USENIX '05 Annual Technical Conference
April 10-15, 2005
Marriott Anaheim
Anaheim, CA

solaris™

# SOLARIS™ Kernel
## Performance, Observability & Debugging


**Richard McDougall**
Distinguished Engineer
Performance and
Availability Engineering

Sun Microsystems, Inc.

r@sun.com

**James Mauro**
Senior Staff Engineer
Performance and
Availability Engineering

Sun Microsystems, Inc.

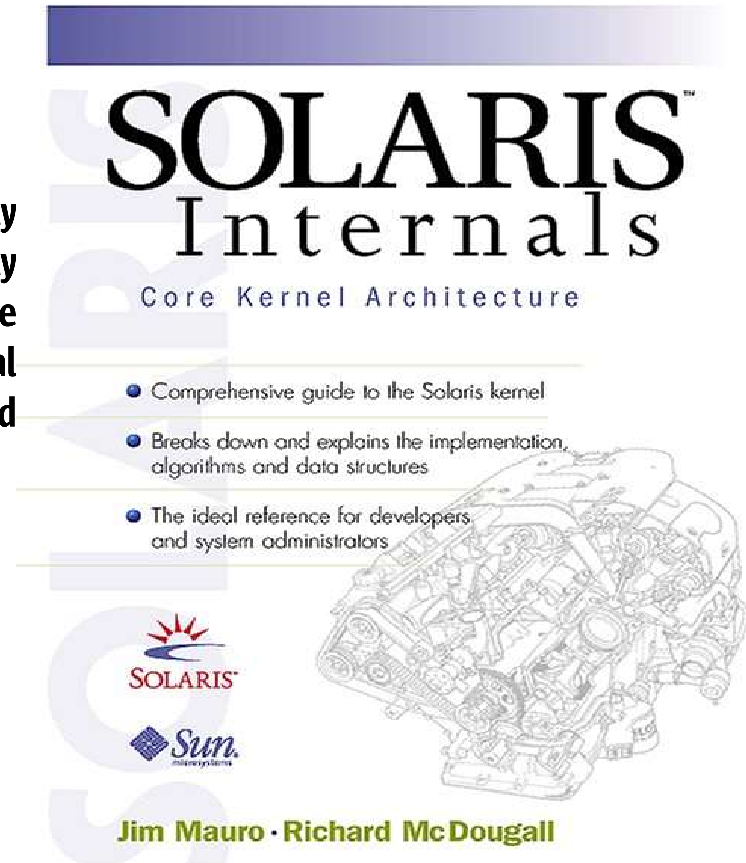
james.mauro@sun.com


Sun microsystems
We make the net work.

## About The Instructors

**Richard McDougall is a Distinguished Engineer in the Performance and Availability Engineering group at Sun Microsystems, where he focuses on large systems architecture, performance, measurement and observability. Richard's interests and expertise include the development of tools for measuring and Sizing Solaris systems. Among his numerous contributions, Richard designed and implemented numerous enhancements to the Solaris kernel virtual memory subsystem, and file system IO layer.**

**Jim Mauro is a Senior Staff Engineer in the Performance and Availability Engineering group at Sun Microsystems, where he focuses on availability benchmarking and system performance tuning. Jim's past efforts include developing a framework for measuring system availability, individual availability benchmarks, improving SunCluster availability and establishing company-wide metrics for assessing system availability.**




Richard and Jim authored Solaris Internals:
Core Kernel Architecture,
Prentice Hall, 2001. ISBN 0-13-022496-0

r@sun.com
james.mauro@sun.com

## Credits

Phil Harman: Multi-threading diagrams and Solaris introduction
Bryan Cantril, Mike Shapiro, Adam Leventhal: Solaris dtrace tutorial
Scott Fehrman: Virtual memory graphics
Kevin Sheehan: IO Topology slides

# Agenda – Day 1

- Session 1 - 9:00AM to 10:30PM
  - Goals, non goals and assumptions
  - Solaris Kernel Overview & Features
  - Observability & Tracing Tools & Utilities
- Session 2 - 11:00PM to 12:30PM
  - Memory
    - Virtual Memory
    - Physical Memory
    - Memory dynamics
    - Performance and Observability
    - Memory Resource Management

# Agenda – Day 1 (cont)

- Session 3 - 2:00PM to 3:30PM
  - Processes, threads & scheduling
    - The Solaris Multhreaded Process Model
    - The Dispatcher & Scheduling Classes
    - Performance & Observability
    - Processor Controls and Binding
- Session 4 - 4:00PM to 5:30PM
  - File Systems and I/O
    - I/O Overview
    - The Solaris VFS/Vnode Model
    - UFS – The Solaris Unix File System
    - Performance & Observability

# Agenda – Day 2

- Session 1 - 9:00AM to 10:30PM
  - DTrace
    - A Deeper Dive
  - A System View
  - Traps & Interrupts
- Session 2 - 11:00PM to 12:30PM
  - Advanced Memory Topics
    - Memory monitoring and measuring
    - Utilizing and tuning large memory

# Agenda – Day 2 (cont)

- Session 3 - 2:00PM to 3:30PM
  - Processes, threads & scheduling
    - A Deeping Dive
    - The Runtime Linker
    - Watching Processes with Dtrace
    - Process/Thread Lab
- Session 4 - 4:00PM to 5:30PM
  - Disk I/O Performance
  - File System Performance
  - Network Attached Storage
  - File System Performance Characterization
  - ZFS
  - Resource Management
  - Large System Performance

# Goals, Non-goals & Assumptions

- Goals
  - Architectural overview of the Solaris kernel
  - Drill down into key subsystems
  - The tools – what they are, what they do, when and how to use them
  - Correlate performance & observability to key functions
  - Resource control & management framework
- Non-goals
  - Detailed look at core kernel algorithms
  - Networking internals
- Assumptions
  - General familiarity with the Solaris environment
  - General familiarity with operating systems concepts

# Why Performance, Observability & Debugging?

- Reality, what a concept
  - Chasing performance problems
    - Sometimes they are even well defined
  - Chasing pathological behaviour
    - My app should be doing X, but it's doing Y
      - It's only doing it sometimes
  - Understand utilization
    - Resource consumption
      - CPU, Memory, IO
    - Capacity planning
  - In general, attaining a good understanding of the system, the workload, and how they interact
- 90% of system activity falls into one of the above categories, for a variety of roles
  - Admins, DBA's, Developers, etc...

# Before You Begin...

*“Would you tell me, please, which way I ought to go from here?”* asked Alice

“That depends a good deal on where you want to get to” said the Cat

“I don't much care where...” said Alice

“Then it doesn't matter which way you go” said the Cat

Lewis Carroll
*Alice's Adventures in Wonderland*

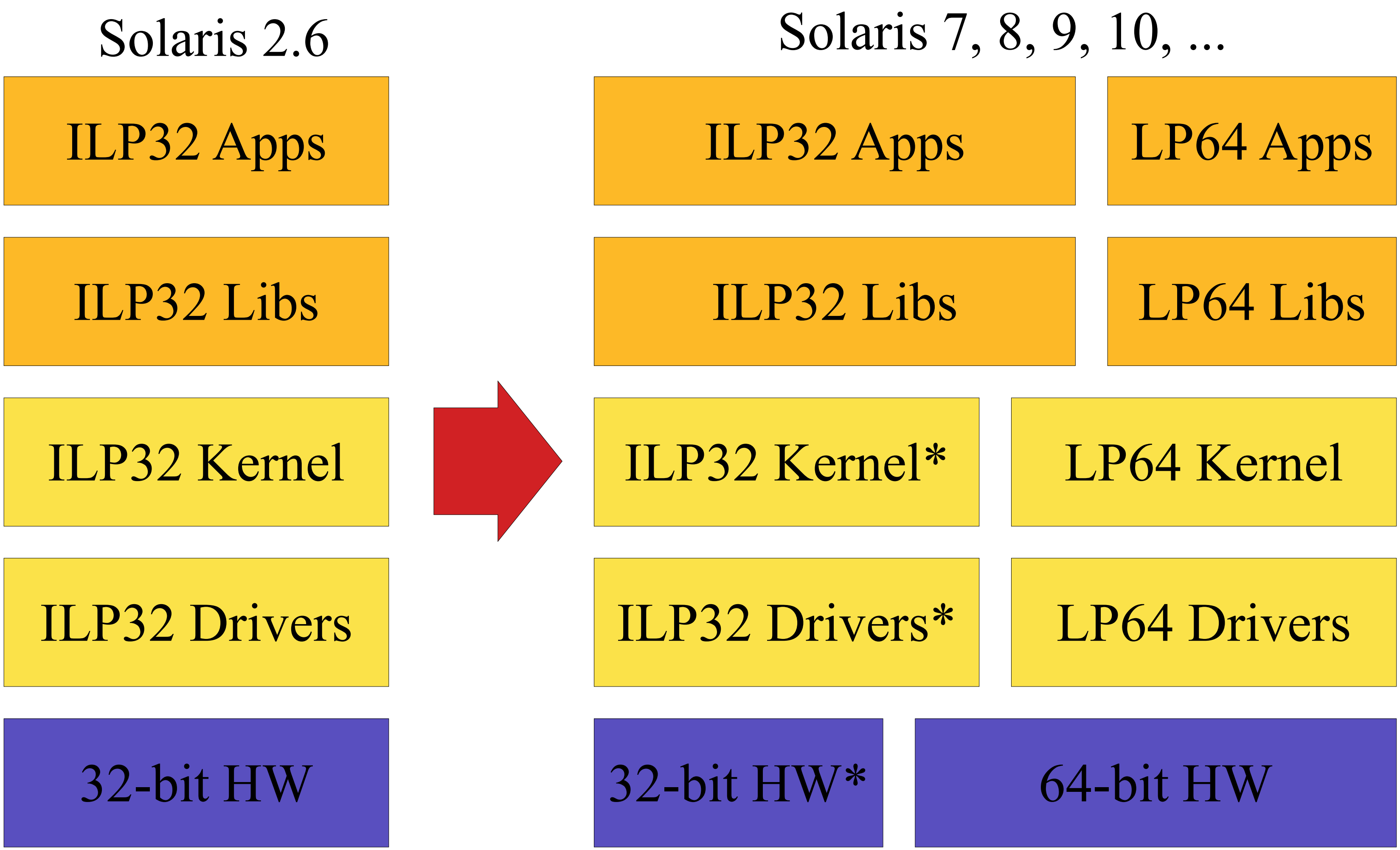
# General Methods & Approaches

- Define the problem
  - In terms of a business metric
  - Something measurable
- System View
  - Resource usage
    - CPU, Memory, Network, IO
- Process View
  - Execution profile
    - Where's the time being spent
  - May lead to a thread view
- Drill down depends on observations & goals
  - The path to root-cause has many forks
  - “bottlenecks” move
    - Moving to the next knee-in-the-curve

# Solaris Kernel Features

- Dynamic
- Multithreaded
- Preemptive
- Multithreaded Process Model
- Multiple Scheduling Classes
  - Including realtime support
- Tightly Integrated File System & Virtual Memory
- Virtual File System
- 64-bit kernel
  - 32-bit and 64-bit application support
- Resource Management
- Service Management & Fault Handling
- Integrated Networking

# The 64-bit Revolution



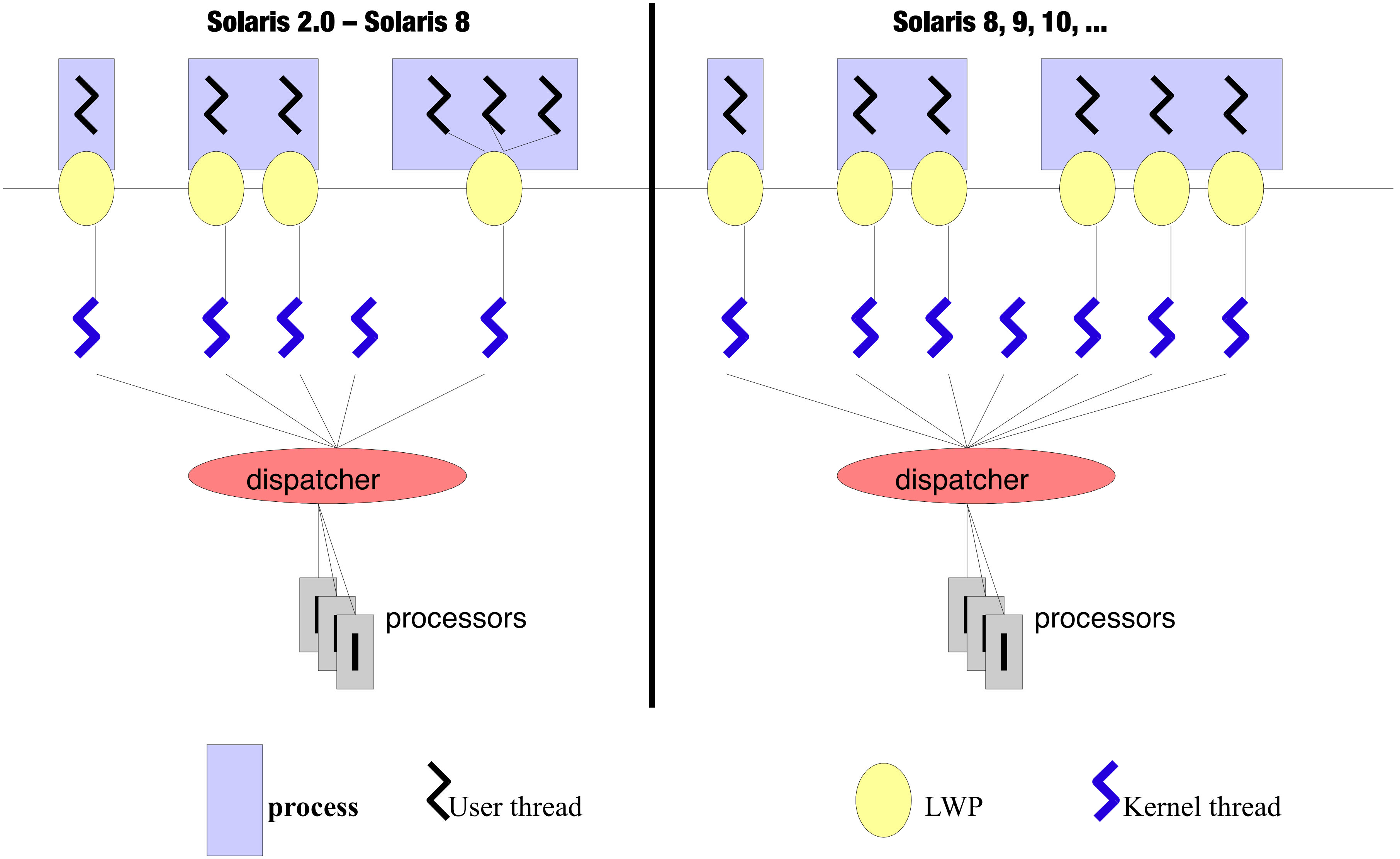

* Solaris 10: 64-bit kernel only on SPARC

# Solaris 8

## A Few Selected Highlights

- A new 1:1 threads implementation
  - /usr/lib/lwp/libthread.so
- Page cache enhancements (segmap)
  - Cyclic page cache
- **/dev/poll** for scalable I/O
- Modular debugging with **mdb**(1)
- You want statistics?
  - **kstat**(1M), **prstat**(1M), **lockstat**(1M), **busstat**(1M), **cpustat**(1M), ...
- UFS Direct I/O

# The Multithreading Revolution



# Solaris 9

## A Subset of the 300+ New Features

### Manageability

- Solaris Containers
- Solaris™ 9 Resource Manager
- IPQoS
- Solaris™ Volume Manager (SVM)
- Soft Disk Partitions
- Filesystem for DBMS
- UFS Snapshots
- Solaris™ Flash
- Solaris™ Live Upgrade 2.0
- Patch Manager
- Product Registry
- Sun ONE DS integration
- Legacy directory proxy
- Secure LDAP client
- Solaris WBEM Services
- Solaris instrumentation
- FRU ID
- Sun™ Management Center

### Availability

- Solaris Live Upgrade 2.0
- Dynamic Reconfiguration
- Sun StorEdge™ Traffic Manager Software
- IP Multipathing
- Reconfiguration Coordination Manager
- Driver Fault Injection Framework
- Mobile IP
- Reliable NFS
- TCP timers

### Security

- IPSec v4 and v6
- SunScreen Firewall
- Enhanced RBAC
- Kerberos V5
- IKE
- PAM enhancements
- Secure sockets layer (SSL)
- Solaris™ Secure Shell
- Extensible password encryption
- Solaris™ Security Toolkit
- TCP Wrappers
- Kernel and user-level encryption frameworks
- Random number generator
- SmartCard APIs

### Scalability

- IPv6
- Thread enhancements
- Memory optimization
  - Advanced page coloring
  - Mem Placement Optimization
  - Multi Page Size Support
- Hotspot JVM tuning
- NFS performance increase
- UFS Direct I/O
- Dynamic System Domains
- Enhanced DNLC
- RSM API
- J2SE™ 1.4 software with 64-bit and IPv6
- NCA enhancements

*. . . and more:*

- **Compatibility Guarantee**
- **Java Support**
- **Linux Compatibility**
- **Network Services**
- **G11N and Accessibility**
- **GNOME Desktop**

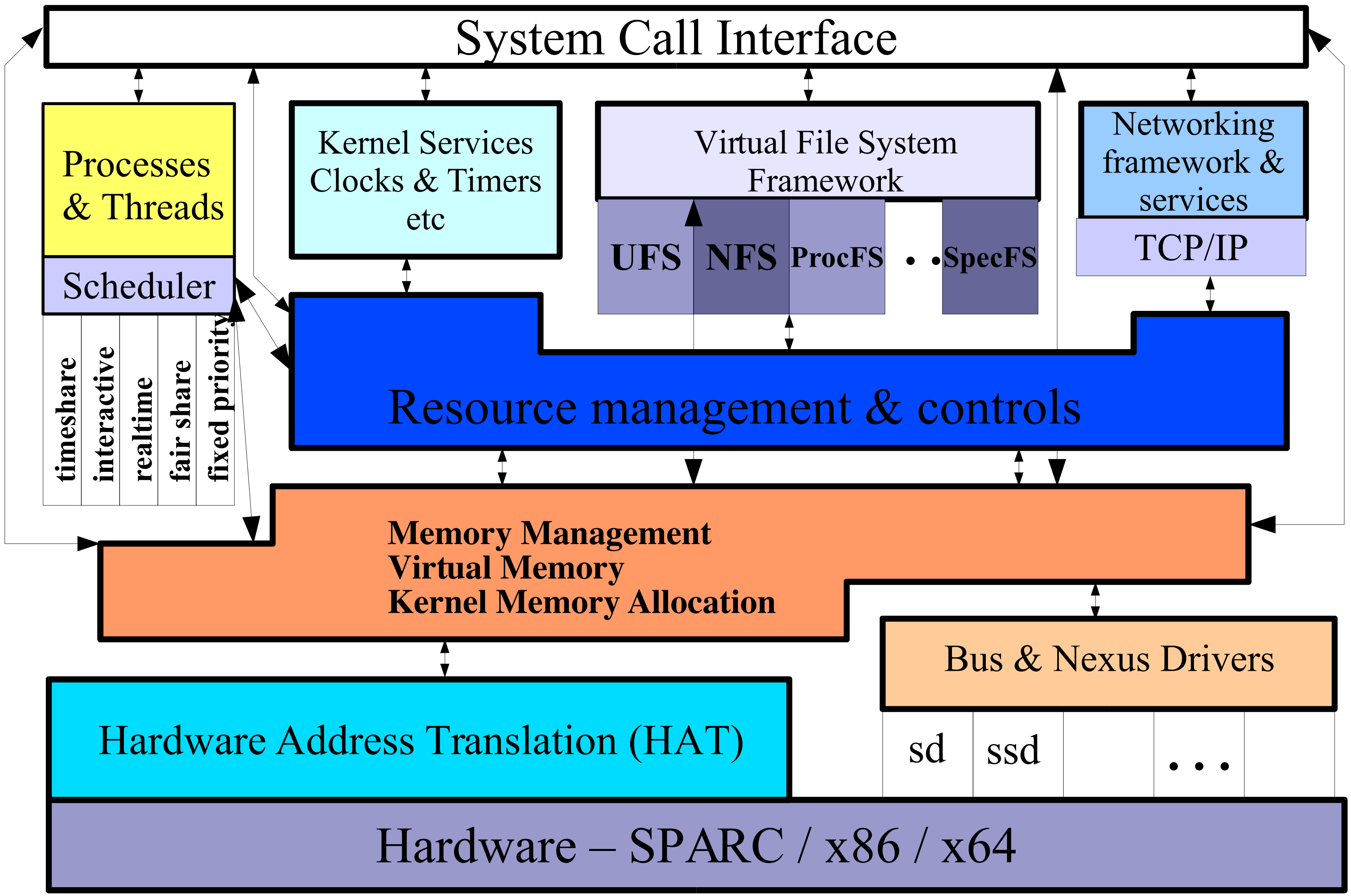
# Solaris 10

## The Headline Grabbers

- Solaris Containers (Zones)
- Solaris Dynamic Tracing (dtrace)
- Predictive Self Healing
  - System Management Framework
  - Fault Management Architecture
- Process Rights Management
- Premier x86 support
- Optimized 64-bit Opteron support (x64)
- Zetabyte Filesystem (ZFS)

... and much, much more!

# Solaris Kernel Overview



# Introduction To Performance & Observability Tools

# Solaris Performance and Tracing Tools

## Process stats

- cputrack - per-processor hw counters
- pargs – process arguments
- pflags – process flags
- pcred – process credentials
- pldd – process's library dependencies
- psig – process signal disposition
- pstack – process stack dump
- pmap – process memory map
- pfiles – open files and names
- prstat – process statistics
- ptree – process tree
- ptime – process microstate times
- pwdx – process working directory

## Process control

- pgrep – grep for processes
- pkill – kill processes list
- pstop – stop processes
- prun – start processes
- prctl – view/set process resources
- pwait – wait for process
- preap – reap a zombie process

## Process Tracing/ debugging

- abitrace – trace ABI interfaces
- dtrace – trace the world
- mdb – debug/control processes
- truss – trace functions and system calls

## Kernel Tracing/ debugging

- dtrace – trace and monitor kernel
- lockstat – monitor locking statistics
- lockstat -k – profile kernel
- mdb – debug live and kernel cores

## System Stats

- acctcom – process accounting
- busstat – Bus hardware counters
- cpustat – CPU hardware counters
- iostat – IO & NFS statistics
- kstat – display kernel statistics
- mpstat – processor statistics
- netstat – network statistics
- nfsstat – nfs server stats
- sar – kitchen sink utility
- vmstat – virtual memory stats

# Solaris 10 Dynamic Tracing - DTrace

***" [expletive deleted] It's like they saw inside my head and gave me The One True Tool."***

- A Slashdotter, in a post referring to DTrace

# DTrace

## Solaris Dynamic Tracing – An Observability Revolution

- Seamless, *global* view of the system from user-level thread to kernel
- Not reliant on pre-determined trace points, but *dynamic instrumentation*
- Data *aggregation* at source minimizes postprocessing requirements
- Built for live use on *production* systems

# DTrace

## Solaris Dynamic Tracing – An Observability Revolution

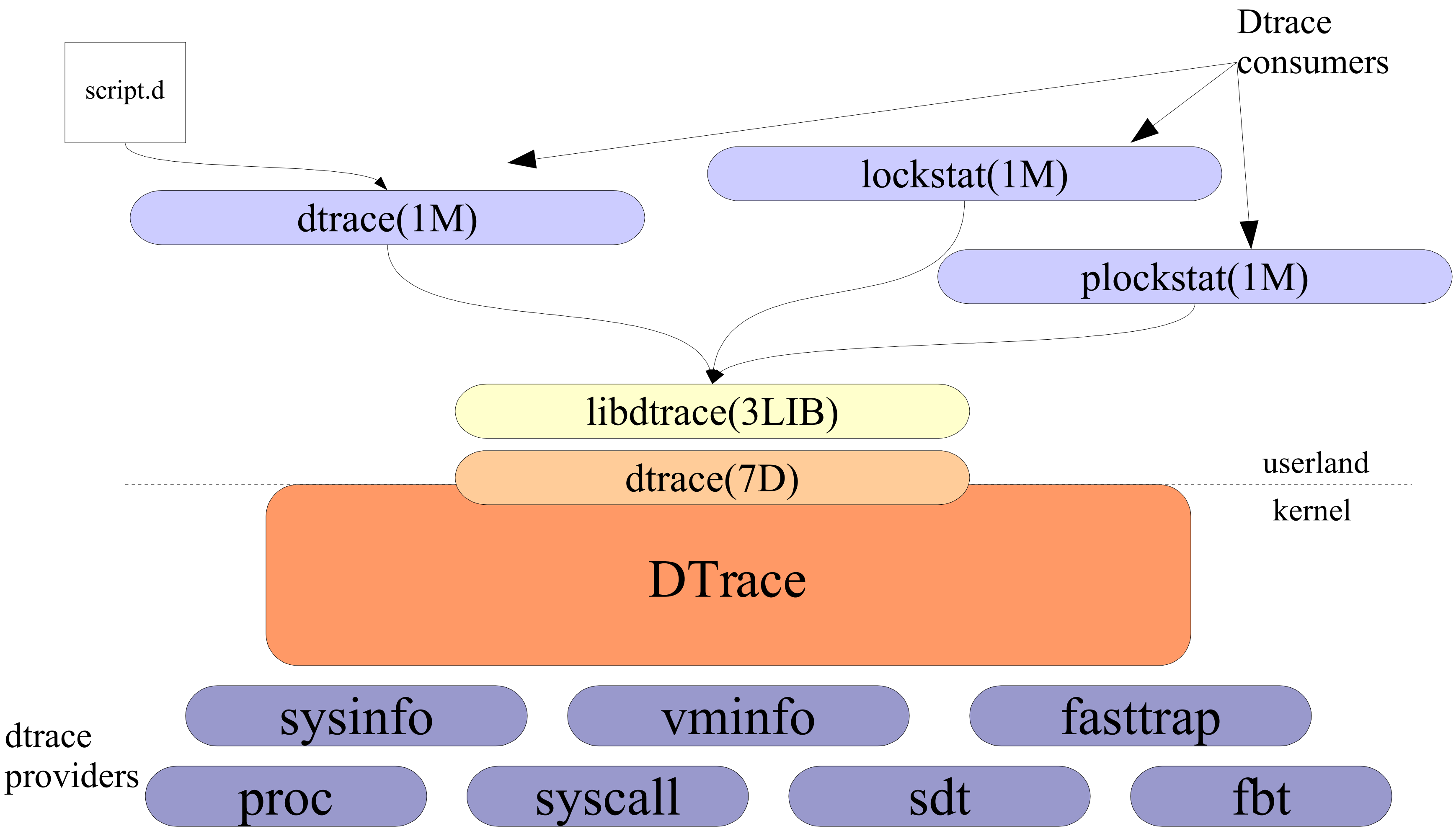
- Ease-of-use and *instant gratification* engenders serious *hypothesis testing*
- Instrumentation directed by high-level control language (not unlike AWK or C) for easy scripting and command line use
- Comprehensive probe coverage and powerful data management allow for *concise* answers to *arbitrary* questions

# DTrace Components

- Probes
  - A point of instrumentation
  - Has a name (string), and a unique probe ID (integer)
- Providers
  - DTrace-specific facilities for managing probes, and the interaction of collected data with consumers
- Consumers
  - A process that interacts with dtrace
  - typically `dtrace`(1)
- Using dtrace
  - Command line – `dtrace`(1)
  - Scripts written in the 'D' language

```
#!/usr/sbin/dtrace -s
```

# DTrace – The Big Picture



# DTrace

- Built-in variables
  - pid, tid, execname, probefunc, timestamp, zoneid, etc
- User defined variables
  - thread local
  - global
  - clause local
  - associative arrays
- All ANSI 'C' Operators
  - Arithmetic, Logical, Relational
- Predicates
  - Conditional expression before taking action
- Aggregations
  - process collected data at the source

# DTrace – command line

```
usenix> dtrace -n 'syscall:::entry { @scalls[probefunc] = count() }'
dtrace: description 'syscall:::entry ' matched 228 probes
^C

  lwp_self                                                          1
  fork1                                                             1
  fdsync                                                            1
  sigpending                                                        1
  rexit                                                             1
  fxstat                                                            1
  ...
  write                                                           205
  writev                                                          234
  brk                                                             272
  munmap                                                          357
  mmap                                                            394
  read                                                            652
  pollsys                                                         834
  ioctl                                                          1116
usenix>
```

# DTrace – D scripts

```
usenix> cat syscalls_pid.d
#!/usr/sbin/dtrace -s

dtrace:::BEGIN
{
        vtotal = 0;
}

syscall:::entry
/pid == $target/
{
        self->vtime = vtimestamp;
}
```

This is a complete dtrace program clause, including dtrace probename, a predicate and an action defined for when the probe fires that sets a thread-local variable

```
syscall:::return
/self->vtime/
{
        @vtime[probefunc] = sum(vtimestamp - self->vtime);
        vtotal += (vtimestamp - self->vtime);
        self->vtime = 0;
}

dtrace:::END
{
        normalize(@vtime, vtotal / 100);
        printa(@vtime);
}
```

# DTrace – Running syscalls_pid.d

```
usenix> ./syscalls_pid.d -c date
dtrace: script './sc.d' matched 458 probes
Sun Feb 20 17:01:28 PST 2005
dtrace: pid 2471 has exited
CPU     ID                    FUNCTION:NAME
  0      2                             :END
  getpid                                                    0
  gtime                                                     0
  sysi86                                                    1
  close                                                     1
  getrlimit                                                 2
  setcontext                                                2
  fstat64                                                   4
  brk                                                       8
  open                                                      8
  read                                                      9
  munmap                                                    9
  mmap                                                     11
  write                                                    15
  ioctl                                                    24
```

# Allowing dtrace for non-root users

- Setting dtrace privileges

```
Add a line for your user in /etc/user_attr:

rmc::::defaultpriv=dtrace_kernel,basic,proc_owner,dtrace_proc
```

# DTrace

## The Solaris Dynamic Tracing Observability Revolution

- Not just for diagnosing problems
- Not just for kernel engineers
- Not just for service personel
- Not just for application developers
- Not just for system administrators
- Serious fun
- Not to be missed!

# Modular Debugger - mdb(1)

- Solaris 8 `mdb`(1) replaces `adb`(1) and `crash`(1M)
- Allows for examining a live, running system, as well as post-mortem (dump) analysis
- Solaris 9 `mdb`(1) adds...
  - Extensive support for debugging of processes
  - /etc/crash and adb removed
  - Symbol information via compressed typed data
  - Documentation
- MDB Developers Guide
  - mdb implements a rich API set for writing custom dcmds
  - Provides a framework for kernel code developers to integrate with `mdb`(1)

# Modular Debugger - mdb(1)

- mdb(1) basics
  - 'd' commands (dcmd)
    - ::dcmds -l for a list
    - expression::dcmd
    - e.g. 0x300acde123::ps
  - walkers
    - ::walkers for a list
    - expression::walk <walker_name>
    - e.g. ::walk cpu
  - macros
    - !ls /usr/lib/adb for a list
    - expression$<macro
    - e.g. cpu0$<cpu

# Modular Debugger – mdb(1)

- Symbols and typed data
  - address::print (for symbol)
  - address::print <type>
  - e.g. cpu0::print cpu_t
  - cpu_t::sizeof
- Pipelines
  - expression, dcmd or walk can be piped
  - ::walk <walk_name> | ::dcmd
  - e.g. ::walk cpu | ::print cpu_t
  - Link Lists
  - address::list <type> <member>
  - e.g. 0x70002400000::list page_t p_vpnext
- Modules
  - Modules in /usr/lib/mdb, /usr/platform/lib/mdb etc
  - mdb can use adb macros
  - Developer Interface - write your own dcmds and walkers

# Kernel Statistics

- Solaris uses a central mechanism for kernel statistics
  - "kstat"
  - Kernel providers
    - raw statistics (c structure)
    - typed data
    - classed statistics
  - Perl and C API
  - `kstat`(1M) command

```
# kstat -n system_misc
module: unix                            instance: 0
name:   system_misc                     class:    misc

        avenrun_15min                   90
        avenrun_1min                    86
        avenrun_5min                    87
        boot_time                       1020713737
        clk_intr                        2999968
        crtime                          64.1117776
        deficit                         0
        lbolt                           2999968
        ncpus                           2
```

# Procfs Tools

- Observability (and control) for active processes through a pseudo file system (/proc)
- Extract interesting bits of information on running processes
- Some commands work on core files as well

```
pargs     pfiles
pflags    pstop
pcred     prun
pldd      pwait
psig      ptree
pstack    ptime
pmap      preap*
```

*why do Harry Cooper & Ben wish they had `preap`?

# pflags, pcred, pldd

```
sol8# pflags $$
482764: -ksh
  data model = _ILP32  flags = PR_ORPHAN
  /1:  flags = PR_PCINVAL|PR_ASLEEP [ waitid(0x7,0x0,0xffbff938,0x7) ]
```

```
sol8$ pcred $$
482764: e/r/suid=36413  e/r/sgid=10
  groups: 10 10512 570
```

```
sol8$ pldd $$
482764: -ksh
/usr/lib/libsocket.so.1
/usr/lib/libnsl.so.1
/usr/lib/libc.so.1
/usr/lib/libdl.so.1
/usr/lib/libmp.so.2
```

# psig

```
sol8$ psig $$
15481: -zsh
HUP  caught 0
INT  blocked,caught 0
QUIT blocked,ignored
ILL  blocked,default
TRAP blocked,default
ABRT blocked,default
EMT  blocked,default
FPE  blocked,default
KILL default
BUS  blocked,default
SEGV blocked,default
SYS  blocked,default
PIPE blocked,default
ALRM blocked,caught 0
TERM blocked,ignored
USR1 blocked,default
USR2 blocked,default
CLD  caught 0
PWR  blocked,default
WINCH  blocked,caught 0
URG  blocked,default
POLL blocked,default
STOP default
```

# pstack

```
sol8$ pstack 5591
5591:  /usr/local/mozilla/mozilla-bin
-----------------  lwp# 1 / thread# 1  --------------------
 fe99a254 poll    (513d530, 4, 18)
 fe8dda58 poll    (513d530, fe8f75a8, 18, 4, 513d530, ffbeed00) + 5c
 fec38414 g_main_poll (18, 0, 0, 27c730, 0, 0) + 30c
 fec37608 g_main_iterate (1, 1, 1, ff2a01d4, ff3e2628, fe4761c9) + 7c0
 fec37e6c g_main_run (27c740, 27c740, 1, fe482b30, 0, 0) + fc
 fee67a84 gtk_main (b7a40, fe482874, 27c720, fe49c9c4, 0, 0) + 1bc
 fe482aa4 ???????? (d6490, fe482a6c, d6490, ff179ee4, 0, ffe)
 fe4e5518 ???????? (db010, fe4e5504, db010, fe4e6640, ffbeeed0, 1cf10)
 00019ae8 ???????? (0, ff1c02b0, 5fca8, 1b364, 100d4, 0)
 0001a4cc main    (0, ffbef144, ffbef14c, 5f320, 0, 0) + 160
 00014a38 _start   (0, 0, 0, 0, 0, 0) + 5c
-----------------  lwp# 2 / thread# 2  --------------------
 fe99a254 poll    (fe1afbd0, 2, 88b8)
 fe8dda58 poll    (fe1afbd0, fe840000, 88b8, 2, fe1afbd0, 568) + 5c
 ff0542d4 ???????? (75778, 2, 3567e0, b97de891, 4151f30, 0)
 ff05449c PR_Poll  (75778, 2, 3567e0, 0, 0, 0) + c
 fe652bac ???????? (75708, 80470007, 7570c, fe8f6000, 0, 0)
 ff13b5f0 Main__8nsThreadPv (f12f8, ff13b5c8, 0, 0, 0, 0) + 28
 ff055778 ???????? (f5588, fe840000, 0, 0, 0, 0)
 fe8e4934 _lwp_start (0, 0, 0, 0, 0, 0)
```

# pfiles

```
sol8$ pfiles $$
pfiles $$
15481: -zsh
  Current rlimit: 256 file descriptors
   0: S_IFCHR mode:0620 dev:118,0 ino:459678 uid:36413 gid:7 rdev:24,11
      O_RDWR
   1: S_IFCHR mode:0620 dev:118,0 ino:459678 uid:36413 gid:7 rdev:24,11
  O_RDWR
   2: S_IFCHR mode:0620 dev:118,0 ino:459678 uid:36413 gid:7 rdev:24,11
  O_RDWR
   3: S_IFDOOR mode:0444 dev:250,0 ino:51008 uid:0 gid:0 size:0
  O_RDONLY|O_LARGEFILE FD_CLOEXEC  door to nscd[328]
  10: S_IFCHR mode:0620 dev:118,0 ino:459678 uid:36413 gid:7 rdev:24,11
  O_RDWR|O_LARGEFILE
```

# pwdx, pstop, pwait, ptree

```
sol8$ pwdx $$
15481: /home/rmc

sol8$ pstop $$
[argh!]

sol8$ pwait 23141

sol8$ ptree $$
285   /usr/sbin/inetd -ts
  15554 in.rlogind
    15556 -zsh
  15562 ksh
  15657 ptree 15562
```

# pgrep

```
sol8$ pgrep -u rmc
481
480
478
482
483
484
.....
```

# prstat(1)

- top-like utility to monitor running processes
- Sort on various thresholds (cpu time, RSS, etc)
- Enable system-wide microstate accounting
  - Monitor time spent in each microstate
- Solaris 9 - “projects” and “tasks” aware

```
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  2597 ks130310 4280K 2304K cpu1     0    0   0:01:25  22% imapd/1
 29195 bc21502  4808K 4160K sleep   59    0   0:05:26 1.9% imapd/1
  3469 tjobson  6304K 5688K sleep   53    0   0:00:03 1.0% imapd/1
  3988 tja      8480K 7864K sleep   59    0   0:01:53 0.5% imapd/1
  5173 root     2624K 2200K sleep   59    0  11:07:17 0.4% nfsd/18
  2528 root     5328K 3240K sleep   59    0  19:06:20 0.4% automountd/2
   175 root     4152K 3608K sleep   59    0   5:38:27 0.2% ypserv/1
  4795 snoqueen 5288K 4664K sleep   59    0   0:00:19 0.2% imapd/1
  3580 mauroj   4888K 4624K cpu3    49    0   0:00:00 0.2% prstat/1
  1365 bf117072 3448K 2784K sleep   59    0   0:00:01 0.1% imapd/1
  8002 root       23M   23M sleep   59    0   2:07:21 0.1% esd/1
  3598 wabbott  3512K 2840K sleep   59    0   0:00:00 0.1% imapd/1
 25937 pdanner  4872K 4232K sleep   59    0   0:00:03 0.1% imapd/1
 11130 smalm    5336K 4720K sleep   59    0   0:00:08 0.1% imapd/1
```

# truss(1)

- “trace” the system calls of a process/command
- Extended to support user-level APIs (-u, -U)
- Can also be used for profile-like functions (-D, -E)
- Is thread-aware as of Solaris 9 (pid/lwp_id)

```
usenix> truss -c -p 2556
^C
syscall                    seconds   calls  errors
read                          .013    1691
pread                         .015    1691
pread64                       .056     846
                          --------  ------   ----
sys totals:                   .085    4228      0
usr time:                     .014
elapsed:                     7.030
usenix> truss -D -p 2556
/2:      0.0304 pread(11, "02\0\0\001\0\0\0\n c\0\0".., 256, 0) = 256
/2:      0.0008 read(8, "1ED0C2 I", 4)                           = 4
/2:      0.0005 read(8, " @C9 b @FDD4 EC6", 8)                   = 8
/2:      0.0006 pread(11, "02\0\0\001\0\0\0\n c\0\0".., 256, 0) = 256
/2:      0.0134 pread64(10, "\0\0\0\0\0\0\0\0\0\0\0\0".., 8192, 0x18C8A000) = 8192
/2:      0.0006 pread(11, "02\0\0\001\0\0\0\n c\0\0".., 256, 0) = 256
/2:      0.0005 read(8, "D6 vE5 @", 4)                           = 4
/2:      0.0005 read(8, "E4CA9A -01D7AAA1", 8)                   = 8
/2:      0.0006 pread(11, "02\0\0\001\0\0\0\n c\0\0".., 256, 0) = 256
```

# lockstat(1M)

- Provides for kernel lock statistics (mutex locks, reader/writer locks)
- Also serves as a kernel profiling tool
- Use “-i 971” for the interval to avoid collisions with the clock interrupt, and gather fine-grained data

```
#lockstat -i 971 sleep 300 > lockstat.out

#lockstat -i 971 -I sleep 300 > lockstatI.out
```

# Examining Kernel Activity
## Kernel Profiling

```
# lockstat -kIi997 sleep 10

Profiling interrupt: 10596 events in 5.314 seconds (1994 events/sec)

Count indv cuml rcnt     nsec CPU+PIL             Caller
-------------------------------------------------------------------------------
 5122  48%  48% 1.00     1419 cpu[0]              default_copyout
 1292  12%  61% 1.00     1177 cpu[1]              splx
 1288  12%  73% 1.00     1118 cpu[1]              idle
  911   9%  81% 1.00     1169 cpu[1]              disp_getwork
  695   7%  88% 1.00     1170 cpu[1]              i_ddi_splhigh
  440   4%  92% 1.00     1163 cpu[1]+11           splx
  414   4%  96% 1.00     1163 cpu[1]+11           i_ddi_splhigh
  254   2%  98% 1.00     1176 cpu[1]+11           disp_getwork
   27   0%  99% 1.00     1349 cpu[0]              uiomove
   27   0%  99% 1.00     1624 cpu[0]              bzero
   24   0%  99% 1.00     1205 cpu[0]              mmrw
   21   0%  99% 1.00     1870 cpu[0]              (usermode)
    9   0%  99% 1.00     1174 cpu[0]              xcopyout
    8   0%  99% 1.00      650 cpu[0]              ktl0
    6   0%  99% 1.00     1220 cpu[0]              mutex_enter
    5   0%  99% 1.00     1236 cpu[0]              default_xcopyout
    3   0% 100% 1.00     1383 cpu[0]              write
    3   0% 100% 1.00     1330 cpu[0]              getminor
    3   0% 100% 1.00      333 cpu[0]              utl0
    2   0% 100% 1.00      961 cpu[0]              mmread
    2   0% 100% 1.00     2000 cpu[0]+10           read_rtc
```

# trapstat(1)

- Solaris 9, Solaris 10 (and beyond...)
- Statistics on CPU traps
  – Very processor architecture specific
- “-t” flag details TLB/TSB miss traps
  – Extremely useful for determining if large pages will help performance
    - Solaris 9 Multiple Page Size Support (MPSS)

# The *stat Utilities

- `mpstat(1)`
  - System-wide view of CPU activity
- `vmstat(1)`
  - Memory statistics
  - Don't forget “vmsat -p” for per-page type statistics
- `netstat(1)`
  - Network packet rates
  - Use with care – it does induce probe effect
- `iostat(1)`
  - Disk I/O statistics
  - Rates (IOPS), bandwidth, service times
- `sar(1)`
  - The kitchen sink

# cputrack(1)

- Gather CPU hardware counters, per process

```
solaris>  cputrack -N 20 -c pic0=DC_access,pic1=DC_miss -p 19849
   time lwp      event       pic0       pic1
  1.007   1       tick  34543793     824363
  1.007   2       tick         0          0
  1.007   3       tick 1001797338   5153245
  1.015   4       tick 976864106   5536858
  1.007   5       tick 1002880440   5217810
  1.017   6       tick 948543113   3731144
  2.007   1       tick  15425817     745468
  2.007   2       tick         0          0
  2.014   3       tick 1002035102   5110169
  2.017   4       tick 976879154   5542155
  2.030   5       tick 1018802136   5283137
  2.033   6       tick 1013933228   4072636
......

solaris> bc -l
824363/34543793
.02386428728310177171
((100-(824363/34543793)))
99.97613571271689822829
```

# Solaris Memory Architecture

# Virtual Memory

- Simple programming model/abstraction
- Fault Isolation
- Security
- Management of Physical Memory
- Sharing of Memory Objects
- Caching

# Solaris Virtual Memory

- Overview
- Internal Architecture
- Memory Allocation
- Paging Dynamics
- Swap Implementation & Sizing
- Kernel Memory Allocation
- SPARC MMU Overview
- Memory Analysis Tools

# Solaris Virtual Memory Glossary

| | |
|---|---|
| Address Space | Linear memory range visible to a program, that the instructions of the program can directly load and store. Each Solaris process has an address space; the Solaris kernel also has it's own address space. |
| Virtual Memory | Illusion of real memory within an address space. |
| Physical Memory | Real memory (e.g. RAM) |
| Mapping | A memory relationship between the address space and an object managed by the virtual memory system. |
| Segment | A co-managed set of similar mappings within an address space. |
| Text Mapping | The mapping containing the programs instructions and read-only objects. |
| Data Mapping | The mapping containing the programs initialized data |
| Heap | A mapping used to contain the programs heap (malloc'd) space |
| Stack | A mapping used to hold the programs stack |
| Page | A linear chunk of memory managed by the virtual memory system |
| VNODE | A file-system independant file object within the Solaris kernel |
| Backing Store | The storage medium used to hold a page of virtual memory while it is not backed by physical memory |
| Paging | The action of moving a page to or from it's backing store |

# Solaris Virtual Memory Glossary (cont)

| | |
|---|---|
| Swapping | The action of swapping an entire address space to/from the swap device |
| Swap Space | A storage device used as the backing store for anonymous pages. |
| Scanning | The action of the virtual memory system takes when looking for memory which can be freed up for use by other subsystems. |
| Named Pages | Pages which are mappings of an object in the file system. |
| Anonymous Memory | Pages which do not have a named backing store |
| Protection | A set of booleans to describe if a program is allowed to read, write or execute instructions within a page or mapping. |
| ISM | Intimate Shared Memory - A type of System V shared memory optimized for sharing between many processes |
| DISM | Pageable ISM |
| NUMA | Non-uniform memory architecture - a term used to describe a machine with differing processor-memory latencies. |
| Lgroup | A locallity group - a grouping of processors and physical memory which share similar memory latencies |
| MMU | The hardware functional unit in the microprocessor used to dynamically translate virtual addresses into physical addresses. |
| HAT | The Hardware Address Translation Layer - the Solaris layer which manages the translation of virtual addresses to physical addresses |

# Solaris Virtual Memory Glossary (cont)

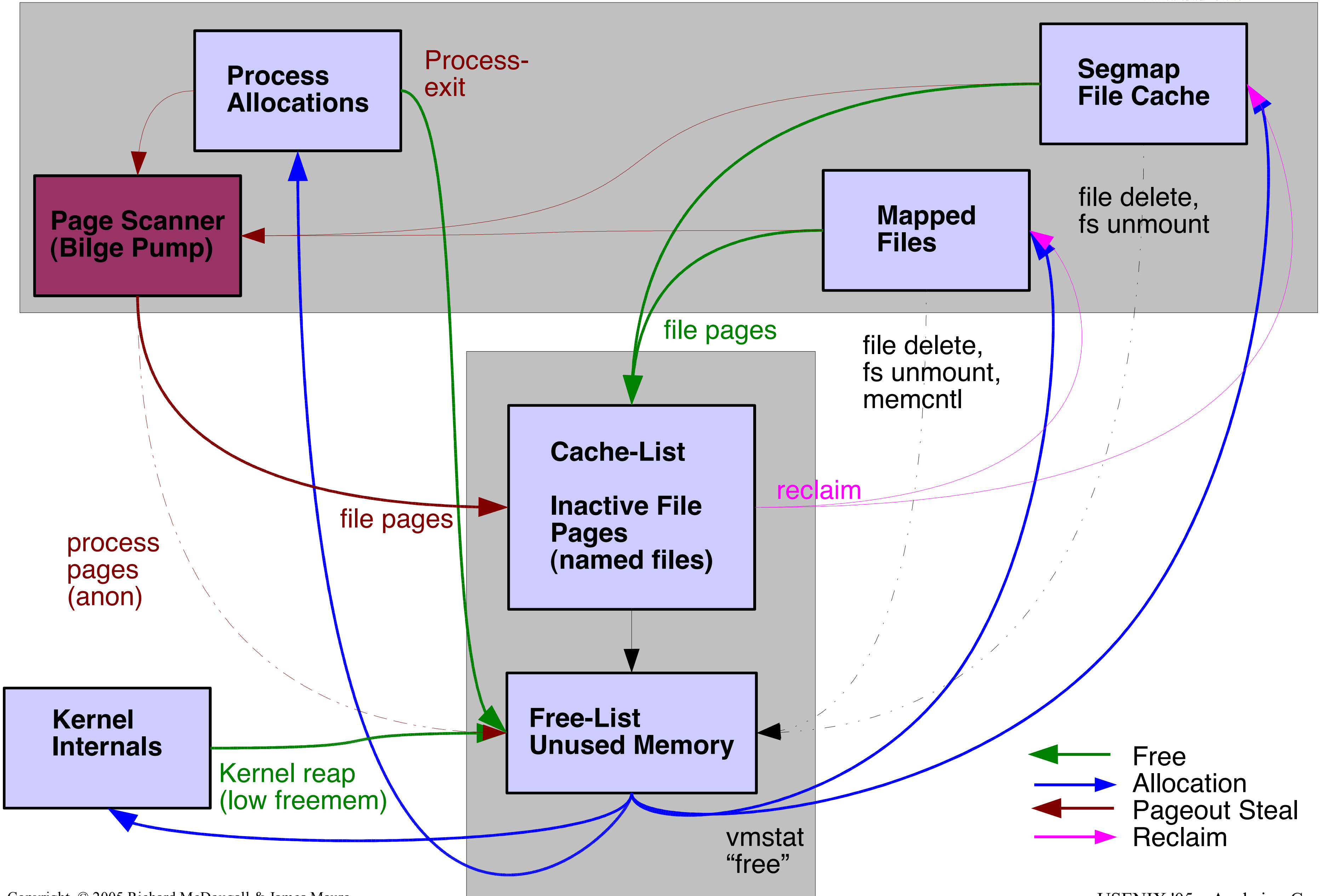
| | |
|---|---|
| TTE | Translation Table Entry - The UltraSPARC hardware's table entry which holds the data for virtual to physical translation |
| TLB | Translation Lookaside Buffer - the hardware's cache of virtual address translations |
| Page Size | The translation size for each entry in the TLB |
| TSB | Translation Software Buffer - UltraSPARC's software cache of ttes, used for lookup when a translation is not found in the TLB |

# Solaris Virtual Memory

- Demand Paged, Globally Managed
- Integrated file caching
- Layered to allow virtual memory to describe multiple memory types (Physical memory, frame buffers)
- Layered to allow multiple MMU architectures

# Part 1:
# Physical Memory Management

# Memory Allocation Transitions

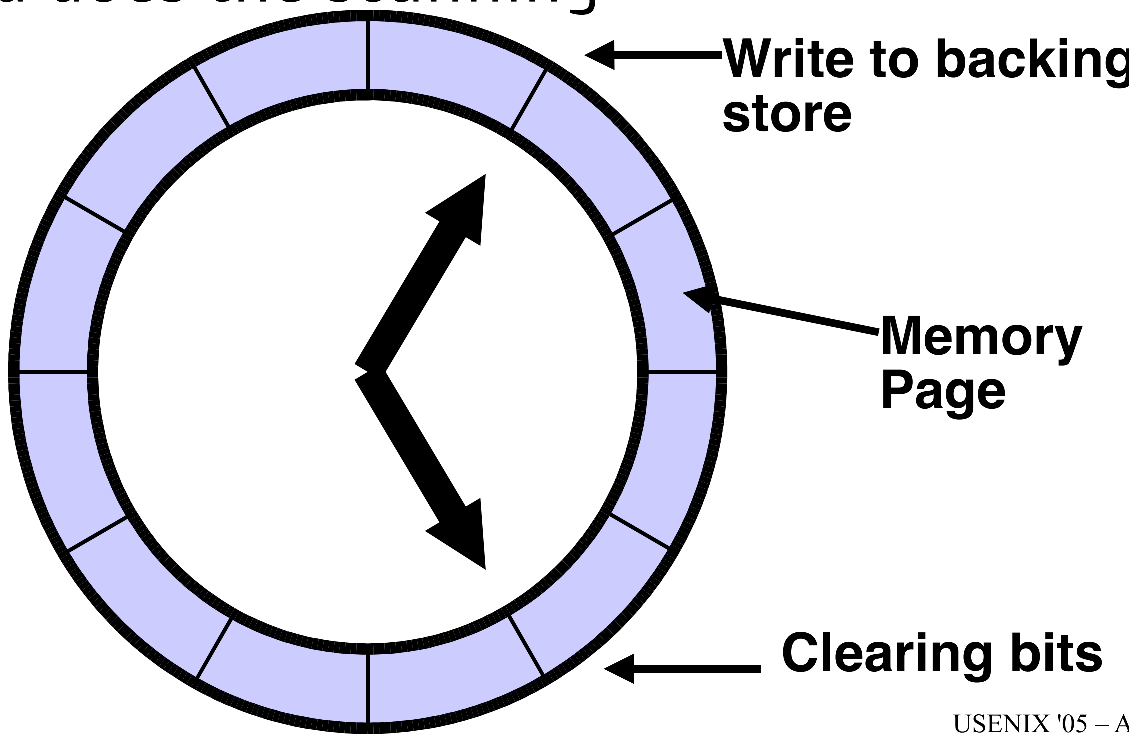


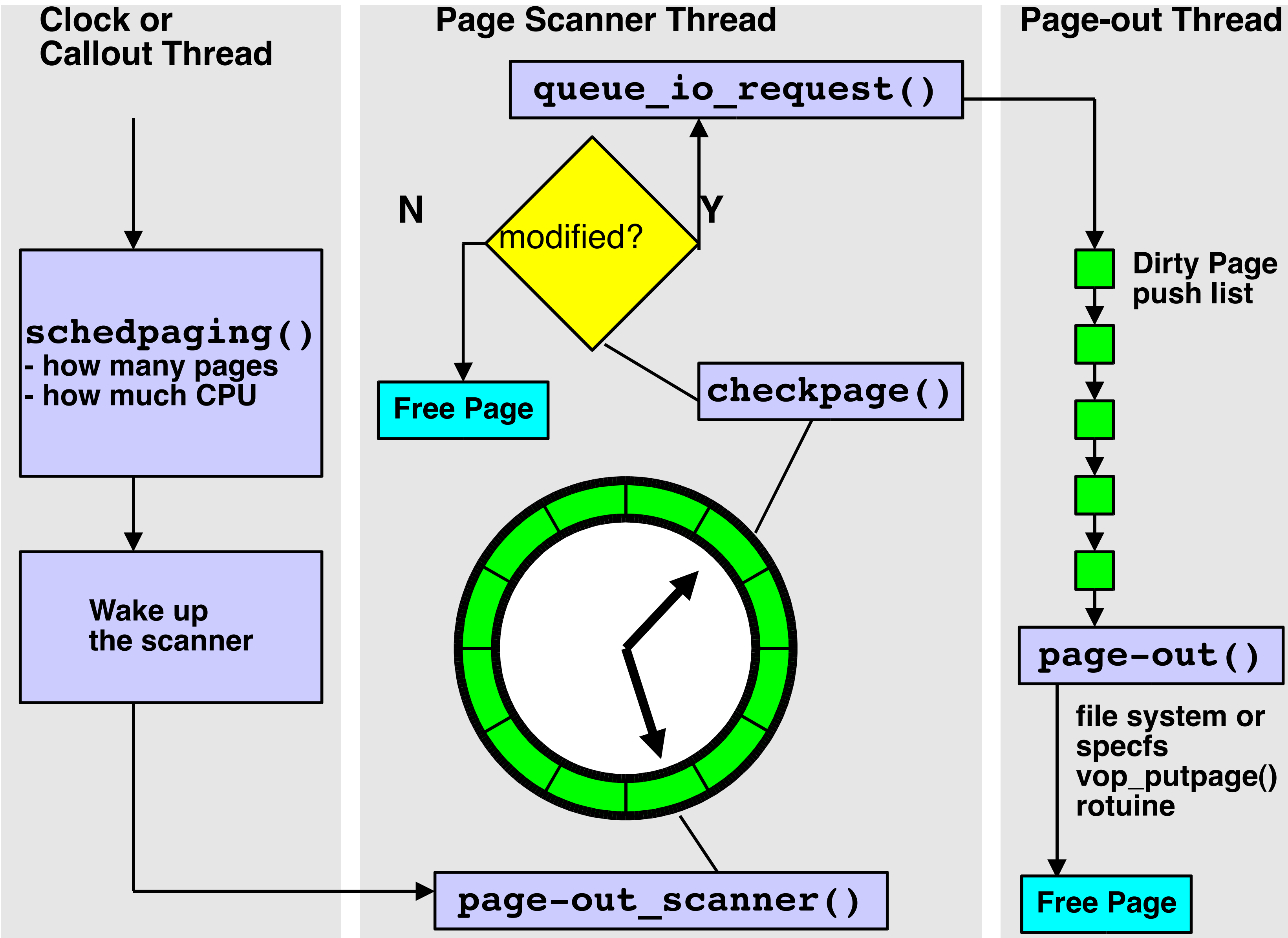
# Page Lists

- Free List
  - does not have a vnode/offset associated
  - put on list at process exit.
  - may be always small (pre Solaris 8)
- Cache List
  - still have a vnode/offset
  - seg_map free-behind and seg_vn executables and libraries (for reuse)
  - reclaims are in **`vmstat`** "re"
- Sum of these two are in **`vmstat`** "free"

# Page Scanning

- Steals pages when memory is low
- Uses a Least Recently Used process.
- Puts memory out to "backing store"
- Kernel thread does the scanning




Clock or
Callout Thread
Page Scanner Thread
Page-out Thread
queue_io_request()
N
modified?
Y
Dirty Page
push list
schedpaging()
- how many pages
- how much CPU
Free Page
checkpage()
Wake up
the scanner
page-out()
file system or
specfs
vop_putpage()
rotuine
page-out_scanner()
Free Page

# Scanning Algorithm

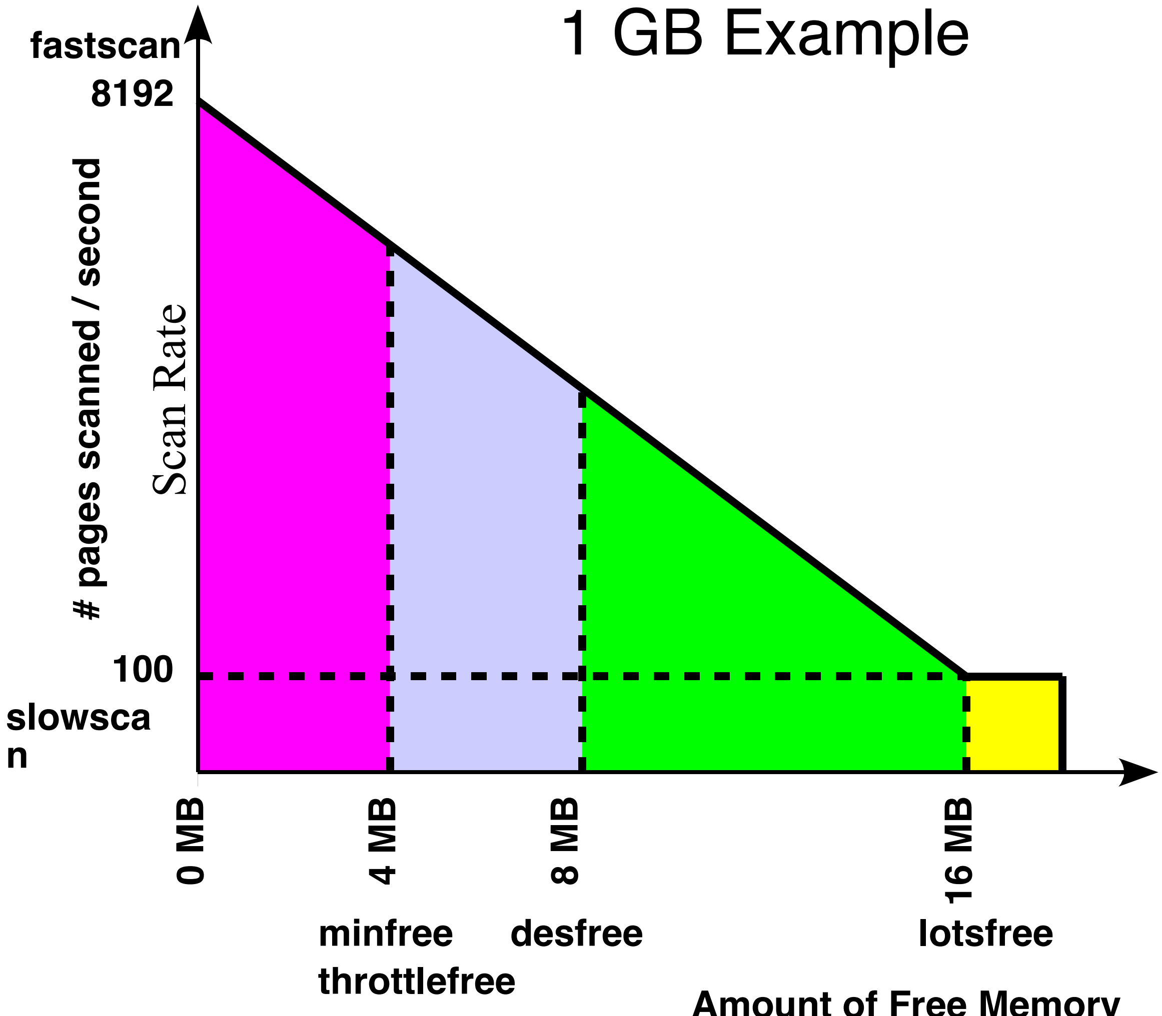
- Free memory is lower than (lotsfree)
- Starts scanning @ slowscan (pages/sec)
- Scanner Runs:
  - four times / second when memory is short
  - Awoken by page allocator if very low
- Limits:
  - Max # of pages /sec. swap device can handle
  - How much CPU should be used for scanning

$$\text{scanrate} = \left( \frac{\text{lotsfree - freemem}}{\text{lotsfree}} \text{ x fastscan} \right) + \left( \text{slowscan x } \frac{\text{freemem}}{\text{lotsfree}} \right)$$

# Scanning Parameters

| Parameter | Description | Min | Default ( Solaris 8) |
|---|---|---|---|
| lotsfree | starts stealing anonymous memory pages | 512K | 1/64 th of memory |
| desfree | scanner is started at 100 times/second | minfree | ½ of lotsfee |
| minfree | start scanning every time a new page is created | | ½ of desfree |
| throttlefree | page_create routine makes the caller wait until free pages are available | | minfree |
| fastscan | scan rate (pages per second) when free memory = minfree | slowscan | minimum of 64MB/s or ½ memory size |
| slowscan | scan rate (pages per second) when free memory = lotsfree | | 100 |
| maxpgio | max number of pages per second that the swap device can handle | ~60 | 60 or 90 pages per spindle |
| hand-spreadpages | number of pages between the front hand (clearing) and back hand (checking) | 1 | fastscan |
| min_percent_cpu | CPU usage when free memory is at lotsfree | 4% (~1 clock tick) | of a single CPU |

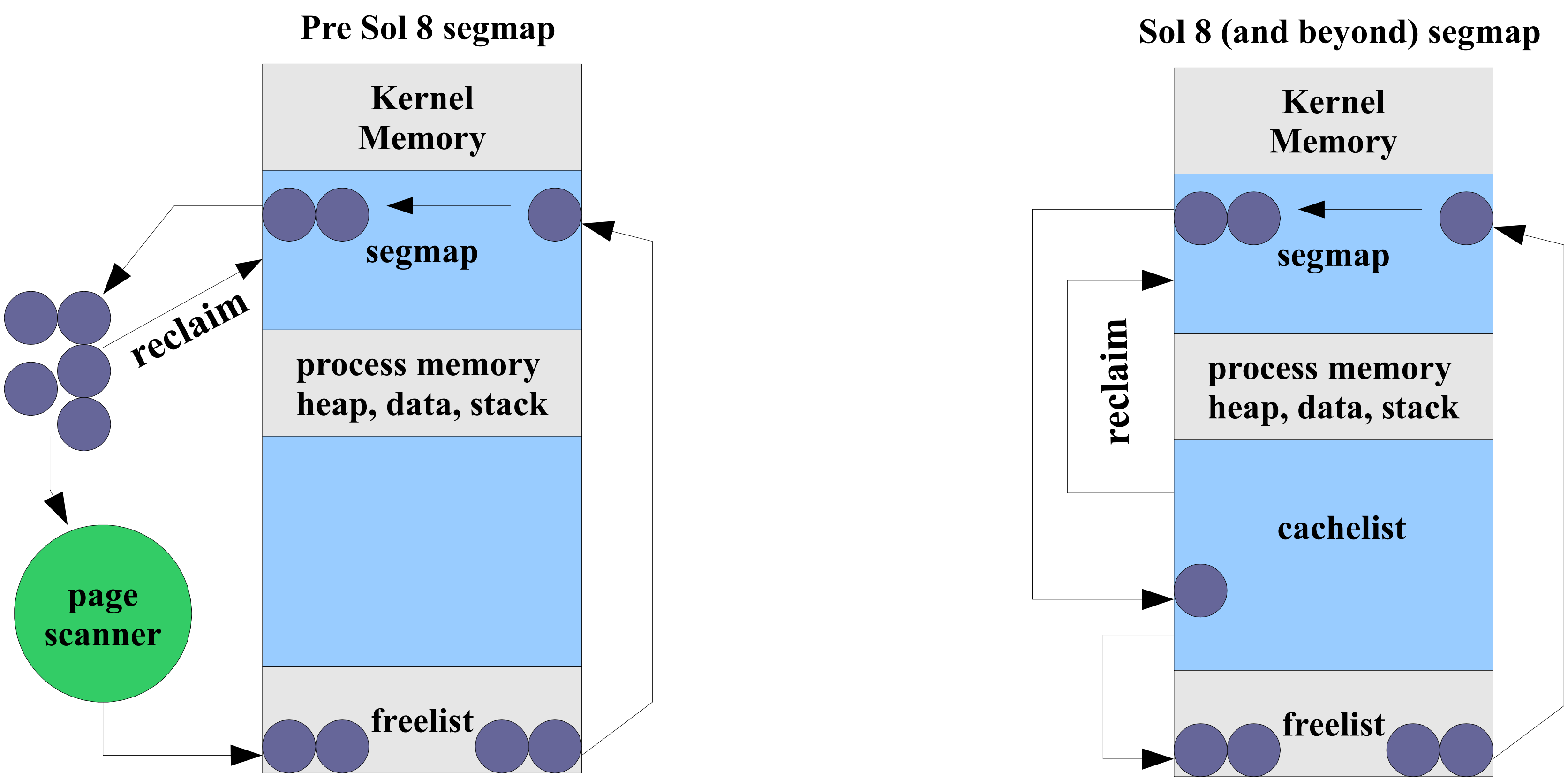
# Scan Rate


1 GB Example
fastscan
8192
# pages scanned / second
Scan Rate
100
slowsca
n
0 MB
4 MB
8 MB
16 MB
minfree
throttlefree
desfree
lotsfree
Amount of Free Memory

# The Solaris 8/9 Cache

- Page list is broken into two:
  - Cache List: pages with a valid vnode/offset
  - Free List: pages has no vnode/offset
- Unmapped pages where just released
- Non-dirty pages, not mapped, should be on the "free list"
- Places pages on the "tail" cache/free list
- Free memory = cache + free

# The Solaris 8/9 Cache



# The Solaris 8/9 Cache

- Now `vmstat` reports a useful `free`
- Throw away your old `/etc/system` pager configuration parameters
  - `lotsfree, desfree, minfree`
  - `fastscan, slowscan`
  - `priority_paging, cachefree`

# Solaris 8/9 - VM Changes

- Observability
  - Free memory now contains file system cache
    - Higher free memory
    - vmstat 'free' column is meaningful
  - Easier visibility for memory shortages
    - Scan rates != 0 - Memory shortage
- Correct Defaults
  - No tuning required – delete all /etc/system VM parameters!

# Memory Summary

## Physical Memory:

```
# prtconf
System Configuration:  Sun Microsystems  sun4u
Memory size: 512 Megabytes
```

## Kernel Memory:

```
# sar -k 1 1

SunOS ian 5.8 Generic_108528-03 sun4u    08/28/01

13:04:58 sml_mem   alloc  fail  lg_mem   alloc  fail  ovsz_alloc  fail
13:04:59 10059904 7392775     0 133349376 92888024     0    10346496     0
```

## Free Memory:

```
# vmstat 3 3

 procs     memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr f0 s0 s1 s6   in   sy   cs us sy id
 0 0 0 478680 204528  0   2  0  0  0  0  0  0  0  1  0  209 1886  724 35  5 61
 0 0 0 415184 123400  0   2  0  0  0  0  0  0  0  0  0  238  825  451  2  1 98
 0 0 0 415200 123416  0   0  0  0  0  0  0  0  0  3  0  219  788  427  1  1 98
```

# Solaris 9 Memory Summary

```
# mdb -k
Loading modules: [ unix krtld genunix ufs_log ip usba s1394 nfs random
ptm ipc logindmux cpc ]
> ::memstat
Page Summary                Pages                MB  %Tot
------------     ----------------  ----------------  ----
Kernel                      10145                79    4%
Anon                        21311               166    9%
Exec and libs               15531               121    6%
Page cache                  69613               543   28%
Free (cachelist)           119633               934   48%
Free (freelist)             11242                87    5%

Total                      247475              1933
```

# Vmstat

r = run queue length
b = processes blocked waiting for I/O
w = idle processes that have been swapped at some time

swap = free and unreserved swap in KBytes
free = free memory measured in pages

re = kilobytes reclaimed from cache/free list
mf = minor faults - the page was in memory but was not mapped
pi = kilobytes paged-in from the file system or swap device
po = kilobytes paged-out to the file system or swap device
fr = kilobytes that have been destroyed or freed
de = kilobytes freed after writes
sr = pages scanned / second

s0-s3 = disk I/Os per second for disk 0-3

in = interrupts / second
sy = system calls / second
cs = context switches / second

us = user cpu time
sy = kernel cpu time
id = idle + wait cpu time

```
# vmstat 5 5
 procs      memory                  page                   disk          faults           cpu
 r b w   swap   free     re  mf pi po  fr  de  sr  f0  s0  s1  s2    in    sy     cs    us sy id
 ...
 0 0 0 46580232 337472  18 194 30  0   0   0   0   0   0   0   0   5862  81260 28143  19  7 74
 0 0 0 45311368 336280  32 249 48  0   0   0   0   0   0   0   0   6047  93562 29039  21 10 69
 0 0 0 46579816 337048  12 216 60  0   0   0   0   0  10   0   7   5742 100944 27032  20  7 73
 0 0 0 46580128 337176   3 111  3  0   0   0   0   0   0   0   0   5569  93338 26204  21  6 73
```

# Vmstat -p

swap = free and unreserved swap in KBytes
free = free memory measured in pages

re = kilobytes reclaimed from cache/free list
mf = minor faults - the page was in memory but was not mapped
fr = kilobytes that have been destroyed or freed
de = kilobytes freed after writes
sr = kilobytes scanned / second

executable pages: kilobytes in - out - freed

anonymous pages: kilobytes in - out – freed

file system pages: kilobytes in - out - freed

```
# vmstat -p 5 5
```

| memory | | page | | | | | executable | | | anonymous | | | filesystem | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| swap | free | re | mf | fr | de | sr | epi | epo | epf | api | apo | apf | fpi | fpo | fpf |
| ... | | | | | | | | | | | | | | | |
| 46715224 | 891296 | 24 | 350 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 27 | 0 | 0 |
| 46304792 | 897312 | 151 | 761 | 25 | 0 | 0 | 17 | 0 | 0 | 1 | 0 | 0 | 280 | 25 | 25 |
| 45886168 | 899808 | 118 | 339 | 1 | 0 | 0 | 3 | 0 | 0 | 1 | 0 | 0 | 641 | 1 | 1 |
| 46723376 | 899440 | 29 | 197 | 0 | 0 | 0 | 0 | 0 | 0 | 40 | 0 | 0 | 60 | 0 | 0 |

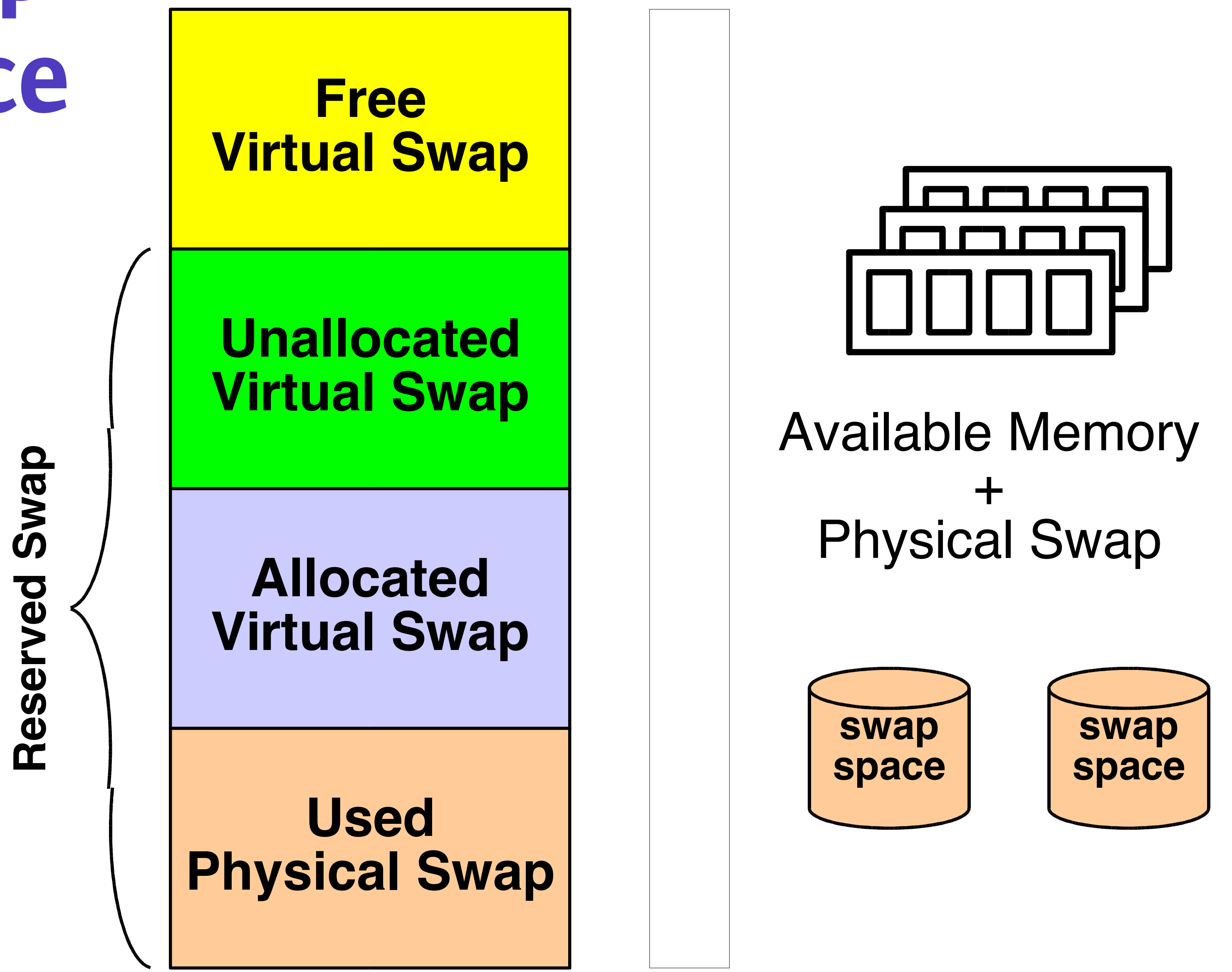
# Swapping

- Scheduler/Dispatcher:
  - Dramatically affects process performance
  - Used when demand paging is not enough
- Soft swapping:
  - Avg. freemem below desfree for 30 sec.
  - Look for inactive processes, at least **`maxslp`**
- Hard swapping:
  - Run queue >= 2 (waiting for CPU)
  - Avg. freemem below desfree for 30 sec.
  - Excessive paging, **`(pageout + pagein ) > maxpgio`**
  - Aggressive; unload kernel mods & free cache

# Swap space states

- Reserved:
  - Virtual space is reserved for the segment
  - Represents the virtual size being created
- Allocated:
  - Virtual space is allocated when the first physical page is assigned
  - A swapfs vnode / offset are assigned
- Swapped out:
  - When a shortage occurs
  - Page is swapped out by the scanner, migrated to swap storage

# Swap Space


Free
Virtual Swap
Unallocated
Virtual Swap
Allocated
Virtual Swap
Used
Physical Swap
Reserved Swap
Available Memory
+
Physical Swap
swap
space
swap
space

# Swap Usage

- Virtual Swap:
  - reserved: unallocated + allocated
  - available = bytes

```
# swap -s
total: 175224k bytes unallocated + 24464k allocated = 199688k reserved, 416336k available
```

- Physical Swap:
  - space available for physical page-outs
  - free = blocks (512 bytes)

```
# swap -l
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t1d0s1   32,9      16 524864 524864
```

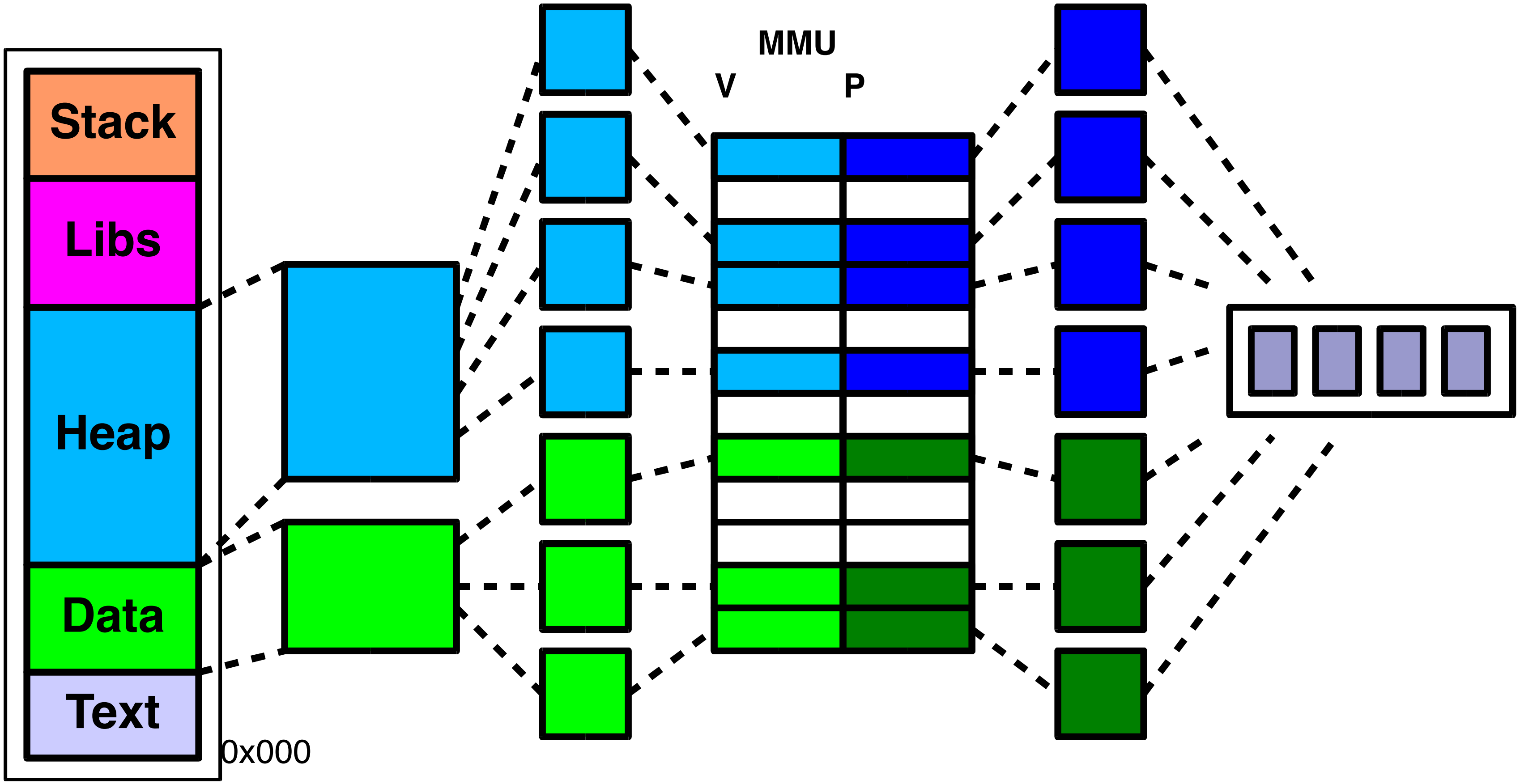
- Ensure both are non-zero
  - swap -s "available"
  - swap -l "free"

# Part 2:
# Address Spaces: A Deeper Dive

# Example Program

```
#include <sys/types.h>

const char * const_str = "My const string";
char * global_str = "My global string";
int    global_int = 42;

int
main(int argc, char * argv[])
{
        int local_int = 123;
        char * s;
        int i;
        char command[1024];


        global_int = 5;
        s = (char *)malloc(14000);
        s[0] = 'a';
        s[100] = 'b';
        s[8192] = 'c';

}
```

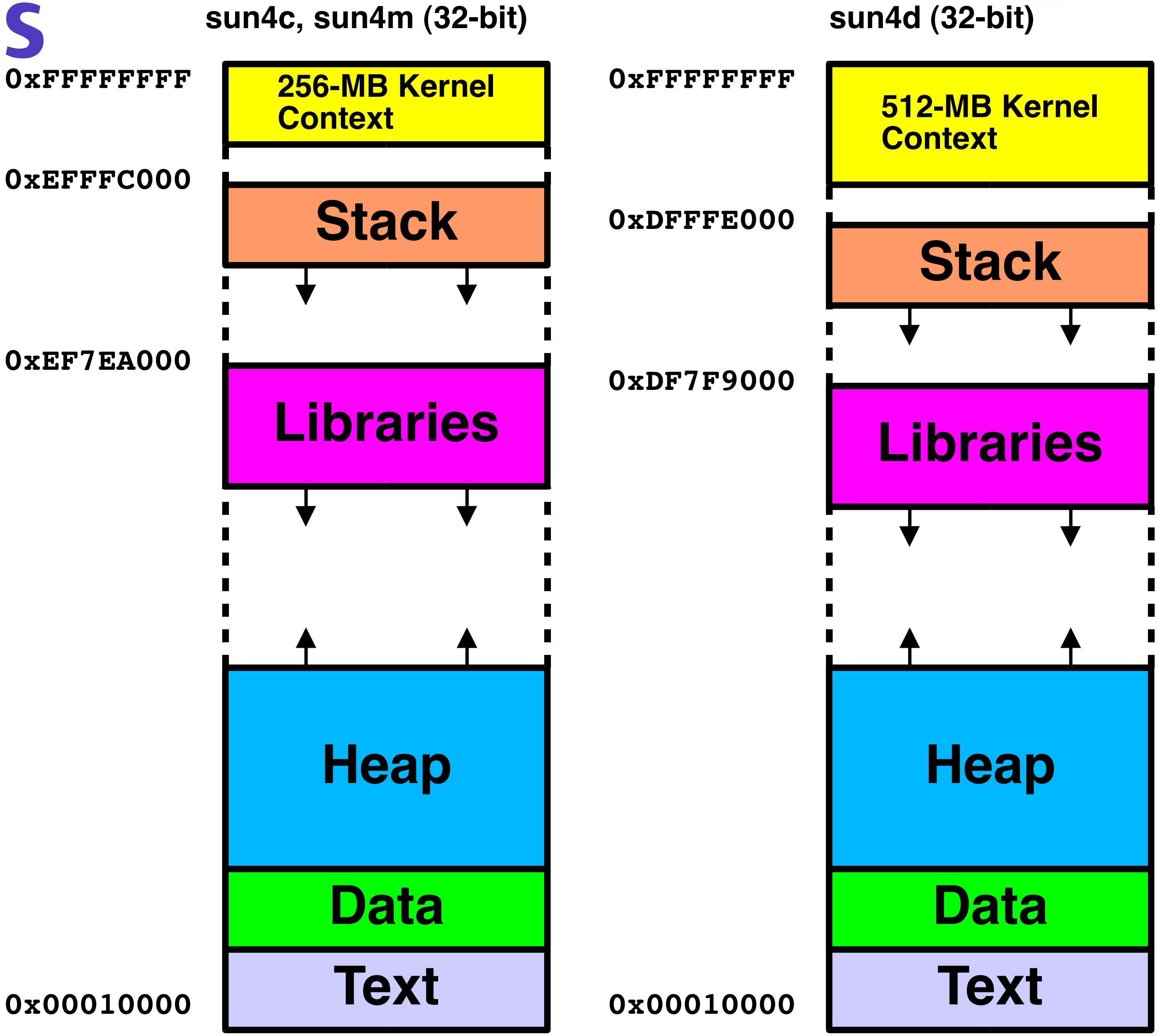
# Virtual to Physical

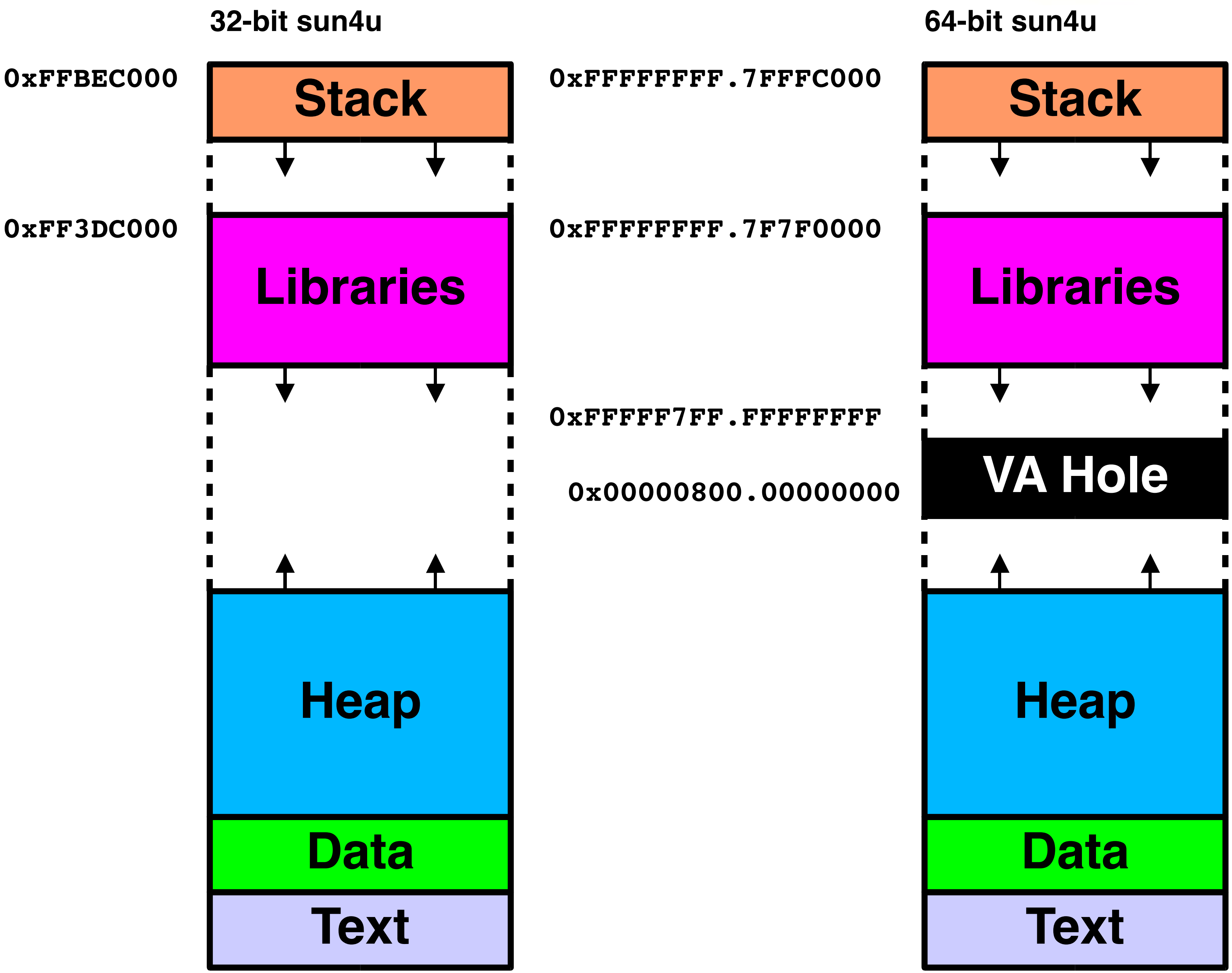


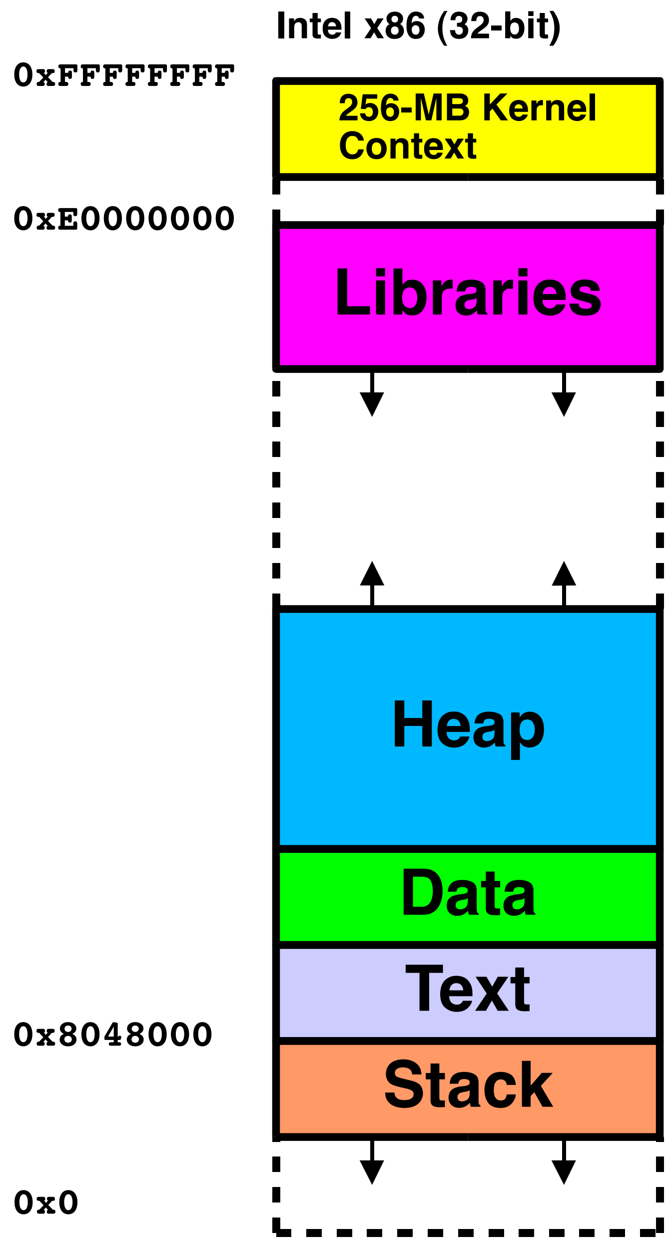
# Address Space

- Process Address Space
  - Process Text and Data
  - Stack (anon memory) and Libraries
  - Heap (anon memory)
- Kernel Address Space
  - Kernel Text and Data
  - Kernel Map Space (data structs, caches)
  - 32-bit Kernel map (64-bit Kernels only)
  - Trap table
  - Critical virtual memory data structures
  - Mapping File System Cache (segmap)

# Address Space




32-bit sun4u
64-bit sun4u
0xFFBEC000
Stack
0xFFFFFFFF.7FFFC000
Stack
0xFF3DC000
Libraries
0xFFFFFFFF.7F7F0000
Libraries
0xFFFFF7FF.FFFFFFFF
VA Hole
0x00000800.00000000
Heap
Data
Text
Heap
Data
Text



Intel x86 (32-bit)
0xFFFFFFFF
256-MB Kernel Context
0xE0000000
Libraries
Heap
Data
Text
0x8048000
Stack
0x0

# Pmap -x (Solaris 8)

```
Sol8# /usr/proc/bin/pmap -x $$

18084:   csh
 Address   Kbytes Resident Shared Private Permissions       Mapped File
00010000      144      144    136       8 read/exec         csh
00044000       16       16      -      16 read/write/exec   csh
00048000      120      104      -     104 read/write/exec     [ heap ]
FF200000      672      624    600      24 read/exec         libc.so.1
FF2B8000       24       24      -      24 read/write/exec   libc.so.1
FF2BE000        8        8      -       8 read/write/exec   libc.so.1
FF300000       16       16      8       8 read/exec         libc_psr.so.1
FF320000        8        8      -       8 read/exec         libmapmalloc.so.1
FF332000        8        8      -       8 read/write/exec   libmapmalloc.so.1
FF340000        8        8      -       8 read/write/exec     [ anon ]
FF350000      168      112     88      24 read/exec         libcurses.so.1
FF38A000       32       32      -      32 read/write/exec   libcurses.so.1
FF392000        8        8      -       8 read/write/exec   libcurses.so.1
FF3A0000        8        8      -       8 read/exec         libdl.so.1
FF3B0000      136      136    128       8 read/exec         ld.so.1
FF3E2000        8        8      -       8 read/write/exec   ld.so.1
FFBE6000       40       40      -      40 read/write/exec     [ stack ]
--------   ------   ------ ------  ------
total Kb     1424     1304    960     344
```

# Process Heap Sizes

| Solaris Version | Max Heap Size | Notes |
|---|---|---|
| Solaris 2.5 | 2 GBytes | |
| Solaris 2.5.1 | 2 GBytes | |
| Solaris 2.5.1 w/ patch 103640-08 or greater | 3.75 GBytes | Need to reboot to increaase limit above 2 GB with ulimit |
| Solaris 2.5.1 w/ patch 103640-23 or greater | 3.75 GBytes | Do not need to be root to increase limit |
| Solaris 2.6 | 3.75 GBytes | Need to increase beyond 2 BG with ulimit |
| Solaris 7 or 8 (32-bit mode) | 3.75 / 3.90 GBytes | non-sun4u / sun4u |
| Solaris 7 or 8 (64-bit mode) | 16 TBytes (Ultra) | Virtually unlimited |

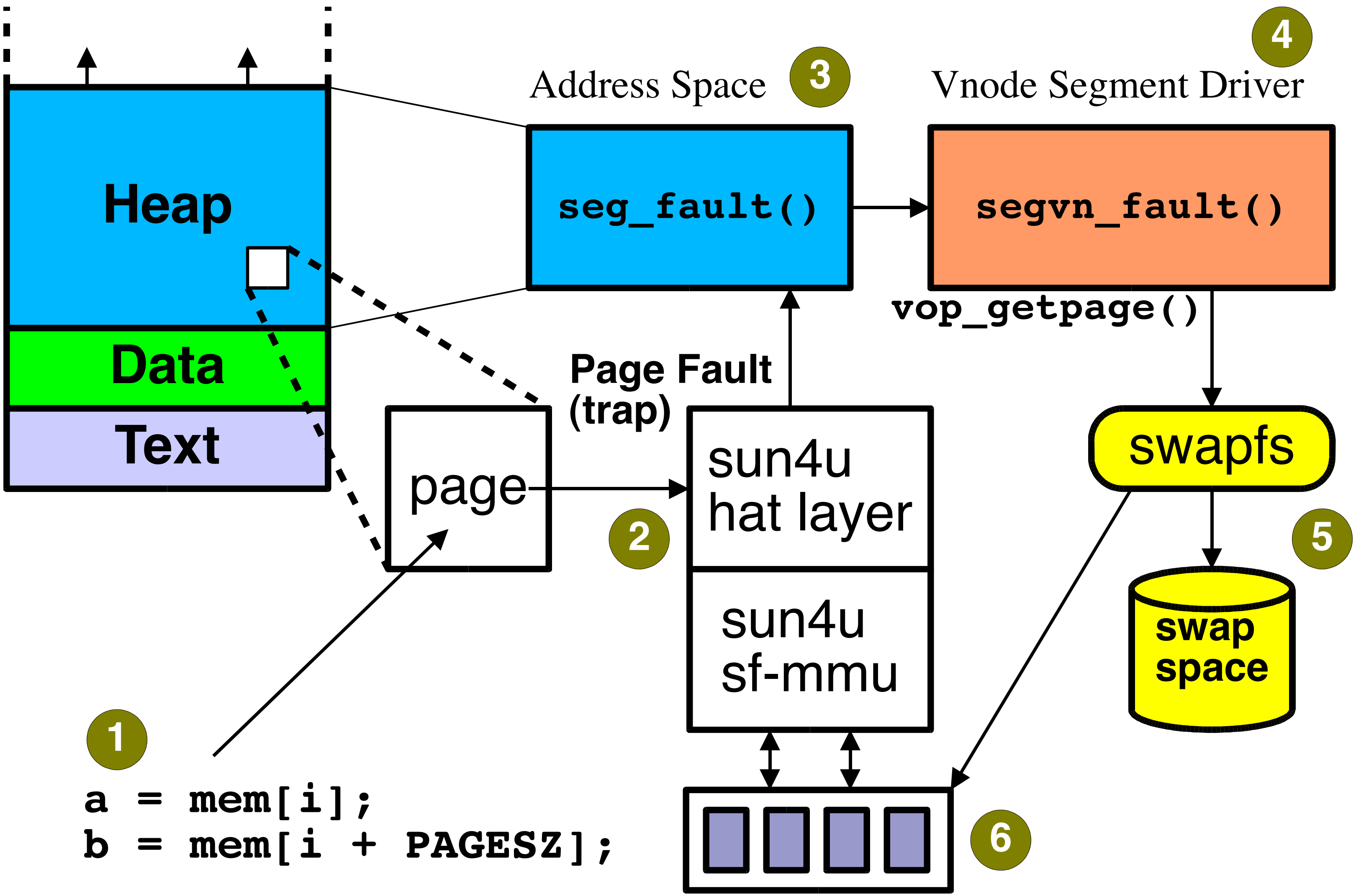
# Address Space Management

- Duplication; **`fork() -> as_dup()`**
- Destruction; **`exit()`**
- Creation of new segments
- Removal of segments
- Page protection (read, write, executable)
- Page Fault routing
- Page Locking
- Watchpoints

# Page Faults

- MMU-generated exception:
- Major Page Fault:
  - Failed access to VM location, in a segment
  - Page does not exist in physical memory
  - New page is created or copied from swap
  - If addr not in a valid segment (SIG-SEGV)
- Minor Page Fault:
  - Failed access to VM location, in a segment
  - Page is in memory, but no MMU translation
- Page Protection Fault:
  - An access that violates segment protection

# Page Fault Example:




```
a = mem[i];
b = mem[i + PAGESZ];
```

# Vmstat -p

swap = free and unreserved swap in KBytes
free = free memory measured in pages

re = kilobytes reclaimed from cache/free list
mf = minor faults - the page was in memory but was not mapped
fr = kilobytes that have been destroyed or freed
de = kilobytes freed after writes
sr = kilobytes scanned / second

executable pages: kilobytes in - out - freed

anonymous pages: kilobytes in - out – freed

file system pages: kilobytes in - out - freed

```
# vmstat -p 5 5
```

| memory | | page | | | | | executable | | | anonymous | | | filesystem | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| swap | free | re | mf | fr | de | sr | epi | epo | epf | api | apo | apf | fpi | fpo | fpf |
| ... | | | | | | | | | | | | | | | |
| 46715224 | 891296 | 24 | 350 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 27 | 0 | 0 |
| 46304792 | 897312 | 151 | 761 | 25 | 0 | 0 | 17 | 0 | 0 | 1 | 0 | 0 | 280 | 25 | 25 |
| 45886168 | 899808 | 118 | 339 | 1 | 0 | 0 | 3 | 0 | 0 | 1 | 0 | 0 | 641 | 1 | 1 |
| 46723376 | 899440 | 29 | 197 | 0 | 0 | 0 | 0 | 0 | 0 | 40 | 0 | 0 | 60 | 0 | 0 |

# Examining paging with dtrace VM Providor

- The dtrace VM provider provides a probe for each VM statistic
- We can observe all VM statistics via kstat:

```
$ kstat -n vm
module: cpu                                instance: 0
name:   vm                                 class:    misc
        anonfree                           0
        anonpgin                           0
        anonpgout                          0
        as_fault                           3180528
        cow_fault                          37280
        crtime                             463.343064
        dfree                              0
        execfree                           0
        execpgin                           442
        execpgout                          0
        fsfree                             0
        fspgin                             2103
        fspgout                            0
        hat_fault                          0
        kernel_asflt                       0
        maj_fault                          912
```

# Examining paging with dtrace

- Suppose one were to see the following output from vmstat(1M):

```
kthr memory page disk faults cpu
r b w swap    free   re   mf   pi po fr de sr  cd s0s1 s2  in   sy   cs us sy id
0 1 0 1341844 836720 26  311 1644  0  0  0  0 216 0  0  0 797  817  697 9  10 81
0 1 0 1341344 835300 238 934 1576  0  0  0  0 194 0  0  0 750 2795  791 7  14 79
0 1 0 1340764 833668 24  165 1149  0  0  0  0 133 0  0  0 637  813  547 5  4  91
0 1 0 1340420 833024 24  394 1002  0  0  0  0 130 0  0  0 621 2284  653 14 7  79
0 1 0 1340068 831520 14  202  380  0  0  0  0  59 0  0  0 482 5688 1434 25 7  68
```

- The pi column in the above output denotes the number of pages paged in. The vminfo provider makes it easy to learn more about the source of these page-ins:

```
dtrace -n pgin {@[execname] = count()}
dtrace: description ÕpginÕ matched 1 probe
^C
xterm 1
ksh 1
ls 2
lpstat 7
sh 17
soffice 39
javaldx 103
soffice.bin 3065
```

# Examining paging with dtrace

- From the above, we can see that a process associated with the StarOffice Office Suite, soffice.bin, is reponsible for most of the page-ins.
- To get a better picture of soffice.bin in terms of VM behavior, we may wish to enable all vminfo probes.
- In the following example, we run dtrace(1M) while launching StarOffice:

```
dtrace -P vminfo/execname == "soffice.bin"/{@[probename] = count()}
dtrace: description vminfo matched 42 probes
^C
pgout 16
anonfree 16
anonpgout 16
pgpgout 16
dfree 16
execpgin 80
prot_fault 85
maj_fault 88
pgin 90
pgpgin 90
cow_fault 859
zfod 1619
pgfrec 8811
pgrec 8827
as_fault 9495
```

# Examining paging with dtrace

- To further drill down on some of the VM behavior of StarOffice during startup, we could write the following D script:

```
vminfo:::maj_fault,
vminfo:::zfod,
vminfo:::as_fault
/execname == "soffice.bin" && start == 0/
{
  /*
   * This is the first time that a vminfo probe has been hit; record
   * our initial timestamp.
   */
  start = timestamp;
}
vminfo:::maj_fault,
vminfo:::zfod,
vminfo:::as_fault
/execname == "soffice.bin"/
{
  /*
   * Aggregate on the probename, and lquantize() the number of seconds
   * since our initial timestamp. (There are 1,000,000,000 nanoseconds
   * in a second.) We assume that the script will be terminated before
   * 60 seconds elapses.
   */
  @[probename] = lquantize((timestamp - start) / 1000000000, 0, 60);
}
```

# Examining paging with dtrace

```
# dtrace -s ./soffice.d
dtrace: script Õ./soffice.dÕ matched 10 probes
^C
maj_fault
value ------------- Distribution ------------- count
7    |                                          0
8    | @@@@@@@@@                                88
9    | @@@@@@@@@@@@@@@@@@@@                     194
10   | @                                        18
11   |                                          0
12   |                                          0
13   |                                          2
14   |                                          0
15   |                                          1
16   | @@@@@@@@                                 82
17   |                                          0
18   |                                          0
19   |                                          2
20   |                                          0
```

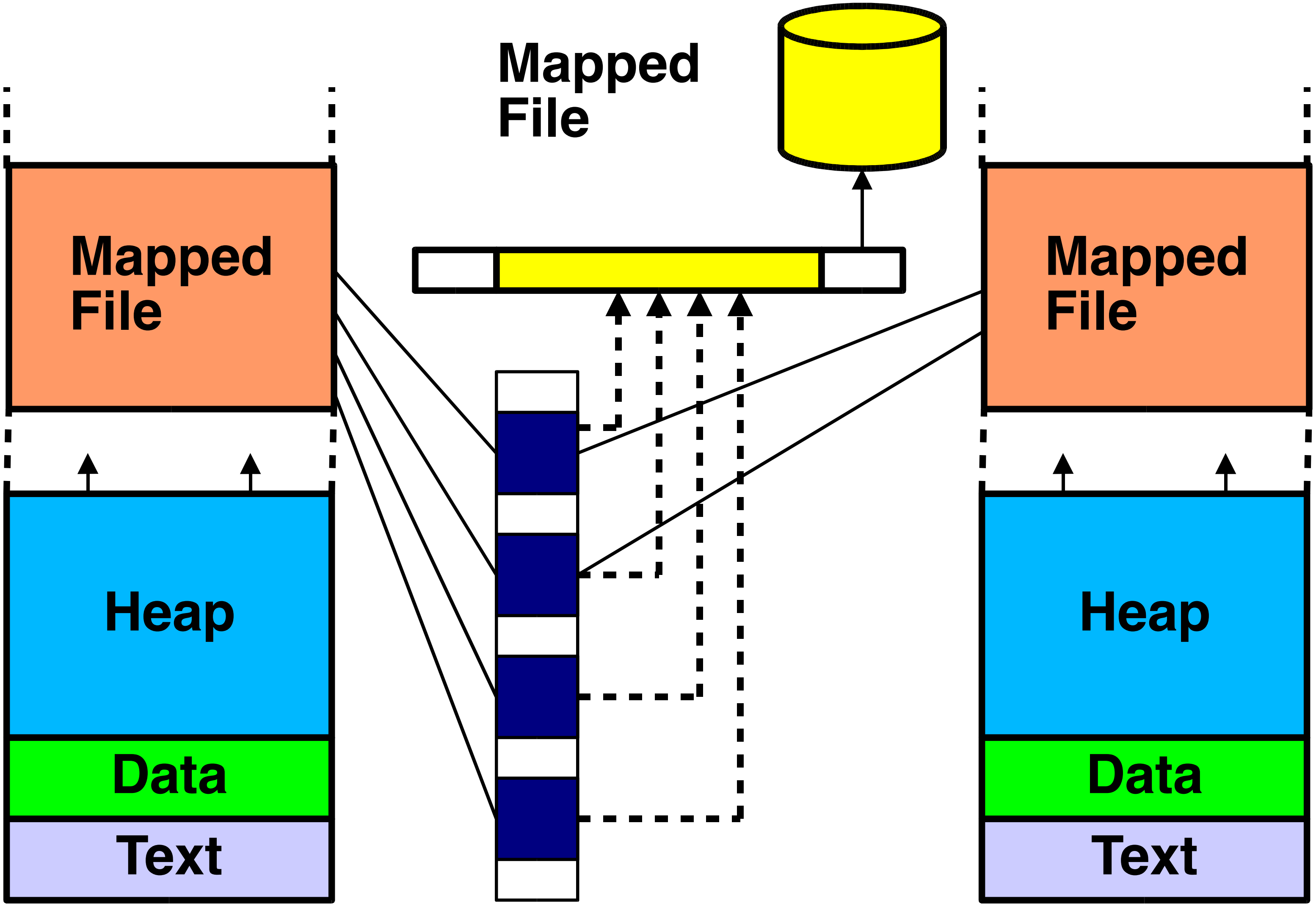
# Examining paging with dtrace

```
Zfod
value ------------- Distribution ------------- count
< 0 |                                          0
0   |@@@@@@@                                   525
1   |@@@@@@@@                                  605
2   |@@                                        208
3   |@@@                                       280
4   |                                          4
5   |                                          0
6   |                                          0
7   |                                          0
8   |                                          44
9   |@@                                        161
10  |                                          2
11  |                                          0
12  |                                          0
13  |                                          4
14  |                                          0
15  |                                          29
16  |@@@@@@@@@@@@@@                            1048
17  |                                          24
18  |                                          0
19  |                                          0
20  |                                          1
21  |                                          0
22  |                                          3
23  |                                          0
```

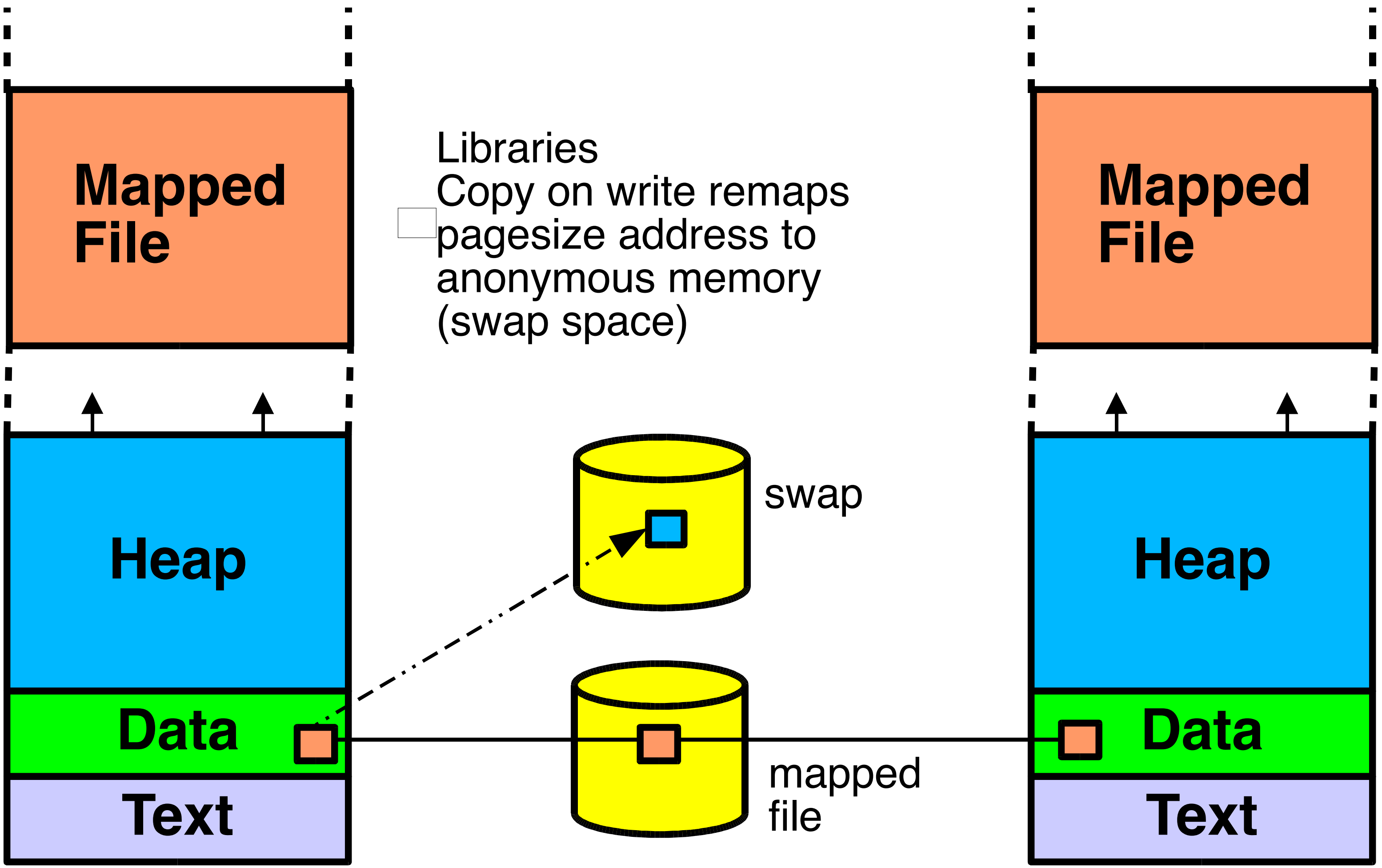
# Examining paging with dtrace

```
as_fault
value ------------- Distribution ------------- count
< 0 |                                           0
0   |@@@@@@@@@@@@@                              4139
1   |@@@@@@@                                    2249
2   |@@@@@@@                                    2402
3   |@                                          594
4   |                                           56
5   |                                           0
6   |                                           0
7   |                                           0
8   |                                           189
9   |@@                                         929
10  |                                           39
11  |                                           0
12  |                                           0
13  |                                           6
14  |                                           0
15  |                                           297
16  |@@@@                                       1349
17  |                                           24
18  |                                           0
19  |                                           21
20  |                                           1
21  |                                           0
22  |                                           92
23  |                                           0
```

# Shared Mapped File



# Copy-on-write



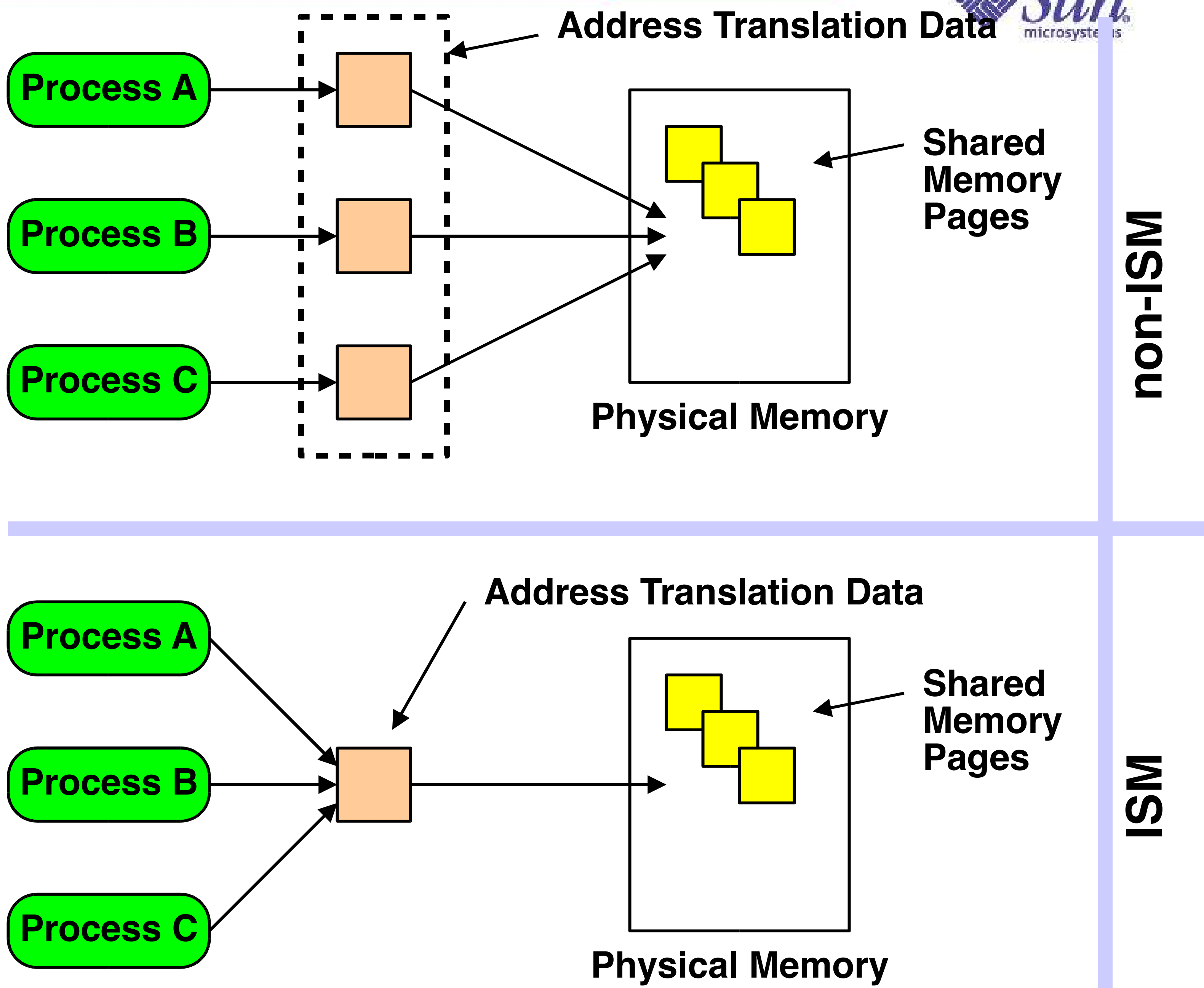
# Anonymous Memory

- Pages not "directly" backed by a vnode
- Heap, Stack and Copy-On-Write pages
- Pages are reserved when "requested"
- Pages allocated when "touched"
- Anon layer:
  - creates slot array for pages
  - Slots point to Anon structs
- Swapfs layer:
  - Pseudo file system for anon layer
  - Provides the backing store

# Intimate Shared Memory

- System V shared memory (ipc) option
- Shared Memory optimization:
  - Additionally share low-level kernel data
  - Reduce redundant mapping info (V-to-P)
- Shared Memory is locked, never paged
  - No swap space is allocated
- Use **`SHM_SHARE_MMU`** flag in **`shmat()`**

# ISM


Address Translation Data
Process A
Process B
Process C
Shared
Memory
Pages
Physical Memory
non-ISM
Address Translation Data
Process A
Process B
Process C
Shared
Memory
Pages
Physical Memory
ISM

# Pmap -x (Solaris 8)

```
Sol8# /usr/proc/bin/pmap -x $$

18084:   csh
 Address   Kbytes Resident Shared Private Permissions       Mapped File
00010000      144      144    136       8 read/exec         csh
00044000       16       16      -      16 read/write/exec   csh
00048000      120      104      -     104 read/write/exec     [ heap ]
FF200000      672      624    600      24 read/exec         libc.so.1
FF2B8000       24       24      -      24 read/write/exec   libc.so.1
FF2BE000        8        8      -       8 read/write/exec   libc.so.1
FF300000       16       16      8       8 read/exec         libc_psr.so.1
FF320000        8        8      -       8 read/exec         libmapmalloc.so.1
FF332000        8        8      -       8 read/write/exec   libmapmalloc.so.1
FF340000        8        8      -       8 read/write/exec     [ anon ]
FF350000      168      112     88      24 read/exec         libcurses.so.1
FF38A000       32       32      -      32 read/write/exec   libcurses.so.1
FF392000        8        8      -       8 read/write/exec   libcurses.so.1
FF3A0000        8        8      -       8 read/exec         libdl.so.1
FF3B0000      136      136    128       8 read/exec         ld.so.1
FF3E2000        8        8      -       8 read/write/exec   ld.so.1
FFBE6000       40       40      -      40 read/write/exec     [ stack ]
--------   ------   ------ ------  ------
total Kb     1424     1304    960     344
```

# Solaris 9 pmap

- New pmap
  - Process private memory usage and memory sharing
    - Old "private" replaced with "Anon"
    - Shared = Resident - Anon
  - Page sizes
  - Swap reservations

# Solaris 9 pmap

```
example$ pmap -x 15492
15492:  ./maps
 Address  Kbytes     RSS    Anon  Locked Mode   Mapped File
00010000       8       8       -       - r-x--  maps
00020000       8       8       8       - rwx--  maps
00022000   20344   16248   16248       - rwx--    [ heap ]
03000000    1024    1024       -       - rw-s-  dev:0,2 ino:4628487
04000000    1024    1024     512       - rw---  dev:0,2 ino:4628487
05000000    1024    1024     512       - rw--R  dev:0,2 ino:4628487
06000000    1024    1024    1024       - rw---    [ anon ]
07000000     512     512     512       - rw--R    [ anon ]
08000000    8192    8192       -    8192 rwxs-    [ dism shmid=0x5]
09000000    8192    4096       -       - rwxs-    [ dism shmid=0x4]
0A000000    8192    8192       -    8192 rwxsR    [ ism shmid=0x2 ]
0B000000    8192    8192       -    8192 rwxsR    [ ism shmid=0x3 ]
FF280000     680     672       -       - r-x--  libc.so.1
FF33A000      32      32      32       - rwx--  libc.so.1
FF390000       8       8       -       - r-x--  libc_psr.so.1
FF3A0000       8       8       -       - r-x--  libdl.so.1
FF3B0000       8       8       8       - rwx--    [ anon ]
FF3C0000     152     152       -       - r-x--  ld.so.1
FF3F6000       8       8       8       - rwx--  ld.so.1
FFBFA000      24      24      24       - rwx--    [ stack ]
-------- ------- ------- ------- -------
total Kb   50464   42264   18888   16384
```

# Process Swap Reservations

```
example$ pmap -S 15492
15492:  ./maps
 Address  Kbytes     Swap Mode   Mapped File
00010000       8        - r-x--  maps
00020000       8        8 rwx--  maps
00022000   20344    20344 rwx--    [ heap ]
03000000    1024        - rw-s-  dev:0,2 ino:4628487
04000000    1024     1024 rw---  dev:0,2 ino:4628487
05000000    1024      512 rw--R  dev:0,2 ino:4628487
06000000    1024     1024 rw---    [ anon ]
07000000     512      512 rw--R    [ anon ]
08000000    8192        - rwxs-    [ dism shmid=0x5]
09000000    8192        - rwxs-    [ dism shmid=0x4]
0A000000    8192        - rwxs-    [ dism shmid=0x2]
0B000000    8192        - rwxsR    [ ism shmid=0x3]
FF280000     680        - r-x--  libc.so.1
FF33A000      32       32 rwx--  libc.so.1
FF390000       8        - r-x--  libc_psr.so.1
FF3A0000       8        - r-x--  libdl.so.1
FF3B0000       8        8 rwx--    [ anon ]
FF3C0000     152        - r-x--  ld.so.1
FF3F6000       8        8 rwx--  ld.so.1
FFBFA000      24       24 rwx--    [ stack ]
-------- ------- -------
total Kb   50464   23496
```

# Unbundled Tools

- MemTool
  - Loadable kernel module + utilities to examine process memory usage and UFS buffer cache usage
  - memps - list files in memory and amount of memory
  - memtool - GUI to list files and also cross reference amount used by each process
  - prtmem, prtswap - displays system memory or swap summary
  - Obtain from memtool-request@devnull.eng.sun.com

# Processes, Threads, Scheduling Classes & The Dispatcher

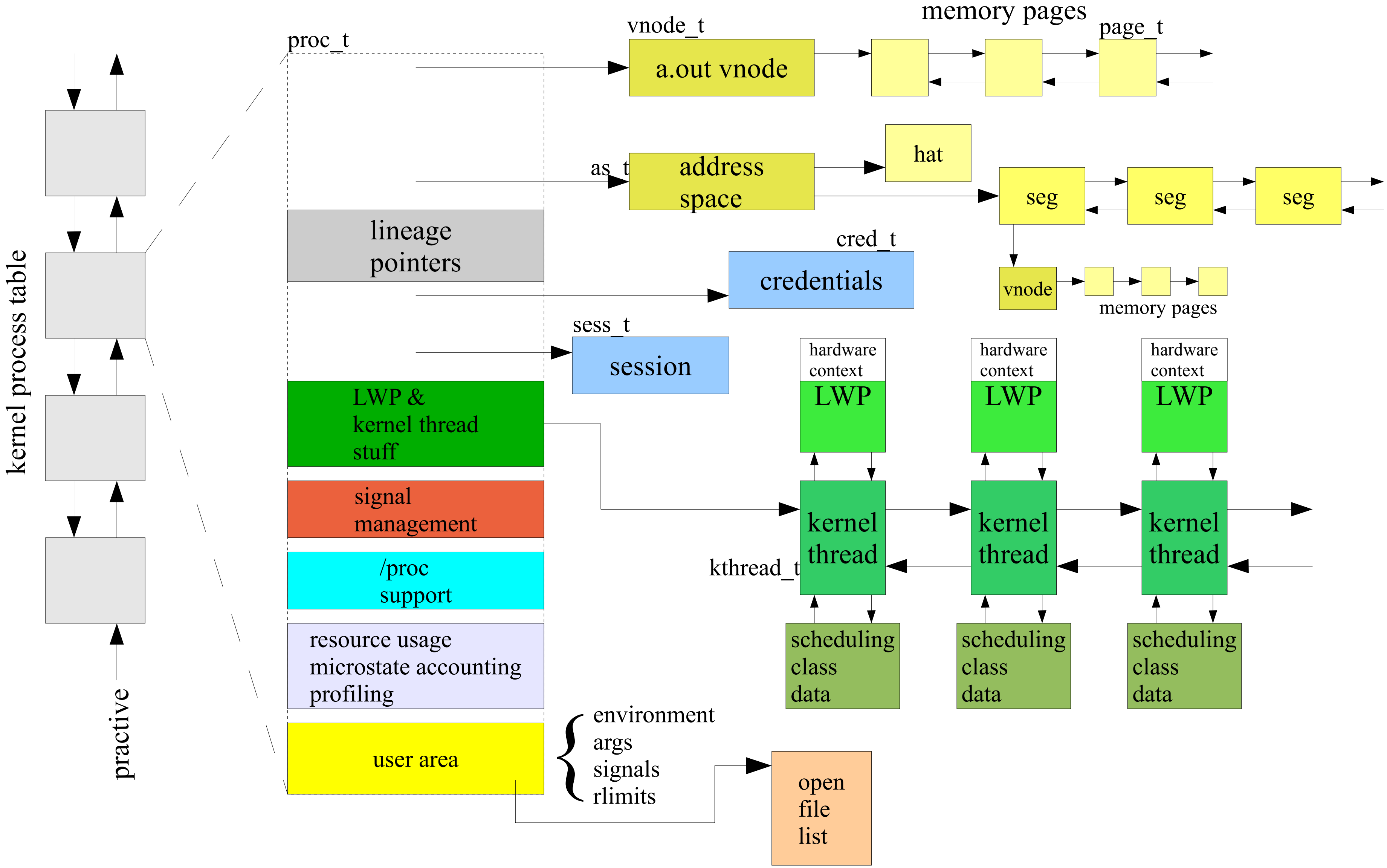
# Process/Threads Glossary

| | |
|---|---|
| Process | The executable form of a program. An Operating System abstraction that encapulates the execution context of a program |
| Thread | An executable entity |
| User Thread | A thread within the address space of a process |
| Kernel Thread | A thread in the address space of the kernel |
| Lightweight Process | LWP – An execution context for a kernel thread |
| Dispatcher | The kernel subsystem that manages queues of runnable kernel threads |
| Scheduling Class | Kernel classes that define the scheduling parameters (e.g. priorities) and algorithms used to multiplex threads onto processors |
| Dispatch Queues | Per-processor sets of queues of runnable threads (run queues) |
| Sleep Queues | Queues of sleeping threads |
| Turnstiles | A special implementation of sleep queues that provide priority inheritance. |

# Solaris Process Model

- Solaris implements a multithreaded process model
  - Kernel threads are scheduled/executed
  - LWPs allow for each thread to execute system calls
  - Every kernel thread has an associated LWP
  - A non-threaded process has 1 kernel thread/LWP
  - A threaded process will have multiple kernel threads
  - All the threads in a process share all of the process context
    - Address space
    - Open files
    - Credentials
    - Signal dispositions
  - Each thread has its own stack

# Solaris Process



# Kernel Process Table

- Linked list of all processes (proc structures)
- kmem_cache allocator dynamically allocates space needed for new proc structures
  - Up to v.v_proc

```
borntorun> kstat -n var
module: unix                               instance: 0
name:   var                                class:    misc
     crtime                             61.041156087
     snaptime                           113918.894449089
     v_autoup                           30
     v_buf                              100
     v_bufhwm                           20312
     [snip]
     v_maxsyspri                        99
     v_maxup                            15877
     v_maxupttl                         15877
     v_nglobpris                        110
     v_pbuf                             0
     v_proc                             15882
     v_sptmap                           0
```

```
# mdb -k
Loading modules: [ unix krtld genunix ufs_log ip nfs random ptm ipc ]
> max_nprocs/D
max_nprocs:
max_nprocs:     15882
>
```

# System-wide Process View - ps(1)

```
F S      UID    PID   PPID   C  PRI NI     ADDR      SZ     WCHAN    STIME TTY        TIME CMD
0 S     root    824    386   0   40 020       ?     252         ?   Sep 06 console  0:00 /usr/lib/saf/ttymon -g -h
-p mcdoug
0 S     root    823    386   0   40 20       ?     242         ?   Sep 06 ?          0:00 /usr/lib/saf/sac -t 300
0 S   nobody   1718    716   0   40 20       ?     834         ?   Sep 07 ?          0:35 /usr/apache/bin/httpd
0 S     root    591    374   0   40 20       ?     478         ?   Sep 06 ?          0:00 /
usr/lib/autofs/automountd
0 S     root    386    374   0   40 20       ?     262         ?   Sep 06 ?          0:01 init
1 S     root    374    374   0    0 SY       ?       0         ?   Sep 06 ?          0:00 zsched
0 S   daemon    490    374   0   40 20       ?     291         ?   Sep 06 ?          0:00 /usr/sbin/rpcbind
0 S   daemon    435    374   0   40 20       ?     450         ?   Sep 06 ?          0:00 /usr/lib/crypto/kcfd
0 S     root    603    374   0   40 20       ?     475         ?   Sep 06 ?          0:12 /usr/sbin/nscd
0 S     root    580    374   0   40 20       ?     448         ?   Sep 06 ?          0:02 /usr/sbin/syslogd
0 S     root    601    374   0   40 20       ?     313         ?   Sep 06 ?          0:00 /usr/sbin/cron
0 S   daemon    548    374   0   40 20       ?     319         ?   Sep 06 ?          0:00 /usr/lib/nfs/statd
0 S   daemon    550    374   0   40 20       ?     280         ?   Sep 06 ?          0:00 /usr/lib/nfs/lockd
0 S     root    611    374   0   40 20       ?     329         ?   Sep 06 ?          0:00 /usr/sbin/inetd -s
0 S     root    649    374   0   40 20       ?     152         ?   Sep 06 ?          0:00 /usr/lib/utmpd
0 S   nobody    778    716   0   40 20       ?     835         ?   Sep 06 ?          0:26 /usr/apache/bin/httpd
0 S     root    678    374   0   40 20       ?     612         ?   Sep 06 ?          0:00 /usr/dt/bin/dtlogin
-daemon
```

# System-wide Process View - prstat(1)

```
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 26292 root     5368K 3080K run     24    0   0:00:00 1.5% pkginstall/1
 26188 rmc      4880K 4512K cpu0    49    0   0:00:00 0.6% prstat/1
   202 root     3304K 1800K sleep   59    0   0:00:07 0.3% nscd/24
 23078 root       20M   14M sleep   59    0   0:00:56 0.2% lupi_zones/1
 23860 root     5104K 2328K sleep   59    0   0:00:01 0.1% sshd/1
 23001 root     5136K 2184K sleep   59    0   0:00:03 0.1% sshd/1
 24866 root     5136K 2160K sleep   59    0   0:00:00 0.1% sshd/1
 25946 rmc      2936K 2176K sleep   59    0   0:00:00 0.1% ssh/1
   830 root     2472K  696K sleep   59    0   0:18:53 0.1% mibiisa/7
 25947 root     5160K 3000K sleep   59    0   0:00:00 0.1% sshd/1
   340 root     2504K  680K sleep   59    0   0:19:13 0.0% mibiisa/7
   829 root     2488K  696K sleep   59    0   0:18:48 0.0% mibiisa/7
   387 root     2096K  376K sleep   59    0   0:00:00 0.0% init/1
 25955 rmc      1344K 1024K sleep   59    0   0:00:00 0.0% ksh/1
   815 root     2016K    8K sleep   59    0   0:00:00 0.0% ttymon/1
   365 root     4760K  128K sleep   59    0   0:00:00 0.0% zoneadmd/4
   364 root     4776K  128K sleep   59    0   0:00:00 0.0% zoneadmd/4
   374 root        0K    0K sleep   60    -   0:00:00 0.0% zsched/1
   361 root     2016K    8K sleep   59    0   0:00:00 0.0% ttymon/1
   349 root     8600K  616K sleep   59    0   0:00:20 0.0% snmpd/1
   386 root     2096K  360K sleep   59    0   0:00:00 0.0% init/1
   345 root     3160K  496K sleep   59    0   0:00:00 0.0% sshd/1
   591 root     3824K  184K sleep   59    0   0:00:00 0.0% automountd/2
   373 root        0K    0K sleep   60    -   0:00:00 0.0% zsched/1
  1718 nobody   6672K 2056K sleep   59    0   0:00:35 0.0% httpd/1
   322 root     3112K   16K sleep   59    0   0:00:00 0.0% dmispd/1
   328 root     2728K   40K sleep   59    0   0:00:01 0.0% vold/3
   488 daemon   2328K   16K sleep   59    0   0:00:00 0.0% rpcbind/1
   312 root     4912K   24K sleep   59    0   0:00:00 0.0% dtlogin/1
   250 root     4760K  704K sleep   59    0   0:00:16 0.0% sendmail/1
   246 root     1888K    0K sleep   59    0   0:00:00 0.0% smcboot/1
   823 root     1936K  224K sleep   59    0   0:00:00 0.0% sac/1
   242 root     1896K    8K sleep   59    0   0:00:00 0.0% smcboot/1
   248 smmsp    4736K  696K sleep   59    0   0:00:08 0.0% sendmail/1
   245 root     1888K    0K sleep   59    0   0:00:00 0.0% smcboot/1
   824 root     2016K    8K sleep   59    0   0:00:00 0.0% ttymon/1
   204 root     2752K  536K sleep   59    0   0:00:00 0.0% inetd/1
   220 root     1568K    8K sleep   59    0   0:00:00 0.0% powerd/3
   313 root     2336K  216K sleep   59    0   0:00:00 0.0% snmpdx/1
   184 root     4312K  872K sleep   59    0   0:00:01 0.0% syslogd/13
   162 daemon   2240K   16K sleep   60  -20   0:00:00 0.0% lockd/2
Total: 126 processes, 311 lwps, load averages: 0.48, 0.48, 0.41
```

# The Life Of A Process

- Process creation
  - fork(2) system call creates all processes
    - SIDL state
  - exec(2) overlays newly created process with executable image
- State Transitions
  - Typically runnable (SRUN), running (SONPROC) or sleeping (aka blocked, SSLEEP)
  - Maybe stopped (debugger) SSTOP
- Termination
  - SZOMB state
  - implicit or explicit exit(), signal (kill), fatal error

# Process Creation

- Traditional UNIX fork/exec model
  - fork(2) - replicate the entire process, including all threads
  - fork1(2) - replicate the process, only the calling thread
  - vfork(2) - replicate the process, but do not dup the address space
    - The new child borrows the parents address space, until exec()

```
main(int argc, char *argv[])
{
     pid_t pid;
     pid = fork();
     if (pid == 0)  /* in the child */
               exec();
     else if (pid > 0)   /* in the parent */
               wait();
     else
               fork failed
}
```

# fork(2) in Solaris 10

- Solaris 10 unified the process model
  - libthread merged with libc
  - threaded and non-threaded processes look the same
- fork(2) now replicates only the calling thread
  - Previously, fork1(2) needed to be called to do this
  - Linking with -lpthread in previous releases also resulted in fork1(2) behaviour
- forkall(2) added for applications that require a fork to replicate all the threads in the process

# exec(2) – Load a new process image

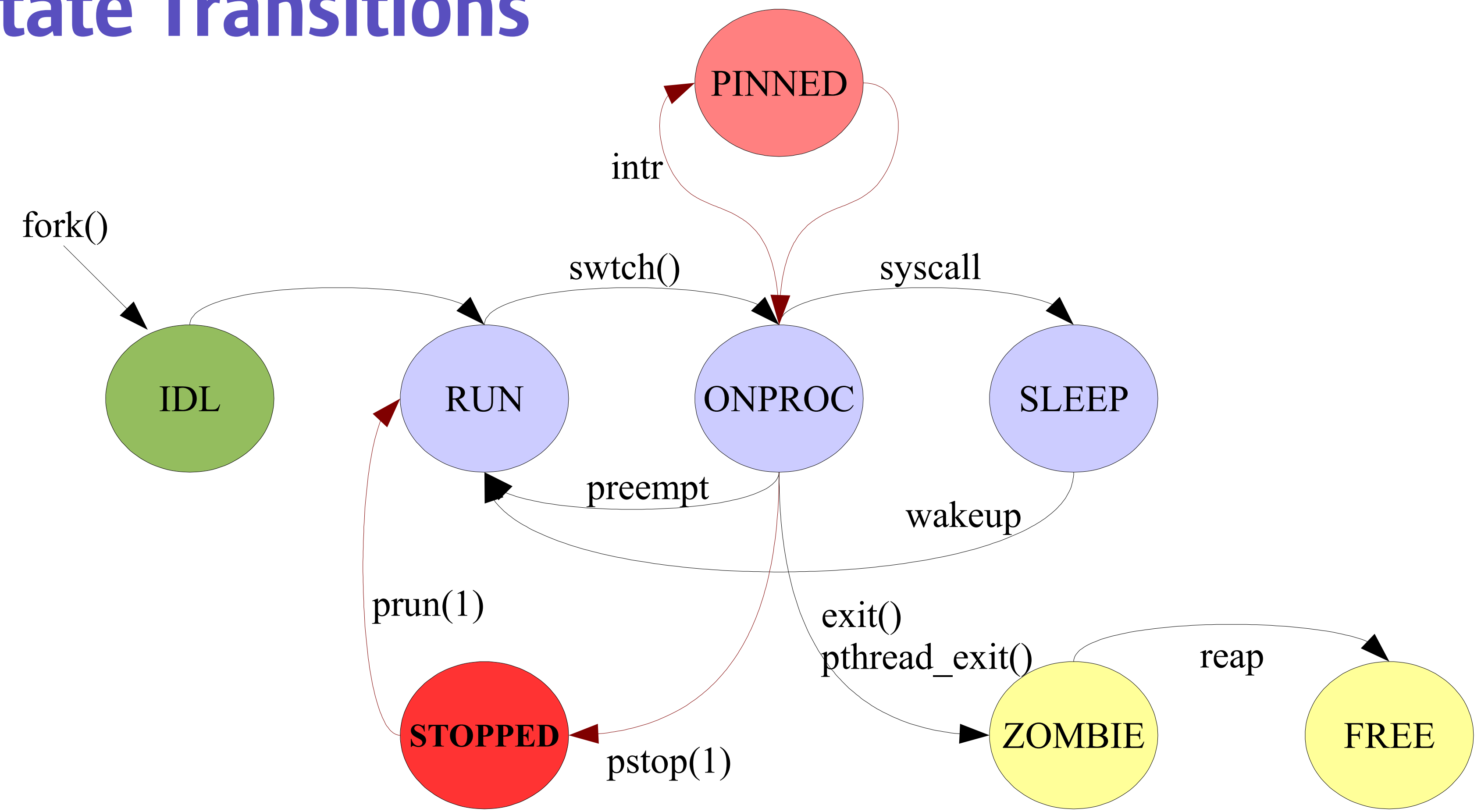
- Most fork(2) calls are followed by an exec(2)
- exec – execute a new file
- exec overlays the process image with a new process constructed from the binary file passed as an arg to exec(2)
- The exec'd process inherits much of the caller's state:
  - nice value, scheduling class, priority, PID, PPID, GID, task ID, project ID, session membership, real UID & GID, current working directory, resource limits, processor binding, times, etc, ...

# Process / Thread States

- It's really kernel threads that change state
- Kernel thread creation is not flagged as a distinct state
  - Initial state is TS_RUN
- Kernel threads are TS_FREE when the process, or LWP/kthread, terminates

| Process State | Kernel Thread State |
|---|---|
| SIDL | |
| SRUN | TS_RUN |
| SONPROC | TS_ONPROC |
| SSLEEP | TS_SLEEP |
| SSTOP | TS_STOPPED |
| SZOMB | TS_ZOMB |
| | TS_FREE |

# State Transitions



# Watching Process States

```
  PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 27946 root     4880K 4520K cpu0    59    0   0:00:00 0.7% prstat/1
 28010 root     4928K 2584K run     29    0   0:00:00 0.7% pkginstall/1
 23078 root       20M   14M sleep   59    0   0:00:57 0.3% lupi_zones/1
 25947 root     5160K 2976K sleep   59    0   0:00:04 0.3% sshd/1
 24866 root     5136K 2136K sleep   59    0   0:00:01 0.2% sshd/1
   202 root     3304K 1800K sleep   59    0   0:00:09 0.2% nscd/24
 23001 root     5136K 2176K sleep   59    0   0:00:04 0.1% sshd/1
 23860 root     5248K 2392K sleep   59    0   0:00:05 0.1% sshd/1
 25946 rmc      3008K 2184K sleep   59    0   0:00:02 0.1% ssh/1
 25690 root     1240K  928K sleep   59    0   0:00:00 0.1% sh/1
   830 root     2472K  696K sleep   59    0   0:18:53 0.1% mibiisa/7
   349 root     8600K  768K sleep   59    0   0:00:20 0.0% snmpd/1
   340 root     2504K  680K sleep   59    0   0:19:14 0.0% mibiisa/7
   829 root     2488K  696K sleep   59    0   0:18:48 0.0% mibiisa/7
 27328 root     1240K  928K sleep   59    0   0:00:00 0.0% sh/1
   490 daemon   2328K   16K sleep   59    0   0:00:00 0.0% rpcbind/1
   815 root     2016K    8K sleep   59    0   0:00:00 0.0% ttymon/1
   365 root     4760K  128K sleep   59    0   0:00:00 0.0% zoneadmd/4
   364 root     4776K  128K sleep   59    0   0:00:00 0.0% zoneadmd/4
   374 root        0K    0K sleep   60    -   0:00:00 0.0% zsched/1
   361 root     2016K    8K sleep   59    0   0:00:00 0.0% ttymon/1
   386 root     2096K  360K sleep   59    0   0:00:00 0.0% init/1
   387 root     2096K  376K sleep   59    0   0:00:00 0.0% init/1
   345 root     3160K  480K sleep   59    0   0:00:00 0.0% sshd/1
   591 root     3824K  184K sleep   59    0   0:00:00 0.0% automountd/2
   373 root        0K    0K sleep   60    -   0:00:00 0.0% zsched/1
  1718 nobody   6672K 2032K sleep   59    0   0:00:35 0.0% httpd/1
   322 root     3112K   16K sleep   59    0   0:00:00 0.0% dmispd/1
   328 root     2728K   40K sleep   59    0   0:00:01 0.0% vold/3
   488 daemon   2328K   16K sleep   59    0   0:00:00 0.0% rpcbind/1
   312 root     4912K   24K sleep   59    0   0:00:00 0.0% dtlogin/1
   250 root     4760K  696K sleep   59    0   0:00:16 0.0% sendmail/1
   246 root     1888K    0K sleep   59    0   0:00:00 0.0% smcboot/1
   823 root     1936K  224K sleep   59    0   0:00:00 0.0% sac/1
   242 root     1896K    8K sleep   59    0   0:00:00 0.0% smcboot/1
   248 smmsp    4736K  680K sleep   59    0   0:00:08 0.0% sendmail/1
   245 root     1888K    0K sleep   59    0   0:00:00 0.0% smcboot/1
   824 root     2016K    8K sleep   59    0   0:00:00 0.0% ttymon/1
   204 root     2752K  520K sleep   59    0   0:00:00 0.0% inetd/1
   220 root     1568K    8K sleep   59    0   0:00:00 0.0% powerd/3
   313 root     2336K  216K sleep   59    0   0:00:00 0.0% snmpdx/1
Total: 127 processes, 312 lwps, load averages: 0.62, 0.62, 0.53
```

# Microstates

- Fine-grained state tracking for processes/threads
  - Off by default in Solaris 8 and Solaris 9
  - On by default in Solaris 10
- Can be enabled per-process via /proc
- prstat -m reports microstates
  - As a percentage of time for the sampling period
    - USR – user mode
    - SYS - kernel mode
    - TRP – trap handling
    - TFL – text page faults
    - DFL – data page faults
    - LCK – user lock wait
    - SLP - sleep
    - LAT – waiting for a processor (sitting on a run queue)

# prstat – process microstates

```
sol8$ prstat -m
   PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/NLWP
   739 root     0.3 0.3 0.0 0.0 0.0 0.0  99 0.0 126   3 345   5 Xsun/1
 15611 root     0.1 0.3 0.0 0.0 0.0 0.0 100 0.0  23   0 381   0 prstat/1
  1125 tlc      0.3 0.0 0.0 0.0 0.0 0.0 100 0.0  29   0 116   0 gnome-panel/1
 15553 rmc      0.1 0.2 0.0 0.0 0.0 0.0 100 0.0  24   0 381   0 prstat/1
  5591 tlc      0.1 0.0 0.0 0.0 0.0  33  66 0.0 206   0  1K   0 mozilla-bin/6
  1121 tlc      0.0 0.0 0.0 0.0 0.0 0.0 100 0.1  50   0 230   0 metacity/1
  2107 rmc      0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  25   0  36   0 gnome-termin/1
   478 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  17   0  14   0 squid/1
   798 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  11   0  23   0 Xsun/1
  1145 tlc      0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  25   1  34   0 mixer_applet/1
  1141 rmc      0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  25   0  32   0 mixer_applet/1
  1119 tlc      0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   5   0  40   0 gnome-smprox/1
  1127 tlc      0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   7   0  29   0 nautilus/3
  1105 rmc      0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   7   0  27   0 nautilus/3
   713 root     0.0 0.0 0.0 0.0 0.0  85  15 0.0   2   0 100   0 mibiisa/7
   174 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   5   0  50   5 ipmon/1
  1055 tlc      0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   5   0  30   0 dsdm/1
Total: 163 processes, 275 lwps, load averages: 0.07, 0.07, 0.07
```

# prstat – user summary

```
sol8$ prstat -t
NPROC USERNAME  SIZE   RSS MEMORY      TIME  CPU
   128 root      446M  333M   1.4%  47:14:23  11%
     2 measter  6600K 5016K   0.0%   0:00:07 0.2%
     1 clamb    9152K 8344K   0.0%   0:02:14 0.1%
     2 rmc      7192K 6440K   0.0%   0:00:00 0.1%
     1 bricker  5776K 4952K   0.0%   0:00:20 0.1%
     2 asd        10M 8696K   0.0%   0:00:01 0.1%
     1 fredz    7760K 6944K   0.0%   0:00:05 0.1%
     2 jenks    8576K 6904K   0.0%   0:00:01 0.1%
     1 muffin     15M   14M   0.1%   0:01:26 0.1%
     1 dte      3800K 3016K   0.0%   0:00:04 0.0%
     2 adjg     8672K 7040K   0.0%   0:00:03 0.0%
     3 msw        14M   10M   0.0%   0:00:00 0.0%
     1 welza    4032K 3248K   0.0%   0:00:29 0.0%
     2 kimc     7848K 6344K   0.0%   0:00:25 0.0%
     4 jcmartin   13M 9904K   0.0%   0:00:03 0.0%
     1 rascal     17M   16M   0.1%   0:02:11 0.0%
     1 rab      3288K 2632K   0.0%   0:02:11 0.0%
     1 gjmurphy 3232K 2392K   0.0%   0:00:00 0.0%
     1 ktheisen   15M   14M   0.1%   0:01:16 0.0%
     1 nagendra 3232K 2400K   0.0%   0:00:00 0.0%
     2 ayong    8320K 6832K   0.0%   0:00:02 0.0%
Total: 711 processes, 902 lwps, load averages: 3.84, 4.30, 4.37
```

# Solaris 8 ptools

```
/usr/bin/pflags [ -r ]  [ pid | core ]  ...
/usr/bin/pcred [ pid | core ]  ...
/usr/bin/pmap [ -rxlF ]  [ pid | core ]  ...
/usr/bin/pldd [ -F ]  [ pid | core ]  ...
/usr/bin/psig pid ...
/usr/bin/pstack [ -F ]  [ pid | core ]  ...
/usr/bin/pfiles [ -F ]  pid ...
/usr/bin/pwdx [ -F ]  pid ...
/usr/bin/pstop pid ...
/usr/bin/prun pid ..
/usr/bin/pwait [ -v ]  pid ...
/usr/bin/ptree [ -a ]  [  [ pid | user ]  ... ]
/usr/bin/ptime command  [ arg ... ]
/usr/bin/pgrep  [  -flnvx ]   [  -d delim   ]   [ -P ppidlist  ]
[ -g pgrplist ]  [ -s sidlist ]  [ -u euidlist ]  [ -U uidlist ]
[ -G gidlist ]  [ -J projidlist ]  [  -t termlist   ]   [ -T
taskidlist ]  [ pattern ]
/usr/bin/pkill [ -signal ]   [ -fnvx ]  [ -P ppidlist ]  [ -g
pgrplist ]   [  -s sidlist   ]   [ -u euidlist  ]  [ -U uidlist]
[ -G gidlist  ]   [  -J projidlist ]   [ -t termlist  ]   [-T
taskidlist ]  [ pattern ]
```

# Solaris 9 / 10 ptools

```
/usr/bin/pflags [-r] [pid | core] ...
/usr/bin/pcred [pid | core] ...
/usr/bin/pldd [-F] [pid | core] ...
/usr/bin/psig [-n] pid...
/usr/bin/pstack [-F] [pid | core] ...
/usr/bin/pfiles [-F] pid...
/usr/bin/pwdx [-F] pid...
/usr/bin/pstop pid...
/usr/bin/prun pid...
/usr/bin/pwait [-v] pid...
/usr/bin/ptree [-a] [pid | user] ...
/usr/bin/ptime command [arg...]
/usr/bin/pmap -[xS] [-rslF] [pid | core] ...
/usr/bin/pgrep  [-flvx]  [-n  |  -o]   [-d delim][-P ppidlist]  [-
g pgrplist]   [-s sidlist]    [-u euidlist]   [-U uidlist]  [-
G gidlist] [-J projidlist]  [-t termlist]   [-T taskidlist]
[pattern]
/usr/bin/pkill  [-signal][-fvx][-n  | -o][-P ppidlist]  [-
g pgrplist]   [-s sidlist]    [-u euidlist]   [-U uidlist]  [-
G gidlist] [-J projidlist]  [-t termlist]   [-T taskidlist]
[pattern]
/usr/bin/plimit [-km] pid...
{-cdfnstv} soft,hard... pid...
/usr/bin/ppgsz  [-F] -o option[,option]  cmd | -p pid...
/usr/bin/prctl [-t [basic | privileged | system] ] [ -e | -d  action]
[-rx] [ -n name [-v value]] [-i idtype] [id...]
/usr/bin/preap [-F] pid
/usr/bin/pargs [-aceFx] [pid | core] ...
```

# pflags, pcred, pldd

```
sol8# pflags $$
482764: -ksh
  data model = _ILP32  flags = PR_ORPHAN
  /1:  flags = PR_PCINVAL|PR_ASLEEP [ waitid(0x7,0x0,0xffbff938,0x7) ]
```

```
sol8$ pcred $$
482764: e/r/suid=36413  e/r/sgid=10
  groups: 10 10512 570
```

```
sol8$ pldd $$
482764: -ksh
/usr/lib/libsocket.so.1
/usr/lib/libnsl.so.1
/usr/lib/libc.so.1
/usr/lib/libdl.so.1
/usr/lib/libmp.so.2
```

# psig

```
sol8$ psig $$
15481: -zsh
HUP  caught 0
INT  blocked,caught 0
QUIT blocked,ignored
ILL  blocked,default
TRAP blocked,default
ABRT blocked,default
EMT  blocked,default
FPE  blocked,default
KILL default
BUS  blocked,default
SEGV blocked,default
SYS  blocked,default
PIPE blocked,default
ALRM blocked,caught 0
TERM blocked,ignored
USR1 blocked,default
USR2 blocked,default
CLD  caught 0
PWR  blocked,default
WINCH  blocked,caught 0
URG  blocked,default
POLL blocked,default
STOP default
```

# pstack

```
sol8$ pstack 5591
5591:  /usr/local/mozilla/mozilla-bin
-----------------  lwp# 1 / thread# 1  --------------------
 fe99a254 poll    (513d530, 4, 18)
 fe8dda58 poll    (513d530, fe8f75a8, 18, 4, 513d530, ffbeed00) + 5c
 fec38414 g_main_poll (18, 0, 0, 27c730, 0, 0) + 30c
 fec37608 g_main_iterate (1, 1, 1, ff2a01d4, ff3e2628, fe4761c9) + 7c0
 fec37e6c g_main_run (27c740, 27c740, 1, fe482b30, 0, 0) + fc
 fee67a84 gtk_main (b7a40, fe482874, 27c720, fe49c9c4, 0, 0) + 1bc
 fe482aa4 ???????? (d6490, fe482a6c, d6490, ff179ee4, 0, ffe)
 fe4e5518 ???????? (db010, fe4e5504, db010, fe4e6640, ffbeeed0, 1cf10)
 00019ae8 ???????? (0, ff1c02b0, 5fca8, 1b364, 100d4, 0)
 0001a4cc main    (0, ffbef144, ffbef14c, 5f320, 0, 0) + 160
 00014a38 _start   (0, 0, 0, 0, 0, 0) + 5c
-----------------  lwp# 2 / thread# 2  --------------------
 fe99a254 poll    (fe1afbd0, 2, 88b8)
 fe8dda58 poll    (fe1afbd0, fe840000, 88b8, 2, fe1afbd0, 568) + 5c
 ff0542d4 ???????? (75778, 2, 3567e0, b97de891, 4151f30, 0)
 ff05449c PR_Poll  (75778, 2, 3567e0, 0, 0, 0) + c
 fe652bac ???????? (75708, 80470007, 7570c, fe8f6000, 0, 0)
 ff13b5f0 Main__8nsThreadPv (f12f8, ff13b5c8, 0, 0, 0, 0) + 28
 ff055778 ???????? (f5588, fe840000, 0, 0, 0, 0)
 fe8e4934 _lwp_start (0, 0, 0, 0, 0, 0)
```

# pfiles

```
sol8$ pfiles $$
pfiles $$
15481: -zsh
  Current rlimit: 256 file descriptors
   0: S_IFCHR mode:0620 dev:118,0 ino:459678 uid:36413 gid:7 rdev:24,11
      O_RDWR
   1: S_IFCHR mode:0620 dev:118,0 ino:459678 uid:36413 gid:7 rdev:24,11
  O_RDWR
   2: S_IFCHR mode:0620 dev:118,0 ino:459678 uid:36413 gid:7 rdev:24,11
  O_RDWR
   3: S_IFDOOR mode:0444 dev:250,0 ino:51008 uid:0 gid:0 size:0
  O_RDONLY|O_LARGEFILE FD_CLOEXEC  door to nscd[328]
  10: S_IFCHR mode:0620 dev:118,0 ino:459678 uid:36413 gid:7 rdev:24,11
  O_RDWR|O_LARGEFILE
```

# pwdx, pstop, pwait, ptree

```
sol8$ pwdx $$
15481: /home/rmc

sol8$ pstop $$
[argh!]

sol8$ pwait 23141

sol8$ ptree $$
285   /usr/sbin/inetd -ts
  15554 in.rlogind
    15556 -zsh
  15562 ksh
  15657 ptree 15562
```

# pgrep

```
sol8$ pgrep -u rmc
481
480
478
482
483
484
.....
```

# Tracing

- Trace user signals and system calls - truss
  - Traces by stopping and starting the process
  - Can trace system calls, inline or as a summary
  - Can also trace shared libraries and a.out
- Linker/library interposing/profiling/tracing
  - LD_ environment variables enable link debugging
  - man ld.so.1
  - using the LD_PRELOAD env variable
- Trace Normal Formal (TNF)
  - Kernel and Process Tracing
  - Lock Tracing
- Kernel Tracing
  - lockstat, tnf, kgmon

# Process Tracing – Truss

```
# truss -d dd if=500m of=/dev/null bs=16k count=2k 2>&1 |more
Base time stamp:  925931550.0927  [ Wed May  5 12:12:30 PDT 1999 ]
 0.0000 execve("/usr/bin/dd", 0xFFBEF68C, 0xFFBEF6A4)  argc = 5
 0.0034 open("/dev/zero", O_RDONLY)                    = 3
 0.0039 mmap(0x00000000, 8192, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE, 3, 0) = 0xFF3A0000
 0.0043 open("/usr/lib/libc.so.1", O_RDONLY)           = 4
 0.0047 fstat(4, 0xFFBEF224)                           = 0
 0.0049 mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF390000
 0.0051 mmap(0x00000000, 761856, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF280000
 0.0054 munmap(0xFF324000, 57344)                      = 0
 0.0057 mmap(0xFF332000, 25284, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 663552) = 0xFF332000
 0.0062 close(4)                                       = 0
 0.0065 open("/usr/lib/libdl.so.1", O_RDONLY)          = 4
 0.0068 fstat(4, 0xFFBEF224)                           = 0
 0.0070 mmap(0xFF390000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0) = 0xFF390000
 0.0073 close(4)                                       = 0
 0.0076 open("/usr/platform/SUNW,Ultra-2/lib/libc_psr.so.1", O_RDONLY) = 4
 0.0079 fstat(4, 0xFFBEF004)                           = 0
 0.0082 mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF380000
 0.0084 mmap(0x00000000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF370000
 0.0087 close(4)                                       = 0
 0.0100 close(3)                                       = 0
 0.0103 munmap(0xFF380000, 8192)                       = 0
 0.0110 open64("500m", O_RDONLY)                       = 3
 0.0115 creat64("/dev/null", 0666)                     = 4
 0.0119 sysconfig(_CONFIG_PAGESIZE)                    = 8192
 0.0121 brk(0x00023F40)                                = 0
 0.0123 brk(0x0002BF40)                                = 0
 0.0127 sigaction(SIGINT, 0xFFBEF470, 0xFFBEF4F0)      = 0
 0.0129 sigaction(SIGINT, 0xFFBEF470, 0xFFBEF4F0)      = 0
 0.0134 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0137 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)  = 16384
 0.0140 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0143 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)  = 16384
 0.0146 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0149 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)  = 16384
 0.0152 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0154 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)  = 16384
```

# Process Tracing – System Call Summary

- Counts total cpu seconds per system call and calls

```
# truss -c dd if=500m of=/dev/null bs=16k count=2k

syscall          seconds   calls  errors
_exit                .00       1
read                 .34    2048
write                .03    2056
open                 .00       4
close                .00       6
brk                  .00       2
fstat                .00       3
execve               .00       1
sigaction            .00       2
mmap                 .00       7
munmap               .00       2
sysconfig            .00       1
llseek               .00       1
creat64              .00       1
open64               .00       1
                    ----     ---     ---
sys totals:          .37    4136       0
usr time:            .00
elapsed:             .89
```

# Library Tracing - truss -u

```
# truss -d -u a.out,libc dd if=500m of=/dev/null bs=16k count=2k
Base time stamp:  925932005.2498  [ Wed May  5 12:20:05 PDT 1999 ]
 0.0000 execve("/usr/bin/dd", 0xFFBEF68C, 0xFFBEF6A4)  argc = 5
 0.0073 open("/dev/zero", O_RDONLY)              = 3
 0.0077 mmap(0x00000000, 8192, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE, 3, 0) = 0xFF3A0000
 0.0094 open("/usr/lib/libc.so.1", O_RDONLY)          = 4
 0.0097 fstat(4, 0xFFBEF224)                    = 0
 0.0100 mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF390000
 0.0102 mmap(0x00000000, 761856, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF280000
 0.0105 munmap(0xFF324000, 57344)                 = 0
 0.0107 mmap(0xFF332000, 25284, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 663552) = 0xFF332000
 0.0113 close(4)                          = 0
 0.0116 open("/usr/lib/libdl.so.1", O_RDONLY)         = 4
 0.0119 fstat(4, 0xFFBEF224)                    = 0
 0.0121 mmap(0xFF390000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0) = 0xFF390000
 0.0124 close(4)                          = 0
 0.0127 open("/usr/platform/SUNW,Ultra-2/lib/libc_psr.so.1", O_RDONLY) = 4
 0.0131 fstat(4, 0xFFBEF004)                    = 0
 0.0133 mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF380000
 0.0135 mmap(0x00000000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF370000
 0.0138 close(4)                          = 0
 0.2369 close(3)                          = 0
 0.2372 munmap(0xFF380000, 8192)                  = 0
 0.2380 -> libc:atexit(0xff3b9e8c, 0x23400, 0x0, 0x0)
 0.2398 <- libc:atexit() = 0
 0.2403 -> libc:atexit(0x12ed4, 0xff3b9e8c, 0xff334518, 0xff332018)
 0.2419 <- libc:atexit() = 0
 0.2424 -> _init(0x0, 0x12ed4, 0xff334518, 0xff332018)
 0.2431 <- _init() = 0
 0.2436 -> main(0x5, 0xffbef68c, 0xffbef6a4, 0x23400)
 0.2443   -> libc:setlocale(0x6, 0x12f14, 0x0, 0x0)
 0.2585   <- libc:setlocale() = 0xff31f316
```

# Library Tracing – apptrace(1)

```
sunsys> apptrace ls
ls       -> libc.so.1:atexit(func = 0xff3caa24) = 0x0
ls       -> libc.so.1:atexit(func = 0x13ad4) = 0x0
ls       -> libc.so.1:setlocale(category = 0x6, locale = "") = "/en_US.ISO8859-1/en_"
ls       -> libc.so.1:textdomain(domainname = "SUNW_OST_OSCMD") = "SUNW_OST_OSCMD"
ls       -> libc.so.1:time(tloc = 0x0) = 0x3aee2678
ls       -> libc.so.1:isatty(fildes = 0x1) = 0x1
ls       -> libc.so.1:getopt(argc = 0x1, argv = 0xffbeeff4, optstring = "RaAdC1xmnlogrtucpFbq")
ls       -> libc.so.1:getenv(name = "COLUMNS") = "<nil>"
ls       -> libc.so.1:ioctl(0x1, 0x5468, 0x2472a)
ls       -> libc.so.1:malloc(size = 0x100) = 0x25d10
ls       -> libc.so.1:malloc(size = 0x9000) = 0x25e18
ls       -> libc.so.1:lstat64(path = ".", buf = 0xffbeee98) = 0x0
ls       -> libc.so.1:qsort(base = 0x25d10, nel = 0x1, width = 0x4, compar = 0x134bc)
ls       -> libc.so.1:.div(0x50, 0x3, 0x50)
ls       -> libc.so.1:.div(0xffffffff, 0x1a, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x0, 0xffffffff)
ls       -> libc.so.1:.mul(0x1, 0x1, 0x0)
```

# User Threads

- The programming abstraction for creating multithreaded programs
  - Parallelism
  - POSIX and UI thread APIs
    - thr_create(3THR)
    - pthread_create(3THR)
  - Synchronization
    - Mutex locks, reader/writer locks, semaphores, condition variables
- Solaris 2 originally implemented an MxN threads model (T1)
  - "unbound" threads
- Solaris 8 introduced the 1 level model (T2)
  - /usr/lib/lwp/libthread.so
- T2 is the default in Solaris 9 and Solaris 10

# Threads Primer Example:

```
#include <pthread.h>
#include <stdio.h>

mutex_t mem_lock;

void childthread(void *argument)
{
     int i;

     for(i = 1; i <= 100; ++i) {
          print("Child Count - %d\n", i);
     }
     pthread_exit(0);
 }

int main(void)
{
     pthread_t thread, thread2;
     int ret;

     if ((pthread_create(&thread, NULL, (void *)childthread, NULL)) < 0) {
          printf ("Thread Creation Failed\n");
          return (1);
     }
     pthread_join(thread,NULL);
     print("Parent is continuing....\n");
     return (0);
}
```

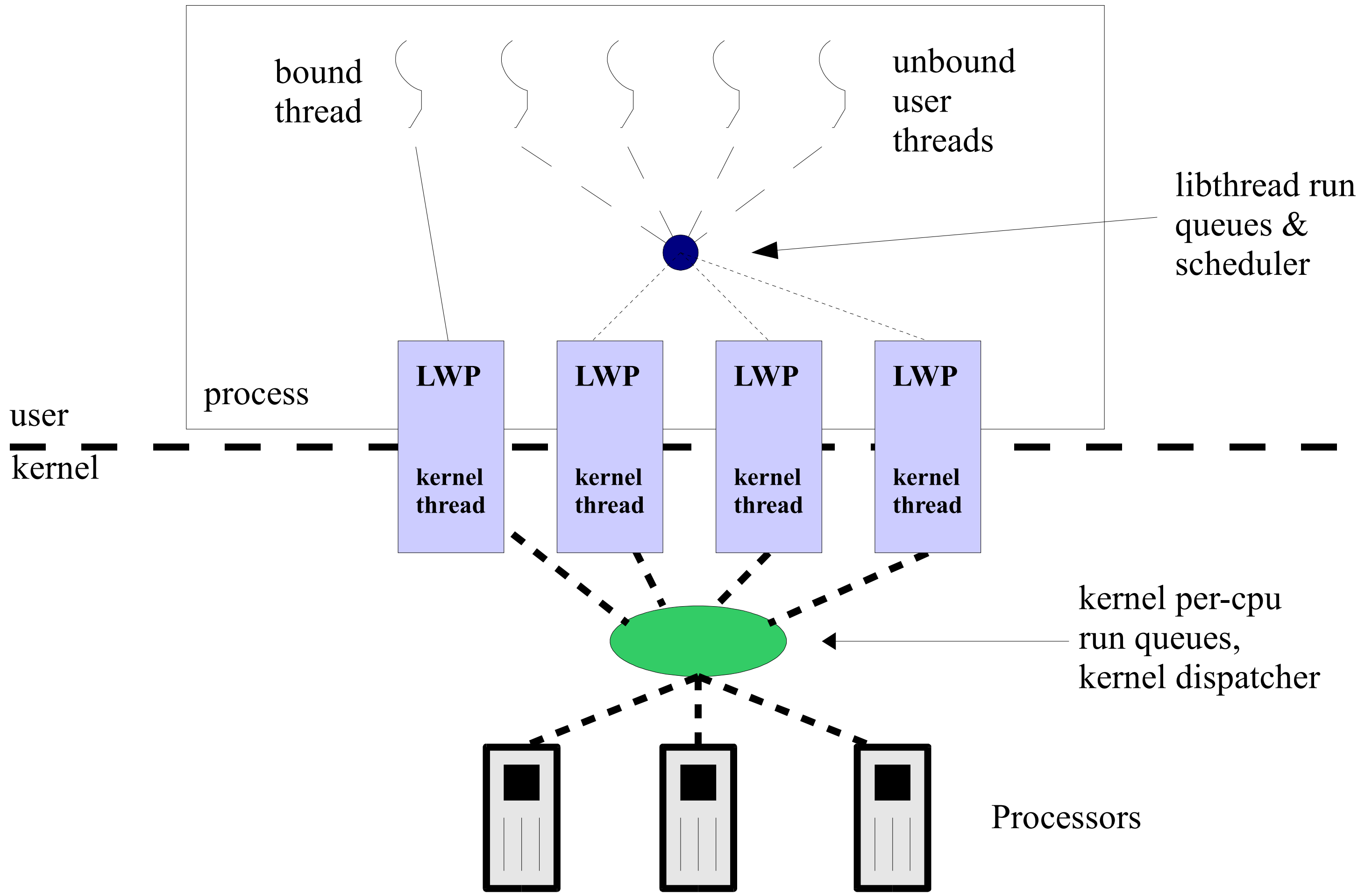
# T1 – Multilevel MxN Model

- /usr/lib/libthread.so.1
- Based on the assumption that kernel threads are expensive, user threads are cheap.
- User threads are virtualized, and may be multiplexed onto one or more kernel threads
  - LWP pool
- User level thread synchronization - threads sleep at user level. (Process private only)
- Concurrency via set_concurrency() and bound LWPs

# T1 – Multilevel Model

- Unbound Thread Implementation
  - User Level scheduling
  - Unbound threads switched onto available lwps
  - Threads switched when blocked on sync object
  - Thread temporary bound when blocked in system call
  - Daemon lwp to create new lwps
  - Signal direction handled by Daemon lwp
  - Reaper thread to manage cleanup
  - Callout lwp for timers

# T1- Multilevel Model
## (default in Solaris 8)



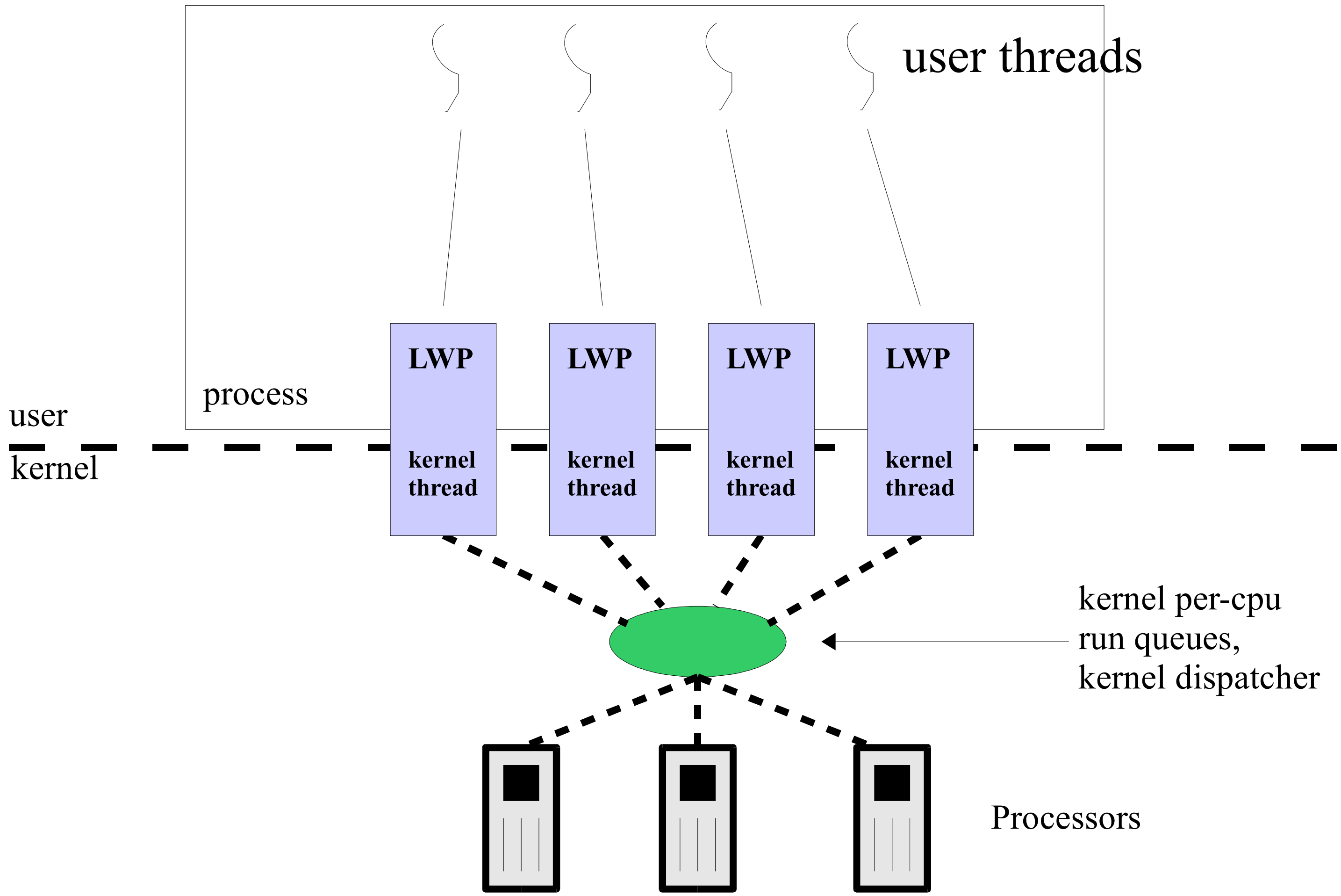
# T1 – Multilevel Model

- Pros:
  - Fast user thread create and destroy
  - Allows many-to-few thread model, to mimimize the number of kernel threads and LWPs
  - Uses minimal kernel memory
  - No system call required for synchronization
  - Process Private Synchronization only
  - Can have thousands of threads
  - Fast context-switching
- Cons:
  - Complex, and tricky programming model wrt achieving good scalability - need to bind or use set_concurrency()
  - Signal delivery
  - Compute bound threads do not surrender, leading to excessive CPU consumption and potential starving
  - Complex to maintain (for Sun)

# T2 – Single Level Threads Model

- All user threads bound to LWPs
  - All bound threads
- Kernel level scheduling
  - No more libthread.so scheduler
- Simplified Implementation
- Uses kernel's synchronization objects
  - Slightly different behaviour LIFO vs. FIFO
  - Allows adaptive lock behaviour
- More expensive thread create/destroy, synchronization
- More responsive scheduling, synchronization

# T2 – Single Level Threads Model



# T2 - Single Level Thread Model

- Scheduling wrt Synchronization (S8U7/S9/S10)
  - Adaptive locks give preference to a thread that is running, potentially at the expense of a thread that is sleeping
  - Threads that rely on fairness of scheduling/CPU could end up ping-ponging, at the expense of another thread which has work to do.
- Default S8U7/S9/S10 Behaviour
  - Adaptive Spin
    - 1000 of iterations (spin count) for adaptive mutex locking before giving up and going to sleep.
  - Maximum number of spinners
    - The number of simultaneously spinning threads
    - attempting to do adaptive locking on one mutex is limited to 100.
  - One out of every 16 queuing operations will put a thread at the end of the queue, to prevent starvation.
  - Stack Cache
    - The maximum number of stacks the library retains after threads exit for re-use when more threads are created is 10.

# Watching Threads

```
  PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/LWPID
29105 root     5400K 3032K sleep   60    0   0:00:00 1.3% pkginstall/1
29051 root     5072K 4768K cpu0    49    0   0:00:00 0.8% prstat/1
  202 root     3304K 1256K sleep   59    0   0:00:07 0.3% nscd/23
25947 root     5160K  608K sleep   59    0   0:00:05 0.2% sshd/1
23078 root       20M 1880K sleep   59    0   0:00:58 0.2% lupi_zones/1
25946 rmc      3008K  624K sleep   59    0   0:00:02 0.2% ssh/1
23860 root     5248K  688K sleep   59    0   0:00:06 0.2% sshd/1
29100 root     1272K  976K sleep   59    0   0:00:00 0.1% mpstat/1
24866 root     5136K  600K sleep   59    0   0:00:02 0.0% sshd/1
  340 root     2504K  672K sleep   59    0   0:11:14 0.0% mibiisa/2
23001 root     5136K  584K sleep   59    0   0:00:04 0.0% sshd/1
  830 root     2472K  600K sleep   59    0   0:11:01 0.0% mibiisa/2
  829 root     2488K  648K sleep   59    0   0:11:01 0.0% mibiisa/2
    1 root     2184K  400K sleep   59    0   0:00:01 0.0% init/1
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/13
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/12
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/11
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/10
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/9
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/8
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/7
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/6
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/5
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/4
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/3
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/2
  202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/1
  126 daemon   2360K    8K sleep   59    0   0:00:00 0.0% rpcbind/1
  814 root     1936K  280K sleep   59    0   0:00:00 0.0% sac/1
   64 root     2952K    8K sleep   59    0   0:00:00 0.0% picld/5
   64 root     2952K    8K sleep   59    0   0:00:00 0.0% picld/4
   64 root     2952K    8K sleep   59    0   0:00:00 0.0% picld/3
   64 root     2952K    8K sleep   59    0   0:00:00 0.0% picld/2
   64 root     2952K    8K sleep   59    0   0:00:00 0.0% picld/1
   61 daemon   3640K    8K sleep   59    0   0:00:00 0.0% kcfd/3
   61 daemon   3640K    8K sleep   59    0   0:00:00 0.0% kcfd/2
   61 daemon   3640K    8K sleep   59    0   0:00:00 0.0% kcfd/1
   55 root     2416K    8K sleep   59    0   0:00:00 0.0% syseventd/14
   55 root     2416K    8K sleep   59    0   0:00:00 0.0% syseventd/13
   55 root     2416K    8K sleep   59    0   0:00:00 0.0% syseventd/12
   55 root     2416K    8K sleep   59    0   0:00:00 0.0% syseventd/11
Total: 125 processes, 310 lwps, load averages: 0.50, 0.38, 0.40
```

# Thread Semantics Added to pstack, truss

```
# pstack 909/2
909:    dbwr -a dbwr -i 2 -s b0000000 -m /var/tmp/fbencAAAmxaqxb
-----------------  lwp# 2  --------------------------------
 ceab1809 lwp_park (0, afffde50, 0)
 ceaabf93 cond_wait_queue (ce9f8378, ce9f83a0, afffde50, 0) + 3b
 ceaac33f cond_wait_common (ce9f8378, ce9f83a0, afffde50) + 1df
 ceaac686 _cond_reltimedwait (ce9f8378, ce9f83a0, afffdea0) + 36
 ceaac6b4 cond_reltimedwait (ce9f8378, ce9f83a0, afffdea0) + 24
 ce9e5902 __aio_waitn (82d1f08, 1000, afffdf2c, afffdf18, 1) + 529
 ceaf2a84 aio_waitn64 (82d1f08, 1000, afffdf2c, afffdf18) + 24
 08063065 flowoplib_aiowait (b4eb475c, c40f4d54) + 97
 08061de1 flowop_start (b4eb475c) + 257
 ceab15c0 _thr_setup (ce9a8400) + 50
 ceab1780 _lwp_start (ce9a8400, 0, 0, afffdff8, ceab1780, ce9a8400)
```

```
pae1> truss -p 2975/3
/3:  close(5)                              = 0
/3:  open("/space1/3", O_RDWR|O_CREAT, 0666)      = 5
/3:  lseek(5, 0, SEEK_SET)                           = 0
/3:  write(5, " U U U U U U U U U U U U".., 1056768)    = 1056768
/3:  lseek(5, 0, SEEK_SET)                           = 0
/3:  read(5, " U U U U U U U U U U U U".., 1056768)     = 1056768
/3:  close(5)                              = 0
/3:  open("/space1/3", O_RDWR|O_CREAT, 0666)      = 5
/3:  lseek(5, 0, SEEK_SET)                           = 0
/3:  write(5, " U U U U U U U U U U U U".., 1056768)    = 1056768
```

# Thread Microstates

```
   PID USERNAME  USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID
   918 rmc       0.2 0.4 0.0 0.0 0.0 0.0  99 0.0  27   2  1K   0 prstat/1
   919 mauroj    0.1 0.4 0.0 0.0 0.0 0.0  99 0.1  44  12  1K   0 prstat/1
   907 root      0.0 0.1 0.0 0.0 0.0 0.0  97 3.1 121   2  20   0 filebench/2
   913 root      0.1 0.0 0.0 0.0 0.0 100 0.0 0.0  15   2 420   0 filebench/2
   866 root      0.0 0.0 0.0 0.0 0.0 0.0  96 4.1  44  41 398   0 filebench/2
   820 root      0.0 0.0 0.0 0.0 0.0 0.0  95 5.0  43  42 424   0 filebench/2
   814 root      0.0 0.0 0.0 0.0 0.0 0.0  95 5.0  43  41 424   0 filebench/2
   772 root      0.0 0.0 0.0 0.0 0.0 0.0  96 3.6  46  39 398   0 filebench/2
   749 root      0.0 0.0 0.0 0.0 0.0 0.0  96 3.7  45  41 398   0 filebench/2
   744 root      0.0 0.0 0.0 0.0 0.0 0.0  95 4.7  47  39 398   0 filebench/2
   859 root      0.0 0.0 0.0 0.0 0.0 0.0  95 4.9  44  41 424   0 filebench/2
   837 root      0.0 0.0 0.0 0.0 0.0 0.0  96 4.0  43  43 405   0 filebench/2
   792 root      0.0 0.0 0.0 0.0 0.0 0.0  96 4.0  44  43 405   0 filebench/2
   773 root      0.0 0.0 0.0 0.0 0.0 0.0  96 3.8  47  37 398   0 filebench/2
   768 root      0.0 0.0 0.0 0.0 0.0 0.0  95 5.3  44  41 398   0 filebench/2
   740 root      0.0 0.0 0.0 0.0 0.0 0.0  95 4.8  44  41 398   0 filebench/2
   894 root      0.0 0.0 0.0 0.0 0.0 0.0  96 3.9  43  42 405   0 filebench/2
   891 root      0.0 0.0 0.0 0.0 0.0 0.0  96 3.9  44  41 405   0 filebench/2
   890 root      0.0 0.0 0.0 0.0 0.0 0.0  95 4.8  45  41 431   0 filebench/2
   861 root      0.0 0.0 0.0 0.0 0.0 0.0  96 3.9  43  43 405   0 filebench/2
   851 root      0.0 0.0 0.0 0.0 0.0 0.0  95 4.8  43  41 398   0 filebench/2
   848 root      0.0 0.0 0.0 0.0 0.0 0.0  95 4.5  42  43 398   0 filebench/2
  [snip]
   787 root      0.0 0.0 0.0 0.0 0.0 0.0  95 4.5  43  41 424   0 filebench/2
   776 root      0.0 0.0 0.0 0.0 0.0 0.0  95 4.8  43  42 398   0 filebench/2
   774 root      0.0 0.0 0.0 0.0 0.0 0.0  96 4.2  43  40 398   0 filebench/2
   756 root      0.0 0.0 0.0 0.0 0.0 0.0  96 3.8  44  41 398   0 filebench/2
   738 root      0.0 0.0 0.0 0.0 0.0 0.0  96 4.4  43  42 398   0 filebench/2
   735 root      0.0 0.0 0.0 0.0 0.0 0.0  96 3.9  47  39 405   0 filebench/2
   734 root      0.0 0.0 0.0 0.0 0.0 0.0  96 4.3  44  41 398   0 filebench/2
   727 root      0.0 0.0 0.0 0.0 0.0 0.0  96 4.4  43  43 398   0 filebench/2
   725 root      0.0 0.0 0.0 0.0 0.0 0.0  96 4.4  43  43 398   0 filebench/2
Total: 257 processes, 3139 lwps, load averages: 7.71, 2.39, 0.97
```

# Scheduling Classes & The Kernel Dispatcher

# Solaris Scheduling

- Solaris implements a central dispatcher, with multiple scheduling classes
  - Scheduling classes determine the priority range of the kernel threads on the system-wide (global) scale, and the scheduling algorithms applied
  - Each scheduling class references a dispatch table
    - Values used to determine time quantums and priorities
    - Admin interface to “tune” thread scheduling
  - Solaris provides command line interfaces for
    - Loading new dispatch tables
    - Changing the scheduling class and priority and threads
  - Observability through
    - `ps`(1)
    - `prstat`(1)
    - `dtrace`(1)

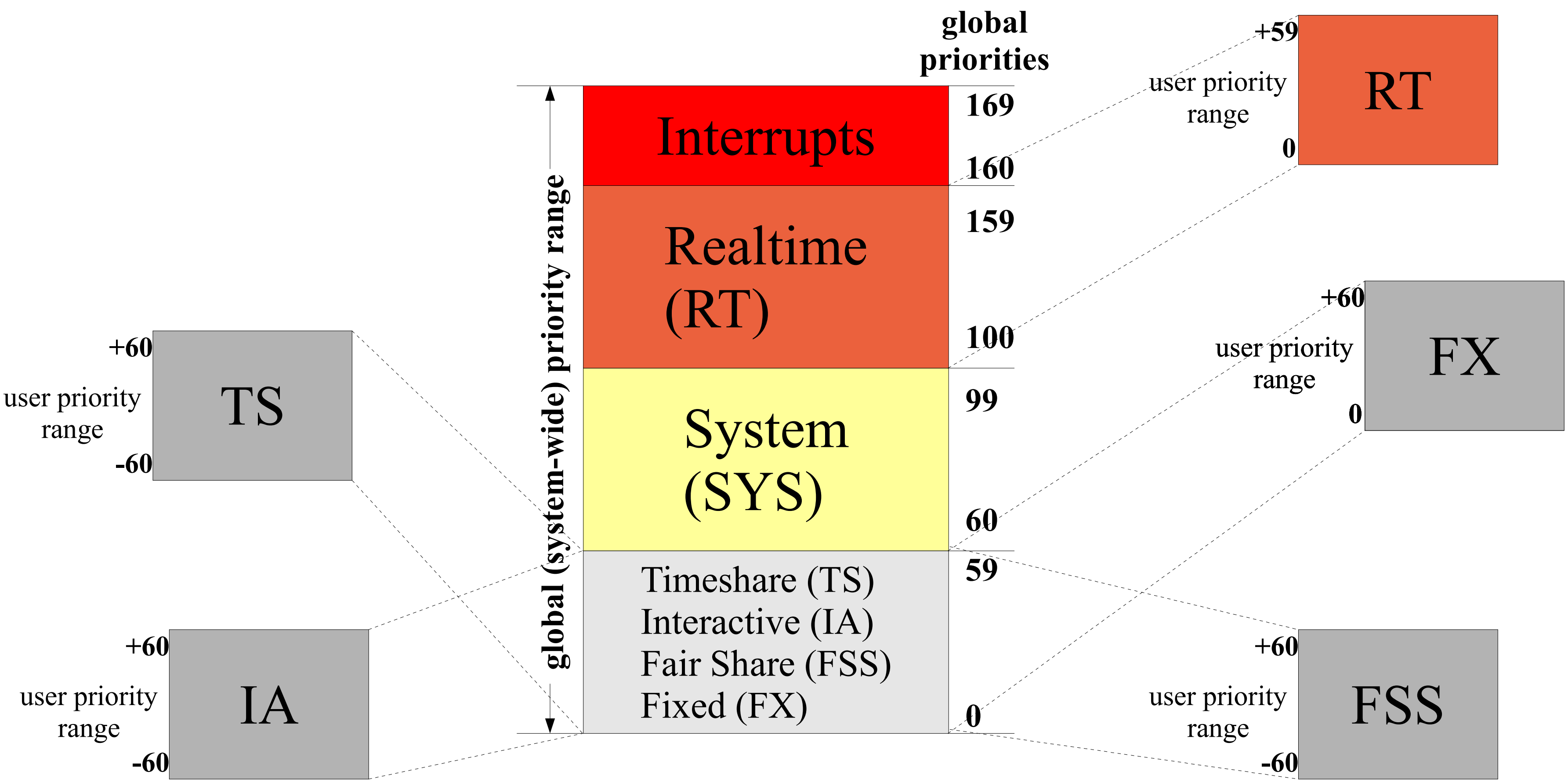
# Scheduling Classes

- Traditional Timeshare (TS) class
  - attempt to give every thread a fair shot at execution time
- Interactive (IA) class
  - Desktop only
  - Boost priority of active (current focus) window
  - Same dispatch table as TS
- System (SYS)
  - Only available to the kernel, for OS kernel threads
- Realtime (RT)
  - Highest priority scheduling class
  - Will preempt kernel (SYS) class threads
  - Intended for realtime applications
    - Bounded, consistent scheduling latency

# Scheduling Classes – Solaris 9 & 10

- Fair Share Scheduler (FSS) Class
  - Same priority range as TS/IA class
  - CPU resources are divided into shares
  - Shares are allocated (projects/tasks) by administrator
  - Scheduling decisions made based on shares allocated and used, not dynamic priority changes
- Fixed Priority (FX) Class
  - The kernel will not change the thread's priority
  - A “batch” scheduling class
- Same set of commands for administration and management
  - `dispadmin`(1M), `priocntl`(1)
  - Resource management framework
    - `rctladm`(1M), `prctl`(1)

# Scheduling Classes and Priorities



# Scheduling Classes

- Use `dispadmin(1M)` and `priocntl(1)`

```
# dispadmin -l
CONFIGURED CLASSES
==================

SYS   (System Class)
TS    (Time Sharing)
FX    (Fixed Priority)
IA    (Interactive)
FSS   (Fair Share)
RT    (Real Time)
# priocntl -l
CONFIGURED CLASSES
==================

SYS (System Class)

TS (Time Sharing)
      Configured TS User Priority Range: -60 through 60

FX (Fixed priority)
      Configured FX User Priority Range: 0 through 60

IA (Interactive)
      Configured IA User Priority Range: -60 through 60

FSS (Fair Share)
      Configured FSS User Priority Range: -60 through 60

RT (Real Time)
      Maximum Configured RT Priority: 59
#
```

# Scheduling Classes

- The kernel maintains an array of sclass structures for each loaded scheduling class
  - References the scheduling classes init routine, class functions structure, etc
- Scheduling class information is maintained for every kernel thread
  - Thread pointer to the class functions array, and per-thread class-specific data structure
  - Different threads in the same process can be in different scheduling classes
- Scheduling class operations vectors and CL_XXX macros allow a single, central dispatcher to invoke scheduling-class specific functions

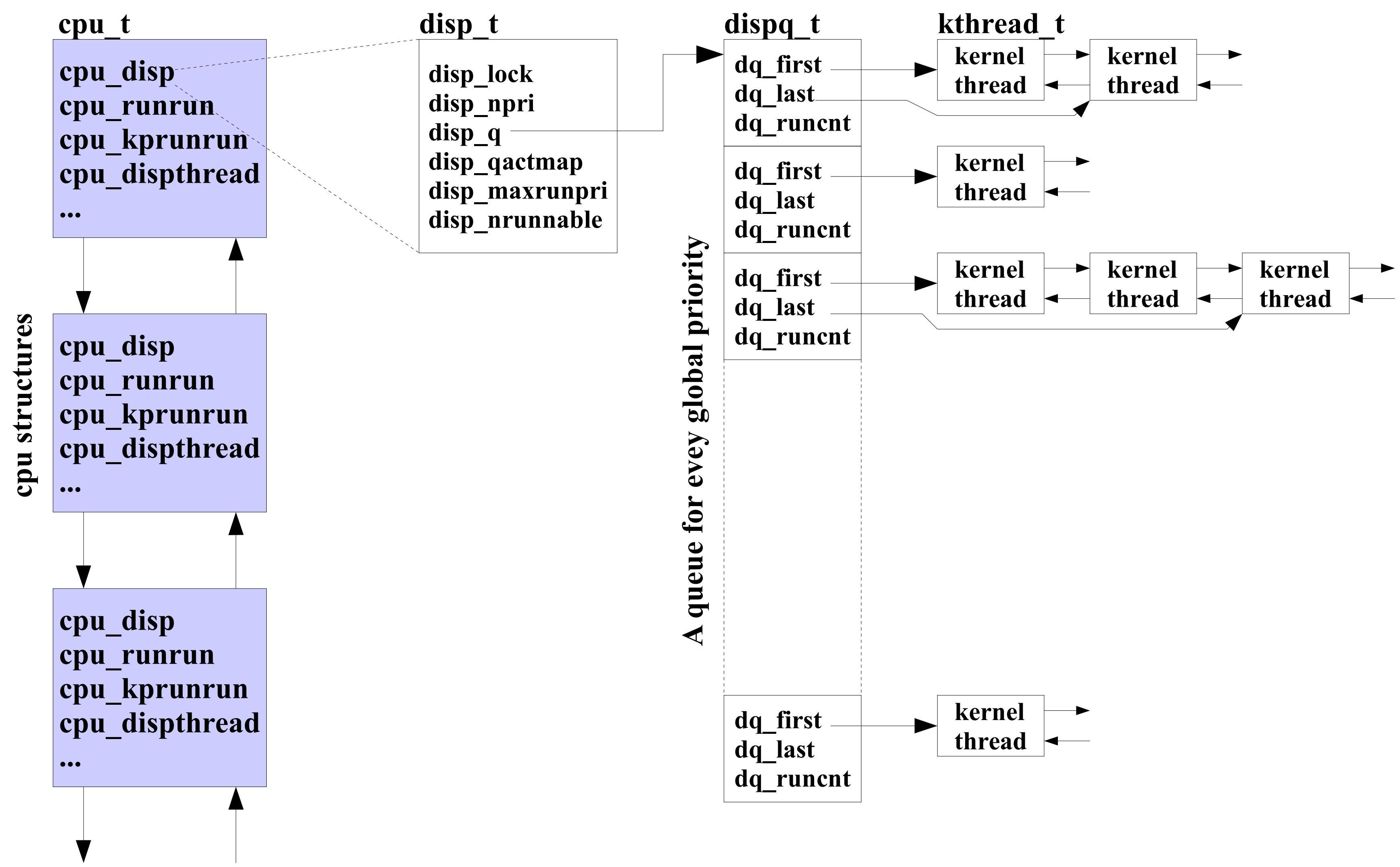
# Scheduling Class & Priority of Threads

```
solaris10> ps -eLc
   PID   LWP  CLS PRI TTY        LTIME CMD
     0     1  SYS  96 ?           0:00 sched
     1     1   TS  59 ?           0:00 init
     2     1  SYS  98 ?           0:00 pageout
     3     1  SYS  60 ?           5:08 fsflush
   402     1   TS  59 ?           0:00 sac
   269     1   TS  59 ?           0:00 utmpd
   225     1   TS  59 ?           0:00 automoun
   225     2   TS  59 ?           0:00 automoun
   225     4   TS  59 ?           0:00 automoun
    54     1   TS  59 ?           0:00 sysevent
    54     2   TS  59 ?           0:00 sysevent
    54     3   TS  59 ?           0:00 sysevent
    [snip]
   426     1   IA  59 ?           0:00 dtgreet
   343     1   TS  59 ?           0:00 mountd
   345     1   FX  60 ?           0:00 nfsd
   345     3   FX  60 ?           0:00 nfsd
   350     1   TS  59 ?           0:00 dtlogin
   375     1   TS  59 ?           0:00 snmpdx
   411     1   IA  59 ?           0:00 dtlogin
   412     1   IA  59 ??          0:00 fbconsol
   403     1   TS  59 console     0:00 ttymon
   405     1   TS  59 ?           0:00 ttymon
   406     1   IA  59 ?           0:03 Xsun
   410     1   TS  59 ?           0:00 sshd
   409     1   TS  59 ?           0:00 snmpd
  1040     1   TS  59 ?           0:00 in.rlogi
  1059     1   TS  49 pts/2       0:00 ps
solaris10>
```

# Dispatch Queues & Dispatch Tables

- Dispatch queues
  - Per-CPU run queues
    - Actually, a queue of queues
  - Ordered by thread priority
  - Queue occupation represented via a bitmap
  - For Realtime threads, a system-wide kernel preempt queue is maintained
    - Realtime threads are placed on this queue, not the per-CPU queues
    - If processor sets are configured, a kernel preempt queue exists for each processor set
- Dispatch tables
  - Per-scheduling class parameter tables
  - Time quantums and priorities
  - tuneable via `dispadmin`(1M)

# Per-CPU Dispatch Queues



# Timeshare Dispatch Table

- TS and IA class share the same dispatch table
  - RES. Defines the granularity of ts_quantum
  - ts_quantum. CPU time for next ONPROC state
  - ts_tqexp. New priority if time quantum expires
  - ts_slpret. New priority when state change from TS_SLEEP to TS_RUN
  - ts_maxwait. “waited to long” ticks
  - ts_lwait. New priority if “waited to long”

```
# dispadmin -g -c TS
# Time Sharing Dispatcher Configuration
RES=1000

# ts_quantum             ts_tqexp    ts_slpret  ts_maxwait     ts_lwait                 PRIORITY LEVEL
       200                   0           50          0            50                       #         0
       200                   0           50          0            50                       #         1
        .........
       160                   0           51          0            51                       #        10
       160                   1           51          0            51                       #        11
        ..........
       120                  10           52          0            52                       #        20
       120                  11           52          0            52                       #        21
        .........
       80                   20           53          0            53                       #        30
       80                   21           53          0            53                       #        31
        ..........
       40                   30           55          0            55                       #        40
       40                   31           55          0            55                       #        41
        ...........
       20                   49           59        32000          59                       #        59
```

# RT, FX & FSS Dispatch Tables

- RT
  - Time quantum only
  - For each possible priority
- FX
  - Time quantum only
  - For each possible priority
- FSS
  - Time quantum only
  - Just one, not defined for each priority level
    - Because FSS is share based, not priority based
- SYS
  - No dispatch table
  - Not needed, no rules apply
- INT
  - Not really a scheduling class

# Dispatch Queue Placement

- Queue placement is based a few simple parameters
  - The thread priority
  - Processor binding/Processor set
  - Processor thread last ran on
    - Warm affinity
  - Depth and priority of existing runnable threads
  - Solaris 9 added Memory Placement Optimization (MPO) enabled will keep thread in defined locality group (lgroup)

```
if (thread is bound to CPU-n) && (pri < kpreemptpri)
        CPU-n dispatch queue
if (thread is bound to CPU-n) && (pri >= kpreemptpri)
        CPU-n dispatch queue
if (thread is not bound) && (pri < kpreemptpri)
    place thread on a CPU dispatch queue
if (thread is not bound) && (pri >= kpreemptpri)
    place thread on cp_kp_queue
```

# Thread Selection

- The kernel dispatcher implements a select-and-ratify thread selection algorithm
  - disp_getbest(). Go find the highest priority runnable thread, and select it for execution
  - disp_ratify(). Commit to the selection. Clear the CPU preempt flags, and make sure another thread of higher priority did not become runnable
    - If one did, place selected thread back on a queue, and try again
- Warm affinity is implemented
  - Put the thread back on the same CPU it executed on last
    - Try to get a warm cache
  - `rechoose_interval` kernel parameter
    - Default is 3 clock ticks

# Thread Preemption

- Two classes of preemption
  - User preemption
    - A higher priority thread became runnable, but it's not a realtime thread
    - Flagged via cpu_runrun in CPU structure
    - Next clock tick, you're outta here
  - Kernel preemption
    - A realtime thread became runnable. Even OS kernel threads will get preempted
    - Poke the CPU (cross-call) and preempt the running thread now
  - Note that threads that use-up thier time quantum are evicted via the preempt mechanism
  - Monitor via "icsw" column in `mpstat`(1)

# Thread Execution

- Run until
  - A preemption occurs
    - Transition from S_ONPROC to S_RUN
    - placed back on a run queue
  - A blocking system call is issued
    - e.g. read(2)
    - Transition from S_ONPROC to S_SLEEP
    - Placed on a sleep queue
  - Done and exit
    - Clean up
  - Interrupt to the CPU you're running on
    - pinned for interrupt thread to run
    - unpinned to continue

# Sleep & Wakeup

- Condition variables used to synchronize thread sleep/wakeup
  - A block condition (waiting for a resource or an event) enters the kernel cv_xxx() functions
  - The condition variable is set, and the thread is placed on a sleep queue
  - Wakeup may be directed to a specific thread, or all threads waiting on the same event or resource
    - One or more threads moved from sleep queue, to run queue

# Observability and Performance

- Use `prstat`(1) and `ps`(1) to monitor running processes and threads
- Use `mpstat`(1) to monitor CPU utilization, context switch rates and thread migrations
- Use `dispadmin`(1M) to examine and change dispatch table parameters
- User `priocntl`(1) to change scheduling classes and priorities
  - `nice`(1) is obsolete (but there for compatibility)
  - User priorities also set via `priocntl`(1)
  - Must be root to use RT class

# Turnstiles & Priority Inheritance

- Turnstiles are a specific implementation of sleep queues that provide priority inheritance
- Priority Inheritance (PI) addresses the priority inversion problem
  - Priority inversion is when a higher priority thread is prevented from running because a lower priority thread is holding a lock the higher priority thread needs
    - Blocking chains can form when “mid” priority threads get in the mix
- Priority inheritance
  - If a resource is held, ensure all the threads in the blocking chain are at the requesting thread's priority, or better
    - All lower priority threads inherit the priority of the

# Processors, Processor Controls & Binding

# Processor Controls

- Processor controls provide for segration of workload(s) and resources
- Processor status, state, management and control
  - Kernel linked list of CPU structs, one for each CPU
  - Bundled utilities
    - `psradm`(1)
    - `psrinfo`(1)
  - Processors can be taken offline
    - Kernel will not schedule threads on an offline CPU
  - The kernel can be instructed not to bind device interrupts to processor(s)
    - Or move them if bindings exist

# Processor Control Commands

- psrinfo(1M) - provides information about the processors on the system. Use "-v" for verbose
- psradm(1M) - online/offline processors. Pre Sol 7, offline processors still handled interrupts. In Sol 7, you can disable interrupt participation as well
- psrset(1M) - creation and management of processor sets
- pbind(1M) - original processor bind command. Does not provide exclusive binding
- processor_bind(2), processor_info(2), pset_bind(2), pset_info(2), pset_creat(2), p_online(2)
  - system calls to do things programmatically

# Processor Sets

- Partition CPU resources for segragating workloads, applications and/or interrupt handling
- Dynamic
  - Create, bind, add, remove, etc, without reboots
- Once a set is created, the kernel will only schedule threads onto the set that have been explicity bound to the set
  - And those threads will only ever be scheduled on CPUs in the set they've been bound to
- Interrupt disabling can be done on a set
  - Dedicate the set, through binding, to running application threads
  - Interrupt segregation can be effective if interrupt load is heavy
    - e.g. high network traffic

# Example: Managing a cpuhog

# Timeshare (TS) Scheduling (prstat -l)

```
 PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/LWPID
  746 mauroj    118M  118M sleep   59    0   0:00:20 3.5% cpuhog/6
  746 mauroj    118M  118M sleep   59    0   0:00:19 3.3% cpuhog/5
  746 mauroj    118M  118M sleep   33    0   0:00:19 3.2% cpuhog/22
  746 mauroj    118M  118M sleep   59    0   0:00:20 3.2% cpuhog/30
  746 mauroj    118M  118M sleep   40    0   0:00:20 3.1% cpuhog/23
  746 mauroj    118M  118M sleep   59    0   0:00:19 3.1% cpuhog/31
  746 mauroj    118M  118M sleep   59    0   0:00:18 3.0% cpuhog/26
  746 mauroj    118M  118M sleep   59    0   0:00:19 3.0% cpuhog/17
  746 mauroj    118M  118M sleep   59    0   0:00:20 2.9% cpuhog/8
  746 mauroj    118M  118M cpu8    20    0   0:00:18 2.9% cpuhog/9
  746 mauroj    118M  118M sleep   51    0   0:00:18 2.9% cpuhog/10
  746 mauroj    118M  118M sleep   51    0   0:00:20 2.9% cpuhog/2
  746 mauroj    118M  118M cpu13   42    0   0:00:19 2.9% cpuhog/15
  746 mauroj    118M  118M sleep   59    0   0:00:17 2.8% cpuhog/20
  746 mauroj    118M  118M sleep   59    0   0:00:19 2.8% cpuhog/32
  746 mauroj    118M  118M sleep   59    0   0:00:18 2.8% cpuhog/18
  746 mauroj    118M  118M sleep   59    0   0:00:17 2.7% cpuhog/27
  746 mauroj    118M  118M sleep   59    0   0:00:17 2.7% cpuhog/21
  746 mauroj    118M  118M sleep   33    0   0:00:17 2.7% cpuhog/12
  746 mauroj    118M  118M sleep   59    0   0:00:17 2.7% cpuhog/16
  746 mauroj    118M  118M sleep   42    0   0:00:17 2.7% cpuhog/3
  746 mauroj    118M  118M sleep   31    0   0:00:17 2.7% cpuhog/13
  746 mauroj    118M  118M sleep   55    0   0:00:19 2.7% cpuhog/7
  746 mauroj    118M  118M sleep   33    0   0:00:18 2.5% cpuhog/4
  746 mauroj    118M  118M sleep   59    0   0:00:18 2.4% cpuhog/24
  746 mauroj    118M  118M cpu4    39    0   0:00:16 2.3% cpuhog/14
  746 mauroj    118M  118M sleep   43    0   0:00:15 2.3% cpuhog/11
  746 mauroj    118M  118M cpu0    59    0   0:00:17 2.3% cpuhog/33
  746 mauroj    118M  118M sleep   31    0   0:00:15 2.2% cpuhog/19
  746 mauroj    118M  118M sleep   59    0   0:00:15 2.2% cpuhog/29
  746 mauroj    118M  118M sleep   30    0   0:00:15 2.1% cpuhog/25
  746 mauroj    118M  118M sleep   59    0   0:00:15 2.0% cpuhog/28
  747 mauroj   4704K 4408K cpu5    49    0   0:00:00 0.0% prstat/1
```

# Timeshare – No partitioning

```
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0   18   0  777   412  303   88   38   24   43    0   173   73   0   0  27
  1   30   0   13   124  101   86   34   16   44    0   181   91   0   0   9
  4   22   0    4   131  112   69   31   15   37    0    84   98   0   0   2
  5   26   0    7   116  100   59   26   10   44    0    76   99   1   0   0
  8   24   0    6   121  100   64   33   16   33    0   105   96   2   0   2
  9   22   0    5   116  100   63   27   11   39    0    73   96   2   0   2
 12   20   0    4   119  101   76   26   18   29    0    70   86   0   0  14
 13   20   0   13   115  100   72   26   14   40    0    80   84   2   0  14
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0   26   0  761   407  301   45   28   14   43    0    80   87   0   0  13
  1   18   0    5   116  101   86   27   23   35    1    73   89   0   0  11
  4   24   0    7   124  110   64   29   12   30    0    60   99   1   0   0
  5   14   0   22   115  101   82   30   23   45    0    97   71   2   0  27
  8   28   0    7   113  100   61   24   11   42    0    69   94   4   0   2
  9   24   0    5   116  101   75   25   22   41    0    83   78   5   0  17
 12   34   0    8   119  101   71   28   18   29    0    63   90   8   0   2
 13   20   0    8   122  100   74   33   17   33    0    71   76   5   0  19
```

# Creating a Processor Set for cpuhog

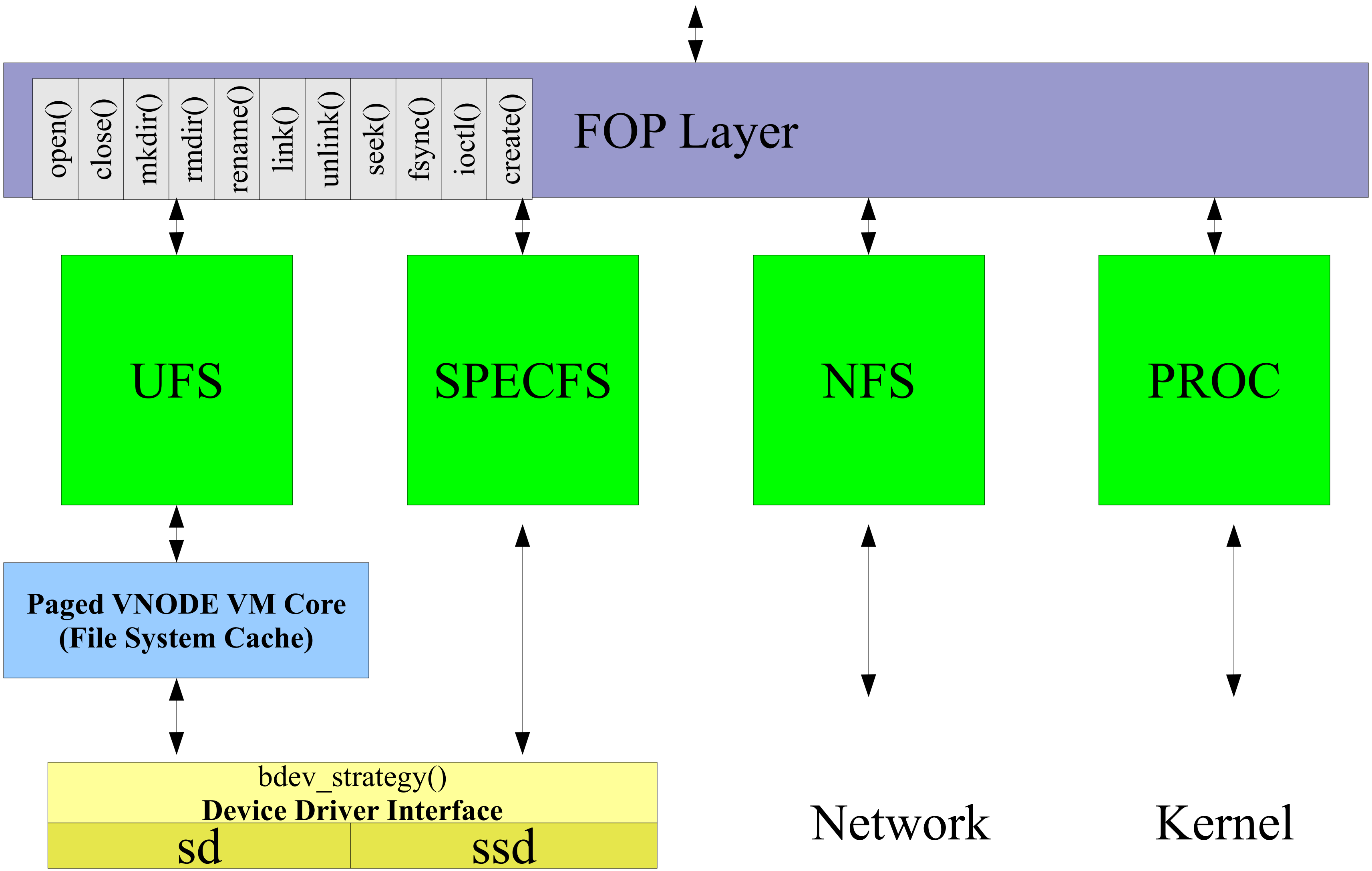
```
# psrinfo
0    on-line   since 09/19/2003 01:18:13
1    on-line   since 09/19/2003 01:18:17
4    on-line   since 09/19/2003 01:18:17
5    on-line   since 09/19/2003 01:18:17
8    on-line   since 09/19/2003 01:18:17
9    on-line   since 09/19/2003 01:18:17
12   on-line   since 09/19/2003 01:18:17
13   on-line   since 09/19/2003 01:18:17
# psrset -c 8 9 12 13
created processor set 1
processor 8: was not assigned, now 1
processor 9: was not assigned, now 1
processor 12: was not assigned, now 1
processor 13: was not assigned, now 1
# psrset -e 1 ./cpuhog 1 0


# mpstat 1
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0   0  746   401  301   12    0    1   10    0     0    0   0   0 100
  1    0   0    0   101  100   12    0    0    0    0    27    0   0   0 100
  4    0   0    5   109  107   14    0    0    0    0     0    0   0   0 100
  5    0   0    0   103  102   10    0    0    0    0     0    0   0   0 100
  8   71   0    9   124  100   81   42    6   51    0   101  100   0   0   0
  9   66   0   13   121  100   84   39    3   48    0   111   99   1   0   0
 12   49   0    5   117  100   71   27    6   29    0    88   99   1   0   0
 13   55   0    4   124  100   76   40    6   35    0    90  100   0   0   0
```

# File System Architecture and Implementation

# Solaris Disk based File Systems

- UFS: The original Solaris file system, adapted from Berkeley FFS
  - Block based, supplemented with meta-data logging
- QFS: Designed for high-bandwidth, muliple nodes etc
  - Extend based
  - Meta-data separation
- ZFS: Coming soon
  - Log structured
  - Checksummed, always consistent

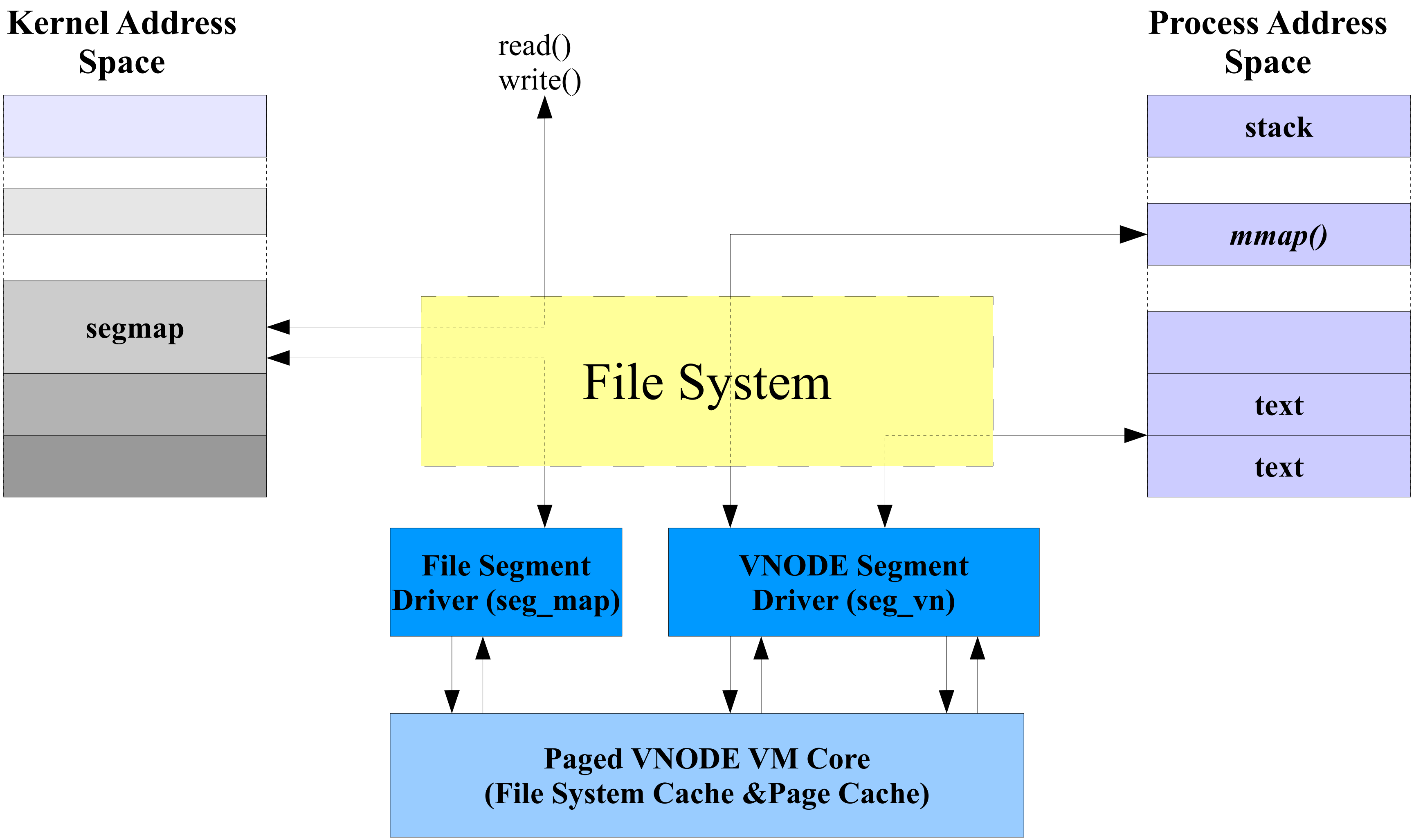
# File System Architecture



# File system I/O via Virtual Memory

- File system I/O is performed by the VM system
  - Reads are performed by page-in
  - Write are performed by page-out
- Practical Implications
  - Virtual memory caches files, cache is dynamic
  - Minimum I/O size is the page size
  - Read/modify/write may occur on sub page-size writes
- Memory Allocation Policy:
  - File system cache is lower priority than app, kernel etc
  - File system cache grows when there is free memory available
  - File system cache shrinks when there is demand elsewhere.

# File System I/O



# File System Reads: A UFS Read

- Application calls read()
- Read system call calls fop_read()
- FOP layer redirector calls underlying filesystem
- FOP jumps into ufs_read
- UFS locates a mapping for the corresponding pages in the file system page cache using vnode/offset
- UFS asks segmap for a mapping to the pages
- If the page exists in the fs, data is copied to App.
  - We're done.
- If the page doesn't exist, a Major fault occurs
  - VM system invokes ufs_getpage()
  - UFS schedules a page size I/O for the page
  - When I/O is complete, data is copied to App

# Vmstat -p

swap = free and unreserved swap in KBytes
free = free memory measured in pages

re = kilobytes reclaimed from cache/free list
mf = minor faults - the page was in memory but was not mapped
fr = kilobytes that have been destroyed or freed
de = kilobytes freed after writes
sr = kilobytes scanned / second

executable pages: kilobytes in - out - freed

anonymous pages: kilobytes in - out – freed

file system pages: kilobytes in - out - freed

```
# vmstat -p 5 5
      memory              page              executable       anonymous       filesystem
   swap    free    re  mf   fr   de   sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf
 ...
 46715224 891296 24   350 0    0    0    0    0    0    4    0    0    27    0    0
 46304792 897312 151 761 25   0    0   17    0    0    1    0    0   280   25   25
 45886168 899808 118 339 1    0    0    3    0    0    1    0    0   641    1    1
 46723376 899440 29   197 0    0    0    0    0    0   40    0    0    60    0    0
```

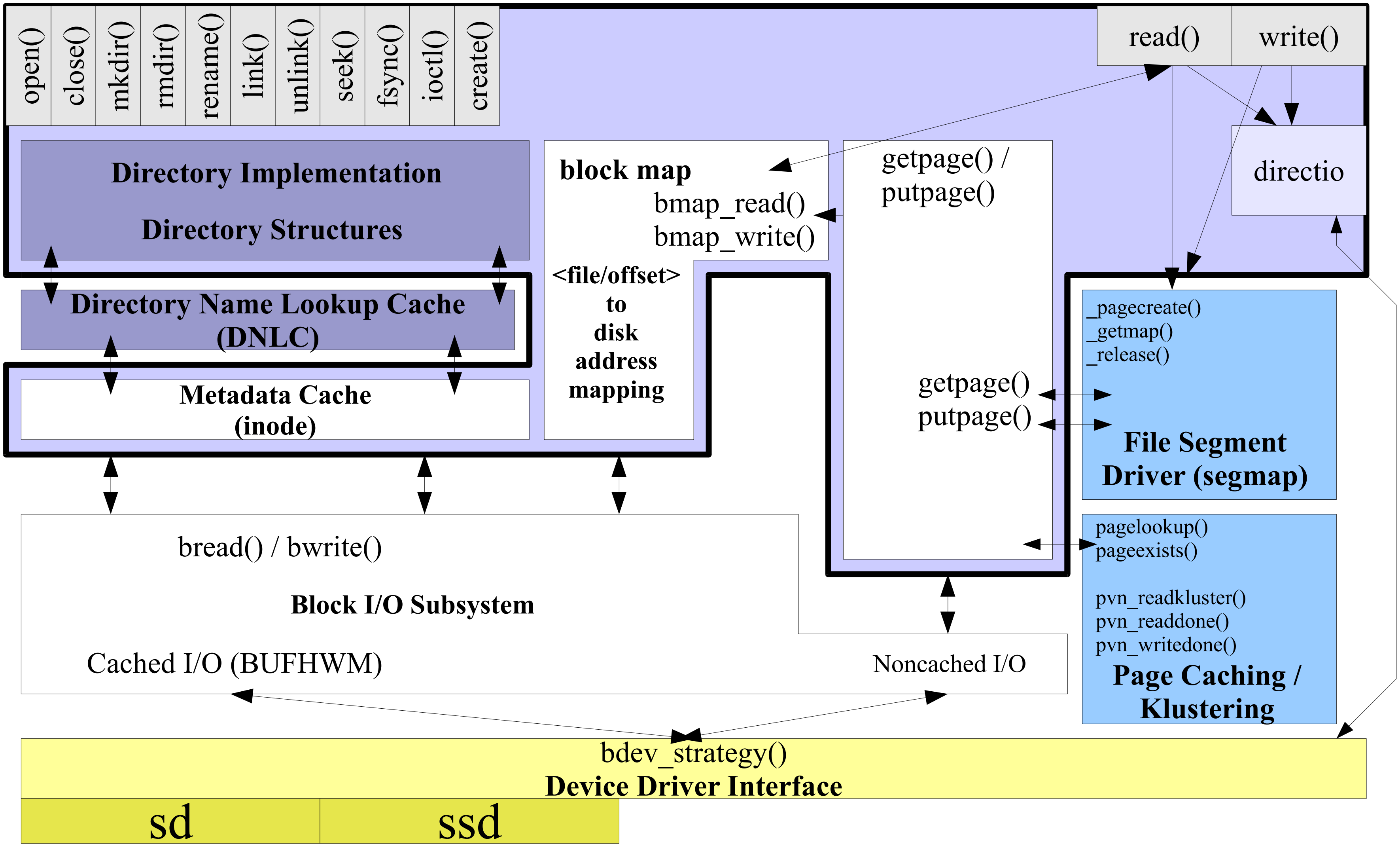
# Memory Mapped I/O

- Application maps file into process with mmap()
- Application references memory mapping
- If the page exists in the cache, we're done.
- If the page doesn't exist, a Major fault occurs
  - VM system invokes ufs_getpage()
  - UFS schedules a page size I/O for the page
  - When I/O is complete, data is copied to App.

# File System Implementation

- Read/write Operations
  - Read/write interface to file system page cache
  - File system page-in/page out functions handle real I/O
  - Block mapping for converting file/offset into disk device/offset via on-disk meta-data
- Directory Operations
  - Open/close
  - File system “lookup” path converts request for file/dir. Name into vnodes
  - Directories are stored as regular files, hold information with filename->vnode mapping
- Meta-data Operations
  - On-disk inodes store meta-data such as owner, mtime etc...

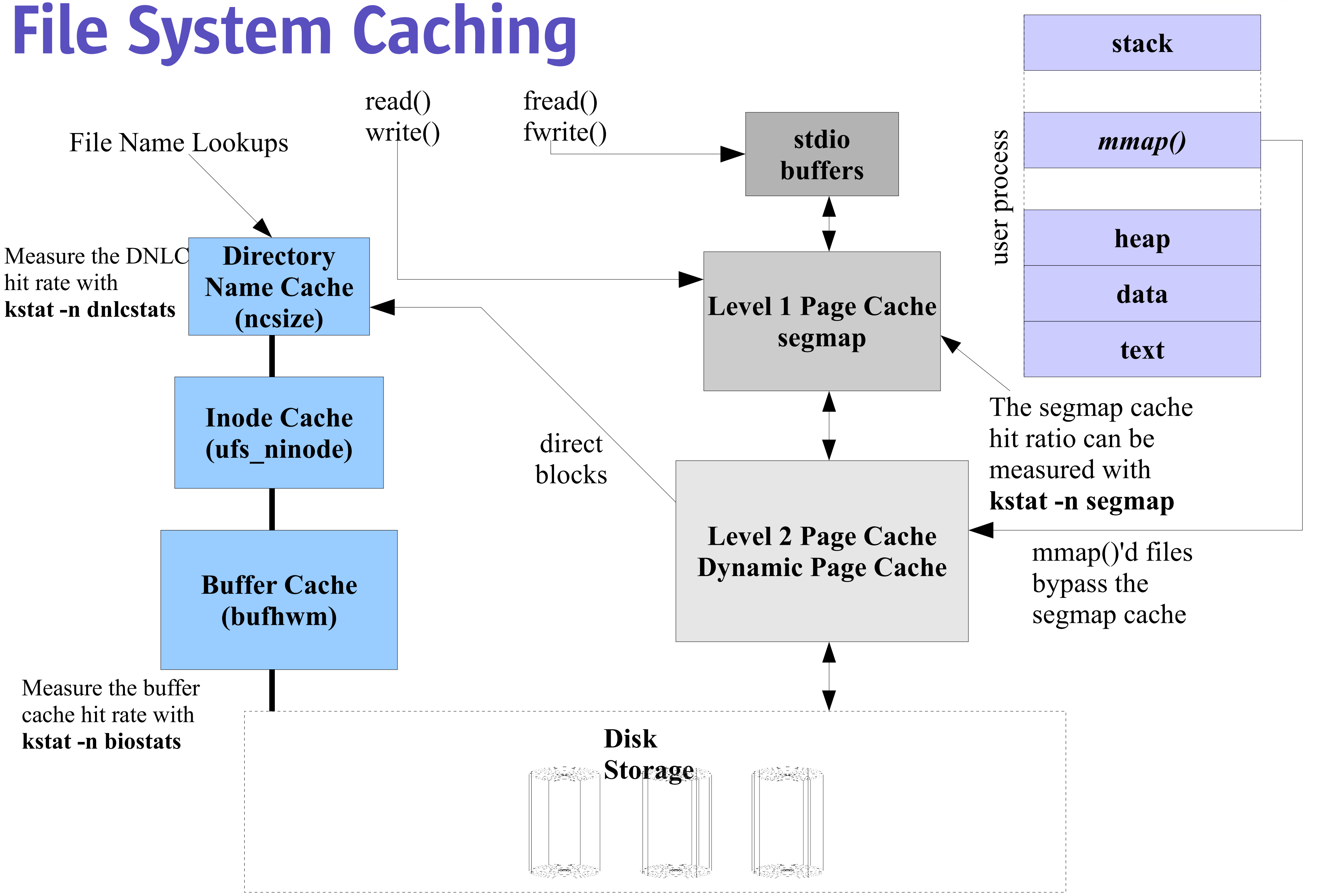
# Disk-based File System Architecture



# The big caches:

- File system/page cache
  - Holds the “data” of the files
- Buffer Cache
  - Holds the meta-data of the file system: direct/indirect blocks, inodes etc...
- Directory Name Cache
  - Caches mappings of filename->vnode from recent lookups
  - Prevents excessive re-reading of directory from disk
- File system specific: Inode cache
  - Caches inode meta-data in memory
  - Holds owner, mtimes etc

# File System Caching



# Direct I/O

- Introduced in Solaris 2.6
- Bypasses page cache
  - Zero copy: DMA from controller to user buffer
- Eliminate any paging interaction
  - No 8k block size I/O restriction
  - I/Os can be any multiple of 512 bytes
  - Avoids write breakup of O_SYNC writes
- But
  - No caching! Avoid unless application caches
  - No read ahead – application must do it's own
- Works on multiple file systems
  - UFS, NFS, VxFS, QFS

# Direct I/O

- Enabling direct I/O
  - Direct I/O is a global setting, per file or filesystem
  - Mount option

```
# mount -o forcedirectio /dev/dsk... /mnt
```

  - Library call

```
directio(fd, DIRECTIO_ON | DIRECTIO_OFF)
```

- Some applications can call directio(3c)
  - e.g. Oracle – see later slides

# Enabling Direct I/O

- Monitoring Direct I/O via directiostat
  - See http://www.solarisinternals.com/tools

```
# directiostat 3
  lreads lwrites  preads pwrites     Krd      Kwr holdrds  nflush
       0       0       0       0       0        0       0       0
       0       0       0       0       0        0       0       0
       0       0       0       0       0        0       0       0
```

```
lreads = logical reads to the UFS via directio

lwrites = logical writes to the UFS via directio

preads = physical reads to media

pwrites = physical writes to media

Krd = kilobytes read

Kwr = kilobytes written

nflush = number of cached pages flushed

holdrds = number of times the read was a "hole" in the file.
```

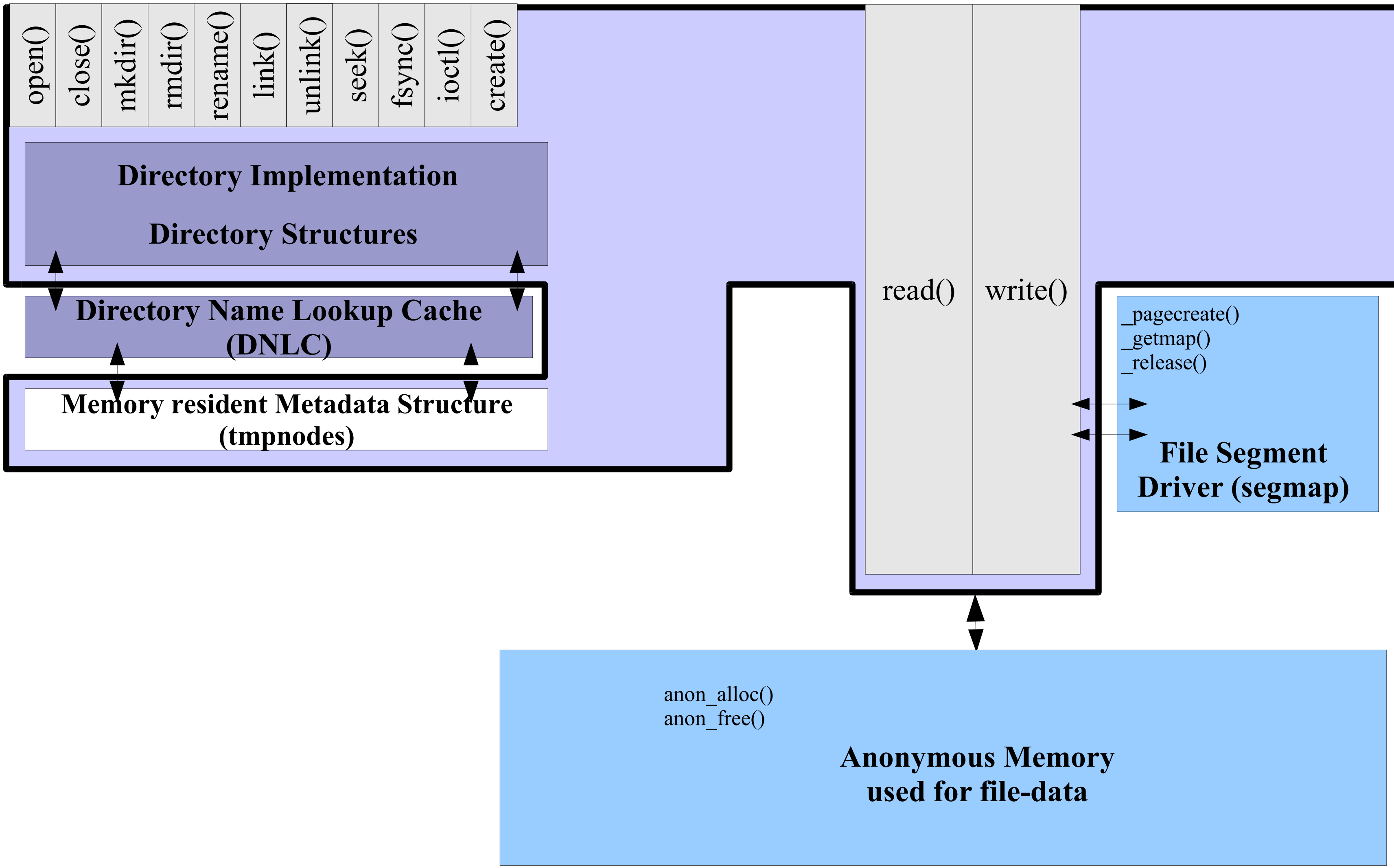
# Using Direct I/O

- Enable per-mount point is the simplest option
- Rememer it's a system wide setting
- Use sparingly, only applications which don't want caching will benefi
  - It disables caching, read ahead, write behind
  - e.g. Databases that have their own cache
  - e.g. Streaming high bandwidth in/out
- Check the side effects
  - Even though some applications can benefit, it may have side affects for others using the same files
    - e.g. Broken backup utils doing small I/O's will hurt due to lack of prefetch

# The TMPFS filesystem: A mountable RAM-Disk

- A RAM-file system
  - The file system equivalent of a RAM-DISK
  - Uses anonymous memory for file contents and meta-data
- Mounted on /tmp by default
- Other mounts can be created
  - See mount_tmpfs
- Practical Properties
  - Creating files in tmpfs uses RAM just like a process
  - Uses swap just like a process's anonymous memory
  - Overcommit will cause anon paging
- Best Practices
  - Don't put large files in /tmp
  - Configure an upper limit on /tmp space with “-osize=”

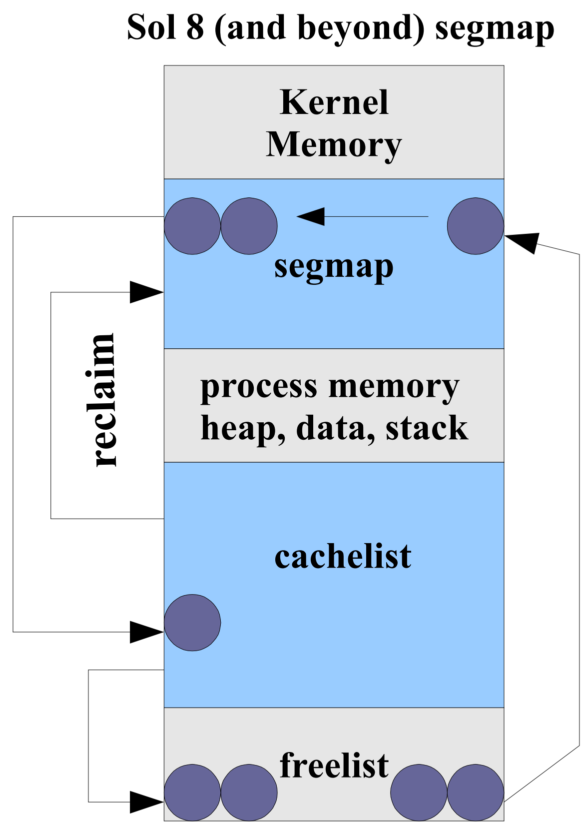
# TMPFS File System Architecture



# Lab: tmpfs

```
sol8# mount -F tmpfs swap /mnt
sol8# mkfile 100m /mnt/100m
sol9# mdb -k
> ::memstat
Page Summary                Pages                MB  %Tot
------------     ----------------  ----------------  ----
Kernel                      31592               123   12%
Anon                        59318               231   23%
Exec and libs               22786                89    9%
Page cache                  27626               107   11%
Free (cachelist)            77749               303   30%
Free (freelist)             38603               150   15%
Total                      257674              1006

sol8# umount /mnt
sol9# mdb -k
> ::memstat
Page Summary                Pages                MB  %Tot
------------     ----------------  ----------------  ----
Kernel                      31592               123   12%
Anon                        59311               231   23%
Exec and libs               22759                88    9%
Page cache                   2029                 7    1%
Free (cachelist)            77780               303   30%
Free (freelist)             64203               250   25%
Total                      257674              1006
```

# The Solaris 8 File System Cache



# Tuning segmap

- By default, segmap is sized at 12% of physical memory
  - Effectively sets the minimum amount of file system cache on the system by caching in segmap over and above the dynamically sized cachelist
- On Solaris 8/9
  - If the system memory is used primarily as a cache, cross calls (mpstat xcall) can be reduced by increasing the size of segmap via the system parameter segmap_percent (12 by default)
  - segmap_percent = 100 is like Solaris 7 without priority paging, and will cause a paging storm
  - Must keep segmap_percent at a reasonable value to prevent paging pressure on applications e.g. 50%

# Tuning segmap_percent

- There are kstat statistics for segmap hit rates
  - Estimate hit rate as (get_reclaim+get_use) / getmap

```
# kstat -n segmap
module: unix                            instance: 0
name:   segmap                          class:    vm

        crtime                          17.299814595
        fault                           17361
        faulta                          0
        free                            0
        free_dirty                      0
        free_notfree                    0
        get_nofree                      0
        get_reclaim                     67404
        get_reuse                       0
        get_unused                      0
        get_use                         83
        getmap                          71177
        pagecreate                      757
        rel_abort                       0
        rel_async                       3073
        rel_dontneed                    3072
        rel_free                        616
        rel_write                       2904
        release                         67658
        snaptime                        583596.778903492
```

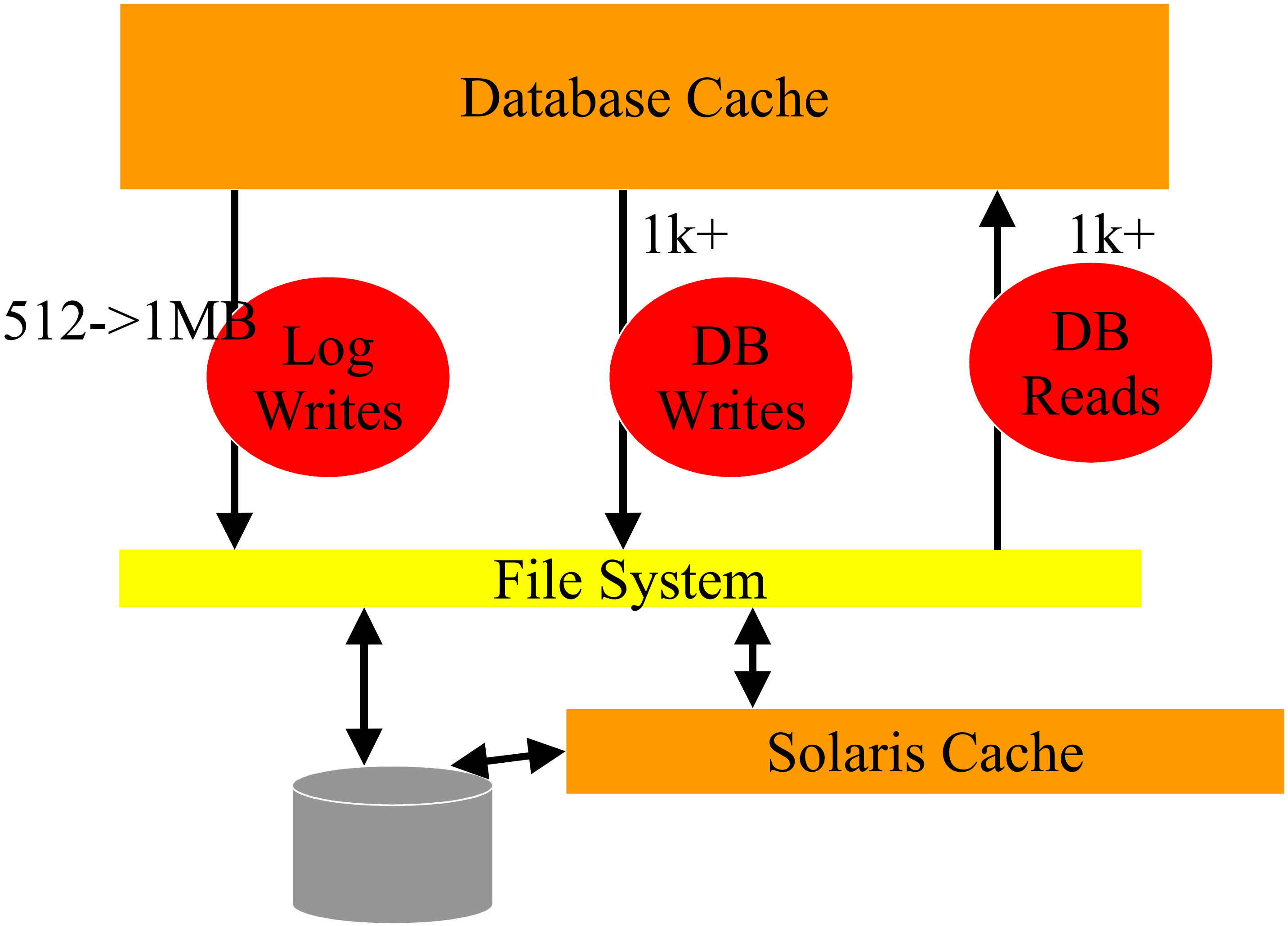
# UFS Access times

- Access times are updated when file is accessed or modified
  - e.g. A web server reading files will storm the disk with atime writes!
- Options allow atimes to be eliminated or deferred
  - dfratime: defer atime write until write
  - noatime: do not update access times, great for web servers and databases

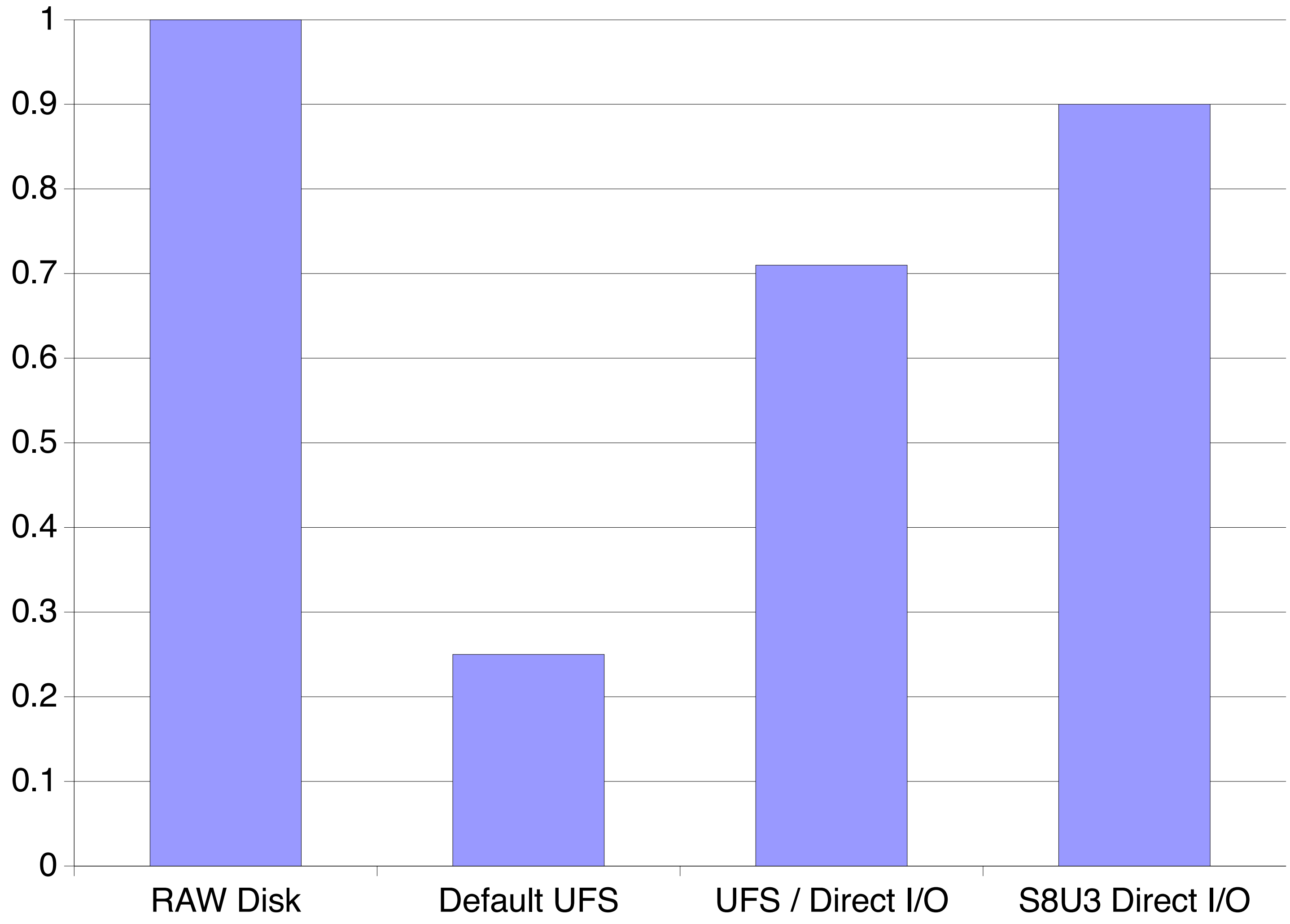
# Asynchronous I/O

- An API for single-threaded process to launch multiple outstanding I/Os
  - Multi-threaded programs could just just multiple threads
  - Oracle databases uses this extensively
  - See aio_read(), aio_write() etc...
- Slightly different varients for RAW disk vs file system
  - UFS, NFS etc: libaio creates lwp's to handle requests via standard pread/pwrite system calls
  - RAW disk: I/Os are passed into kernel via kaio(), and then managed via task queues in the kernel
    - Moderately faster than user-level LWP emulation

# Putting it all together: Database File I/O



# UFS is now Enhanced for Databases:



# Key UFS Features

- Direct I/O
  - Solaris 2.6+
- Logging
  - Solaris 7+
- Async I/O
  - Oracle 7.x, -> 8.1.5 - Yes
  - 8.1.7, 9i - New Option
- Concurrent Write Direct I/O
  - Solaris 8, 2/01

# Database big rules...

- Always put re-do logs on Direct I/O
- Cache as much as possible in the SGA
- Use 64-Bit RDBMS (Oracle 8.1.7+)
- Always use Asynch I/O
- Use Solaris 8 Concurrent Direct I/O
- Place as many tables as possible on Direct I/O, assuming SGA sized correct
- Place write-intensive tables on Direct I/O

# UFS write throttle

- UFS will block when there are too much pending dirty pages
  - Application writes by default go to memory, and are written asynchronously
  - Throttle blocks to prevent filling memory with async. Writes
- Solaris 8 Defaults
  - Block when 384k of unwritten cache
    - Set *ufs_HW*=<bytes>
  - Resume when 256k of unwritten cache
    - Set *ufs_LW*=<bytes>
- Solaris 9+ Defaults
  - Block when >16MB of unwritten cache
  - Resume when <8MB of unwritten cache

# Other items for Solaris UFS

- Solaris 8 Update 2/01
  - File system Snapshots
  - Enhanced logging w/ Direct I/O
  - Concurrent Direct I/O
  - 90% of RAW disk performance
  - Enhanced Directory Lookup
  - File create times in large directories significantly improved
  - Creating file systems
    - Faster newfs (1TB was ~20 hours)
- Solaris 9
  - Scalable Logging (for File Servers) 12/02
    - Postmark White paper
  - >1TB Filesystems (16TB) 8/03

# Solaris Volume Manager

- Solaris 9
  - Integration with live upgrade 5/03
  - >1TB Volumes 5/03
  - >1TB Devices/EFI Support 11/03
  - Dynamic Reconfiguration Support 11/03
- Future
  - Cluster Ready Volume Manager
  - Disk Set Migration: Import/Export
  - Volume Creation Service

# Volume Manager/FS Features

| Feature | Solaris | VxVM | VxFS |
|---|---|---|---|
| Online Unmount | Yes | | |
| Raid 0,1,5,1+0 | Yes | Yes | |
| Logging/No FSCK | Sol 7 | | Yes |
| Soft Partitions | Sol 8 | Yes | |
| Device Path Independence | Sol 8 | Yes | |
| Database Performance | Sol 8 2/02 | | QuickIO |
| Integration with Install | Sol 9 | | |
| Multi-Pathing | Sol 9 | Yes/DMP | |
| Grow Support | Sol 9 | Yes | Yes |
| Fast Boot | Sol 9 | | |
| Integration with LU | Sol 9 5/03 | | |
| >1TB Volumes | Sol 9 5/03 | 3.5 | |
| >1TB Filesystems | Sol 9 8/03 | | 3.5/VxVM |
| >1TB Devices/EFI Support | Sol 9 8/03 | | |
| Dynamic Reconfiguration Integration | Sol 9 8/03 | | |
| Cluster Ready Volume Manager | Future | VxCVM | |
| Disk Group Migration: Import/Export | Future | Yes | |

# Summary

- Solaris continues to evolve in both performance and resource management innovations
- Observability tools and utilities continue to get better
- Resource management facilities providing for improved overall system utilization and SLA management

# Resources

- **http://www.solarisinternals.com**
- **http://www.sun.com/solaris**
- **http://www.sun.com/blueprints**
- **http://www.sun.com/bigadmin**
- **http://docs.sun.com**
  - "What's New in the Solaris 9 Operating Environment"
- **http://sdc.sun.com/solaris8**
- **http://sun.com/solaris/fcc/lifecycle.html**

# Thank You!

# Questions?

# Solaris Kernel
# Performance, Observability & Debugging
# Day 2

# Agenda – Day 1

- Session 1 - 9:00AM to 10:30PM
  - Goals, non goals and assumptions
  - Solaris Kernel Overview & Features
  - Observability & Tracing Tools & Utilities
- Session 2 - 11:00PM to 12:30PM
  - Memory
    - Virtual Memory
    - Physical Memory
    - Memory dynamics
    - Performance and Observability
    - Memory Resource Management

# Agenda – Day 1 (cont)

- Session 3 - 2:00PM to 3:30PM
  - Processes, threads & scheduling
    - The Solaris Multhreaded Process Model
    - The Dispatcher & Scheduling Classes
    - Performance & Observability
    - Processor Controls and Binding
- Session 4 - 4:00PM to 5:30PM
  - File Systems and I/O
    - I/O Overview
    - The Solaris VFS/Vnode Model
    - UFS – The Solaris Unix File System
    - Performance & Observability

# Agenda – Day 2

- Session 1 - 9:00AM to 10:30PM
  - DTrace
    - A Deeper Dive
  - A System View
  - Traps & Interrupts
- Session 2 - 11:00PM to 12:30PM
  - Advanced Memory Topics
    - Memory monitoring and measuring
    - Utilizing and tuning large memory

# Agenda – Day 2 (cont)

- Session 3 - 2:00PM to 3:30PM
  - Processes, threads & scheduling
    - A Deeping Dive
    - The Runtime Linker
    - Watching Processes with Dtrace
    - Process/Thread Lab
- Session 4 - 4:00PM to 5:30PM
  - Disk I/O Performance
  - File System Performance
  - Network Attached Storage
  - File System Performance Characterization
  - ZFS
  - Resource Management
  - Large System Performance

# Performance & Observability Tools, Day 2

# Solaris Performance and Tracing Tools

## Process stats

- cputrack - per-processor hw counters
- pargs – process arguments
- pflags – process flags
- pcred – process credentials
- pldd – process's library dependencies
- psig – process signal disposition
- pstack – process stack dump
- pmap – process memory map
- pfiles – open files and names
- prstat – process statistics
- ptree – process tree
- ptime – process microstate times
- pwdx – process working directory

## Process control

- pgrep – grep for processes
- pkill – kill processes list
- pstop – stop processes
- prun – start processes
- prctl – view/set process resources
- pwait – wait for process
- preap – reap a zombie process

## Process Tracing/ debugging

- abitrace – trace ABI interfaces
- dtrace – trace the world
- mdb – debug/control processes
- truss – trace functions and system calls

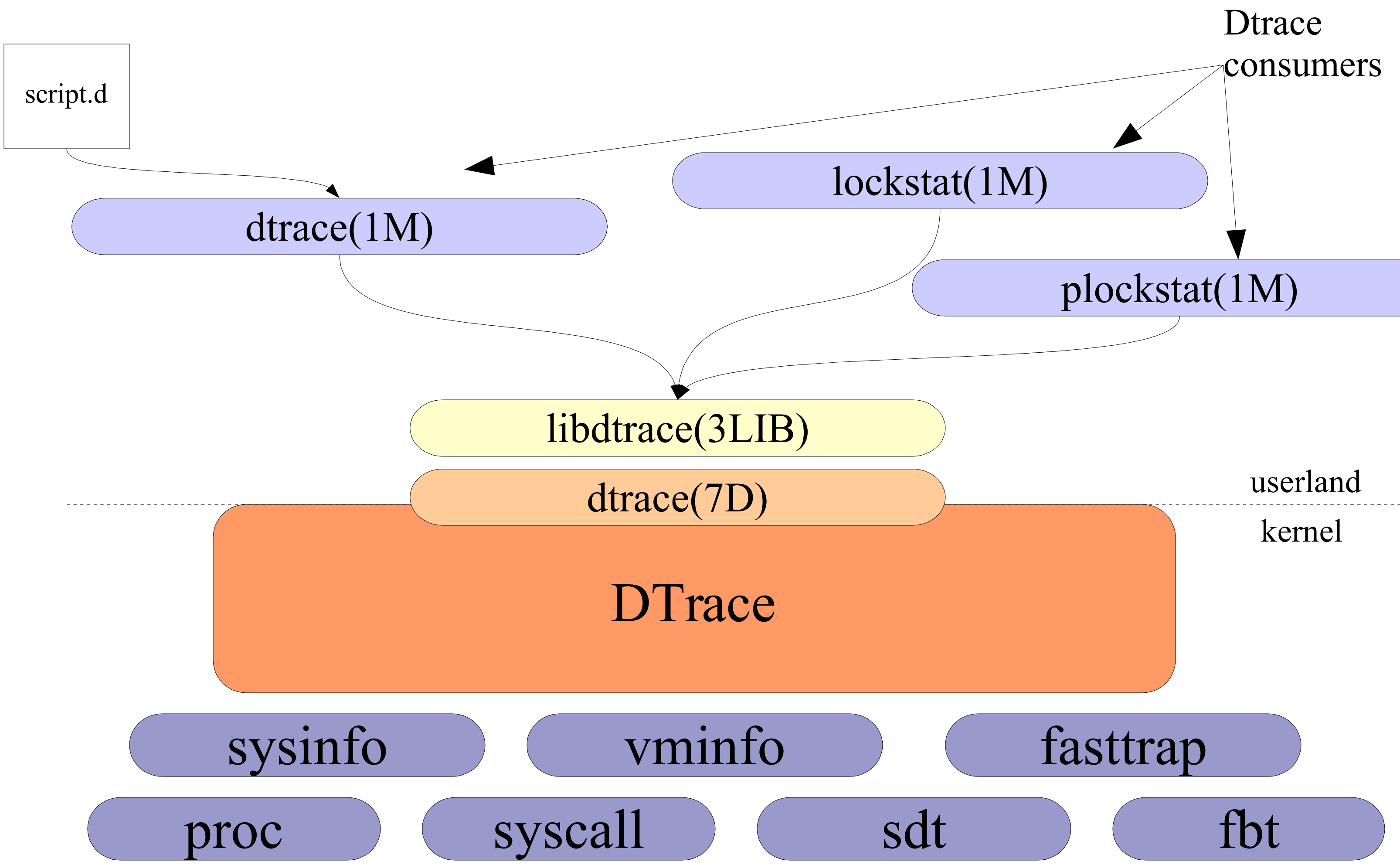
## Kernel Tracing/ debugging

- dtrace – trace and monitor kernel
- lockstat – monitor locking statistics
- lockstat -k – profile kernel
- mdb – debug live and kernel cores

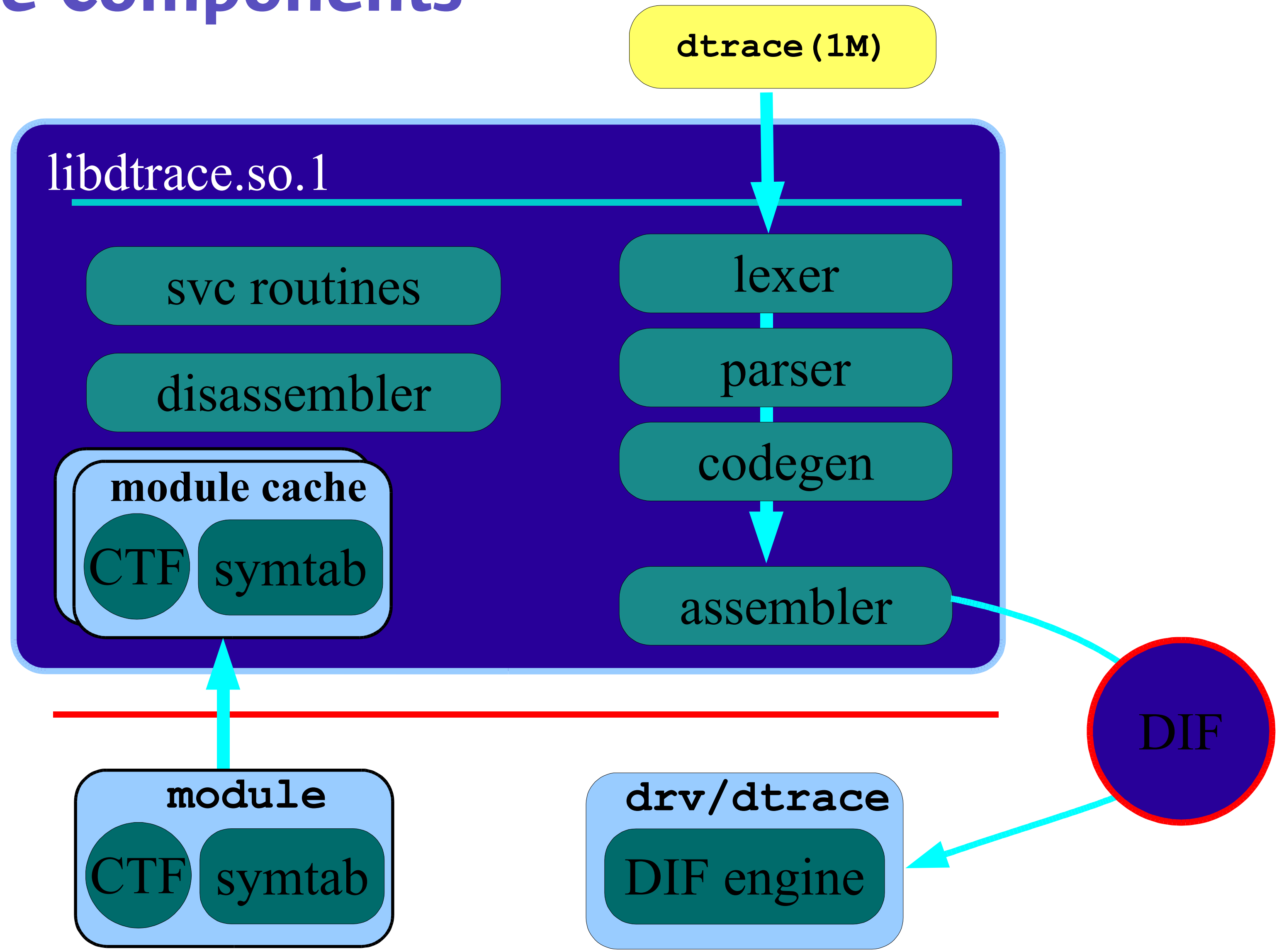
## System Stats

- acctcom – process accounting
- busstat – Bus hardware counters
- cpustat – CPU hardware counters
- iostat – IO & NFS statistics
- kstat – display kernel statistics
- mpstat – processor statistics
- netstat – network statistics
- nfsstat – nfs server stats
- sar – kitchen sink utility
- vmstat – virtual memory stats

# Dtrace – The Big Picture



# DTrace Components



# DTrace Probes

- A *probe* is a point of instrumentation
- A probe is made available by a *provider*
- Each probe identifies the *module* and *function* that it instruments
- Each probe has a *name*
- These four attributes define a tuple that uniquely identifies each probe
- Each probe is assigned an integer identifier

# DTrace Providers

- A provider represents a methodology for instrumenting the system
- Providers make probes available to the DTrace framework
- DTrace informs providers when a probe is to be enabled
- Providers transfer control to DTrace when an enabled probe is hit

# DTrace Providers, cont.

- DTrace has quite a few providers, e.g.:
  - The *function boundary tracing (FBT)* provider can dynamically instrument every function entry and return in the kernel
  - The *syscall* provider can dynamically instrument the system call table
  - The *lockstat* provider can dynamically instrument the kernel synchronization primitives
  - The *profile* provider can add a configureable-rate profile interrupt of to the system

# DTrace Consumers

- A DTrace consumer is a process that interacts with DTrace
- No limit on concurrent consumers; DTrace handles the multiplexing
- Some programs are DTrace consumers only as an implementation detail
- dtrace(1M) is a DTrace consumer that acts as a generic front-end to the DTrace facility

# Listing probes

- Probes can be listed with the "-l" option to dtrace (1M)
- Can list probes
  - in a specific function with "-f *function*"
  - in a specific module with "-m *module*"
  - with a specific name with "-n *name*"
  - from a specific provider with "-P *provider*"
- For each probe, provider, module, function and name are displayed

# Fully specifying probes

- To specify multiple components of a probe tuple, separate the components with a colon
- Empty components match anything
- For example, "syscall::open:entry" specifies a probe:
  - from the "syscall" provider
  - in any module
  - in the "open" function
  - named "entry"

# Enabling probes

- Probes are enabled by specifying them without the “-l” option
- When enabled in this way, probes are enabled with the *default action*
- The default action will indicate only that the probe fired; no other data will be recorded
- For example, “dtrace -m nfs” enables every probe in the “nfs” module

# Actions

- *Actions* are taken when a probe fires
- Actions are completely programmable
- Most actions *record* some specified state in the system
- Some actions *change* the state of the system system in a well-defined way
  - These are called *destructive actions*
  - Disabled by default
- Many actions take as parameters expressions in the *D language*

# The D language

- D is a C-like language specific to DTrace, with some constructs similar to awk(1)
- Complete access to kernel C types
- Complete access to statics and globals
- Complete support for ANSI-C operators
- Support for strings as first-class citizen
- We'll introduce D features as we need them...

# Built-in D variables

- For now, our D expressions will consist only of built-in variables
- Example of built-in variables:
  - `pid` is the current process ID
  - `execname` is the current executable name
  - `timestamp` is the current value of a nanosecond counter
  - `vtimestamp` is the same as timestamp, except virtualized to the on CPU cycle time of the thread
  - `curthread` is a pointer to the `kthread_t` structure that represents the current thread
  - `probemod, probefunc` and `probename` are the current probe's module, function and name

# Actions: "`trace`"

- `trace()` records the result of a D expression to the trace buffer
- For example:
  - `trace(pid)` traces the current process ID
  - `trace(execname)` traces the name of the current executable
  - `trace(curthread->t_pri)` traces the `t_pri` field of the current thread
  - `trace(probefunc)` traces the function name of the probe

# Actions, cont.

- Actions are indicated by following a probe specification with "{ *action* }"
- For example:

```
dtrace -n 'readch{trace(pid)}'
dtrace -m 'ufs{trace(execname)}'
dtrace -n 'syscall:::entry {trace
   (probefunc)}'
```

- Multiple actions can be specified; they must be separated by semicolons:

```
dtrace -n 'xcalls{trace(pid); trace
   (execname)}'
```

# D Scripts

- Complicated DTrace enablings become difficult to manage on the command line
- dtrace(1M) supports *scripts*, specified with the "-s" option
  - `dtrace -s script.d`
- Alternatively, executable DTrace interpreter files may be created
- Interpreter files always begin with:

```
#!/usr/sbin/dtrace -s
```

# D Scripts, cont.

- For example, a script to trace the executable name upon entry of each system call:

```
#!/usr/sbin/dtrace -s

syscall:::entry
{
        trace(execname);
}
```

# Predicates

- *Predicates* allow actions to only be taken when certain conditions are met
- A predicate is a D expression
- Actions will only be taken if the predicate expression evaluates to true
- A predicate takes the form "/*expression*/" and is placed between the probe description and the action

# Predicates, cont.

- For example, tracing the pid of every process named “date” that performs an open(2):

```
#!/usr/sbin/dtrace -s

syscall::open:entry
/execname == "date"/
{
        trace(pid);
}
```

# Actions: More actions

- `tracemem()` records memory at a specified location for a specified length
- `stack()` records the current *kernel* stack trace
- `ustack()` records the current *user* stack trace
- `exit()` tells the DTrace consumer to exit with the specified status

# Actions: Destructive actions

- Must specify “-w” option to DTrace
- `stop()` stops the current process
- `raise()` sends a specified signal to the current process
- `breakpoint()` triggers a kernel breakpoint
- `panic()` induces a kernel panic
- `chill()` spins for a specified number of nanoseconds

# Output formatting

- The `printf()` function combines the `trace` action with the ability to precisely control output
- `printf` takes a printf(3C)-like format string as an argument, followed by corresponding arguments to print
- e.g.:

```
printf("%d was here", pid);
printf("I am %s", execname);
```

# Output formatting, cont.

- Normally, dtrace(1M) provides details on the firing probe, plus any explicitly traced data
- Use the quiet option ("-q") to `dtrace`(1M) to supress the probe details
- The quiet option may also be set in a D script by embedding:

```
#pragma D option quiet
```

# Global D variables

- D allows you to define your own variables that are global to your D program
- Like awk(1), D tries to infer variable type upon instantiation, obviating an explicit variable declaration
  - But you can still declare variables if you want to...

```
#!/usr/sbin/dtrace -s

int x;
dtrace:::BEGIN
{
    x = 123;
}
```

# Global D variables, cont.

- Example:

```
#!/usr/sbin/dtrace -s

#pragma D option quiet

sysinfo:::zfod
{
        zfods++;
}

profile:::tick-1sec
{
        printf(“%d zfods\n”,  zfods);
        zfods = 0;
}
```

# Thread-local D variables

- D allows for *thread-local* variables
- A thread-local variable has the same name – but disjoint data storage – for each thread
- By definition, thread-local variables elminate the race conditions that are endemic to global variables
- Denoted by prepending “`self->`” to the variable name

# Thread-local D variables, cont

- Thread-local variables that have never been assigned in the current thread have the value zero
- Underlying thread-local storage for a thread-local variable is deallocated by assigning zero to it

# Thread-local D variables, cont.

- Example 1:

```
#!/usr/sbin/dtrace -s

#pragma D option quiet

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

syscall::poll:return
/self->ts && timestamp - self->ts > 1000000000/
{
        printf("%s polled for %d seconds\n", execname,
            (timestamp - self->ts) / 1000000000);
        self->ts = 0;
}
```

# Thread-local D variables, cont.

- Example 2:

```
syscall::ioctl:entry
/execname == “date”/
{
        self->follow = 1;
}

fbt:::
/self->follow/
{}

syscall::ioctl:return
/self->follow/
{
       self->follow = 0;
}
```

# Aggregations

- When trying to understand suboptimal performance, one often looks for *patterns* that point to bottlenecks
- When looking for patterns, one often doesn't want to study each datum – one wishes to *aggregate* the data and look for larger trends
- Traditionally, one has had to use conventional tools (e.g. awk(1), perl(1))

# Aggregations, cont.

- DTrace supports the aggregation of data as a first class operation
- An *aggregating function* is a function f(x), where x is a set of data, such that:

  $$f(f(x_0) \quad f(x_1) \quad \ldots \quad f(x_n)) = f(x_0 \quad x_1 \quad \ldots \quad x_n)$$

- E.g., COUNT, SUM, MAXIMUM, and MINIMUM are aggregating functions; MEDIAN, and MODE are not

# Aggregations, cont.

- An *aggregation* is the result of an aggregating function keyed by an arbitrary tuple
- For example, to count all system calls on a system by system call name:

```
dtrace -n 'syscall:::entry \
  { @syscalls[probefunc] = count(); }'
```

- By default, aggregation results are printed when `dtrace`(1M) exits

# Aggregations, cont.

- Aggregations need not be named
- Aggregations can be keyed by more than one expression
- For example, to count all ioctl system calls by both executable name and file descriptor:

```
dtrace -n 'syscall::ioctl:entry \
   { @[execname, arg0] = count(); }'
```

# Aggregations, cont.

- Some other aggregating functions:
  - `avg()`: the average of specified expressions
  - `min()`: the minimum of specified expressions
  - `max()`: the maximum of specified expressions
  - `quantize()`: power-of-two distribution of specified expressions
- For example, distribution of write(2) sizes by executable name:

```
dtrace -n 'syscall::write:entry \
  { @[execname] = quantize(arg2); }'
```

# Allowing dtrace for non-root users

- Setting dtrace privileges

```
Add a line for your user in /etc/user_attr:

rmc::::defaultpriv=dtrace_kernel,basic,proc_owner,dtrace_proc
```

# A System View

# mpstat(1)

```
solaris10> mpstat 2
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    3   0   10   345  219   44    0    1    3    0    28    0   0   0  99
  1    3   0    5    39    1   65    1    2    1    0    23    0   0   0 100
  2    3   0    3    25    5   22    1    1    2    0    25    0   1   0  99
  3    3   0    3    19    0   27    1    2    1    0    22    0   0   0  99
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    4   0 11565 14115  228 7614 1348 2732 3136 1229 255474   10  28   0  61
  1    0   0 10690 14411   54 7620 1564 2546 2900 1182 229899   10  28   0  63
  2    0   0 10508 14682    6 7714 1974 2568 2917 1222 256806   10  29   0  60
  3    0   0 9438 14676    0 7284 1582 2362 2622 1126 249150   10  30   0  60
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0   0 11570 14229  224 7608 1278 2749 3218 1251 254971   10  28   0  61
  1    0   0 10838 14410   63 7601 1528 2669 2992 1258 225368   10  28   0  62
  2    0   0 10790 14684    6 7799 2009 2617 3154 1299 231452   10  28   0  62
  3    0   0 9486 14869    0 7484 1738 2397 2761 1175 237387   10  28   0  62
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0   0 10016 12580  224 6775 1282 2417 2694  999 269428   10  27   0  63
  1    0   0 9475 12481   49 6427 1365 2229 2490  944 271428   10  26   0  63
  2    0   0 9184 12973    3 6812 1858 2278 2577  985 231898    9  26   0  65
  3    0   0 8403 12849    0 6382 1428 2051 2302  908 239172    9  25   0  66
...
```

# prstat(1)

```
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 21487 root      603M   87M sleep   29   10   0:01:50  35% filebench/9
 21491 morgan   4424K 3900K cpu2    59    0   0:00:00 0.0% prstat/1
   427 root       16M   16M sleep   59    0   0:08:40 0.0% Xorg/1
 21280 morgan   2524K 1704K sleep   49    0   0:00:00 0.0% bash/1
 21278 morgan   7448K 1888K sleep   59    0   0:00:00 0.0% sshd/1
   489 root       12M 9032K sleep   59    0   0:03:05 0.0% dtgreet/1
 21462 root      493M 3064K sleep   59    0   0:00:01 0.0% filebench/2
   209 root     4132K 2968K sleep   59    0   0:00:13 0.0% inetd/4
   208 root     1676K  868K sleep   59    0   0:00:00 0.0% sac/1
   101 root     2124K 1232K sleep   59    0   0:00:00 0.0% syseventd/14
   198 daemon   2468K 1596K sleep   59    0   0:00:00 0.0% statd/1
   113 root     1248K  824K sleep   59    0   0:00:00 0.0% powerd/2
   193 daemon   2424K 1244K sleep   59    0   0:00:00 0.0% rpcbind/1
   360 root     1676K  680K sleep   59    0   0:00:00 0.0% smcboot/1
   217 root     1760K  992K sleep   59    0   0:00:00 0.0% ttymon/1
Total: 48 processes, 160 lwps, load averages: 1.32, 0.83, 0.43
```

# prstat(1) – Threads

```
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/LWPID
 21495 root      603M   86M sleep   11   10   0:00:03 2.8% filebench/4
 21495 root      603M   86M sleep    3   10   0:00:03 2.8% filebench/3
 21495 root      603M   86M sleep   22   10   0:00:03 2.8% filebench/7
 21495 root      603M   86M sleep   60   10   0:00:03 2.7% filebench/5
 21495 root      603M   86M cpu1    21   10   0:00:03 2.7% filebench/8
 21495 root      603M   86M sleep   21   10   0:00:03 2.7% filebench/2
 21495 root      603M   86M sleep   12   10   0:00:03 2.7% filebench/9
 21495 root      603M   86M sleep   60   10   0:00:03 2.6% filebench/6
 21462 root      493M 3064K sleep   59    0   0:00:01 0.1% filebench/1
 21497 morgan   4456K 3924K cpu0    59    0   0:00:00 0.0% prstat/1
 21278 morgan   7448K 1888K sleep   59    0   0:00:00 0.0% sshd/1
   427 root       16M   16M sleep   59    0   0:08:40 0.0% Xorg/1
 21280 morgan   2524K 1704K sleep   49    0   0:00:00 0.0% bash/1
   489 root       12M 9032K sleep   59    0   0:03:05 0.0% dtgreet/1
   514 root     3700K 2812K sleep   59    0   0:00:02 0.0% nscd/14
Total: 48 processes, 159 lwps, load averages: 1.25, 0.94, 0.51
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/LWPID
 21495 root      603M   86M run     60   10   0:00:04 3.2% filebench/8
 21495 root      603M   86M run     59   10   0:00:04 3.2% filebench/4
 21495 root      603M   86M cpu3    59   10   0:00:04 3.1% filebench/7
 21495 root      603M   86M cpu1    22   10   0:00:04 3.1% filebench/9
 21495 root      603M   86M cpu2    59   10   0:00:04 3.0% filebench/2
 21495 root      603M   86M sleep    1   10   0:00:04 3.0% filebench/3
 21495 root      603M   86M sleep    1   10   0:00:04 3.0% filebench/6
 21495 root      603M   86M run      3   10   0:00:04 3.0% filebench/5
 21462 root      493M 3064K sleep   59    0   0:00:01 0.1% filebench/1
 21497 morgan   4828K 4232K cpu0    59    0   0:00:00 0.0% prstat/1
 21278 morgan   7448K 1888K sleep   59    0   0:00:00 0.0% sshd/1
   427 root       16M   16M sleep   59    0   0:08:40 0.0% Xorg/1
 21280 morgan   2524K 1704K sleep   59    0   0:00:00 0.0% bash/1
   489 root       12M 9032K sleep   59    0   0:03:05 0.0% dtgreet/1
   514 root     3700K 2812K sleep   59    0   0:00:02 0.0% nscd/14
Total: 48 processes, 159 lwps, load averages: 1.28, 0.95, 0.51
```

# prstat(1) - Microstates

```
   PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID
 21495 root     6.1  15 0.0 0.0 0.0  51  26 1.9 11K  4K .7M   0 filebench/7
 21495 root     5.7  14 0.0 0.0 0.0  53  26 1.7  9K  4K .6M   0 filebench/3
 21495 root     5.4  13 0.1 0.0 0.0  54  26 1.8 10K  4K .6M   0 filebench/5
 21495 root     5.2  13 0.0 0.0 0.0  54  26 1.8  9K  4K .6M   0 filebench/4
 21495 root     5.2  13 0.0 0.0 0.0  55  26 1.7  9K  4K .6M   0 filebench/6
 21495 root     4.7  12 0.0 0.0 0.0  56  25 1.8  9K  4K .5M   0 filebench/9
 21495 root     4.4  11 0.0 0.0 0.0  57  26 1.6  8K  3K .5M   0 filebench/8
 21495 root     4.1  11 0.0 0.0 0.0  58  26 1.6  7K  3K .4M   0 filebench/2
 21499 morgan   0.0 0.1 0.0 0.0 0.0 0.0 100 0.0  17   2 311   0 prstat/1
   427 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  18   4  72   9 Xorg/1
   489 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  26   1  45   0 dtgreet/1
   471 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   2   2   6   0 snmpd/1
     7 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  15   0   5   0 svc.startd/6
 21462 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  13   0   5   0 filebench/2
   514 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  15   0  47   0 nscd/23
Total: 48 processes, 159 lwps, load averages: 1.46, 1.03, 0.56
   PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID
 21495 root     5.3  14 0.0 0.0 0.0  51  28 1.7  9K  3K .6M   0 filebench/2
 21495 root     5.1  14 0.0 0.0 0.0  51  28 1.9 10K  4K .5M   0 filebench/3
 21495 root     5.3  13 0.1 0.0 0.0  51  28 1.8 10K  4K .6M   0 filebench/8
 21495 root     5.2  13 0.0 0.0 0.0  51  28 1.9 10K  4K .6M   0 filebench/4
 21495 root     5.0  13 0.1 0.0 0.0  52  28 2.0 10K  4K .5M   0 filebench/5
 21495 root     4.9  12 0.0 0.0 0.0  52  29 1.8  9K  4K .5M   0 filebench/6
 21495 root     4.7  12 0.0 0.0 0.0  53  28 1.8  9K  4K .5M   0 filebench/9
 21495 root     4.8  12 0.0 0.0 0.0  52  29 1.8  9K  4K .5M   0 filebench/7
 21499 morgan   0.0 0.1 0.0 0.0 0.0 0.0 100 0.0  18   9 184   0 prstat/1
   427 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  20   0  80  10 Xorg/1
   489 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  28   0  50   0 dtgreet/1
     7 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  15   0   5   0 svc.startd/6
 21462 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  15   0   5   0 filebench/2
   492 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   3   0  10   0 sendmail/1
     9 root     0.0 0.0 0.0 0.0 0.0 100 0.0 0.0   3   0   1   0 svc.configd/14
Total: 48 processes, 159 lwps, load averages: 1.47, 1.04, 0.56
```

# DTrace – Getting Below The Numbers syscalls

```
solaris10> mpstat 2
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0   0 15078 18098  223 10562 3172 3982 3134 1848 187661    9  35   0  56
  1    0   0 13448 16972   61 8849 1539 3407 2931 1777 231317   10  36   0  54
  2    0   0 12031 17263    6 8695 1467 3325 2854 1738 241761   11  34   0  55
  3    0   0 11051 17694    1 8399 1509 3096 2546 1695 248747   10  35   0  55
^C
solaris10> dtrace -n 'syscall:::entry { @[probefunc]=count() }'
dtrace: description 'syscall:::entry ' matched 229 probes
^C

  fstat                                                             1
 ...
  setcontext                                                       10
  setitimer                                                        18
  p_online                                                         21
  lwp_sigmask                                                      22
  lwp_park                                                         29
  pollsys                                                          41
  ioctl                                                           157
  yield                                                          2991
  unlink                                                         3586
  xstat                                                          3588
  write                                                          4212
  open64                                                        10762
  close                                                         10762
  llseek                                                        11374
  read                                                          21543
  pread                                                         78918
  lwp_mutex_timedlock                                          578710
  lwp_mutex_unlock                                             578711
```

# Dtrace – Getting Below The Numbers xcalls

```
# dtrace -n 'xcalls { @[probefunc] = count() }'
dtrace: description 'xcalls ' matched 3 probes
^C

  send_one_mondo                                                  346343
#
```

```
# cat xcalls.d
#!/usr/sbin/dtrace -s

send_one_mondo:xcalls
{
        @s[stack(20)] = count();
}

END
{
        printa(@s);
}
#
```

# Dtrace - xcalls

```
  ...

  SUNW,UltraSPARC-II`send_one_mondo+0x20
  SUNW,UltraSPARC-II`send_mondo_set+0x1c
  unix`xt_some+0xc4
  unix`xt_sync+0x3c
  unix`hat_unload_callback+0x6ec
  unix`bp_mapout+0x74
  genunix`biowait+0xb0
  ufs`ufs_putapage+0x3f4
  ufs`ufs_putpages+0x2a4
  genunix`segmap_release+0x300
  ufs`ufs_dirremove+0x638
  ufs`ufs_remove+0x150
  genunix`vn_removeat+0x264
  genunix`unlink+0xc
  unix`syscall_trap+0xac
17024

  SUNW,UltraSPARC-II`send_one_mondo+0x20
  SUNW,UltraSPARC-II`send_mondo_set+0x1c
  unix`xt_some+0xc4
  unix`sfmmu_tlb_range_demap+0x190
  unix`hat_unload_callback+0x6d4
  unix`bp_mapout+0x74
  genunix`biowait+0xb0
  ufs`ufs_putapage+0x3f4
  ufs`ufs_putpages+0x2a4
  genunix`segmap_release+0x300
  ufs`ufs_dirremove+0x638
  ufs`ufs_remove+0x150
  genunix`vn_removeat+0x264
  genunix`unlink+0xc
  unix`syscall_trap+0xac
17025
```

# lockstat(1M)

- Provides for kernel lock statistics (mutex locks, reader/writer locks)
- Also serves as a kernel profiling tool
- Use “-i 971” for the interval to avoid collisions with the clock interrupt, and gather fine-grained data

```
#lockstat -i 971 sleep 300 > lockstat.out

#lockstat -i 971 -I sleep 300 > lockstatI.out
```

# Lock Statistics – mpstat

```
# mpstat 1
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  8    0   0 6611   456  300 1637    7   26 1110    0   135   33  45   2  21
  9    1   0 1294   250  100 2156    3   29 1659    0    68    9  63   0  28
 10    0   0 3232   308  100 2357    2   36 1893    0   104    2  66   2  30
 11    0   0  647   385  100 1952    1   19 1418    0    21    4  83   0  13
 12    0   0  190   225  100  307    0    1  589    0     0    0  98   0   2
 13    0   0  624   373  100 1689    2   14 1175    0    87    7  80   2  12
 14    0   0  392   312  100 1810    1   12 1302    0    49    2  80   2  15
 15    0   0  146   341  100 2586    2   13 1676    0     8    0  82   1  17
 16    0   0  382   355  100 1968    2    7 1628    0     4    0  88   0  12
 17    0   0   88   283  100  689    0    4  474    0    95    1  94   2   3
 18    0   0 3571   152  104  568    0    7 2007    0    15    0  93   1   6
 19    0   0 3133   278  100 2043    2   24 1307    0   113    7  69   1  22
 20    0   0  385   242  127 2127    2   22 1296    0    36    0  73   0  26
 21    0   0  152   369  100 2259    0   10 1400    0   140    2  84   2  12
 22    0   0 3964   241  120 1754    3   25 1085    0    91   11  62   1  26
 23    0   2  555   193  100 1827    2   23 1148    0   288    7  64   7  22
 24    0   0  811   245  113 1327    2   23 1228    0   110    3  76   4  17
 25    0   0  105   500  100 2369    0   11 1736    0     6    0  88   0  11
 26    0   0  163   395  131 2383    2   16 1487    0    64    2  79   1  18
 27    0   1  718  1278 1051 2073    4   23 1311    0   237    9  67   6  19
 28    0   0  868   271  100 2287    4   27 1309    0   139    9  55   0  36
 29    0   0  931   302  103 2480    3   29 1569    0   165    9  66   2  23
 30    0   0 2800   303  100 2146    2   13 1266    0   152   11  70   3  16
 31    0   1 1778   320  100 2368    2   24 1381    0   261   11  56   5  28
```

# Examining Adaptive Locks Excessive Spinning

```
# lockstat sleep 10

Adaptive mutex spin: 293311 events in 10.015 seconds (29288 events/sec)

Count indv cuml rcnt     spin Lock                   Caller
-------------------------------------------------------------------------------
218549 75%  75% 1.00     3337 0x71ca3f50             entersq+0x314
26297   9%  83% 1.00     2533 0x71ca3f50             putnext+0x104
19875   7%  90% 1.00     4074 0x71ca3f50             strlock+0x534
14112   5%  95% 1.00     3577 0x71ca3f50             qcallbwrapper+0x274
 2696   1%  96% 1.00     3298 0x71ca51d4             putnext+0x50
 1821   1%  97% 1.00       59 0x71c9dc40             putnext+0xa0
 1693   1%  97% 1.00     2973 0x71ca3f50             qdrain_syncq+0x160
  683   0%  97% 1.00       66 0x71c9dc00             putnext+0xa0
  678   0%  98% 1.00       55 0x71c9dc80             putnext+0xa0
  586   0%  98% 1.00       25 0x71c9ddc0             putnext+0xa0
  513   0%  98% 1.00       42 0x71c9dd00             putnext+0xa0
  507   0%  98% 1.00       28 0x71c9dd80             putnext+0xa0
  407   0%  98% 1.00       42 0x71c9dd40             putnext+0xa0
  349   0%  98% 1.00     4085 0x8bfd7e1c             putnext+0x50
  264   0%  99% 1.00       44 0x71c9dcc0             putnext+0xa0
  187   0%  99% 1.00       12 0x908a3d90             putnext+0x454
  183   0%  99% 1.00     2975 0x71ca3f50             putnext+0x45c
  170   0%  99% 1.00     4571 0x8b77e504             strwsrv+0x10
  168   0%  99% 1.00     4501 0x8dea766c             strwsrv+0x10
  154   0%  99% 1.00     3773 0x924df554             strwsrv+0x10
```

# Examining Adaptive Locks Excessing Blocking

```
Adaptive mutex block: 2818 events in 10.015 seconds (281 events/sec)

Count indv cuml rcnt     nsec Lock                       Caller
-------------------------------------------------------------------------------
 2134  76%  76% 1.00  1423591 0x71ca3f50                 entersq+0x314
  272  10%  85% 1.00   893097 0x71ca3f50                 strlock+0x534
  152   5%  91% 1.00   753279 0x71ca3f50                 putnext+0x104
  134   5%  96% 1.00   654330 0x71ca3f50                 qcallbwrapper+0x274
   65   2%  98% 1.00   872630 0x71ca51d4                 putnext+0x50
    9   0%  98% 1.00   260444 0x71ca3f50                 qdrain_syncq+0x160
    7   0%  98% 1.00  1390807 0x8dea766c                 strwsrv+0x10
    6   0%  99% 1.00   906048 0x88876094                 strwsrv+0x10
    5   0%  99% 1.00  2266267 0x8bfd7e1c                 putnext+0x50
    4   0%  99% 1.00   468550 0x924df554                 strwsrv+0x10
    3   0%  99% 1.00   834125 0x8dea766c                 cv_wait_sig+0x198
    2   0%  99% 1.00   759290 0x71ca3f50                 drain_syncq+0x380
    2   0%  99% 1.00  1906397 0x8b77e504                 cv_wait_sig+0x198
    2   0%  99% 1.00   645358 0x71dd69e4                 qdrain_syncq+0xa0
```

# Examining Spin Locks Excessing Spinning

```
Spin lock spin: 52335 events in 10.015 seconds (5226 events/sec)

Count indv cuml rcnt     spin Lock                        Caller
-------------------------------------------------------------------------------
23531  45%  45% 1.00     4352 turnstile_table+0x79c       turnstile_lookup+0x48
 1864   4%  49% 1.00       71 cpu[19]+0x40                disp+0x90
 1420   3%  51% 1.00       74 cpu[18]+0x40                disp+0x90
 1228   2%  54% 1.00       23 cpu[10]+0x40                disp+0x90
 1159   2%  56% 1.00       60 cpu[16]+0x40                disp+0x90
 1138   2%  58% 1.00       22 cpu[24]+0x40                disp+0x90
 1108   2%  60% 1.00       57 cpu[17]+0x40                disp+0x90
 1082   2%  62% 1.00       24 cpu[11]+0x40                disp+0x90
 1039   2%  64% 1.00       25 cpu[29]+0x40                disp+0x90
 1009   2%  66% 1.00       17 cpu[23]+0x40                disp+0x90
 1007   2%  68% 1.00       21 cpu[31]+0x40                disp+0x90
  882   2%  70% 1.00       29 cpu[13]+0x40                disp+0x90
  846   2%  71% 1.00       25 cpu[28]+0x40                disp+0x90
  833   2%  73% 1.00       27 cpu[30]+0x40                disp+0x90
```

# Examining Reader/Writer Locks Excessing Blocking

```
R/W writer blocked by writer: 1 events in 10.015 seconds (0 events/sec)

Count indv cuml rcnt     nsec Lock                         Caller
-------------------------------------------------------------------------------
    1 100% 100% 1.00   169634 0x9d42d620                   segvn_pagelock+0x150
-------------------------------------------------------------------------------

R/W reader blocked by writer: 3 events in 10.015 seconds (0 events/sec)

Count indv cuml rcnt     nsec Lock                         Caller
-------------------------------------------------------------------------------
    3 100% 100% 1.00  1841415 0x75b7abec                   mir_wsrv+0x18
-------------------------------------------------------------------------------
```

# Examining Kernel Activity
## Kernel Profiling

```
# lockstat -kIi997 sleep 10

Profiling interrupt: 10596 events in 5.314 seconds (1994 events/sec)

Count indv cuml rcnt     nsec CPU+PIL                Caller
-------------------------------------------------------------------------------
 5122  48%  48% 1.00     1419 cpu[0]                 default_copyout
 1292  12%  61% 1.00     1177 cpu[1]                 splx
 1288  12%  73% 1.00     1118 cpu[1]                 idle
  911   9%  81% 1.00     1169 cpu[1]                 disp_getwork
  695   7%  88% 1.00     1170 cpu[1]                 i_ddi_splhigh
  440   4%  92% 1.00     1163 cpu[1]+11              splx
  414   4%  96% 1.00     1163 cpu[1]+11              i_ddi_splhigh
  254   2%  98% 1.00     1176 cpu[1]+11              disp_getwork
   27   0%  99% 1.00     1349 cpu[0]                 uiomove
   27   0%  99% 1.00     1624 cpu[0]                 bzero
   24   0%  99% 1.00     1205 cpu[0]                 mmrw
   21   0%  99% 1.00     1870 cpu[0]                 (usermode)
    9   0%  99% 1.00     1174 cpu[0]                 xcopyout
    8   0%  99% 1.00      650 cpu[0]                 ktl0
    6   0%  99% 1.00     1220 cpu[0]                 mutex_enter
    5   0%  99% 1.00     1236 cpu[0]                 default_xcopyout
    3   0% 100% 1.00     1383 cpu[0]                 write
    3   0% 100% 1.00     1330 cpu[0]                 getminor
    3   0% 100% 1.00      333 cpu[0]                 utl0
    2   0% 100% 1.00      961 cpu[0]                 mmread
    2   0% 100% 1.00     2000 cpu[0]+10              read_rtc
```

# trapstat(1)

- Solaris 9 only
- Statistics on CPU traps
  - Very processor architecture specific
- “-t” flag details TLB/TSB miss traps
  - Extremely useful for determining if large pages will help performance
    - Solaris 9 Multiple Page Size Support (MPSS)

# Hardware / Software Interface
## “where does CPU time go that you can't see?”

# Traps

- Traps
  - Entry into the kernel via one of several points of origin
    - system calls
    - interrupts
    - TLB fill
    - Register window spill/fill
  - Change from user mode (%usr) to kernel mode (%sys)
- Trap Accounting
  - %usr
    - user mode TLB fill, register spill/fill
  - %sys
    - system calls
    - Kernel TLB fill, register spill/fill
    - interrupts (Solaris 9 only)
  - %idle
    - interrupts (Solaris 8 and earlier)

# Trap CPU Accounting

```
# trapstat 3
vct  name                |    cpu0    cpu1
---------------------+--------------
 20  fp-disabled         |       0       0
 24  cleanwin            |    2568    2721
 41  level-1             |     100       0
 44  level-4             |       3       0
 46  level-6             |     315       0
 4a  level-10            |     100       0
 4d  level-13            |      28     118
 4e  level-14            |     100       0
 60  int-vec             |     377     118
 64  itlb-miss           |    8988    9619
 68  dtlb-miss           |   50789   39492
 6c  dtlb-prot           |       0       5
 84  spill-1-normal      |     885   12546
 88  spill-2-normal      |       0       2
 8c  spill-3-normal      |     162     191
 90  spill-4-normal      |       0       3
 98  spill-6-normal      |    5888    4041
 a4  spill-1-other       |     544     694
 a8  spill-2-other       |       0       2
 ac  spill-3-other       |    2938    2823
 b0  spill-4-other       |       0       6
 c4  fill-1-normal       |     931   12496
 c8  fill-2-normal       |       0       4
 cc  fill-3-normal       |    2712    3142
 d0  fill-4-normal       |       0       2
 d8  fill-6-normal       |    5660    4042
103  flush-wins          |      64     128
108  syscall-32          |    1526    1495
124  getts               |     463     331
127  gethrtime           |     493     518
140  syscall-64          |       0       3
---------------------+--------------
ttl                      |   85634   94542
```

# Trap CPU Accounting

```
# trapstat -t 3
cpu  | itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim | %tim
-----+-------------------------------+-------------------------------+-----
  0 k|        25  0.0         0  0.0 |     29558  0.5         6  0.0 | 0.6
  0 u|      9728  0.1         1  0.0 |     17943  0.3         3  0.0 | 0.5
-----+-------------------------------+-------------------------------+-----
  1 k|         0  0.0         0  0.0 |     19001  1.2         3  0.0 | 1.2
  1 u|      7872  0.2         0  0.0 |     16300  0.5         0  0.0 | 0.8
=====+===============================+===============================+=====
ttl  |     17625  0.2         1  0.0 |     82802  1.3        12  0.0 | 1.5
```

# System Calls

```
pae1> truss -c -p `pgrep bbrw`
^C
syscall               seconds   calls  errors
read                   11.729    1633
write                  16.492    1631
open                     .184    1631
close                    .102    1631
lseek                    .154    3264
                     --------  ------   ----
sys totals:            28.663    9790      0
usr time:                .078
elapsed:                8.140
pae1>
```

```
pae1> dtrace -n 'syscall:::entry { @[execname]=count(); }'
dtrace: description 'syscall:::entry ' matched 226 probes
^C

  inetd                                                         1
  svc.startd                                                    5
  sendmail                                                     11
  dtrace                                                      318
  bbrw                                                       6772
pae1>
```

# System Calls

```
pae1> dtrace -n 'syscall:::entry { @[probefunc]=count(); }'
dtrace: description 'syscall:::entry ' matched 226 probes
^C

  mmap                                                              1
  setcontext                                                        1
  schedctl                                                          1
  fstat                                                             1
  pollsys                                                           2
  sigaction                                                         2
  sysconfig                                                         3
  portfs                                                            5
  lwp_sigmask                                                       5
  brk                                                               6
  pset                                                              6
  gtime                                                             8
  lwp_park                                                         11
  p_online                                                         32
  ioctl                                                           255
  read                                                           1174
  open                                                           1176
  close                                                          1176
  write                                                          1177
  lseek                                                          2350
```

# System Calls

```
pae1> dtrace -n 'syscall:::entry { @[probefunc,execname]=count(); }'
dtrace: description 'syscall:::entry ' matched 226 probes
^C

  setcontext              dtrace                     1
  lwp_park                inetd                      1
  write                   dtrace                     1
  fstat                   dtrace                     1
  lwp_park                svc.configd                1
  lwp_sigmask             dtrace                     1
  pollsys                 sendmail                   1
  lwp_park                svc.startd                 1
  schedctl                dtrace                     1
  mmap                    dtrace                     1
  lwp_sigmask             sendmail                   2
  sigaction               dtrace                     2
  sysconfig               dtrace                     3
  pset                    sendmail                   3
  gtime                   sendmail                   4
  brk                     dtrace                     6
  portfs                  svc.startd                 7
  lwp_park                dtrace                     7
  p_online                dtrace                    32
  ioctl                   dtrace                   296
  write                   bbrw
  open                    bbrw                    1390
  close                   bbrw                    1390
  read                    bbrw                    1391
  lseek                   bbrw                    2781
```

# Interrupts

- An asynchronous event, not associated with the currently executing instruction
- Like traps, interrupts result in a vectored transfer of control to a specific routine, e.g. a device interrupt handler (part of the device driver).
- Also like traps, interrupts are hardware architecture specific
- Interrupts can be "hard" or "soft"
  – "Hard"ware interrupts generated by I/O devices
  – Soft interrupts are established via a call to the kernel add_softintr() function

# Interrupts

- Device interrupts
  - Round-robin binding of interrupting devices to processors
  - Intended to provide an even distribution of interrupt handling by processors
  - Observability of binding is currently an issue
  - `mpstat`(1) for interrupt rates
    - intr column. Interrupts per second
    - ithr column. Interrupts as threads
  - Each CPU is initialized with 10 interrupt threads
    - linked of CPU structure
    - An incoming interrupt gets a kernel thread structure from the list

# Interrupts

- When a CPU takes an interrupt, the currently running thread is “pinned” (not context switched out), some context is “borrowed”, and the interrupt thread runs
- If the interrupt thread completes
  - Simply unpin the pinned thread, and let it resume
- If the interrupt thread blocks
  - Must be upgraded to a “complete” thread, so it can block
    - This is the ithr column in `mpstat`
  - Allow the pinned thread to resume

# Interrupt partitioning

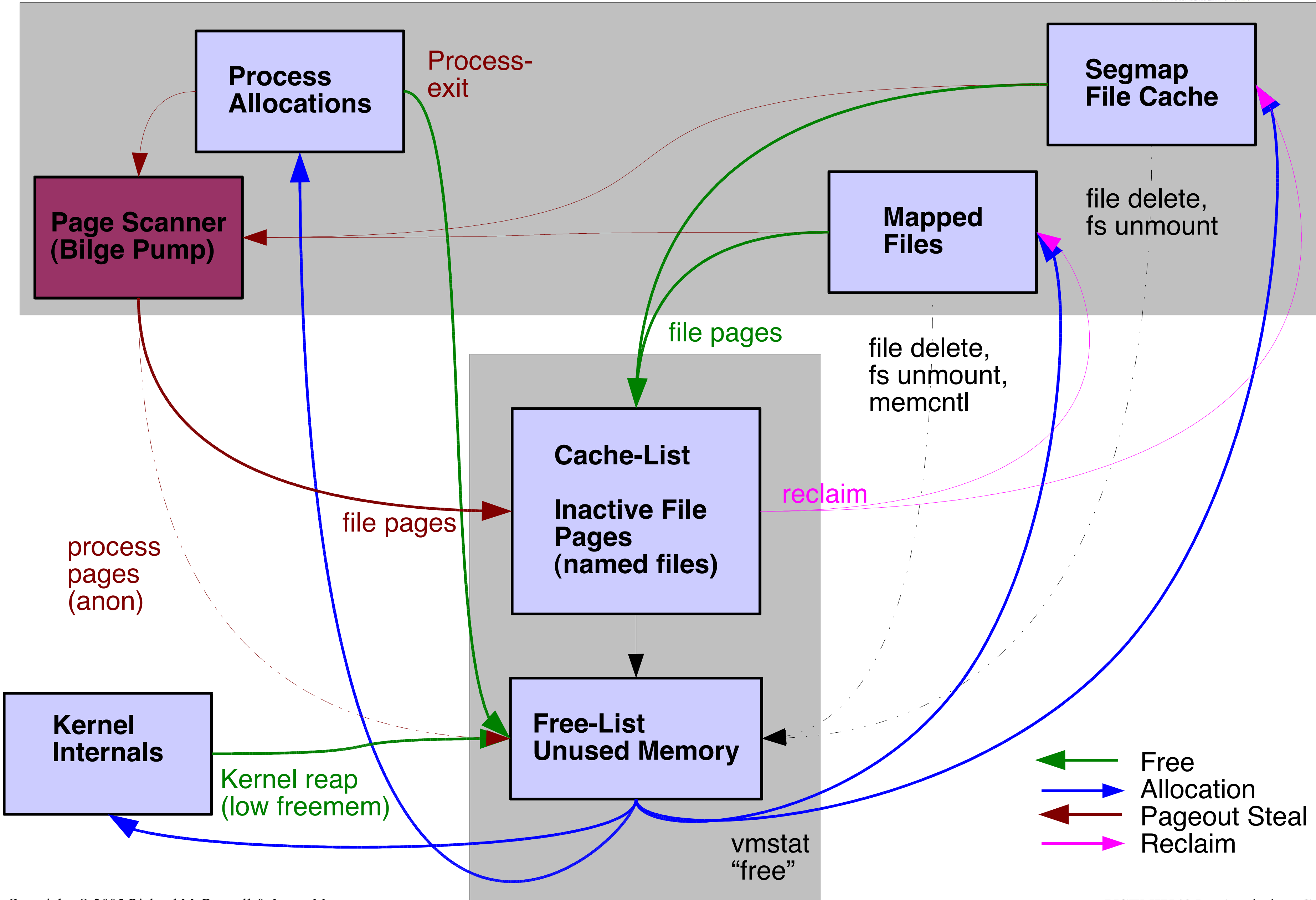
- Can be useful for some workloads
- Use processor sets and no-interrupt processor state capability
  - Construct processor sets for application processes
  - Place the processors in those sets in no-intr state
    - The kernel dynamically round-robins device to processor interrupt binding
- Leave enough processors without bound application processes to handle interrupts
- Note: we're working on making this easier

# Advanced Memory Topics

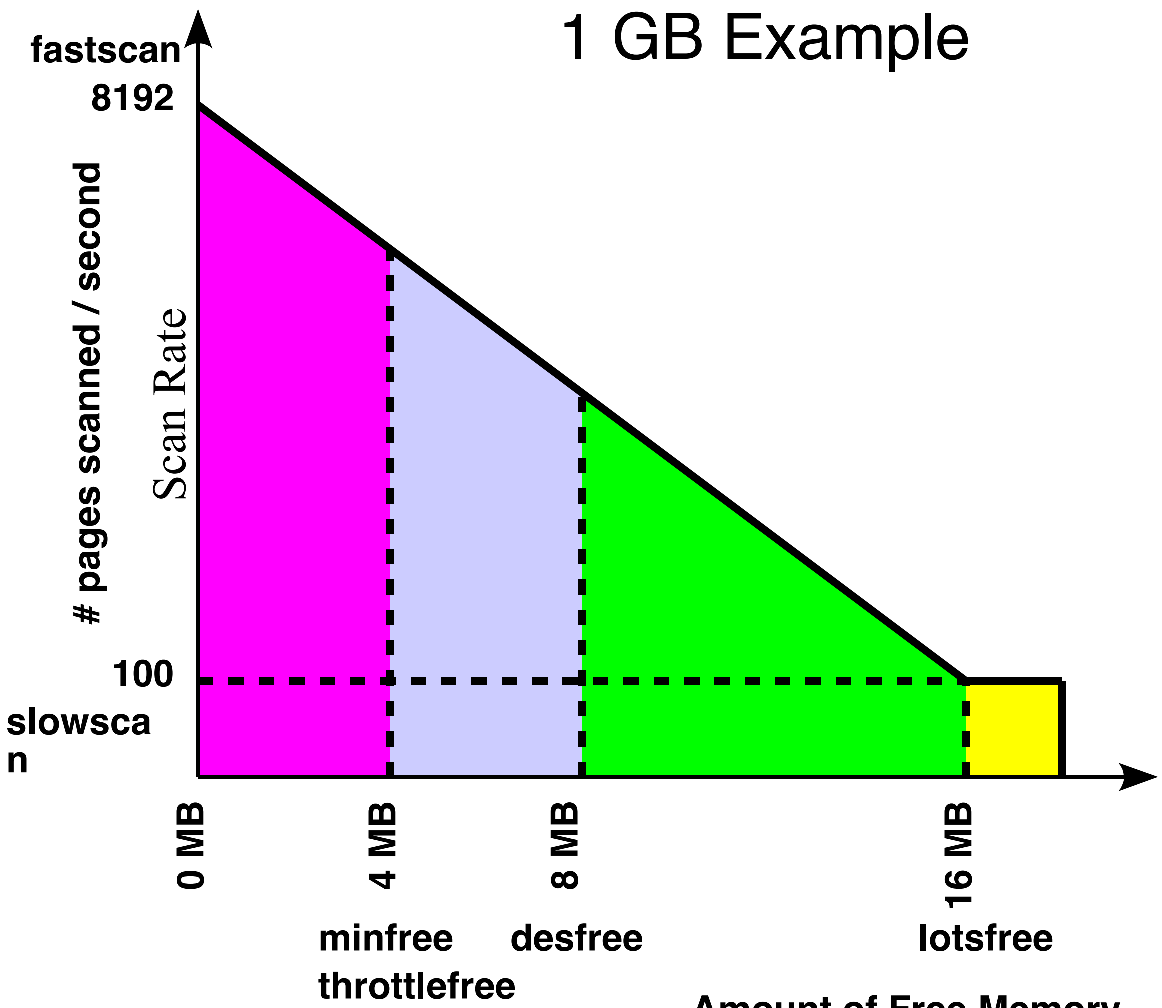
# A Quick Guide to Analyzing Memory

- Quick Memory Health Check
  - Check free memory and scanning with vmstat
  - Check memory usage with ::memstat in mdb
- Paging Activity
  - Use vmstat -p to check if there are anonymous page-ins
- Attribution
  - Use DTrace to see which processes/files are causing paging
- Time based analysis
  - Use DTrace to estimate the impact of paging on application performance
- Process Memory Usage
  - Use pmap to inspect process memory useage and sharing
- MMU/Page Size Performance
  - Use trapstat to observe time spent in TLB misses

# Memory Allocation Transitions

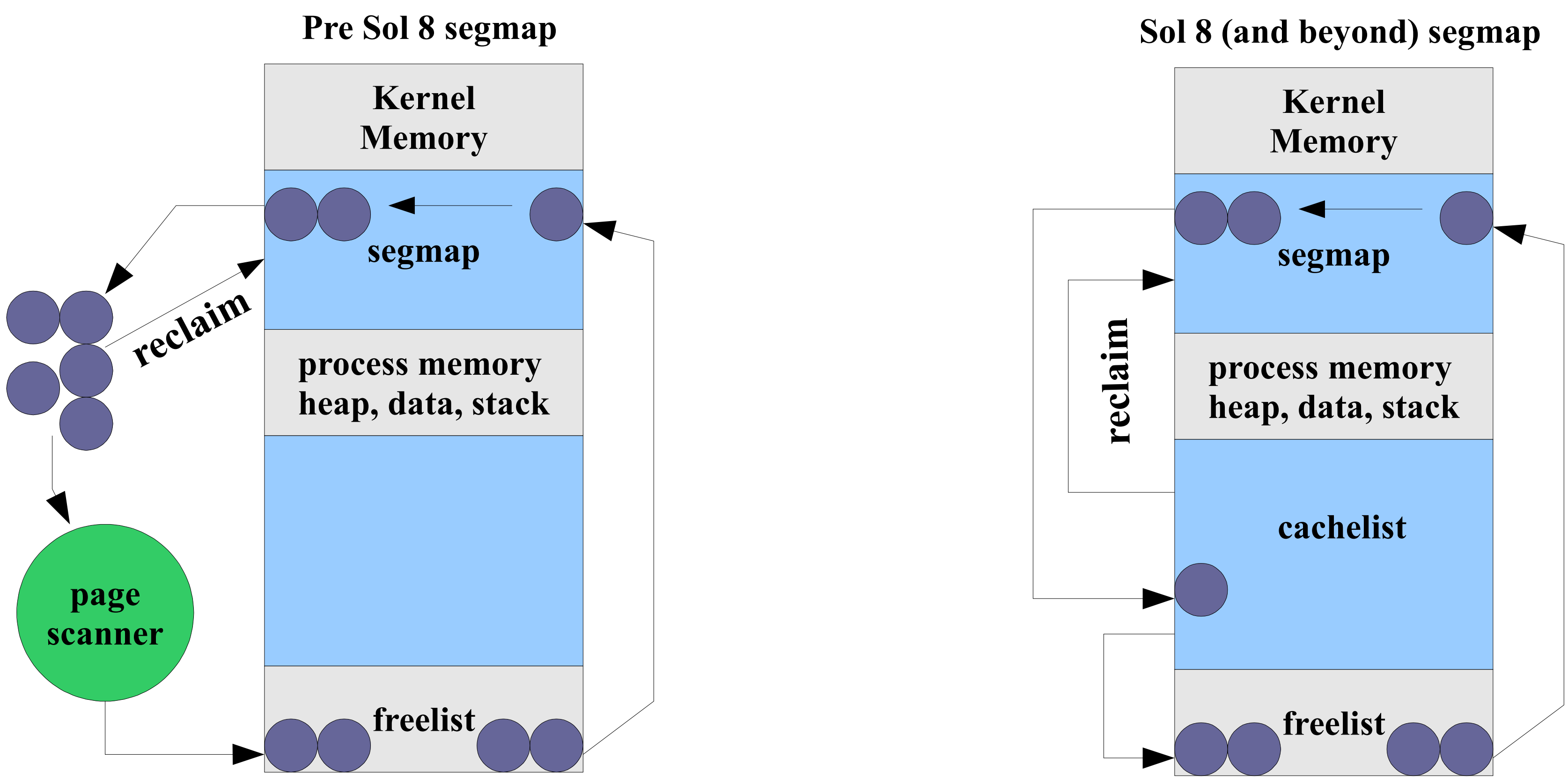


# Scan Rate



1 GB Example
fastscan
8192
# pages scanned / second
Scan Rate
100
slowscan
0 MB
4 MB
8 MB
16 MB
minfree
throttlefree
desfree
lotsfree
Amount of Free Memory

# The Solaris 8 File System Cache



# Vmstat -p

swap = free and unreserved swap in KBytes
free = free memory measured in pages

re = kilobytes reclaimed from cache/free list
mf = minor faults - the page was in memory but was not mapped
fr = kilobytes that have been destroyed or freed
de = kilobytes freed after writes
sr = kilobytes scanned / second

executable pages: kilobytes in - out - freed

anonymous pages: kilobytes in - out – freed

file system pages: kilobytes in - out - freed

```
# vmstat -p 5 5
     memory           page          executable      anonymous      filesystem
   swap    free  re  mf  fr  de  sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf
 ...
 46715224 891296 24  350 0   0   0    0    0    0    4    0    0   27    0    0
 46304792 897312 151 761 25  0   0   17    0    0    1    0    0  280   25   25
 45886168 899808 118 339 1   0   0    3    0    0    1    0    0  641    1    1
 46723376 899440 29  197 0   0   0    0    0    0   40    0    0   60    0    0
```

# Memory Summary

## Physical Memory:

```
# prtconf
System Configuration:  Sun Microsystems  sun4u
Memory size: 512 Megabytes
```

## Kernel Memory:

```
# sar -k 1 1

SunOS ian 5.8 Generic_108528-03 sun4u    08/28/01

13:04:58 sml_mem   alloc  fail  lg_mem   alloc  fail  ovsz_alloc  fail
13:04:59 10059904 7392775     0 133349376 92888024     0    10346496     0
```

## Free Memory:

```
# vmstat 3 3

 procs     memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr f0 s0 s1 s6   in   sy   cs us sy id
 0 0 0 478680 204528  0   2  0  0  0  0  0  0  0  1  0  209 1886  724 35  5 61
 0 0 0 415184 123400  0   2  0  0  0  0  0  0  0  0  0  238  825  451  2  1 98
 0 0 0 415200 123416  0   0  0  0  0  0  0  0  0  3  0  219  788  427  1  1 98
```

# Solaris 9+ Memory Summary

```
sol9# mdb -k
Loading modules: [ unix krtld genunix ufs_log ip usba s1394 nfs random
ptm ipc logindmux cpc ]
> ::memstat
Page Summary                Pages                MB  %Tot
------------     ----------------  ----------------  ----
Kernel                      10145                79    4%
Anon                        21311               166    9%
Exec and libs               15531               121    6%
Page cache                  69613               543   28%
Free (cachelist)           119633               934   48%
Free (freelist)             11242                87    5%

Total                      247475              1933
```

# Memory Kstats – via kstat(1m)

```
sol8# kstat -n system_pages
module: unix                                instance: 0
name:   system_pages                        class:    pages
        availrmem                           343567
        crtime                              0
        desfree                             4001
        desscan                             25
        econtig                             4278190080
        fastscan                            256068
        freemem                             248309
        kernelbase                          3556769792
        lotsfree                            8002
        minfree                             2000
        nalloc                              11957763
        nalloc_calls                        9981
        nfree                               11856636
        nfree_calls                         6689
        nscan                               0
        pagesfree                           248309
        pageslocked                         168569
        pagestotal                          512136
        physmem                             522272
        pp_kernel                           64102
        slowscan                            100
        snaptime                            6573953.83957897
```

# Memory Kstats – via kstat Perl API

```
%{$now} = %{$kstats->{0}{system_pages}};
print "$now->{pagesfree}\n";

sol8# wget http://www.solarisinternals.com/si/downloads/prtmem.pl
sol8# prtmem.pl 10
prtmem started on 04/01/2005 15:46:13 on devnull, sample interval 5
seconds
                     Total        Kernel         Delta          Free         Delta
15:46:18              2040           250             0           972           -12
15:46:23              2040           250             0           968            -3
15:46:28              2040           250             0           968             0
15:46:33              2040           250             0           970             1
```

# Checking Paging Activity

- Good Paging
  - Plenty of memory free
  - Only file system page-in/page-outs (vmstat: fpi, fpo > 0)

```
%sol8# vmstat -p 3
     memory           page          executable      anonymous      filesystem
   swap  free  re  mf  fr  de  sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf
 1512488 837792 160 20 12   0   0    0    0    0    0    0    0   12   12   12
 1715812 985116 7  82   0   0   0    0    0    0    0    0    0   45    0    0
 1715784 983984 0   2   0   0   0    0    0    0    0    0    0   53    0    0
 1715780 987644 0   0   0   0   0    0    0    0    0    0    0   33    0    0
```

# Checking Paging Activity

- Bad Paging
  - Non zero Scan rate (vmstat: sr >0)
  - Low free memory (vmstat: free < 1/16th physical)
  - Anonymous page-in/page-outs (vmstat: api, apo > 0)

```
sol8# vmstat -p 3
     memory           page          executable      anonymous      filesystem
   swap  free  re  mf   fr  de  sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf
 2276000 1589424 2128 19969 1 0 0    0    0    0    0    0    0    0    1    1
 1087652 388768 12 129675 13879 0 85590 0 0   12    0 3238 3238   10 9391 10630
 608036 51464  20 8853 37303 0 65871 38   0  781   12 19934 19930 95 16548 16591
  94448  8000  17 23674 30169 0 238522 16 0  810   23 28739 28804 56  547  556
```

# Using prstat to estimate paging slow-downs

- Microstates show breakdown of elapsed time
  - prstat -m
  - USR through LAT columns summed show 100% of wallclock execution time for target thread/process
  - DFL shows time spent waiting in major faults in anon:

```
sol8$ prstat -mL
   PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID
 15625 rmc      0.1 0.7 0.0 0.0  95 0.0 0.9 3.2  1K 726  88   0 filebench/2
 15652 rmc      0.1 0.7 0.0 0.0  94 0.0 1.8 3.6  1K  1K  10   0 filebench/2
 15635 rmc      0.1 0.7 0.0 0.0  96 0.0 0.5 3.2  1K  1K   8   0 filebench/2
 15626 rmc      0.1 0.6 0.0 0.0  95 0.0 1.4 2.6  1K 813  10   0 filebench/2
 15712 rmc      0.1 0.5 0.0 0.0  47 0.0  49 3.8  1K 831 104   0 filebench/2
 15628 rmc      0.1 0.5 0.0 0.0  96 0.0 0.0 3.1  1K 735   4   0 filebench/2
 15725 rmc      0.0 0.4 0.0 0.0  92 0.0 1.7 5.7 996 736   8   0 filebench/2
 15719 rmc      0.0 0.4 0.0 0.0  40  40  17 2.9  1K 708 107   0 filebench/2
 15614 rmc      0.0 0.3 0.0 0.0  92 0.0 4.7 2.4 874 576  40   0 filebench/2
```

# Using DTrace for memory Analysis

- The “vminfo” provider has probes at the all the places memory statistics are gathered.
- Everything visible via vmstat -p and kstat are defined as probes
  - arg0: the value by which the statistic is to be incremented. For most probes, this argument is always 1, but for some it may take other values; these probes are noted in Table 5-4.
  - arg1: a pointer to the current value of the statistic to be incremented. This value is a 64Ðbit quantity that is incremented by the value in arg0. Dereferencing this pointer allows consumers to determine the current count of the statistic corresponding to the probe.

# Using DTrace for Memory Analysis

- For example, if you should see the following paging activity with vmstat, indicating page-in from the swap device, you could drill down to investigate.

```
sol8# vmstat -p 3
     memory           page          executable      anonymous      filesystem
   swap  free  re  mf  fr  de  sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf
 1512488 837792 160 20 12   0   0    0    0    0 8102    0    0   12   12   12
 1715812 985116 7   82  0   0   0    0    0    0 7501    0    0   45    0    0
 1715784 983984 0    2  0   0   0    0    0    0 1231    0    0   53    0    0
 1715780 987644 0    0  0   0   0    0    0    0 2451    0    0   33    0    0
```

```
sol10$ dtrace -n anonpgin '{@[execname] = count()}'
dtrace: description anonpgin matched 1 probe
  svc.startd                                                      1
  sshd                                                            2
  ssh                                                             3
  dtrace                                                          6
  vmstat                                                         28
  filebench                                                     913
```

# Using DTrace to estimate paging slow-downs

- DTrace has probes for paging
- By measuring elapsed time at the paging probes, we can see who's waiting for paging:

```
sol10$ ./whospaging.d

Who's waiting for pagein (milliseconds):
  wnck-applet                                                   21
  gnome-terminal                                                75

Who's on cpu (milliseconds):
  wnck-applet                                                   13
  gnome-terminal                                                14
  metacity                                                      23
  Xorg                                                          90
  sched                                                       3794
```

# Using DTrace to estimate paging slow-downs

- DTrace has probes for paging
- By measuring elapsed time at the paging probes, we can see who's waiting for paging:

```
sol10$ ./pagingtime.d 22599

  <on cpu>                                                          913
  <paging wait>                                                     230704
```

# To a Terrabyte and Beyond: Utilizing and Tuning Large Memory

# Who said this?

"640k ought to be enough for everyone"

# Who said this?

"640k ought to be enough for everyone"

– Bill Gates, 1981

# Large Memory

- Large Memory in Perspective
- 64-bit Solaris
- 64-bit Hardware
- Solaris enhancements for Large Memory
- Large Memory Databases
- Configuring Solaris for Large Memory
- Using larger page sizes

# Application Dataset Growth

- Commercial applications
  - RDMBS caching for SQL & Disk blocks using up to 500GB
  - Supply Chain models now reaching 200GB
- Virtual Machines
  - 1 Address space for all objects,  JVM today is 100GB+
- Scientific/Simulation/Modelling
  - Oil/Gas, Finite element, Bioinformatics models 500GB+
  - Medium size mechanical models larger than 4GB
- Desktops: Low end 512MB today, 4GB in 2006?

# Large memory in perspective

- 640k:
  - 19 bits of address space is enough?
  - 3 years later we ran out of bits...
- 32-bit systems will last for ever?
  - 4 Gigabytes
  - 10 years after introduction we ran out of bits again

# 64-bits – enough for everyone?

- 64-bits – finally we won't run out...
- 16 Exabytes!
- That's 16,384 Peta-bytes
- However: 1PB is feasible today
- That's only 14 bits x 1Petabyte
- If we grow by 1 bit per year
  - We'll run out of bits again in 2020...

# Solaris 7,8,9...

## Full 64-bit support

# Sun's 64-Bit History

- SPARC V9 – first 64-Bit SPARC
  - Circa 1995
  - 64-bit registers, program counters etc
  - 32 & 64Bit load/store instructions allow 32-Bit applications to execute alongside 64-Bit

# Sun's 64-Bit History

- Solaris Releases
  - 2.0 - 64bit drivers, 40 bit SCSI & Filesystems
  - 2.6 - 64bit files & API
  - 7 - full 64-bit
    - Longs and Pointers 64-bits
    - Optimal source compatibility with 32-bit apps

# 64-Bit Solaris

- LP64 Data Model
- 32-bit or 64-bit kernel, with 32-bit & 64-bit application support
- Comprehensive 32-bit application compatibility

# Why 64-bit for large memory?

- Extends the existing programming model to large memory
- Existing POSIX APIs extend to large data types (e.g. file offets. file handle limits eliminated)
- Simple transition of existing source to 64-bits

# Developer Perspective

- Virtually unlimited address space
  - Data objects, files, large hardware devices can be mapped into virtual address space
  - 64-bit data types, parameter passing
  - Caching can be implemented in application, yielding much higher performance
- Small Overheads

# Exploiting 64-bits

- Commercial: Java Virtual Machine, SAP, Microfocus Cobol, ANTS, XMS, Multigen
- RDBMS: Oracle, DB2, Sybase, Informix, Times Ten
- Mechanical/Design: PTC, Unigraphics, Mentor Graphics, Cadence, Synopsis etc...
- Supply Chain: I2, SAP, Manguistics
- HPC: PTC, ANSYS, ABAQUS, Nastran, LS-Dyna, Fluent etc...

# Large Memory Hardware

- DIMMS
  - 2GB DIMMS: 16GB/CPU
  - 1GB DIMMS: 8GB/CPU
  - 512MB DIMMS: 4GB/CPU
- E6800: 192GB Max
  - 8GB/CPU
- F25k: 1152GB Max
  - 16GB/CPU

# Large Memory Solaris

- Solaris 7: 64-bits
- Solaris 8: 80GB
- Solaris 8 U6: 320GB
- Solaris 8 U7: 576GB
- Solaris 9: 1.1TB

# Large Memory Solaris (ctd)

- Solaris 2.6
  - ISM, 4MB Page Support
- Solaris 8
  - New VM, large memory fs cache
- Solaris 8, 2/02
  - Large working sets MMU perf
  - Raise 8GB limit to 128GB
  - Dump Performance improved
  - Boot performance improved
- Solaris 9
  - Generic multiple page size facility and tools

# Configuring Solaris

- fsflush use too much CPU on Solaris 8
  - Set “autoup” in /etc/system
  - Symptom is one CPU using 100%sys
- Corrective Action
  - Default is 30s, recommend setting larger
  - e.g. 10x nGB of memory

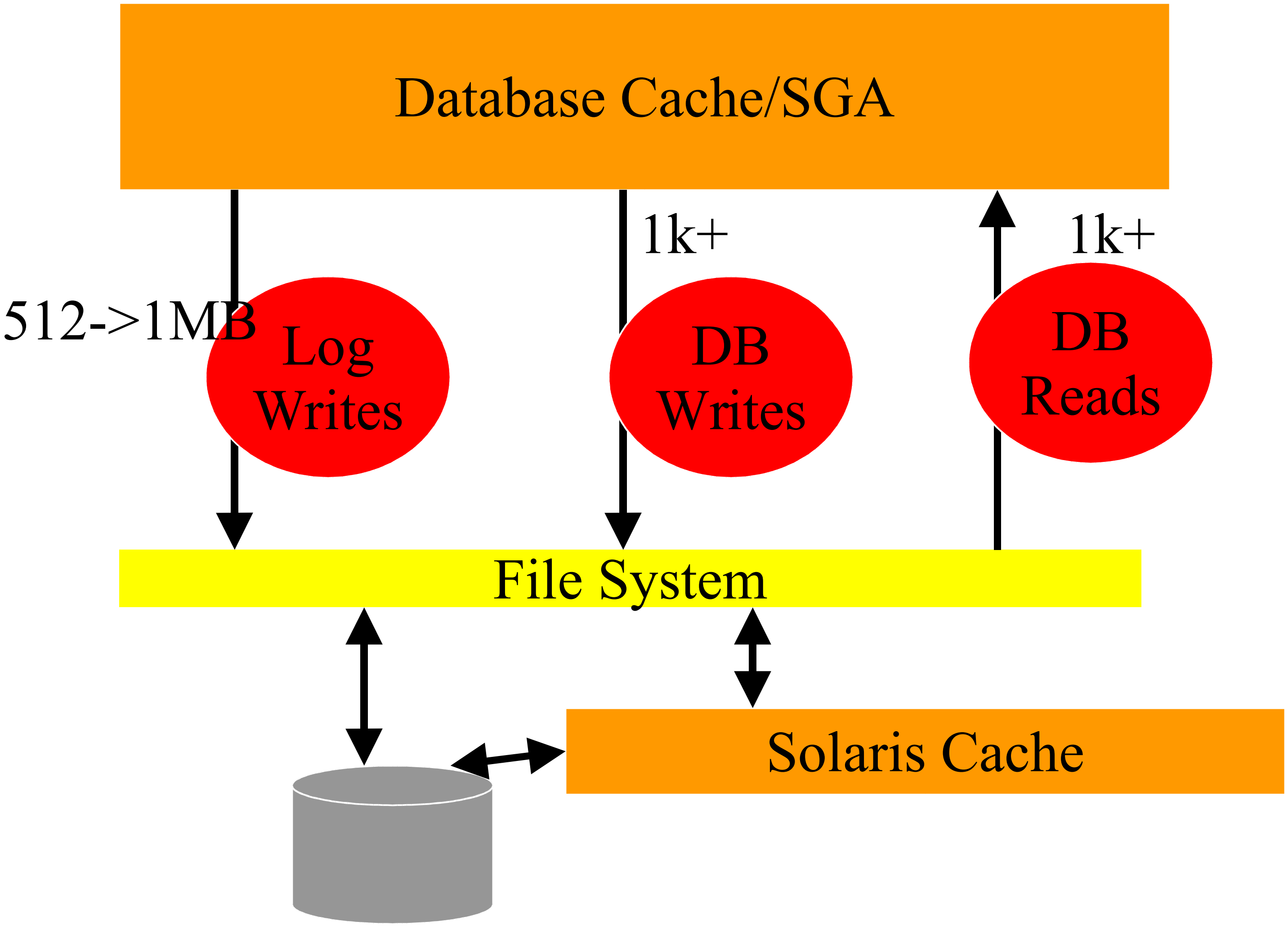
# Large Dump Performance

- Configure "kernel only"
  - Dumpadm
- Estimate dump as 20% of memory size
- Configure separate dump device
  - Reliable dumps
  - Asynchronous saves during boot
- Configure a fast dump device
  - T3 Stripe as a dump device

# Databases

- Expolit memory to reduce/eliminate I/O!
- Eliminating I/O is the easiest way to tune it...
- Increase cache hit rates:
  - 95% means 1 out 20 accesses result in I/O
  - 99% means 1 out of 100 – 500% reduction in I/O!
- We can often fit entire RDBMS in memory
- Write mostly I/O pattern results

# Oracle File I/O



# 64-Bit Oracle

- Required to cache more than 3.75GB
- Available since DBMS 8.1.7
- Sun has tested up to 540GB SGA
- Recommended by Oracle and Sun
- Cache for everything except PQ
- Pay attention to cold-start times

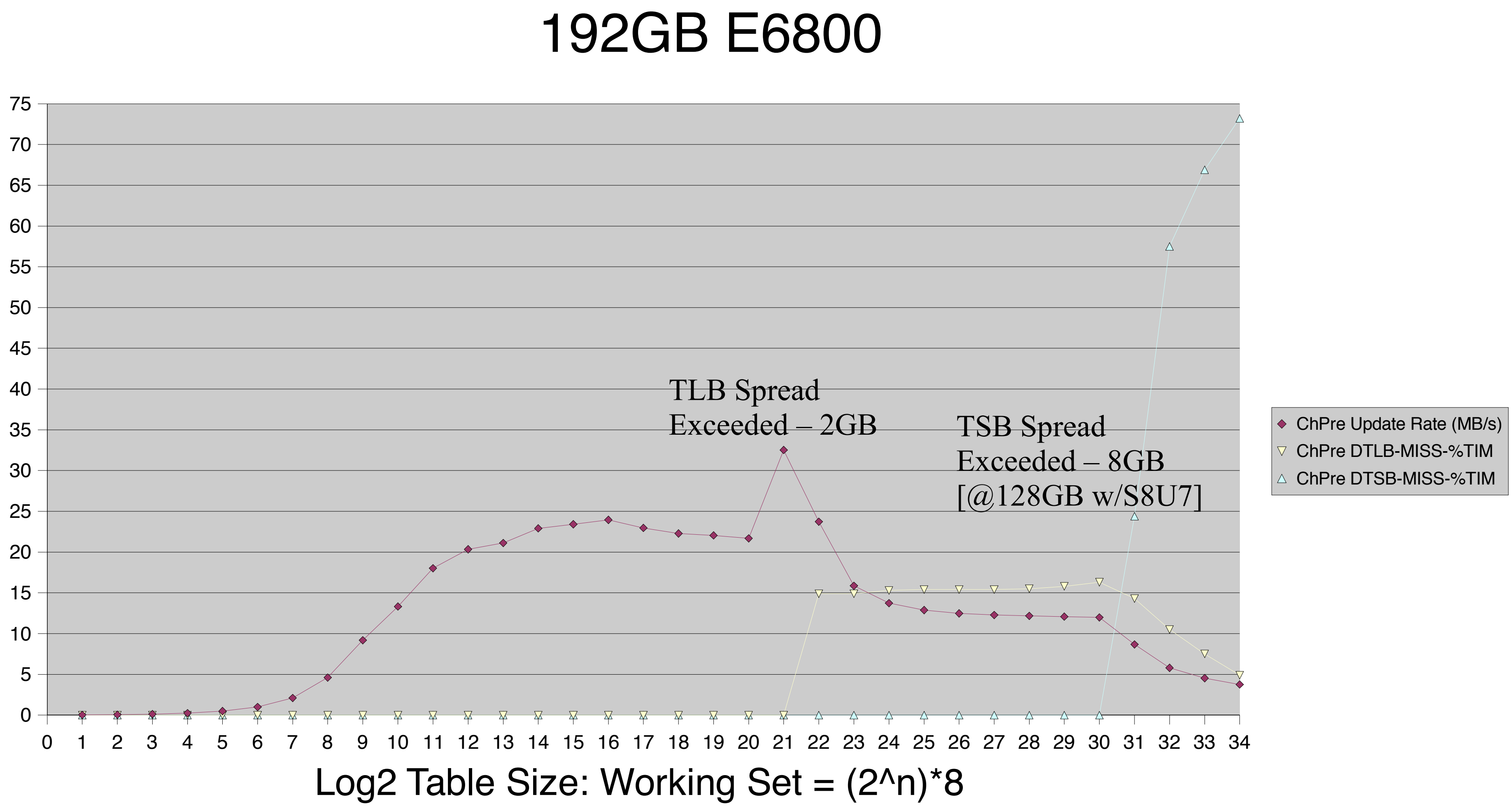
# Solaris 8/9 - Large Pages

- Solaris 8
  - Large (4MB) pages with ISM/DISM for shared memory
- Solaris 9
  - "Multiple Page Size Support"
  - Optional large pages for heap/stack
  - Programatically via madvise()
  - Shared library for existing binaries (LD_PRELOAD)
  - Tool to observe potential gains -
    - # trapstat -T

# Do I need Large Pages?

- Is the application memory intensive?
- How much time is being wasted in MMU traps?
  - MMU traps are not visible with %usr/%sys
  - MMU traps are counted in the current context
  - e.g. User-bound process reports as %usr

# TLB Performance Knees



# Trapstat Introduction

```
sol9# trapstat -t 1 111
cpu m| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
-----+-------------------------------+-------------------------------+----
  0 u|         1  0.0         0  0.0 |   2171237 45.7         0  0.0 |45.7
  0 k|         2  0.0         0  0.0 |      3751  0.1         7  0.0 | 0.1
=====+===============================+===============================+====
 ttl |         3  0.0         0  0.0 |   2192238 46.2         7  0.0 |46.2
```

- This application might run almost 2x faster!

# Observing MMU traps

```
sol9# trapstat -T 1 111
cpu m size| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
----------+-------------------------------+-------------------------------+----
  0 u   8k|        30  0.0         0  0.0 |   2170236 46.1         0  0.0 |46.1
  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|         1  0.0         0  0.0 |      4174  0.1        10  0.0 | 0.1
  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 |         0  0.0         0  0.0 | 0.0
==========+===============================+===============================+====
      ttl |        31  0.0         0  0.0 |   2174410 46.2        10  0.0 |46.2
```

# Observing MMU traps

```
sol9# trapstat -t 1 111
cpu m| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
-----+-------------------------------+-------------------------------+----
  0 u|         1  0.0         0  0.0 |   2171237 45.7         0  0.0 |45.7
  0 k|         2  0.0         0  0.0 |      3751  0.1         7  0.0 | 0.1
=====+===============================+===============================+====
 ttl |         3  0.0         0  0.0 |   2192238 46.2         7  0.0 |46.2
```

# Available Page Sizes

```
sol9# pagesize -a
8192
65536
524288
4194304
```

# Setting Page Sizes

- Solution: Use the wrapper program
  - Sets page size preference
  - Doesn't persist across exec()

```
sol9# ppgsz -o heap=4M ./testprog
```

# Checking Allocated Page Sizes

```
Sol9# pmap -sx `pgrep testprog`
2953:   ./testprog
 Address  Kbytes     RSS    Anon  Locked Pgsz Mode   Mapped File
00010000       8       8       -       -   8K r-x--  dev:277,83 ino:114875
00020000       8       8       8       -   8K rwx--  dev:277,83 ino:114875
00022000    3960    3960    3960       -   8K rwx--    [ heap ]
00400000  131072  131072  131072       -   4M rwx--    [ heap ]
FF280000     120     120       -       -   8K r-x--  libc.so.1
FF340000       8       8       8       -   8K rwx--  libc.so.1
FF390000       8       8       -       -   8K r-x--  libc_psr.so.1
FF3A0000       8       8       -       -   8K r-x--  libdl.so.1
FF3B0000       8       8       8       -   8K rwx--    [ anon ]
FF3C0000     152     152       -       -   8K r-x--  ld.so.1
FF3F6000       8       8       8       -   8K rwx--  ld.so.1
FFBFA000      24      24      24       -   8K rwx--    [ stack ]
-------- ------- ------- ------- -------
total Kb  135968  135944  135112       -
```

# TLB traps eliminated

```
sol9# trapstat -T 1 111
cpu m size| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
----------+-------------------------------+-------------------------------+----
  0 u   8k|        30  0.0         0  0.0 |        36  0.1         0  0.0 | 0.1
  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|         1  0.0         0  0.0 |      4174  0.1        10  0.0 | 0.1
  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 |         0  0.0         0  0.0 | 0.0
==========+===============================+===============================+====
      ttl |        31  0.0         0  0.0 |      4200  0.2        10  0.0 | 0.2
```

# Solution: Use the preload lib.

```
sol9# LD_PRELOAD=$LD_PRELOAD:mpss.so.1
sol9# export LD_PRELOAD=$LD_PRELOAD:mpss.so.1
sol9# export MPSSHEAP=4M
sol9# ./testprog
```

```
MPSSHEAP=size
MPSSSTACK=size
MPSSHEAP and MPSSSTACK specify the preferred page
sizes for the heap and stack, respectively. The speci-
fied page size(s) are applied to all created
processes.
MPSSCFGFILE=config-file
config-file is a text file which contains one or more
mpss configuration entries of the form:
exec-name:heap-size:stack-size
```

# What about Solaris 8?

```
sol8# cpustat -c pic0=Cycle_cnt,pic1=DTLB_miss 1
time cpu event pic0 pic1
1.006 0 tick 663839993 3540016
2.006 0 tick 651943834 3514443
3.006 0 tick 630482518 3398061
4.006 0 tick 634483028 3418046
5.006 0 tick 651910256 3511458
6.006 0 tick 651432039 3510201
7.006 0 tick 651512695 3512047
8.006 0 tick 613888365 3309406
9.006 0 tick 650806115 3510292
```

# Tips for UltraSPARC revs

- UltraSPARC II
  - Up to four page sizes can be used
  - 8k,64k,512k,4M
- UltraSPARC III 750Mhz
  - Optimized for 8k
  - Only one large page size
  - 7 TLB entries for large pages
  - Pick from 64k, 512k, 4M
- UltraSPARC III+ (900Mhz+)
  - Only one large page size
  - 512 TLB entries for large pages

# Solaris 8/9 – Large Pages

- Solaris 8
  - Large (4MB) pages with ISM/DISM for shared memory
- Solaris 9
  - "Multiple Page Size Support"
  - Optional large pages for heap/stack
  - Programatically via madvise()
  - Shared library for existing binaries (LD_PRELOAD)
  - Tool to observe potential gains -
    - # trapstat -T

# Trapstat Introduction

```
sol9# trapstat -T 1
cpu m size| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
----------+-------------------------------+-------------------------------+----
  0 u   8k|        30  0.0         0  0.0 |   2170236 46.1         0  0.0 |46.1
  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|         1  0.0         0  0.0 |      4174  0.1        10  0.0 | 0.1
  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 |         0  0.0         0  0.0 | 0.0
==========+===============================+===============================+====
      Ttl |        31  0.0         0  0.0 |   2174410 46.2        10  0.0 |46.2
```

- All of the misses are occuring on 8k pages

# OK, Lets do something about it

- By default, only applications using special shared memory use larger pages
  - Intimate Shared Memory for databases
  - shmat() with SHM_SHARE_MMU
- Solaris 9 introduces a generic framework
  - Multiple Page Sizes for Solaris (MPSS)

# Available Page Sizes

```
sol9# pagesize -a
8192
65536
524288
4194304
```

# Setting Page Sizes

- Solution: Use the wrapper program
  - Sets page size preference
  - Doesn't persist across exec()

```
sol9# ppgsz -o heap=4M ./testprog
```

# Checking Allocated Page Sizes

```
Sol9# pmap -sx `pgrep testprog`
2953:   ./testprog
 Address  Kbytes     RSS    Anon  Locked Pgsz Mode   Mapped File
00010000       8       8       -       -   8K r-x--  dev:277,83 ino:114875
00020000       8       8       8       -   8K rwx--  dev:277,83 ino:114875
00022000    3960    3960    3960       -   8K rwx--    [ heap ]
00400000  131072  131072  131072       -   4M rwx--    [ heap ]
FF280000     120     120       -       -   8K r-x--  libc.so.1
FF340000       8       8       8       -   8K rwx--  libc.so.1
FF390000       8       8       -       -   8K r-x--  libc_psr.so.1
FF3A0000       8       8       -       -   8K r-x--  libdl.so.1
FF3B0000       8       8       8       -   8K rwx--    [ anon ]
FF3C0000     152     152       -       -   8K r-x--  ld.so.1
FF3F6000       8       8       8       -   8K rwx--  ld.so.1
FFBFA000      24      24      24       -   8K rwx--    [ stack ]
-------- ------- ------- ------- -------
total Kb  135968  135944  135112       -
```

# TLB traps eliminated

```
sol9# trapstat -T 1 111
cpu m size| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
----------+-------------------------------+-------------------------------+----
  0 u   8k|        30  0.0         0  0.0 |        36  0.1         0  0.0 | 0.1
  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|         1  0.0         0  0.0 |      4174  0.1        10  0.0 | 0.1
  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 |         0  0.0         0  0.0 | 0.0
==========+===============================+===============================+====
      ttl |        31  0.0         0  0.0 |      4200  0.2        10  0.0 | 0.2
```

# Solution: Use the preload lib.

```
sol9# LD_PRELOAD=$LD_PRELOAD:mpss.so.1
sol9# export LD_PRELOAD=$LD_PRELOAD:mpss.so.1
sol9# export MPSSHEAP=4M
sol9# ./testprog
```

```
MPSSHEAP=size
MPSSSTACK=size
MPSSHEAP and MPSSSTACK specify the preferred page
sizes for the heap and stack, respectively. The speci-
fied page size(s) are applied to all created
processes.
MPSSCFGFILE=config-file
config-file is a text file which contains one or more
mpss configuration entries of the form:
exec-name:heap-size:stack-size
```

# Processes, Threads, Scheduling Classes & The Dispatcher
# Day 2 – A Deeper Dive

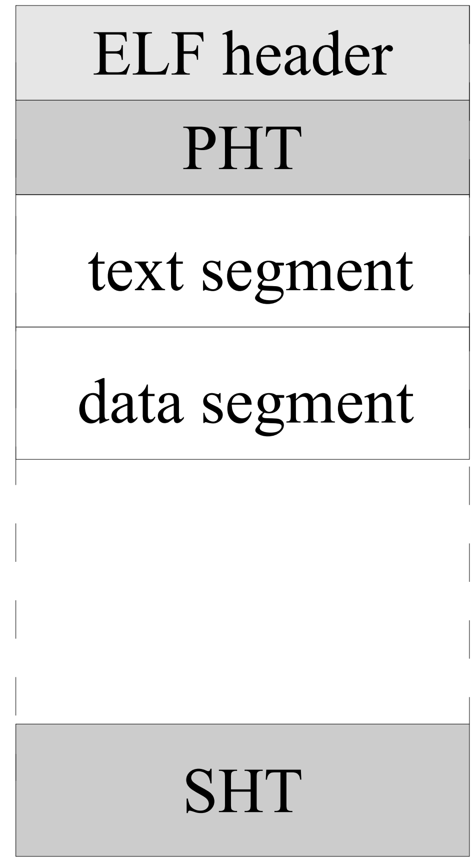
# Where Do Processes Come From?

# Executable Files

- Processes orginate as executable programs that are exec'd
- Executable & Linking Format (ELF)
  - Standard executable binary file Application Binary Interface (ABI) format
  - Two standards components
    - Platform independent
    - Platform dependent (e.g. SPARC)
  - Defines both the on-disk image format, and the in-memory image
  - ELF files components defined by
    - ELF header
    - Program Header Table (PHT)
    - Section Header Table (SHT)

# Executable & Linking Format (ELF)

- ELF header
  - Roadmap to the file
- PHT
  - Array of Elf_Phdr structures, each defines a segment for the loader (exec)
- SHT
  - Array of Elf_Shdr structures, each defines a section for the linker (ld)



# ELF Files

- ELF on-disk object created by the link-editor at the tail-end of the compilation process (although we still call it an a.out by default...)
- ELF objects can be statically linked or dynamically linked
  - Compiler "-B static" flag, default is dynamic
  - Statically linked objects have all references resolved and bound in the binary (libc.a)
  - Dynamically linked objects rely on the run-time linker, ld.so.1, to resolve references to shared objects at run time (libc.so.1)
  - Static linking is discouraged, and not possible for 64-bit binaries

# Examing ELF Files

- Use `elfdump`(1) to decompose ELF files

```
borntorun> elfdump -e /bin/ls

ELF Header
  ei_magic:   { 0x7f, E, L, F }
  ei_class:   ELFCLASS32          ei_data:      ELFDATA2MSB
  e_machine:  EM_SPARC            e_version:    EV_CURRENT
  e_type:     ET_EXEC
  e_flags:                     0
  e_entry:               0x10f00  e_ehsize:     52  e_shstrndx:   26
  e_shoff:                0x4654  e_shentsize:  40  e_shnum:      27
  e_phoff:                  0x34  e_phentsize:  32  e_phnum:       6
borntorun>
```

# Examining ELF Files

- `elfdump` -c dumps section headers

```
borntorun> elfdump -c /bin/ls
Section Header[11]:  sh_name: .text
    sh_addr:      0x10f00         sh_flags:    [ SHF_ALLOC  SHF_EXECINSTR ]
    sh_size:      0x2ec4          sh_type:     [ SHT_PROGBITS ]
    sh_offset:    0xf00           sh_entsize: 0
    sh_link:      0               sh_info:     0
    sh_addralign: 0x8

Section Header[17]:  sh_name: .got
    sh_addr:      0x24000         sh_flags:    [ SHF_WRITE  SHF_ALLOC ]
    sh_size:      0x4             sh_type:     [ SHT_PROGBITS ]
    sh_offset:    0x4000          sh_entsize: 0x4
    sh_link:      0               sh_info:     0
    sh_addralign: 0x2000

Section Header[18]:  sh_name: .plt
    sh_addr:      0x24004         sh_flags:    [ SHF_WRITE  SHF_ALLOC  SHF_EXECINSTR ]
    sh_size:      0x28c           sh_type:     [ SHT_PROGBITS ]
    sh_offset:    0x4004          sh_entsize: 0xc
    sh_link:      0               sh_info:     0
    sh_addralign: 0x4

Section Header[22]:  sh_name: .data
    sh_addr:      0x24380         sh_flags:    [ SHF_WRITE  SHF_ALLOC ]
    sh_size:      0x154           sh_type:     [ SHT_PROGBITS ]
    sh_offset:    0x4380          sh_entsize: 0
    sh_link:      0               sh_info:     0
    sh_addralign: 0x8

Section Header[24]:  sh_name: .bss
    sh_addr:      0x24540         sh_flags:    [ SHF_WRITE  SHF_ALLOC ]
    sh_size:      0xbc4           sh_type:     [ SHT_NOBITS ]
    sh_offset:    0x4540          sh_entsize: 0
    sh_link:      0               sh_info:     0
    sh_addralign: 0x8
```

# Examing ELF Linker Dependencies

- Use `ldd`(1) to invoke the runtime linker (ld.so) on a binary file, and `pldd`(1) on a running process

```
borntorun> ldd netstat
        libdhcpagent.so.1 =>   /usr/lib/libdhcpagent.so.1
        libcmd.so.1 =>     /usr/lib/libcmd.so.1
        libsocket.so.1 => /usr/lib/libsocket.so.1
        libnsl.so.1 =>     /usr/lib/libnsl.so.1
        libkstat.so.1 =>  /usr/lib/libkstat.so.1
        libc.so.1 =>  /usr/lib/libc.so.1
        libdl.so.1 =>      /usr/lib/libdl.so.1
        libmp.so.2 =>      /usr/lib/libmp.so.2
        /usr/platform/SUNW,Ultra-60/lib/libc_psr.so.1
```

```
borntorun> pldd $$
495: ksh
/usr/lib/libsocket.so.1
/usr/lib/libnsl.so.1
/usr/lib/libc.so.1
/usr/lib/libdl.so.1
/usr/lib/libmp.so.2
/usr/platform/sun4u/lib/libc_psr.so.1
/usr/lib/locale/en_US.ISO8859-1/en_US.ISO8859-1.so.2
borntorun>
```

# Runtime Linker Debug

```
solaris> LD_DEBUG=help date
00000:
...
00000: args       display input argument processing (ld only)
00000: audit      display runtime link-audit processing (ld.so.1 only)
00000: basic      provide basic trace information/warnings
00000: bindings   display symbol binding; detail flag shows absolute:relative
00000:              addresses (ld.so.1 only)
00000: cap        display hardware/software capability processing
00000: detail     provide more information in conjunction with other options
00000: demangle   display C++ symbol names in their demangled form
00000: entry      display entrance criteria descriptors (ld only)
00000: files      display input file processing (files and libraries)
00000: got        display GOT symbol information (ld only)
00000: help       display this help message
00000: libs       display library search paths; detail flag shows actual
00000:              library lookup (-l) processing
00000: long       display long object names without truncation
00000: map        display map file processing (ld only)
00000: move       display move section processing
00000: reloc      display relocation processing
00000: sections   display input section processing (ld only)
00000: segments   display available output segments and address/offset
00000:              processing; detail flag shows associated sections (ld only)
00000: statistics display processing statistics (ld only)
00000: strtab     display information about string table compression; detail
00000:              shows layout of string tables (ld only)
00000: support    display support library processing (ld only)
00000: symbols    display symbol table processing; detail flag shows internal
00000:              symbol table addition and resolution (ld only)
00000: tls        display TLS processing info
00000: unused     display unused/unreferenced files; detail flag shows unused
00000:              sections (ld only)
00000: versions   display version processing
Thu Mar 10 21:28:23 EST 2005
solaris>
```

# Runtime Linker Debug - Libs

```
solaris> LD_DEBUG=libs /opt/filebench/bin/filebench
13686:
13686: hardware capabilities - 0x2b  [ VIS  V8PLUS  DIV32  MUL32 ]
...
13686: find object=libc.so.1; searching
13686:  search path=/lib  (default)
13686:  search path=/usr/lib  (default)
13686:  trying path=/lib/libc.so.1
13686: 1:
13686: 1: calling .init (from sorted order): /lib/libc.so.1
13686: 1:
13686: 1:
13686: 1: calling .init (done): /lib/libc.so.1
13686: 1:
13686: 1:
13686: 1:
13686: 1: transferring control: /opt/filebench/bin/filebench
13686: 1:
13686: 1:  trying path=/platform/SUNW,Ultra-Enterprise/lib/libc_psr.so.1
...
13686: find object=libm.so.2; searching
13686:  search path=/usr/lib/lwp/sparcv9  (RPATH from file /
opt/filebench/bin/sparcv9/filebench)
13686:  trying path=/usr/lib/lwp/sparcv9/libm.so.2
13686:  search path=/lib/64  (default)
13686:  search path=/usr/lib/64  (default)
13686:  trying path=/lib/64/libm.so.2
13686:
13686: find object=libl.so.1; searching
13686:  search path=/usr/lib/lwp/sparcv9  (RPATH from file /
opt/filebench/bin/sparcv9/filebench)
13686:  trying path=/usr/lib/lwp/sparcv9/libl.so.1
13686:  search path=/lib/64  (default)
13686:  search path=/usr/lib/64  (default)
13686:  trying path=/lib/64/libl.so.1
13686:  trying path=/usr/lib/64/libl.so.1
```

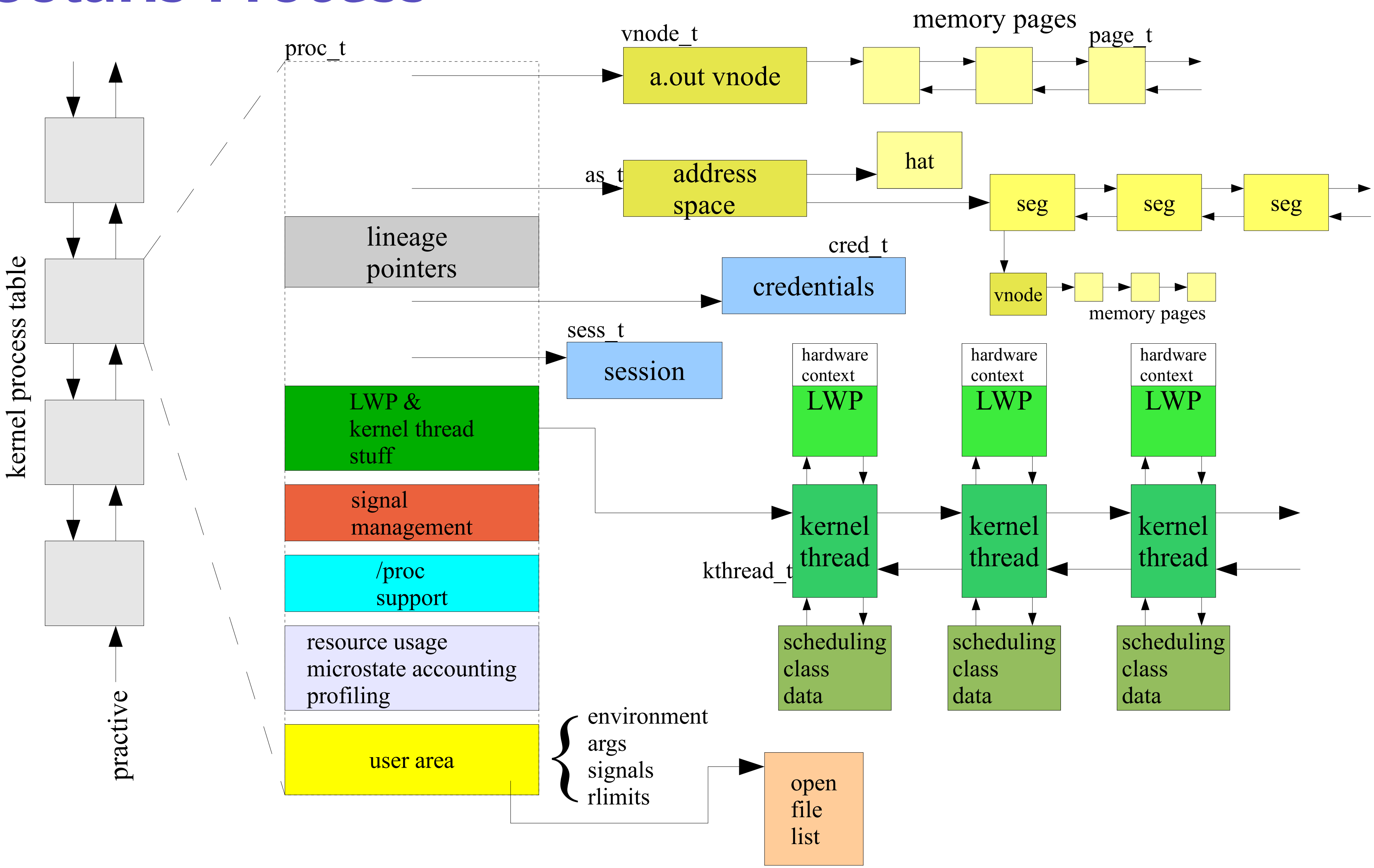
# Runtime Linker Debug - Bindings

```
solaris> LD_DEBUG=bindings /opt/filebench/bin/filebench
15151:
15151: hardware capabilities - 0x2b  [ VIS  V8PLUS  DIV32  MUL32 ]
15151:
15151:
15151: configuration file=/var/ld/ld.config: unable to process file
15151:
15151: binding file=/opt/filebench/bin/filebench to 0x0 (undefined weak): symbol
`__1cG__CrunMdo_exit_code6F_v_'
15151: binding file=/opt/filebench/bin/filebench to file=/lib/libc.so.1: symbol `__iob'
15151: binding file=/lib/libc.so.1 to 0x0 (undefined weak): symbol `__tnf_probe_notify'
15151: binding file=/lib/libc.so.1 to file=/opt/filebench/bin/filebench: symbol `_end'
15151: binding file=/lib/libc.so.1 to 0x0 (undefined weak): symbol `_ex_unwind'
15151: binding file=/lib/libc.so.1 to file=/lib/libc.so.1: symbol `__fnmatch_C'
15151: binding file=/lib/libc.so.1 to file=/lib/libc.so.1: symbol `__getdate_std'
...
15151: binding file=/opt/filebench/bin/sparcv9/filebench to file=/lib/64/libc.so.1: symbol
`__iob'
15151: binding file=/opt/filebench/bin/sparcv9/filebench to file=/lib/64/libc.so.1: symbol
`optarg'
15151: binding file=/lib/64/libm.so.2 to file=/opt/filebench/bin/sparcv9/filebench: symbol
`free'
15151: binding file=/lib/64/libm.so.2 to file=/lib/64/libm.so.2: symbol `__signgamf'
15151: binding file=/lib/64/libm.so.2 to file=/lib/64/libm.so.2: symbol `__signgaml'
15151: binding file=/lib/64/libm.so.2 to file=/lib/64/libm.so.2: symbol `__xpg6'
...
15151: 1: binding file=/lib/64/libc.so.1 to file=/lib/64/libc.so.1: symbol `_sigemptyset'
15151: 1: binding file=/lib/64/libc.so.1 to file=/lib/64/libc.so.1: symbol `_sigaction'
15151: 1: binding file=/opt/filebench/bin/sparcv9/filebench to file=/lib/64/libc.so.1:
symbol `feof'
15151: 2: binding file=/opt/filebench/bin/sparcv9/filebench to file=/lib/64/libc.so.1:
symbol `sleep'
15151: 1: binding file=/opt/filebench/bin/sparcv9/filebench to file=/lib/64/libc.so.1:
symbol `printf'
15151: 1: binding file=/lib/64/libc.so.1 to file=/lib/64/libc.so.1: symbol `_findbuf'
```

# Runtime Linker – Debug

- Explore the options in *The Linker and Libraries Guide*

# Solaris Process



# Process Structure

```
# mdb -k
Loading modules: [ unix krtld genunix specfs dtrace ufs ip sctp usba fctl nca lofs nfs random
sppp crypto ptm logindmux cpc ]
> ::ps
S    PID   PPID   PGID    SID    UID      FLAGS             ADDR NAME
R      0      0      0      0      0 0x00000001 fffffffffbc1ce80 sched
R      3      0      0      0      0 0x00020001 ffffffff880838f8 fsflush
R      2      0      0      0      0 0x00020001 ffffffff88084520 pageout
R      1      0      0      0      0 0x42004000 ffffffff88085148 init
R  21344      1  21343  21280   2234 0x42004000 ffffffff95549938 tcpPerfServer
...
> ffffffff95549938::print proc_t
{
    p_exec = 0xffffffff9285dc40
    p_as = 0xffffffff87c776c8
    p_cred = 0xffffffff8fdeb448
    p_lwpcnt = 0x6
    p_zombcnt = 0
    p_tlist = 0xffffffff8826bc20
        .....
        u_ticks = 0x16c6f425
        u_comm = [ "tcpPerfServer" ]
        u_psargs = [ "/export/home/morgan/work/solaris_studio9/bin/tcpPerfServer 9551 9552" ]
        u_argc = 0x3
        u_argv = 0x8047380
        u_envp = 0x8047390
        u_cdir = 0xffffffff8bf3d7c0
        u_saved_rlimit = [
            {
                rlim_cur = 0xfffffffffffffffd
                rlim_max = 0xfffffffffffffffd
            }
        ......
            fi_nfiles = 0x3f
            fi_list = 0xffffffff8dc44000
            fi_rlist = 0
        }
    p_model = 0x100000
    p_rctls = 0xffffffffa7cbb4c8
    p_dtrace_probes = 0
    p_dtrace_count = 0
    p_dtrace_helpers = 0
    p_zone = zone0
```

# The Life Of A Process

- Process creation
  - fork(2) system call creates all processes
    - SIDL state
  - exec(2) overlays newly created process with executable image
- State Transitions
  - Typically runnable (SRUN), running (SONPROC) or sleeping (aka blocked, SSLEEP)
  - May stopped (debugger) SSTOP
- Termination
  - SZOMB state
  - implicit or explicit exit(), signal (kill), fatal error

# Process Creation

- Traditional UNIX fork/exec model
  - fork(2) - replicate the entire process, including all threads
  - fork1(2) - replicate the process, only the calling thread
  - vfork(2) - replicate the process, but do not dup the address space
    - The new child borrows the parents address space, until exec()

```
main(int argc, char *argv[])
{
        pid_t pid;
        pid = fork();
        if (pid == 0)   /* in the child */
                exec();
        else if (pid > 0)    /* in the parent */
                wait();
        else
                fork failed
}
```

# Process create example

## C code calling fork()

```
#include <sys/types.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
        pid_t   ret, cpid, ppid;

        ppid = getpid();
        ret = fork();
        if (ret == -1) {
                perror("fork");
                exit(0);
        } else if (ret == 0) {
                printf("In child...\n");
        } else {
                printf("Child PID: %d\n",ret);
        }
        exit(0);
}
```

## D script to generate kernel trace

```
#!/usr/sbin/dtrace -Fs

syscall::fork1:entry
/ pid == $target /
{
        self->trace = 1;
}

fbt:::
/ self->trace /
{
}

syscall::fork1:return
/ pid == $target /
{
        self->trace = 0;
        exit(0);
}
```

# Fork Kernel Trace

```
CPU FUNCTION
  0  -> fork1
  0  <- fork1
  0  -> cfork
  0    -> secpolicy_basic_fork
  0    <- secpolicy_basic_fork
  0    -> priv_policy
  0    <- priv_policy
  0    -> holdlwps
  0      -> schedctl_finish_sigblock
  0      <- schedctl_finish_sigblock
  0      -> pokelwps
  0      <- pokelwps
  0    <- holdlwps
  0    -> flush_user_windows_to_stack
  0    -> getproc
  0      -> page_mem_avail
  0      <- page_mem_avail
  0      -> zone_status_get
  0      <- zone_status_get
  0      -> kmem_cache_alloc
  0        -> kmem_cpu_reload
  0        <- kmem_cpu_reload
  0      <- kmem_cache_alloc
  0      -> pid_assign
  0        -> kmem_zalloc
  0          <- kmem_cache_alloc
  0        <- kmem_zalloc
  0        -> pid_lookup
  0        -> pid_getlockslot
  0      -> crgetruid
  0      -> crgetzoneid
  0      -> upcount_inc
  0        -> rctl_set_dup
                ...
  0          -> project_cpu_shares_set
  0          -> project_lwps_set
  0          -> project_ntasks_set
                ...
  0        <- rctl_set_dup
```

# Fork Kernel Trace (cont)

```
0    -> as_dup
              ...
0        <- hat_alloc
0      <- as_alloc
0      -> seg_alloc
0      -> rctl_set_fill_alloc_gp
0    <- rctl_set_dup_ready
0    -> rctl_set_dup
              ...
0    -> forklwp
0      <- flush_user_windows_to_stack
0      -> save_syscall_args
0      -> lwp_create
0        <- thread_create
0        -> lwp_stk_init
0        -> kmem_zalloc
0      <- lwp_create
0      -> init_mstate
0      -> lwp_forkregs
0      -> forkctx
0      -> ts_alloc
0      -> ts_fork
0    <- forklwp
0    -> contract_process_fork
0    -> ts_forkret
0      -> continuelwps
0      -> ts_setrun
0      -> setbackdq
0      -> generic_enq_thread
0    <- ts_forkret
0    -> swtch
0      -> disp
0    <- swtch
0    -> resume
0      -> savectx
0      <- savectx
0      -> restorectx
0    <- resume
0  <- cfork
0  <= fork1
```

# Watching Forks

## D script for watching fork(2)

```
#!/usr/sbin/dtrace -qs

syscall::forkall:entry
{
        @fall[execname] = count();
}
syscall::fork1:entry
{
        @f1[execname] = count();
}
syscall::vfork:entry
{
        @vf[execname] = count();
}

dtrace:::END
{
        printf("forkall\n");
        printa(@fall);
        printf("fork1\n");
        printa(@f1);
        printf("vfork\n");
        printa(@vf);
}
```

## Example run

```
# ./watchfork.d
^C
forkall

fork1

  start-srvr             1
  bash                   3
  4cli                   6
vfork
```

# exec(2) – Load a new process image

- Most fork(2) calls are followed by an exec(2)
- exec – execute a new file
- exec overlays the process image with a new process constructed from the binary file passed as an arg to exec(2)
- The exec'd process inherits much of the caller's state:
  - nice value, scheduling class, priority, PID, PPID, GID, task ID, project ID, session membership, real UID & GID, current working directory, resource limits, processor binding, times, etc, ...

# Watching exec(2) with DTrace

- The D script...

```
#pragma D option quiet
proc:::exec
{
     self->parent = execname;
}
proc:::exec-success
/self->parent != NULL/
{
     @[self->parent, execname] = count();
     self->parent = NULL;
}
proc:::exec-failure
/self->parent != NULL/
{
     self->parent = NULL;
}
END
{
     printf("%-20s %-20s %s\n", "WHO", "WHAT", "COUNT");
     printa("%-20s %-20s %@d\n", @);
}
```

# Watching exec(2) with DTrace

- Example output:

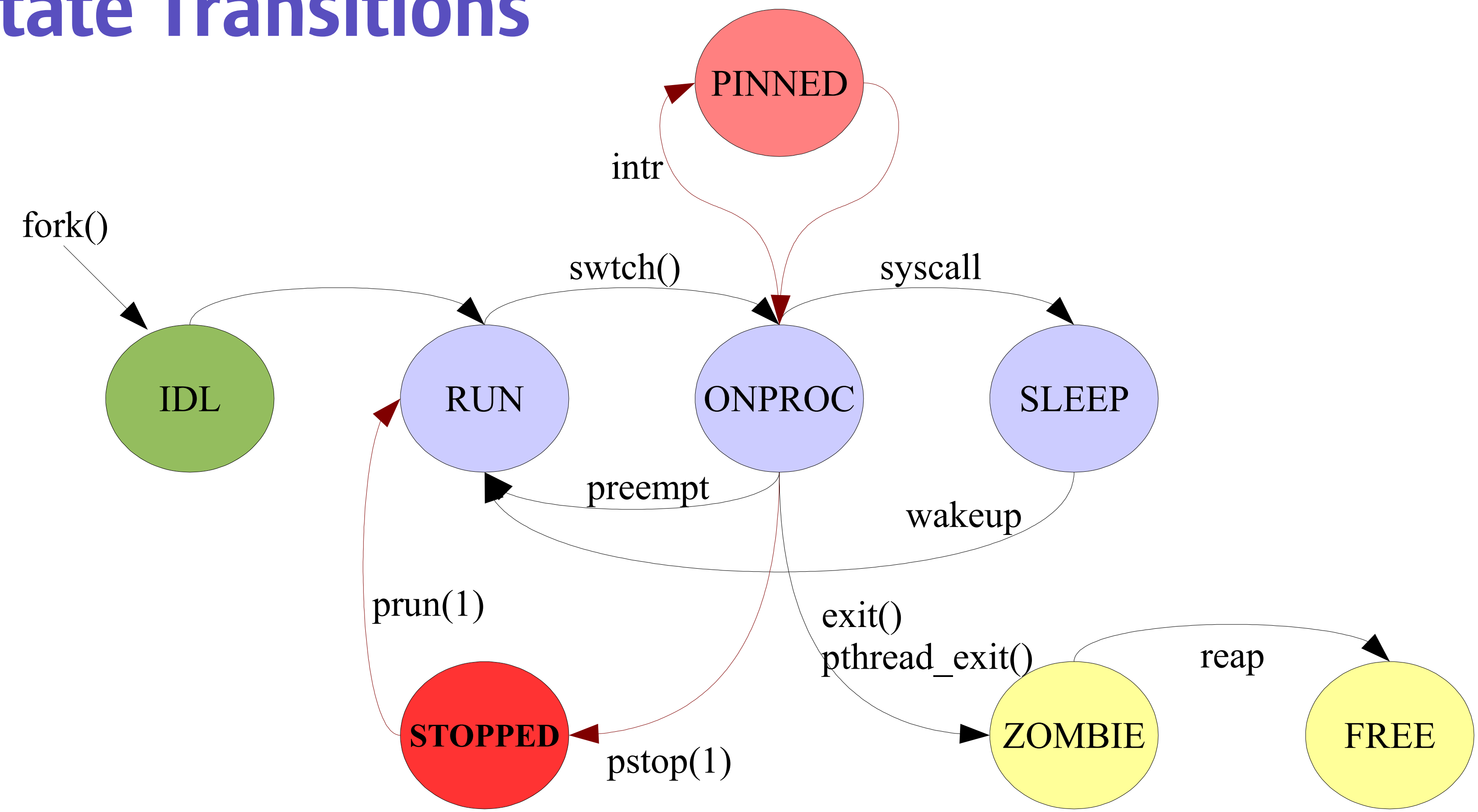
```
# dtrace -s ./whoexec.d
^C
WHO        WHAT       COUNT
make.bin   yacc       1
tcsh       make       1
make.bin   spec2map   1
sh         grep       1
lint       lint2      1
sh         lint       1
sh         ln         1
cc         ld         1
make.bin   cc         1
lint       lint1      1
```

# Process / Thread States

- It's really kernel threads that change state
- Kernel thread creation is not flagged as a distinct state
  - Initial state is TS_RUN
- Kernel threads are TS_FREE when the process, or LWP/kthread, terminates

| Process State | Kernel Thread State |
|---|---|
| SIDL | |
| SRUN | TS_RUN |
| SONPROC | TS_ONPROC |
| SSLEEP | TS_SLEEP |
| SSTOP | TS_STOPPED |
| SZOMB | TS_ZOMB |
| | TS_FREE |

# State Transitions



# Watching Process States

```
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 27946 root     4880K 4520K cpu0    59    0   0:00:00 0.7% prstat/1
 28010 root     4928K 2584K run     29    0   0:00:00 0.7% pkginstall/1
 23078 root       20M   14M sleep   59    0   0:00:57 0.3% lupi_zones/1
 25947 root     5160K 2976K sleep   59    0   0:00:04 0.3% sshd/1
 24866 root     5136K 2136K sleep   59    0   0:00:01 0.2% sshd/1
   202 root     3304K 1800K sleep   59    0   0:00:09 0.2% nscd/24
 23001 root     5136K 2176K sleep   59    0   0:00:04 0.1% sshd/1
 23860 root     5248K 2392K sleep   59    0   0:00:05 0.1% sshd/1
 25946 rmc      3008K 2184K sleep   59    0   0:00:02 0.1% ssh/1
 25690 root     1240K  928K sleep   59    0   0:00:00 0.1% sh/1
   830 root     2472K  696K sleep   59    0   0:18:53 0.1% mibiisa/7
   349 root     8600K  768K sleep   59    0   0:00:20 0.0% snmpd/1
   340 root     2504K  680K sleep   59    0   0:19:14 0.0% mibiisa/7
   829 root     2488K  696K sleep   59    0   0:18:48 0.0% mibiisa/7
 27328 root     1240K  928K sleep   59    0   0:00:00 0.0% sh/1
   490 daemon   2328K   16K sleep   59    0   0:00:00 0.0% rpcbind/1
   815 root     2016K    8K sleep   59    0   0:00:00 0.0% ttymon/1
   365 root     4760K  128K sleep   59    0   0:00:00 0.0% zoneadmd/4
   364 root     4776K  128K sleep   59    0   0:00:00 0.0% zoneadmd/4
   374 root        0K    0K sleep   60    -   0:00:00 0.0% zsched/1
   361 root     2016K    8K sleep   59    0   0:00:00 0.0% ttymon/1
   386 root     2096K  360K sleep   59    0   0:00:00 0.0% init/1
   387 root     2096K  376K sleep   59    0   0:00:00 0.0% init/1
   345 root     3160K  480K sleep   59    0   0:00:00 0.0% sshd/1
   591 root     3824K  184K sleep   59    0   0:00:00 0.0% automountd/2
   373 root        0K    0K sleep   60    -   0:00:00 0.0% zsched/1
  1718 nobody   6672K 2032K sleep   59    0   0:00:35 0.0% httpd/1
   322 root     3112K   16K sleep   59    0   0:00:00 0.0% dmispd/1
   328 root     2728K   40K sleep   59    0   0:00:01 0.0% vold/3
   488 daemon   2328K   16K sleep   59    0   0:00:00 0.0% rpcbind/1
   312 root     4912K   24K sleep   59    0   0:00:00 0.0% dtlogin/1
   250 root     4760K  696K sleep   59    0   0:00:16 0.0% sendmail/1
   246 root     1888K    0K sleep   59    0   0:00:00 0.0% smcboot/1
   823 root     1936K  224K sleep   59    0   0:00:00 0.0% sac/1
   242 root     1896K    8K sleep   59    0   0:00:00 0.0% smcboot/1
   248 smmsp    4736K  680K sleep   59    0   0:00:08 0.0% sendmail/1
   245 root     1888K    0K sleep   59    0   0:00:00 0.0% smcboot/1
   824 root     2016K    8K sleep   59    0   0:00:00 0.0% ttymon/1
   204 root     2752K  520K sleep   59    0   0:00:00 0.0% inetd/1
   220 root     1568K    8K sleep   59    0   0:00:00 0.0% powerd/3
   313 root     2336K  216K sleep   59    0   0:00:00 0.0% snmpdx/1
Total: 127 processes, 312 lwps, load averages: 0.62, 0.62, 0.53
```

# DTrace - exec(2)

- Tracing exec

```
#pragma D option quiet
proc:::exec
{
     self->parent = execname;
}
proc:::exec-success
/self->parent != NULL/
{
     @[self->parent, execname] = count();
     self->parent = NULL;
}
proc:::exec-failure
/self->parent != NULL/
{
     self->parent = NULL;
}
END
{
     printf("%-20s %-20s %s\n", "WHO", "WHAT", "COUNT");
     printa("%-20s %-20s %@d\n", @);
}
```

# Dtrace

- Example output:

```
# dtrace -s ./whoexec.d
^C
WHO         WHAT       COUNT
make.bin    yacc       1
tcsh        make       1
make.bin    spec2map   1
sh          grep       1
lint        lint2      1
sh          lint       1
sh          ln         1
cc          ld         1
make.bin    cc         1
lint        lint1      1
```

# Microstates

- Fine-grained state tracking for processes/threads
  - Off by default in Solaris 8 and Solaris 9
  - On by default in Solaris 10
- Can be enabled per-process via /proc
- `prstat -m` reports microstates
  - As a percentage of time for the sampling period
    - USR – user mode
    - SYS - kernel mode
    - TRP – trap handling
    - TFL – text page faults
    - DFL – data page faults
    - LCK – user lock wait
    - SLP - sleep
    - LAT – waiting for a processor (sitting on a run queue)

# prstat – process microstates

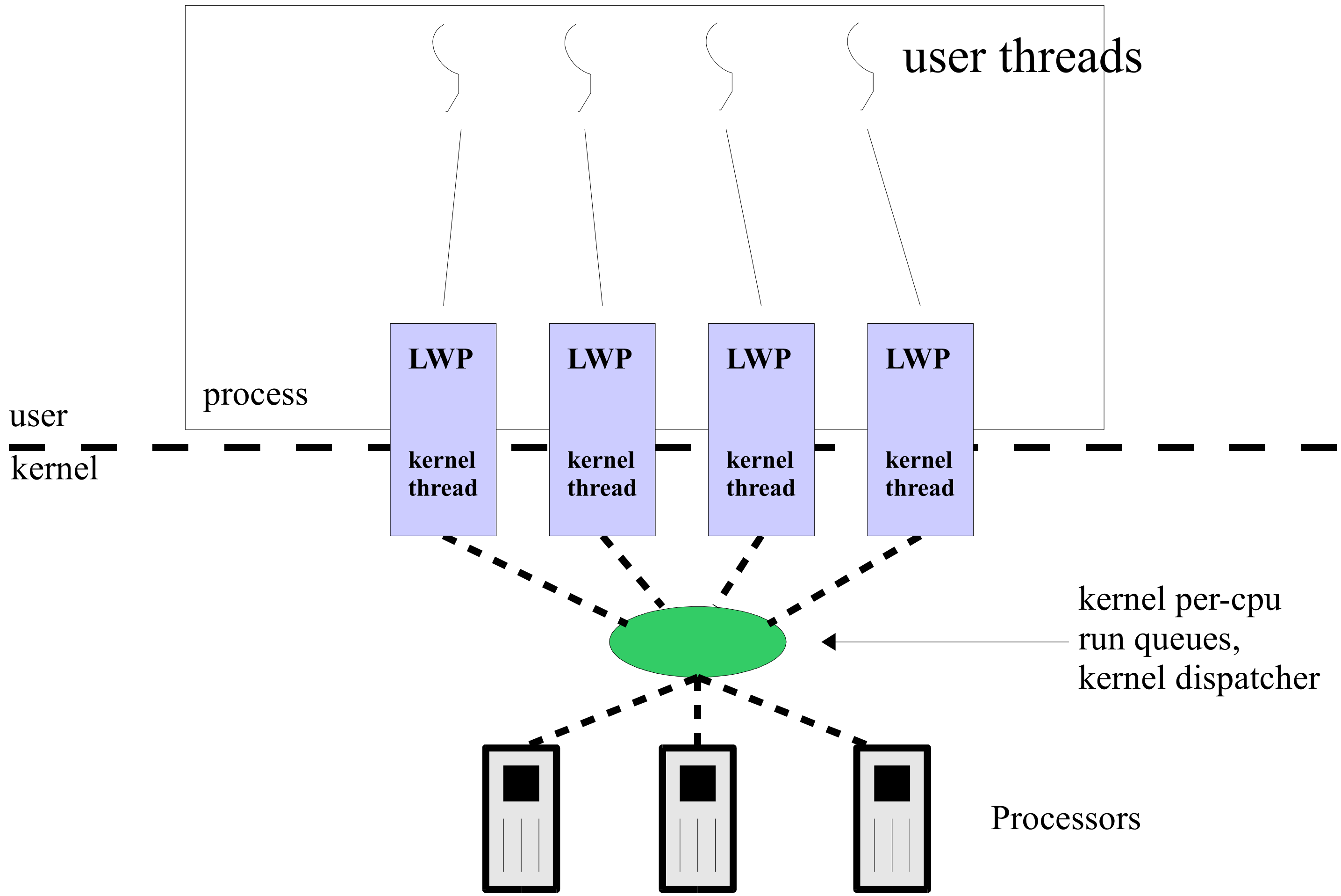
```
   PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID
 16787 allanp    18 6.7 0.2 0.0 0.0 0.0  74 0.2 493 119  1K   0 gzip/1
 16794 allanp   8.4  11 0.3 0.0 0.0 0.0  79 0.5 972 444  8K   0 tar/1
 15793 root     2.7 7.6 0.0 0.0 0.0  78  11 0.4 972 114 35K   0 filebench/5
 16784 root     3.7 6.6 0.4 0.0 0.0 0.0  89 0.1 127  44 10K   0 in.rshd/1
 15793 root     2.6 7.6 0.0 0.0 0.0  76  13 0.4  1K 147 31K   0 filebench/29
 15793 root     2.4 7.0 0.0 0.0 0.0  78  13 0.4 934 137 29K   0 filebench/18
 15793 root     2.2 7.1 0.0 0.0 0.0  78  12 0.4 974 124 27K   0 filebench/33
 15793 root     2.4 6.8 0.0 0.0 0.0  78  12 0.4 872 111 30K   0 filebench/30
 15793 root     2.3 6.5 0.0 0.0 0.0  80  11 0.4 860 126 29K   0 filebench/13
 15793 root     2.4 6.4 0.0 0.0 0.0  79  11 0.4 793 106 31K   0 filebench/20
 15793 root     2.4 6.2 0.0 0.0 0.0  81  10 0.3 749  99 32K   0 filebench/27
 15793 root     2.0 5.9 0.0 0.0 0.0  80  11 0.3 798 112 25K   0 filebench/2
 15793 root     1.9 6.0 0.0 0.0 0.0  79  13 0.3 731  99 23K   0 filebench/4
 15793 root     1.9 5.5 0.0 0.0 0.0  80  12 0.3 729 106 24K   0 filebench/8
 15793 root     2.0 5.5 0.0 0.0 0.0  81  11 0.3 709 131 24K   0 filebench/11
Total: 93 processes, 226 lwps, load averages: 3.25, 1.81, 1.36
```

# Threads

# T2 – Single Level Threads Model

- The default model in Solaris 9 and 10
- All user threads bound to LWPs
  - All bound threads
- Kernel level scheduling
  - No more libthread.so scheduler
- Simplified Implementation
- Uses kernel's synchronization objects
  - Slightly different behaviour LIFO vs. FIFO
  - Allows adaptive lock behaviour
- More expensive thread create/destroy, synchronization
- More responsive scheduling, synchronization

# T2 – Single Level Threads Model



# T2 - Single Level Thread Model

- Scheduling wrt Synchronization (S8U7/S9/S10)
  - Adaptive locks give preference to a thread that is running, potentially at the expense of a thread that is sleeping
  - Threads that rely on fairness of scheduling/CPU could end up ping-ponging, at the expense of another thread which has work to do.
- Default S8U7/S9/S10 Behaviour
  - Adaptive Spin
    - 1000 of iterations (spin count) for adaptive mutex locking before giving up and going to sleep.
  - Maximum number of spinners
    - The number of simultaneously spinning threads
    - attempting to do adaptive locking on one mutex is limited to 100.
  - One out of every 16 queuing operations will put a thread at the end of the queue, to prevent starvation.
  - Stack Cache
    - The maximum number of stacks the library retains after threads exit for re-use when more threads are created is 10.

# Watching Threads

```
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/LWPID
 29105 root     5400K 3032K sleep   60    0   0:00:00 1.3% pkginstall/1
 29051 root     5072K 4768K cpu0    49    0   0:00:00 0.8% prstat/1
   202 root     3304K 1256K sleep   59    0   0:00:07 0.3% nscd/23
 25947 root     5160K  608K sleep   59    0   0:00:05 0.2% sshd/1
 23078 root       20M 1880K sleep   59    0   0:00:58 0.2% lupi_zones/1
 25946 rmc      3008K  624K sleep   59    0   0:00:02 0.2% ssh/1
 23860 root     5248K  688K sleep   59    0   0:00:06 0.2% sshd/1
 29100 root     1272K  976K sleep   59    0   0:00:00 0.1% mpstat/1
 24866 root     5136K  600K sleep   59    0   0:00:02 0.0% sshd/1
   340 root     2504K  672K sleep   59    0   0:11:14 0.0% mibiisa/2
 23001 root     5136K  584K sleep   59    0   0:00:04 0.0% sshd/1
   830 root     2472K  600K sleep   59    0   0:11:01 0.0% mibiisa/2
   829 root     2488K  648K sleep   59    0   0:11:01 0.0% mibiisa/2
     1 root     2184K  400K sleep   59    0   0:00:01 0.0% init/1
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/13
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/12
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/11
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/10
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/9
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/8
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/7
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/6
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/5
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/4
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/3
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/2
   202 root     3304K 1256K sleep   59    0   0:00:00 0.0% nscd/1
   126 daemon   2360K    8K sleep   59    0   0:00:00 0.0% rpcbind/1
   814 root     1936K  280K sleep   59    0   0:00:00 0.0% sac/1
    64 root     2952K    8K sleep   59    0   0:00:00 0.0% picld/5
    64 root     2952K    8K sleep   59    0   0:00:00 0.0% picld/4
    64 root     2952K    8K sleep   59    0   0:00:00 0.0% picld/3
    64 root     2952K    8K sleep   59    0   0:00:00 0.0% picld/2
    64 root     2952K    8K sleep   59    0   0:00:00 0.0% picld/1
    61 daemon   3640K    8K sleep   59    0   0:00:00 0.0% kcfd/3
    61 daemon   3640K    8K sleep   59    0   0:00:00 0.0% kcfd/2
    61 daemon   3640K    8K sleep   59    0   0:00:00 0.0% kcfd/1
    55 root     2416K    8K sleep   59    0   0:00:00 0.0% syseventd/14
    55 root     2416K    8K sleep   59    0   0:00:00 0.0% syseventd/13
    55 root     2416K    8K sleep   59    0   0:00:00 0.0% syseventd/12
    55 root     2416K    8K sleep   59    0   0:00:00 0.0% syseventd/11
Total: 125 processes, 310 lwps, load averages: 0.50, 0.38, 0.40
```

# Examining A Thread Structure

```
# mdb -k
Loading modules: [ unix krtld genunix specfs dtrace ufs ip sctp usba fctl nca lofs nfs random sppp
crypto ptm logindmux cpc ]
> ::ps
S    PID   PPID   PGID    SID    UID      FLAGS             ADDR NAME
R      0      0      0      0      0 0x00000001 fffffffffbc1ce80 sched
R      3      0      0      0      0 0x00020001 ffffffff880838f8 fsflush
R      2      0      0      0      0 0x00020001 ffffffff88084520 pageout
R      1      0      0      0      0 0x42004000 ffffffff88085148 init
R  21344      1  21343  21280   2234 0x42004000 ffffffff95549938 tcpPerfServer
> ffffffff95549938::print proc_t
{
    p_exec = 0xffffffff9285dc40
    p_as = 0xffffffff87c776c8
        ...
    p_tlist = 0xffffffff8826bc20
        ...
> ffffffff8826bc20::print kthread_t
{
    t_link = 0
    t_stk = 0xfffffe8000161f20
    t_startpc = 0
    t_bound_cpu = 0
    t_affinitycnt = 0
    t_bind_cpu = 0xffff
    t_cid = 0x1
    t_clfuncs = ts_classfuncs+0x48
    t_cldata = 0xffffffffa5f0b2a8
    t_cpu = 0xffffffff87c80800
    t_lbolt = 0x16c70239
    t_disp_queue = 0xffffffff87c86d28
    t_disp_time = 0x16c7131a
    t_kpri_req = 0
    t_stkbase = 0xfffffe800015d000
    t_sleepq = sleepq_head+0x1270
    t_dtrace_regv = 0
    t_hrtime = 0x1dc821f2628013
}
```

# Thread Semantics Added to pstack, truss

```
# pstack 909/2
909:    dbwr -a dbwr -i 2 -s b0000000 -m /var/tmp/fbencAAAmxaqxb
-----------------  lwp# 2  --------------------------------
 ceab1809 lwp_park (0, afffde50, 0)
 ceaabf93 cond_wait_queue (ce9f8378, ce9f83a0, afffde50, 0) + 3b
 ceaac33f cond_wait_common (ce9f8378, ce9f83a0, afffde50) + 1df
 ceaac686 _cond_reltimedwait (ce9f8378, ce9f83a0, afffdea0) + 36
 ceaac6b4 cond_reltimedwait (ce9f8378, ce9f83a0, afffdea0) + 24
 ce9e5902 __aio_waitn (82d1f08, 1000, afffdf2c, afffdf18, 1) + 529
 ceaf2a84 aio_waitn64 (82d1f08, 1000, afffdf2c, afffdf18) + 24
 08063065 flowoplib_aiowait (b4eb475c, c40f4d54) + 97
 08061de1 flowop_start (b4eb475c) + 257
 ceab15c0 _thr_setup (ce9a8400) + 50
 ceab1780 _lwp_start (ce9a8400, 0, 0, afffdff8, ceab1780, ce9a8400)
```

```
pae1> truss -p 2975/3
/3:   close(5)                              = 0
/3:   open("/space1/3", O_RDWR|O_CREAT, 0666)       = 5
/3:   lseek(5, 0, SEEK_SET)                         = 0
/3:   write(5, " U U U U U U U U U U U U"..., 1056768)    = 1056768
/3:   lseek(5, 0, SEEK_SET)                         = 0
/3:   read(5, " U U U U U U U U U U U U"..., 1056768)     = 1056768
/3:   close(5)                              = 0
/3:   open("/space1/3", O_RDWR|O_CREAT, 0666)       = 5
/3:   lseek(5, 0, SEEK_SET)                         = 0
/3:   write(5, " U U U U U U U U U U U U"..., 1056768)    = 1056768
```

# Thread Microstates

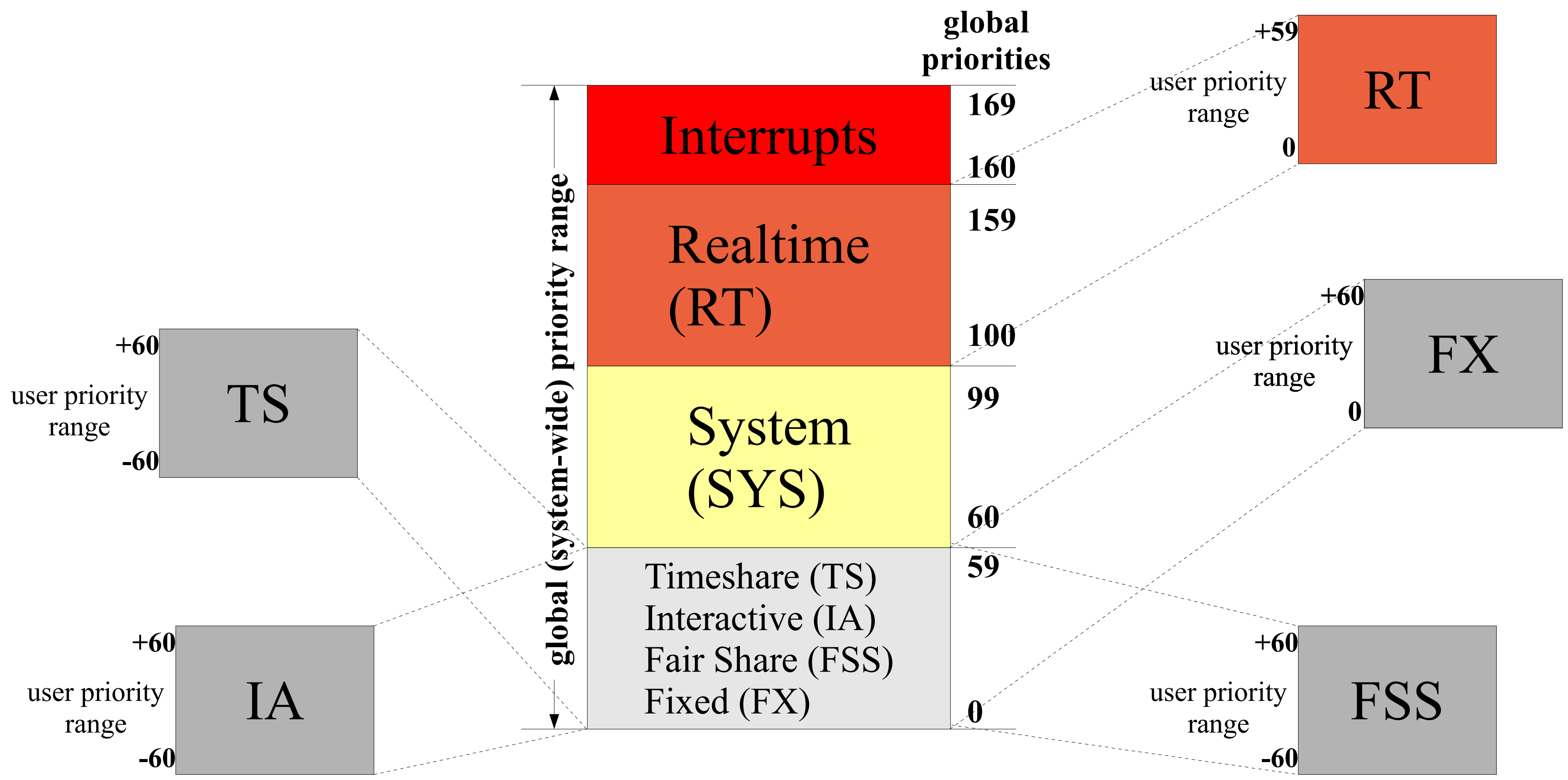
```
 PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID
 28987 root     2.0 5.8 0.0 0.0 0.0  79  12 0.9 745 107 23K   0 filebench/33
 28987 root     2.0 5.8 0.0 0.0 0.0  79  13 0.9 738 112 22K   0 filebench/17
 28987 root     2.0 5.7 0.0 0.0 0.0  79  12 0.9 723 112 23K   0 filebench/31
 28987 root     1.9 5.7 0.0 0.0 0.0  79  13 0.9 756 111 22K   0 filebench/12
 28987 root     2.1 5.4 0.0 0.0 0.0  80  12 0.9 653 100 20K   0 filebench/21
 28987 root     1.9 5.6 0.0 0.0 0.0  79  12 0.9 724 109 22K   0 filebench/8
 28987 root     1.9 5.6 0.0 0.0 0.0  79  12 1.0 726 105 21K   0 filebench/3
 28987 root     1.9 5.5 0.0 0.0 0.0  79  12 0.8 682 103 22K   0 filebench/9
 28987 root     1.9 5.5 0.0 0.0 0.0  79  12 0.9 731 111 21K   0 filebench/15
 28987 root     1.9 5.5 0.0 0.0 0.0  79  12 0.9 699 107 21K   0 filebench/16
 28987 root     1.8 5.5 0.0 0.0 0.0  79  13 0.9 710 113 20K   0 filebench/20
 28987 root     1.8 5.4 0.0 0.0 0.0  79  12 0.9 698 107 20K   0 filebench/5
 28987 root     1.9 5.4 0.0 0.0 0.0  79  13 0.9 699 103 21K   0 filebench/22
 28987 root     1.9 5.4 0.0 0.0 0.0  80  12 0.9 679 107 21K   0 filebench/26
 28987 root     1.8 5.4 0.0 0.0 0.0  79  12 0.9 702 107 21K   0 filebench/18
Total: 1 processes, 33 lwps, load averages: 3.10, 2.04, 1.70
```

# Who's Creating Threads?

```
# dtrace -n 'thread_create:entry { @[execname]=count()}'
dtrace: description 'thread_create:entry ' matched 1 probe
^C

  sh                                                              1
  sched                                                           1
  do1.6499                                                        2
  do1.6494                                                        2
  do1.6497                                                        2
  do1.6508                                                        2
  in.rshd                                                        12
  do1.6498                                                       14
  do1.6505                                                       16
  do1.6495                                                       16
  do1.6504                                                       16
  do1.6502                                                       16
  automountd                                                     17
  inetd                                                          19
  filebench                                                      34
  find                                                          130
  csh                                                           177
```

# Scheduling Classes & The Kernel Dispatcher

# Scheduling Classes and Priorities



# Scheduling Classes

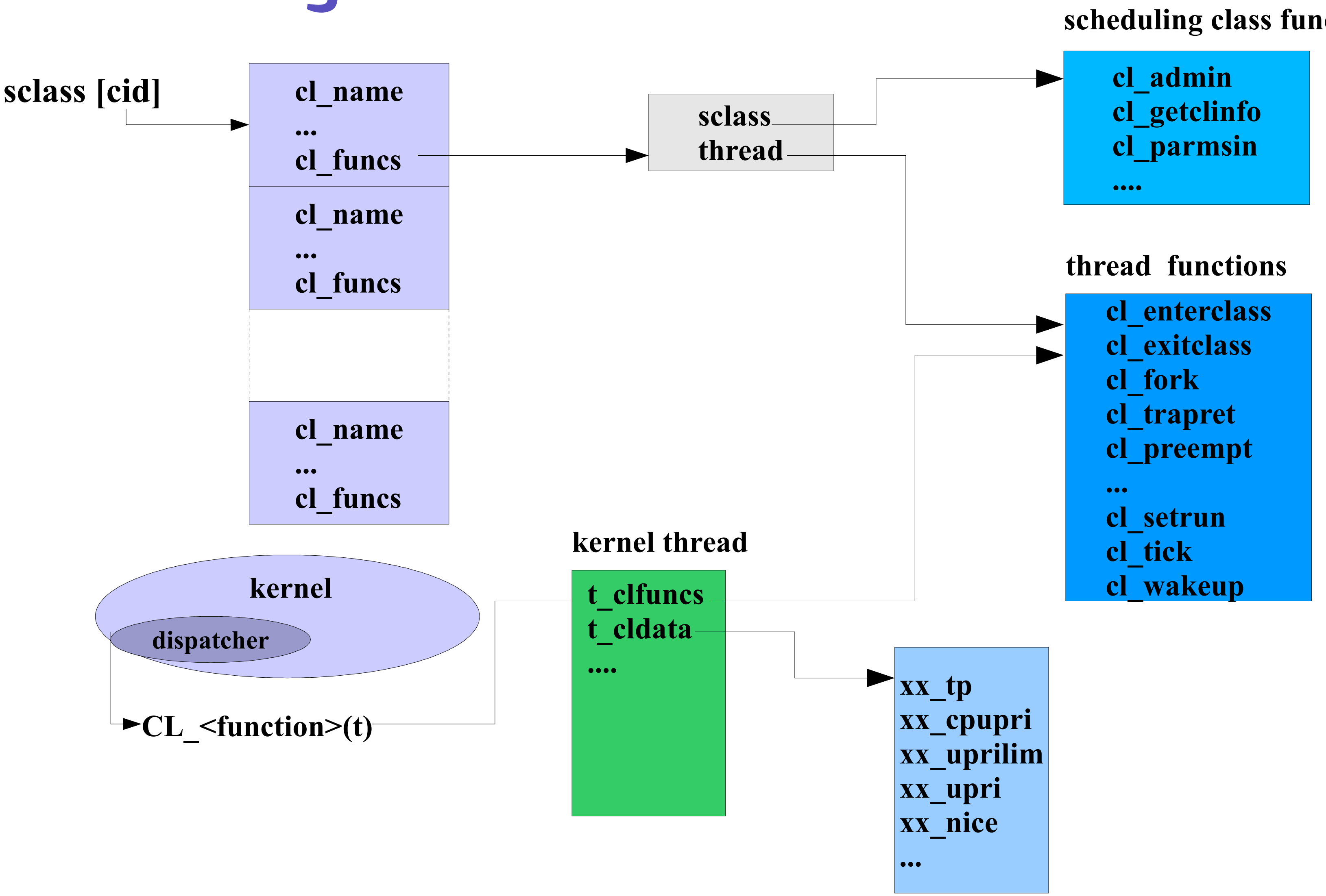
- Use `dispadmin(1M)` and `priocntl(1)`

```
# dispadmin -l
CONFIGURED CLASSES
==================

SYS   (System Class)
TS    (Time Sharing)
FX    (Fixed Priority)
IA    (Interactive)
FSS   (Fair Share)
RT    (Real Time)
# priocntl -l
CONFIGURED CLASSES
==================

SYS (System Class)

TS (Time Sharing)
      Configured TS User Priority Range: -60 through 60

FX (Fixed priority)
      Configured FX User Priority Range: 0 through 60

IA (Interactive)
      Configured IA User Priority Range: -60 through 60

FSS (Fair Share)
      Configured FSS User Priority Range: -60 through 60

RT (Real Time)
      Maximum Configured RT Priority: 59
#
```

# Scheduling Classes

- The kernel maintains an array of sclass structures for each loaded scheduling class
  - References the scheduling classes init routine, class functions structure, etc
- Scheduling class information is maintained for every kernel thread
  - Thread pointer to the class functions array, and per-thread class-specific data structure
  - Different threads in the same process can be in different scheduling classes
- Scheduling class operations vectors and CL_XXX macros allow a single, central dispatcher to invoke scheduling-class specific functions

# Scheduling Class Functions



# Scheduling Class Array

```
# mdb -k
Loading modules: [ unix krtld genunix ip ufs_log nfs isp random ptm logindmux ]
> ::class
SLOT NAME       INIT FCN                 CLASS FCN
   0 SYS        sys_init                 sys_classfuncs
   1 TS         ts_init                  ts_classfuncs
   2 FX         fx_init                  fx_classfuncs
   3 IA         ia_init                  ia_classfuncs
   4 FSS        fss_init                 fss_classfuncs
   5 RT         rt_init                  rt_classfuncs
>
```

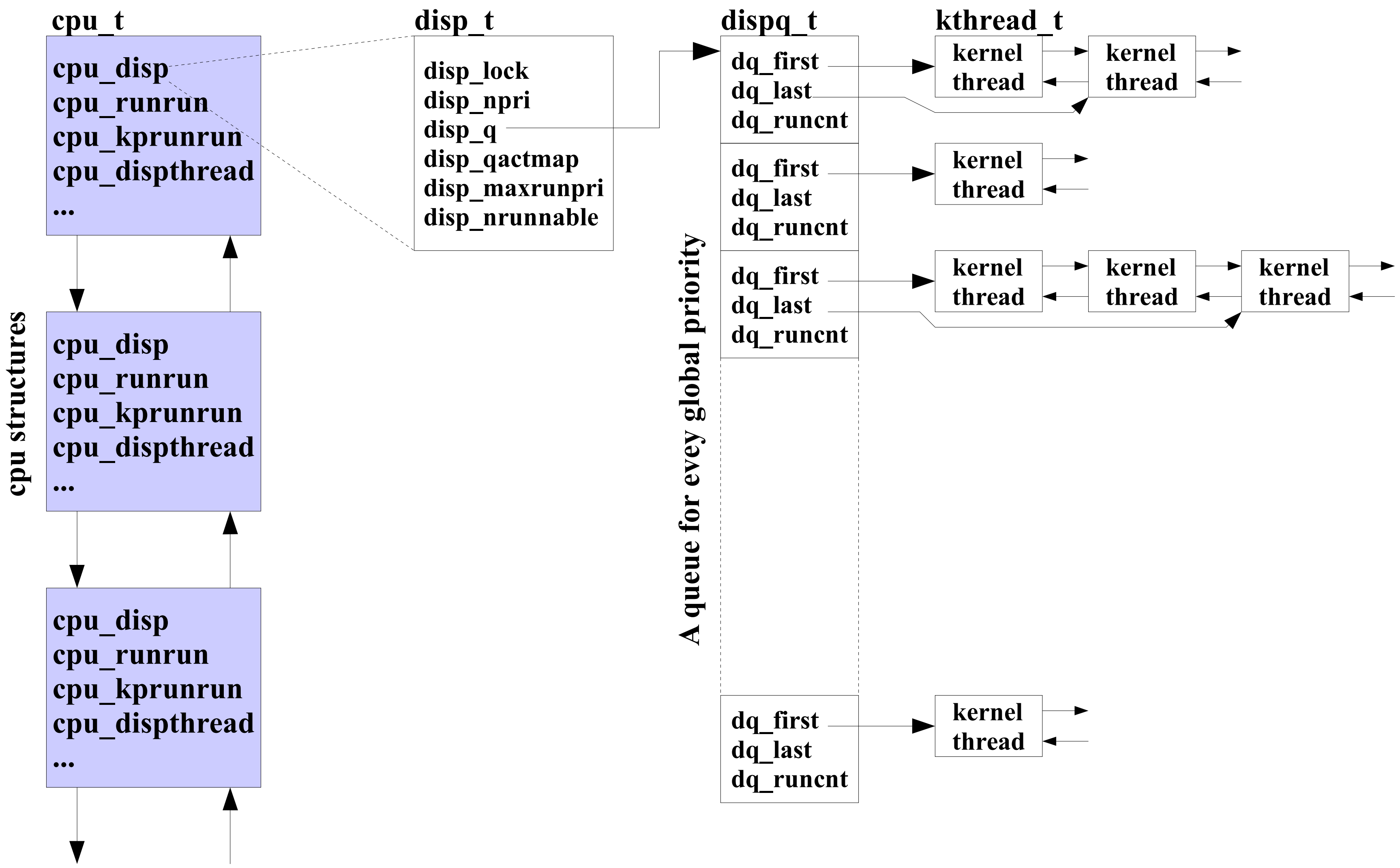
# Scheduling Class & Priority of Threads

```
solaris10> ps -eLc
   PID   LWP  CLS PRI TTY        LTIME CMD
     0     1  SYS  96 ?           0:00 sched
     1     1   TS  59 ?           0:00 init
     2     1  SYS  98 ?           0:00 pageout
     3     1  SYS  60 ?           5:08 fsflush
   402     1   TS  59 ?           0:00 sac
   269     1   TS  59 ?           0:00 utmpd
   225     1   TS  59 ?           0:00 automoun
   225     2   TS  59 ?           0:00 automoun
   225     4   TS  59 ?           0:00 automoun
    54     1   TS  59 ?           0:00 sysevent
    54     2   TS  59 ?           0:00 sysevent
    54     3   TS  59 ?           0:00 sysevent
    [snip]
   426     1   IA  59 ?           0:00 dtgreet
   343     1   TS  59 ?           0:00 mountd
   345     1   FX  60 ?           0:00 nfsd
   345     3   FX  60 ?           0:00 nfsd
   350     1   TS  59 ?           0:00 dtlogin
   375     1   TS  59 ?           0:00 snmpdx
   411     1   IA  59 ?           0:00 dtlogin
   412     1   IA  59 ??          0:00 fbconsol
   403     1   TS  59 console     0:00 ttymon
   405     1   TS  59 ?           0:00 ttymon
   406     1   IA  59 ?           0:03 Xsun
   410     1   TS  59 ?           0:00 sshd
   409     1   TS  59 ?           0:00 snmpd
  1040     1   TS  59 ?           0:00 in.rlogi
  1059     1   TS  49 pts/2       0:00 ps
solaris10>
```

# Dispatch Queues & Dispatch Tables

- Dispatch queues
  - Per-CPU run queues
    - Actually, a queue of queues
  - Ordered by thread priority
  - Queue occupation represented via a bitmap
  - For Realtime threads, a system-wide kernel preempt queue is maintained
    - Realtime threads are placed on this queue, not the per-CPU queues
    - If processor sets are configured, a kernel preempt queue exists for each processor set
- Dispatch tables
  - Per-scheduling class parameter tables
  - Time quantums and priorities
  - tuneable via dispadmin(1M)

# Per-CPU Dispatch Queues



# Timeshare Dispatch Table

- TS and IA class share the same dispatch table
  - RES. Defines the granularity of ts_quantum
  - ts_quantum. CPU time for next ONPROC state
  - ts_tqexp. New priority if time quantum expires
  - ts_slpret. New priority when state change from TS_SLEEP to TS_RUN
  - ts_maxwait. “waited to long” ticks
  - ts_lwait. New priority if “waited to long”

```
# dispadmin -g -c TS
# Time Sharing Dispatcher Configuration
RES=1000

# ts_quantum            ts_tqexp    ts_slpret  ts_maxwait     ts_lwait                PRIORITY LEVEL
       200                  0          50           0             50                     #        0
       200                  0          50           0             50                     #        1
        .........
       160                  0          51           0             51                     #        10
       160                  1          51           0             51                     #        11
        ..........
       120                 10          52           0             52                     #        20
       120                 11          52           0             52                     #        21
        .........
       80                  20          53           0             53                     #        30
       80                  21          53           0             53                     #        31
        ..........
       40                  30          55           0             55                     #        40
       40                  31          55           0             55                     #        41
        ...........
       20                  49          59       32000             59                     #        59
```

# RT, FX & FSS Dispatch Tables

- RT
  - Time quantum only
  - For each possible priority
- FX
  - Time quantum only
  - For each possible priority
- FSS
  - Time quantum only
  - Just one, not defined for each priority level
    - Because FSS is share based, not priority based
- SYS
  - No dispatch table
  - Not needed, no rules apply
- INT
  - Not really a scheduling class

# Dispatch Queue Placement

- Queue placement is based a few simple parameters
  - The thread priority
  - Processor binding/Processor set
  - Processor thread last ran on
    - Warm affinity
  - Depth and priority of existing runnable threads
  - Solaris 9 added Memory Placement Optimization (MPO) enabled will keep thread in defined locality group (lgroup)

```
if (thread is bound to CPU-n) && (pri < kpreemptpri)
        CPU-n dispatch queue
if (thread is bound to CPU-n) && (pri >= kpreemptpri)
        CPU-n dispatch queue
if (thread is not bound) && (pri < kpreemptpri)
    place thread on a CPU dispatch queue
if (thread is not bound) && (pri >= kpreemptpri)
    place thread on cp_kp_queue
```

# Thread Selection

- The kernel dispatcher implements a select-and-ratify thread selection algorithm
  - disp_getbest(). Go find the highest priority runnable thread, and select it for execution
  - disp_ratify(). Commit to the selection. Clear the CPU preempt flags, and make sure another thread of higher priority did not become runnable
    - If one did, place selected thread back on a queue, and try again
- Warm affinity is implemented
  - Put the thread back on the same CPU it executed on last
    - Try to get a warm cache
  - `rechoose_interval` kernel parameter
    - Default is 3 clock ticks

# Thread Preemption

- Two classes of preemption
  - User preemption
    - A higher priority thread became runnable, but it's not a realtime thread
    - Flagged via cpu_runrun in CPU structure
    - Next clock tick, you're outta here
  - Kernel preemption
    - A realtime thread became runnable. Even OS kernel threads will get preempted
    - Poke the CPU (cross-call) and preempt the running thread now
  - Note that threads that use-up thier time quantum are evicted via the preempt mechanism
  - Monitor via “icsw” column in `mpstat`(1)

# Thread Execution

- Run until
  - A preemption occurs
    - Transition from S_ONPROC to S_RUN
    - placed back on a run queue
  - A blocking system call is issued
    - e.g. read(2)
    - Transition from S_ONPROC to S_SLEEP
    - Placed on a sleep queue
  - Done and exit
    - Clean up
  - Interrupt to the CPU you're running on
    - pinned for interrupt thread to run
    - unpinned to continue

# Scheduler Activations

- Introduced in Solaris 2.6 as a preemption control mechanism
- Allows for asking the kernel for a few more ticks
- If a thread is about to be context switched off
  - If an activation has been enabled, the kernel will give the thread a couple extra clock ticks
- Intended to optimize situations where a thread is holding a resouce (e.g. a lock, a latch, etc)
  - It's not desirable to put a thread to sleep that is holding a resource other threads may need to run
  - Let the thread finish, so it can release the resource

```
...
schedctl_init()
schedctl_start()
     get resource
          do work
     release resource
schedctl_stop()
```

# Sleep & Wakeup

- Condition variables used to synchronize thread sleep/wakeup
  - A block condition (waiting for a resource or an event) enters the kernel cv_xxx() functions
  - The condition variable is set, and the thread is placed on a sleep queue
  - Wakeup may be directed to a specific thread, or all threads waiting on the same event or resource
    - One or more threads moved from sleep queue, to run queue

# Dtrace sched provider probes:

- Change-pri – change pri
- Dequeue – exit run q
- Enqueue – enter run q
- Off-cpu – start running
- On-cpu – stop running
- Preempt - preempted
- Remain-cpu
- Schedctl-nopreempt – hint not to no-preempt
- Schedctl-preempt – hint that it is ok to preempt
- Schedctl-yield - hint to give up runnable state
- Sleep – go to sleep
- Surrender – preempt from another cpu
- Tick – tick based accounting
- Wakeup – wakeup from sleep

# Observability and Performance

- Use `prstat`(1) and `ps`(1) to monitor running processes and threads
- Use `mpstat`(1) to monitor CPU utilization, context switch rates and thread migrations
- Use `dispadmin`(1M) to examine and change dispatch table parameters
- User `priocntl`(1) to change scheduling classes and priorities
  - `nice`(1) is obsolete (but there for compatibility)
  - User priorities also set via `priocntl`(1)
  - Must be root to use RT class

# Kernel Synchronization Primitives

- The Solaris kernel is a multithreaded
- Synchronization primitives provide a mechanism for the parallel execution of threads, and synchronized access to data
  - We don't want mutliple threads writing/reading the same field in the same data structure at the same time
    - Or manipulating pointers on a linked list for the same entry at the same time
- Mutual Exclusion (mutex) locks
  - Fastest, and most common primitive used
- Reader/Writer (RW) locks
  - Allow for multiple readers, one writer
  - Useful if long hold times are required

# Kernel Synchronization Primitives

- Mutex Locks
  - Assembly language entry point
    - Very fast – unheld locks acquired in just a few instructions
  - Solaris implements adaptive locks
    - Dynamic adjustment to sleep or spin on held lock based on state of lock holder
      - If holder is running, spin
      - If holder is sleeping, sleep
  - Monitor via `mpstat`(1) smtx column
  - Drill down using `lockstat`(1)
- Reader/Writer locks
  - More complex wakeup mechanism
  - Monitor via `mpstat`(1) srw column

# Lock Statistics – lockstat

```
Adaptive mutex spin: 287 events

Count indv cuml rcnt     spin Lock                   Caller
-------------------------------------------------------------------------------
  112  39%  39% 1.00      301 0x3000014d8e0          sdstrategy+0xac
   50  17%  56% 1.00        2 push_lock              queue_io_request+0x10
   22   8%  64% 1.00        1 push_lock              pageout+0x2c4
   19   7%  71% 1.00      244 0x3000014d8e0          sdintr+0x3c
   15   5%  76% 1.00       22 0x300003a6ee8          vmem_free+0x3c
   10   3%  79% 1.00        6 0x3000014d760          sdstart+0x53c
    8   3%  82% 1.00       12 0x300003a6ee8          vmem_xalloc+0xa4
    5   2%  84% 1.00       93 fhc_bdlist_mutex       fhc_bdlist_lock+0x8
    4   1%  85% 1.00        2 0x3000398f4a8          rdip+0x13c
    4   1%  87% 1.00       11 0x3000014d760          sdintr+0x3c
    4   1%  88% 1.00        1 0x30002c53e28          vn_rele+0x24
    3   1%  89% 1.00        5 0x3000014d760          sdstrategy+0xac
    3   1%  90% 1.00      815 0x3000014d8e0          sdstart+0x588
    3   1%  91% 1.00        1 0x300002061e0          isp_scsi_start+0x1f0
    2   1%  92% 1.00      675 0x3000014d8e0          sdstart+0x53c
    2   1%  93% 1.00       22 0x3000014d8e0          sdstrategy+0x2e0
    2   1%  93% 1.00    12401 pidlock                cv_wait_sig_swap+0x1b0
    2   1%  94% 1.00    20249 pidlock                exit+0x288
    2   1%  95% 1.00    25181 pidlock                lwp_exit+0x354
    1   0%  95% 1.00        8 cpc_mutex+0x50         page_list_add+0xec
    1   0%  95% 1.00     2526 pidlock                waitid+0xa8
    1   0%  96% 1.00      142 pidlock                sigcld_repost+0x48
    1   0%  96% 1.00        2 0x300002b6950          pm_idle_component+0xc
    1   0%  97% 1.00        2 ph_mutex+0x1a8         page_lookup+0x238
```

# lockstat - kernel profiling

```
# lockstat -I sleep 20
Profiling interrupt: 3882 events in 20.011 seconds (194 events/sec)

Count indv cuml rcnt     nsec CPU+PIL                Caller
-------------------------------------------------------------------------------
  509  13%  13% 1.00      119 cpu[1]                 i_ddi_splx+0x1c
  420  11%  24% 1.00      122 cpu[0]                 i_ddi_splx+0x1c
  157   4%  28% 1.00       76 cpu[1]+10              spl6+0x14
  144   4%  32% 1.00       68 cpu[0]                 disp_getwork+0x18
  142   4%  35% 1.00       70 cpu[0]                 disp_getwork
  132   3%  39% 1.00       77 cpu[1]+10              i_ddi_splx
  116   3%  42% 1.00       81 cpu[1]                 spl6
  115   3%  45% 1.00       72 cpu[0]+10              spl6+0x14
  115   3%  48% 1.00       72 cpu[0]+10              i_ddi_splx
  105   3%  50% 1.00       73 cpu[1]                 disp_getwork
   96   2%  53% 1.00       64 cpu[0]                 disp_getwork+0x10
   96   2%  55% 1.00       79 cpu[0]                 spl6
   73   2%  57% 1.00       65 cpu[0]+10              disp_getwork+0x60
   71   2%  59% 1.00       69 cpu[1]                 disp_getwork+0x18
   60   2%  61% 1.00       72 cpu[1]+10              disp_getwork+0x60
   60   2%  62% 1.00       67 cpu[1]                 idle+0x74
   60   2%  64% 1.00       67 cpu[1]+10              disp_getwork+0x4c
```

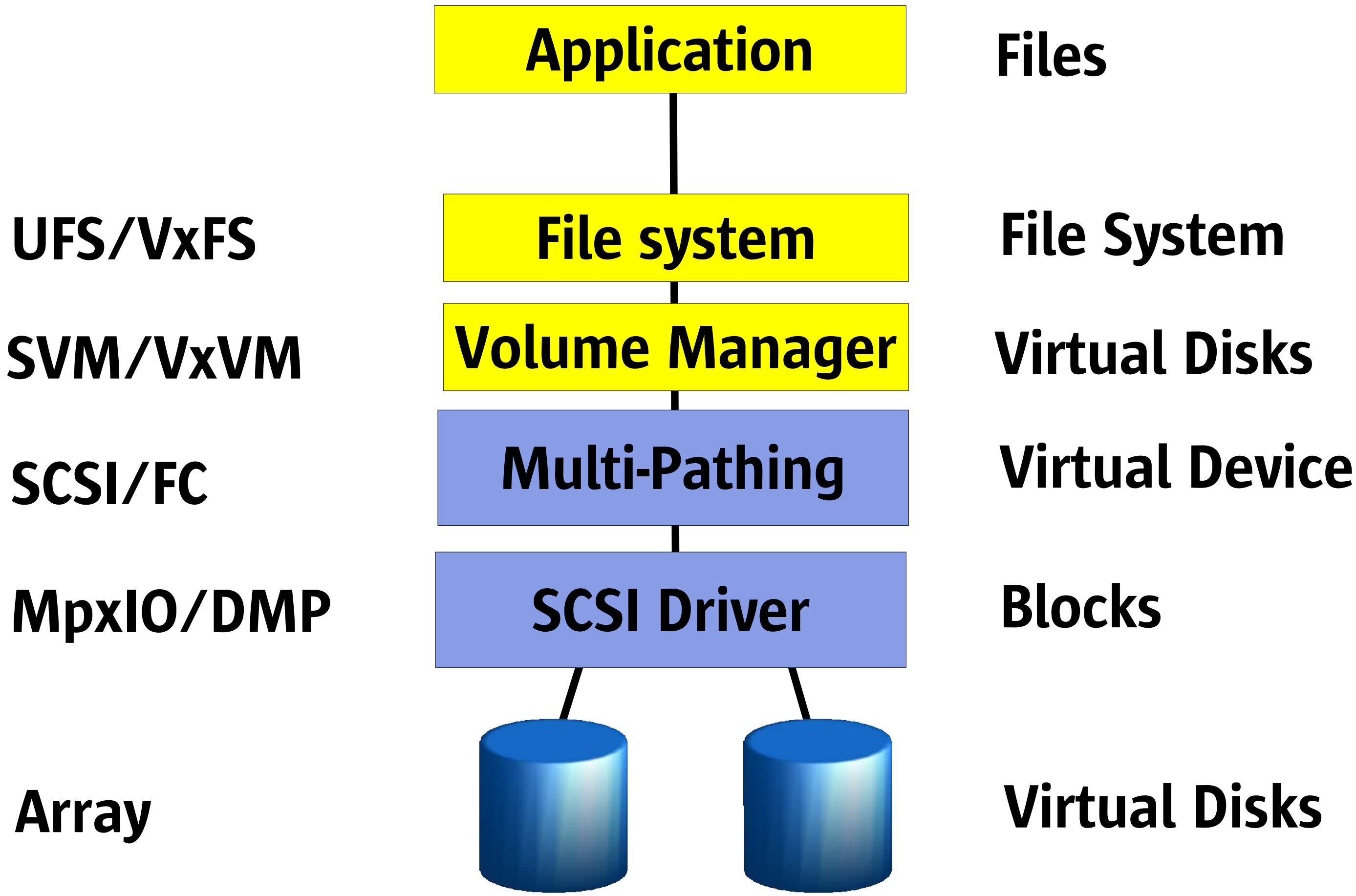
# Turnstiles & Priority Inheritance

- Turnstiles are a specific implementation of sleep queues that provide priority inheritance
- Priority Inheritance (PI) addresses the priority inversion problem
  - Priority inversion is when a higher priority thread is prevented from running because a lower priority thread is holding a lock the higher priority thread needs
    - Blocking chains can form when “mid” priority threads get in the mix
- Priority inheritance
  - If a resource is held, ensure all the threads in the blocking chain are at the requesting thread's priority, or better
    - All lower priority threads inherit the priority of the

# Process, Thread, Scheduling Lab

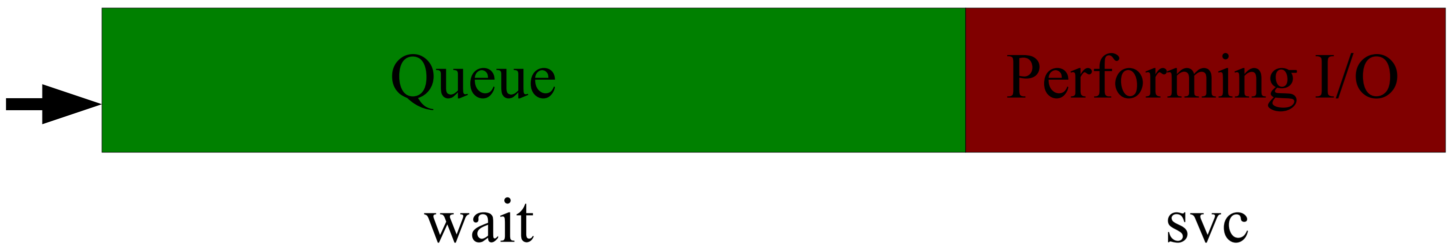
# Disk I/O Performance

# The Solaris File System/IO Stack



# Solaris iostat

```
# iostat -xnz
                              extended device statistics
    r/s    w/s   kr/s    kw/s wait actv wsvc_t asvc_t  %w  %b device
  687.8    0.0 38015.3    0.0  0.0  1.9    0.0    2.7   0 100 c0d0
```




- Wait: number of threads queued for I/O
- Actv: number of threads performing I/O
- wsvc_t: Average time spend waiting on queue
- asvc_t: Average time performing I/O
- %w: Only useful if one thread is running on the entire machine – time spent waiting for I/O
- %b: Device utilizing – only useful if device can do just 1 I/O at a time (invalid for arrays etc...)

# Lab: 1 thread I/O example

```
sol8$ cd labs/disks
sol8$ ./1thread
 1079: 0.007: Random Read Version 1.8 05/02/17 IO personality successfully loaded
 1079: 0.008: Creating/pre-allocating files
 1079: 0.238: Waiting for preallocation threads to complete...
 1079: 0.238: Re-using file /filebench/bigfile0
 1079: 0.347: Starting 1 rand-read instances
 1080: 1.353: Starting 1 rand-thread threads
 1079: 4.363: Running for 600 seconds...
sol8$ iostat -xncz 5
      cpu
 us sy wt id
 22  3  0 75
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
   62.7    0.3  501.4    2.7  0.0  0.9    0.0   14.1   0  89 c1d0
```

# Lab: 64 thread I/O example

```
sol8$ cd labs/disks
sol8$ ./64thread
 1089: 0.095: Random Read Version 1.8 05/02/17 IO personality successfully loaded
 1089: 0.096: Creating/pre-allocating files
 1089: 0.279: Waiting for preallocation threads to complete...
 1089: 0.279: Re-using file /filebench/bigfile0
 1089: 0.385: Starting 1 rand-read instances
 1090: 1.389: Starting 64 rand-thread threads
 1089: 4.399: Running for 600 seconds...

sol8$ iostat -xncz 5
      cpu
 us sy wt id
 15  1  0 83
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
   71.0    0.3  568.0   17.3 61.8  2.0  866.5   28.0 100 100 c1d0
```

# Solaris iostat: New opts. since Solaris 8

- New Formatting flags -C, -l, -m, -r, -s, -z, -T
  - -C: report disk statistics by controller
  - -l n: Limit the number of disks to n
  - -m: Display mount points (most useful with -p)
  - -r: Display data n comma separated format
  - -s: Suppress state change messages
  - -z: Suppress entries with all zero values
  - -T d|u Display a timestamp in date (d) or unix time_t (u)

# Examining Physical IO by file with dtrace

```
#pragma D option quiet

BEGIN
{
        printf("%10s %58s %2s %8s\n", "DEVICE", "FILE", "RW", "Size");
}

io:::start
{
        printf("%10s %58s %2s %8d\n", args[1]->dev_statname,
            args[2]->fi_pathname, args[0]->b_flags & B_READ ? "R" : "W",
            args[0]->b_bcount);
}

# dtrace -s ./iotrace

    DEVICE                                                   FILE RW     SIZE
     cmdk0                             /export/home/rmc/.sh_history  W     4096
     cmdk0                               /opt/Acrobat4/bin/acroread  R     8192
     cmdk0                               /opt/Acrobat4/bin/acroread  R     1024
     cmdk0                               /var/tmp/wscon-:0.0-gLaW9a  W     3072
     cmdk0                         /opt/Acrobat4/Reader/AcroVersion  R     1024
     cmdk0           /opt/Acrobat4/Reader/intelsolaris/bin/acroread  R     8192
     cmdk0           /opt/Acrobat4/Reader/intelsolaris/bin/acroread  R     8192
     cmdk0           /opt/Acrobat4/Reader/intelsolaris/bin/acroread  R     4096
     cmdk0           /opt/Acrobat4/Reader/intelsolaris/bin/acroread  R     8192
     cmdk0           /opt/Acrobat4/Reader/intelsolaris/bin/acroread  R     8192
```

# Lab: Physical Trace Example

```
sol8$ cd labs/disks
sol8$ ./64thread
 1089: 0.095: Random Read Version 1.8 05/02/17 IO personality successfully loaded
 1089: 0.096: Creating/pre-allocating files
 1089: 0.279: Waiting for preallocation threads to complete...
 1089: 0.279: Re-using file /filebench/bigfile0
 1089: 0.385: Starting 1 rand-read instances
 1090: 1.389: Starting 64 rand-thread threads
 1089: 4.399: Running for 600 seconds...

sol8$ iotrace.d
    DEVICE                                                    FILE RW      Size
     cmdk0                                     /filebench/bigfile0  R      8192
     cmdk0                                     /filebench/bigfile0  R      8192
     cmdk0                                     /filebench/bigfile0  R      8192
     cmdk0                                     /filebench/bigfile0  R      8192
     cmdk0                                     /filebench/bigfile0  R      8192
     cmdk0                                     /filebench/bigfile0  R      8192
     cmdk0                                     /filebench/bigfile0  R      8192
     cmdk0                                     /filebench/bigfile0  R      8192
     cmdk0                                     /filebench/bigfile0  R      8192
     cmdk0                                     /filebench/bigfile0  R      8192
     cmdk0                                     /filebench/bigfile0  R      8192
```

# An Introduction to File System Performance

# Filesystem performance

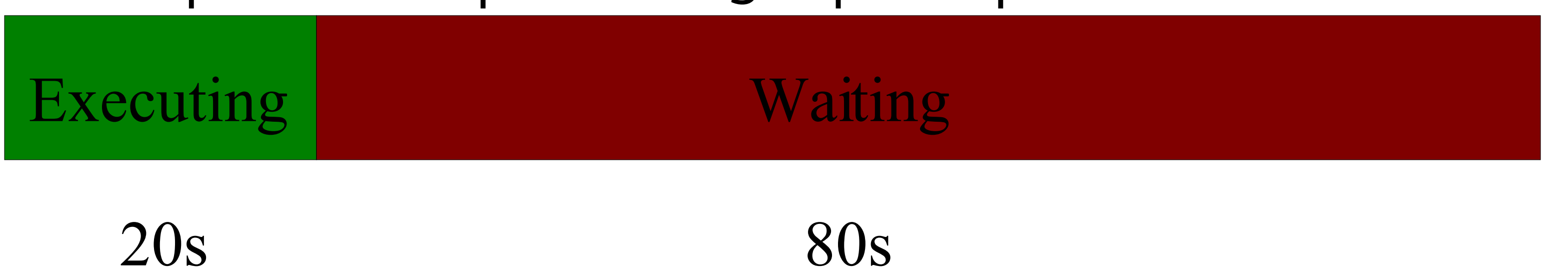
- Attribution
  - How much is my application being slowed by I/O?
  - i.e. How much faster would my app run if I optimized I/O?
- Accountability
  - What is causing I/O device utilization?
  - i.e. What user is causing this disk to be hot?
- Tuning/Optimizing
  - Tuning for sequential, random I/O and/or meta-data intensive applications

# Solaris FS Perf Tools

- iostat: raw disk statistics
- sar -b: meta-data buffer cachestat
- vmstat -s: monitor dnlc
- Filebench: emulate and measure various FS workloads
- DTrace: trace physical I/O
- DTrace: top for files – logical and physical per file
- DTrace: top for fs – logical and physical per filesystem

# Simple performance model

- Single threaded processes are simpler to estimate
  - Calculate elapsed vs. waiting for I/O time, express as a percentage
  - i.e. My app spent 80% of it's execution time waiting for I/O
  - Inverse is potential speed up – e.g. 80% of time waiting equates to a potential 5x speedup




- The key is to estimate the time spent waiting

# Estimating wait time

- Elapsed vs. cpu seconds
  - Time <cmd>, estimate wait as real – user - sys
- Etruss
  - Uses microstates to estimate I/O as wait time
  - http://www.solarisinternals.com
- Measure explicitly with dtrace
  - Measure and total I/O wait per thread

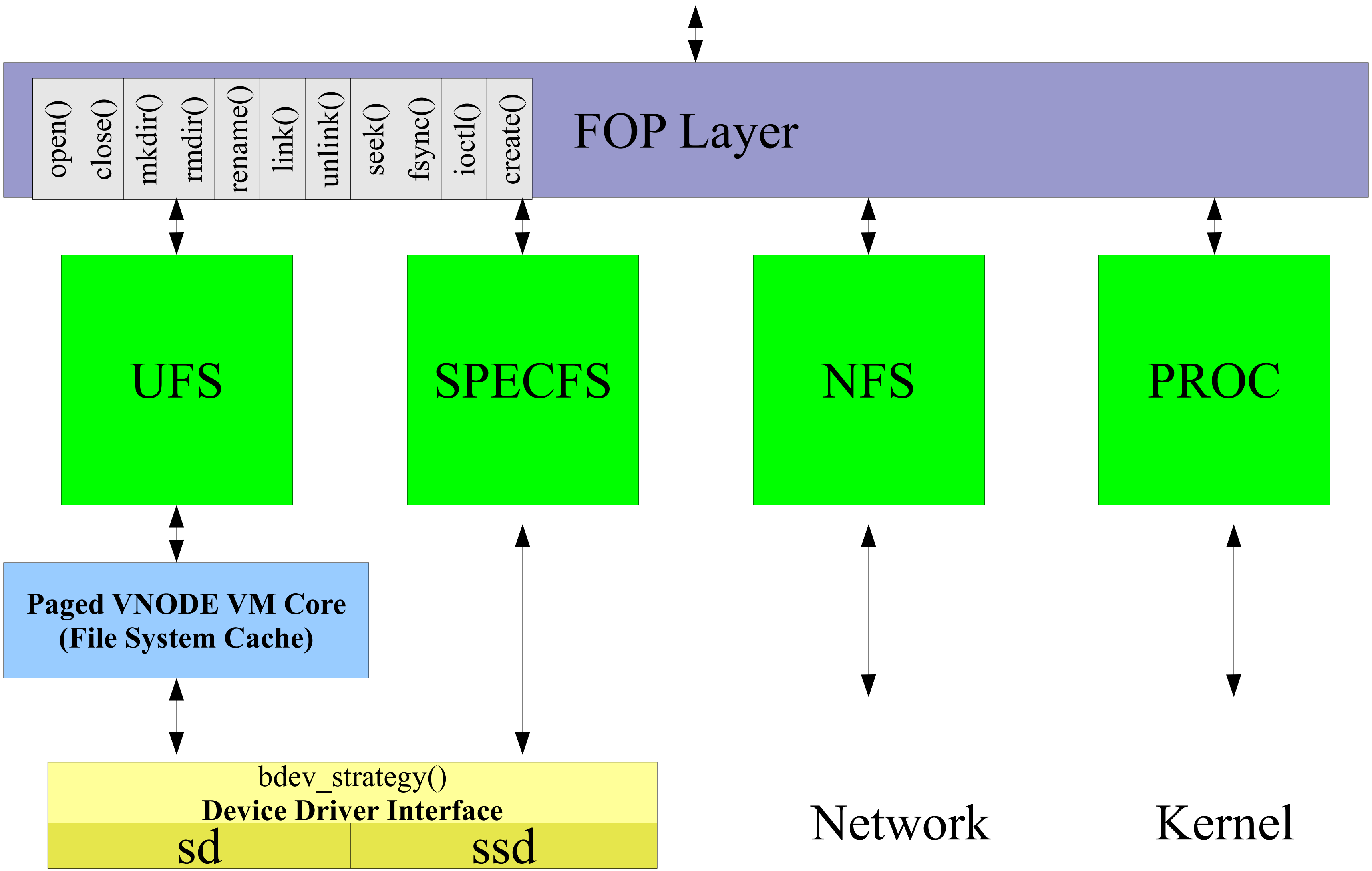
# Examining IO wait with dtrace

- Measuring on-cpu vs io-wait time:

```
sol10$ ./iowait.d 639
^C
Time breakdown (milliseconds):
 <on cpu>                                                          2478
 <I/O wait>                                                        6326

I/O wait breakdown (milliseconds):
 file1                                                              236
 file2                                                              241
 file4                                                              244
 file3                                                              264
 file5                                                              277
 file7                                                              330
 .
 .
 .
```

# Using Dtrace to examine File System Performance

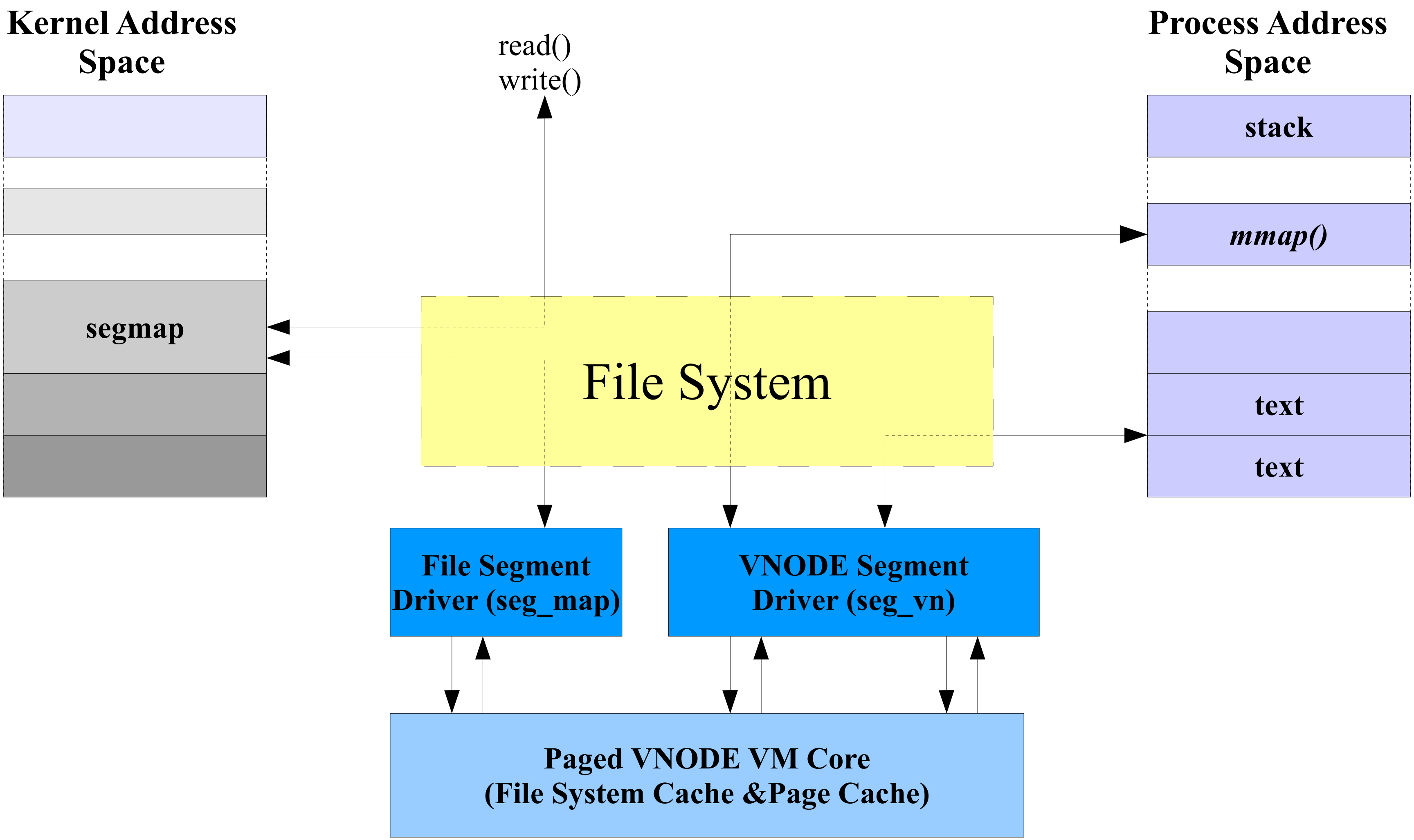
# File System Architecture



# File system I/O via Virtual Memory

- File system I/O is performed by the VM system
  - Reads are performed by page-in
  - Write are performed by page-out
- Practical Implications
  - Virtual memory caches files, cache is dynamic
  - Minimum I/O size is the page size
  - Read/modify/write may occur on sub page-size writes
- Memory Allocation Policy:
  - File system cache is lower priority than app, kernel etc
  - File system cache grows when there is free memory available
  - File system cache shrinks when there is demand elsewhere.

# File System I/O



# File System Reads: A UFS Read

- Application calls read()
- Read system call calls fop_read()
- FOP layer redirector calls underlying filesystem
- FOP jumps into ufs_read
- UFS locates a mapping for the corresponding pages in the file system page cache using vnode/offset
- UFS asks segmap for a mapping to the pages
- If the page exists in the fs, data is copied to App.
  - We're done.
- If the page doesn't exist, a Major fault occurs
  - VM system invokes ufs_getpage()
  - UFS schedules a page size I/O for the page
  - When I/O is complete, data is copied to App

# Vmstat -p

swap = free and unreserved swap in KBytes
free = free memory measured in pages

re = kilobytes reclaimed from cache/free list
mf = minor faults - the page was in memory but was not mapped
fr = kilobytes that have been destroyed or freed
de = kilobytes freed after writes
sr = kilobytes scanned / second

executable pages: kilobytes in - out - freed

anonymous pages: kilobytes in - out – freed

file system pages: kilobytes in - out - freed

```
# vmstat -p 5 5
      memory              page                executable      anonymous       filesystem
   swap    free    re  mf  fr  de  sr   epi  epo  epf   api  apo  apf   fpi  fpo  fpf
 ...
 46715224 891296 24   350 0   0   0     0    0    0     4    0    0    27    0    0
 46304792 897312 151 761 25   0   0    17    0    0     1    0    0   280   25   25
 45886168 899808 118 339 1    0   0     3    0    0     1    0    0   641    1    1
 46723376 899440 29  197 0    0   0     0    0    0    40    0    0    60    0    0
```

# Lab: Observing the File System I/O Path

```
Sol10# cd labs/fs_paging
sol10# ./fsread
 2055: 0.004: Random Read Version 1.8 05/02/17 IO personality successfully loaded
 2055: 0.004: Creating/pre-allocating files
 2055: 0.008: Waiting for preallocation threads to complete...
 2055: 28.949: Pre-allocated file /filebench/bigfile0
 2055: 30.417: Starting 1 rand-read instances
 2056: 31.425: Starting 1 rand-thread threads
 2055: 34.435: Running for 600 seconds...

sol10# vmstat -p 3
     memory           page          executable      anonymous      filesystem
   swap  free  re  mf  fr  de  sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf
 1057528 523080 22 105  0   0   8    5    0    0    0    0    0   63    0    0
 776904 197472  0  12   0   0   0    0    0    0    0    0    0  559    0    0
 776904 195752  0   0   0   0   0    0    0    0    0    0    0  555    0    0
 776904 194100  0   0   0   0   0    0    0    0    0    0    0  573    0    0

sol10# ./pagingflow.d
  0  => pread64                                            0
  0    | pageio_setup:pgin                                40
  0    | pageio_setup:pgpgin                              42
  0    | pageio_setup:maj_fault                           43
  0    | pageio_setup:fspgin                              45
  0    | bdev_strategy:start                              52
  0    | biodone:done                                  11599
  0  <= pread64                                        11626
```

# Lab: Observing File System I/O

```
Sol10# cd labs/fs_paging
sol10# ./fsread
 2055: 0.004: Random Read Version 1.8 05/02/17 IO personality successfully loaded
 2055: 0.004: Creating/pre-allocating files
 2055: 0.008: Waiting for preallocation threads to complete...
 2055: 28.949: Pre-allocated file /filebench/bigfile0
 2055: 30.417: Starting 1 rand-read instances
 2056: 31.425: Starting 1 rand-thread threads
 2055: 34.435: Running for 600 seconds...

sol10# ./fspaging.d
Event           Device                                                         Path RW     Size
get-page                                                         /filebench/bigfile0       8192
getpage-io       cmdk0                                           /filebench/bigfile0  R    8192
get-page                                                         /filebench/bigfile0       8192
getpage-io       cmdk0                                           /filebench/bigfile0  R    8192
get-page                                                         /filebench/bigfile0       8192
getpage-io       cmdk0                                           /filebench/bigfile0  R    8192
get-page                                                         /filebench/bigfile0       8192
```

# Lab: Observing File System I/O: Sync Writes

```
Sol10# cd labs/fs_paging
sol10# ./fswritesync
 2276: 0.008: Random Write Version 1.8 05/02/17 IO personality successfully loaded
 2276: 0.009: Creating/pre-allocating files
 2276: 0.464: Waiting for preallocation threads to complete...
 2276: 0.464: Re-using file /filebench/bigfile0
 2276: 0.738: Starting 1 rand-write instances
 2277: 1.742: Starting 1 rand-thread threads
 2276: 4.743: Running for 600 seconds...

sol10# ./fspaging.d
Event           Device                                                 Path RW      Size   Offset
put-page                                                 /filebench/bigfile0        8192
putpage-io        cmdk0                                  /filebench/bigfile0  W     8192 18702224
other-io          cmdk0                                               <none>  W      512    69219
put-page                                                 /filebench/bigfile0        8192
putpage-io        cmdk0                                  /filebench/bigfile0  W     8192 11562912
other-io          cmdk0                                               <none>  W      512    69220
put-page                                                 /filebench/bigfile0        8192
putpage-io        cmdk0                                  /filebench/bigfile0  W     8192 10847040
other-io          cmdk0                                               <none>  W      512    69221
put-page                                                 /filebench/bigfile0        8192
putpage-io        cmdk0                                  /filebench/bigfile0  W     8192 22170752
other-io          cmdk0                                               <none>  W      512    69222
put-page                                                 /filebench/bigfile0        8192
putpage-io        cmdk0                                  /filebench/bigfile0  W     8192 25189616
other-io          cmdk0                                               <none>  W      512    69223
put-page                                                 /filebench/bigfile0        8192
```

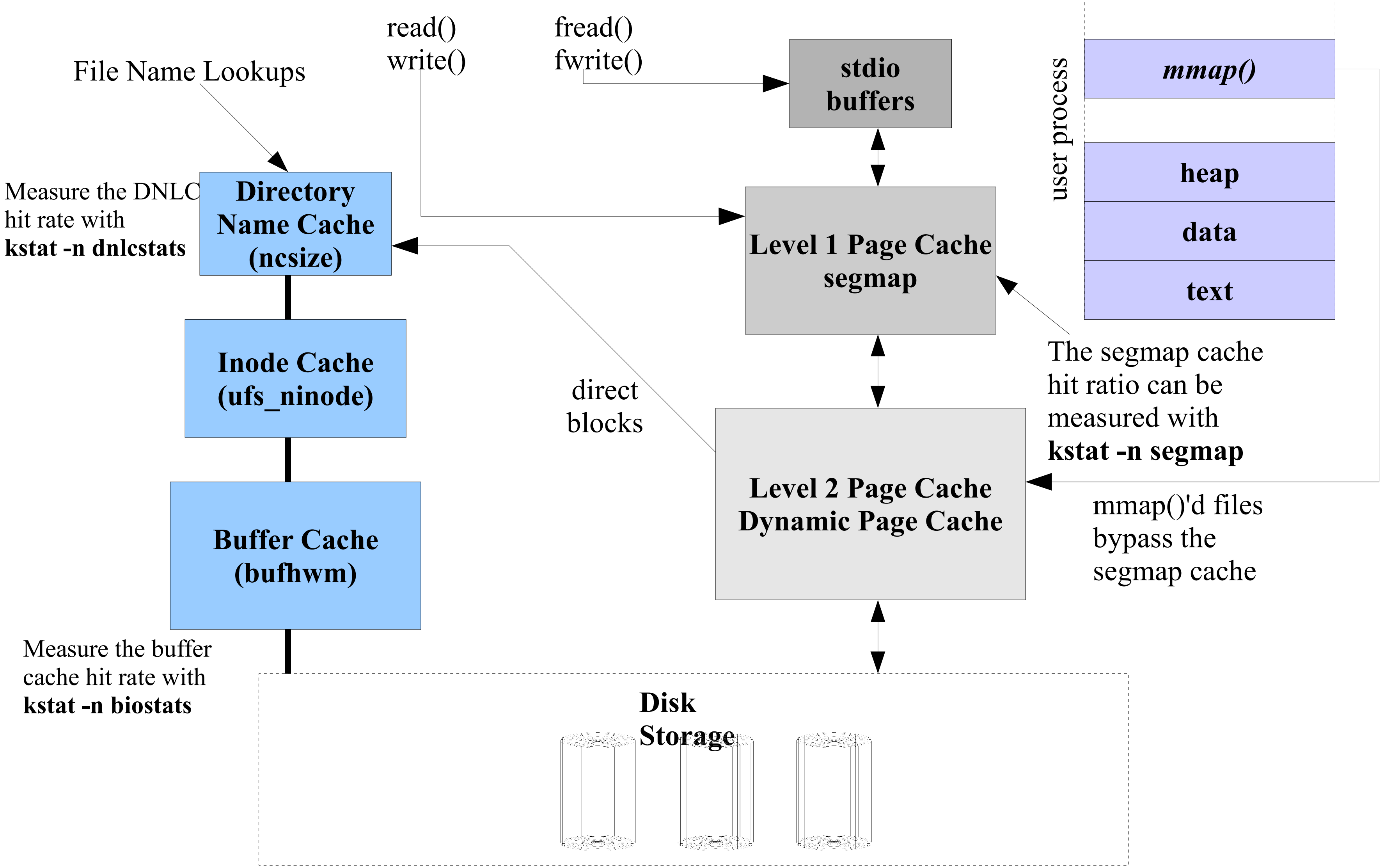
# Memory Mapped I/O

- Application maps file into process with mmap()
- Application references memory mapping
- If the page exists in the cache, we're done.
- If the page doesn't exist, a Major fault occurs
  - VM system invokes ufs_getpage()
  - UFS schedules a page size I/O for the page
  - When I/O is complete, data is copied to App.

# The big caches:

- File system/page cache
  - Holds the “data” of the files
- Buffer Cache
  - Holds the meta-data of the file system: direct/indirect blocks, inodes etc...
- Directory Name Cache
  - Caches mappings of filename->vnode from recent lookups
  - Prevents excessive re-reading of directory from disk
- File system specific: Inode cache
  - Caches inode meta-data in memory
  - Holds owner, mtimes etc

# File System Caching

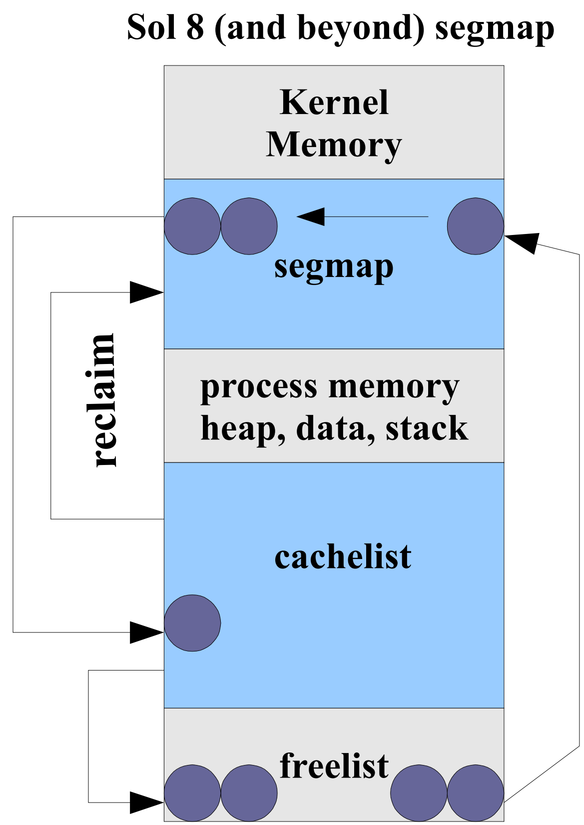


# Optimizing Random I/O File System Performance

# Random I/O

- Attempt to cache as much as possible
  - The best I/O is the one you don't have to do
  - Eliminate physical I/O
  - Add more RAM to expand caches
  - Cache at the highest level
    - Cache in app if we can
    - In Oracle if possible
- Match common I/O size to FS block size
  - e.g. Write 2k on 8k FS = Read 8k, Write 8k

# The Solaris 8 File System Cache



# Tuning segmap

- By default, segmap is sized at 12% of physical memory
  - Effectively sets the minimum amount of file system cache on the system by caching in segmap over and above the dynamically sized cachelist
- On Solaris 8/9
  - If the system memory is used primarily as a cache, cross calls (mpstat xcall) can be reduced by increasing the size of segmap via the system parameter segmap_percent (12 by default)
  - segmap_percent = 100 is like Solaris 7 without priority paging, and will cause a paging storm
  - Must keep segmap_percent at a reasonable value to prevent paging pressure on applications e.g. 50%

# Tuning segmap_percent

- There are kstat statistics for segmap hit rates
  - Estimate hit rate as (get_reclaim+get_use) / getmap

```
# kstat -n segmap
module: unix                              instance: 0
name:   segmap                            class:    vm

        crtime                            17.299814595
        fault                             17361
        faulta                            0
        free                              0
        free_dirty                        0
        free_notfree                      0
        get_nofree                        0
        get_reclaim                       67404
        get_reuse                         0
        get_unused                        0
        get_use                           83
        getmap                            71177
        pagecreate                        757
        rel_abort                         0
        rel_async                         3073
        rel_dontneed                      3072
        rel_free                          616
        rel_write                         2904
        release                           67658
        snaptime                          583596.778903492
```

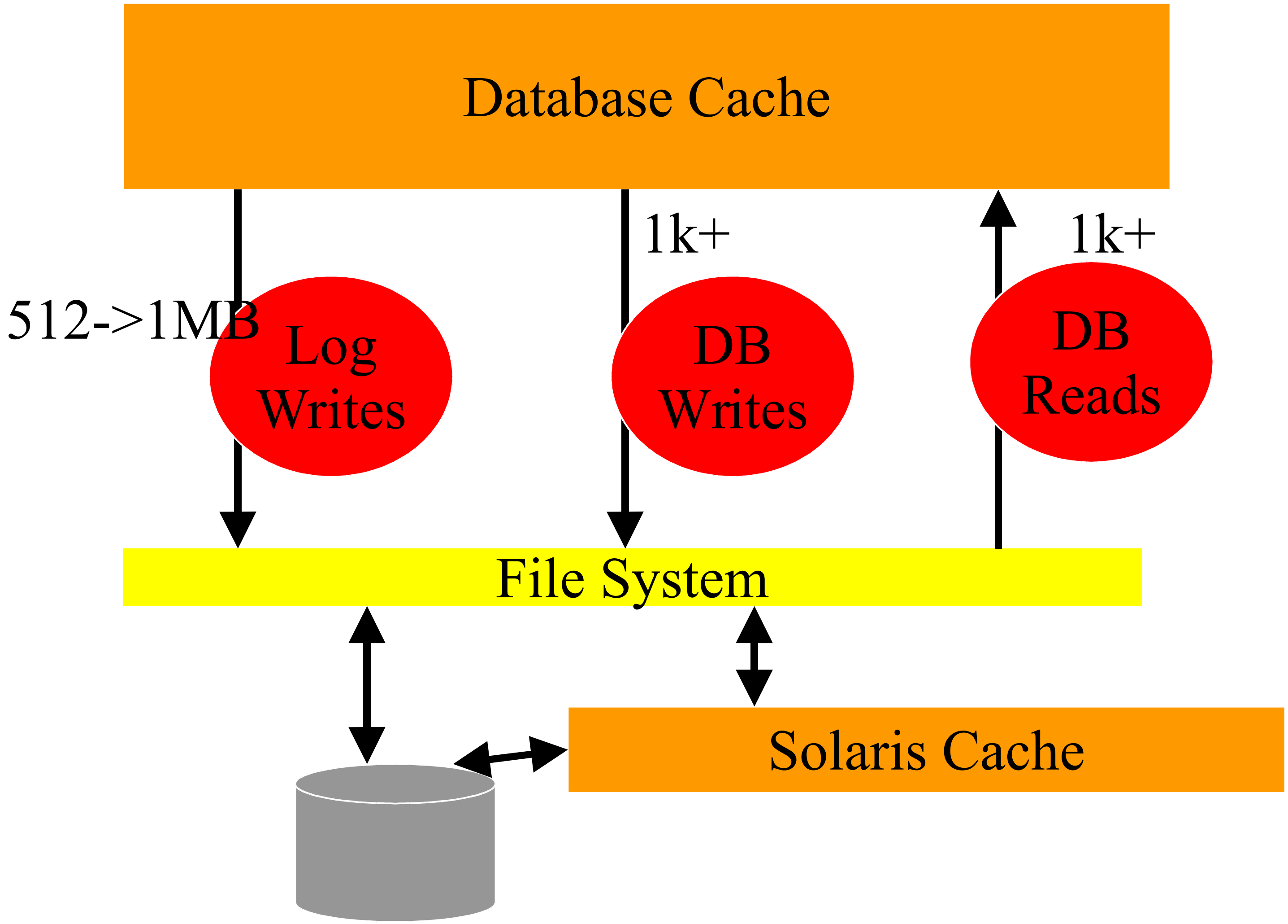
# UFS Access times

- Access times are updated when file is accessed or modified
  - e.g. A web server reading files will storm the disk with atime writes!
- Options allow atimes to be eliminated or deferred
  - dfratime: defer atime write until write
  - noatime: do not update access times, great for web servers and databases

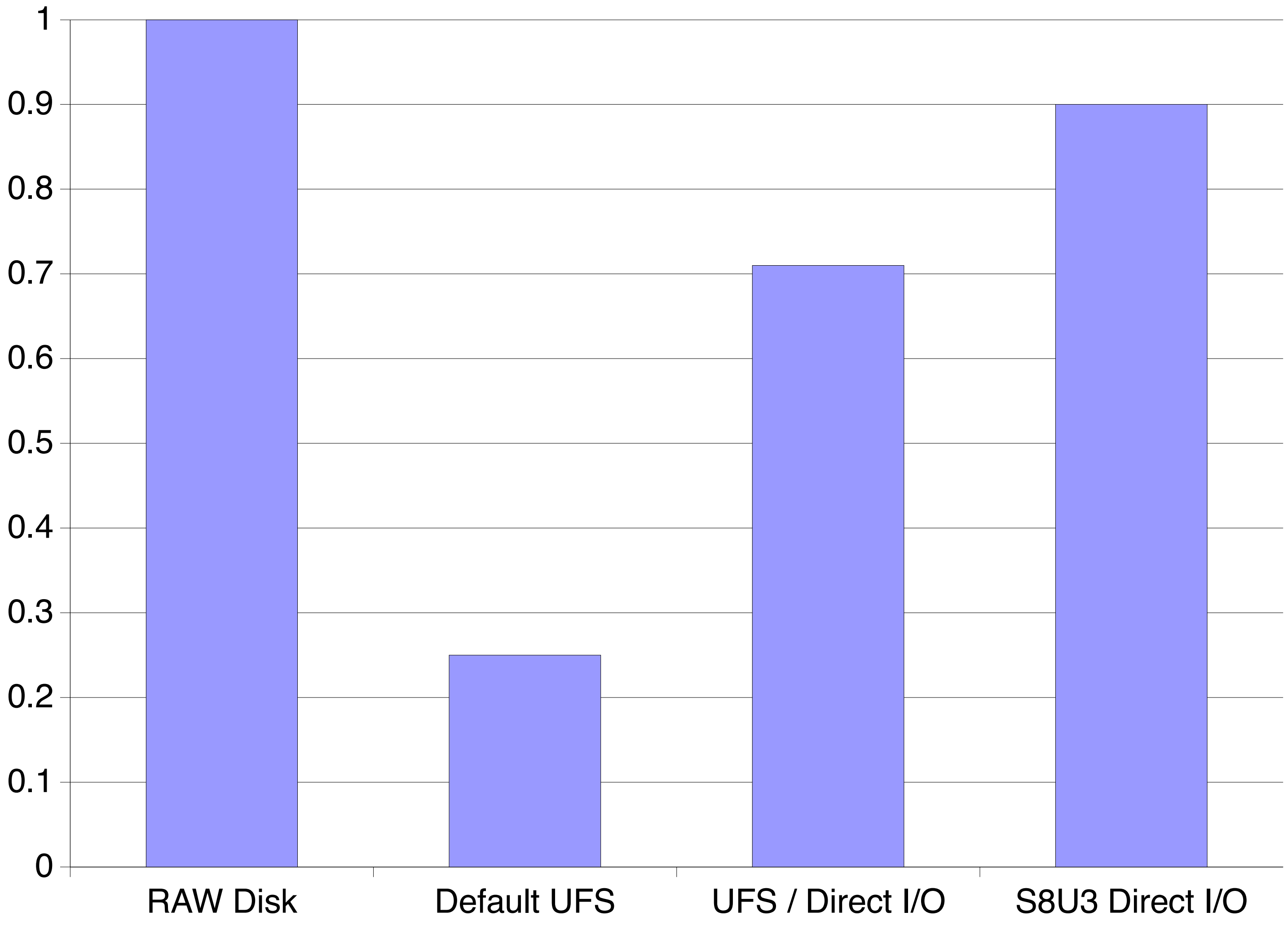
# Asynchronous I/O

- An API for single-threaded process to launch multiple outstanding I/Os
  - Multi-threaded programs could just just multiple threads
  - Oracle databases uses this extensively
  - See aio_read(), aio_write() etc...
- Slightly different varients for RAW disk vs file system
  - UFS, NFS etc: libaio creates lwp's to handle requests via standard pread/pwrite system calls
  - RAW disk: I/Os are passed into kernel via kaio(), and then managed via task queues in the kernel
    - Moderately faster than user-level LWP emulation

# Putting it all together: Database File I/O



# UFS is now Enhanced for Databases:



# Key UFS Features

- Direct I/O
  - Solaris 2.6+
- Logging
  - Solaris 7+
- Async I/O
  - Oracle 7.x, -> 8.1.5 - Yes
  - 8.1.7, 9i - New Option
- Concurrent Write Direct I/O
  - Solaris 8, 2/01

# Database big rules...

- Always put re-do logs on Direct I/O
- Cache as much as possible in the SGA
- Use 64-Bit RDBMS (Oracle 8.1.7+)
- Always use Asynch I/O
- Use Solaris 8 Concurrent Direct I/O
- Place as many tables as possible on Direct I/O, assuming SGA sized correct
- Place write-intensive tables on Direct I/O

# Optimizing Sequential I/O File System Performance

# Sequential I/O

- Disk performance fundamentals
  - Disk seek latency will dominate for random I/O
    - ~5ms per seek
  - A typical disk will do ~200 I/Os per second random I/O
  - 200 x 8k = 1.6MB/s
  - Seekless transfers are typically capable of ~50MB/s
    - Requires I/O sizes of 64k+
- Optimizing for sequential I/O
  - Maximizing I/O sizes
  - Eliminating seeks
  - Minimizing OS copies

# Sequential I/O – Looking at disks via iostat

- Use iostat to determine average I/O size
  - I/O size = kbytes/s divided by I/Os per second

```
# iostat -xnz
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
  687.8    0.0 38015.3    0.0  0.0  1.9    0.0    2.7   0 100 c0d0
```

- What is the I/O size in our example?
  - 38015 / 687 = 56k
  - Too small for best sequential performance!

# Sequential I/O – Maximizing I/O Sizes

- Application
  - Ensure application is issuing large writes
    - 1MB is a good starting point
  - truss or dtrace app
- File System
  - Ensure file system groups I/Os and does read ahead
  - A well tuned fs will group small app I/Os into large Physical I/Os
  - e.g. UFS cluster size
- IO Framework
  - Ensure large I/O's can pass though
  - System param *maxphys* set largest I/O size
- Volume Manager
  - md_maxphys for SVM, or equiv for Veritas
- SCSI or ATA drivers often set defaults to upper layers

# Sequential on UFS

- Sequential mode is detected by 2 adjacent operations
  - e.g read 8k, read8k
- UFS uses “clusters” to group reads/write
  - UFS “maxcontig” param, units are 8k
  - Maxcontig becomes the I/O size for sequential
  - Cluster size defaults to 1MB on Sun FCAL
    - 56k on x86, 128k on SCSI
    - Auto-detected from SCSI driver's default
    - Set by default at newfs time (can be overridden)
  - e.g. Set cluster to 1MB for optimal sequential perf...
  - Check size with “mkfs -m”, set with “tunefs -a”

```
# mkfs -m /dev/dsk/c0d0s0
mkfs -F ufs -o nsect=63,ntrack=32,bsize=8192,fragsize=1024,cgsize=49,free=1,rps=60,
nbpi=8143,opt=t,apc=0,gap=0,nrpos=8,maxcontig=7,mtb=n /dev/dsk/c0d0s0 14680512

# tunefs -a 128 /dev/rdsk/...
```

# Examining UFS Block Layout with filestat

```
# filestat /home/bigfile
Inodes per cyl group: 64
Inodes per block: 64
Cylinder Group no: 0
Cylinder Group blk: 64
File System Block Size: 8192
Device block size: 512
Number of device blocks: 204928

Start Block End Block   Length (Device Blocks)
----------- ----------- ----------------------
66272 ->    66463       192
66480 ->    99247       32768
1155904 ->  1188671     32768
1277392 ->  1310159     32768
1387552 ->  1420319     32768
1497712 ->  1530479     32768
1607872 ->  1640639     32768
1718016 ->  1725999     7984
1155872 ->  1155887     16
Number of extents:      9

Average extent size: 22769 Blocks


Note: The filestat command can be found on http://www.solarisinternals.com
```

# Sequential on UFS

- Cluster Read
  - When sequential detected, read ahead entire cluster
  - Subsequent reads will hit in cache
  - Sequential blocks will not pollute cache by default
    - i.e. Sequential reads will be freed sooner
    - Sequential reads go to head of cachelist by default
    - Set system param *cache_read_ahead*=1 if all reads should be cached
- Cluster Write
  - When sequential detected, writes are deferred until cluster is full

# UFS write throttle

- UFS will block when there are too much pending dirty pages
  - Application writes by default go to memory, and are written asynchronously
  - Throttle blocks to prevent filling memory with async. Writes
- Solaris 8 Defaults
  - Block when 384k of unwritten cache
    - Set *ufs_HW*=<bytes>
  - Resume when 256k of unwritten cache
    - Set *ufs_LW*=<bytes>
- Solaris 9+ Defaults
  - Block when >16MB of unwritten cache
  - Resume when <8MB of unwritten cache

# Update on Recent Solaris UFS + SVM Developments

# Other items for Solaris UFS

- Solaris 8 Update 2/01
  - File system Snapshots
  - Enhanced logging w/ Direct I/O
  - Concurrent Direct I/O
  - 90% of RAW disk performance
  - Enhanced Directory Lookup
  - File create times in large directories significantly improved
  - Creating file systems
    - Faster newfs (1TB was ~20 hours)
- Solaris 9
  - Scalable Logging (for File Servers) 12/02
    - Postmark White paper
  - >1TB Filesystems (16TB) 8/03

# Solaris Volume Manager

- Solaris 9
  - Integration with live upgrade 5/03
  - >1TB Volumes 5/03
  - >1TB Devices/EFI Support 11/03
  - Dynamic Reconfiguration Support 11/03
- Future
  - Cluster Ready Volume Manager
  - Disk Set Migration: Import/Export
  - Volume Creation Service

# Volume Manager/FS Features

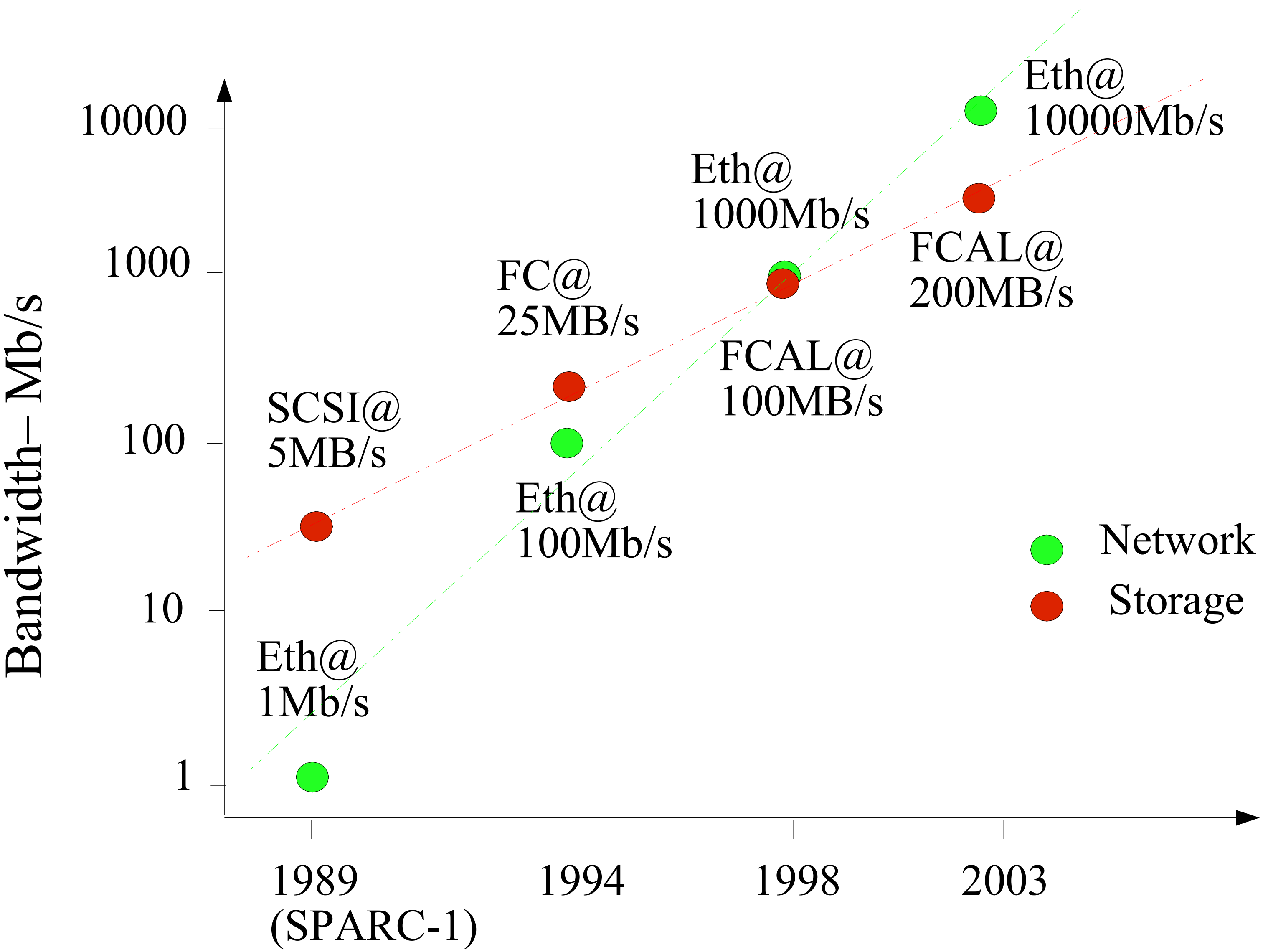
| Feature | Solaris | VxVM | VxFS |
|---|---|---|---|
| Online Unmount | Yes | | |
| Raid 0,1,5,1+0 | Yes | Yes | |
| Logging/No FSCK | Sol 7 | | Yes |
| Soft Partitions | Sol 8 | Yes | |
| Device Path Independence | Sol 8 | Yes | |
| Database Performance | Sol 8 2/02 | | QuickIO |
| Integration with Install | Sol 9 | | |
| Multi-Pathing | Sol 9 | Yes/DMP | |
| Grow Support | Sol 9 | Yes | Yes |
| Fast Boot | Sol 9 | | |
| Integration with LU | Sol 9 5/03 | | |
| >1TB Volumes | Sol 9 5/03 | 3.5 | |
| >1TB Filesystems | Sol 9 8/03 | | 3.5/VxVM |
| >1TB Devices/EFI Support | Sol 9 8/03 | | |
| Dynamic Reconfiguration Integration | Sol 9 8/03 | | |
| Cluster Ready Volume Manager | Future | VxCVM | |
| Disk Group Migration: Import/Export | Future | Yes | |

# Network Attached Storage

# Network File Systems – Some background

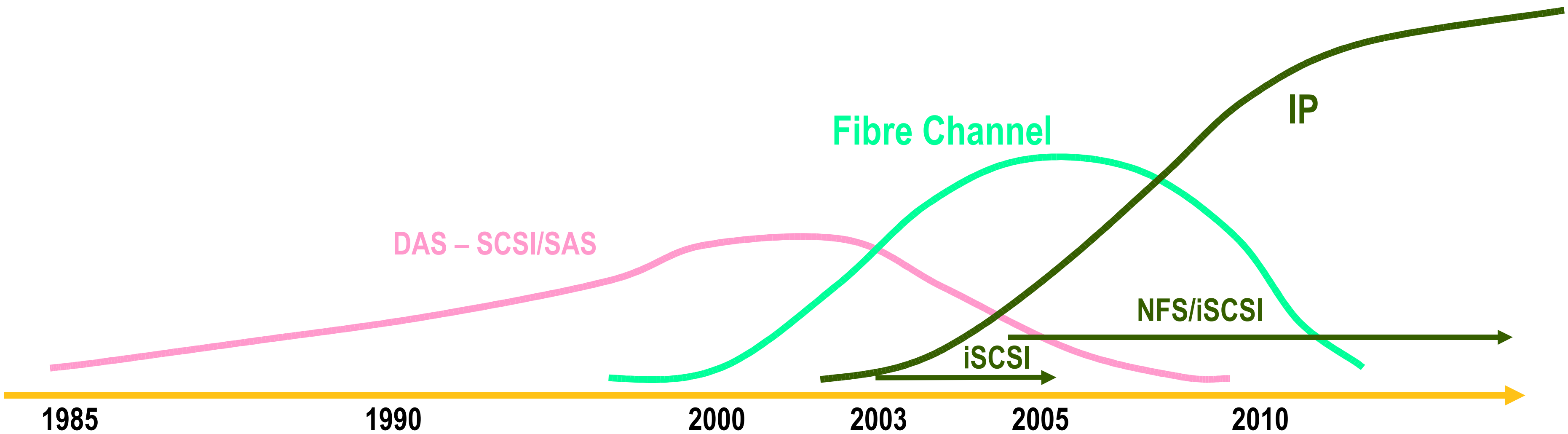
- Heritage
  - Added to SunOS in 1985 as a network storage interconnect protocol
  - Superceeded “netdisk” - blocks over the network
  - Superior administration model
    - Full file system semantics on the client
    - Full separation of client from administration
    - All file system administration performed on the server
- So, why SANs?
  - Networking performance wasn't sufficient
  - Fibrechannel introduced to provide fast transport

# However, now network >= storage/SCSI



# Back to the future: Storage over IP

- Networks are fast re-emerging as commodity storage interconnects
- Key technologies
  - iSCSI – blocks over IP
  - Datacenter grade NFS – full file system semantics



# IP Storage in Solaris

- iSCSI client in '05
  - Interim step to allow move to IP storage
  - Replace FC SAN hardware with IP switches
  - Bridge IP-iSCSI networks into SAN with iSCSI/FC bridge
  - Rising fast from the low end
    - iSCSI over GBE allows low connection cost to SAN
- Datacenter grade NFS
  - NFS is displacing block-based interconnects
  - NFS4 over GBE currently @ 105MB/s (wirespeed)
  - OLTP Database over NFS3/4 on par with local FS
  - Rapid rise in use of NFS for data-center apps
  - NFSV4 adds security

# NFS client: Sequential Performance

- Optimize networking
  - Use Gigabit networks (10GBE is avail. Now too...)
  - Use Jumbo-frames where possible
  - Dedicate a network to storage
    - Just like we do with a SAN...
- Maximize NFS block size
  - Solaris 8 has a max of 32k
  - Solaris 9 allows up to 1MB!
    - Solaris 9+ NFS Server required
    - Set *nfs3_bsize* and *nfs3_max_transfer_size* system parameter on client and server
  - Futher tuning down of blocksize can be done via mount options: rsize, wsize

# NFS client: Sequential Performance

- Myth: NFS is only good for 5MB/s...
  - Sequential performance is making great strides
- Performance Rules of thumb
  - NFS3 on 1GHz SPARC, Solaris 8
    - 30MB/s
  - NFS3 on 1GHz SPARC, Solaris 9
    - 55MB/s
  - NFS4 on 1GHz SPARC, Solaris 10
    - 90MB/s
  - NFS4 on 2Ghz Opteron, Solaris 10
    - 105MB/s (wirespeed!)

# NFS Client: Optimizing for sequential

- Tunings for Casinni (CE)
  - Place “interrupts=1;” in /platform/sun4u/kernel/drv/ce.conf
  - Use Cassini helper thread all the time
    - Set “ce:se_cpu_threshold=1;” and “cd_ce_taskq_disable=0;” system parameters
  - Increase interrupt blanking

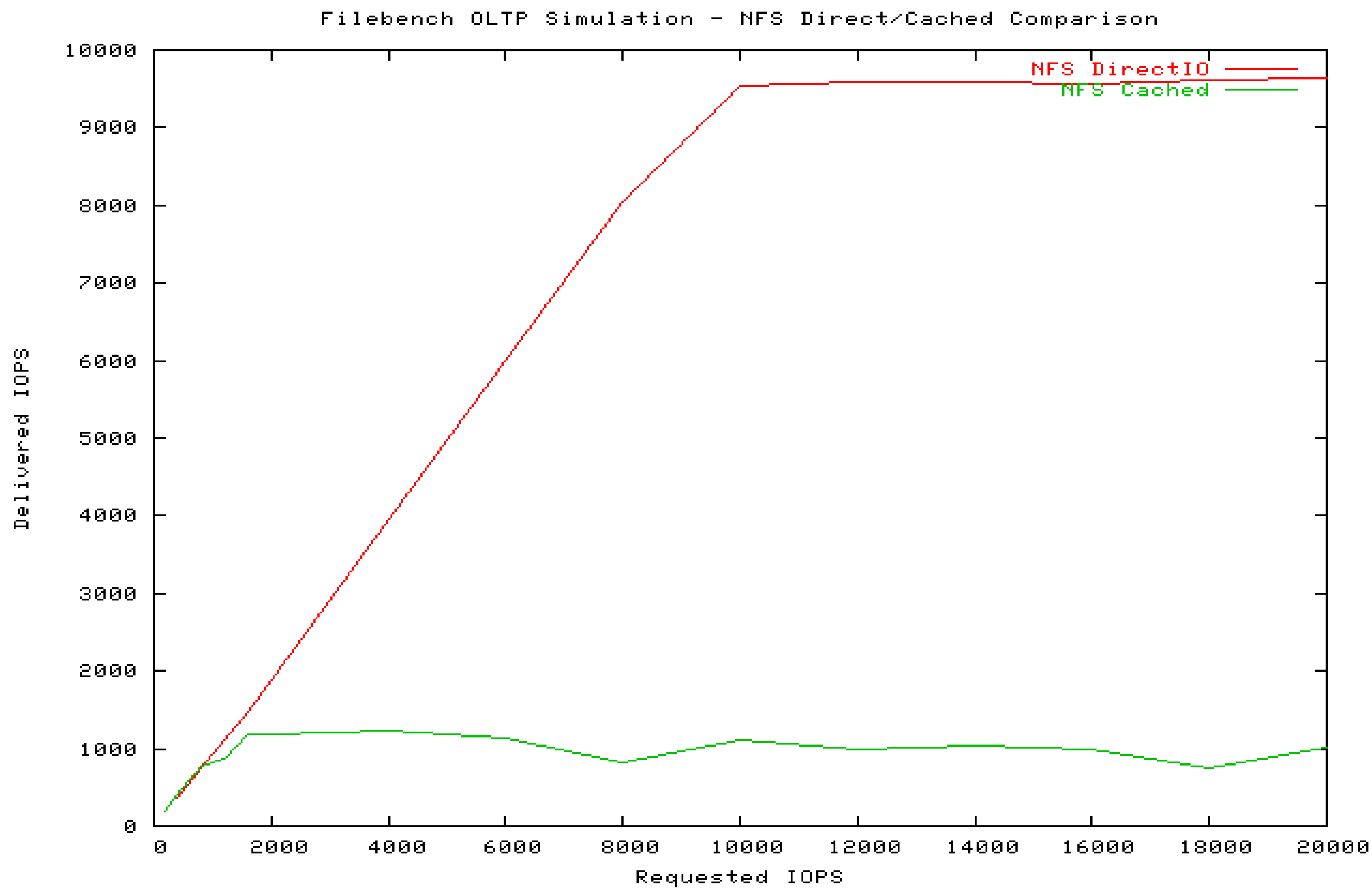
```
# ndd -set /dev/ce instance 0 (repeat for each instance)
# ndd -set /dev/ce rx_intr_time 30 (this now applies to instance set prior above)
```

  - This will optimize sequential/bandwidth
  - Note that these settings are *optimal for bandwidth*, and are not the default because they will have a *negative* effect on small-message performance on large SMP systems!

# NFS Client: Database Performance

- Sun's PAE group recently completed a database over NFS study
- Goal: Optimize NFS for databases
  - Investigate using standard OLTP Benchmark
  - Utilize Oracle 9i
  - Compare Optimized UFS vs NFS over GBE
- Starting point: Solaris 8 NFS only 10% of local UFS
  - Standard NFS client locking limits to 800 I/Os per second
- Result: Solaris 9 12/03 NFS is at 100% of local UFS
  - Solaris 9 NFS client will do 50k+ I/Os per second
  - Enable Direct I/O mode to yield improvements

# NFS Database Simulation using FileBench



# NFS Client: Open/Close optimization

- NFS uses open-to-close semantics to attempt to optimize consistency between clients
  - Drawback is wait on close for flush...
- NFS client has a mount option to optimize for this condition
  - Open/close intensive applications can be improved
  - Useful only if no sharing is occuring

```
# mount -o nocto ...
```

# NFS Server

- Ensure sufficient NFS server threads
  - Solaris 8

```
# echo "\$<threadlist |mdb -k |grep svc_run |wc -l"
```

    - Increase nfsd argument in /etc/init.d/nfs.server
  - Solaris 9+

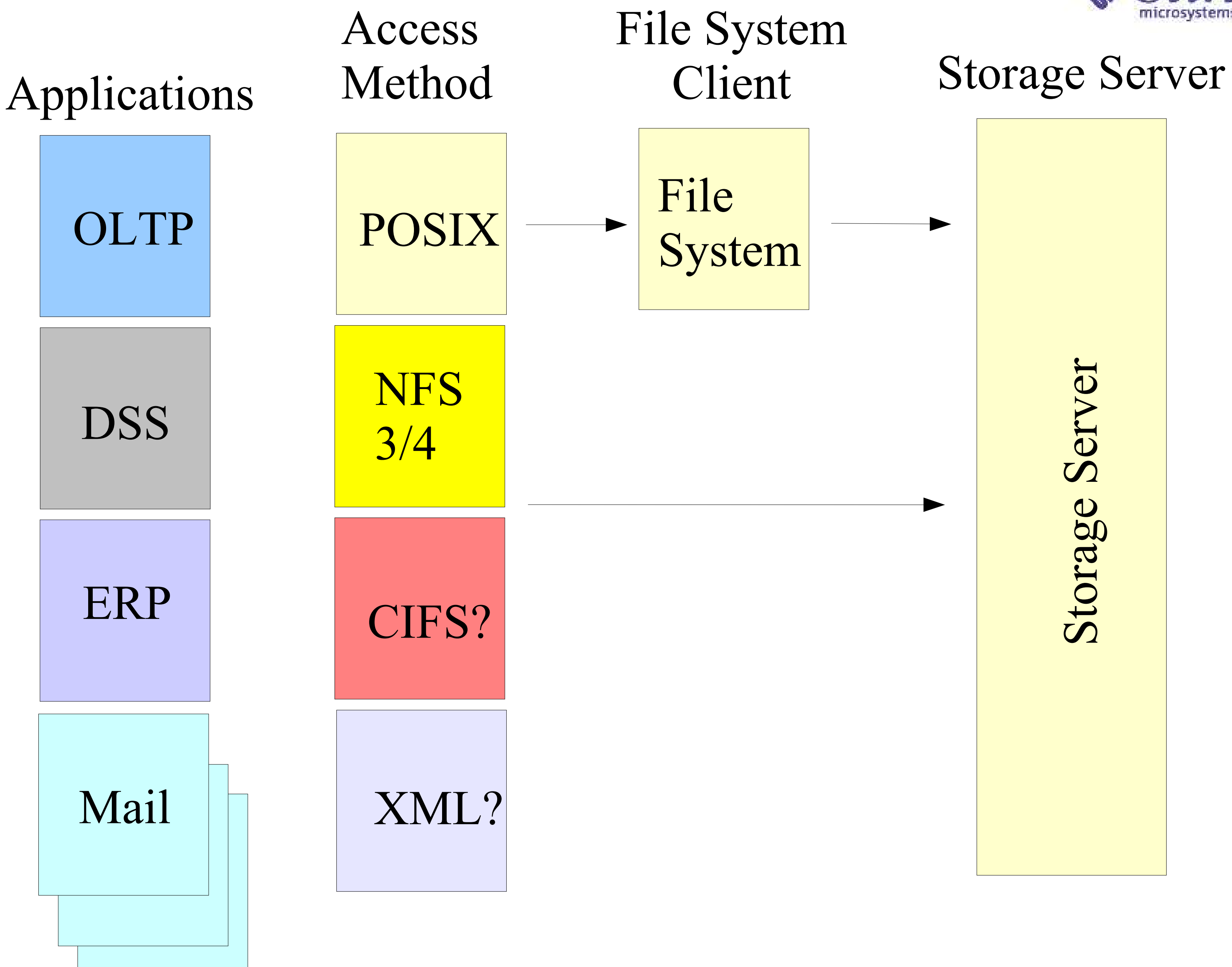
```
# pstack `pgrep nfsd` | grep nfssys |wc -l
```

  - Increase NFSD_SERVERS entry in /etc/default/nfs
  - Increase to 1024 on large systems
- Increase transfer size for sequential optimization
  - Set *nfs3_bsize* and *nfs3_max_transfer_size* system parameter on server

# Filesystem Performance Characterization

# Requirements for file-level benchmarking

- Represent Apps rather than I/Os
- Trace-derived synthesis
- Thread-level representation
- Inter-thread dependency/sync.
- Forward Path
- Extensible to new protocols
- Modular to include test of client: process/thread model, cpu efficiency etc...
- Pre-structuring/aging of file sets
- Scalable
  – Throughput, #Users
  – #Files/Directories
  – Working set size
  – #Clients
  – Client resources (mem/cpu)



Applications
Access Method
File System Client
Storage Server
OLTP
DSS
ERP
Mail
POSIX
NFS 3/4
CIFS?
XML?
File System
Storage Server

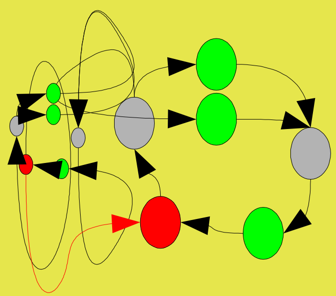
# Characterization Strategies

- I/O Microbenchmarking
  - Pros: Easy to run
  - Cons: Small test coverage, Hard to correlate to real apps
- Trace Capture/Replay
  - I/O Trace, NFS Trace, Application Trace
  - Pros: Accurate reconstruction of real application I/O mix
  - Cons: Large traces, difficult to reconstruct I/O dependencies
- Model Based
  - Distillation of trace into representive model
  - Probability based, Simulation based
  - Pros: Easy to run, Scalable in multiple dimensions
  - Cons: Care required to ensure accurate real-world representation

# Model based methodology study

Application Level Trace
- Thread
- File/Dir
- Attrs etc...

Workload Model



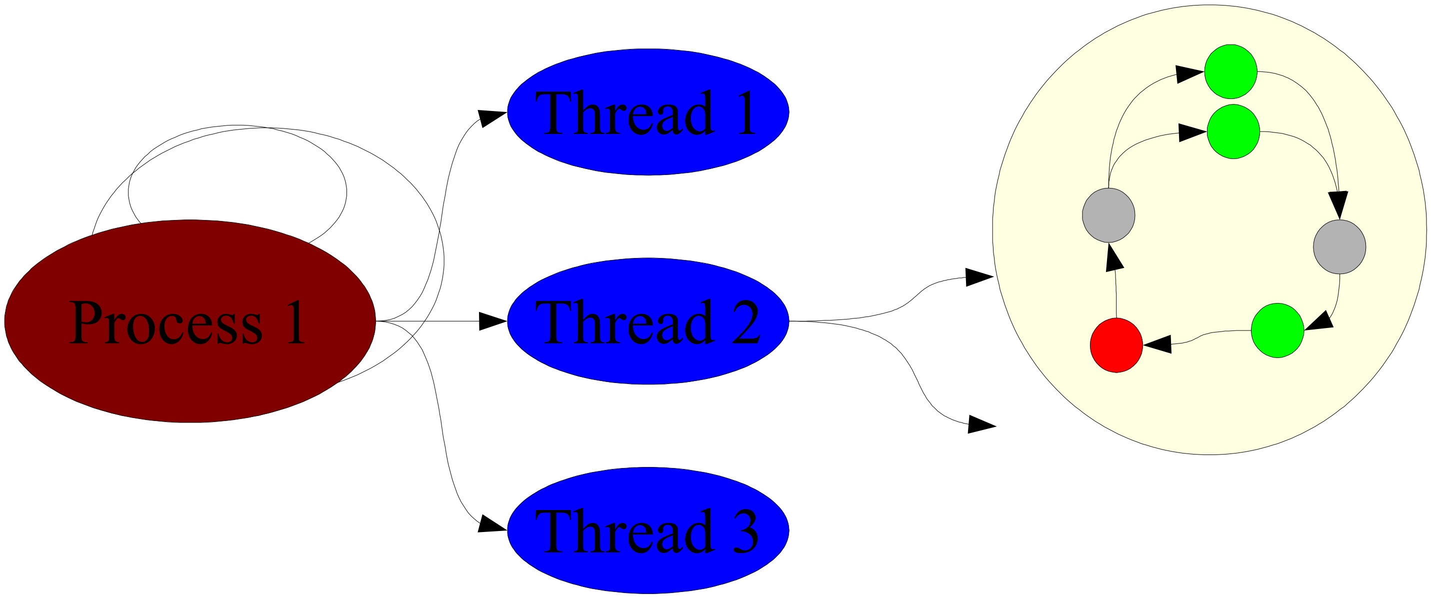
Workload Replay
- Scale Factors

Measurement Target
- FS, Client, Server etc
- Measurement Attrs

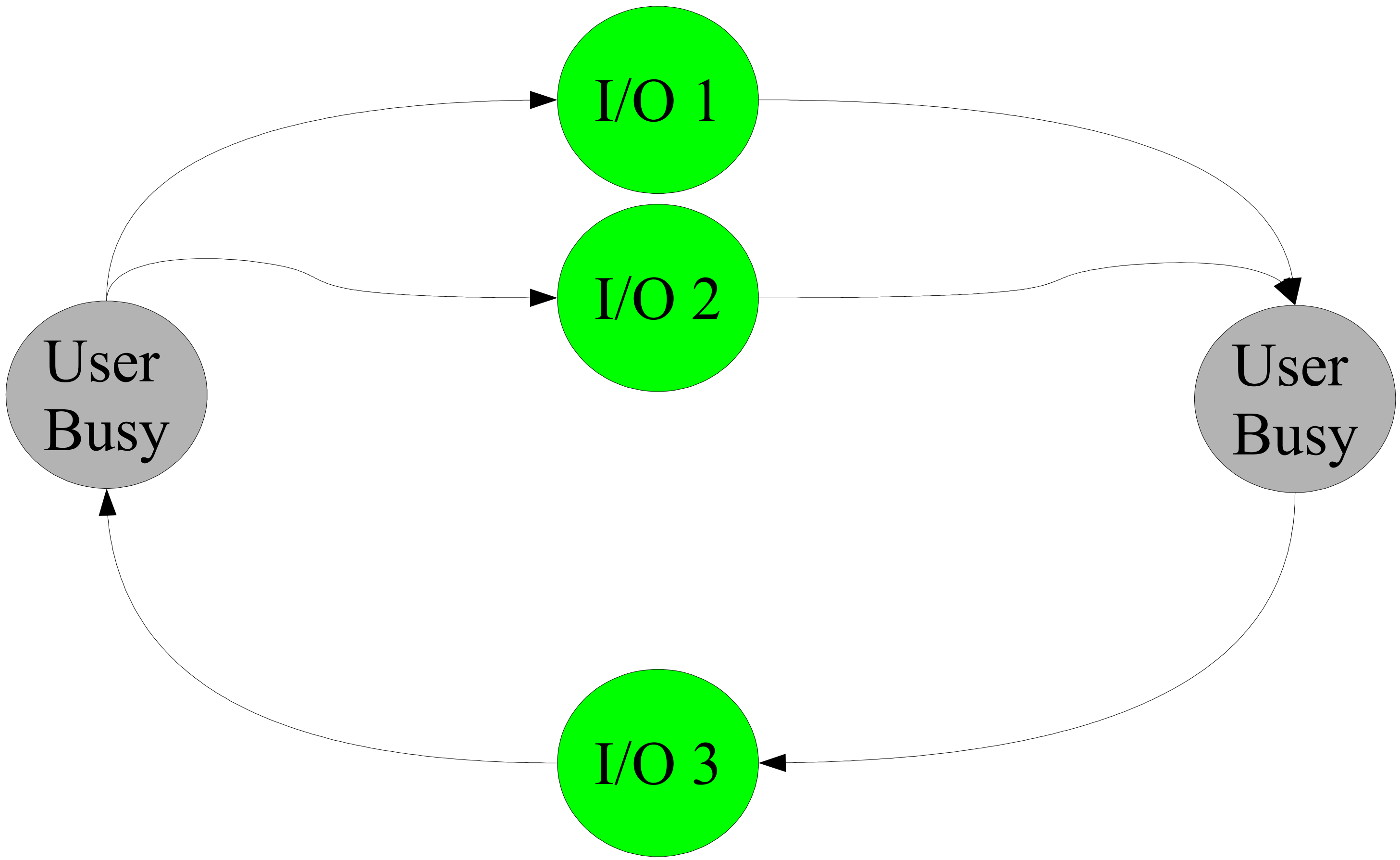
# Model Allows Complex/Important Scaling Curves

- e.g.
  - Thoughput/Latency vs. Working set size
  - Thoughput/Latency vs. #users
  - CPU Efficiency vs. Thoughput
  - Caching efficiency vs. Workingset size/Memsize

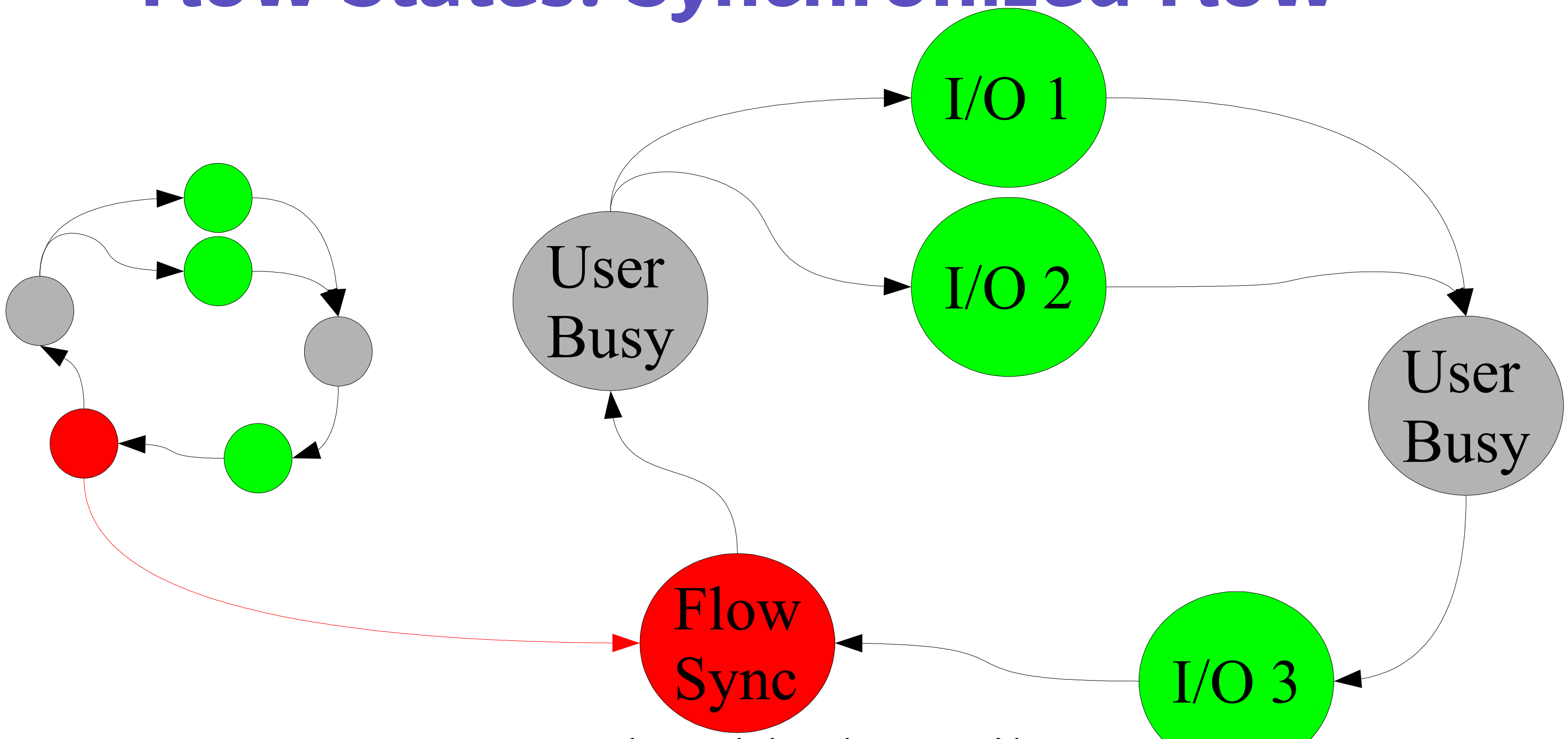
# Characterize and Simulate via Cascades of Workload Flows:



# Flow States: Open Ended Flow



# Flow States: Synchronized Flow




Flow blocks until completion of other flow

# Examples of Per-flow Operations

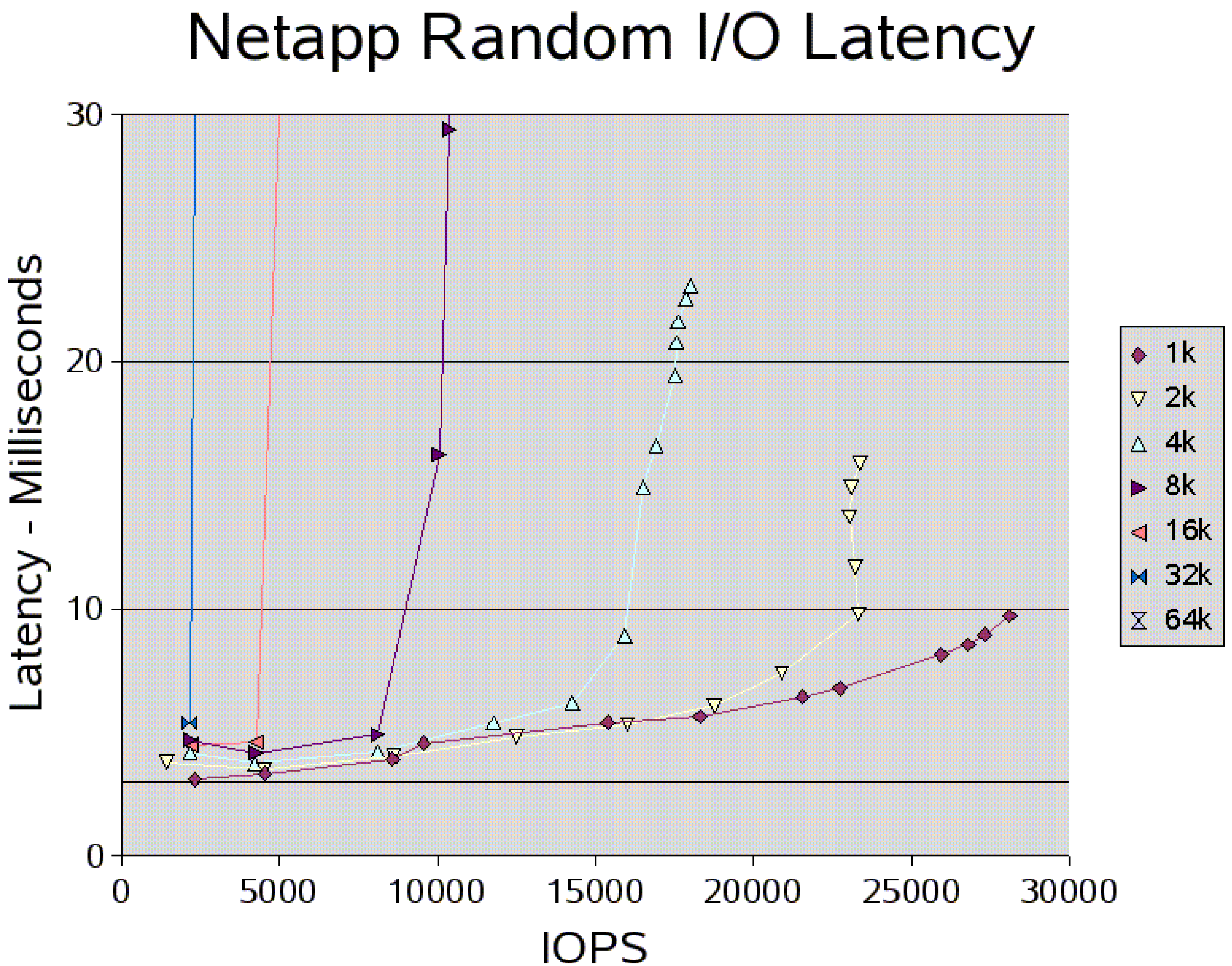
- Types
  - Read
  - Write
  - Create
  - Delete
  - Append
  - Getattr
  - Setattr
  - Readdir
  - Semaphore block/post
  - Rate limit
  - Throughput limit

- Attributes
  - Sync_Data
  - Sync_Metadata
  - IO Size
  - I/O Pattern, probabilities
  - Working set size
  - Etc...

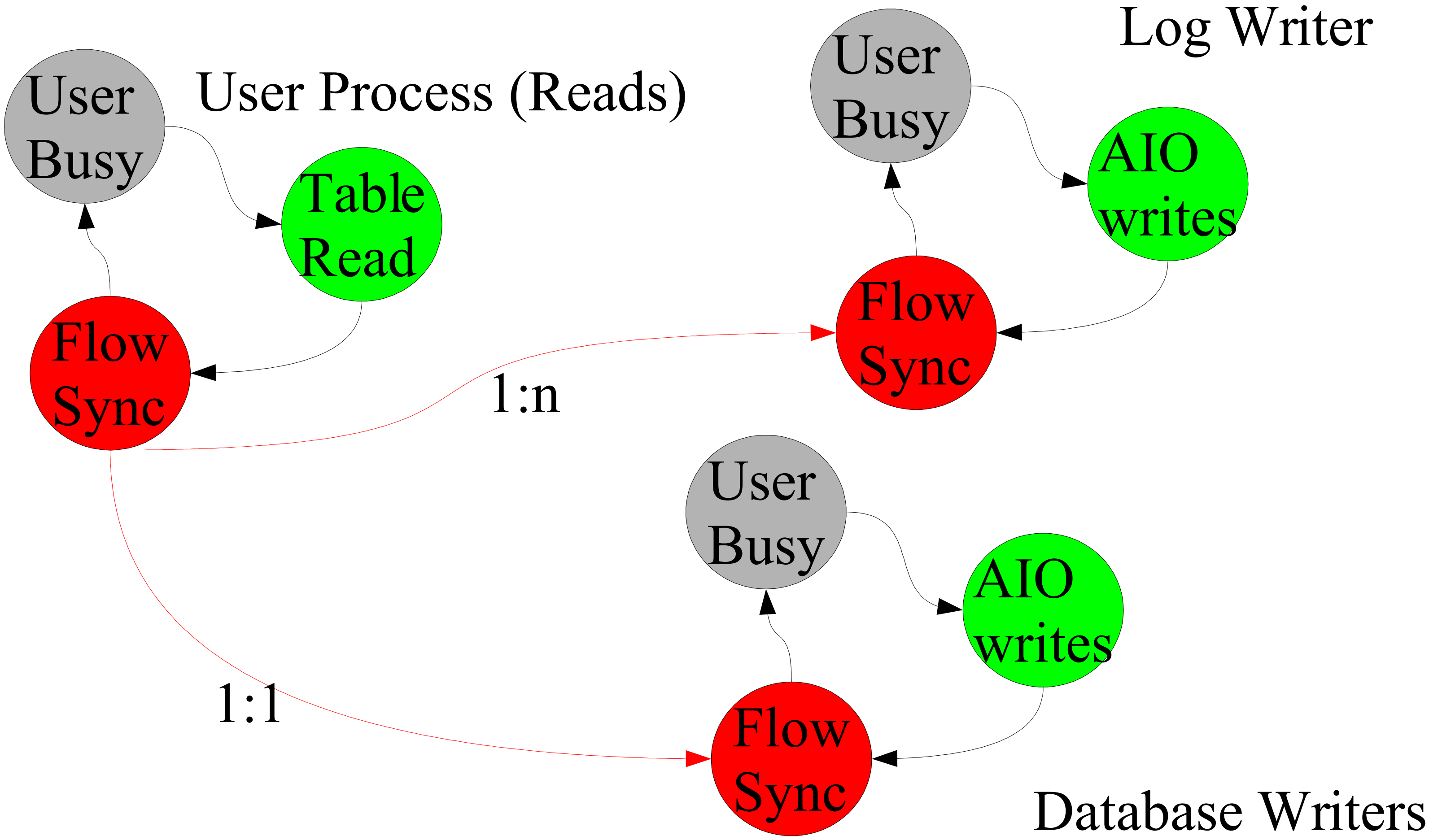
# Simple Random I/O Workload Description

```
define file name=bigfile0,path=$dir,size=$filesize,prealloc,reuse,paralloc

define process name=rand-read,instances=1
{
  thread name=rand-thread,memsize=5m,instances=$nthreads
  {
    flowop read name=rand-read1,filename=bigfile0,iosize=$iosize,random
    flowop eventlimit name=rand-rate
  }
}
```

# Random I/O – NFS V3

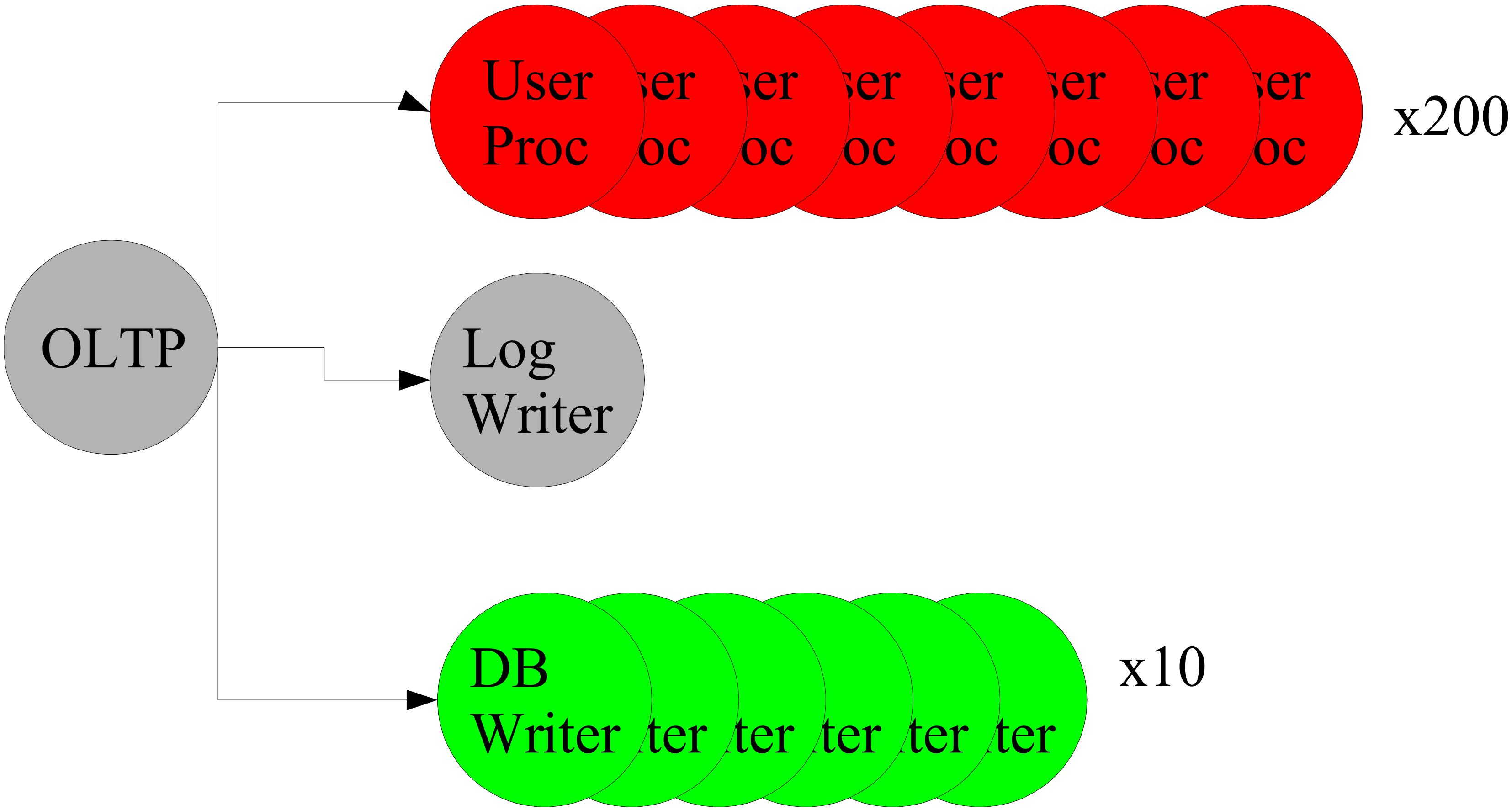


# Database Emulation Overview



# Database Emulation Process Tree

User Process (Reads)



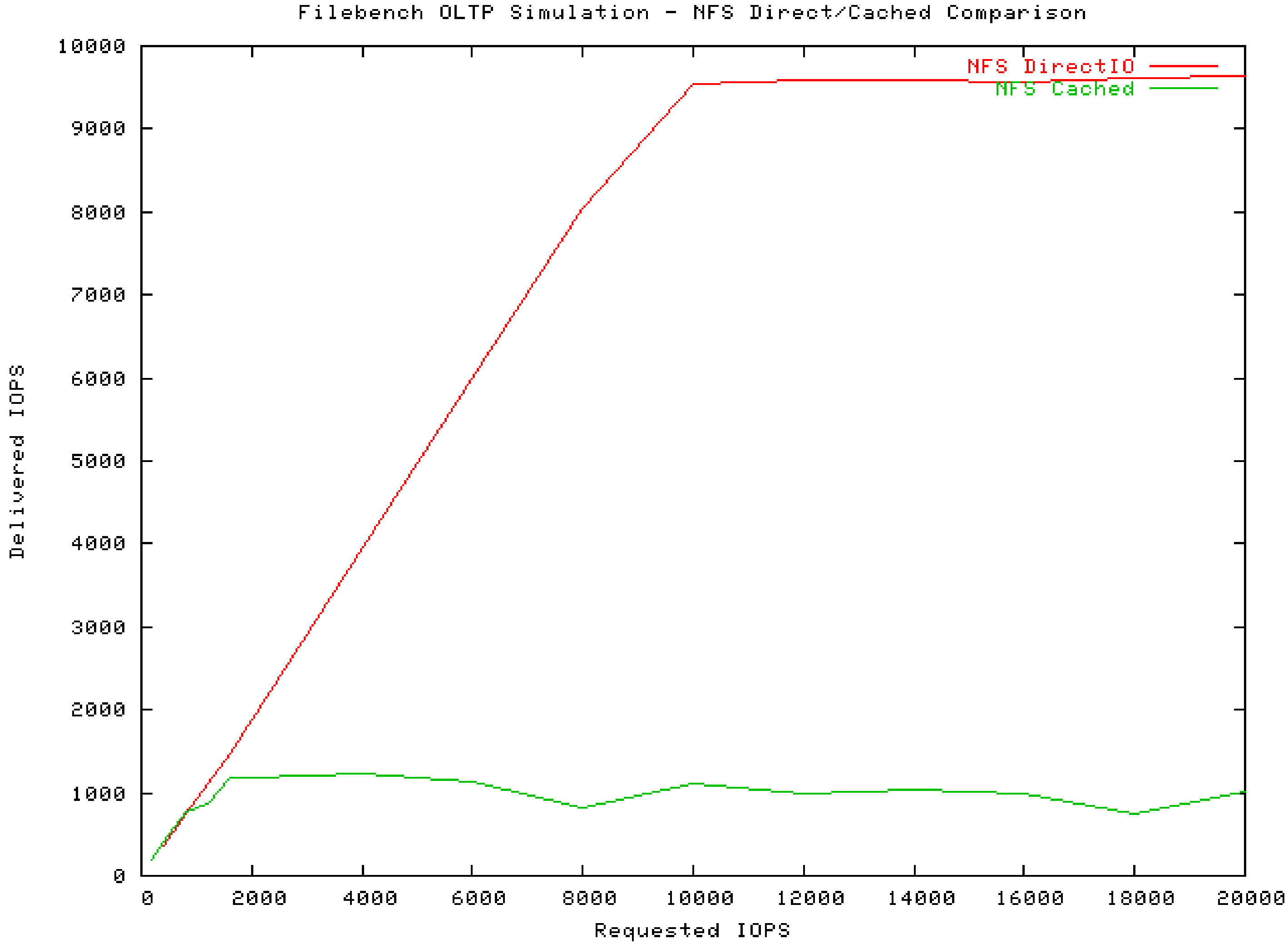
# Simplified OLTP Database Program

```
define file name=logfile,path=$dir,size=1g,reuse,prealloc,paralloc
define file name=datafilea,path=$dir,size=$filesize,reuse,prealloc,paralloc
define process name=dbwr,instances=$ndbwriters
{
 thread name=dbwr,memsize=$memperthread,useism
 {
  flowop aiowrite name=dbaiowrite-a,filename=datafilea,
     iosize=$iosize,workingset=10g,random,dsync,directio,iters=10
  flowop hog name=dbwr-hog,value=10000
  flowop semblock name=dbwr-block,value=100,highwater=10000
  flowop aiowait name=dbwr-aiowait
 }
}

define process name=lgwr,instances=1
{
 thread name=lgwr,memsize=$memperthread,useism
 {
  flowop write name=lg-write,filename=logfile,
     iosize=256k,workingset=1g,random,dsync,directio
  flowop semblock name=lg-block,value=320,highwater=1000
 }
}
define process name=shadow,instances=$nshadows
{
 thread name=shadow,memsize=$memperthread,useism
 {
  flowop read name=shadowread-a,filename=datafilea,
    iosize=$iosize,workingset=10g,random,dsync,directio
  flowop hog name=shadowhog,value=$usermode
  flowop sempost name=shadow-post-lg,value=1,target=lg-block,blocking
  flowop sempost name=shadow-post-dbwr,value=1,target=dbwr-block,blocking
  flowop eventlimit name=random-rate
 }
}
```

# OLTP Program – Benchmark Result Detail

```
Flowop totals:

shadow-post-dbwr         4554ops/s   0.0mb/s    215.7ms/op        91us/op-cpu
shadow-post-lg           4555ops/s   0.0mb/s      0.7ms/op        21us/op-cpu
shadowhog                4546ops/s   0.0mb/s      2.5ms/op       111us/op-cpu
shadowread               4455ops/s   0.9mb/s     23.2ms/op        89us/op-cpu
lg-block                  100ops/s   0.0mb/s    605.2ms/op       305us/op-cpu
lg-write                  100ops/s   0.4mb/s     96.2ms/op      1962us/op-cpu
dbwr-aiowait             4445ops/s   0.0mb/s    144.0ms/op       242us/op-cpu
dbwr-block               4445ops/s   0.0mb/s      9.6ms/op        44us/op-cpu
dbwr-hog                 4445ops/s   0.0mb/s      1.1ms/op        50us/op-cpu
dbaiowrite               4449ops/s   0.9mb/s      0.2ms/op        17us/op-cpu

IO Summary: 9087.7 ops/s, 4547/4496 r/w    18.0mb/s,      129uscpu/op
```

# NFS OLTP – IOPS Scaling



# Workload Discussion

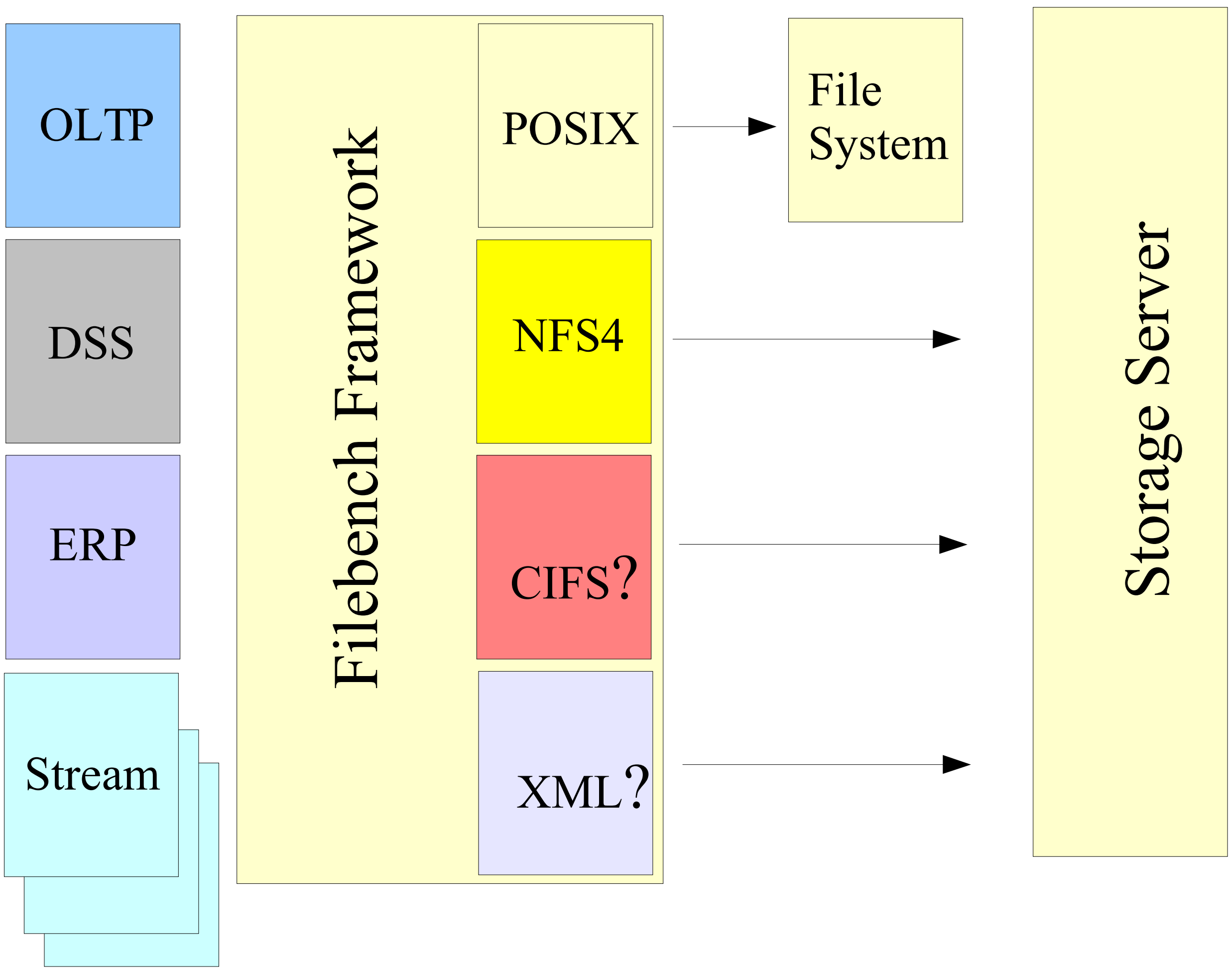
**File Access**

| Workload | File Size | # files | #Streams | Sharing | I/O Mix | Seek Mode | Access type mmap/posix |
|---|---|---|---|---|---|---|---|
| Web Server | Small | Large | Large | Low | <5% | 90% Random Read/10% Sequential Write | Both |
| Small DB | Large | Small | ~100 | High | 50r/50w, 1% large sequential | 99% Random | POSIX |
| Large DB | Large | Small | ~1000 | High | 50r/50w, 1% large sequential | 99% Random | POSIX |
| DB Mail Server | Large | Small | >1000 | High | ? | | |
| NFS Mail Server | Moderate | Moderate | >10k | Low | ? | Sequential | POSIX |
| HPTC | Huge | Small | Small | Low | 50r/50w | Sequential | POSIX |
| SW Development | Small | Large | >1000 | Low | 5r/5w/90a | Sequential | POSIX |
| Video Streaming | | | | | | | |

**I/O Characteristics**

| Workload | App/IO CPU Content | Typical IOPS | Data Set Size | Working Set Size | Typical I/O Size | Typical Bandwidth |
|---|---|---|---|---|---|---|
| Web Server | 99/1 | <1000 per cient | | | <64k | <1MB/s |
| Small DB | 90/10 | ~1000 | 1-10GB | 50.00% | Random 2-8k, 128k sequential | ~10MB/s |
| Large DB | 80/20 | >10000 | 10GB-1TB | 30.00% | Random 2-8k, 128k sequential | 50MB/s |
| DB Mail Server | 90/10? | | | | Small? | ? |
| NFS Mail Server | 90/10? | Low | | | Large reads, small writes | 1-10MB/s |
| HPTC | 80/20? | ~1000? | | | ~1MB | >100MBs Client, 1GB/s Server |
| SW Development | 95/5? | ~1000 | | | ~32k | ~100mb/s |

# Filebench Achitecture



# Running filebench...

**Example varmail run:**

```
filebench> load varmail

 Varmail personality successfully loaded
 Usage: set $dir=<dir>
        set $filesize=<size>    defaults to 16384
        set $nfiles=<value>     defaults to 1000
        set $dirwidth=<value>   defaults to 20
        set $nthreads=<value>   defaults to 1
        set $meaniosize=<value> defaults to 16384
        run <runtime>

filebench> set $dir=/tmp

filebench> run 10

 Fileset mailset: 1000 files, avg dir = 20, avg depth = 2.3,mbytes=15
 Preallocated fileset mailset in 1 seconds
 Starting 1 filereader instances
 Starting 1 filereaderthread threads
 Running for 10 seconds...
 IO Summary: 21272 iops 2126.0 iops/s, (1063/1063 r/w) 32.1mb/s,338us cpu/op, 0.3ms latency
```

# Example Performance Comparison

- Throughput:

| operations/s | FS-A | FS-B | |
|---|---|---|---|
| copyfiles | 1403 | 1431 | +2.0% |
| createfiles | 2433 | 2438 | +0.2% |
| deletefiles | 778 | 833 | +7.1% |
| fileserver | 4264 | 2202 | -48.4% |
| oltp | 16840 | 866 | -94.9% |
| randomread | 78 | 37 | -53.3% |
| singlestreamread | 35 | 36 | +2.9% |
| multistreamread | 50 | 60 | +20.0% |
| varmail | 2231 | 5591 | +150.6% |
| webproxy | 7781 | 2255 | -71.0% |

# Example Performance Comparison

- Client Microseconds per operation:

| | uSec/op FS-A | FS-B | |
|---|---|---|---|
| copyfiles | 1076 | 2294 | 2.1x |
| createfiles | 2131 | 8952 | 4.2x |
| deletefiles | 1001 | 1999 | 2.0x |
| fileserver | 3152 | 24994 | 7.9x |
| oltp | 586 | 13557 | 23.1x |
| randomread | 742 | 2329 | 3.1x |
| singlestreamread | 16553 | 27372 | 1.7x |
| multistreamread | 18001 | 25032 | 1.4x |
| varmail | 1078 | 3168 | 2.9x |
| webproxy | 4242 | 22418 | 5.3x |

# Filebench Status

- Porting Status
  - S8, 10, x86, SPARC, Linux (2.6/Fedora)
- Early access workload models
  - Random Read/Write (Random block I/O)
  - Sequential I/O (single or multi-stream block I/O)
  - OLTP Database (Oracle Emulator)
  - File Server (Multi-user file intensive)
  - Varmail (Postmark style /var/mail emulation)
  - Webserver (Multi-threaded read + sequential weblog)
  - Webproxy (Multi-threaded read, create, write, delete)
  - Copyfiles (Copy a file tree)

# ZFS:

# Coming Soon...

# ZFS Overview

- **Provable data integrity**

  Detects and corrects silent data corruption

- **Immense capacity**

  The world's first 128-bit filesystem

- **Simple administration**

  “You're going to put a lot of people out of work.”
  -- ZFS beta customer

- **Smokin' performance**

  Already faster than UFS and VxFS, sometimes by multiples

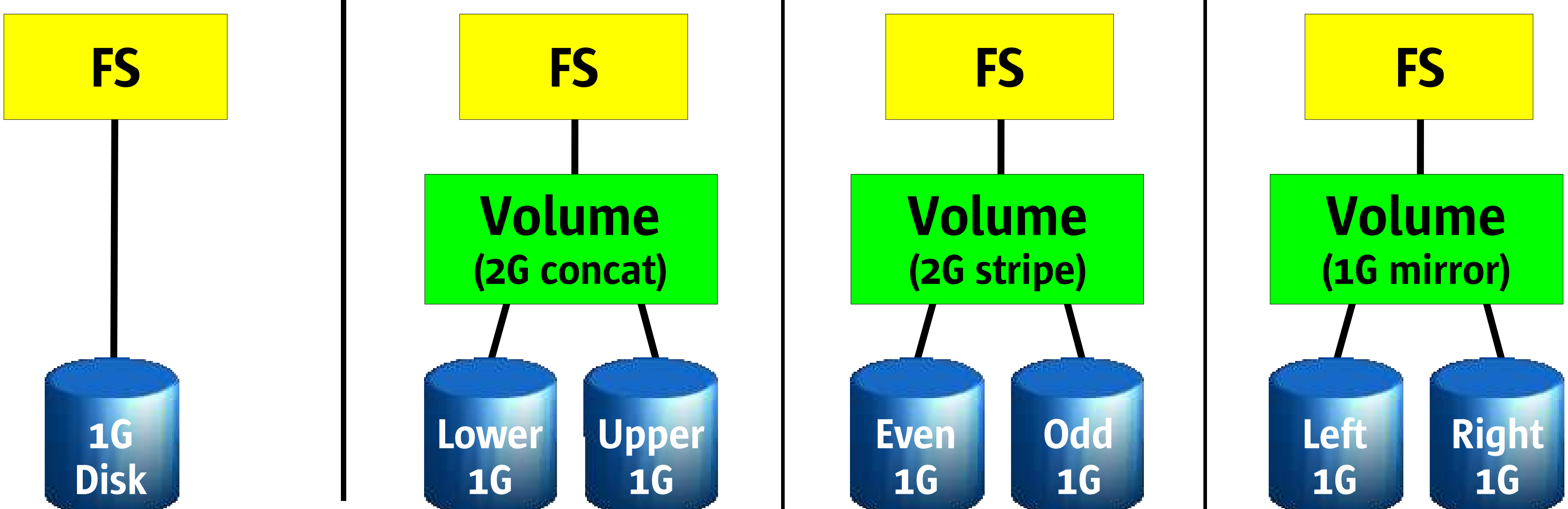
# ZFS Principles

- Pooled storage
  - Completely eliminates the antique notion of volumes
  - Does for storage what VM did for memory
- End-to-end data integrity
  - Historically considered “too expensive”
  - Turns out, no it isn't
  - And the alternative is unacceptable
- Everything is transactional
  - Keeps things always consistent on disk
  - Removes almost all constraints on I/O order
  - Allows us to get huge performance wins

# Background: Why Volumes Exist

In the beginning, each filesystem managed a single disk.

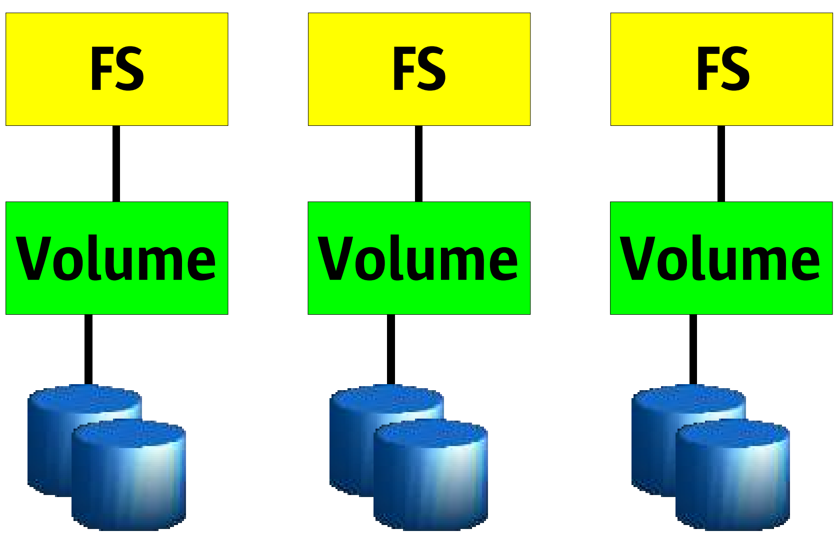
- Customers wanted more space, bandwidth, reliability
  - Rewrite filesystems to handle many disks: hard
  - Insert a little shim (“volume”) to cobble disks together: easy
- An industry grew up around the FS/volume model
  - Filesystems, volume managers sold as separate products
  - Inherent problems in FS/volume interface can't be fixed



# FS/Volume Model vs. ZFS

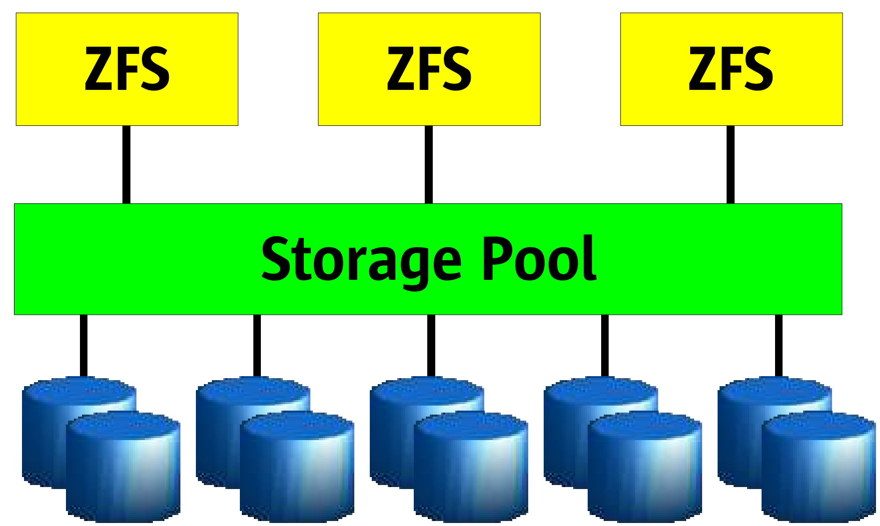
## Traditional Volumes

- Abstraction: virtual disk
- Partition/volume for each FS
- Grow/shrink by hand
- Each FS has limited bandwidth
- Storage is fragmented, stranded




## ZFS Pooled Storage

- Abstraction: malloc/free
- No partitions to manage
- Grow/shrink automatically
- All bandwidth always available
- Pool allows space to be shared



# FS/Volume Model vs. ZFS

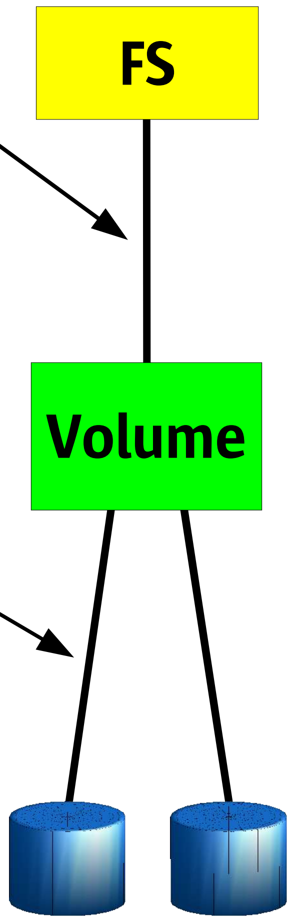
## FS/Volume I/O Stack

Block Device Interface

- "Write this block, then that block, ..."
- Loss of power = loss of on-disk consistency
- Workaround: journaling, which is slow & complex

Block Device Interface

- Write each block to each disk immediately to keep mirrors in sync
- Loss of power = resync
- Synchronous and slow



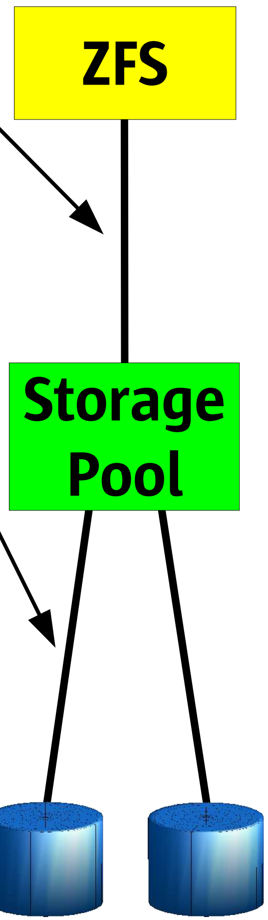

## ZFS I/O Stack

Object-Based Transactions

- "Make these 7 changes to these 3 objects"
- All-or-nothing
- Always consistent on disk
- No journal – not needed

Transaction Group Batch I/O

- Write whole group of transactions at a time; again, all-or-nothing
- No resync if power lost
- Schedule, aggregate, and issue I/O at will
- Runs at platter speed



# ZFS Administation

- Pooled storage – no more volumes!
  - All storage is shared – no wasted space
  - Filesystems are cheap: like directories with mount options
  - Grow and shrink are automatic
  - No more fsck(1M), format(1M), /etc/vfstab, /etc/dfs/dfstab...
- Unlimited snapshots
  - Read-only snapshots: point-in-time copy of the data
  - Read/write snapshots: multiple variants (branches) of the data
  - Lets users recover lost data without sysadmin intervention
- Host-neutral on-disk format
  - Change server from x86 to SPARC, it just works
  - Adaptive endianness ensures neither platform pays a tax
- 100% online administration

# Resource Management

# Solaris Resource Management

- Introduction to Resource Management
- Solaris 9 Resource Manager
- Futures/Roadmap

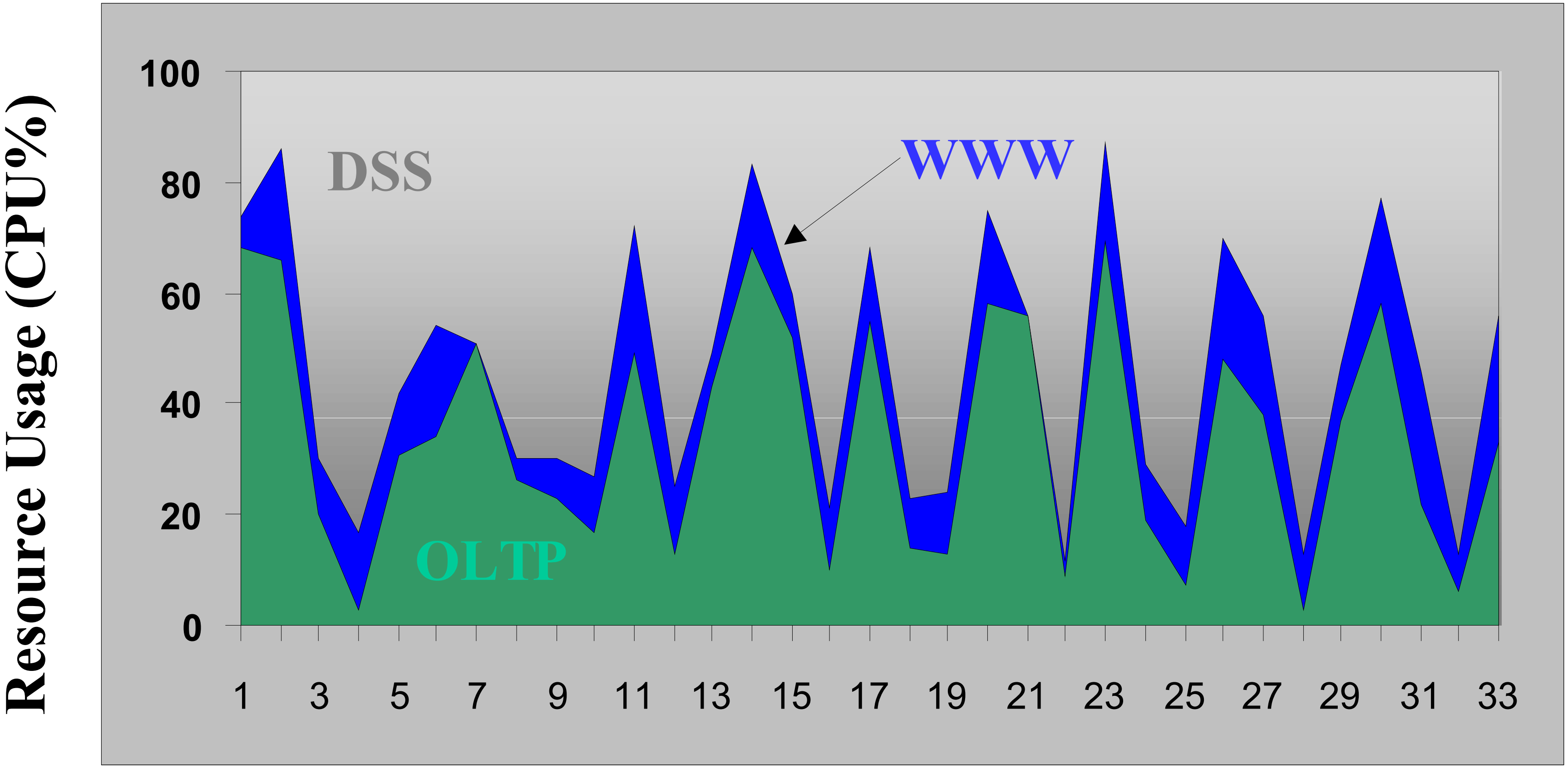
# Service Level Management:

- Computer, how is my sales-order web service running?
  - A little slower than you'd like actually, each operation is taking 10 seconds, which is outside your 1 second goal
- Computer, why is that?
  - It's because each web transaction is having to wait 9.6 seconds for CPU on the database server
- Oh. Computer, please make my sales-order web service the most important application on the server.
  - Sure, I'll give more CPU to the sales-order web service, by taking some from the ad—hoc queries that your development engineers are running
- Computer, how is my sales order web service running now?
  - Exactly how you'd like, each operation is taking .4 seconds

# Service Level Management Today

- 1 Service level per "system"
- Tune or adjust size of system to meet service levels
- ~1 box per application as a result
- Poor utilization per box (~15%)
- Server consolidation!

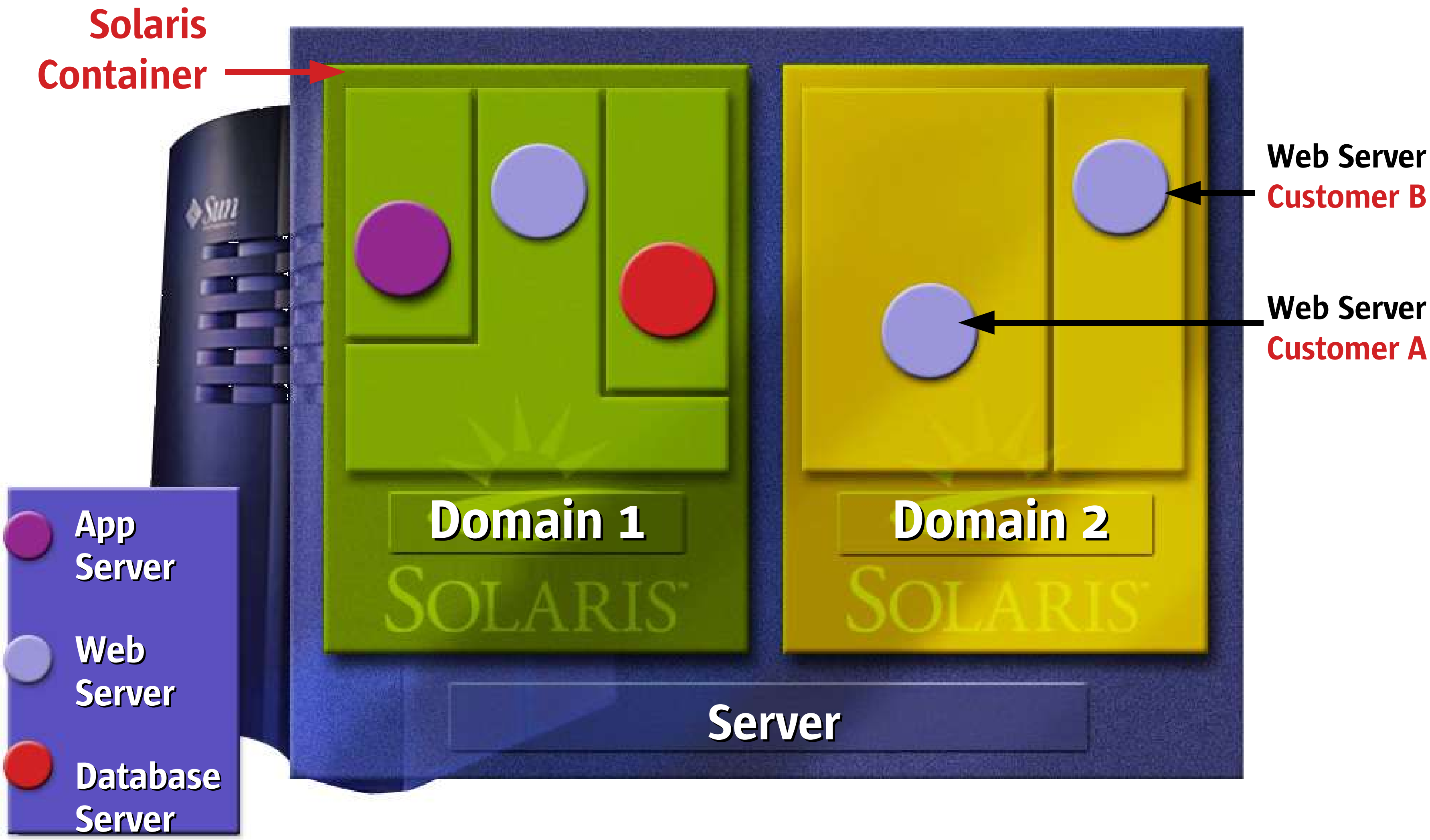
# Policy based performance:



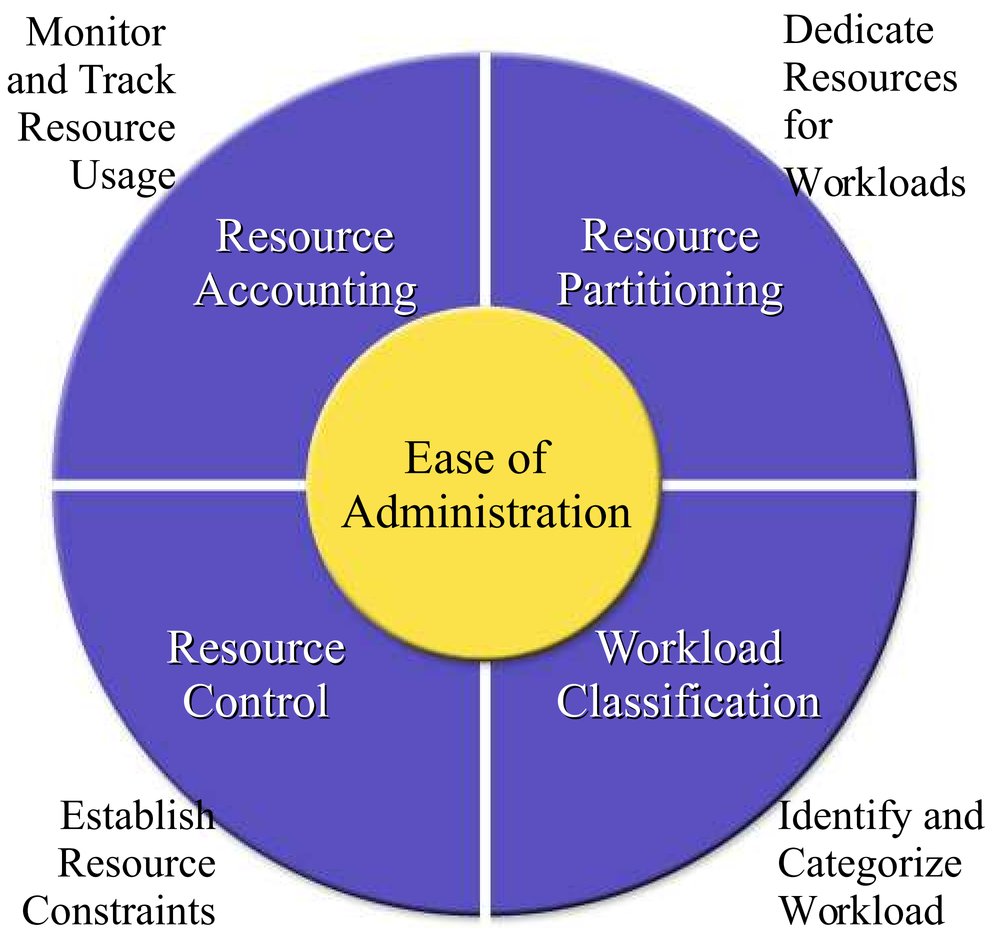
# Enabling Server Consolidation

- Full Resource Containment and control
  - Provide predictable service levels
- Security isolation
  - Prevent unauthorized access
- Fault isolation
  - Minimize fault propagation and unplanned downtime

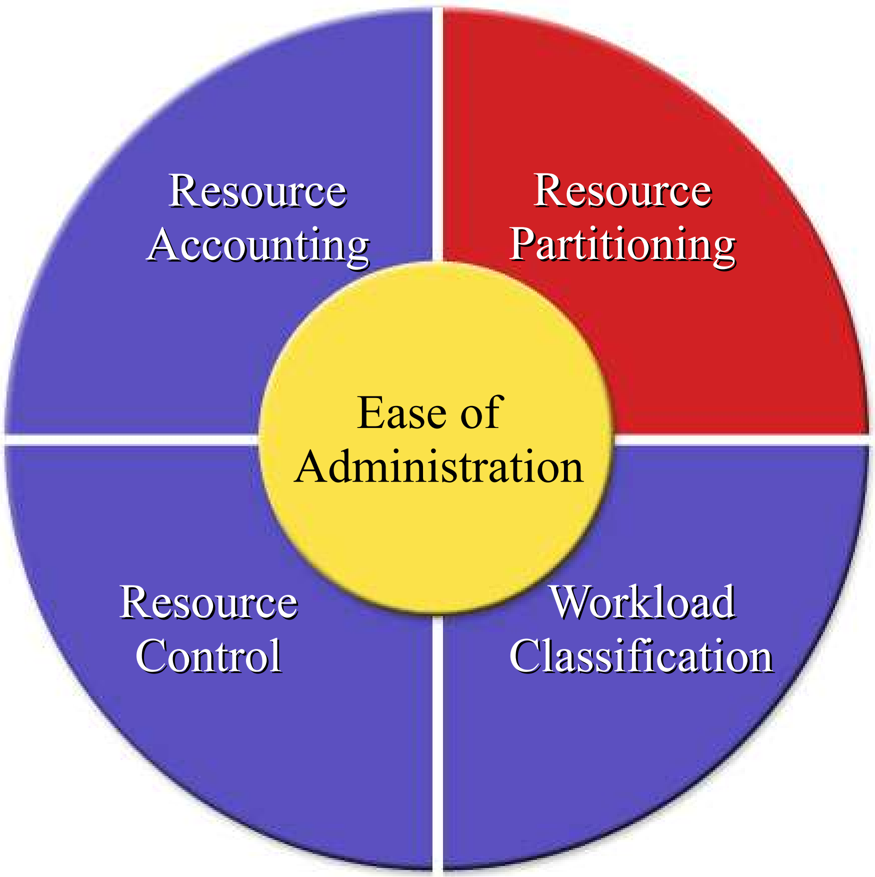
# Server Virtualization



# Resource Management Is...



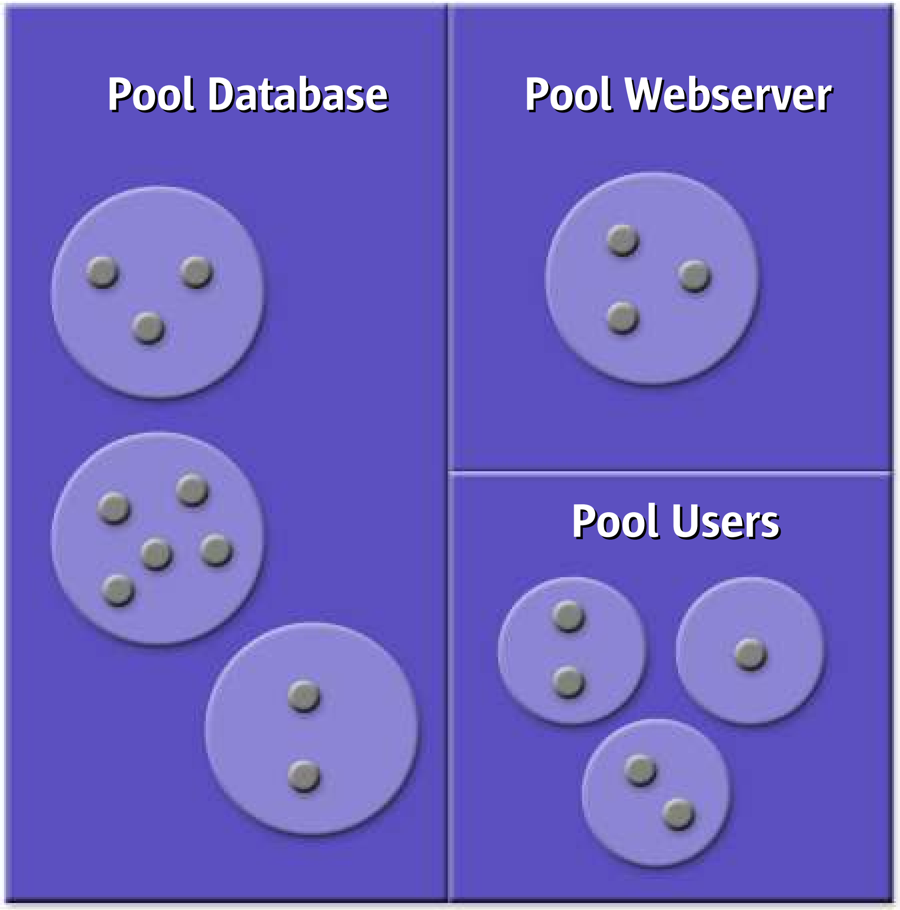

# Resource Partitioning




- Provides resource containment
- Ability to dedicate a set of resources for workloads
- Improves service level predictability

# Resource Pools

- Partition systems into logical groups of resources for exclusive use of workloads
- Provide workloads with consistent service levels through resource containment



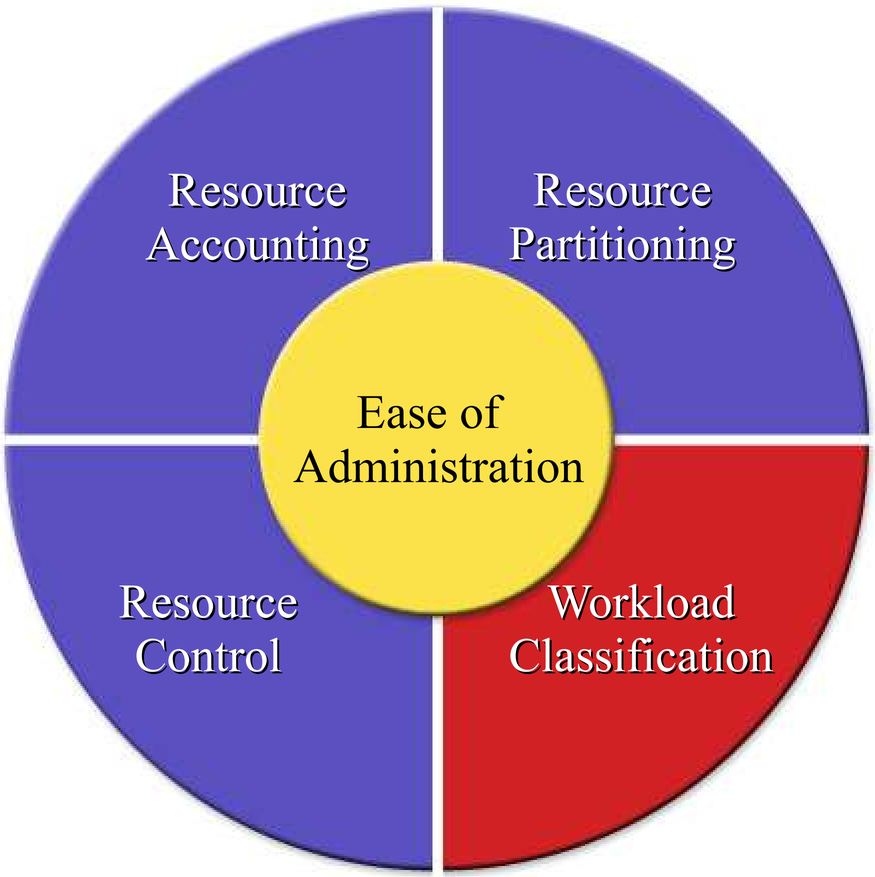
# Resource Pools — More Details

- Each pool can be configured to use a different scheduling class (e.g., TS, FSS)
- Pool configuration is persistent upon reboot
- Ability to define multiple pool configurations to suit business requirements

# Resource Pools — More Details

- Some Commands
  - `Pooladm(1M)`
    - Apply pool configuration to the system
  - `poolcfg(1M)`
    - Create, modify, and delete pool configurations
  - `poolbind(1M)`
    - Bind processes, tasks, or projects to pools

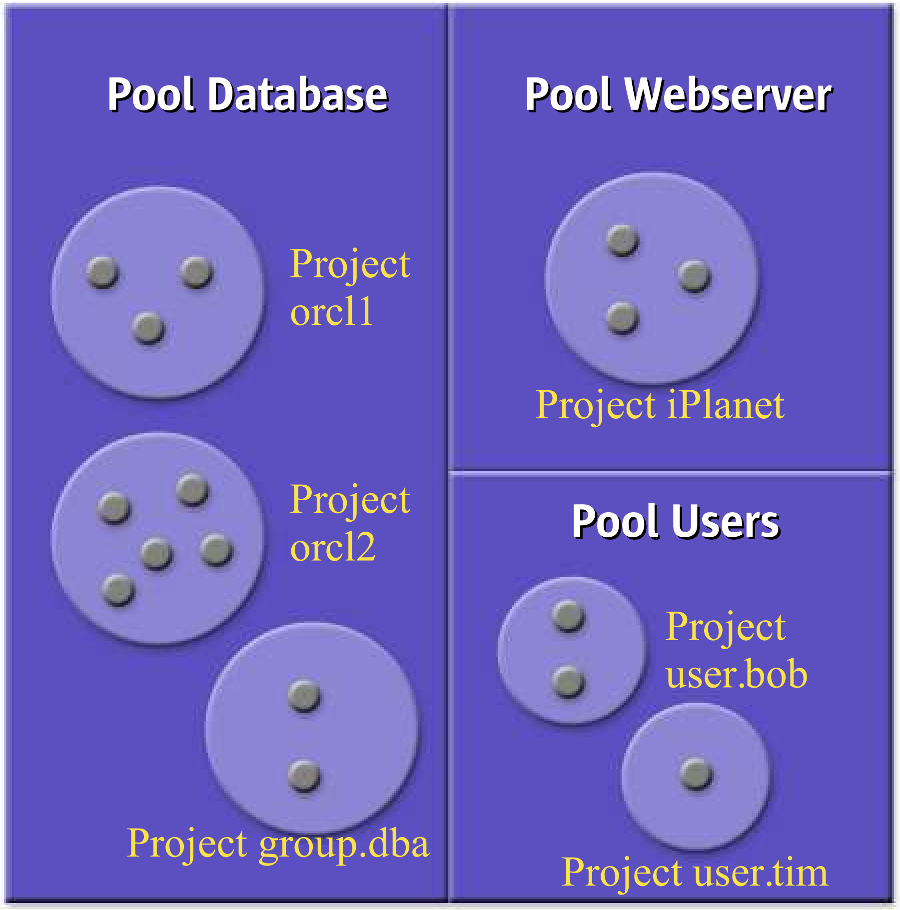
# Workload Classification




- Ability to give workloads a label
- Ability to distinguish between workloads and track them

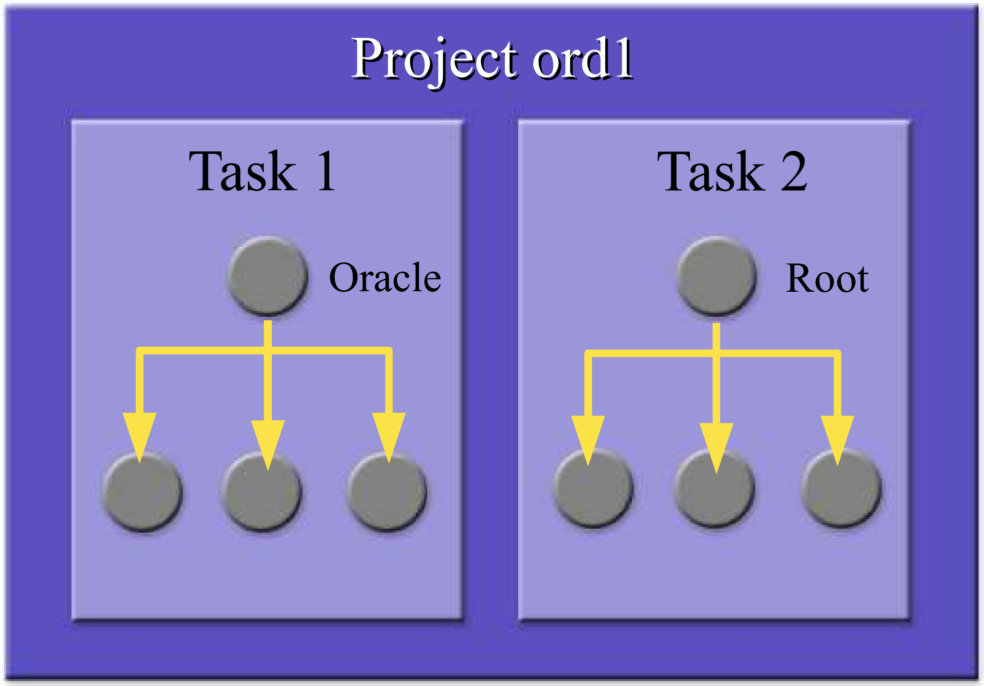
# Project

- A tag to classify a workload (a single or a group of users/ applications)



# Project — More Details

- A project consists of one or more tasks
  - Task: a collection of processes doing a single job within a project



# Project — More Details

- Project configurations can be stored in local files/NIS/LDAP

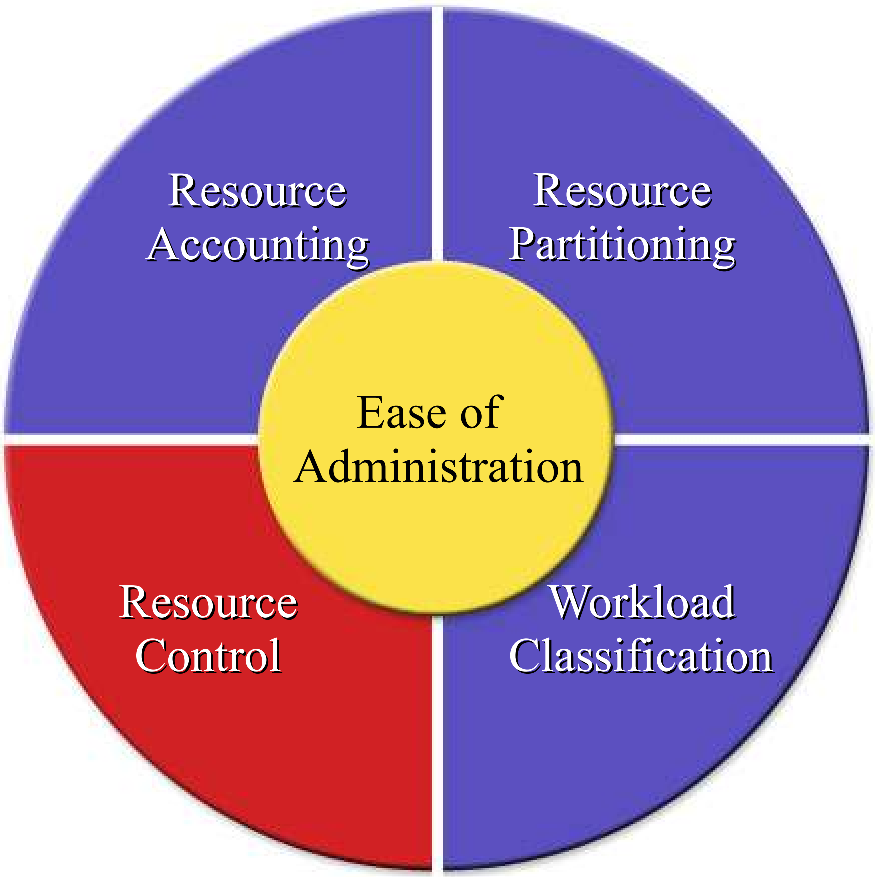
**/etc/project**

```
System:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
user.oltp:1003::root:dba:project.cpu-shares=(
  privileged,40,none)
user.webserver:1200::root::project.cpu-shares=(
  privileged,0,none)
```

# Project — More Details

- Some Commands
  - `proj{add,mod,del}(1M)`
    - Add, modify, delete projects
  - `projects(1M)`
    - Print project membership of user
  - `newtask(1M)`
    - Create a new task

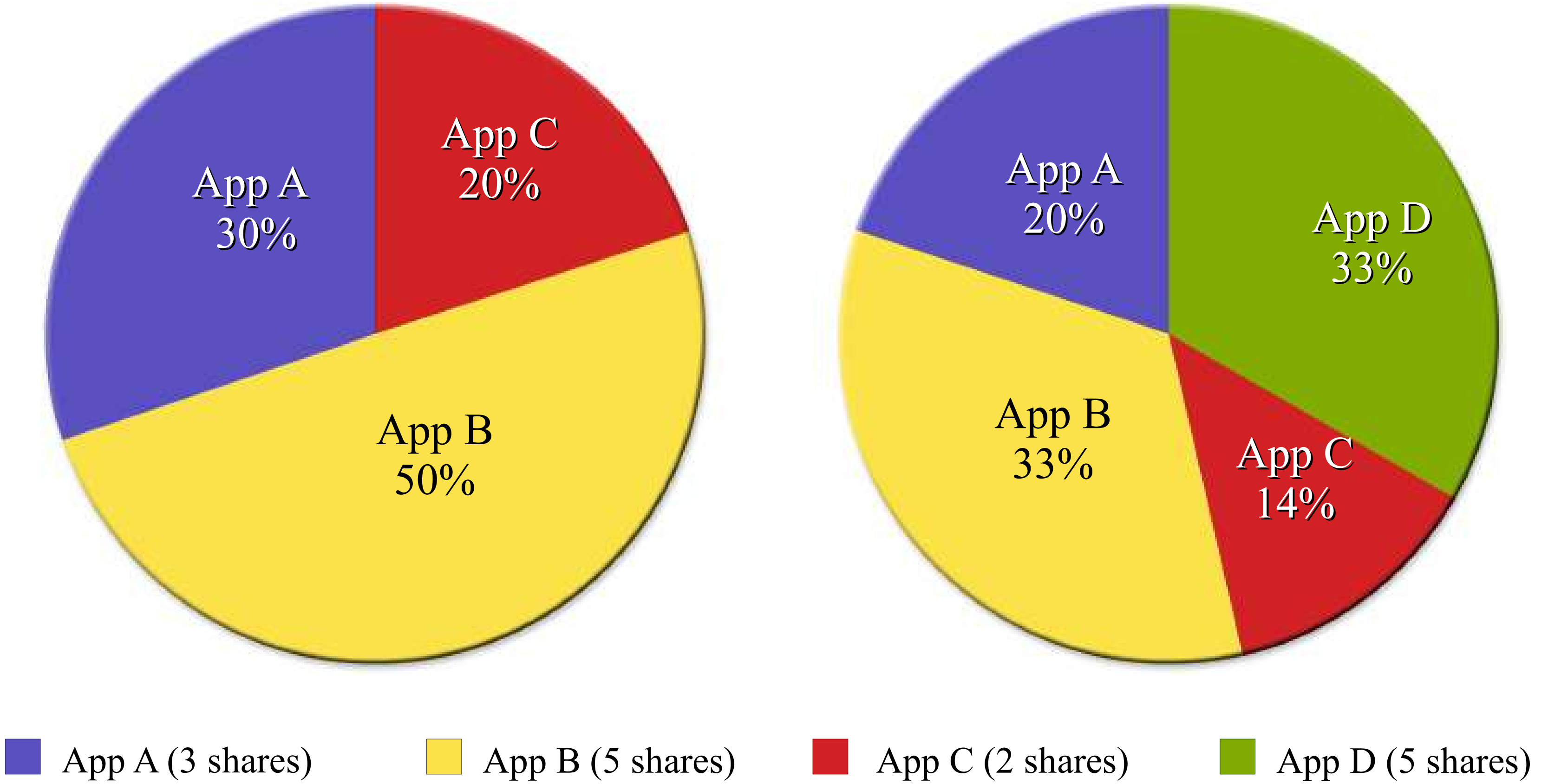
# Resource Control




- Prevent processes from running wild
- Take appropriate actions when limits are reached

# Fair Share Scheduler

- Shares describe relative ratio...



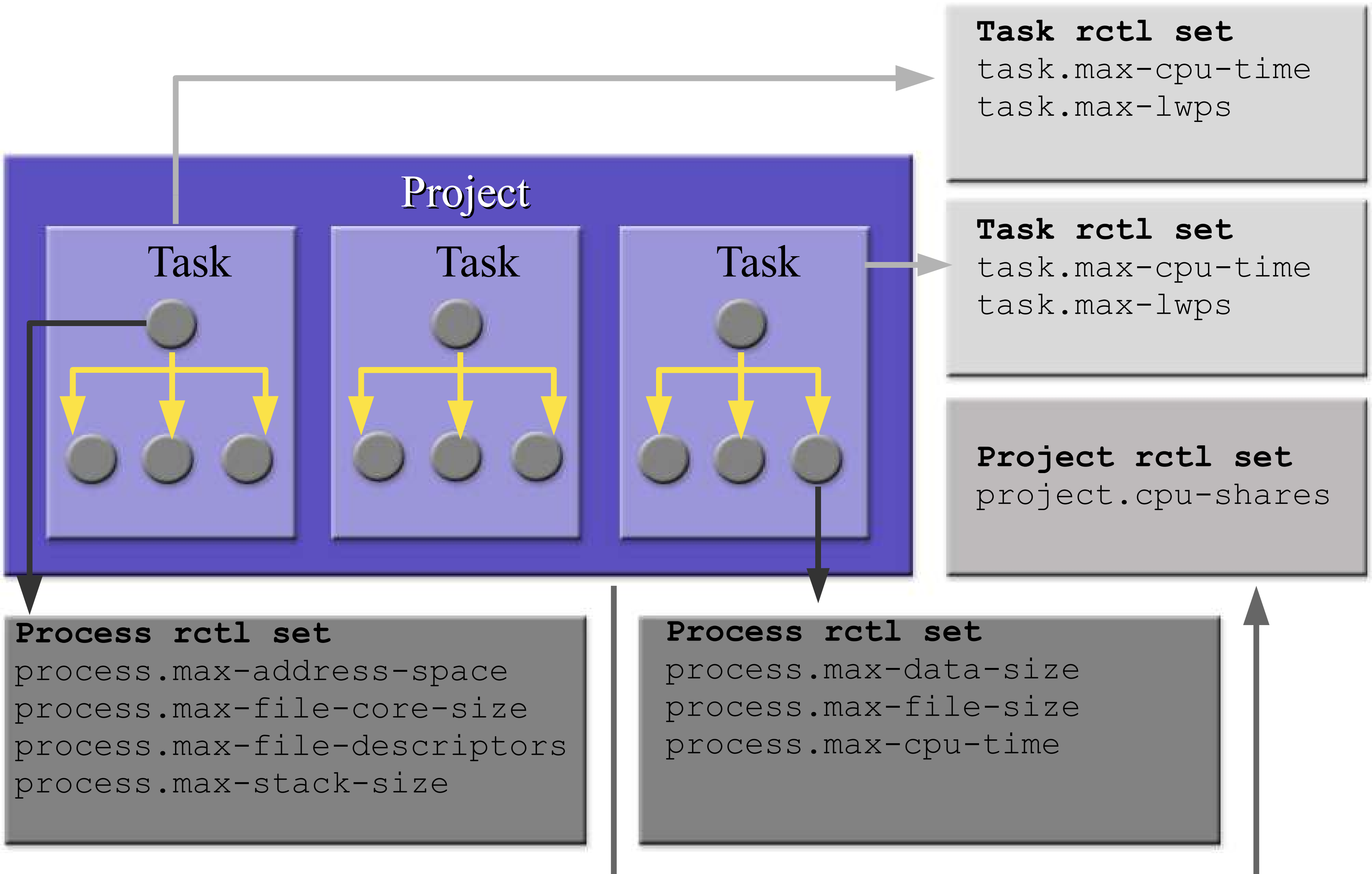
# Fair Share Scheduler

- Allocate CPU ‘shares’ on a per-project basis
- ‘Shares’ define relative importance between projects
- Provide a fine grained mechanism for controlling CPU usage within a pool

# Resource Controls

- Extension of classic rlimits
- Set explicit resource limits on a per-process, per-task, or per-project basis
- Possible actions
  - Send a signal (e.g., SIGTERM, SIGKILL, SIGSTOP, etc.) when a threshold is reached (any user)
  - Deny resource request when the threshold is exceeded (root only)
- Configured through project database

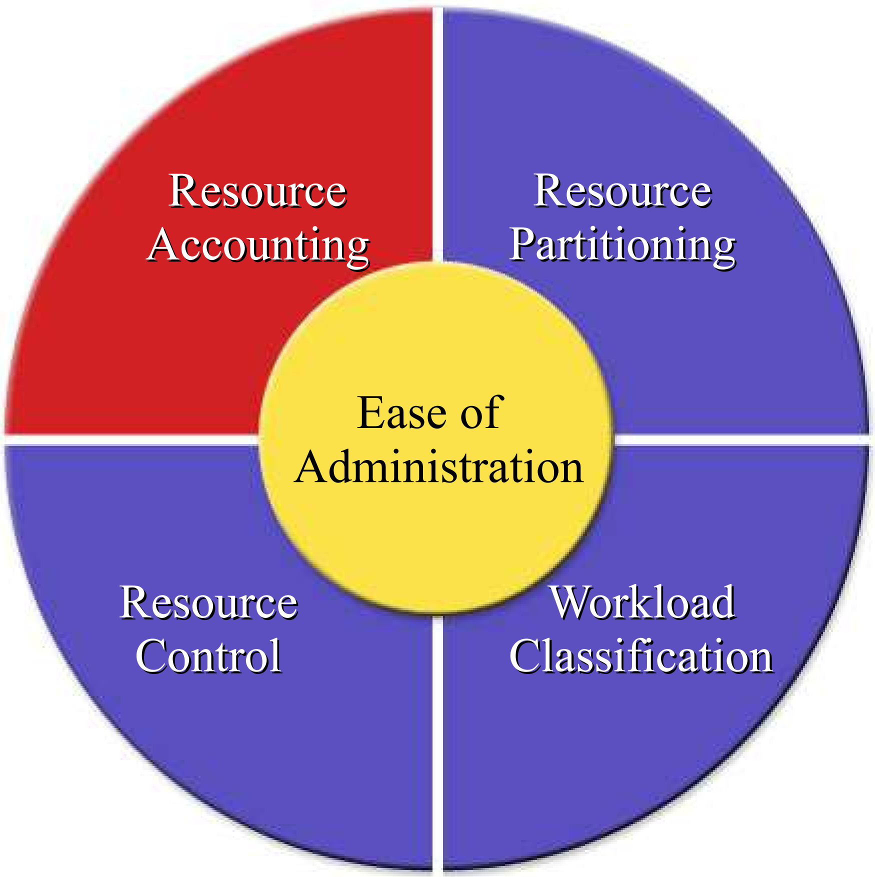
# Resource Controls



# IPQOS

- Control and measure network bandwidth
- Flexible Policies
  - Per host, port
  - Project
  - Per content – e.g. Per URL
- Integrated with accounting
- Delivered in Solaris 9 Update

# Resource Accounting




- Monitor and track resource usage
- Get a snapshot of system activity

# Workload Performance Tools

- Integration with Solaris statistical tools
- Generate statistics on processes, tasks, and projects
  - E.g., prstat, ps, pgrep
- Get a snapshot of system/workload activity for health monitoring and capacity planning purposes

# Using ps with workloads...

```
# ps -ae -o pid,user,taskid,project,comm
  PID      USER TASKID  PROJECT COMMAND
    0      root      0   system sched
    1      root      1   system /etc/init
    2      root      0   system pageout
    3      root      0   system fsflush
  443      root      1   system /usr/lib/saf/sac
  344      root      1   system /usr/lib/utmpd
  199      root      1   system /usr/lib/netsvc/yp/ypbind
   54      root      1   system /
usr/lib/sysevent/syseventd
  250    daemon      1   system /usr/lib/nfs/statd
  251      root      1   system /usr/lib/nfs/lockd
  337     smmsp      1   system /usr/lib/sendmail
 1652      root      8 user.root remotedprovider
  327    oracle      1 database dbwr
...
```

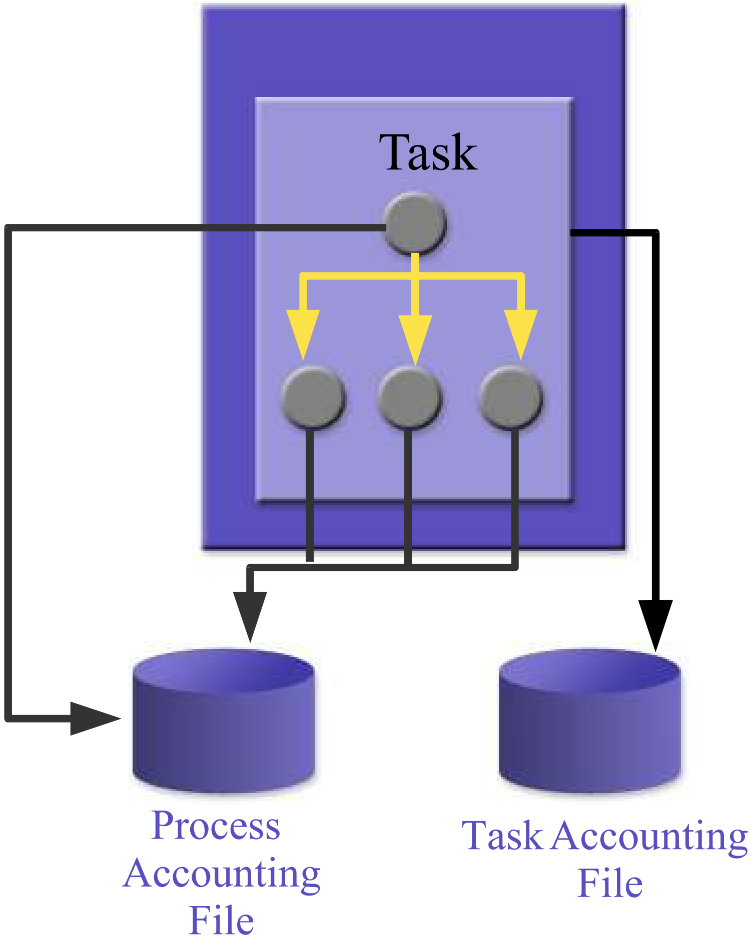
# Using prstat-J with workloads

```
  PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 4793 root     5800K 4920K cpu0     1    0   0:00:00 0.1% prstat/1
 4751 root     5800K 4920K sleep    1    0   0:00:02 0.0% prstat/1
 4779 joostp     80M   54M sleep   34    0   0:00:15 0.0% java/26
  463 root     2488K 2064K sleep   59    0   0:04:23 0.0% mibiisa/7
  282 root     3120K 2592K sleep   59    0   0:00:01 0.0% nscd/20
  413 root     2176K 1504K sleep   56    0   0:00:00 0.0% nfsd/2
  410 root     2608K 1824K sleep   58    0   0:00:00 0.0% mountd/1
 1748 root      122M 6976K sleep   59    0   0:00:00 0.0% Xsun/1
  434 root     3544K 2392K sleep   59    0   0:00:00 0.0% snmpXdmid/2
  362 root     1352K  856K sleep   59    0   0:00:00 0.0% afbdaemon/1
  254 root     4104K 2504K sleep    1    0   0:00:03 0.0% automountd/3
  189 root     2416K 1312K sleep   21    0   0:00:00 0.0% keyserv/3
  185 root     2192K 1344K sleep   59    0   0:00:00 0.0% rpcbind/1
   61 root     2864K 2040K sleep   29    0   0:00:00 0.0% picld/4
   54 root     2296K 1440K sleep   29    0   0:00:00 0.0% syseventd/13
PROJID     NPROC  SIZE   RSS MEMORY      TIME  CPU PROJECT
     1         7   94M   61M   6.2%   0:17:18 0.1% database
 29773         4   86M   59M   6.0%   0:00:15 0.0% appserver
     0        39  226M   72M   7.3%   0:04:27 0.0% system
```

# Extended Accounting

- Provides a more flexible and extensible way of gathering process and task accounting data
- Aggregate process and task statistics to get project accounting data
- Accounting information is available through public APIs

# Extended Accounting

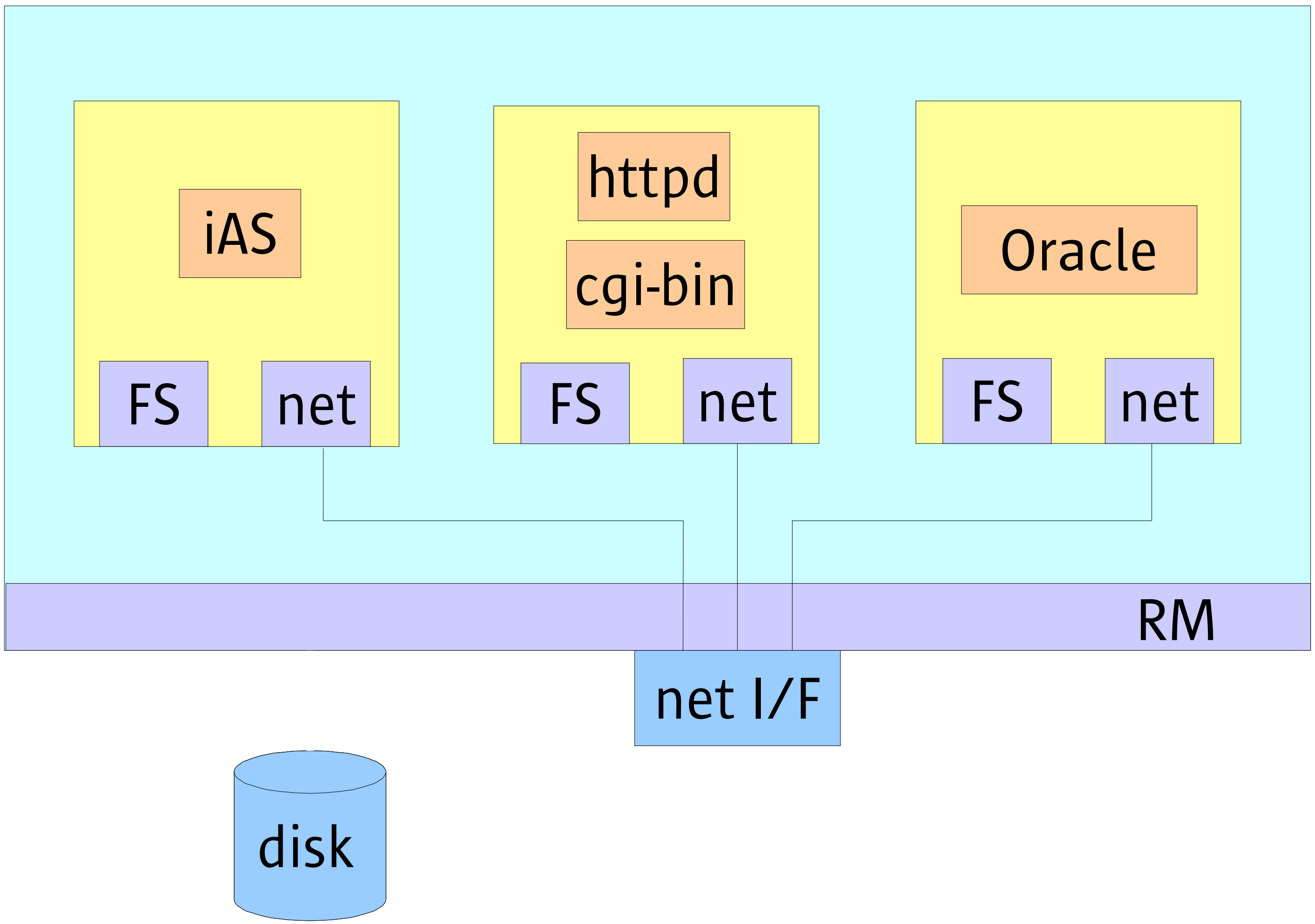



- Process record is written when a process exits
- Task record is written when last process exits task
- Intermediate process and task statistics can be forced

# Enabling Server Consolidation

- Full Resource Containment and control
  - Provide predictable service levels
- Security and Administrative isolation
  - Prevent unauthorized access
- Fault isolation
  - Minimize fault propagation and unplanned downtime

# Solaris Futures: Zones




Virtualize OS services

Secure boundaries

Namespace control

Network isolation

Application fault containment

# Zone Configuration

```
# zonecfg -z nebbiolo-z1
zonecfg> import SUNWdefault
zonecfg> add rootpath /aux0/my-zone
zonecfg> add net myhme
zonecfg> setprop net myhme physical hme0
zonecfg> setprop net myhme address 129.146.126.203
zonecfg> verify
zonecfg> commit
zonecfg> ^D

# zoneadm -v info
ZID ZONENAME STATE ROOT
0 global running /
100 nebbiolo-z1 configured /aux0/my-zone
```

# Solaris Futures: Zones

- Virtualize OS layer: file system, devices, network
- Secure boundary around virtualized instance
- Provides:
  - Privacy: can't see outside zone
  - Security: can't affect activity outside zone
  - Failure isolation: application failure in one zone doesn't affect others
- Minimal (if any) performance overhead
- Resource controls provided by Solaris RM

# Zones: Security

- Root can't be trusted
  - Most operations requiring root disabled
  - Exceptions: file operations, set[ug]id, other "local" operations
- Processes within zone only see/control other processes within zone
- May want to allow specific additional privileges
  - Zone in separate processor set can call priocntl

# Zones: File Systems

- Each zone allocated part of file system hierarchy
- One zone can't see another zone's data
- Loopback mounts allow sharing of read-only data (e.g., /usr)
- Can't escape (unlike chroot)

# Zones: Networking

- Assign set of IP addresses to each zone
- Processes can't bind to addresses not assigned to their zone
  - INADDR_ANY mapped to local set
- Allows multiple services binding to same port in different zones
- TBD: availability of snoop, etc. within zone

# Zones: Devices

- Primarily logical (pseudo) devices within zone
  - Access storage through file system
  - /dev/null, /dev/zero, /dev/random, etc. all safe
  - /dev/ip, /dev/tcp need to be "virtualized"
- Could partition physical devices (e.g. tape drives)
  - But be careful of shared HW (adapters, buses, etc.)
- Some pseudo devices also a problem
  - /dev/cpc, /dev/kmem, ...

# Zones: Name Service

- Can be completely localized
  - multiple copies of nscd, etc.
  - needed to support different administrative domains, ensure data is kept private
    - "Give customers their own root password"
  - User ids have different meanings in different zones
- Also can be global
  - each zone uses same network name service

# Zones and Resource Management

- Complementary technologies
- Zone & RM boundaries can be matched
- Other configurations possible
  - n zones     1 pool
- Per-zone limits

# Enabling Server Consolidation

- Full Resource Containment and control
  - Provide predictable service levels
- Security and Administrative isolation
  - Prevent unauthorized access
- Fault isolation
  - Minimize fault propagation and unplanned downtime

# Summary

- Solaris continues to evolve in both performance and resource management innovations
- Observability tools and utilities continue to get better
- Resource management facilities providing for improved overall system utilization and SLA management

# Resources

- **http://www.solarisinternals.com**
- **http://www.sun.com/solaris**
- **http://www.sun.com/blueprints**
- **http://www.sun.com/bigadmin**
- **http://docs.sun.com**
  - "What's New in the Solaris 9 Operating Environment"
- **http://sdc.sun.com/solaris8**
- **http://sun.com/solaris/fcc/lifecycle.html**

# Thank You!

# Questions?