.

(This can be due to busy resources, lockings etc...)

Bandwidth time

The maximum data that can be processed or passed.

Throughput

The average amount of data that can be moved from one component to another in a time interval.

Utilization

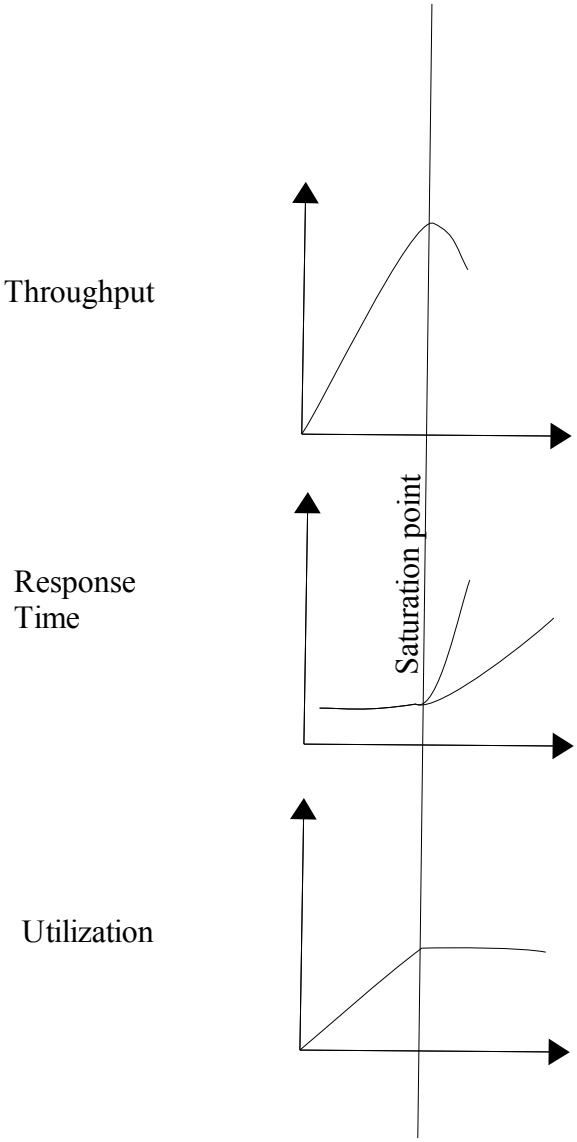
The percentage of elapsed time that a component or service spends processing requests.

This is a very tricky to interpret from several reasons:

Low utilization can be due to lack of available work. Or due to lack of underestimating of the production capacity of the system.

High utilization can be due to efficient use or due to bottleneck.

Performance graphs



It is very recommended to visualize, performance data in order to better understand system behaviour.

This Graphs show the relationships between performance indicators.

It is assumed that when CPU %sys > %user
Then your system is probably not optimized to run user processes but serves your system instead of serving your users. This situation mostly requires thorough performance analysis.

Monitoring Tools

ProcFS

Procfs - p-commands (/usr/proc/bin), prstat, ps, truss, preap

procfs is a file-system that provides access to the state of each process in the system.

Procfs is derived from plan9 (UNIX successor) and is implemented in Solaris.

The directory /proc containes sub-directories where each process is a sub-directory represented by its pid.

Each process sub-directory contains a well defined hierarchy of files and sub directories that provide interface to access process information.

KSTAT

Kernel statistics – kstat, vmstat, iostat, mpstat, sar

As opposed to many OS that have user programs that check the system introducing greater intrusion effect on the OS. The Solaris OS counts and stores performance relevant information as part of it's normal operation – through kstat. The Solaris OS also provides tools that report kstat data to the administrator.

kstat -m eri – shows all statistics found on eri network adapter driver.

kstat -m cpu_info – shows cpu component properties.

kstat -l – shows all available statistics on the system.

each kstat displays as a fully qualified string of the form:

driver:instance:kernel_struct:kstat

driver – the kernel driver

instance – the number of the driver in the system (kernel instance)

kernel_struct – kernel data structure that contains the kstat data.

Ks tat – the name of the statistics element.

/usr/include/sys/kstat.h includes structures that can be used in applications to supply kstat-formatted data to the kernel.

Dtrace

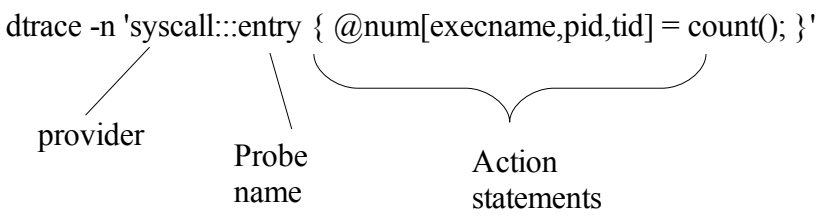
Dtrace – lockstat, plockstat, intrstat, dtrace

Dtrace is a new dynamic tracing framework intrudced in Solaris 10.

Dtrace was written by Bryan Cantrill, Adam Leventhal and Michael Shapiro (Distinguish kernel programmers)

Dtrace allows dynamic tracing of events in both user and kernel mode.

Dtrace one-liner examples:



Dtrace

Objectives

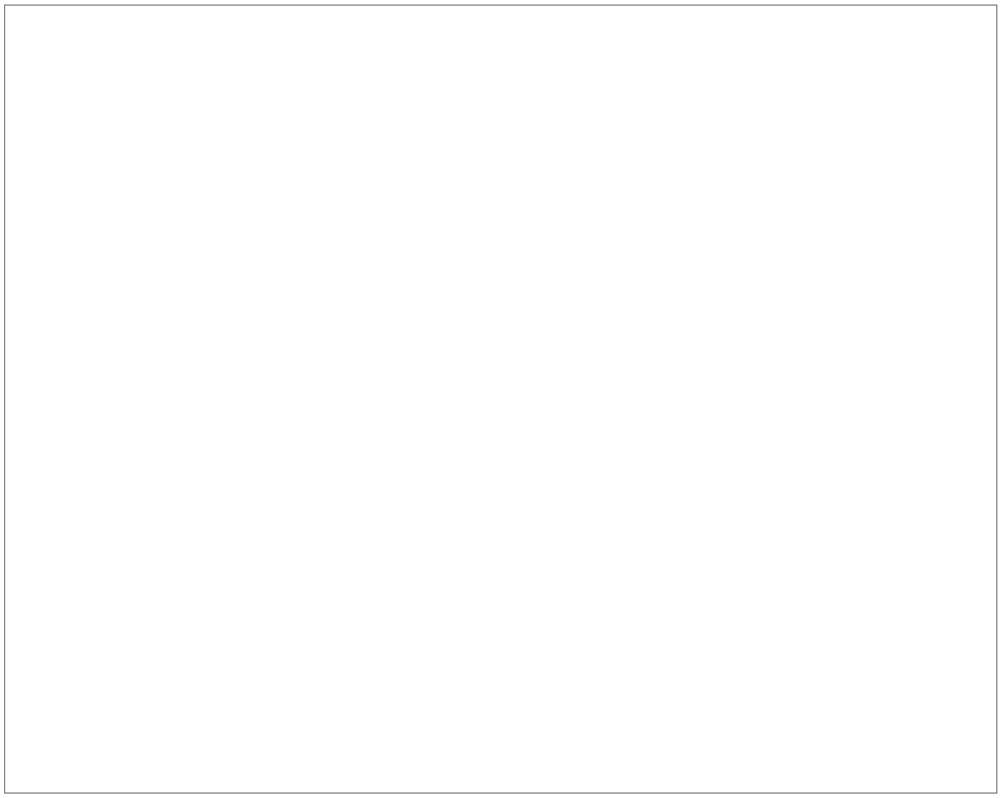
- Introduction
- Dtrace Architecture
- Providers
- Probes

Introduction

What is Dtrace ? - Dtrace, along with the D scripting language, is a powerful tracing and analysis tool introduced in Solaris 10. Dtrace can be used to examine the behavior of kernel or user programs.
More than 50,000 probes for probing Solaris and SunOS kernel are inside Solaris 10. Probes cause no overhead until they are enabled (fired).

Since D language is implemented to be run safely in its core architecture, it can be used by System administrator or application developers even on live production systems. Users of Dtrace can create custom programs with the D scripting language to dynamically instrument the system.

Dtrace architecture



Dtrace concepts

Probe – A probe is a point of instrumentation in Solaris 10 kernel or any Solaris 10 user program. A probe is a specific point in kernel or user program source code in a way to record or display relevant information about this point.

Provider – A logical name for a set of probes inside Dtrace.

Here are some of the providers exist in Solaris:

dtrace provider – provides BEGIN, END and ERROR probes.

Syscall provider – provides system call probes.

Lockstat – for tracing access to synchronization objects

sysinfo - provides probes for the sys named kernel statistics corresponding to `kstat -n sys`

mib – provides probes used by Solaris Management information base used by SNMP.

fbt – function boundary trace for tracing entry and return for kernel functions

pid - allows tracing of entry and return of any function in a user process, and any instruction specified by an absolute address or function offset.

IO – provides probes related to disk I/O and NFS.

plockstat – allows tracing of user-level synchronization objects.

profile - for time-based interrupt firing at fixed intervals.

proc – probes for process and lwp creation , termination, execution and signal handling and delivery.

sched - for traping CPU scheduling events such as context switching

vminfo - for tracing access to user-level synchronization objects.

sdt – Static Defined Tracing – allows programers to choose locations of interest to Dtrace users.

fpuinfo - for tracing kernel simulation of hardware floating point instructions on SPARC architectures.

Module – The name of the software where the probe resides. (It can be a kernel module or a user application section where the code is defined

Consumer – Any process that interacts with the Dtrace framework and get information from it.

IP provider – exits only on OpenSolaris build 93 and more.

Probe descriptions

A probe can display the following information:

- Any argument that is passed to the function
- Any global variable in the kernel
- A timestamp that indicates when the function was called
- A stack trace that indicates the section of code that called the function
- The process that was running at the time the function was called
- The thread that made the function call

Any probe in the system is identified by a 4-tuple (`provider:module:function:name`) . This avoids name clashes in the kernel, which is a code base of millions of lines of code.

Probe names for example:

syscall::read:entry

profile:::tick-5sec

To view all probes on the system:

dtrace -l

to view available probes belonging to the syscall provider :

dtrace -ln syscall:::

To set a probe(fire a probe) leave of the -l

dtrace -n syscall::read:entry

Now we can add a tracing function like trace which will print any time the probe is matched:

dtrace -n 'syscall::read:entry { trace(execname) }'

Dtrace examples:

How do I see in realtime whenever a new process is created:

```
/usr/sbin/dtrace -n 'proc:::exec{printf("%s execing %s, , uid/zone =%d/
%sn",execname,args[0],uid,zonename)}'
```

```
CPU     ID                    FUNCTION:NAME
 0  19298                 exec_common:exec hatimerun execing /sbin/sh, , uid/zone =0/globaln
 0  19298                  exec_common:exec sh execing /opt/traffixsystems/bin/probenc, , uid/zone
=0/globaln
 0  19298                  exec_common:exec probenc execing /usr/bin/grep, , uid/zone =0/globaln
 1  19298                  exec_common:exec gds_probe execing /usr/cluster/bin/hatimerun, , uid/zone
=0/globaln
 1  19298                  exec_common:exec probenc execing /usr/bin/pgrep, , uid/zone =0/globaln
 1  19298                  exec_common:exec bash execing /usr/bin/ls, , uid/zone =0/globaln
```

How do I determine block size of the disk I/O on the system:

```
/usr/sbin/dtrace -n 'io:::start{@[execname, args[2]->fi_pathname] = quantize(args[0]-
>b_bufsize)}'
```

Variables types

Dtrace supports most common variable types such as:

integer

string

pointer

dtrace built in variables (like: pid – process id, execname – process name, tid – thread id, arg0 … arg9 – first 10 probes args, args[ ] - list of all probe arguments, curpsinfo – pointer to process information, curlwpinfo – pointer to lwp information, curthread – pointer to kernel thread, timestamp, probeprov/mod/func/name – probe provider/mod/func/probe name respectively)

In addition we also have: Structs, scalar arrays and associative arrays.

Aggregations

Aggregations are statistical functions that help process data automatically.

Common aggregation functions are:

count – number of times called

min – smallest value called

max – largest value called

sum – sum value of calls

avg – average value of calls

quantize(expr) – power of two frequency distribution of calls

lquantize(expr,val1,val2,step) – linear quantizing of expression val1 – the starting value, val2 – the ending value, step – the step value. (see example bellow).

Example usage: @arrayname[keys] = aggfunc(args)

dtrace -n 'syscall::read:entry { @sum[execname]= count() }'

A simple quantize example (malloc.d):

```
/usr/sbin/dtrace -s
pid$1:libc:malloc:entry
{
            @[“malloc sizes”] = quantize(arg0) ;
}
```

./malloc.d 658

output example:

```
malloc sizes
value  ------------- Distribution ------------- count
0                                       | 0
626                                     | 1
5560                                    @| 2
7578                                    @@| 4
32733                                   @@@@@@@@| 8
61198                                   @@@@@@@@@@@@@@| 16
69568                                   @@@@@@@@@@@@@@@@| 32
10726                                   @@| 64
6272                                    @| 128
1305                                    | 256
886                                     | 512
3146                                    @| 1024
```

Another example:

```
syscall::write:entry
{
        self->ts = timestamp;
}

syscall::write:return
/self->ts/
{
        @time[execname] = quantize(timestamp - self->ts);
        self->ts = 0;
}
```

Quantize in this example will display a binary distribution of the time elapsed from starting the write system call to the end of the write system call.

A simple lquantize example (lmalloc.d):

```
/usr/sbin/dtrace -s
pid$1:libc:malloc:entry
{
            @[“malloc sizes”] = lquantize(arg0,4,8,1) ;
}
```

./lmalloc.d 658

output example:

```
[root@n1: /] #./lmalloc.d 24951
dtrace: script './lmalloc.d' matched 1 probe
^C

  malloc sizes
          value  ------------- Distribution ------------- count
            < 4 |                                       257
              4 |                                       292
              5 |                                       635
              6 |                                       242
              7 |                                       65
      >= 8 |@@@@@@@@@@@@@@@  70551
```

Predicates

predicates are similar to filters which can filter probes.

For example the predicate `/self->ts/` founded in the example before will filter only the probes that have self->ts configured.

References

http://docs.sun.com/app/docs/doc/817-6223 **-** dtrace guide in docs.sun.com .

http://www.usenix.org/event/usenix04/tech/general/full_papers/cantrill/cantrill_html/ - Dtrace Usenix '04 paper.

http://opensolaris.org/os/community/dtrace/dtracetoolkit/ - Dtrace toolkit project website.

Solaris Monitoring Tools

Objectives

- Describe and use monitoring tools provided with the Solaris OS
- Describe and use kstat based utilities
- Describe and use procfs based utilities
- Describe online resources for third-party utilities
- Enable system accounting

Identifying Performance problems using the Solaris OS Monitoring tools.

Tool	Reports
sar	Overall performance of the system
vmstat	Virtual memory statistics
iostat	I/O statistics
mpstat	per-processor statistics
netstat	Network statistics
nfsstat	NFS statistics
lockstat	Locking statistics
busstat	Bus statistics
fsstat	File system statistics
poolstat	Pool statistics
rcapstat	Resource cap enforcement statistics
prstat	Active process statistics
intrstat	Interrupts statistics
cpustat	CPU statistics
fmstat	Fault management module statistics
ipfstat	Packet filter statistics
medstat	Mediator hosts statistics
metastat	Meta device statistics
zpool iostat	ZFS statistics

Sar – System Activity reporter suite

SAR suite of utilities was first introduced in Solaris as a collecting mechanism for System statistics.

Sar provides reports on the following:

- File system and System calls
- Physical and kernel memory usage
- Paging and swap activities
- Block device and Terminal Type Identifier (TTY) activities
- Process status
- IPC and CPU usage activity

Sar suite of utilities contains sar, sadc, sa1, sa2. (found under - /usr/lib/sa/)

sadc – the actual program that access and collect system data. It writes all it data in binary format.

sa1 - stores binary form information in /var/adm/sa directory

sa2 – produces textual reports from a raw data.

Example output:

```
[root@n1: /] #sar 2 4

SunOS n1.grigale.com 5.10 Generic_139555-08 sun4u    09/21/2009

00:07:54   %usr   %sys   %wio   %idle
00:07:56     0      0      0     100
00:07:58     0      0      0      99
00:08:00     0      0      0     100
00:08:02     0      0      0      99

Average      0      0      0      99
```

%usr – The percentage of time that the CPU is running in user mode.
%sys – The percentage of time that the CPU is running in system mode.
%wio – In Solaris 10, this statistic is no longer used.
%idle – The percentage of time for which the CPU is idle.

sar -c reports system call activity.

`scall/s` – The number of system calls per second.
`sread/s` – The number of read system calls per second.
`swrit/s` – The number of write system calls per second.
`fork/s` – The number of fork system calls per second.
`exec/s` – The number of exec system calls per second.
`rchar/s` – The number of characters read per second.
`wchar/s` – The number of characters written per second.

To get more details who is performing large system call activity you may run dtrace one-liner:

```
dtrace -n 'syscall:::entry{@[execname,pid] = count();}'
```

Virtual Memory related statistics – vmstat

vmstat – provides the following activities:

- process, interrupts, cpu activity and cpu cashes activity.
- Virtual memory and swaping activities.
- Disk activity.

`Kthr` - Report the number of kernel threads in each of the three following states:

r	the number of kernel threads in run queue
b	the number of blocked kernel threads that are waiting for resources I/O, paging, and so forth
w	the number of swapped out light-weight processes (LWPs) that are waiting for processing resources to finish.

`Memory` - Report on usage of virtual and real memory.

swap	available swap space (Kbytes)
free	size of the free list (Kbytes)

`page` - Report information about page faults and paging activity. The information on each of the following activities is given in units per second.

re	page reclaims
mf	minor faults
pi	kilobytes paged in
po	kilobytes paged out
fr	kilobytes freed
de	(deficit) anticipated short-term memory shortfall (Kbytes)
sr	pages scanned by clock algorithm

`disk` - Report the number of disk operations per second. There are slots for up to four disks, labeled with a single letter and number. The letter indicates the type of disk (s = SCSI, i = IPI, and so forth); the number is the logical unit number.

`Faults` - Report the trap/interrupt rates (per second).

Vmstat -p :
paging activity can be expanded by using the -p option

`epi`	Executable page-ins.
`epo`	Executable page-outs.
`epf`	Executable page-frees.
`api`	Anonymous page-ins.
`apo`	Anonymous page-outs.
`apf`	Anonymous page-frees.
`fpi`	File system page-ins.
`fpo`	File system page-outs.
`fpf`	File system page-frees.

iostat command

iostat utility reports about the following:

- CPU utilization and TTY activity
- Disk activity and device errors
- NFS mount activity

`r/s` – number of reads
`w/s` – number of writes
`kr/s` – number of Kbytes read
`kw/s` – number of Kbytes written
`wait` – average number of transactions waiting for service
`actv` – average number of transactions that are being serviced
`svc_t` – average service time in milliseconds
`%w` – percentage of time there are transactions waiting for service
`%b` – percentage of time for which the disk is busy
`tin` – number of characters read from a terminal
`tout` – number of characters written to a terminal

Please note that iostat may not feet when using external volume managers. For zfs use: 'zpool iostat', for VxVM use: 'vxstat'

mpstat command

mpstat reports about the following:

- cpu activity on a per-processor basis
- (-a) Aggregate by processor set
- (-p) reports processor set membership

`CPU` – The CPU ID on the system
`minf` – number of minor faults
`mjf` – number of major faults
`xcal` – number of inter-processor cross-calls
`intr` – number of interrupts
`ithr` – number of interrupts handled as threads
`csw` – number of context switches
`icsw` – number of involuntary context switches
`migr` – number of thread migrations to other processors
`smtx` – number of times locks are not acquired at the first attempt on mutexes
`srw` – number of times locks are not acquired at the first attempt on reader and writer locks
`syscl` – number of system calls
`usr` – percentage of user time of the CPU
`sys` – percentage of system time of the CPU
`wt` – percentage of CPU wait time (deprecated in Solaris 10)
`idl` – percentage of idle time of the CPU

netstat command

netstat reports about the following:

- (-i) interface state.
- (-r) routing table.
- (-D) dhcp statistics.
- (-m) streams statistics.
- (-M) multicast routing table.

For netstat -i :

`Name` – The name and instance of the Ethernet interface
`Mtu` – The maximum size of the packets that are transmitted on the network
`Net/Dest` – The network to which the interface is attached
`Address` – The address of the interface on the attached network
`Ipkts` – The number of packets received
`Ierrs` – The number of received packets that have errors
`Opkts` – The number of packets sent
`Oerrs` – The number of sent packets that incurred errors
`Collis` – The number of output packets that result in collision
`Queue` – The number of packets that are queued

nfsstat command

nfsstat reports NFS statistics

- NFS client statistics
- NFS server statistics
- nfs file system statistics
- RPC statistics

nfsstat |grep ':' - will allow to show nfs main sections.

The procfs utilities

Tool	**Reports**
pflags	
pcred	Prints credential of the process
pmap	Prints VM mapping
pldd	Prints .so (shared objects) used by the process.
psig	Prints signal disposition table
pstack	Prints stack trace for each process thread (LWP)
pfiles	Prints open files.
pwdx	Prints current working directory of the process.
pstop	Stops the process
prun	Continue the process
pwait	Waits till the process exits.
ptree	Shows genealogical tree for the process.
ptime	Prints real, user and sys time for taken to execute a command
preap	Forces a defunct process to be reaped by its parent
pargs	Prints command line arguments for the process

Tunable parameters

Objectives

- View Tuning parameters per subject
- Setting tuning parameters

Using Tools to View Tuning Parameters

Function	Description
Boot	/etc/system system configuration file for customizing kernel parameters, module loading/unloading and boot device and file-system information. Read only once at boot time. Bootadm – boot administration command
Network	ndd – used for setting network driver parameters and tcp/ip parameters. dladm – command for network settings route – manipulate routing tables routeadm – IP forwarding and routing configuration.
Configuration files	/etc/default – default behavior configuration files. /usr/kernel/drv or /kernel/drv – various drivers configuration files.
Custom commands	prctl – process control routeadm – routing settings ifconfig – network temporary settings tunefs – filesystem tuning settings sysdef – output system definition, device drivers and kernel tunable parameters.
Mdb	Enables viewing of kernel parameters and modifying kernel parameters while the system is running.
Dynamic tracing	Dtrace – dynamic tracing.

Changing kernel parameters using MDB

Start mdb with kernel debugging mode (-k). Writable switch (-w) allows changing of kernel parameters online.

[root@n1: /] #mdb -kw
Loading modules: [unix genunix dtrace specfs ufs pcisch ssd fcp fctl qlc ip hook neti sctp arp usba s1394 nca md lofs zfs sd cpc audiosup random crypto wrsmd fcip logindmux ptm sppp nfs ipc]

>

In order to view lotsfree kernel parameter size use:
>::nm ! egrep 'Shndx|\|lotsfree$'

Watch the value of lotsfree parameter using /E
>lotsfree/E

In order to set lotsfree kernel parameter value use $W and then /Z

>lotsfree/Z 0x14CCB8 (which means - 1363128 bytes)

NOTE – avoid changing kernel parameters using mdb on production system. As it would take one typo to cause a system panic.

Debugger commands (dcmd's)

> A debugger command, or dcmd (pronounced dee-command) in mdb terminology, is a routine in the debugger that can access any of the properties of the current target. mdb parses commands from standard input, and then executes the corresponding dcmds. Each dcmd can also accept a list of string or numerical arguments, as shown in the syntax description below. mdb contains a set of built-in dcmds, described below, that are always available. You can also extend the capabilities of mdb itself by writing your own dcmds, as described in the Solaris Modular Debugger Guide.

in order to view available dcmd's use -
>::dcmds - list all dcmds
>::dmods -l - list all dcmds and walkers per module.

Read modifiers

E decimal unsigned long long (8 bytes)
D Decimal signed int (4 bytes)

Write modifiers

W Write the lowest 4 bytes of the value of each expression to the target beginning at the location specified by dot.

Z Write the complete 8 bytes of the value of each expression to the target beginning at the location specified by dot.

For listing more modifiers use:
>::formats

Logging output to a file:
>::log mymdb.out

Viewing kernel statistics using dtrace

```
# cat kernel_params.d
#! /usr/sbin/dtrace -qs
BEGIN
{
printf("%12s %12s %12s\n", "Processes", "Threads",
"Freemem" );
}
tick-3sec
{
printf("%12d %12d %12d \n", `nproc, `nthread,
`freemem*8 );
}
```

Output example:
[root@n1: /root] #./kernel.d

Processes	Threads	Freemem
84	620	1403264
84	620	1403264
84	620	1403264
84	621	1403264
84	621	1403264

Network Parameters

In order to view network parameters:

TCP/IP parameters

[root@n1: /root] #ndd /dev/tcp \?
? (read only)
tcp_time_wait_interval (read and write)
tcp_conn_req_max_q (read and write)
tcp_conn_req_max_q0 (read and write)
tcp_conn_req_min (read and write)
...
tcp_keepalive_interval (read and write)
tcp_xmit_hiwat (read and write)
tcp_xmit_lowat (read and write)

tcp_recv_hiwat (read and write)

UDP parameters

```
[root@n1: /root] #ndd /dev/udp \?
?                       (read only)
udp_wroff_extra             (read and write)
udp_ipv4_ttl                (read and write)
..
udp_do_checksum             (read and write)
udp_smallest_anon_port      (read and write)
udp_largest_anon_port       (read and write)
udp_xmit_hiwat              (read and write)
udp_xmit_lowat              (read and write)
udp_recv_hiwat              (read and write)
udp_max_buf                 (read and write)
udp_ndd_get_info_interval   (read and write)
..
udp_status                  (read only)
udp_bind_hash               (read only)
```

Viewing IP parameters

viewing ip_forwarding value -
ndd [-get] /dev/ip ip_forwarding
0

Setting ip_forwarding value

ndd -set /dev/ip ip_forwarding 1

another ways to set ip_forwarding:

1. touch /etc/notrouter
2. routeadm -u -e ipv4-forwarding (-d to disable).

Using sysdef

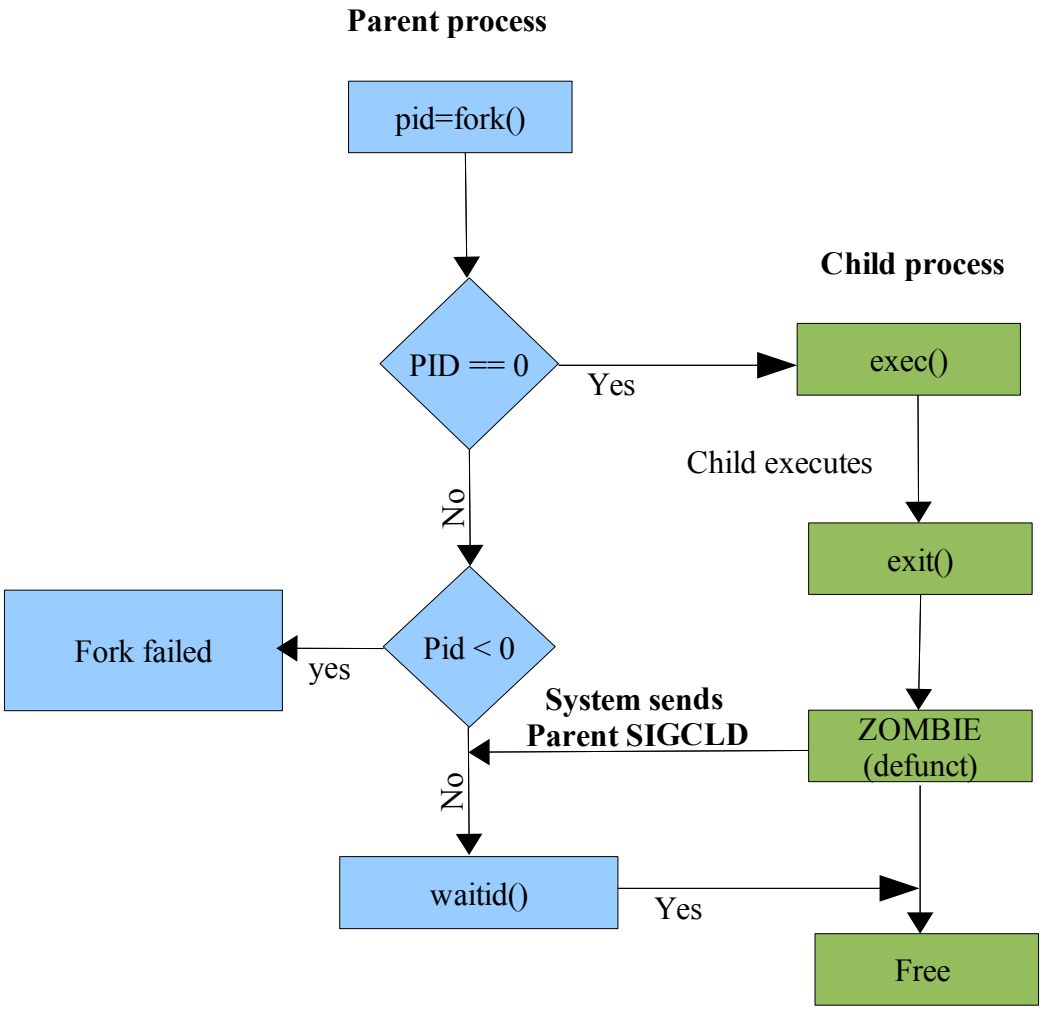
sysdef under tunable parameters section.
Bufhwm (buffer cache high water mark)
v.v_maxup (maximum processes per user)

Processes and Threads

Objectives

- The Process
- The Thread
- Threading Models
- Monitor processes
- locks

The Process



Process is one of the fundamental concepts in UNIX systems.

When any program is running in the system it runs as a process.

Process is created by the fork system call.

Process contains address space which holds: text, data, stack and heap segments.

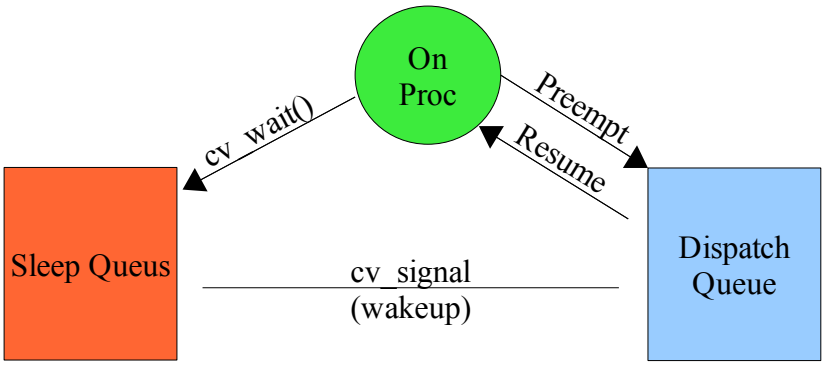
Process can contain one (single threaded) or more threads (multi threaded).

Each user thread in the system is served by a kernel thread or kernel lwp (light weight process)

Process states

Ready – the process is eligible for CPU time
On Proc – the process is served by a CPU
Zombie – the process has terminated but stil has proc structure
Sleep – The process thread is waiting for a resource
Stopped – the thread is stopped by a signal or debugger

Thread/Process states diagram



Context switching

When a process is running on a CPU, all of process environment variables are loaded into the cpu. This process is called context switching.

(csw)voluntary context switch – occurs when a thread goes to sleep queue prior to finishing it's time quantum. (due to a wait for resource, I/O wait, lock release or timer operation)

(icsw)involuntary context switch – when a thread finishes it's time quantum or if another thread with higher priority is eligible to run.

Preemption – the process of stopping one thread of running on the expense of running a higher priority thread .

Average context switch on USIII – 5-7 millisecond.

cswstat.d

```
#pragma D option quiet

dtrace:::BEGIN
{
     /* print header */
     printf("%-20s  %8s %12s %12s\n", "TIME", "NUM", "CSWTIME(us)",
         "AVGTIME(us)");
     times = 0;
     num = 0;
}

sched:::off-cpu
{
     /* csw start */
     num++;
     start[cpu] = timestamp;
}

sched:::on-cpu
/start[cpu]/
{
     /* csw end */
     times += timestamp - start[cpu];
     start[cpu] = 0;
}

profile:::tick-1sec
{
     /* print output */
     printf("%20Y  %8d %12d %12d\n", walltimestamp, num, times/1000,
         num == 0 ? 0 : times/(1000 * num));
     times = 0;
     num = 0;
}
```

The Thread

Each thread in the process has the same text, data and heap segments as other threads in the process. Each thread has a unique stack which belongs only to it.

Has its own stack but shares the rest of address space, data, files and context information with other threads in the process.

All threads in a process share the same PID but have their own thread id(TID/LWP).

ps -cLef - will also show threads. (-c scheduling properties, -L – list LWP's)

Thread models

T1 thread model – MXN Multi level thread model implementation
A new thread is created with lwp_create kernel function. Each user level thread can be mapped into LWP kernel thread.

Pros:

- Fast user thread create and destroy
- Minimize the number of LWP's on the system
- user minimal kernel memory
- no system call required for synchronization
- process private synchronization only
- can have thousands of threads
- Fast context switching

Cons:

- Complex, tricky programming
- Signal delivery
- compute bound threads do not surrender
- complex to maintain

T2 thread model – Single level thread model
1X1 thread model implementation. Each user thread is mapped to one LWP kernel thread.

- All user threads bound to LWP's.
- Kernel level scheduling – no user level scheduling
- simplified implementation
- Uses kernel synchronization objects.
- More expensive thread create/destroy, synchronization
- More responsive scheduling, synchronization

In Solaris 8 you could choose one of these two threading models.

In Solaris 9-10 only T2 exists.

Performance issues

Why use multithreading

- Simplifies tasks by breaking one single task into smaller tasks
- Usually scales better on SMP.
- Shares address space resources.
- Avoid overhead and complex of IPC
- reduces the demand of MMU address translations.
- Creating thread is 5-6 times less expensive than a process.

Locking

In order to avoid two threads or more from accessing the same data while being changed by one of them, programmers must use locks.

Most common use for locks is to protect writable shared data. Fine-grained strategy will implies more locks to protect smaller areas of data, for faster and more scalable concurrent processing.

More locks Fast access on the expense of Complex code and Memory consumption

Kernel Locking primitives

Lock	Description
Mutex (Mutual exclusion lock)	Provides exclusive access to data. The state of the lock is associated with the critical data.
Conditional variable	A condition variable provides a thread an address to go to sleep, until a condition is true. Access to the condition variable is protected with a mutex lock.
Counting semaphore	Controls thread access to a number of resources. As each thread takes a resource the count of the number of resources is decremented. When the count goes to 0 subsequent threads requesting a resource will sleep. When a thread gives up a resource it will wake up any threads waiting.
Multiple-reader, single-writer	Read access to the data is allowed at any time, but only one thread can acquire the lock in order to write data.

Good threading design is a great challenge.

Locking problems

- **Deadlock** - A thread acquires Lock 1 and needs Lock 2. Another thread has already acquired Lock 2 and needs Lock 1. Since no thread can release it's locks - which leads to deadlock.
- Thread dies while acquiring lock in a code section, without releasing.
- **Race condition** – two threads are trying to access the same data, the outcome is different result depends on the winner.
- Multiple locks are acquired in order but are not released in the same order, leading to deadlock or race condition.

Lockstat – reports statistics on kernel-level locking. (using /dev/lockstat device)

```
lockstat -H -D 4 sleep 6
```

`Count` - Number of times this event occurred, or the rate (times per second) if -R was specified.
`indv` - Percentage of all events represented by this individual event.
`Genr` - Percentage of all events generated by this function.
`cuml` - Cumulative percentage; a running total of the individuals.
`rcnt` - Average reference count
`nsec` - Average duration of the events in nanoseconds, as appropriate for the event.
`Lock` - Address of the lock; displayed symbolically if possible.
`CPU+PIL` - CPU plus processor interrupt level (PIL). For example, if CPU 4 is interrupted while at PIL 6, this will be reported as cpu[4]+6.
`Caller` - Address of the caller; displayed symbolically if possible.

Look for high indv values accompanied by slow or fast locking times.

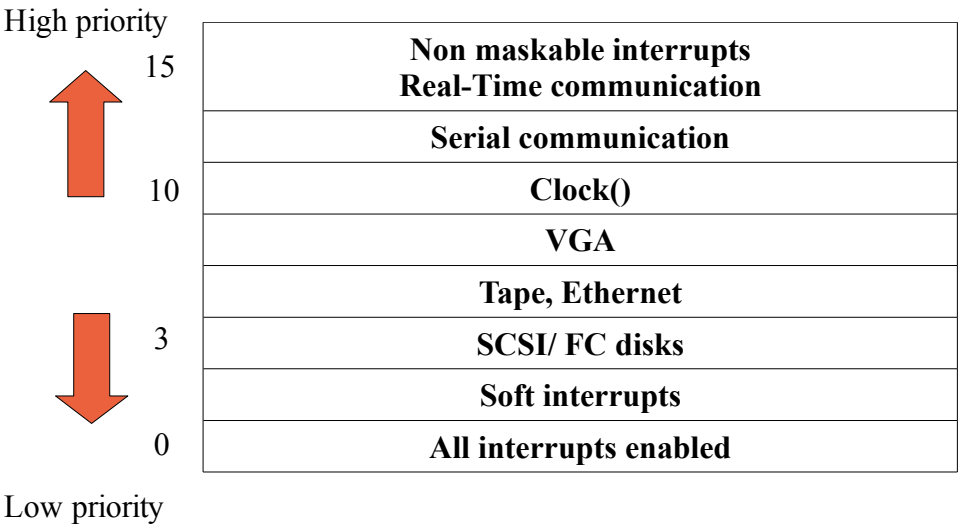
The lockstat command reflects kernel locks and from Solaris 10 is using Dtrace.
The plockstat command reflects user programs locks and is also using Dtrace.

```
# cat adaptive_mutex.d
#! /usr/sbin/dtrace -qs
lockstat:::adaptive-block
{
printf("%4s%4s%9s%9s%20s\t%s\n",
"CPU", "TID", "PID", "UID", "Wait time",
"Command" );
printf("%4d%4d%9d%9d%20d\t%s\n",
curcpu->cpu_id,
curlwpsinfo->pr_lwpid,
curpsinfo->pr_pid,
curpsinfo->pr_uid,
arg1,
curpsinfo->pr_psargs );
stack();
}
```

you may leave a window with this script running and then run find several times to see locks.

Interrupts

An interrupt is a signal requiring attention from the OS.
Hardware interrupts are issued by devices.
Software interrupts are issued by real time threads or pseudo kernel mechanisms (like cross calls).



Interrupts with no buffering capabilities will have higher priority than buffered ones.
Intense interrupts activity may lead to severe effects on system performance.

CPU Scheduling

Objectives

- CPU performance statistics
- CPU Scheduling
- Display and change scheduling parameters
- Change scheduling behavior of a process
- Describe scheduling framework

CPU performance statistics

vmstat 3

CPU-related fields in this output are described below:
`Kthr` - Report the number of kernel threads in each of the three following states:

- r — the number of kernel threads in run queue
- b — the number of blocked kernel threads that are waiting for resources I/O, paging, and so forth
- w — the number of swapped out light-weight processes (LWPs) that are waiting for processing resources to finish.

`faults:sy` – System calls made
`cpu:cs` – CPU context switches made
`cpu:us` – Percentage of time spent in user mode
`cpu:sy` – Percentage of time spent in system mode
`cpu:id` – Percentage of time spent idle

sar -q 3 4 – shows run-queue size and avg. queuing time

`runq-sz, %runocc` – Run queue of kernel threads size in memory and runnable precentage.

sar -w 3 3 – reports swapping acitivities

swpin/s, bswin/s - swap-in operations/blocks transferred (block=512 bytes) respectively.
`swpot/s, bswot/s -` swap-out operations/blocks transferred respectively.
`pswch/s -` process switches per second

Controlling CPU's

psrinfo – shows per-processor information.

psradm – allows offline/online and non-intr assignment on processors.

psrset – manages processors sets.

mpstat – SMP statistics.

mpstat command

mpstat 3 3

`CPU` – The CPU ID on the system
`minf` – number of minor faults
`mjf` – number of major faults
`xcal` – number of inter-processor cross-calls
`intr` – number of interrupts
`ithr` – number of interrupts handled as threads
`csw` – number of context switches
`icsw` – number of involuntary context switches
`migr` – number of thread migrations to other processors
`smtx` – number of times locks are not acquired at the first
attempt on mutexes
`srw` – number of times locks are not acquired at the first attempt on reader and writer locks
`syscl` – number of system calls
`usr` – percentage of user time of the CPU
`sys` – percentage of system time of the CPU
`wt` – percentage of CPU wait time (deprecated in Solaris 10)
`idl` – percentage of idle time of the CPU

CPU Scheduling

Solaris consists of Scheduling policies.
dispadmin -l - will show all available active policies.

TS – Time sharing scheduling class, this is the default class (Unix time sharing).
IA – Interactive scheduling class, this scheduling class raises priority of the active window in a Window management environment. (Runs on Gnome/JDS and CDE).
SYS – System scheduling class, used only for system service threads like the page daemon.
FX – Fixed Priority class, no changes is made to the process priority.
RT – RealTime priority, for application that wants to have the highest priority in the system (Soft realtime), except for higher level interrupts.
FSS – Fair Share scheduling, shares CPU resources among projects in Solaris 8 and up.

TS and IA classes characteristics

Time slice – processes with same priority share cpu time in rotation.
Mean time to wait scheduling – if a process didn't finish its quantum it gets higher priority if it consumed all it's time- it gets lower priority.

Means - I/O bound processes tend to raise priority and CPU bound processes tend to lower their priority.

SYS class characteristics

no time slicing.
Fixed priority assigned at creation.

RealTime class characteristics

Time sliced
Fixed priority

TS/IA dispatch parameter table

`dispadmin -c TS -g -r 1000` `(-g for showing dispatch parameter)`

If process didn't finish all it's alloted time (behaves good). It gets priority as described by – `ts_slpret`
If process finished all of its alloted time (behaves bad). It gets priority as described by - `ts_tqexp`

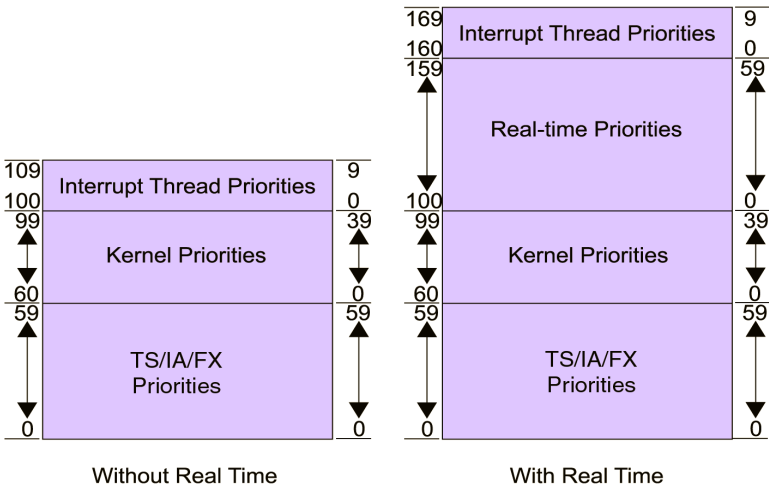
`ts_quantum` – The time allotted to the process to run.

`ts_maxwait` – The maximum number of seconds a process should wait on the queue before being serviced. If exceeded, this prompts the scheduler to raise the priority of the process. The default `ts_maxwait` time is zero and is checked once per second.
`ts_lwait` – The priority level a process changes to if it waits longer than `ts_maxwait` seconds.
`PRIORITY LEVEL` – Indicates the priority level for these parameters.

Changing default scheduling class
dispadmin -d - shows default scheduling class currently implemented on the system.
/etc/dispadmin.conf – scheduling configuration file. (DEFAULT_SCHEDULER=TS)
This file is only created when a default scheduling class is first set.

`priocntl -l` - Listing currently configured scheduling classes.

`priocntl -d -i pid $$` - display current scheduling setting on a process.
`priocntl -e -c RT <cmd>` - runs a new process with different scheduling class



Recent blogs:

Sunguard BRASS
http://www.sun.com/third-party/global/sungard/BRASS_Success_Story.pdf or
http://www.docstoc.com/docs/21110765/SunGard-BRASS---Speeding-transaction-throughput-using-powerful

Using FX scheduling class and processor sets in T-Servers to improve system performance
http://blogs.sun.com/BestPerf/entry/using_solaris_resource_management_uitlities

On the performance impact of FSS (Solaris, HP-UX, AIX)
http://www.cs.umb.edu/~eb/goalmode/cmg2000final.htm

Changing TS examples:
http://docs.sun.com/app/docs/doc/816-0219/6m6njqbcn?a=view

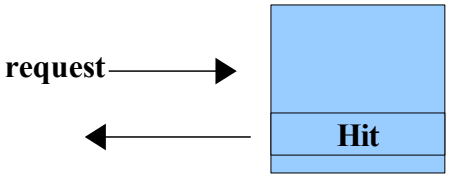
System Caches

Objectives

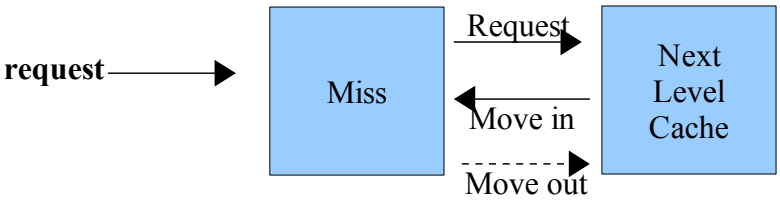
- Cache operations
- Cache faults
- Characteristics of a cache
- Cache problems in SMP environment
- Cache problems associated with cache design

Cache operations

Cache hit -



Cache miss -



Cache faults

Major fault – page in the memory is missing (mjf - vmstat)

Minor fault – validating cache entry with memory (minf - vmstat)

Micro fault – TLB translation is missing (**trapstat -t 5 10**, instruction – itlb, data - dtlb)

Factors affecting cache hit rate

Cache size – overall cache number of entries or blocks.

Cache line or block size – size of each entry or block in the cache.

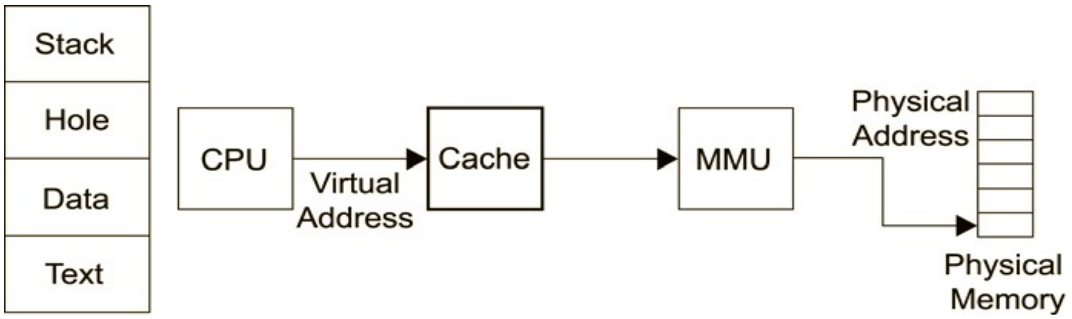
Locality of data – data does not differ from each other so often, leading to thrashing of an entry.

Cache architecture – how fast access and search in the cache is implemented.

Virtual or physical address caching

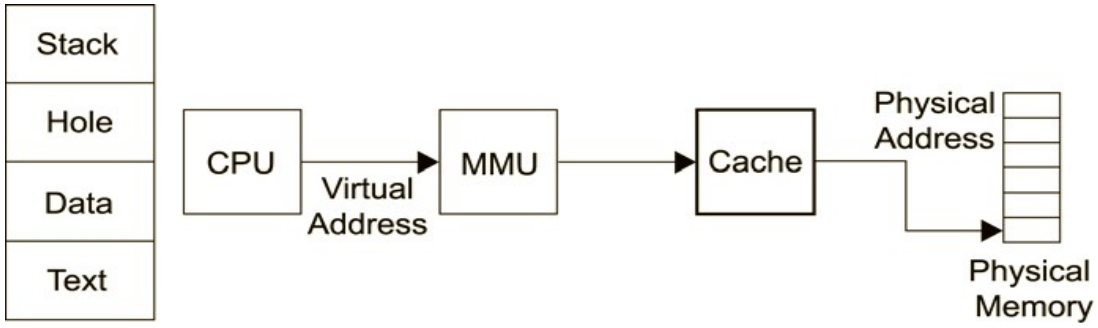
Virtual address caching – not reusable among processes (L1 – for speed)

Boost cpu if there is cache hit since mmu does not need to translate to virtual address.



Physical address caching – reusable among processes (L2 - cost effective)

even if there is a cache hit the mmu should translate the virtual address to the corresponding physical address.



Direct Mapped and Set-Associative caches

Direct mapped	**Set-Associative**
Each entry has only one mapping of data.	Each entry has *n* possible locations, the cache is called an *n*-way set associative cache.
Easy to implement.	More complex to implement.
Data stored at some location forces flushing another entry that is indexed to the same location.	Data can be stored at an entry without flushing that entry. Other addresses that map to the same cache line can therefore remain in the cache.

Harvard Architecture (L1 Cache)

A cache developed in 1940 by IBM and used by harvard university.

Implement separation of Instructions (I$) and data (D$) caches.

On USIIIcu CPU -

I$ - 32KB 4 way set associative

D$ - 64KB 4 way set associative

Prefetch - 2KB 4 way set associative

Write - 2KB 4 way set associative

In L2 cache there is no distinct between Instruction and Data entries.

Write through and write back caches

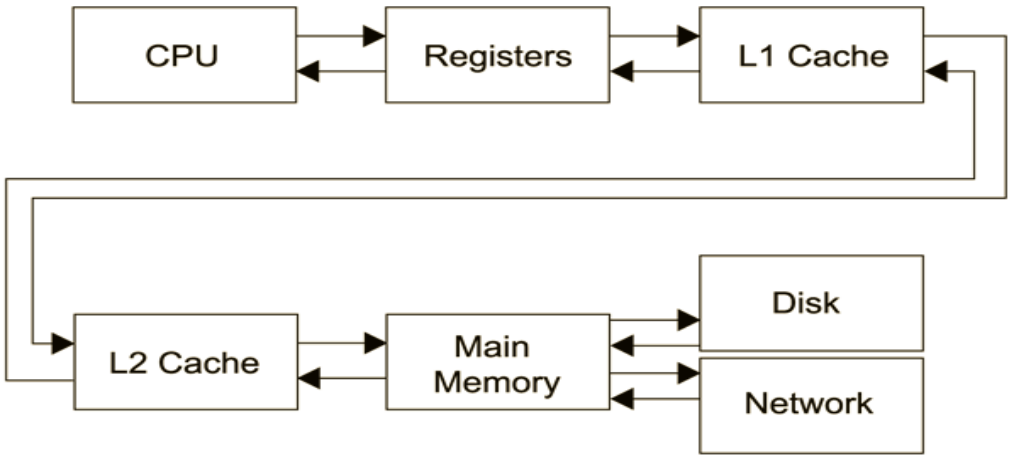
Another distinction in cache types derives from the manner in which changes made to a cache are copied to the slower storage:

• Write-back cache – Flushes data to slower storage asynchronously, without the need to report on write completion.

• Write-through cache – Updates slower storage whenever the cache contents change synchronously, with a need to report on write completion..

Write-back caches collect write operations for later delivery to main memory, This allows implementing optimization or the write operations. However, if the system crashes before the data is copied to lower level, all the data in the cache is lost – leading to a data corruption.

This is why write back cache should reside on a non-volatile device !!!



Memory Management

Objectives

- Terms and definitions
- Kernel address space
- Process address space
- HAT
- MPSS
- The Page scanner
- The Memory Scheduler

Terms and definitions

Physical memory – The RAM

Total physical memory - **prtconf | grep ^Mem**

Virtual memory – The RAM + Swap devices.

64bit address space gives 16 Exabyte address space.

Page – a small fixed size unit where the kernel stores data. Nowadays CPU's implement several page sizes.

Page in – is the process of taking in a page from the disk to RAM.

Page out – is the process of taking out a page from RAM to disk.

Swap in – is the process of transferring of a page from swap to RAM.

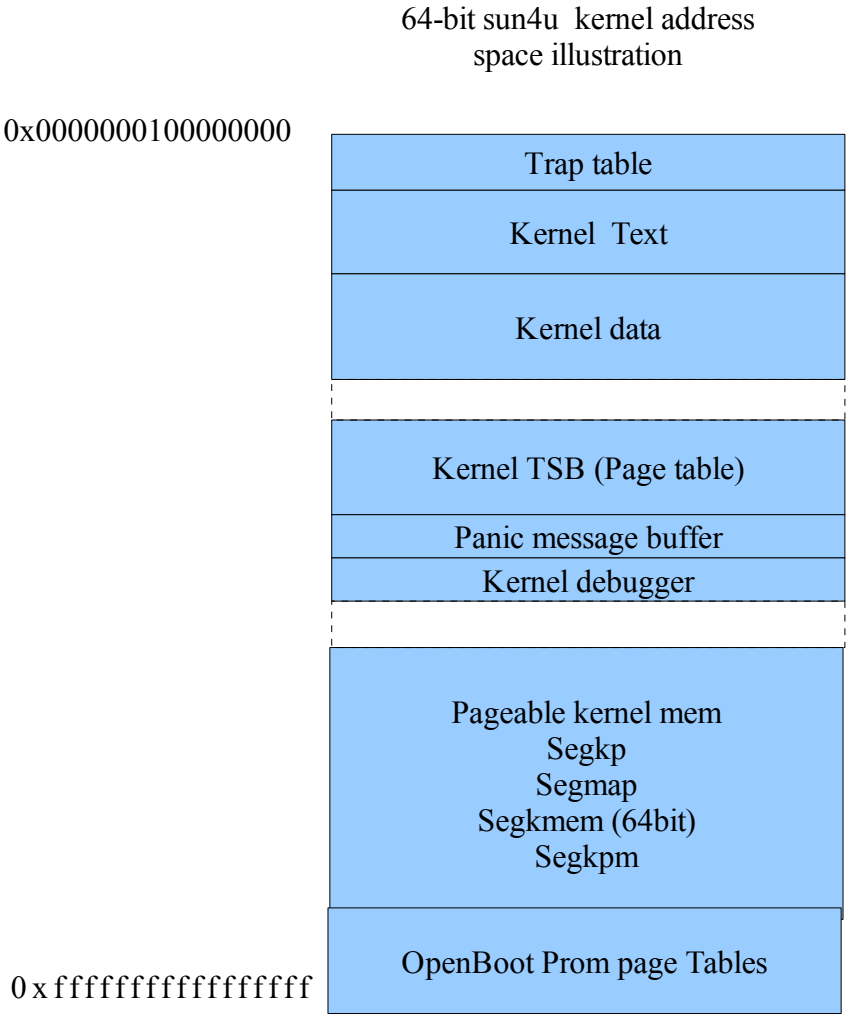
Swap out – is the process of transferring of a page from RAM to swap.

Anonymous pages – all pages that are not backed out by backing store are copied to the swap area.

Anonymous page out – the process of taking out pages from RAM. (Actually there is no need to copy the pages unless they are not synchronized by their backing store.

Kernel address space

Kernel virtual memory layout differs from platform to platform, mostly based on the platform's MMU architecture.



segvn – Process memory segment

segkp – pageable kernel memory which holds all segvn and process memory (malloc)

segmap – File-system cache memory segment

Segkmem – kernel memory segment (kernel heap)

Segkpm - Physical page mapping

The Cache list

Free list – is the list of all free pages.

Cache list – is the list of pages used by file-system caches.

Vmstat:memory:free = freelist+cachelist

The cache list is linked to the end of the free list. If the free list is exhausted, pages will be taken from the head of the cache list.

by using mdb ::memstat dcmd.

```
[root@n1: /] #mdb -kw
Loading modules: [ unix genunix dtrace specfs ufs pcisch ssd fcp fctl qlc ip hoo
     k neti sctp arp usba s1394 nca md lofs zfs sd cpc audiosup random crypto wrsmd f
          cip logindmux ptm sppp nfs ipc ]
> ::memstat
Page Summary            Pages             MB  %Tot
------------     ----------------  ----------------  ----
Kernel                  36730             286   14%
Anon                    28235             220   11%
Exec and libs            5509              43    2%
Page cache              18705             146    7%
Free (cachelist)        40442             315   16%
Free (freelist)        127248             994   50%

Total                  256869            2006
Physical               255291            1994
>
```

Viewing physical memory (physmem)

>**physmem/E**

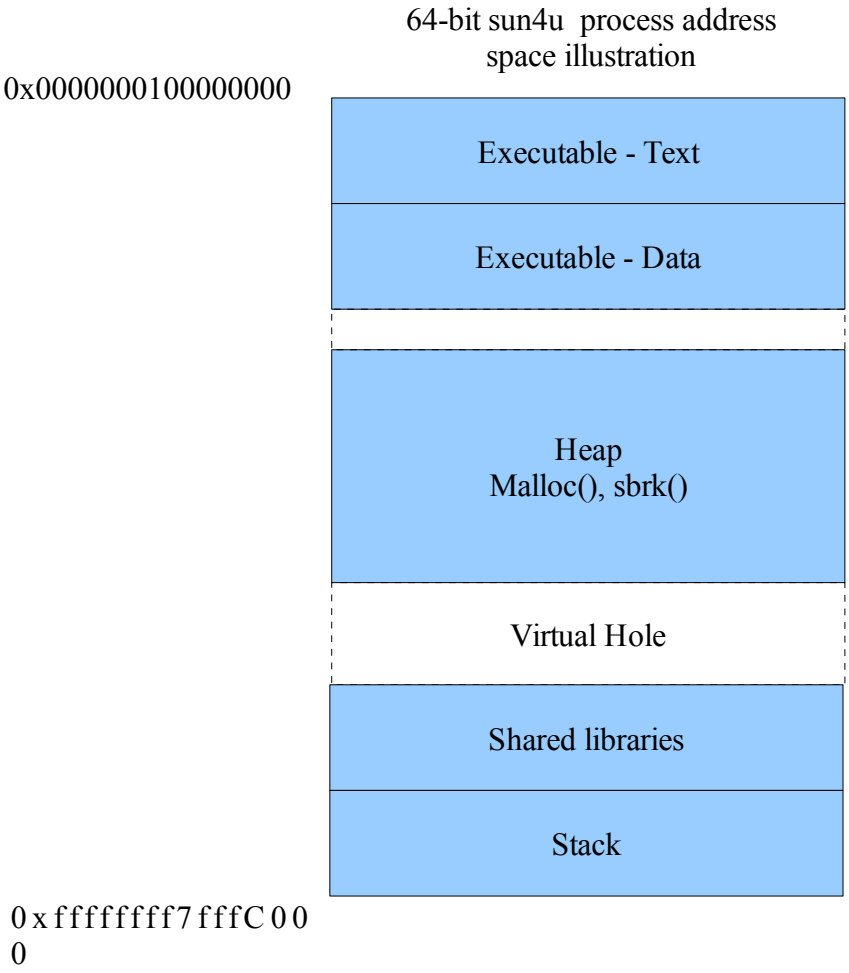
this valeu is in pages, so since each page size on the system is 8KB

We can now do a calculation:

```
^C[root@n1: /] #mdb -kw
> physmem/E
physmem:
physmem:       256869
>
[root@n1: /] #bc
256869*8/1024
2006

2006MB RAM (including rounding)
```

Process address space



Text – stores the binary instruction code comes from the executable program file.

Data – stores all global data, constants and static variables retrieved from the program file.

Heap – stores all process data that is created at run time.

Shared libraries – stores all relevant shared libraries. This area is position independent so all process will be able to use the same libraries without the need to store unique a copy.

Stack – holds the current state of process. Stack size for the process is 8MB and for each extra thread 2MB (on 64bit systems) or 1MB (on 32 bit systems)

Process address space is a virtual address space starting from 0x0000000100000000 and ending with 0xffffffff7fffC000. When a process is being served by the CPU, the CPU must use physical addresses, so there is a need to translate virtual address space to physical address space.

This translation is done by the MMU.

Hardware Address Translation (HAT)

In order to ease on translations, Translation Lookaside Buffer (TLB) and Translation Storage Buffer (TSB) mechanisms were introduced.

TLB - is basically a virtual to physical translation cache implemented in hardware.

TSB - is basically a virtual to physical translation cache implemented in software.(The TSB is a software structure used as a translation entry cache to allow the TLB to be quickly filled; it is discussed in detail in the UltraSPARC II User's Manual.) the TSB is widley known as the page table.

ISM/DISM – Dynamic Intimate Shared Memory

Intimate shared memory – is a mechanism that allows grouping several contiguous pages into big segments of pages(called superpages) for the use of designated applications (ISM was mainly designed for oracle database). This grouping of contiguous base pages in ISM was to a fix size determined at boot, leaving no dynamic resize capabilities. The introduction of DISM made resizing of ISM chunks possible without rebooting.

MPSS (Multiple Page Size Support)

Multiple page size support is a mechanism introduced in Solaris 9.

MPSS enables the use of larger pages, eliminating the need for:

1. Large page tables
2. Large amount of translations in MMU
3. Less misses in TLB.

For Sparc CPU there are:

UltraSparc II-III: 4 page sizes – 8KB, 64KB, 512KB, 4MB

UltraSparc IV and up: 6 page sizes - 8KB, 64KB, 512KB, 4MB, 32MB, 256MB

UltraSparc T1/2: 4 page sizes – 8KB, 64KB, 32MB, 256MB.

For AMD/Intel CPU's there 2 page sizes which are – 4KB, 2MB

You may implement MPSS in 3 ways:

1. At compilation time – by adding relevant compiler specific directives.
2. As Shell environment before running the program – LD_PRELOAD=$LDPRELOAD:mpss.so.1; MPSSHEAP=512K
3. As a command for implementing mpss hints on running processes -

   ppgsz -o heap=512k -p 624

The trapstat utility

Trapstat utility is a utility that gather and displays run-time trap statistics on UltraSPARC-based systems.

-t option enables TLB statistics.

-T option enables TLB statistics with page size information.

TLB Trap statistics are divided to user (u) and kernel (k) mode TLB events.

Trapstat -t example:

```
[root@n1: /] #trapstat -t
cpu m| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
-----+-------------------------------+-------------------------------+----
  0 u|       333  0.0         8  0.0 |       511  0.0         4  0.0 | 0.0
  0 k|        79  0.0         0  0.0 |       336  0.0         1  0.0 | 0.0
-----+-------------------------------+-------------------------------+----
  1 u|       200  0.0         0  0.0 |       380  0.0         0  0.0 | 0.0
  1 k|        33  0.0         0  0.0 |       624  0.0         0  0.0 | 0.0
=====+===============================+===============================+====
 ttl |       645  0.0         8  0.0 |      1851  0.0         5  0.0 | 0.0

cpu m| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
-----+-------------------------------+-------------------------------+----
  0 u|       270  0.0         7  0.0 |       490  0.0         1  0.0 | 0.0
  0 k|        46  0.0         0  0.0 |       250  0.0         1  0.0 | 0.0
-----+-------------------------------+-------------------------------+----
  1 u|       211  0.0        11  0.0 |       370  0.0         7  0.0 | 0.0
  1 k|        30  0.0         0  0.0 |       483  0.0         2  0.0 | 0.0
=====+===============================+===============================+====
 ttl |       557  0.0        18  0.0 |      1593  0.0        11  0.0 | 0.0

^C
```

trapstat -T output example:

```
[root@n1: /] #trapstat -T

cpu m size| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
----------+-------------------------------+-------------------------------+----
  0 u   8k|      324  0.0       16  0.0 |      479  0.0        2  0.0 | 0.0
  0 u  64k|        0  0.0        0  0.0 |        0  0.0        0  0.0 | 0.0
  0 u 512k|        0  0.0        0  0.0 |        0  0.0        0  0.0 | 0.0
  0 u   4m|        0  0.0        0  0.0 |        0  0.0        0  0.0 | 0.0
- - - - - + - - - - - - - - - - - - - - - + - - - - - - - - - - - - - - - + - -
  0 k   8k|       94  0.0        0  0.0 |      587  0.0        3  0.0 | 0.0
  0 k  64k|        0  0.0        0  0.0 |        0  0.0        0  0.0 | 0.0
  0 k 512k|        0  0.0        0  0.0 |        0  0.0        0  0.0 | 0.0
  0 k   4m|        0  0.0        0  0.0 |        1  0.0        0  0.0 | 0.0
----------+-------------------------------+-------------------------------+----
  1 u   8k|      353  0.0        1  0.0 |      696  0.0        2  0.0 | 0.0
  1 u  64k|        0  0.0        0  0.0 |        0  0.0        0  0.0 | 0.0
  1 u 512k|        0  0.0        0  0.0 |        0  0.0        0  0.0 | 0.0
  1 u   4m|        0  0.0        0  0.0 |        8  0.0        0  0.0 | 0.0
- - - - - + - - - - - - - - - - - - - - - + - - - - - - - - - - - - - - - + - -
  1 k   8k|       59  0.0        0  0.0 |      315  0.0        1  0.0 | 0.0
  1 k  64k|        0  0.0        0  0.0 |        0  0.0        0  0.0 | 0.0
  1 k 512k|        0  0.0        0  0.0 |        0  0.0        0  0.0 | 0.0
  1 k   4m|        0  0.0        0  0.0 |        2  0.0        0  0.0 | 0.0
==========+===============================+===============================+====
     ttl |      830  0.0       17  0.0 |     2088  0.0        8  0.0 | 0.0

^C
```

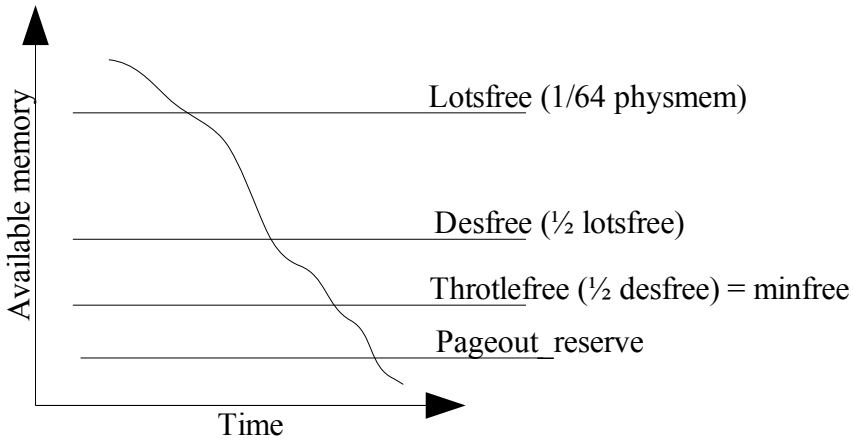
Determining when to use Large pages

In order to determine whether we can improve application performance by using a larger page size, we need to determine the amount of the time the CPU spends servicing TLB misses on behalf of a target application. If a TLB miss occurs in a user-mode application, it will be counted as user time.

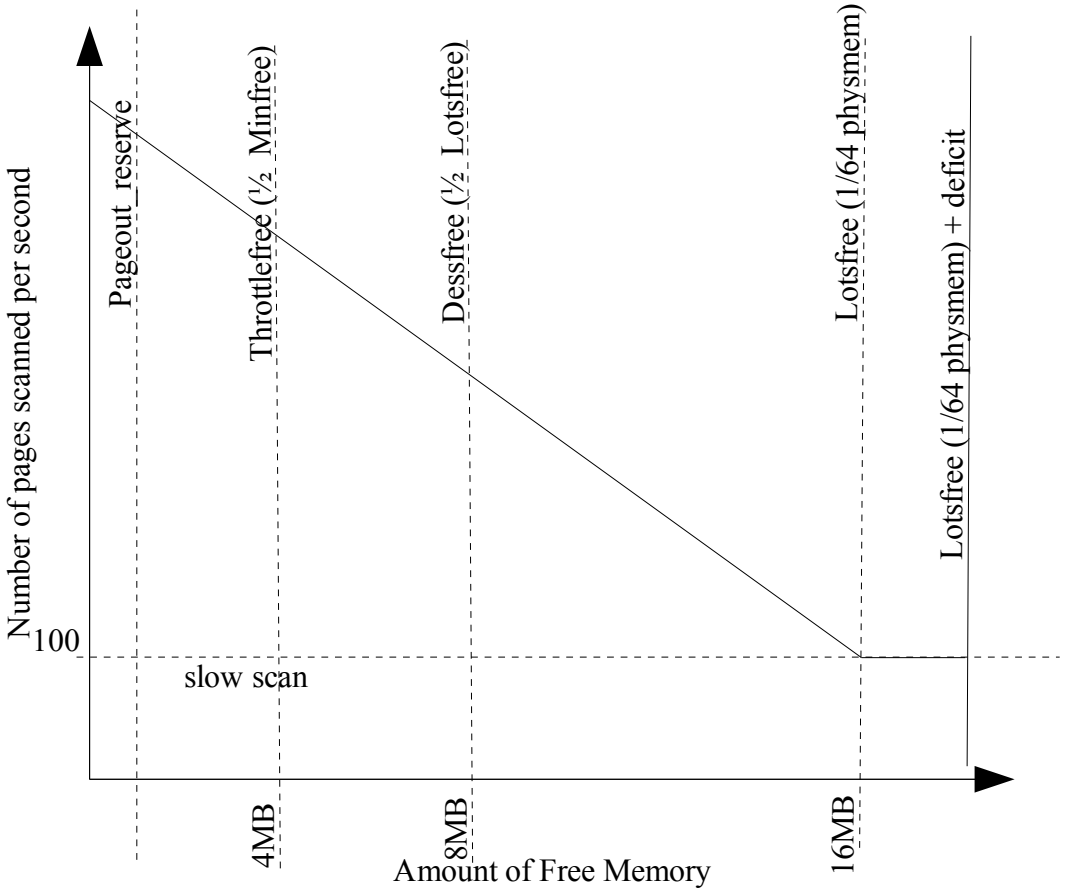
Paging and Swapping

Paging is done by the page daemon

Swapping is done by the swapper



Page Scanner Rate, Interpolated by Number of Free Pages
(1GB example)



Parameter	Description	Min	Default
Lotsfree	The scanner starts stealing anonymous memory pages when free memory falls bellow lotsfree	512KB	1/64 of memory
Desfree	If free memory falls below desfree for more than 30 second, then the page-out scanner is started 100 times/second	Minfree	Lotsfree/2
Minfree	If free memory falls below minfree, the the page scanner is signaled to start every time a new page is created		Desfree/2
Throttlefree	The number at which point the page_create routines make the caller wait until free pages are available		Minfree
Slowscan	The rate of pages scanned per second when free memory-lotsfree		100
Maxpgio	A throttle for the maximum number of pages per second that the swap device can handle	40/60/90 based on architecture	40 (UltraSPARC III)

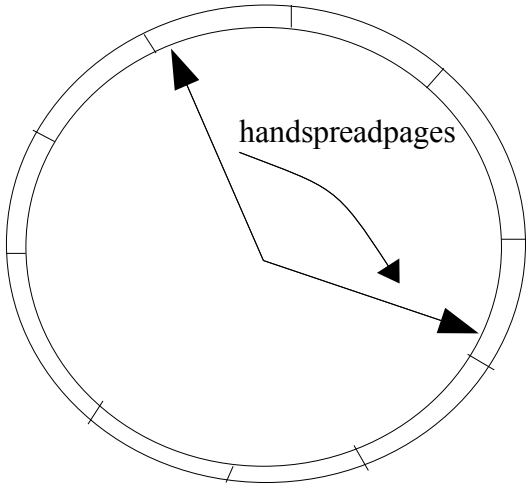
The Page scanner

The page scanner is implemented as two kernel threads, both uses process number 2 – the "pageout". The scanner starts scanning when free memory is lower than lotsfree+deficit. It start to scan at a rate determined by the slowscan parameter (usually 100) up to fastscan (dynamically calculated).

The page scanner tracks page usage by reading a per-page hardware bit from the MMU for each page. Tow bits are kept for each page reference bit and dirty bit.

Reference bit – the page was referenced since the last clear.

Dirty bit – the page was modified since the last clear.



The two hands, the front and back hands, rotate clockwise in page order around the list. The front hand rotates ahead of the back hand, clearing the referenced and modified bits for each page. The trailing back hand then inspects the referenced and modified bits some time later. Pages that haven't been referenced or modified are swapped out and freed. The rate at which the hands rotate around the page list is controlled by the amount of free memory in the system, and the gap between the hands is fixed by a dynamically calculated value, handspreadpages.

The Memory Scheduler

In addition to the page-out process, the CPU Scheduler can swap out entire processes to conserve memory.

Soft swapping

soft swapping takes place when the 30 second average for free memory is below desfree. Then, the memory scheduler looks for processes that have been inactive for at least maxslp seconds (default is 20 seconds). When it finds such a process it swaps out the thread structures for each thread, then pages out all of the private pages of memory for that process.

Hard swapping

Hard swapping will take place when all of the following are true:

- More than two processes are on the run queue. Waiting for CPU.
- The average free memory over 30 seconds is consistently less than desfree.
- Excessive paging is going on. (page-out + page-in > maxpgio)

System Buses

Objectives

- Buses overview
- prtdiag utility
- Memory interleaving

Buses overview

System buses

A bus that connects several system boards (also known as mother boards) in case multiple system boards exist. This bus also connects all other buses together.

Some known system buses are:

UPA – Ultra Port Architecture bus (UPA 1996)

GigaPlane bus - allows hot pluggable UPA buses and found on E4000/5x00/6x00.

Gigaplane XB bus - found on SF3800/4800/4900/6800/6900

SunFire plane bus – found on SF3800/4800/4900/6800/6900 SF12K/15K E20/25K.

Peripheral buses

A bus that connects peripherals to the system.

Some known peripheral bus are:

PCI – Peripheral Component Interconnect, allows 33/66 MHZ 32/64 bit slots.

XPCI – X-PCI 133MHZ, 64 bit. Uses a hub topology.

ePCI - ePCI uses a switch topology.

SCSI – Small Computer Serial Interface.

SATA – Serial ATA.

Guidelines for PCI cards placement.

On e/X/PCI based systems it is recommended to spread PCI cards especially Fibre-Channel HBA's and Giga-bit ethernet cards across several PCI controllers.

For example:

spread pci cards across the following controllers -

/pci@8,700000/

/pci@8,600000/

You may view PCI controller information in Sun system handbook found on sunsolve.sun.com .

Prtdiag utility

The prtdiag utility displays hardware configuration of a system.

Especially -

CPU's – E-cache (E$) size, CPU Speed, CPU implementation, status, Location

IO Devices – Bus Type, card Frequency, Slot+Status, Name + Path, Model

Memory configuration – Segment Table, Bank Table, Memory module groups

Other information (mainly shown along with -v option) - USB devices, Environmental status, HW Revisions, System PROM information

Output example:

```
[root@n1: /] #prtdiag
System Configuration: Sun Microsystems  sun4u SUNW,Sun-Blade-1000 (2 X UltraSPARC-III+)
System clock frequency: 150 MHZ
Memory size: 2GB

========================================= CPUs ========================================
           E$         CPU                 CPU
CPU  Freq     Size       Implementation        Mask    Status     Location
---  --------  ----------  ---------------------  -----   ------     --------
0    900 MHz  8MB        SUNW,UltraSPARC-III+    2.2    on-line    +-board/cpu0
1    900 MHz  8MB        SUNW,UltraSPARC-III+    2.2    on-line    +-board/cpu1

====================================== IO Devices ======================================
Bus     Freq  Slot +      Name +
Type    MHz   Status      Path                        Model
------  ----  ----------  --------------------------  -------------------
pci     33    +s/system-board  ebus/ns87317-ecpp (parallel)
            okay       /pci@8,700000/ebus@5/parallel@1,300278

pci     33    +s/system-board  ebus/se (serial)
            okay       /pci@8,700000/ebus@5/serial@1,400000

pci     33    +s/system-board  pci108e,1101 (network)       SUNW,pci-eri
            okay       /pci@8,700000/network@5,1

pci     33    +s/system-board  pciclass,0c0010 (firewire)
            okay       /pci@8,700000/firewire@5,2

pci     33    +s/system-board  scsi-pci1000,f (scsi-2)
            okay       /pci@8,700000/scsi@6
```

```
pci    33   +s/system-board  scsi-pci1000,f (scsi-2)
          okay       /pci@8,700000/scsi@6,1

pci    33   +s/system-board  pci108e,1000
          okay       /pci@8,700000/pci@3/pci108e,1000

pci    33   +s/system-board  SUNW,qfe (network)          SUNW,pci-qfe
          okay       /pci@8,700000/pci@3/SUNW,qfe@0,1

pci    33   +s/system-board  pci108e,1000
          okay       /pci@8,700000/pci@3/pci108e,1000

pci    33   +s/system-board  SUNW,qfe (network)          SUNW,pci-qfe
          okay       /pci@8,700000/pci@3/SUNW,qfe@1,1

pci    33   +s/system-board  pci108e,1000
          okay       /pci@8,700000/pci@3/pci108e,1000

pci    33   +s/system-board  SUNW,qfe (network)          SUNW,pci-qfe
          okay       /pci@8,700000/pci@3/SUNW,qfe@2,1

pci    33   +s/system-board  pci108e,1000
          okay       /pci@8,700000/pci@3/pci108e,1000

pci    33   +s/system-board  SUNW,qfe (network)          SUNW,pci-qfe
          okay       /pci@8,700000/pci@3/SUNW,qfe@3,1

pci    33   +s/system-board  SUNW,qlc-pci1077,2200 (scsi-+
          okay       /pci@8,600000/SUNW,qlc@4

pci    33   +s/system-board  SUNW,ifp-pci1077,2100 (fcal)
          okay       /pci@8,600000/SUNW,ifp


============================== Memory Configuration ==============================
Segment Table:
------------------------------------------------------------------------
Base Address      Size      Interleave Factor  Contains
------------------------------------------------------------------------
0x0               2GB               2         BankIDs 0,2

Bank Table:
------------------------------------------------------------
        Physical Location
```

```
ID      ControllerID  GroupID   Size      Interleave Way
-----------------------------------------------------------
0       0             0         1GB       0
2       0             0         1GB       1

Memory Module Groups:
-------------------------------------------------------
ControllerID   GroupID  Labels        Status
-------------------------------------------------------
0              0        chassis/system-board/J0100
0              0        chassis/system-board/J0202
0              0        chassis/system-board/J0304
0              0        chassis/system-board/J0406

================================ usb Devices ================================

Name         Port#
------------ -----
hub          4

================================ hub#4 Devices ================================

Name         Port#
------------ -----
device       1

============================== device#1 Devices ==============================

Name         Port#
------------ -----
keyboard
```

mouse

Memory interleaving

You may adjust memory interleaving factor by obp parameter interleave-factor or by System controller commands in Mid-range/High End systems.

Advantages

Data is layed across more several banks giving data transfer rate to and from memory.

Disadvantages

If interleaving is anabled across system boards no Dynamic reconfiguration operation may occur

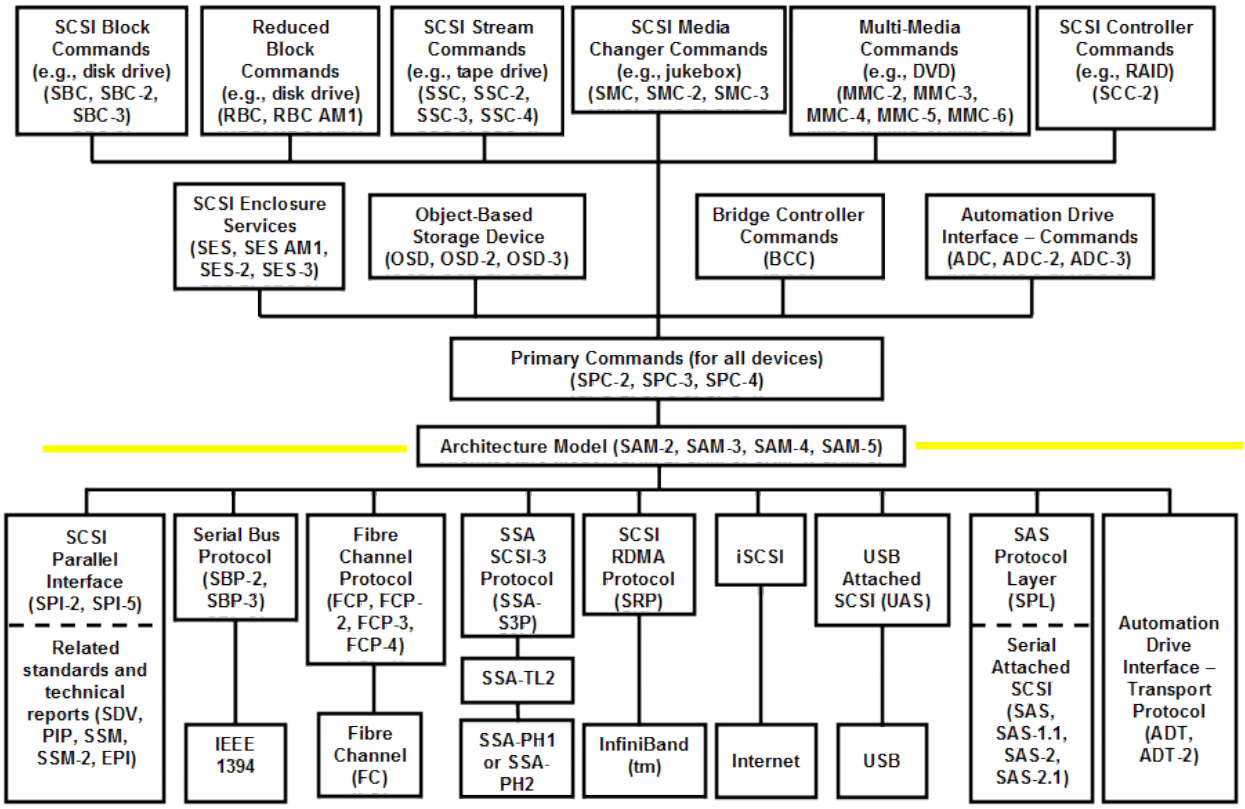
(e.g – hot plugging of a system board or move of resources from one domain to another.)

I/O Tuning

Objectives

- SCSI standards architecture
- Basic Terms and definitions
- Reducing IOPS
- Monitoring I/O
- I/O Tuning

SCSI Standards Architecture



SCSI evolved during many years from dedicated external device interface into versatile suite of protocols implemented universally almost everywhere.

Modern SCSI protocol is described by SAM (SCSI Architecture Model). SAM describes how connect together rich family of different command sets and broad range of different low-level transport protocols.

Examples are:

- Command sets:
    - SPC (SCSI Primary Commands)
    - SBC (SCSI Block Commands)
    - SST (SCSI Stream Commands)

- Transport Protocols:
    - FCP (Fibre Channel Protocol)
    - iSCSI (SCSI over TCP/IP)
    - SAS (Serial Attached SCSI)
    - FCoE (Fibre Channel over Ethernet)

Basic Terms and Definitions

Target (device server) – the entity that executes SCSI commands (i.e. disk, tape, printer, etc)

Initiator (client) – the entity that submits SCSI commands for execution and consume the responses generated by the target.

Port – the medium attachment point that is used to deliver SCSI commands from the Initiator to the Target via a particular medium. The exact definition of the port is specific to each transport protocol and defined in the respective standard.

World Wide Name (WWN) – the identifier assigned to an entity which describes it in a unique way.

Global Unique Identifier (GUID) – another form of unique identifier.

Logical Unit (LU) - an instance of a device server within the specified target.

Logical Unit Number (LUN) - the number of the LU, used to address the SCSI command to the LU. Term LUN is often used instead of LU (albeit incorrectly).

Host Bus Adapter (HBA) - physical device that implements one or more ports for a given physical medium. Usually used as an initiator in the server/host.

ISP – Intelligent SCSI Processor, is a processor found inside HBA's and implement all SCSI initiator commands.

IOPS – The number of I/O operations per second

Reducing IOPS

The following may reduce IOPS

- Maximizing the blocks of data to the maximum possible.
- Tuning of main memory (check for sufficient memory, use large pages when possible)
- Tune file-system properly (UFS – buffer cache, ZFS -ZIL and hybrid pool architecture)

Monitoring I/O

The iostat -x and sar -d commands both report I/O statistics information.

iostat -x

output example:

```
                 extended device statistics
device     r/s    w/s   kr/s   kw/s wait actv  svc_t  %w  %b
fd0        0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
lofi126    0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
ramdisk1   0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
sd6        0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
ssd0       0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
ssd2       0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
nfs1       0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
```

device name of the diskr/s reads per second

w/s writes per second

kr/s kilobytes read per second
The average I/O size during the interval can be computed from kr/s divided by r/s.

kw/s kilobytes written per second

The average I/O size during the interval can be computed from kw/s divided by w/s.

wait average number of transactions waiting for service (queue length)

This is the number of I/O operations held in the device driver queue waiting for acceptance by the device.

actv average number of transactions actively being serviced (removed from the queue but not yet completed)

This is the number of I/O operations accepted, but not yet serviced, by the device.

svc_t average response time of transactions, in milliseconds

The svc_t output reports the overall response time, rather than the service time, of a device. The overall time includes the time that transac-

%w percent of time there are transactions waiting for service (queue non-empty)

%b percent of time the disk is busy (transactions in progress)

Other useful iostat options:
-n – displays devices in their cXtXdXsX format.
-e – displays device errors.
-z – filters out zero statistics lines.

sar -d

output example:

```
16:37:44   device       %busy   avque   r+w/s  blks/s  avwait  avserv
16:37:48   fd0             0     0.0       0       0     0.0     0.0
           lofi126         0     0.0       0       0     0.0     0.0
           nfs1            0     0.0       0       0     0.0     0.0
           ramdisk1        0     0.0       0       0     0.0     0.0
           sd6             0     0.0       0       0     0.0     0.0
           ssd0            0     0.0       0       0     0.0     0.0
           ssd0,a          0     0.0       0       0     0.0     0.0
           ssd0,c          0     0.0       0       0     0.0     0.0
           ssd2            0     0.0       0       0     0.0     0.0
           ssd2,a          0     0.0       0       0     0.0     0.0
           ssd2,c          0     0.0       0       0     0.0     0.0
```

%busy, avque

portion of time device was busy servicing a transfer request, average number of requests outstanding during that time.

read/s, write/s, blks/s

number of read/write transfers from or to device, number of bytes transferred in 512-byte units.

avwait

average wait time in milliseconds.

avserv

average service time in milliseconds.

Network Performance

Objectives

- Understanding Network Performance
- TCP tuning options
- TCP tunable parameters

Understanding network performance

Network protocols

A network protocol is similar to human language. Except that the entities exchanging the messages and taking actions are hardware or software components of some device.

> *A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event.*

In order to provide structure to the design of network protocols, network designers organize protocols- and the network hardware and software that implement the protocols – in layers.

TCP/IP model

Application layer
Transport layer
Network layer
Link layer
Physical layer

Protocol layering - Each layer uses lower layer and services upper layer.

Advantages:
Break networking problems to modular layers.(dealing with complex systems)
Modularity makes it easier to update system components.
Eases maintenance – change to implementation of a layer: no impact on other layers.
Changes to interface service of a layer – impact only the upper level.

Disadvantages:
Overhead of each layer implementation.
A drawback of a layer – that it may duplicate lower layer functionality(e.g – error recovery is implemented both in link basis and end-to-end basis)
Second drawback – functionality in one layer may need information that is present in another layer, by thus violating the goal of separation by layers.

When taken together the protocols of the various layers are called the protocol stack.
The Internet Protocol stack consists of 5 layers: physical, link, network, transport and application layers.

Application layer

Contain the application protocols. Like HTTP, FTP, DNS SMTP etc...
This layer creates a message or a stream, which consists the user data.

Transport layer

Contain protocols to transport application-layer messages between the client and server sides of an application. This layer contains especially TCP and UDP.
This layer creates a Transport header which contains: port source and port destination.

Network layer

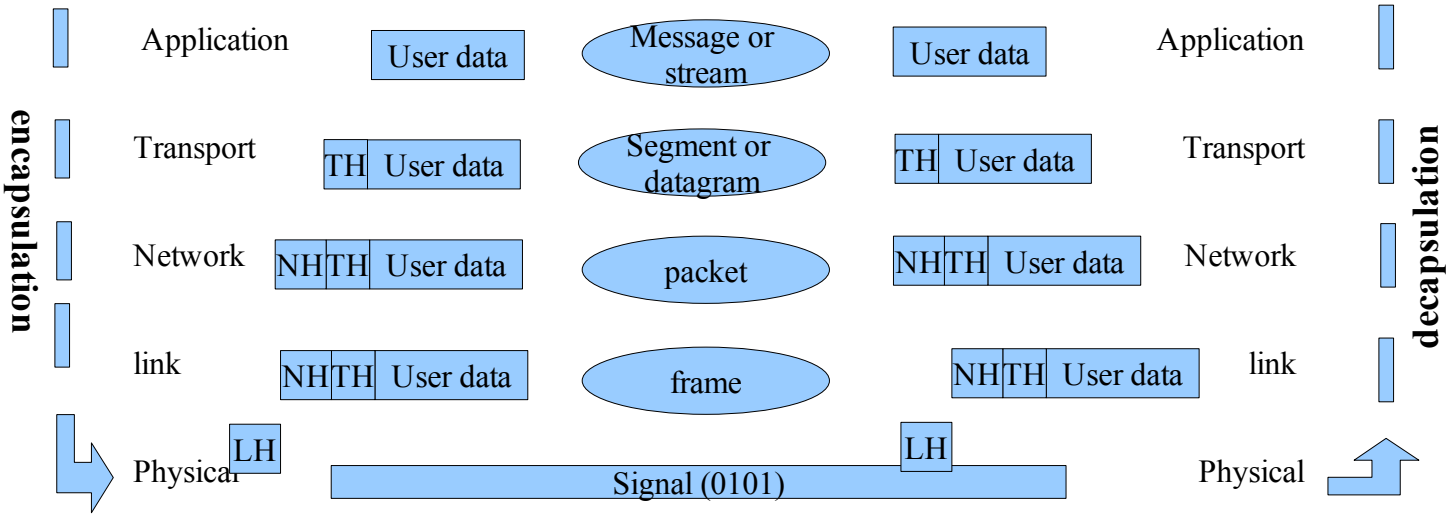
The network layer is responsible for routing packets known as datagrams from source host to a destination host. The protocol that mostly implemented in the Internet is the Internet Protocol(IP).
The network layer also contains routing protocols like: RIP, OSPF, BGP etc...
This layer creates a Network header which contains source IP and destination IP address.

Link layer

Responsible for delivering frames between one node to the next node in a route.
The protocols that are most widely implemented are: Ethernet and PPP(Point-to-Point Protocol).
This layer creates Link header, which contains source MAC address and destination MAC address.

Physical layer

Responsible for moving individual bits within the frame from one node to the next.
The protocols that are most widely implemented are: twisted-pair copper wire, coaxial cable, fiber-optic etc...



Introduction to Ethernet

Ethernet Characteristics

- Ethernet frame structure
- Ethernet Multiple Access protocol

Ethernet frame structure

Preamble	Destination address	Source address	Type	DATA	CRC

Preamble – the Ethernet frame begins with a 8 byte(each of the first 7 bytes has a value of 10101010, the 8th byte is 10101011 preamble field which is responsible for waking up the receiver adapter(NIC) and to synchronize their clocks, the last two 1s indicate the NIC that an "important stuff" is about to come. (this is because there is always a drift from the actual bit rate that the link can provide.)

Dest address – Destination MAC address

Source address – Sender MAC address

Type – IP, ARP, IPX(Novel), or AppleTalk

CRC – Cyclic Redundant Check

Minimum frame size is 96 bits, in order to detect jam signal

Ethernet Multiple Access protocol

Ethernet Access protocol is called CSMA/CD – Carrier Sense Multiple Access / Collision detection.

CSMA/CD characteristics:

1. An adapter may begin to transmit at any time(No slots are used)

2. Carrier Sense - No talking when somebody else talks.

3. Collision Detection – in case of Collision we stop to talk.

4. Before attempting a retransmission the adapter waits a random time(exponential backoff).

Adapter operation:

1. Adapter obtains network datagram and prepare Ethernet frame and put the frame in the adapter buffer.

2. If there is no signal for about 96 bit time long it starts to transmit the frame.

3. While transmitting the adapter sense if other adapters send any signal.

4. If there is other signal than it's own signal – it stops to transmit and send a 48 bit jam signal.

5. After aborting (that is, sending the jam signal) the adapter enters an exponential backoff phase.

Ethernet devices

Interface/adapter – A boundary between the host and the physical link.

Each interface/adapter has a unique MAC address.

Mac address is a 48 bit address, which distinguish each interface in our LAN.

Hub – A device that acts on individual bits rather than frames.

When a bit arrives from 1 interface it re-create the same bit on all other interfaces that it has.

A device connected to a hub has the same collision domain with all other devices connected to this hub.

Switch – A device that operates on Ethernet frames (Layer 2).

Switch will filter and forward frames based on LAN(MAC) destination addresses.

This is done by a switch table saved in the switch(usually done by hardware).

Switch will fill the switch table using a self-learning mechanism.

Unmanaged switches – are switches with no more configuration then a primitive switch.

Managed switches - are switches which you can operate manually by logging in and adding more functionality to them.

Router – A device that operates on IP packets(Layer 3).

Router will filter and forward packets based on a routing table it contains.

Firewall- A network security device that operates on Ethernet frames(Layer 2) and IP Packets(layer 3).

A basic Firewall is responsible for 3 things:

Monitor, Allow/Deny and Log Network traffic based on a rules table it contains.

Proxy – a network device which provides and caches access to the internet in an environment which is not approved to access out of its LAN boundary or in an environment which lacks network bandwidths.

Switches vs. Routers

Switches – use cut-through, **Routers** – use store-and-forward

Switches – can not avoid loops. **Routers** – can avoid loops by using forwarding table.

Network tuning options

1. Network interface tuning
2. Network Round trip time (RTT)
3. Flow Control & Congestion Control

Network interface tuning

Check link mode and link speed by using -

dladm show-dev

Output example:

```
[root@n1: /] #dladm show-dev
eri0        link: unknown   speed: 100   Mbps      duplex: full
qfe0        link: unknown   speed: 100   Mbps      duplex: full
qfe1        link: unknown   speed: 100   Mbps      duplex: full
qfe2        link: unknown   speed: 0     Mbps      duplex: unknown
qfe3        link: unknown   speed: 0     Mbps      duplex: unknown
```

Link mode – should be full duplex.

Link speed – as much higher as possible.

Network Multipathing

Solaris 10 introduces - Link aggregation

Link aggregation allows aggregation of devices to appear as one device.

Link aggregation works only on GLDv3 compliant devices which are -

ge, bge, e1000g, nge and most newly 1Gbps network cards.

(ce and fibre-channel Gbps interfaces are not supported)

Allow aggregation by using -

dladm create-aggr -d bge0 -d bge1 1

you will then see a new device called - **aggr1** as a network device compound of bge0 and bge1.

IPMP

IPMP enables you to configure redundant network adapters, on the same server (node) and on the same subnet, as an IPMP *failover group.* IPMP work on both GLDv2 and GLDv3 compliant devices.

IPMP features

- Allows to configure a group of redundant network adapters to be on the same subnet that will have redundancy capabilities.
- Increase out band throughput (Out band throughput will increase to double a speed).

Permanent IPMP configuration

Edit `/etc/hosts` and add 2 test IP's for your server:

`/etc/hosts:`

```
10.0.0.220     n1   # this is the server's main IP (which resides on bge0)
10.0.0.240      n1-bge0-test  # this is bge0 test IP
10.0.0.241      n1-bge1-test # this is bge1 test IP
```

Edit `/etc/hostname.bge0` to have the main and test IP's

`/etc/hostname.bge0:`

```
n1 group group0 netmask + broadcast + up
addif n1-bge0-test -failover deprecated netmask + broadcast +  up
```

Edit `/etc/hostname.bge1` to have the test IP and to be a standby interface

`/etc/hostname.bge1:`

```
n1-bge1-test group therapy –failover deprecated  netmask + broadcast + up
```

Please note ! test ip's should not be used as a public IP's !

Flow control

flow control is the process of managing the rate of data transmission between two nodes to prevent a fast sender from outrunning a slow receiver. It provides a mechanism for the receiver to control the transmission speed, so that the receiving node is not overwhelmed with data from transmitting nodes.

Congestion control

Congestion control is the process of eliminating congestion on the network. Congestion control involves all the devices connected to the network.

TCP/IP tunable parameters

In order to view network parameters use ndd command.

To display tcp/ip parameters use -

ndd /dev/tcp \?

output example:

```
[root@n1: /root] #ndd /dev/tcp \?
?                              (read only)
tcp_time_wait_interval       (read and write)
tcp_conn_req_max_q           (read and write)
tcp_conn_req_max_q0          (read and write)
tcp_conn_req_min             (read and write)
...
tcp_keepalive_interval       (read and write)
tcp_xmit_hiwat               (read and write)
tcp_xmit_lowat               (read and write)
tcp_recv_hiwat               (read and write)
```

Tunable parameters

Parameter	Description	Recommendation
tcp_wscale_always (flow control)	Determine who should set the window size.	If set to 1 then window size is negotiated at connection initiation.
tcp_time_wait_interval	The interval before a socket is get closed	Set to 60,000ms (60 seconds)
tcp_conn_req_max_q	The maximum number of pending connections for a tcp listener. (128 by default)	If it seems you excessive connection requests. Set this value carefully, to be higher value than 128.
tcp_conn_req_max_q0	The maximum number of incomplete pending connections for a tcp listener. (1024 is the default)	On heavily loaded web server this can be adjusted to be higher.
tcp_xmit_hiwat	The default send window size. (default 48KB)	Can be set higher manually or by application.
tcp_recv_hiwat	The default receive window size. (default 48KB)	Can be set higher manually or by application.

Tuning a parameter example:

ndd -set /dev/tcp tcp_xmit_hiwat 98304

Displaying TCP statistics

You may view tcp statistics by using -

netstat -s -P tcp

output example:

```
[root@n1: /] #netstat -s -P tcp

TCP     tcpRtoAlgorithm     =     4     tcpRtoMin          =   400
        tcpRtoMax          = 60000     tcpMaxConn         =    -1
        tcpActiveOpens      =  8084     tcpPassiveOpens    =  1697
        tcpAttemptFails     =    41     tcpEstabResets     =  1587
        tcpCurrEstab        =    15     tcpOutSegs         =1646244
        tcpOutDataSegs      =1375904    tcpOutDataBytes    =1212996788
        tcpRetransSegs      =    92     tcpRetransBytes    =115313
        tcpOutAck          =281219     tcpOutAckDelayed   =176197
        tcpOutUrg          =     0     tcpOutWinUpdate    =     5
        tcpOutWinProbe      =     0     tcpOutControl      = 19539
        tcpOutRsts          =  1661     tcpOutFastRetrans  =     0
        tcpInSegs          =1258781
        tcpInAckSegs        =791686     tcpInAckBytes      =1121337478
        tcpInDupAck         =  8350     tcpInAckUnsent     =     0
        tcpInInorderSegs    =551406     tcpInInorderBytes  =224577252
        tcpInUnorderSegs    =   714     tcpInUnorderBytes  =924350
        tcpInDupSegs        =    19     tcpInDupBytes      =  2196
        tcpInPartDupSegs    =     0     tcpInPartDupBytes  =     0
        tcpInPastWinSegs    =     0     tcpInPastWinBytes  =     0
        tcpInWinProbe       =     0     tcpInWinUpdate     =     0
        tcpInClosed         =  1582     tcpRttNoUpdate     =    92
        tcpRttUpdate        =786798     tcpTimRetrans      =    11
        tcpTimRetransDrop   =     0     tcpTimKeepalive    =  1459
        tcpTimKeepaliveProbe=     1     tcpTimKeepaliveDrop =     0
        tcpListenDrop       =     0     tcpListenDropQ0    =     0
        tcpHalfOpenDrop     =     0     tcpOutSackRetrans  =    80
```

Warning and Critical TCP thresholds

Take a close look at the following calculations:

tcpRetransBytes/tcpOutDataBytes – tcp retransmitted bytes percentage.
(>15% warning, >25% Problem)

tcpInDupBytes/tcpInAckBytes – tcp-in duplicate packets.
(>15% warning, >25% Problem)

tcpListenDrop – drops for tcp_conn_req_max_q variable.

TcpListenDropQ0 - drops for tcp_conn_req_max_q0 variable.

Case study: UDP reordering problem

A customer has an application which sends udp packets and need to get the stream in a distinct order at another server (windows server).

The customer checks udp orders and finds that packets are not coming in the right order.

What will you do ?

Case study: Oracle qfs problem

A customer with E25K domain in oracle RAC is trying to get performance when using a BI process to connect to the oracle server. But when using qfs with forcedirectio it get no performance. cpu idle, memory idle, I/O is idle but the server response time is very low.

Case study: s TCP TIME_WAIT
Oracle server is generating a lot of TCP TIME_WAIT sockets.
But there seem to be no communication to the server everything is stucked.

Case study: apache web server has poor performance.
Solaris with apache server is serving clients very purly no collisions no application problems