Solaris 10 & OpenSolaris Performance, Observability & Debugging (POD)

Jim Mauro
Principal Engineer
Sun Microsystems, Inc
james.mauro@sun.com

Significant Material/Content contributions:
Richard McDougall
Chief Performance Architect
VMware
rmc@vmware.com
About The Authors

Jim Mauro is a Principle Engineer in the Systems Group Quality Office at Sun Microsystems, where he focuses on systems performance with real customer workloads. Jim also dabbles in performance for ZFS and Virtualization.

Richard McDougall is the Chief Performance Architect at VMware.


james.mauro@sun.com
rmc@vmware.com
The Books (Shameless Plug)

- Solaris Internals: Solaris 10 and OpenSolaris Kernel Architecture
  - Community effort: over 35 contributing authors
  - Kernel data structures and algorithms
  - A lot of DTrace and mdb(1) examples to support the text
- Solaris Performance and Tools: DTrace and MDB Techniques for Solaris 10 and OpenSolaris
  - Guide to using the tools and utilities, methods, examples, etc
Coming Soon!

DTrace

DYNAMIC TRACING IN SOLARIS, MAC OS X AND FREEBSD

Jim Mauro, Brendan Gregg, Chad Mynhier, Tariq Magdon-Ismail
Before We Begin...

IT DEPENDS

What was the question...?

Batteries Not Included

Your Mileage May Vary (YMMV)
Solaris Performance

- **Resources**
  - [www.solarisinternals.com](http://www.solarisinternals.com)
    - Wikipedia of Solaris Performance
  - [www.opensolaris.org](http://www.opensolaris.org) / [www.opensolaris.com](http://www.opensolaris.com)
    - Downloads, communities, documentation, discussion groups
      - DTrace, ZFS, Performance, etc
    - Technical articles, tips, etc
  - [www.brendangregg.com](http://www.brendangregg.com)
    - DTrace Toolkit (over 230 scripts!)
    - Other Goodies
  - [www.cooltools.sunsorce.net](http://www.cooltools.sunsorce.net)
    - Development, optimized opensource software, analysis
  - [blogs.sun.com](http://blogs.sun.com)
    - Too much to summarize here – search for what you're interested in
  - [sunsolve.sun.com](http://sunsolve.sun.com)
    - Search for bugs related to what you're chasing...
Agenda

- **Session 1 - 9:00AM to 10:30PM**
  - Goals, non goals and assumptions
  - OpenSolaris
  - Solaris 10 Kernel Overview
  - Solaris 10 Features
  - The Tools of the Trade
- **Session 2 - 11:00PM to 12:30PM**
  - Memory
    - Virtual Memory
    - Physical Memory
    - Memory dynamics
    - Performance and Observability
    - Memory Resource Management
Agenda

- **Session 3 - 2:00PM to 3:30PM**
  - Processes, Threads & Scheduling
    - Processes, Threads, Priorities & Scheduling
    - Performance & Observability
      - Load, apps & the kernel
    - Processor Controls and Binding
    - Resource Pools, Projects & Zones

- **Session 4 - 4:00PM to 5:30PM**
  - File Systems and I/O
    - I/O Overview
    - UFS
    - ZFS
    - Performance & Observability
  - Network & Miscellanea
Session 1 - Intro, Tools, Stuff
Goals, Non-goals & Assumptions

- **Goals**
  - Architectural overview of the Solaris kernel
  - 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
OpenSolaris - www.opensolaris.com

- An open source operating system providing for community collaboration and development
- Source code released under the Common Development & Distribution License (CDDL – pronounced “cuddle”)
- Based on “Nevada” Solaris code-base (Solaris 10+)
- New features added to OpenSolaris, then back-ported to Solaris 10
- **OpenSolaris 2008.05**
  - First supported OpenSolaris distro with many new features
    - Live CD and easy-to-use graphical installer
    - ZFS default for root
    - Network-based package management (IPS)
    - Lots of apps
- **OpenSolaris 2009.06 – current release**
  - 2010.03 next planned release (subject to change)
Solaris 10 – Update Releases

- New features, new hardware support bug fixes
  - Check out the “What's New” Document; http://docs.sun.com/app/docs/coll/1531.1?l=en
- Solaris 10 3/05 – First release of S10
- Solaris 10 1/06 – Update 1
- Solaris 10 6/06 – Update 2
- Solaris 10 11/06 – Update 3
- Solaris 10 8/07 – Update 4
- Solaris 10 5/08 – Update 5
- Solaris 10 10/08 – Update 6
- Solaris 10 5/09 – Update 7
- Solaris 10 10/09 – Update 8
Solaris Kernel Features

- Dynamic
- Multithreaded
- Preemptive
- Multithreaded Process Model
- Multiple Scheduling Classes
  - Including realtime support, fixed priority and fair-share scheduling
- 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
Solaris 10 & OpenSolaris

The Headline Grabbers

- Solaris Containers (Zones)
- Solaris Dynamic Tracing (DTrace)
- Predictive Self Healing
  - System Management Framework (SMF)
  - Fault Management Architecture (FMA)
- Process Rights Management (aka Least Privilege)
- Premier x86 support
  - Optimized 64-bit Opteron support (x64)
  - Optimized Intel support
- Zetabyte Filesystem (ZFS)
- Network Stack – TCP/IP, UDP/IP, GLDv3

... and much, much more!
Solaris Performance Optimizations

- Threads, threads, threads...
- FULLY 64-Bit
- CPU Scalability
- UFS, NFS & ZFS File System Performance
- JVM Optimizations
- Memory Scalability
- Networking
- Resource Management
- Compiler Technology
- Observability: DTrace, Analyzer, and Perf Tools
Threads, threads, threads....

- Solaris's big-bet: large numbers of CPUs
  - SMP systems in the early days
  - Now thread-rich multicore CPUs w/hardware strands
    - 256-way T5440 (4 N2 sockets, each 8x8), 256GB RAM
    - 512-way DC3-64 (64 sockets, 256 cores, 512 strands)
- Fully preemptable kernel
- Architected around threads:
  - Kernel threads scheduled, executed
- I/O is thread-rich:
  - (Blocking) Interrupts are handled as threads
  - Worker-pools (tasks-queues) of threads avail for driving I/O
- Rich thread-development environment
The 64-bit Revolution

Solaris 2.6
- ILP32 Apps
- ILP32 Libs
- ILP32 Kernel
- ILP32 Drivers
- 32-bit HW

Solaris 7, 8, 9, 10, OpenSolaris
- ILP32 Apps
- ILP32 Libs
- ILP32 Kernel
- ILP32 Drivers
- 32-bit HW

- LP64 Apps
- LP64 Libs
- LP64 Kernel
- LP64 Drivers
- 64-bit HW (SPARC, X64)
CPU Scalability

• Per-cpu dispatcher queues
  • Fine-grained locking on thread enqueue/dequeue

• Slab / vmem allocator in the kernel
  • Adopted by most other major kernels
  • Ported to user-land – libumem.so (scalable malloc(3C))

• CMP & CMT Optimizations
  • Chip MultiProcessing/Chip MultiThreading
    • Multi-strand, Multi-core designs
    • Optimize thread placement on cores

• NUMA Optimizations (MPO)
  • Locality groups (CPUs and Memory)
Scheduler Enhancements

- **FX** – Fixed Priority Scheduler
  - Integrated into Solaris 9
  - Provides fixed quantum scheduling
  - Fixed priorities
  - Eliminates unnecessary context switches for server-style apps
  - Recommend setting as the default for Databases/Oracle

- **FSS** – Fair Share Schedule
  - Integrated into Solaris 9
  - Replaces SRM 1.X
  - Shares of CPU allocated
  - Adds Projects and Tasks for administration / management
File System Performance

- **UFS & Databases**
  - Direct I/O enables scalable database performance
  - Enhanced Logging Support introduced in S9

- **NFS**
  - Fireengine + RPC optimizations provide high throughput:
    - 108MB/s on GBE, 910MB/s on 10GBE, Solaris 10, x64
  - NFS for Databases Optimizations
    - 50,000+ Database I/O's per second via Direct I/O

- **ZFS**
  - Adaptive Replacement Cache (ARC)
  - Dynamic space management for metadata and data
  - Copy-On-Write (COW) – in-place data is never overwritten
  - Still evolving - new features and performance enhancements
Java VM Performance

- Java SE 6
  - Lock enhancements
  - GC improvements
  - A lot more;

- DTrace & Java
  - jstack() (Java 5)
    - jstackstrsize for more buffer space
  - dvm provider
    - Java 1.4.2 (libdvmpi.so)
    - Java 1.5 (libdvmti.so)
    - https://solaris10-dtrace-vm-agents.dev.java..net
  - Integrated HotSpot provider in Java 6
    - All DVM probes, plus extensions
    - Additional DTrace probes coming in Java 7
Memory Scalability

- Large Memory Optimizations
  - Solaris 9 & 10
  - 1TB shipping today. 4TB coming soon
  - 64GB hardly considered large anymore...

- Large Page Support
  - Evolved since Solaris 2.6
    - Large (4MB) pages with ISM/DISM for shared memory
  - Solaris 9/10
    - Multiple Page Size Support (MPSS)
      - Optional large pages for heap/stack
      - Programmatically via madvise()
      - Shared library for existing binaries (LD_PRELOAD)
      - Tool to observe potential gains
        - # trapstat -t

- Solaris 10 Updates and OpenSolaris
  - Large Pages Out-Of-The-Box (LPOOB)
Networking

- **Fire-engine in Solaris 10**
  - New “vertical perimeter” scaling model
  - 9Gbits/s on 10GBE, @50% of a 2-way x64 system
- **Application to application round trip latency close to 40usec**
- **Nemo: High performance drivers in Solaris 1 Update 2**
  - GLDv3 NIC Driver Interface
  - Enables multiple-ring support
  - Generic VLAN and Trunking Support
- **Yosemite: High performance UDP**
  - Enabled in Solaris 10 Update 2
- **IP Instances**
  - Unique per-Zone TCP/IP stack
- **Crossbow**
  - Virtualization – VNICS
  - Resource management
Resource Management

- CPU & Network Resource Control Framework
  - Often used to control resource allocation to a container/zone.
- Processor Sets/Pools
  - Partition the machine into sets of processor resources
- Fair-Share Scheduler

App A (3 shares)  App C (2 shares)  App D (5 shares)

App B (5 shares)
Compiler Technology

• Studio Compilers performance optimizations
  • Highest level of optimized code for SPARC & x64
  • Studio 11 and Studio 12 available free
    • Outstanding profiler/performance analysis tool
    • collect(1) for collecting experiments
    • analyzer GUI, er_print(1) for command line
  • sgcc – gcc front-end, with an optimized SPARC code generator
    • Use cc(1) or gcc(1) to build apps
  • Analyzer Tool
Observability

- Performance Observability is the primary means to optimization
  - “That which is not measured, gets worse”
- Key Observability Features:
  - DTrace
  - Thread Microstate Measurements
  - *stat tools, ptools, kstats, truss, prstat
  - Lock Analysis Tools: lockstat, plockstat
  - System Statistics: sar, mpstat, vmstat etc...
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 (Disk and Network)
    - 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...
“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/utilization
    - 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
The Utilization Conundrum

• What is utilization?
  • The most popular metric on the planet for determining if something on your system is potentially a bottleneck or out of capacity
  • Properly defined as the amount of time something is busy relative to wall clock (elapsed) time
    • N is busy for .3 seconds over 1 second sampling periods, it's \((0.30 / 1) \times 100\) 30% utilized

• So... What's the problem?
  • Basic utilization metrics assume simple devices capable of only doing 1 thing at a time
    • Old disks, old networks (NICs), old CPUs
  • Bottom Line – 100% utilized is NOT necessarily a pain point
The Utilization Conundrum (cont)

• **Modern Times**
  • Disks, CPUs, NICs are all very sophisticated, with concurrency built-in at the lowest levels
    • Disks – integrated controllers with deep queues and NCQ
    • NICs – multiple ports, multiple IO channels per port

• **Case in point, iostat “%b”**
  • We've been ignoring it for years – it's meaningless because it simply means that an IO thread is in the disks queue every time it looks
  • “100% busy” Disks, or NICs, may be able to do more work with acceptable latency
  • It's all about meeting business requirements
The Utilization Conundrum (cont)

- CPUs
  - Multicore. Multiple execution cores on a chip
  - Multithread (hyperthreads) – multiple threads-per-core
  - CMT – Chip Multithreading
    - Combining multicore and multithread.
- CPU Utilization
  - Each thread (or strand) appears as a CPU to Solaris
  - Each CPU maintains its own set of utilization metrics
    - Derived from CPU microstates – sys, usr, idle
    - Multiple threads sharing the same core can each appear 100% utilized
  - A CPU that shows 100% utilization (0% idle) has about as much meaning as a disk or NIC that shows 100% utilization
    - More to the point, a CPU that is observed to be 100% utilized may be capable of doing more work without a tradeoff in latency
      - e.g. a multi-execution unit pipeline running 1 thread all the time is 100% utilized, but capable of running another thread while maintaining the same service level

Google “Utilization is Virtually Useless as a Metric”
CPU Utilization

• Traditional “stat” tools
  • threads are CPUs
  • CPU microstates

• corestat
  • Unbundled script that uses cpustat(1)
    • cpustat(1) programs hardware counters (PICs) to gather chip statistics
      • Very hardware-specific

• corestat reports and vmstat/mpstat reports may vary due to the very different methods of data gathering
### vmstat

<table>
<thead>
<tr>
<th>kthr</th>
<th>memory</th>
<th>page</th>
<th>disk</th>
<th>faults</th>
<th>cpu</th>
</tr>
</thead>
<tbody>
<tr>
<td>r b w</td>
<td>swap</td>
<td>free</td>
<td>re</td>
<td>mf</td>
<td>pi</td>
</tr>
<tr>
<td>0 11 0</td>
<td>83283928</td>
<td>30844904</td>
<td>10682</td>
<td>15143</td>
<td>42254</td>
</tr>
<tr>
<td>0 16 0</td>
<td>83250248</td>
<td>30829608</td>
<td>9300</td>
<td>13420</td>
<td>27344</td>
</tr>
<tr>
<td>0 11 0</td>
<td>83237872</td>
<td>30848544</td>
<td>10700</td>
<td>13440</td>
<td>28631</td>
</tr>
<tr>
<td>0 13 0</td>
<td>83248120</td>
<td>30848272</td>
<td>11433</td>
<td>14640</td>
<td>21671</td>
</tr>
<tr>
<td>0 21 0</td>
<td>83207744</td>
<td>30848368</td>
<td>10944</td>
<td>14640</td>
<td>21671</td>
</tr>
<tr>
<td>0 11 0</td>
<td>83231536</td>
<td>30861544</td>
<td>10106</td>
<td>15260</td>
<td>90549</td>
</tr>
<tr>
<td>0 14 0</td>
<td>83241696</td>
<td>30866376</td>
<td>7904</td>
<td>12898</td>
<td>27841</td>
</tr>
<tr>
<td>0 15 0</td>
<td>83257208</td>
<td>30888504</td>
<td>9293</td>
<td>15014</td>
<td>28643</td>
</tr>
<tr>
<td>0 12 0</td>
<td>83219888</td>
<td>30894256</td>
<td>8621</td>
<td>16163</td>
<td>28433</td>
</tr>
</tbody>
</table>

### CPU Utilization

<table>
<thead>
<tr>
<th>PCPU</th>
<th>%Usr</th>
<th>%Sys</th>
<th>%Usr+Sys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>28.56</td>
<td>30.77</td>
<td>59.33</td>
</tr>
<tr>
<td>1</td>
<td>28.59</td>
<td>23.12</td>
<td>51.71</td>
</tr>
<tr>
<td>4</td>
<td>45.72</td>
<td>21.97</td>
<td>67.68</td>
</tr>
<tr>
<td>5</td>
<td>18.43</td>
<td>31.62</td>
<td>50.05</td>
</tr>
<tr>
<td>8</td>
<td>13.51</td>
<td>32.17</td>
<td>45.69</td>
</tr>
<tr>
<td>9</td>
<td>17.99</td>
<td>16.03</td>
<td>34.02</td>
</tr>
<tr>
<td>12</td>
<td>25.04</td>
<td>27.74</td>
<td>52.78</td>
</tr>
<tr>
<td>13</td>
<td>13.48</td>
<td>18.45</td>
<td>31.93</td>
</tr>
<tr>
<td>16</td>
<td>54.45</td>
<td>33.69</td>
<td>88.13</td>
</tr>
<tr>
<td>17</td>
<td>47.48</td>
<td>24.56</td>
<td>72.05</td>
</tr>
<tr>
<td>20</td>
<td>42.69</td>
<td>43.48</td>
<td>86.17</td>
</tr>
<tr>
<td>21</td>
<td>25.89</td>
<td>39.09</td>
<td>64.98</td>
</tr>
<tr>
<td>24</td>
<td>63.57</td>
<td>35.05</td>
<td>98.62</td>
</tr>
<tr>
<td>25</td>
<td>23.27</td>
<td>29.89</td>
<td>53.17</td>
</tr>
<tr>
<td>28</td>
<td>71.71</td>
<td>43.87</td>
<td>100.00</td>
</tr>
<tr>
<td>29</td>
<td>33.65</td>
<td>19.33</td>
<td>52.98</td>
</tr>
<tr>
<td>Avg</td>
<td>34.63</td>
<td>29.43</td>
<td>64.05</td>
</tr>
</tbody>
</table>

### corestat

<table>
<thead>
<tr>
<th>CPU Utilization</th>
<th>Mon May 11 11:00:58 2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (Thd)</td>
<td>%Usr</td>
</tr>
<tr>
<td>0</td>
<td>(0,1)</td>
</tr>
<tr>
<td>1</td>
<td>(2,3)</td>
</tr>
<tr>
<td>4</td>
<td>(8,9)</td>
</tr>
<tr>
<td>5</td>
<td>(10,11)</td>
</tr>
<tr>
<td>8</td>
<td>(16,17)</td>
</tr>
<tr>
<td>9</td>
<td>(18,19)</td>
</tr>
<tr>
<td>12</td>
<td>(24,25)</td>
</tr>
<tr>
<td>13</td>
<td>(26,27)</td>
</tr>
<tr>
<td>16</td>
<td>(32,33)</td>
</tr>
<tr>
<td>17</td>
<td>(34,35)</td>
</tr>
<tr>
<td>20</td>
<td>(40,41)</td>
</tr>
<tr>
<td>21</td>
<td>(42,43)</td>
</tr>
<tr>
<td>24</td>
<td>(48,49)</td>
</tr>
<tr>
<td>25</td>
<td>(50,51)</td>
</tr>
<tr>
<td>28</td>
<td>(56,57)</td>
</tr>
<tr>
<td>29</td>
<td>(58,59)</td>
</tr>
<tr>
<td>Avg</td>
<td>34.63</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Average</th>
<th>vmstat</th>
<th>corestat</th>
</tr>
</thead>
<tbody>
<tr>
<td>%sys</td>
<td>20.53</td>
<td>29.43</td>
</tr>
<tr>
<td>%usr</td>
<td>39.05</td>
<td>34.63</td>
</tr>
<tr>
<td>%idle</td>
<td>40.42</td>
<td>35.95</td>
</tr>
</tbody>
</table>
```
vmstat

kthr memory page disk faults cpu
r b w swap free re mf pi po fr de sr ml ml m2 in sy cs us sy id
0 8 0 83293328 30951456 8724 12802 30608 4000 4000 0 0 0 25 0 25 0 13271 87958 17034 28 14 58
0 6 0 83298144 30966096 9688 10972 30122 25454 25438 0 0 0 47 0 47 0 15283 115241 19714 28 17 55
0 8 0 83307976 30980096 9732 11240 30540 25329 25110 0 0 47 0 47 0 13151 96896 17352 26 13 61
0 10 0 83301456 30967616 8312 9286 29132 255 255 0 0 27 0 27 0 12856 89560 15560 27 20 53
0 12 0 83295096 30956080 8820 8621 29532 14728 14728 0 0 50 0 50 0 13639 123786 16865 32 13 55
0 13 0 83255472 30956744 8936 10414 28178 23920 23920 0 0 32 0 31 0 15620 142711 20481 31 20 49
0 15 0 83215552 30959632 9234 9270 37623 21136 21128 0 0 31 0 31 0 17276 140842 22307 35 20 45
0 14 0 83191296 30951064 9249 12303 40026 185 185 0 0 28 0 28 0 16325 143003 20585 40 16 43
0 13 0 83184352 30937424 8859 9732 37956 1182 1182 0 0 30 0 30 0 15235 151314 17797 34 26 39
0 16 0 83208648 30960648 9249 10079 35849 23026 23026 0 0 29 0 29 0 16332 128670 16865 32 17 51
0 15 0 83185728 30980040 9866 6413 39944 11595 11580 0 0 17 0 17 0 17262 103816 22081 25 14 60
0 17 0 83180464 30968800 11455 6607 115326 2521 2521 0 0 39 0 39 0 13151 96896 17352 26 13 61
0 13 0 83186320 30963096 12146 6460 63788 46 46 0 0 20 0 20 0 15810 82579 20030 25 17 58
0 15 0 83184984 30942064 12559 11172 60716 23 23 0 0 31 0 31 0 15018 82876 17789 24 15 61
0 16 0 83192240 30920472 10457 7314 48095 2990 2990 0 0 39 0 39 0 14787 97524 18930 21 15 64
0 18 0 83190632 30917888 9801 9156 44190 529 529 0 0 30 0 30 0 15252 110036 19890 19 18 63
0 10 0 83178232 30890776 9107 7755 46624 272 272 0 0 23 0 23 0 15567 84655 17958 25 14 61
0 11 0 83197376 30911640 9985 13448 42818 12116 12110 0 0 40 0 40 0 15324 101347 18084 29 23 48
0 9 0 83187200 30916064 9256 12272 30049 13563 13555 0 0 34 0 34 0 12918 99437 15620 26 15 58

avg idle – 54.84%
avg sys - 16.95%
avg usr - 28.00%

CPU Utilization Mon May 11 11:02:58 2009
CPU (Thd) %Usr %Sys %Usr+Sys
0 (0,1) 22.85 19.04 41.89
1 (2,3) 8.76 15.82 24.58
4 (8,9) 6.68 24.22 30.91
5 (10,11) 88.10 12.89 100.00
8 (16,17) 61.92 20.17 82.10
9 (18,19) 4.17 18.45 22.61
12 (24,25) 60.86 14.97 75.84
13 (26,27) 2.79 14.66 17.45
16 (32,33) 62.64 17.38 80.02
17 (34,35) 11.88 18.07 29.95
20 (40,41) 24.16 29.25 53.41
21 (42,43) 19.97 21.51 41.48
24 (48,49) 38.06 26.48 64.54
25 (50,51) 12.46 31.90 44.36
28 (56,57) 40.40 26.78 67.18
29 (58,59) 33.50 35.24 68.75
Avg 31.20 21.68 52.88

vmstat | corestat
%sys | 16.95 | 21.68
%usr | 28.00 | 31.20
%idle | 54.84 | 47.12

LISA ’09 Baltimore, Md.
```
CPU Utilization/Capacity

- vmstat/mpstat and corestat will vary depending on the load
  - corestat will generally be more accurate
- Use “prstat -m” LAT category, in conjunction with utilization measurements, delivered workload throughput and run queue depth (vmstat “r” column) to determine for CPU capacity planning
The Workload Stack

- All stack layers are observable
Little's Law

- A relatively simple queueing theory theorem that relates response time to throughput
  - The throughput of a system (Q) is a factor of the rate of incoming work (N), and the average amount of time required to complete the work (R – response time)
  - Independent of any underlying probability distribution for the arrival of work or the performance of work

\[
\text{throughput} = \frac{\text{arrival rate}}{\text{avg processing time}} \quad \text{or} \quad Q = \frac{N}{R}
\]

E.g.

if \( N = 100 \) and \( R = 1 \) second, \( Q = 100 \) TPS

More compelling, it makes it easy to see how these critical performance metrics relate to each other....
Amdahl's Law

- In general terms, defines the expected speedup of a system when part of the system is improved.
- As applied to multiprocessor systems, describes the expected speedup when a unit of work is parallelized.
- Factors in degree of parallelization.

\[ S = \frac{1}{F + \left(1 - F\right) \frac{N}{N}} \]

- \( S \) is the speedup.
- \( F \) is the fraction of the work that is serialized.
- \( N \) is the number of processors.

\[ S = \frac{1}{(0.5 + \frac{(1 - 0.5)}{4})} \quad S = 1.6 \quad 4 \text{ processors, } \frac{1}{2} \text{ of the work is serialized.} \]

\[ S = \frac{1}{(0.25 + \frac{(1 - 0.25)}{4})} \quad S = 2.3 \quad 4 \text{ processors, } \frac{1}{4} \text{ of the work is serialized.} \]
Performance & Observability Tools
Solaris Performance and Tracing Tools

**Process stats**
- `cputrack / cpustat` - processor hw counters
- `plockstat` – process locks
- `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 Tracing/debugging**
- `abitrace` – trace ABI interfaces
- `dtrace` – trace the world
- `mdb` – debug/control processes
- `truss` – trace functions and system calls

**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

**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

**Kernel Tracing/debugging**
- `dtrace` – trace and monitor kernel
- `lockstat` – monitor locking statistics
- `lockstat -k` – profile kernel
- `mdb` – debug live and kernel cores

*why did Harry Cooper & Ben wish they had preap?
Solaris 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

“With DTrace, I can walk into a room of hardened technologists and get them giggling”
- Bryan Cantrill, Inventor of DTrace
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
What is DTrace

• DTrace is a dynamic troubleshooting and analysis tool first introduced in the Solaris 10 and OpenSolaris operating systems.

• DTrace is many things, in particular:
  • A tool
  • A programming language interpreter
  • An instrumentation framework

• DTrace provides observability across the entire software stack from one tool. This allows you to examine software execution like never before.
The Entire Software Stack

- How did you analyze these?

Examples:

- Dynamic Languages: Java, JavaScript, ...
- User Executable: native code, /usr/bin/*
- Libraries: /usr/lib/*
- SySCALL Interface: man -s2
- Kernel: VFS, DNLC, UFS, ZFS, TCP, IP, ...
- Device Drivers: sd, st, hme, eri, ...
- Memory allocation
- File Systems: NIC, Disk HBA, Processors, etc
The Entire Software Stack

- It was possible, but difficult.

Previously:
- debuggers
- truss -ua.out
- apptrace, sotruss
- truss
- prex; tnf*
- lockstat
- mdb
- kstat, PICs, guesswork
The Entire Software Stack

- DTrace is all seeing:

<table>
<thead>
<tr>
<th>Dynamic Languages</th>
<th>User Executable</th>
<th>Libraries</th>
<th>Syscall Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No. Indirectly, yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory allocation</td>
</tr>
<tr>
<td>Yes</td>
</tr>
</tbody>
</table>

  Hardware

<table>
<thead>
<tr>
<th>DTrace visibility:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes, with providers</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>No. Indirectly, yes</td>
</tr>
</tbody>
</table>
What DTrace is like

- DTrace has the combined capabilities of numerous previous tools and more,

<table>
<thead>
<tr>
<th>Tool</th>
<th>Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>truss -u a.out</td>
<td>tracing user functions</td>
</tr>
<tr>
<td>apptrace</td>
<td>tracing library calls</td>
</tr>
<tr>
<td>truss</td>
<td>tracing system calls</td>
</tr>
<tr>
<td>prex; tnf*</td>
<td>tracing some kernel functions</td>
</tr>
<tr>
<td>lockstat</td>
<td>profiling the kernel</td>
</tr>
<tr>
<td>mdb -k</td>
<td>accessing kernel VM</td>
</tr>
<tr>
<td>mdb -p</td>
<td>accessing process VM</td>
</tr>
</tbody>
</table>

Plus a programming language similar to C and awk.
Syscall Example

- Using truss,

```
$ truss date
execve("/usr/bin/date", 0x08047C9C, 0x08047CA4) argc = 1
resolvepath("/usr/lib/ld.so.1", "/lib/ld.so.1", 1023) = 12
resolvepath("/usr/bin/date", "/usr/bin/date", 1023) = 13
xstat(2, "/usr/bin/date", 0x08047A58) = 0
open("/var/ld/ld.config", 0_RDONLY) = 3
fxstat(2, 3, 0x08047988) = 0
mmap(0x00000000, 152, PROT_READ, MAP_SHARED, 3, 0) = 0xFEFB0000
close(3) = 0
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_ANON, -1
sysconfig(_CONFIG_PAGESIZE) = 4096
```

...}

Only examine 1 process

Output is limited to provided options

truss slows down the target
Syscall Example

• Using DTrace,

```
# dtrace -n 'syscall::entry { printf("%16s %x %x", execname, arg0, arg1); }'
dtrace: description 'syscall::entry' matched 233 probes

CPU     ID                    FUNCTION:NAME
1  75943                       read:entry             Xorg f 8047130
1  76211                  setitimer:entry             Xorg 0 8047610
1  76143                     writev:entry             Xorg 22 80477f8
1  76255                    pollsys:entry             Xorg 8046da0 1a
1  75943                       read:entry             Xorg 22 85121b0
1  76035                      ioctl:entry      soffice.bin 6 5301
1  76035                      ioctl:entry      soffice.bin 6 5301
1  76255                    pollsys:entry      soffice.bin 8047530 2

[...]```

You choose the output

Minimum performance cost

Watch every process
What is DTrace for

- Troubleshooting software bugs
  - Proving what the problem is, and isn't.
  - Measuring the magnitude of the problem.
- Detailed observability
  - Observing devices, such as disk or network activity.
  - Observing applications, whether they are from Sun, 3rd party, or in-house.
- Capturing profiling data for performance analysis
  - If there is latency somewhere, DTrace can find it.
Terminology

- Example #1

```plaintext
# dtrace -n 'syscall::exece:return { trace(execname); }'
dtrace: description 'syscall::exece:return' matched 1 probe

<table>
<thead>
<tr>
<th>CPU</th>
<th>ID</th>
<th>FUNCTION</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>man</td>
</tr>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>sh</td>
</tr>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>neqn</td>
</tr>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>tbl</td>
</tr>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>nroff</td>
</tr>
</tbody>
</table>
```

[...]

LISA '09 Baltimore, Md.
Consumer

- Consumers of libdtrace(3LIB),
  
  dtrace command line and scripting interface
  lockstat kernel lock statistics
  plockstat user-level lock statistics
  intrstat run-time interrupt statistics

- libdtrace is currently a private interface and not to be used directly (nor is there any great reason to); the supported interface is dtrace(1M).
  
  • NOTE: You are still encouraged to use libkstat(3LIB) and proc(4) directly, rather than wrapping /usr/bin consumers.
Privileges

Non-root users need certain DTrace privileges to be able to use DTrace.

These privileges are from the Solaris 10 “Least Privilege” feature.

```
$ id
uid=1001(user1) gid=1(other)
$ /usr/sbin/dtrace -n 'syscall::exece:return'
dtrace: failed to initialize dtrace: DTrace requires additional privileges
```

root:::auths=solaris.*,solaris.grant;profiles=Web Console Management,All;lock_after_retries=no
mauroj:::defaultpriv=basic,dtrace_user,dtrace_proc,dtrace_kernel,proc_priocntl,
proc_clock_highres;project=laptop
Probes

- Data is generated from instrumentation points called “probes”.
- DTrace provides thousands of probes.
- Probe examples:

<table>
<thead>
<tr>
<th>Probe Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>syscall::read:entry</td>
<td>A read() syscall began</td>
</tr>
<tr>
<td>proc:::exec-success</td>
<td>A process created successfully</td>
</tr>
<tr>
<td>io:::start</td>
<td>An I/O was issued (disk)</td>
</tr>
<tr>
<td>io:::done</td>
<td>An I/O completed</td>
</tr>
</tbody>
</table>
Probe Names

- Probe names are a four-tuple,

  **Provider**  **Module**  **Function**  **Name**

  `syscall::exece:return`

- **Provider**  A library of related probes.
- **Module**  The module the function belongs to, either a kernel module or user segment.
- **Function**  The function name that contains the probe.
- **Name**  The name of the probe.
Listing Probes

- **dtrace -l** lists all currently available probes that you have privilege to see, with one probe per line.
- Here the root user sees 69,879 available probes.
- The probe count changes – it is dynamic (DTrace).

```
# dtrace -l

ID   PROVIDER  MODULE   FUNCTION NAME
 1    dtrace    dtrace   BEGIN
 2    dtrace    dtrace   END
 3    dtrace    dtrace   ERROR
 4    sched    FX       fx_yield schedctl-yi

[...]

# dtrace -l | wc -l
69880
```
Tracing Probes

- `dtrace -n` takes a probe name and enables tracing,

```bash
# dtrace -n syscall::exece:return
dtrace: description 'syscall::exece:return' matched 1 probe
CPU    ID        FUNCTION:NAME
0  76044      exece:return
0  76044      exece:return
^C
```

- The default output contains,
  - **CPU** CPU id that event occurred on (if this changes, the output may be shuffled)
  - **ID** DTrace numeric probe id
  - **FUNCTION:NAME** Part of the probe name
## Providers

- Examples of providers,

<table>
<thead>
<tr>
<th>Provider</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>syscall</td>
<td>system call entries and returns</td>
</tr>
<tr>
<td>proc</td>
<td>process and thread events</td>
</tr>
<tr>
<td>sched</td>
<td>kernel scheduling events</td>
</tr>
<tr>
<td>sysinfo</td>
<td>system statistic events</td>
</tr>
<tr>
<td>vminfo</td>
<td>virtual memory events</td>
</tr>
<tr>
<td>io</td>
<td>system I/O events</td>
</tr>
<tr>
<td>profile</td>
<td>fixed rate sampling</td>
</tr>
<tr>
<td>pid</td>
<td>user-level tracing</td>
</tr>
<tr>
<td>fbt</td>
<td>raw kernel tracing</td>
</tr>
</tbody>
</table>
## Providers

- **Example of probes,**

<table>
<thead>
<tr>
<th>Provider</th>
<th>Example probe</th>
</tr>
</thead>
<tbody>
<tr>
<td>syscall</td>
<td>syscall:::read::entry</td>
</tr>
<tr>
<td>proc</td>
<td>proc:::exec-success</td>
</tr>
<tr>
<td>sched</td>
<td>sched:::on-cpu</td>
</tr>
<tr>
<td>sysinfo</td>
<td>sysinfo:::readch</td>
</tr>
<tr>
<td>vminfo</td>
<td>vminfo:::maj_fault</td>
</tr>
<tr>
<td>io</td>
<td>io:::start</td>
</tr>
<tr>
<td>profile</td>
<td>profile:::profile-1000hz</td>
</tr>
<tr>
<td>pid</td>
<td>pid172:libc:fopen:entry</td>
</tr>
<tr>
<td></td>
<td>pid172:a.out:main:entry</td>
</tr>
<tr>
<td>fbt</td>
<td>fbt:::bdev_strategy:entry</td>
</tr>
</tbody>
</table>
Providers

• Providers are documented in the DTrace Guide as separate chapters.
• Providers are dynamic; the number of available probes can vary.
• Some providers are “unstable interface”, such as fbt and sdt.
  • This means that their probes, while useful, may vary in name and arguments between Solaris versions.
  • Try to use stable providers instead (if possible).
  • Test D scripts that use unstable providers across target Solaris releases
Provider Documentation

● Some providers assume a little background knowledge, other providers assume a lot. Knowing where to find supporting documentation is important.

● Where do you find documentation on -
  ● Syscalls?
  ● User Libraries?
  ● Application Code?
  ● Kernel functions?
## Provider Documentation

- Additional documentation may be found here,

<table>
<thead>
<tr>
<th>Target</th>
<th>Provider</th>
<th>Additional Docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>syscalls</td>
<td>syscall</td>
<td>man(2)</td>
</tr>
<tr>
<td>libraries</td>
<td>pid:lib*</td>
<td>man(3C)</td>
</tr>
<tr>
<td>app code</td>
<td>pid:a.out</td>
<td>source code, ISV, developers</td>
</tr>
<tr>
<td>raw kernel</td>
<td>fbt</td>
<td>Solaris Internals 2\textsuperscript{nd} Ed, <a href="http://cvs.opensolaris.org">http://cvs.opensolaris.org</a></td>
</tr>
</tbody>
</table>
Actions

- When a probe fires, an action executes.
- Actions are written in the D programming language.
- Actions can,
  - print output
  - save data to variables, and perform calculations
  - walk kernel or process memory
- With destruction actions allowed, actions can,
  - raise signals on processes
  - execute shell commands
  - write to some areas of memory
trace() Example

The `trace()` action accepts one argument and prints it when the probe fired.

```bash
# dtrace -n 'syscall::exece:return { trace(execname); }'
dtrace: description 'syscall::exece:return' matched 1 probe

<table>
<thead>
<tr>
<th>CPU</th>
<th>ID</th>
<th>FUNCTION</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>man</td>
</tr>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>sh</td>
</tr>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>neqn</td>
</tr>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>tbl</td>
</tr>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>nroff</td>
</tr>
<tr>
<td>0</td>
<td>76044</td>
<td>exece:return</td>
<td>col</td>
</tr>
</tbody>
</table>
```

[...]
printf() Example

DTrace ships with a powerful printf(), to print formatted output.

```
# dtrace -n 'syscall::exece:return { printf("%6d %s\n", pid, execname); }'
```

<table>
<thead>
<tr>
<th>CPU</th>
<th>ID</th>
<th>FUNCTION:NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>74415</td>
<td>exece:return 4301 sh</td>
</tr>
<tr>
<td>0</td>
<td>74415</td>
<td>exece:return 4304 neqn</td>
</tr>
<tr>
<td>0</td>
<td>74415</td>
<td>exece:return 4305 nroff</td>
</tr>
<tr>
<td>0</td>
<td>74415</td>
<td>exece:return 4306 sh</td>
</tr>
<tr>
<td>0</td>
<td>74415</td>
<td>exece:return 4308 sh</td>
</tr>
</tbody>
</table>

[...]
DTrace Built-In Variables

- Numerous predefined variables can be used, e.g.,
  - `pid, tid`  Process ID, Thread ID
  - `timestamp`  Nanosecond timestamp since boot
  - `probefunc`  Probe function name (3rd field)
  - `execname`  Process name
  - `arg0, ...`  Function arguments and return value
  - `errno`  Last syscall failure error code
  - `curpsinfo`  Struct containing current process info, e.g.,
    `curpsinfo->pr_psargs` – process + args

- Pointers and structs! DTrace can walk memory using C syntax, and has kernel types predefined.
User-Defined Variable Types

- DTrace supports the following variable types
  - Integers
  - Structs
  - Pointers
  - Strings
  - Associative arrays
  - Aggregates
- Including types from /usr/include/sys
  - e.g. uint32_t.
Predicates

- DTrace predicates are used to filter probes, so that the action fires when a conditional is true.

```
probsname /predicate/ { action }
```

- E.g., syscalls for processes called “bash”,

```
# dtrace -n 'syscall:::entry /execname == "bash"/ { @num[probefunc] = count(); }'
dtrace: description 'syscall:::entry ' matched 233 probes
^C

  exece 2
[...]
  read 29
  write 31
  lwp_sigmask 42
  sigaction 62
```
DTrace – command line

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

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

#!/usr/sbin/dtrace -s

probe descriptions
/ predicate /
{
    action statements
}
A complete dtrace script block, including probename, a predicate, and an action in the probe clause, which sets a thread-local variable:

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

dtrace:::BEGIN
{
    vtotal = 0;
}

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

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);
}
```

```
usenix> cat syscalls_pid.d
```

```bash
#!/usr/sbin/dtrace -s
dtrace:::BEGIN
{
    vtotal = 0;
}
syscall:::entry
/pid == $target/
{
    self->vtime = vtimestamp;
}
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

<table>
<thead>
<tr>
<th>CPU</th>
<th>ID</th>
<th>FUNCTION:NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>:END</td>
</tr>
<tr>
<td></td>
<td></td>
<td>getpid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gtime</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sysi86</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>close</td>
</tr>
<tr>
<td></td>
<td></td>
<td>getrlimit</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>setcontext</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fstat64</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>brk</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>open</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>munmap</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>mmap</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>write</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>ioctl</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td></td>
</tr>
</tbody>
</table>
DTrace time-based probes

- profile – interval time based probes
  - profile-97hz – profile fires on all CPUs
  - tick-97hz – tick fires on 1 CPU
  - Interval can be specified with various suffixes
    - ns, us, ms, s, min (m), hour (h), day (d), hz
  - arg0 – kernel PC
  - arg1 – user PC
- Use arg0 or arg1 in a predicate for user or kernel profile

```bash
#dtrace -n 'profile-97hz / arg0 != 0 / { action } /* Am I in the kernel? */
#dtrace -n 'profile-97hz / arg1 != 0 / { action } /* Am I in user mode? */
```
Using Providers

Using the syscall provider to track bytes passed to write(2)

Using the fbt provider to instrument the kernel ufs_write() function, and track the filename in the probe action

```
# dtrace -n 'syscall::write:entry { trace(arg2) }'
dtrace: description 'write:entry ' matched 2 probes

<table>
<thead>
<tr>
<th>CPU</th>
<th>ID</th>
<th>FUNCTION:NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1026</td>
<td>write:entry</td>
</tr>
<tr>
<td>1</td>
<td>1026</td>
<td>write:entry</td>
</tr>
<tr>
<td></td>
<td>9290</td>
<td>write:entry</td>
</tr>
<tr>
<td>1</td>
<td>1026</td>
<td>write:entry</td>
</tr>
<tr>
<td>1</td>
<td>9290</td>
<td>write:entry</td>
</tr>
<tr>
<td>1</td>
<td>1026</td>
<td>write:entry</td>
</tr>
<tr>
<td>1</td>
<td>9290</td>
<td>write:entry</td>
</tr>
<tr>
<td>1</td>
<td>1026</td>
<td>write:entry</td>
</tr>
<tr>
<td>1</td>
<td>9290</td>
<td>write:entry</td>
</tr>
</tbody>
</table>

# dtrace -n 'fbt:ufs:ufs_write:entry { printf("%s\n",stringof(args[0]->v_path)); }'
dtrace: description 'ufs_write:entry ' matched 1 probe

<table>
<thead>
<tr>
<th>CPU</th>
<th>ID</th>
<th>FUNCTION:NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>16779</td>
<td>ufs_write:entry /etc/svc/repository.db-journal</td>
</tr>
<tr>
<td></td>
<td>16779</td>
<td>ufs_write:entry /etc/svc/repository.db-journal</td>
</tr>
<tr>
<td></td>
<td>16779</td>
<td>ufs_write:entry /etc/svc/repository.db-journal</td>
</tr>
<tr>
<td></td>
<td>16779</td>
<td>ufs_write:entry /etc/svc/repository.db-journal</td>
</tr>
<tr>
<td></td>
<td>16779</td>
<td>ufs_write:entry /etc/svc/repository.db-journal</td>
</tr>
<tr>
<td></td>
<td>16779</td>
<td>ufs_write:entry /etc/svc/repository.db-journal</td>
</tr>
<tr>
<td></td>
<td>16779</td>
<td>ufs_write:entry /etc/svc/repository.db-journal</td>
</tr>
<tr>
<td></td>
<td>16779</td>
<td>ufs_write:entry /etc/svc/repository.db-journal</td>
</tr>
</tbody>
</table>
```
DTrace PID Provider

Using the PID provider to instrument all the function entry points the process calls in libc
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)) to post-process reams of data
• DTrace supports aggregation of data as a first class operation
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.

● Functions:
  ● `avg()` - the average of specified expressions
  ● `min()` - the minimum of specified expressions
  ● `max()` - the maximum of specified expressions
  ● `count()` - number of times the probe fired
  ● `sum()` - running sum
  ● `quantize()` - power-of-two exponential distribution
  ● `lquantize()` - linear frequency distribution

● For example, distribution of write(2) sizes by executable name:

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

- Frequency counting syscalls,

```bash
# dtrace -n 'syscall:::entry { @num[probefunc] = count(); }'
dtrace: description 'syscall:::entry ' matched 233 probes
^C
[...]
   writev           170
   write            257
   read             896
   pollsys         959
   ioctl          1253
```
Quantize

- Very cool function, here we quantize writech sizes:
  - Here we see that `ls` processes usually write between 32 and 127 bytes.
  - Makes sense?

```bash
# dtrace -n 'sysinfo:::writech { @dist[execname] = quantize(arg0); }'
dtrace: description 'sysinfo:::writech ' matched 4 probes
^C
[...]
ls

<table>
<thead>
<tr>
<th>value</th>
<th>Distribution</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>@@@@@@@@@@@@@@@@</td>
<td>118</td>
</tr>
<tr>
<td>64</td>
<td>@@@@@@@@@@@@@@@@@@@</td>
<td>127</td>
</tr>
<tr>
<td>128</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
[...]```
ls -l

# ls -l /etc
dttotal 793
lrwxrwxrwx 1 root root 12 Mar 21 03:28 TIMEZONE -> default/init
drwxr-xr-x 4 root sys  6 Apr 16 06:59 X11
drwxr-xr-x 2 adm  adm  3 Mar 20 09:25 acct
drwxr-xr-x 3 root root  3 Apr 16 23:11 ak
lrwxrwxrwx 1 root root 12 Mar 21 03:28 aliases -> mail/aliases
drwxr-xr-x 5 root sys  5 Feb 20 23:29 amd64
drwxr-xr-x 7 root bin 18 Mar 20 09:20 apache
drwxr-xr-x 4 root bin  7 Feb 20 23:12 apache2
drwxr-xr-x 2 root sys  5 Feb 20 23:27 apoc
-rw-r--r-- 1 root bin  1012 Mar 20 09:33 auto_home
-rw-r--r-- 1 root bin  1066 Mar 20 09:33 auto_master
lrwxrwxrwx 1 root root 16 Mar 21 03:28 autopush -> ..:/sbin/autopu

[...]
Quantize aggregations – write(2) bytes by process

```bash
# dtrace -n 'syscall::write:entry { @[execname] = quantize(arg2); }'
dtrace: description 'syscall::write:entry ' matched 1 probe
^C
```

```text
in.rshd

<table>
<thead>
<tr>
<th>value</th>
<th>Distribution</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>@@@@@@@@@@@@</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>@@@</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>@@@</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>@@@</td>
<td>7</td>
</tr>
<tr>
<td>32</td>
<td>@@@</td>
<td>4</td>
</tr>
<tr>
<td>64</td>
<td>@@@@@@@@@@@@@@</td>
<td>23</td>
</tr>
<tr>
<td>128</td>
<td>@@@@@@@@@@@@@@@@@@@@@@@@@@@@</td>
<td>2</td>
</tr>
<tr>
<td>256</td>
<td>@</td>
<td>1</td>
</tr>
</tbody>
</table>

```text
cat

<table>
<thead>
<tr>
<th>value</th>
<th>Distribution</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
| 256   | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
DTrace Enhancements post S10 FCS

- Multiple aggregations with printa()
- Aggregation key sort options
- `(u)func(%pc), (u)mod(%pc), (u)sym(%pc)` dtrace functions
  - Get symbolic name from address
- `ucaller` function
  - Track function callers
- String parsing routines
- `fds[]`
  - Array of `fileinfo_t`'s indexed by `fd`
- Providers
  - `fsinfo`
  - `sysevent`
  - `Xserver`
  - `iscsi`
Multiple aggregation printa()
Release: 08/07-30

- multiple aggregations in single printa()
- aggregations must have same type signature
- output is effectively joined by key
- 0 printed when no value present for a key
- default behavior is to sort by first aggregation value (ties broken by key order)
Multiple aggregation printa()

/* multagg.d */
system::write:entry
{
    @wbytes[execname, pid] = sum(arg2);
}

system::read:entry
{
    @rbytes[execname, pid] = sum(arg2);
}
END
{
    normalize(@rbytes, 1024);
    normalize(@wbytes, 1024);
    printf("%20s %10s %10s %10s\n", "PROGRAM", "PID", "READS", "WRITES");
    printa("%20s %10d %10d %10d\n", @rbytes, @wbytes);
}
Multiple aggregation printa()

```bash
# dtrace -q -s ./multagg.d
^C
```

<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>PID</th>
<th>READS</th>
<th>WRITES</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtrace</td>
<td>101605</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nautilus</td>
<td>101606</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>battstat-applet-</td>
<td>100854</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>gnome-settings-d</td>
<td>100781</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gnome-session</td>
<td>100728</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>dsdm</td>
<td>100712</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>gnome-terminal</td>
<td>101342</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>xscreensaver</td>
<td>100946</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>soffice.bin</td>
<td>101325</td>
<td>10</td>
<td>48</td>
</tr>
<tr>
<td>init</td>
<td>1</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>gnome-panel</td>
<td>100792</td>
<td>39</td>
<td>4</td>
</tr>
<tr>
<td>gconfd-2</td>
<td>100767</td>
<td>47</td>
<td>5</td>
</tr>
<tr>
<td>nautilus</td>
<td>100794</td>
<td>265</td>
<td>170</td>
</tr>
<tr>
<td>wnck-applet</td>
<td>100822</td>
<td>714</td>
<td>262</td>
</tr>
<tr>
<td>metacity</td>
<td>100789</td>
<td>726</td>
<td>263</td>
</tr>
<tr>
<td>gedit</td>
<td>101607</td>
<td>3986</td>
<td>64</td>
</tr>
<tr>
<td>Xorg</td>
<td>100535</td>
<td>12374</td>
<td>0</td>
</tr>
</tbody>
</table>
Aggregation sorting options
Release: 08/07-30

- aggregations sorted by value by default
- options allow change of behaviour
  - aggsortkey - sort by key order, ties broken by value
  - aggsortrev - reverse sort
  - aggsortpos - position of the aggregation to use as sort primary sort key with multiple aggs
  - aggsortkeypos - position of key to use as primary sort key when with multiple aggs

- Use the above in combination
Aggregation sorting options

/* aggsort.d */
syscall::read:entry
{
  @avg[execname, pid] = avg(arg2);
  @max[execname, pid] = max(arg2);
  @min[execname, pid] = min(arg2);
  @cnt[execname, pid] = count();
}

END
{
  printf("%20s %10s %10s %10s %10s %10s\n", "EXECNAME", "PID", "COUNT", "MIN", "MAX", "AVG");
  printa("%20s %10d %@10d %@10d %@10d %@10d\n", @cnt, @min, @max, @avg);
}
Aggregation sorting options

```bash
# dtrace -q -s ./aggsort.d
^C

<table>
<thead>
<tr>
<th>EXECNAME</th>
<th>PID</th>
<th>COUNT</th>
<th>MIN</th>
<th>MAX</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>battstat-applet-</td>
<td>100981</td>
<td>2</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>gnome-settings-d</td>
<td>100853</td>
<td>3</td>
<td>32</td>
<td>64</td>
<td>53</td>
</tr>
<tr>
<td>soffice.bin</td>
<td>101382</td>
<td>10</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>dsdm</td>
<td>100708</td>
<td>10</td>
<td>32</td>
<td>160</td>
<td>54</td>
</tr>
<tr>
<td>xscreensaver</td>
<td>101082</td>
<td>14</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>gnome-panel</td>
<td>100896</td>
<td>24</td>
<td>12</td>
<td>168</td>
<td>51</td>
</tr>
<tr>
<td>firefox-bin</td>
<td>101363</td>
<td>31</td>
<td>1</td>
<td>1024</td>
<td>35</td>
</tr>
<tr>
<td>gnome-terminal</td>
<td>101029</td>
<td>40</td>
<td>32</td>
<td>4096</td>
<td>163</td>
</tr>
<tr>
<td>nautilus</td>
<td>100906</td>
<td>119</td>
<td>32</td>
<td>480</td>
<td>48</td>
</tr>
<tr>
<td>wnck-applet</td>
<td>100961</td>
<td>161</td>
<td>8</td>
<td>128</td>
<td>32</td>
</tr>
<tr>
<td>metacity</td>
<td>100893</td>
<td>447</td>
<td>4</td>
<td>96</td>
<td>33</td>
</tr>
<tr>
<td>Xorg</td>
<td>100534</td>
<td>926</td>
<td>64</td>
<td>5104</td>
<td>3263</td>
</tr>
</tbody>
</table>
```

Sort by value of first aggregation (default)
aggregation sorting options

```
# dtrace -q -s ./aggsort.d -x aggsortrev
^C
```

<table>
<thead>
<tr>
<th>EXECNAME</th>
<th>PID</th>
<th>COUNT</th>
<th>MIN</th>
<th>MAX</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xorg</td>
<td>100534</td>
<td>875</td>
<td>64</td>
<td>5104</td>
<td>3433</td>
</tr>
<tr>
<td>metacity</td>
<td>100893</td>
<td>434</td>
<td>4</td>
<td>96</td>
<td>33</td>
</tr>
<tr>
<td>wnck-applet</td>
<td>100961</td>
<td>145</td>
<td>8</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>xscreensaver</td>
<td>101082</td>
<td>71</td>
<td>4</td>
<td>32</td>
<td>28</td>
</tr>
<tr>
<td>gnome-terminal</td>
<td>101029</td>
<td>54</td>
<td>32</td>
<td>4096</td>
<td>125</td>
</tr>
<tr>
<td>nautilus</td>
<td>100906</td>
<td>51</td>
<td>32</td>
<td>608</td>
<td>74</td>
</tr>
<tr>
<td>firefox-bin</td>
<td>101363</td>
<td>23</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>dsdm</td>
<td>100708</td>
<td>19</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>gnome-panel</td>
<td>100896</td>
<td>18</td>
<td>12</td>
<td>168</td>
<td>51</td>
</tr>
<tr>
<td>gnome-settings-d</td>
<td>100853</td>
<td>7</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>soffice.bin</td>
<td>101382</td>
<td>6</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

Reverse sort using value of first aggregation
Aggregation sorting options

```bash
# dtrace -q -s ./aggsort.d  -x aggsortkey -x aggsortkeypos=1 -x aggsortrev
^C

<table>
<thead>
<tr>
<th>EXECENAME</th>
<th>PID</th>
<th>COUNT</th>
<th>MIN</th>
<th>MAX</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>soffice.bin</td>
<td>101382</td>
<td>525</td>
<td>4</td>
<td>1440</td>
<td>33</td>
</tr>
<tr>
<td>firefox-bin</td>
<td>101363</td>
<td>29</td>
<td>1</td>
<td>1024</td>
<td>36</td>
</tr>
<tr>
<td>thunderbird-bin</td>
<td>101337</td>
<td>2</td>
<td>1</td>
<td>1024</td>
<td>512</td>
</tr>
<tr>
<td>xscreensaver</td>
<td>101082</td>
<td>11</td>
<td>32</td>
<td>64</td>
<td>34</td>
</tr>
<tr>
<td>gnome-terminal</td>
<td>101029</td>
<td>27</td>
<td>32</td>
<td>4096</td>
<td>220</td>
</tr>
<tr>
<td>wnck-applet</td>
<td>100961</td>
<td>161</td>
<td>8</td>
<td>96</td>
<td>32</td>
</tr>
<tr>
<td>nautilus</td>
<td>100906</td>
<td>79</td>
<td>32</td>
<td>320</td>
<td>40</td>
</tr>
<tr>
<td>gnome-panel</td>
<td>100896</td>
<td>26</td>
<td>12</td>
<td>168</td>
<td>49</td>
</tr>
<tr>
<td>metacity</td>
<td>100893</td>
<td>196</td>
<td>4</td>
<td>128</td>
<td>34</td>
</tr>
<tr>
<td>gnome-settings-d</td>
<td>100853</td>
<td>4</td>
<td>32</td>
<td>64</td>
<td>40</td>
</tr>
<tr>
<td>dsdm</td>
<td>100708</td>
<td>23</td>
<td>32</td>
<td>128</td>
<td>40</td>
</tr>
<tr>
<td>Xorg</td>
<td>100534</td>
<td>885</td>
<td>64</td>
<td>4940</td>
<td>3688</td>
</tr>
</tbody>
</table>
```

Reverse sort by key in second position
Aggregation sorting options

```bash
# dtrace -q -s ./aggsort.d -x aggsortpos=2 -x aggsortrev
```

<table>
<thead>
<tr>
<th>EXECNAME</th>
<th>PID</th>
<th>COUNT</th>
<th>MIN</th>
<th>MAX</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xorg</td>
<td>100534</td>
<td>1655</td>
<td>64</td>
<td>5104</td>
<td>3756</td>
</tr>
<tr>
<td>gnome-terminal</td>
<td>101029</td>
<td>137</td>
<td>32</td>
<td>4096</td>
<td>69</td>
</tr>
<tr>
<td>wnck-applet</td>
<td>100961</td>
<td>453</td>
<td>4</td>
<td>1152</td>
<td>34</td>
</tr>
<tr>
<td>xscreensaver</td>
<td>101082</td>
<td>23</td>
<td>32</td>
<td>1024</td>
<td>115</td>
</tr>
<tr>
<td>nautilus</td>
<td>100906</td>
<td>43</td>
<td>8</td>
<td>736</td>
<td>69</td>
</tr>
<tr>
<td>soffice.bin</td>
<td>101382</td>
<td>637</td>
<td>4</td>
<td>288</td>
<td>30</td>
</tr>
<tr>
<td>gnome-panel</td>
<td>100896</td>
<td>122</td>
<td>8</td>
<td>168</td>
<td>39</td>
</tr>
<tr>
<td>metacity</td>
<td>100893</td>
<td>421</td>
<td>4</td>
<td>128</td>
<td>34</td>
</tr>
<tr>
<td>notification-are</td>
<td>100983</td>
<td>2</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>mixer_applet2</td>
<td>100985</td>
<td>2</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>gnome-settings-d</td>
<td>100853</td>
<td>7</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>dsdm</td>
<td>100708</td>
<td>103</td>
<td>4</td>
<td>32</td>
<td>31</td>
</tr>
<tr>
<td>thunderbird-bin</td>
<td>101337</td>
<td>7</td>
<td>4</td>
<td>32</td>
<td>24</td>
</tr>
<tr>
<td>firefox-bin</td>
<td>101363</td>
<td>39</td>
<td>1</td>
<td>32</td>
<td>5</td>
</tr>
</tbody>
</table>

Reverse sorted by value of third aggregation
(u)mod/(u)func/(u)sym
Release: 08/07-23

• Profiling often requires post-processing when using %a/%A to print arg0/arg1 symbolically
• Samples in format [module]'[func]+[offset]
• Want to first get high level view and then drill down
• (u)mod(%pc) - module name
• (u)func(%pc) - function name
• (u)sym(%pc) - symbol name
Example uses prototype cpc provider to show TLB misses on a global basis broken down by module.
(u)mod/(u)func/(u)sym

```
# ./tlbmissbymod.d
^C

<table>
<thead>
<tr>
<th>EXECUTABLE</th>
<th>MODULE</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xorg</td>
<td>libramdac.so</td>
<td>1</td>
</tr>
<tr>
<td>Xorg</td>
<td>libfb.so</td>
<td>1</td>
</tr>
<tr>
<td>Xorg</td>
<td>radeonDrv.so</td>
<td>1</td>
</tr>
<tr>
<td>Xorg</td>
<td>libc.so.1</td>
<td>3</td>
</tr>
<tr>
<td>battstat-applet-</td>
<td>libglib-2.0.so.0.1200.4</td>
<td>1</td>
</tr>
<tr>
<td>battstat-applet-</td>
<td>libobject-2.0.so.0.1200.4</td>
<td>1</td>
</tr>
<tr>
<td>gconfd-2</td>
<td>libORBit-2.so.0.1.0</td>
<td>1</td>
</tr>
<tr>
<td>gconfd-2</td>
<td>libgconf-2.so.4.1.0</td>
<td>1</td>
</tr>
<tr>
<td>metacity</td>
<td>libpangocairo-1.0.so.0.1400.7</td>
<td>1</td>
</tr>
<tr>
<td>metacity</td>
<td>libgdk_pixbuf-2.0.so.0.1000.6</td>
<td>1</td>
</tr>
<tr>
<td>metacity</td>
<td>libobject-2.0.so.0.1200.4</td>
<td>1</td>
</tr>
<tr>
<td>metacity</td>
<td>libglib-2.0.so.0.1200.4</td>
<td>2</td>
</tr>
<tr>
<td>metacity</td>
<td>libgdk-x11-2.0.so.0.1000.6</td>
<td>1</td>
</tr>
<tr>
<td>metacity</td>
<td>libc.so.1</td>
<td>2</td>
</tr>
</tbody>
</table>
```
ucaller variable
Release: 08/07-23

```
# dtrace -n 'pid$target::malloc:entry[@[ufunc(ucaller)] = count();};]' -p 101384
dtrace: description 'pid$target::malloc:entry' matched 2 probes
^C

sax.uno.so`0xf97b2fe7                                             2
sax.uno.so`0xf97b30e7                                             2
sax.uno.so`0xf97b3104                                             2
sax.uno.so`0xf97b44ba                                             2
sax.uno.so`0xf97b44cf                                             2
sax.uno.so`0xf97b8a18                                             4
libX11.so.4`_XAllocScratch                                        5
libX11.so.4`miRegionOp                                             6
libX11.so.4`XQueryTree                                            8
libX11.so.4`XGetWindowProperty                                    13
libX11.so.4`XSetClassHint                                         13
libX11.so.4`XGetImage                                             179
libuno_sal.so.3`osl_createMutex                                   322
libX11.so.4`XCreateRegion                                         844
libX11.so.4`XCreateGC                                             959
libc.so.1`calloc                                               1074
libglib-2.0.so.0.1200.4`standard_malloc                         3533
libCrun.so.1`__1c2n6FI_pv_                                        28668
```
The fds[] variable
Release: 01/06-16

- array of fileinfo_t's indexed by integer (fd)
- inlines expanded to accommodate associative arrays for this.
- Definition in /usr/lib/dtrace/io.d
- fileinfo_t gets new fi_oflags member
The fds[] variable

#pragma D option quiet

syscall::write:entry
/fds[arg0].fi_oflags & O_APPEND/
{
    printf("%s appending file %s at offset %d\n",
            execname, fds[arg0].fi_pathname, fds[0].fi_offset);
}

# ./fds.d
ksh appending file /.sh_history at offset 349345
ksh appending file /.sh_history at offset 349378
Allowing dtrace for non-root users

- Setting dtrace privileges
- Add a line for the user in `/etc/user_attr`

```
mauroj: : : defaultpriv=basic, dtrace_user, dtrace_kernel, dtrace_proc, proc_owner
```

from privileges(5)

PRIV_DTRACE_KERNEL
Allow DTrace kernel-level tracing.

PRIV_DTRACE_PROC
Allow DTrace process-level tracing. Allow process-level tracing probes to be placed and enabled in processes to which the user has permissions.

PRIV_DTRACE_USER
Allow DTrace user-level tracing. Allow use of the syscall and profile DTrace providers to examine processes to which the user has permissions.
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 - \texttt{mdb(1)}

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

- **Modules**
  - Modules in `/usr/lib/mdb`, `/usr/platform/lib/mdb` etc
  - `mdb` can use `adb` macros
  - Developer Interface - write your own `dcmds` and walkers
```
> ::cpuinfo

<table>
<thead>
<tr>
<th>ID</th>
<th>ADDR</th>
<th>FLG</th>
<th>NRUN</th>
<th>BSPL</th>
<th>PRI</th>
<th>RNRN</th>
<th>KNRN</th>
<th>SWITCH</th>
<th>THREAD</th>
<th>PROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000180c000</td>
<td>1b</td>
<td>0</td>
<td>0</td>
<td>37</td>
<td>no</td>
<td>no</td>
<td>t-0</td>
<td>3002ec8ca0 threads</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>30001b78000</td>
<td>1b</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>no</td>
<td>no</td>
<td>t-0</td>
<td>31122698960 threads</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>30001b7a000</td>
<td>1b</td>
<td>0</td>
<td>0</td>
<td>59</td>
<td>no</td>
<td>no</td>
<td>t-0</td>
<td>30ab913cd00 find</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>30001c18000</td>
<td>1b</td>
<td>0</td>
<td>0</td>
<td>59</td>
<td>no</td>
<td>no</td>
<td>t-0</td>
<td>31132397620 sshd</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>30001c16000</td>
<td>1b</td>
<td>0</td>
<td>0</td>
<td>37</td>
<td>no</td>
<td>no</td>
<td>t-0</td>
<td>3112280f020 threads</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>30001c0e000</td>
<td>1b</td>
<td>0</td>
<td>0</td>
<td>59</td>
<td>no</td>
<td>no</td>
<td>t-1</td>
<td>311227632e0 mdb</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>30001c06000</td>
<td>1b</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>no</td>
<td>no</td>
<td>t-0</td>
<td>2a100609cc0 (idle)</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>30001c02000</td>
<td>1b</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>no</td>
<td>no</td>
<td>t-1</td>
<td>300132c5900 threads</td>
<td></td>
</tr>
</tbody>
</table>

> 30001b78000::cpuinfo -v

<table>
<thead>
<tr>
<th>ID</th>
<th>ADDR</th>
<th>FLG</th>
<th>NRUN</th>
<th>BSPL</th>
<th>PRI</th>
<th>RNRN</th>
<th>KNRN</th>
<th>SWITCH</th>
<th>THREAD</th>
<th>PROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30001b78000</td>
<td>1b</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>no</td>
<td>no</td>
<td>t-3</td>
<td>2a100307cc0 (idle)</td>
<td></td>
</tr>
</tbody>
</table>

  RUNNING ---+READY
  EXISTS
  ENABLE

> 30001b78000::cpuinfo -v

<table>
<thead>
<tr>
<th>ID</th>
<th>ADDR</th>
<th>FLG</th>
<th>NRUN</th>
<th>BSPL</th>
<th>PRI</th>
<th>RNRN</th>
<th>KNRN</th>
<th>SWITCH</th>
<th>THREAD</th>
<th>PROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30001b78000</td>
<td>1b</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>no</td>
<td>no</td>
<td>t-1</td>
<td>300132c5900 threads</td>
<td></td>
</tr>
</tbody>
</table>

  RUNNING ---+READY
  EXISTS
  ENABLE

> 300132c5900::findstack

stack pointer for thread 300132c5900: 2a1016dd1a1
000002a1016dd2f1 user_rtt+0x20()
mdb(1) & dtrace(1) – Perfect Together

```
# mdb -k

Loading modules: [ unix krtld genunix specfs dtrace ufs sd ip sctp usba fcp fctl nca nfs random sopp
lofs crypto ptm logindmux md isp cpc fcip ipc ]

> ufs_read::nm -f ctype

C Type

int (*)(struct vnode *, struct uio *, int, struct cred *, struct caller_context *)

> ::print -t struct vnode

{  
    kmutex_t v_lock {
        void * [1] _opaque
    }
    uint_t v_flag
    uint_t v_count
    void *v_data
    struct vfs *v_vfsp
    struct stdata *v_stream
    enum vtype v_type
    dev_t v_rdev
    struct vfs *v_vfsmountedhere
    struct vnodeops *v_op
    struct page *v_pages
    pgcnt_t v_npages
    ...
    char *v_path
    ...
```

```
# dtrace -n 'ufs_read:entry { printf("%s\n",stringof(args[0]->v_path));}'}

dtrace: description 'ufs_read:entry' matched 1 probe

<table>
<thead>
<tr>
<th>CPU</th>
<th>ID</th>
<th>FUNCTION:NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16777</td>
<td>ufs_read:entry /usr/bin/cut</td>
</tr>
<tr>
<td>1</td>
<td>16777</td>
<td>ufs_read:entry /usr/bin/cut</td>
</tr>
<tr>
<td>1</td>
<td>16777</td>
<td>ufs_read:entry /usr/bin/cut</td>
</tr>
<tr>
<td>1</td>
<td>16777</td>
<td>ufs_read:entry /usr/bin/cut</td>
</tr>
<tr>
<td>1</td>
<td>16777</td>
<td>ufs_read:entry /lib/ld.so.1</td>
</tr>
<tr>
<td>1</td>
<td>16777</td>
<td>ufs_read:entry /lib/ld.so.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
```
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

```bash
# 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
```
zeeroh> kstat -n e1000g0

<table>
<thead>
<tr>
<th></th>
<th>instance: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>module:</td>
<td>e1000g</td>
</tr>
<tr>
<td>name:</td>
<td>e1000g0</td>
</tr>
<tr>
<td>brdcstrcv</td>
<td>31397806</td>
</tr>
<tr>
<td>brdcstxmt</td>
<td>104402</td>
</tr>
<tr>
<td>collisions</td>
<td>0</td>
</tr>
<tr>
<td>crtime</td>
<td>142.615878752</td>
</tr>
<tr>
<td>ierrors</td>
<td>0</td>
</tr>
<tr>
<td>ifspeed</td>
<td>1000000000</td>
</tr>
<tr>
<td>ipackets</td>
<td>33545701</td>
</tr>
<tr>
<td>ipackets64</td>
<td>33545701</td>
</tr>
<tr>
<td>multircv</td>
<td>796061</td>
</tr>
<tr>
<td>multixmt</td>
<td>0</td>
</tr>
<tr>
<td>norcvbuf</td>
<td>0</td>
</tr>
<tr>
<td>noxmtbuf</td>
<td>0</td>
</tr>
<tr>
<td>obytes</td>
<td>169913542</td>
</tr>
<tr>
<td>obytes64</td>
<td>169913542</td>
</tr>
<tr>
<td>oerrors</td>
<td>0</td>
</tr>
<tr>
<td>opackets</td>
<td>1141181</td>
</tr>
<tr>
<td>opackets64</td>
<td>1141181</td>
</tr>
<tr>
<td>rbytes</td>
<td>1559052386</td>
</tr>
<tr>
<td>rbytes64</td>
<td>5854019682</td>
</tr>
<tr>
<td>snaptime</td>
<td>15146011.0147421</td>
</tr>
<tr>
<td>unknowns</td>
<td>819041</td>
</tr>
<tr>
<td>module:</td>
<td>e1000g</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>name:</td>
<td>e1000g0</td>
</tr>
<tr>
<td></td>
<td>opackets</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>module:</th>
<th>e1000g</th>
<th>instance:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>name:</td>
<td>e1000g0</td>
<td>class:</td>
<td>net</td>
</tr>
<tr>
<td></td>
<td>opackets</td>
<td></td>
<td>1141268</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>module:</th>
<th>e1000g</th>
<th>instance:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>name:</td>
<td>e1000g0</td>
<td>class:</td>
<td>net</td>
</tr>
<tr>
<td></td>
<td>opackets</td>
<td></td>
<td>1141273</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>module:</th>
<th>e1000g</th>
<th>instance:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>name:</td>
<td>e1000g0</td>
<td>class:</td>
<td>net</td>
</tr>
<tr>
<td></td>
<td>opackets</td>
<td></td>
<td>1141279</td>
</tr>
</tbody>
</table>
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
  - pflags
  - pc gtk
  - pldd
  - psig
  - pstack
  - pmap
  - pfiles
  - pstop
  - prun
  - pwait
  - ptree
  - ptime
  - preap
pflags, pcred, pldd, pkill

nv70b> **pflags 3645**
3645: ./loader 2 0 /zp/space
data model = _ILP32 flags = MSACCT|MSFORK
/1: flags = ASLEEP  lwp_wait(0x2,0x80471f0)
/2: flags = 0
/3: flags = 0

nv70b> **pcred 3645**
3645: e/r/suid=20821 e/r/sgid=3

nv70b> **pldd 3645**
3645: ./loader 2 0 /zp/space
/lib/libc.so.1

nv70b> **pkill loader**
nv70b>
[1]+ Terminated ./loader 2 0 /zp/space
nv70b> psig 3654
3654: ./loader 2 0 /zp/space
HUP     default
INT     default
QUIT    default
ILL     default
TRAP    default
ABRT    default
EMT     default
FPE     default
KILL    default
BUS     default
SEGV    default
SYS     default
PIPE    default
ALRM    default
TERM    default
USR1    caught timer 0
USR2    default
CLD     default
PWR     default

...
nv70b> pstack 3602
3602: /usr/lib/firefox/firefox-bin -UILocale C -contentLocale C

----------------- lwp# 1 / thread# 1 -------------------
d117ec45 pollsys (82d5910, 8, 0, 0)
d1134212 poll (82d5910, 8, ffffffff) + 52
d0c06653 g_main_context_iterate (80c3260, 1, 1, 815f1f0) + 397
d0c06c8c g_main_loop_run (82f45f8) + 1b8
d0964fae gtk_main (8047028, 808578c, 8189718, 8046d88, cd694803, 818ed20) + b2
cdbc4bb4 __1cKnsAppShell1DRun6M_I__ (818ed20) + 34
cd694803 __1cMnsAppStartupDRun6M_I__ (8189718) + 2b
08061824 XRE_main (5, 8047098, 8085760) + 25f4
0805a61d main (5, 8047098, 80470b0) + 25
0805a56a _start (5, 8047210, 804722d, 8047237, 8047239, 8047248) + 7a

----------------- lwp# 2 / thread# 2 -------------------
d117ec45 pollsys (cdeabc70, 1, 0, 0)
d1134212 poll (cdeabc70, 1, ffffffff) + 52
d0f48bfa _pr_poll_with_poll (80f00a8, 1, ffffffff) + 39a
d0f48dc6 PR_Poll (80f00a8, 1, ffffffff) + 16
ce15657a __1cYnsSocketTransportServiceEPoll16M_PI__ (80efbc0, cdeabf74) + 11e
cce157118 __1cYnsSocketTransportServiceDRun6M_I__ (80efbc0) + 68c
d103fff4 __1cInsThreadEMain6Fpv_v__ (80f0298) + 74
d0f4ab0d _pt_root (80f2720) + d1
d117d952 _thr_setup (cdd90200) + 52
d117dbb0 _lwp_start (cdd90200, 0, 0, 0, 0, 0)

----------------- lwp# 3 / thread# 3 -------------------
d117dc09 lwp_park (0, cd58de58, 0)
nv70b> pfiles 3602
3602: /usr/lib/firefox/firefox-bin -UILocale C -contentLocale C
Current rlimit: 512 file descriptors
  0: S_IFCHR mode:0666 dev:279,0 ino:6815752 uid:0 gid:3 rdev:13,2
      O_RDONLY|O_LARGEFILE
      /devices/pseudo/mm@0:null
  1: S_IFCHR mode:0666 dev:279,0 ino:6815752 uid:0 gid:3 rdev:13,2
      O_WRONLY|O_APPEND|O_CREAT|O_LARGEFILE
      /devices/pseudo/mm@0:null
  2: S_IFCHR mode:0666 dev:279,0 ino:6815752 uid:0 gid:3 rdev:13,2
      O_WRONLY|O_APPEND|O_CREAT|O_LARGEFILE
      /devices/pseudo/mm@0:null
  3: S_IFDOOR mode:0444 dev:290,0 ino:43 uid:0 gid:0 size:0
      O_RDONLY|O_LARGEFILE FD_CLOEXEC door to nscd[3545]
      /var/run/name_service_door
  4: S_IFIFO mode:0666 dev:290,0 ino:40 uid:0 gid:0 size:0
      O_RDWR|O_NONBLOCK FD_CLOEXEC
      . . .
  19: S_IFSOCK mode:0666 dev:287,0 ino:28594 uid:0 gid:0 size:0
      O_RDWR|O_NONBLOCK FD_CLOEXEC
      SOCK_STREAM
      SO_REUSEADDR,SO_SNDBUF(49152),SO_RCVBUF(49152)
      sockname: AF_INET 127.0.0.1 port: 59686
      . . .
```sh
solaris10> pfiles 26337
26337: /usr/lib/ssh/sshd

Current rlimit: 256 file descriptors

0: S_IFCHR mode:0666 dev:270,0 ino:6815752 uid:0 gid:3 rdev:13,2
  O_RDWR|O_LARGEFILE
  /devices/pseudo/mm@0:null

1: S_IFCHR mode:0666 dev:270,0 ino:6815752 uid:0 gid:3 rdev:13,2
  O_RDWR|O_LARGEFILE
  /devices/pseudo/mm@0:null

2: S_IFCHR mode:0666 dev:270,0 ino:6815752 uid:0 gid:3 rdev:13,2
  O_RDWR|O_LARGEFILE
  /devices/pseudo/mm@0:null

3: S_IFDOOR mode:0444 dev:279,0 ino:59 uid:0 gid:0 size:0
  O_RDONLY|O_LARGEFILE FD_CLOEXEC  door to nscd[93]
  /var/run/name_service_door

4: S_IFSOCK mode:0666 dev:276,0 ino:36024 uid:0 gid:0 size:0
  O_RDWR|O_NONBLOCK
    SOCK_STREAM
    SO_REUSEADDR,SO_KEEPALIVE,SO_SNDBUF(49152),SO_RCVBUF(49880)
    sockname: AF_INET6 ::ffff:129.154.54.9  port: 22
    peername: AF_INET6 ::ffff:129.150.32.45  port: 52002

5: S_IFDOOR mode:0644 dev:279,0 ino:55 uid:0 gid:0 size:0
  O_RDONLY FD_CLOEXEC  door to keyserv[179]
  /var/run/rpc_door/rpc_100029.1

....
```
pwdx, pstop, prun

nv70b> pwdx 3666
3666: /zp/home/mauroj/Programs

nv70b> pflags 3666
3666: ./loader 2 0 /zp/space
    data model = _ILP32 flags = MSACCT|MSFORK
/1:  flags = ASLEEP  lwp_wait(0x2,0x80471f0)
/2:  flags = 0
/3:  flags = 0

nv70b> pstop 3666
nv70b> pflags 3666
3666: ./loader 2 0 /zp/space
    data model = _ILP32 flags = MSACCT|MSFORK
    sigpend = 0x00008000,0x00000000
/1:  flags = STOPPED|ISTOP|ASLEEP  lwp_wait(0x2,0x80471f0)
    why = PR_REQUESTED
/2:  flags = STOPPED|ISTOP
    why = PR_REQUESTED
/3:  flags = STOPPED|ISTOP
    why = PR_REQUESTED

nv70b> prun 3666
nv70b> pflags 3666
3666: ./loader 2 0 /zp/space
    data model = _ILP32 flags = MSACCT|MSFORK
/1:  flags = ASLEEP  lwp_wait(0x2,0x80471f0)
/2:  flags = 0
/3:  flags = 0
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

<table>
<thead>
<tr>
<th>PID</th>
<th>USERNAME</th>
<th>SIZE</th>
<th>RSS</th>
<th>STATE</th>
<th>PRI</th>
<th>NICE</th>
<th>TIME</th>
<th>CPU</th>
<th>PROCESS/NLWP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2597</td>
<td>ks130310</td>
<td>4280K</td>
<td>2304K</td>
<td>cpu1</td>
<td>0</td>
<td>0</td>
<td>0:01:25</td>
<td>22%</td>
<td>imapd/1</td>
</tr>
<tr>
<td>29195</td>
<td>bc21502</td>
<td>4808K</td>
<td>4160K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:05:26</td>
<td>1.9%</td>
<td>imapd/1</td>
</tr>
<tr>
<td>3469</td>
<td>tjobson</td>
<td>6304K</td>
<td>5688K</td>
<td>sleep</td>
<td>53</td>
<td>0</td>
<td>0:00:03</td>
<td>1.0%</td>
<td>imapd/1</td>
</tr>
<tr>
<td>3988</td>
<td>tja</td>
<td>8480K</td>
<td>7864K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:01:53</td>
<td>0.5%</td>
<td>imapd/1</td>
</tr>
<tr>
<td>5173</td>
<td>root</td>
<td>2624K</td>
<td>2200K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>11:07:17</td>
<td>0.4%</td>
<td>nfsd/18</td>
</tr>
<tr>
<td>2528</td>
<td>root</td>
<td>5328K</td>
<td>3240K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>19:06:20</td>
<td>0.4%</td>
<td>automountd/2</td>
</tr>
<tr>
<td>175</td>
<td>root</td>
<td>4152K</td>
<td>3608K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>5:38:27</td>
<td>0.2%</td>
<td>ypserv/1</td>
</tr>
<tr>
<td>4795</td>
<td>snoqueen</td>
<td>5288K</td>
<td>4664K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:19</td>
<td>0.2%</td>
<td>imapd/1</td>
</tr>
<tr>
<td>3580</td>
<td>mauroj</td>
<td>4888K</td>
<td>4624K</td>
<td>cpu3</td>
<td>49</td>
<td>0</td>
<td>0:00:00</td>
<td>0.2%</td>
<td>prstat/1</td>
</tr>
<tr>
<td>1365</td>
<td>bf117072</td>
<td>3448K</td>
<td>2784K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:01</td>
<td>0.1%</td>
<td>imapd/1</td>
</tr>
<tr>
<td>8002</td>
<td>root</td>
<td>23M</td>
<td>23M</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>2:07:21</td>
<td>0.1%</td>
<td>esd/1</td>
</tr>
<tr>
<td>3598</td>
<td>wabbott</td>
<td>3512K</td>
<td>2840K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.1%</td>
<td>imapd/1</td>
</tr>
<tr>
<td>25937</td>
<td>pdanner</td>
<td>4872K</td>
<td>4232K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:03</td>
<td>0.1%</td>
<td>imapd/1</td>
</tr>
<tr>
<td>11130</td>
<td>smalm</td>
<td>5336K</td>
<td>4720K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:08</td>
<td>0.1%</td>
<td>imapd/1</td>
</tr>
</tbody>
</table>
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\001\0\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\001\0\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\001\0\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\001\0\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
```
# lockstat -kIi997 sleep 10

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

<table>
<thead>
<tr>
<th>Count</th>
<th>indv</th>
<th>cuml</th>
<th>rcnt</th>
<th>nsec</th>
<th>CPU+PIL</th>
<th>Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>5122</td>
<td>48%</td>
<td>48%</td>
<td>1.00</td>
<td>1419</td>
<td>cpu[0]</td>
<td>default_copyout</td>
</tr>
<tr>
<td>1292</td>
<td>12%</td>
<td>61%</td>
<td>1.00</td>
<td>1177</td>
<td>cpu[1]</td>
<td>splx</td>
</tr>
<tr>
<td>1288</td>
<td>12%</td>
<td>73%</td>
<td>1.00</td>
<td>1118</td>
<td>cpu[1]</td>
<td>idle</td>
</tr>
<tr>
<td>911</td>
<td>9%</td>
<td>81%</td>
<td>1.00</td>
<td>1169</td>
<td>cpu[1]</td>
<td>disp_getwork</td>
</tr>
<tr>
<td>695</td>
<td>7%</td>
<td>88%</td>
<td>1.00</td>
<td>1170</td>
<td>cpu[1]</td>
<td>i_ddi_splhigh</td>
</tr>
<tr>
<td>440</td>
<td>4%</td>
<td>92%</td>
<td>1.00</td>
<td>1163</td>
<td>cpu[1]+11</td>
<td>splx</td>
</tr>
<tr>
<td>414</td>
<td>4%</td>
<td>96%</td>
<td>1.00</td>
<td>1163</td>
<td>cpu[1]+11</td>
<td>i_ddi_splhigh</td>
</tr>
<tr>
<td>254</td>
<td>2%</td>
<td>98%</td>
<td>1.00</td>
<td>1176</td>
<td>cpu[1]+11</td>
<td>disp_getwork</td>
</tr>
<tr>
<td>27</td>
<td>0%</td>
<td>99%</td>
<td>1.00</td>
<td>1349</td>
<td>cpu[0]</td>
<td>uiomove</td>
</tr>
<tr>
<td>27</td>
<td>0%</td>
<td>99%</td>
<td>1.00</td>
<td>1624</td>
<td>cpu[0]</td>
<td>bzero</td>
</tr>
<tr>
<td>24</td>
<td>0%</td>
<td>99%</td>
<td>1.00</td>
<td>1205</td>
<td>cpu[0]</td>
<td>mmrw</td>
</tr>
<tr>
<td>21</td>
<td>0%</td>
<td>99%</td>
<td>1.00</td>
<td>1870</td>
<td>cpu[0]</td>
<td>(usermode)</td>
</tr>
<tr>
<td>9</td>
<td>0%</td>
<td>99%</td>
<td>1.00</td>
<td>1174</td>
<td>cpu[0]</td>
<td>xcopyout</td>
</tr>
<tr>
<td>8</td>
<td>0%</td>
<td>99%</td>
<td>1.00</td>
<td>650</td>
<td>cpu[0]</td>
<td>kt10</td>
</tr>
<tr>
<td>6</td>
<td>0%</td>
<td>99%</td>
<td>1.00</td>
<td>1220</td>
<td>cpu[0]</td>
<td>mutex_enter</td>
</tr>
<tr>
<td>5</td>
<td>0%</td>
<td>99%</td>
<td>1.00</td>
<td>1236</td>
<td>cpu[0]</td>
<td>default_xcopyout</td>
</tr>
<tr>
<td>3</td>
<td>0%</td>
<td>100%</td>
<td>1.00</td>
<td>1383</td>
<td>cpu[0]</td>
<td>write</td>
</tr>
<tr>
<td>3</td>
<td>0%</td>
<td>100%</td>
<td>1.00</td>
<td>1330</td>
<td>cpu[0]</td>
<td>getminor</td>
</tr>
<tr>
<td>3</td>
<td>0%</td>
<td>100%</td>
<td>1.00</td>
<td>333</td>
<td>cpu[0]</td>
<td>utl10</td>
</tr>
<tr>
<td>2</td>
<td>0%</td>
<td>100%</td>
<td>1.00</td>
<td>961</td>
<td>cpu[0]</td>
<td>mmread</td>
</tr>
<tr>
<td>2</td>
<td>0%</td>
<td>100%</td>
<td>1.00</td>
<td>2000</td>
<td>cpu[0]+10</td>
<td>read_RTC</td>
</tr>
</tbody>
</table>
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)
<table>
<thead>
<tr>
<th>CPU</th>
<th>itlb-miss %tim</th>
<th>itsb-miss %tim</th>
<th>dtlb-miss %tim</th>
<th>dtsb-miss %tim</th>
<th>%tim</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 u</td>
<td>360 0.0</td>
<td>0 0.0</td>
<td>324 0.0</td>
<td>0 0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0 k</td>
<td>44 0.0</td>
<td>0 0.0</td>
<td>21517 1.1</td>
<td>175 0.0</td>
<td>1.1</td>
</tr>
<tr>
<td>1 u</td>
<td>2680 0.1</td>
<td>0 0.0</td>
<td>10538 0.5</td>
<td>12 0.0</td>
<td>0.6</td>
</tr>
<tr>
<td>1 k</td>
<td>111 0.0</td>
<td>0 0.0</td>
<td>11932 0.7</td>
<td>196 0.1</td>
<td>0.7</td>
</tr>
<tr>
<td>4 u</td>
<td>3617 0.2</td>
<td>2 0.0</td>
<td>28658 1.3</td>
<td>187 0.0</td>
<td>1.5</td>
</tr>
<tr>
<td>4 k</td>
<td>96 0.0</td>
<td>0 0.0</td>
<td>14462 0.8</td>
<td>173 0.1</td>
<td>0.8</td>
</tr>
<tr>
<td>5 u</td>
<td>2157 0.1</td>
<td>7 0.0</td>
<td>16055 0.7</td>
<td>1023 0.2</td>
<td>1.0</td>
</tr>
<tr>
<td>5 k</td>
<td>91 0.0</td>
<td>0 0.0</td>
<td>12987 0.7</td>
<td>142 0.0</td>
<td>0.7</td>
</tr>
<tr>
<td>8 u</td>
<td>1030 0.1</td>
<td>0 0.0</td>
<td>2102 0.1</td>
<td>0 0.0</td>
<td>0.2</td>
</tr>
<tr>
<td>8 k</td>
<td>124 0.0</td>
<td>1 0.0</td>
<td>11452 0.6</td>
<td>76 0.0</td>
<td>0.6</td>
</tr>
<tr>
<td>9 u</td>
<td>7739 0.3</td>
<td>15 0.0</td>
<td>112351 4.9</td>
<td>664 0.1</td>
<td>5.3</td>
</tr>
<tr>
<td>9 k</td>
<td>78 0.0</td>
<td>3 0.0</td>
<td>65578 3.2</td>
<td>2440 0.6</td>
<td>3.8</td>
</tr>
<tr>
<td>12 u</td>
<td>1398 0.1</td>
<td>5 0.0</td>
<td>8603 0.4</td>
<td>146 0.0</td>
<td>0.5</td>
</tr>
<tr>
<td>12 k</td>
<td>156 0.0</td>
<td>4 0.0</td>
<td>13471 0.7</td>
<td>216 0.1</td>
<td>0.8</td>
</tr>
<tr>
<td>13 u</td>
<td>303 0.0</td>
<td>0 0.0</td>
<td>346 0.0</td>
<td>0 0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>13 k</td>
<td>10 0.0</td>
<td>0 0.0</td>
<td>27234 1.4</td>
<td>153 0.0</td>
<td>1.4</td>
</tr>
<tr>
<td>ttl</td>
<td>19994 0.1</td>
<td>37 0.0</td>
<td>357610 2.1</td>
<td>5603 0.2</td>
<td>2.4</td>
</tr>
</tbody>
</table>
The *stat Utilities

- `mpstat(1)`
  - System-wide view of CPU activity
- `vmstat(1)`
  - Memory statistics
  - Don't forget “vmstat -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

```bash
solaris> cputrack -N 20 -c pic0=DC_access,pic1=DC_miss -p 19849

<table>
<thead>
<tr>
<th>time</th>
<th>lwp</th>
<th>event</th>
<th>pic0</th>
<th>pic1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.007</td>
<td>1</td>
<td>tick</td>
<td>34543793</td>
<td>824363</td>
</tr>
<tr>
<td>1.007</td>
<td>2</td>
<td>tick</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.007</td>
<td>3</td>
<td>tick</td>
<td>1001797338</td>
<td>5153245</td>
</tr>
<tr>
<td>1.015</td>
<td>4</td>
<td>tick</td>
<td>976864106</td>
<td>5536858</td>
</tr>
<tr>
<td>1.007</td>
<td>5</td>
<td>tick</td>
<td>1002880440</td>
<td>5217810</td>
</tr>
<tr>
<td>1.017</td>
<td>6</td>
<td>tick</td>
<td>948543113</td>
<td>3731144</td>
</tr>
<tr>
<td>2.007</td>
<td>1</td>
<td>tick</td>
<td>15425817</td>
<td>745468</td>
</tr>
<tr>
<td>2.007</td>
<td>2</td>
<td>tick</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2.014</td>
<td>3</td>
<td>tick</td>
<td>1002035102</td>
<td>5110169</td>
</tr>
<tr>
<td>2.017</td>
<td>4</td>
<td>tick</td>
<td>976879154</td>
<td>5542155</td>
</tr>
<tr>
<td>2.030</td>
<td>5</td>
<td>tick</td>
<td>1018802136</td>
<td>5283137</td>
</tr>
<tr>
<td>2.033</td>
<td>6</td>
<td>tick</td>
<td>1013933228</td>
<td>4072636</td>
</tr>
</tbody>
</table>

......

solaris> bc -l
824363/34543793
.02386428728310177171
((100−(824363/34543793)))
99.97613571271689822829
```
# cpustat -h
Usage:

    cpustat [-c events] [-p period] [-nstD] [interval [count]]

- **c** events specify processor events to be monitored
- **n** suppress titles
- **p** period cycle through event list periodically
- **s** run user soaker thread for system-only events
- **t** include %tick register
- **D** enable debug mode
- **h** print extended usage information

Use cputrack(1) to monitor per-process statistics.

CPU performance counter interface: SPARC64 VI

event specification syntax:
[picn]=<eventn>[.attr[n]=[<val>]][[.picn]=<eventn>[.attr[n]=[<val>]],...]

event0:  cycle_counts instruction_counts op_stv_wait
    load_store_instructions branch_instructions
    floating_instructions impdep2_instructions
    prefetch_instructions flush_rs 2iid_use toq_rsbr_phantom
    trap_int_vector ts_by_sxmiss active_cycle_count
    op_stv_wait_sxmiss eu_comp_wait swpf_fail_all
    sx_miss_wait_pf jbus_cpi_count jbus_reqbus1_busy

event1:  cycle_counts instruction_counts instruction_flow_counts
    iwr_empty op_stv_wait load_store_instructions
    branch_instructions floating_instructions
    . . . .
# cpustat -c pic0=cycle_counts,pic1=instruction_counts,sys

```
<table>
<thead>
<tr>
<th>time</th>
<th>cpu</th>
<th>event</th>
<th>pic0</th>
<th>pic1</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.011</td>
<td>8</td>
<td>tick</td>
<td>11411962932</td>
<td>5348741180</td>
</tr>
<tr>
<td>5.011</td>
<td>131</td>
<td>tick</td>
<td>11400819057</td>
<td>5249797028</td>
</tr>
<tr>
<td>5.011</td>
<td>130</td>
<td>tick</td>
<td>11400858896</td>
<td>5244266999</td>
</tr>
<tr>
<td>5.011</td>
<td>16</td>
<td>tick</td>
<td>11407356349</td>
<td>5303050712</td>
</tr>
<tr>
<td>5.011</td>
<td>10</td>
<td>tick</td>
<td>11409702171</td>
<td>5344586657</td>
</tr>
<tr>
<td>5.011</td>
<td>18</td>
<td>tick</td>
<td>11407295550</td>
<td>5306963656</td>
</tr>
<tr>
<td>5.011</td>
<td>19</td>
<td>tick</td>
<td>11406349340</td>
<td>5292477138</td>
</tr>
<tr>
<td>5.011</td>
<td>0</td>
<td>tick</td>
<td>11412859729</td>
<td>5222752733</td>
</tr>
<tr>
<td>5.011</td>
<td>17</td>
<td>tick</td>
<td>11408093975</td>
<td>5307307043</td>
</tr>
<tr>
<td>5.011</td>
<td>26</td>
<td>tick</td>
<td>11403459560</td>
<td>5254359643</td>
</tr>
<tr>
<td>5.011</td>
<td>144</td>
<td>tick</td>
<td>11394770612</td>
<td>5325801245</td>
</tr>
<tr>
<td>5.011</td>
<td>24</td>
<td>tick</td>
<td>11403518595</td>
<td>5256957295</td>
</tr>
<tr>
<td>5.011</td>
<td>128</td>
<td>tick</td>
<td>11397600354</td>
<td>5234695931</td>
</tr>
<tr>
<td>5.012</td>
<td>1</td>
<td>tick</td>
<td>11414392475</td>
<td>5284187266</td>
</tr>
<tr>
<td>5.011</td>
<td>137</td>
<td>tick</td>
<td>11397641918</td>
<td>5313760153</td>
</tr>
<tr>
<td>5.011</td>
<td>11</td>
<td>tick</td>
<td>11410206642</td>
<td>5347201300</td>
</tr>
<tr>
<td>5.011</td>
<td>27</td>
<td>tick</td>
<td>11402593843</td>
<td>5285054790</td>
</tr>
</tbody>
</table>
```

```
# bc -l
11394446629/5320324508
2.14168263831774526036
```
Applying The Tools - Example
Start with a System View

What jumps out at us...

- Processors a fully utilized, 90% sys
  - Question: Where is the kernel spending time?
- syscalls-per-second are high
  - Question: What are these system calls, and where are they coming from
- mutex's per second are high
  - Question: Which mutex locks, and why?

```
# mpstat 1
CPU  minf mjf xcal  intr ithr  csww  icsw  migr  smtx  srw  syscl  usr  sys  wt  idl
0    0    0    294  329  227  117  60   12    40597  0  245787  10  90   0   0
1    11   0    0    141   4   73   41   12    37736  0  244729  11  89   0   0
2    0    0    0    140   2   64   37   1    34046  0  243383  10  90   0   0
3    0    0    0    130   0   49   32   2    31666  0  243440  10  90   0   0
CPU  minf mjf xcal  intr ithr  csww  icsw  migr  smtx  srw  syscl  usr  sys  wt  idl
0    0    0    16    432  230  149   68   25    42514  25  250163 10  90   0   0
1    0    0   100    122   5  117   55   26    38418  25  250163 10  90   0   0
2    0    0   129    103   2  124   53   12    34029 12  244908  9  91   0   0
3    0    0    24    123   0  110   45   6    30893 18  242016 10  90   0   0
```
Processor – kernel profile

# lockstat -i997 -Ik sleep 30

Profiling interrupt: 119780 events in 30.034 seconds (3988 events/sec)

<table>
<thead>
<tr>
<th>Count</th>
<th>indiv</th>
<th>cuml</th>
<th>rcnt</th>
<th>nsec</th>
<th>CPU+PIL</th>
<th>Hottest Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>29912</td>
<td>25%</td>
<td>25%</td>
<td>0.00</td>
<td>5461</td>
<td>cpu[2]</td>
<td>kcopy</td>
</tr>
<tr>
<td>29894</td>
<td>25%</td>
<td>50%</td>
<td>0.00</td>
<td>5470</td>
<td>cpu[1]</td>
<td>kcopy</td>
</tr>
<tr>
<td>29876</td>
<td>25%</td>
<td>75%</td>
<td>0.00</td>
<td>5401</td>
<td>cpu[3]</td>
<td>kcopy</td>
</tr>
<tr>
<td>29752</td>
<td>25%</td>
<td>100%</td>
<td>0.00</td>
<td>5020</td>
<td>cpu[0]</td>
<td>kcopy</td>
</tr>
<tr>
<td>119</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>1689</td>
<td>cpu[0]+10</td>
<td>dosoftint</td>
</tr>
<tr>
<td>71</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>1730</td>
<td>cpu[0]+11</td>
<td>sleepq_wakeone_chan</td>
</tr>
<tr>
<td>45</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>5209</td>
<td>cpu[1]+11</td>
<td>lock_play</td>
</tr>
<tr>
<td>39</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>4024</td>
<td>cpu[3]+11</td>
<td>lock_set_spl</td>
</tr>
<tr>
<td>33</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>5156</td>
<td>cpu[2]+11</td>
<td>setbackdq</td>
</tr>
<tr>
<td>30</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>3790</td>
<td>cpu[3]+2</td>
<td>dosoftint</td>
</tr>
<tr>
<td>6</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>5600</td>
<td>cpu[1]+5</td>
<td>ddi_io_getb</td>
</tr>
<tr>
<td>3</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>1072</td>
<td>cpu[0]+2</td>
<td>apic_redistribute_compute</td>
</tr>
</tbody>
</table>

# dtrace -n 'profile-997ms / arg0 != 0 / { @ks[stack()] = count() }'

dtrace: description 'profile-997ms ' matched 1 probe
^C

genunix`syscall_mstate+0x1c7
unix`sys씁call32+0xbd
  1

unix`bzero+0x3
procfs`pr_read_lwpusage_32+0x2f
procfs`prread+0x5d
ugenunix`fop_read+0x29
ugenunix`pread+0x217
ugenunix`pread32+0x26
unix`sysSubstring+0x101
  1
[Continue from previous slide – dtrace stack() aggregation output...]

......
unix`kcopy+0x38
genunix`copyin_nowatch+0x48
genunix`copyin_args32+0x45
genunix`syscall_entry+0xcb
unix`sys_syscall32+0xe1
   1

unix`sys_syscall32+0xae
   1

unix`mutex_exit+0x19
ufs`rdip+0x368
ufs`ufs_read+0x1a6
genunix`fop_read+0x29
genunix`pread64+0x1d7
unix`sys_syscall32+0x101
   2

unix`kcopy+0x2c
genunix`uiomove+0x17f
ufs`rdip+0x382
ufs`ufs_read+0x1a6
genunix`fop_read+0x29
genunix`pread64+0x1d7
unix`sys_syscall32+0x101
   13
Another Kernel Stack View

```
# lockstat -i997 -Ikws 10 sleep 30

Profiling interrupt: 119800 events in 30.038 seconds (3988 events/sec)

<table>
<thead>
<tr>
<th>Count</th>
<th>indv</th>
<th>cuml</th>
<th>rcnt</th>
<th>nsec</th>
<th>CPU+PIL</th>
<th>Hottest Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>29919</td>
<td>25%</td>
<td>25%</td>
<td>0.00</td>
<td>5403</td>
<td>cpu[2]</td>
<td>kcopy</td>
</tr>
</tbody>
</table>

nsec ------ Time Distribution ------ count     Stack
1024 |                               2         uiomove
2048 |                               18        rdip
4096 |                               25        ufs_read
8192 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  29853     fop_read
16384 |                               21        pread64
                      sys_syscall32

<table>
<thead>
<tr>
<th>Count</th>
<th>indv</th>
<th>cuml</th>
<th>rcnt</th>
<th>nsec</th>
<th>CPU+PIL</th>
<th>Hottest Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>29918</td>
<td>25%</td>
<td>50%</td>
<td>0.00</td>
<td>5386</td>
<td>cpu[1]</td>
<td>kcopy</td>
</tr>
</tbody>
</table>

nsec ------ Time Distribution ------ count     Stack
4096 |                               38        uiomove
8192 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  29870     rdip
16384 |                               10        ufs_read
                      fop_read
                      pread64
                      sys_syscall32

<table>
<thead>
<tr>
<th>Count</th>
<th>indv</th>
<th>cuml</th>
<th>rcnt</th>
<th>nsec</th>
<th>CPU+PIL</th>
<th>Hottest Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>29893</td>
<td>25%</td>
<td>75%</td>
<td>0.00</td>
<td>5283</td>
<td>cpu[3]</td>
<td>kcopy</td>
</tr>
</tbody>
</table>

nsec ------ Time Distribution ------ count     Stack
1024 |                               140       uiomove
2048 |                               761       rdip
4096 |                               1443      ufs_read
8192 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  27532     fop_read
16384 |                               17        pread64
                      sys_syscall32
```

LISA ’09 Baltimore, Md.
Who's Doing What...

```bash
# prstat -Lmc 10 10 > prstat.out
# cat prstat.out

PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID
4448 root 12 44 0.0 0.0 0.0 0.0 0.0 43 0.5 2K 460 .1M 0 prstat/1
4447 root 1.2 11 0.0 0.0 0.0 0.0 0.1 14 73 54 65 .2M 0 filebench/27
4447 root 1.1 10 0.0 0.0 0.0 0.0 0.1 15 74 57 52 .2M 0 filebench/29
4447 root 1.1 10 0.0 0.0 0.0 0.1 0.0 15 74 64 53 .2M 0 filebench/19
4447 root 1.1 10 0.0 0.0 0.0 0.0 0.4 14 74 49 55 .2M 0 filebench/7
4447 root 1.1 10 0.0 0.0 0.0 0.0 0.2 14 74 51 44 .2M 0 filebench/17
4447 root 1.1 9.9 0.0 0.0 0.0 0.0 0.3 14 74 48 57 .2M 0 filebench/14
4447 root 1.1 9.9 0.0 0.0 0.0 0.0 0.3 14 74 42 61 .2M 0 filebench/9
4447 root 1.1 9.8 0.0 0.0 0.0 0.0 0.1 15 74 51 49 .2M 0 filebench/25
4447 root 1.1 9.8 0.0 0.0 0.0 0.0 0.0 15 74 60 38 .2M 0 filebench/4
4447 root 1.1 9.7 0.0 0.0 0.0 0.0 0.2 14 75 25 69 .2M 0 filebench/26
4447 root 1.0 9.7 0.0 0.0 0.0 0.1 0.0 15 75 54 46 .2M 0 filebench/12
4447 root 1.1 9.6 0.0 0.0 0.0 0.0 0.3 14 75 40 46 .2M 0 filebench/21
4447 root 1.1 9.6 0.0 0.0 0.0 0.0 0.1 15 75 39 70 .2M 0 filebench/31
4447 root 1.1 9.6 0.0 0.0 0.0 0.0 0.1 15 75 38 75 .2M 0 filebench/22

Total: 59 processes, 218 lwps, load averages: 9.02, 14.30, 10.36

PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID
4447 root 1.3 12 0.0 0.0 0.0 0.0 0.0 0.0 86 43 41 .3M 0 filebench/16
4447 root 1.3 12 0.0 0.0 0.0 0.0 0.0 0.0 87 35 46 .3M 0 filebench/14
4447 root 1.3 12 0.0 0.0 0.0 0.0 0.0 0.0 87 36 60 .3M 0 filebench/7
4447 root 1.3 12 0.0 0.0 0.0 0.0 0.0 0.0 87 27 44 .3M 0 filebench/24
4447 root 1.3 12 0.0 0.0 0.0 0.0 0.0 0.0 87 41 61 .3M 0 filebench/3
4447 root 1.3 12 0.0 0.0 0.0 0.0 0.0 0.0 87 38 49 .3M 0 filebench/13
4447 root 1.3 12 0.0 0.0 0.0 0.0 0.0 0.0 87 14 71 .3M 0 filebench/2
4447 root 1.3 12 0.0 0.0 0.0 0.0 0.0 0.0 87 32 57 .3M 0 filebench/19
4447 root 1.3 12 0.0 0.0 0.0 0.0 0.0 0.0 87 31 57 .3M 0 filebench/27
4447 root 1.3 12 0.0 0.0 0.0 0.0 0.0 0.0 87 34 47 .3M 0 filebench/4
4447 root 1.3 11 0.0 0.0 0.0 0.0 0.0 0.0 87 21 74 .3M 0 filebench/26
4447 root 1.2 11 0.0 0.0 0.0 0.0 0.0 0.0 87 42 51 .3M 0 filebench/9
4447 root 1.3 11 0.0 0.0 0.0 0.0 0.0 0.0 87 16 83 .3M 0 filebench/18
4447 root 1.2 11 0.0 0.0 0.0 0.0 0.0 0.0 87 42 47 .3M 0 filebench/33
4447 root 1.2 11 0.0 0.0 0.0 0.0 0.0 0.0 87 15 76 .3M 0 filebench/15

Total: 59 processes, 218 lwps, load averages: 12.54, 14.88, 10.59

LISA '09 Baltimore, Md.
System Calls – What & Who

# dtrace -n 'syscall:::entry { @sc[probefunc]=count() }'
dtrace: description 'syscall:::entry ' matched 228 probes
^C

fstat 1
mmap 1
schedctl 1
waitsys 1
recvmsg 2
sigaction 2
sysconfig 3
brk 6
pset 9
gtime 16
lwp_park 20
p_online 21
setcontext 29
write 30
nanosleep 32
lwp_sigmask 45
setitimer 54
pollsys 118
ioctl 427
pread64 1583439
pread 3166885
read 3166955

# dtrace -n 'syscall::read:entry { @[execname,pid]=count() }'
dtrace: description 'syscall::read:entry ' matched 1 probe
^C

sshd 4342
Xorg 536
filebench 4376

^C
smtx – Lock Operations

Adaptive mutex spin: 3486197 events in 30.031 seconds (116088 events/sec)

<table>
<thead>
<tr>
<th>Count</th>
<th>indv</th>
<th>cuml</th>
<th>rcnt</th>
<th>spin</th>
<th>Lock</th>
<th>Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>1499963</td>
<td>43%</td>
<td>43%</td>
<td>0.00</td>
<td>84</td>
<td>pr_pidlock</td>
<td>pr_p_lock+0x29</td>
</tr>
<tr>
<td>1101187</td>
<td>32%</td>
<td>75%</td>
<td>0.00</td>
<td>24</td>
<td>0xffffffff810cdec0</td>
<td>pr_p_lock+0x50</td>
</tr>
<tr>
<td>285012</td>
<td>8%</td>
<td>83%</td>
<td>0.00</td>
<td>27</td>
<td>0xffffffff827a9858</td>
<td>rdip+0x506</td>
</tr>
<tr>
<td>212621</td>
<td>6%</td>
<td>89%</td>
<td>0.00</td>
<td>29</td>
<td>0xffffffff827a9858</td>
<td>rdip+0x134</td>
</tr>
<tr>
<td>98531</td>
<td>3%</td>
<td>92%</td>
<td>0.00</td>
<td>103</td>
<td>0xffffffff9321d480</td>
<td>releasef+0x55</td>
</tr>
<tr>
<td>92486</td>
<td>3%</td>
<td>94%</td>
<td>0.00</td>
<td>19</td>
<td>0xffffffff8d5c4990</td>
<td>ufs_lockfs_end+0x81</td>
</tr>
<tr>
<td>89404</td>
<td>3%</td>
<td>97%</td>
<td>0.00</td>
<td>27</td>
<td>0xffffffff8d5c4990</td>
<td>ufs_lockfs_begin+0x9f</td>
</tr>
<tr>
<td>83186</td>
<td>2%</td>
<td>99%</td>
<td>0.00</td>
<td>96</td>
<td>0xffffffff9321d480</td>
<td>getf+0x5d</td>
</tr>
<tr>
<td>6356</td>
<td>0%</td>
<td>99%</td>
<td>0.00</td>
<td>186</td>
<td>0xffffffff810cdec0</td>
<td>clock+0x4e9</td>
</tr>
<tr>
<td>1164</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>141</td>
<td>0xffffffff810cdec0</td>
<td>post_syscall+0x352</td>
</tr>
<tr>
<td>294</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>11</td>
<td>0xffffffff801a4008</td>
<td>segmap_smapadd+0x77</td>
</tr>
<tr>
<td>279</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>11</td>
<td>0xffffffff801a41d0</td>
<td>segmap_getmapflt+0x275</td>
</tr>
<tr>
<td>278</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>11</td>
<td>0xffffffff801a48f0</td>
<td>segmap_smapadd+0x77</td>
</tr>
<tr>
<td>276</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>11</td>
<td>0xffffffff801a5010</td>
<td>segmap_getmapflt+0x275</td>
</tr>
<tr>
<td>276</td>
<td>0%</td>
<td>100%</td>
<td>0.00</td>
<td>11</td>
<td>0xffffffff801a4008</td>
<td>segmap_getmapflt+0x275</td>
</tr>
</tbody>
</table>

Adaptive mutex block: 3328 events in 30.031 seconds (111 events/sec)

<table>
<thead>
<tr>
<th>Count</th>
<th>indv</th>
<th>cuml</th>
<th>rcnt</th>
<th>nsec</th>
<th>Lock</th>
<th>Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>1929</td>
<td>58%</td>
<td>58%</td>
<td>0.00</td>
<td>48944759</td>
<td>pr_pidlock</td>
<td>pr_p_lock+0x29</td>
</tr>
<tr>
<td>263</td>
<td>8%</td>
<td>66%</td>
<td>0.00</td>
<td>47017</td>
<td>0xffffffff810cdec0</td>
<td>pr_p_lock+0x50</td>
</tr>
<tr>
<td>255</td>
<td>8%</td>
<td>74%</td>
<td>0.00</td>
<td>53392369</td>
<td>0xffffffff9321d480</td>
<td>getf+0x5d</td>
</tr>
<tr>
<td>217</td>
<td>7%</td>
<td>80%</td>
<td>0.00</td>
<td>26133</td>
<td>0xffffffff810cdec0</td>
<td>clock+0x4e9</td>
</tr>
<tr>
<td>207</td>
<td>6%</td>
<td>86%</td>
<td>0.00</td>
<td>227146</td>
<td>0xffffffff827a9858</td>
<td>rdip+0x134</td>
</tr>
<tr>
<td>197</td>
<td>6%</td>
<td>92%</td>
<td>0.00</td>
<td>64467</td>
<td>0xffffffff8d5c4990</td>
<td>ufs_lockfs_begin+0x9f</td>
</tr>
<tr>
<td>122</td>
<td>4%</td>
<td>96%</td>
<td>0.00</td>
<td>64664</td>
<td>0xffffffff8d5c4990</td>
<td>ufs_lockfs_end+0x81</td>
</tr>
<tr>
<td>112</td>
<td>3%</td>
<td>99%</td>
<td>0.00</td>
<td>164559</td>
<td>0xffffffff827a9858</td>
<td>rdip+0x506</td>
</tr>
</tbody>
</table>
smtx – Lock Operations (cont)

Spin lock spin: 3491 events in 30.031 seconds (116 events/sec)

<table>
<thead>
<tr>
<th>Count</th>
<th>indv</th>
<th>cuml</th>
<th>rcnt</th>
<th>spin Lock</th>
<th>Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>2197</td>
<td>63%</td>
<td>63%</td>
<td>0.00</td>
<td>2151 turnstile_table+0xbd8</td>
<td>disp_lock_enter+0x35</td>
</tr>
<tr>
<td>314</td>
<td>9%</td>
<td>72%</td>
<td>0.00</td>
<td>3129 turnstile_table+0xe28</td>
<td>disp_lock_enter+0x35</td>
</tr>
<tr>
<td>296</td>
<td>8%</td>
<td>80%</td>
<td>0.00</td>
<td>3162 turnstile_table+0x888</td>
<td>disp_lock_enter+0x35</td>
</tr>
<tr>
<td>211</td>
<td>6%</td>
<td>86%</td>
<td>0.00</td>
<td>3162 turnstile_table+0x8a8</td>
<td>disp_lock_enter+0x35</td>
</tr>
<tr>
<td>127</td>
<td>4%</td>
<td>90%</td>
<td>0.00</td>
<td>856  turnstile_table+0x9f8</td>
<td>turnstile_interlock+0x171</td>
</tr>
<tr>
<td>114</td>
<td>3%</td>
<td>93%</td>
<td>0.00</td>
<td>269  turnstile_table+0x9f8</td>
<td>disp_lock_enter+0x35</td>
</tr>
<tr>
<td>44</td>
<td>1%</td>
<td>95%</td>
<td>0.00</td>
<td>90   0xffffffff827f4de0</td>
<td>disp_lock_enter_high+0x13</td>
</tr>
<tr>
<td>37</td>
<td>1%</td>
<td>96%</td>
<td>0.00</td>
<td>581  0xffffffff827f4de0</td>
<td>disp_lock_enter+0x35</td>
</tr>
</tbody>
</table>

Thread lock spin: 1104 events in 30.031 seconds (37 events/sec)

<table>
<thead>
<tr>
<th>Count</th>
<th>indv</th>
<th>cuml</th>
<th>rcnt</th>
<th>spin Lock</th>
<th>Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>487</td>
<td>44%</td>
<td>44%</td>
<td>0.00</td>
<td>1671 turnstile_table+0xbd8</td>
<td>ts_tick+0x26</td>
</tr>
<tr>
<td>219</td>
<td>20%</td>
<td>64%</td>
<td>0.00</td>
<td>1510 turnstile_table+0xbd8</td>
<td>turnstile_block+0x387</td>
</tr>
<tr>
<td>92</td>
<td>8%</td>
<td>72%</td>
<td>0.00</td>
<td>1941 turnstile_table+0xe8a8</td>
<td>ts_tick+0x26</td>
</tr>
<tr>
<td>77</td>
<td>7%</td>
<td>79%</td>
<td>0.00</td>
<td>2037 turnstile_table+0xe28</td>
<td>ts_tick+0x26</td>
</tr>
<tr>
<td>74</td>
<td>7%</td>
<td>86%</td>
<td>0.00</td>
<td>2296 turnstile_table+0x888</td>
<td>ts_tick+0x26</td>
</tr>
<tr>
<td>36</td>
<td>3%</td>
<td>89%</td>
<td>0.00</td>
<td>292  cpu[0]+0xf8</td>
<td>ts_tick+0x26</td>
</tr>
<tr>
<td>27</td>
<td>2%</td>
<td>92%</td>
<td>0.00</td>
<td>55   cpu[1]+0xf8</td>
<td>ts_tick+0x26</td>
</tr>
<tr>
<td>11</td>
<td>1%</td>
<td>93%</td>
<td>0.00</td>
<td>26   cpu[3]+0xf8</td>
<td>ts_tick+0x26</td>
</tr>
<tr>
<td>10</td>
<td>1%</td>
<td>94%</td>
<td>0.00</td>
<td>11   cpu[2]+0xf8</td>
<td>post_syscall+0x556</td>
</tr>
<tr>
<td>Count</td>
<td>indiv</td>
<td>cuml</td>
<td>rcnt</td>
<td>nsec</td>
<td>Lock</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>17</td>
<td>100%</td>
<td>100%</td>
<td>0.00</td>
<td>465308</td>
<td>0xffffffff831f3be0</td>
</tr>
</tbody>
</table>

R/W writer blocked by readers: 55 events in 30.031 seconds (2 events/sec)

<table>
<thead>
<tr>
<th>Count</th>
<th>indiv</th>
<th>cuml</th>
<th>rcnt</th>
<th>nsec</th>
<th>Lock</th>
<th>Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>100%</td>
<td>100%</td>
<td>0.00</td>
<td>1232132</td>
<td>0xffffffff831f3be0</td>
<td>ufs_getpage+0x369</td>
</tr>
</tbody>
</table>

R/W reader blocked by writer: 22 events in 30.031 seconds (1 events/sec)

<table>
<thead>
<tr>
<th>Count</th>
<th>indiv</th>
<th>cuml</th>
<th>rcnt</th>
<th>nsec</th>
<th>Lock</th>
<th>Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>82%</td>
<td>82%</td>
<td>0.00</td>
<td>56339</td>
<td>0xffffffff831f3be0</td>
<td>ufs_getpage+0x369</td>
</tr>
<tr>
<td>4</td>
<td>18%</td>
<td>100%</td>
<td>0.00</td>
<td>45162</td>
<td>0xffffffff831f3be0</td>
<td>ufs_putpages+0x176</td>
</tr>
</tbody>
</table>

R/W reader blocked by write wanted: 47 events in 30.031 seconds (2 events/sec)

<table>
<thead>
<tr>
<th>Count</th>
<th>indiv</th>
<th>cuml</th>
<th>rcnt</th>
<th>nsec</th>
<th>Lock</th>
<th>Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>98%</td>
<td>98%</td>
<td>0.00</td>
<td>369379</td>
<td>0xffffffff831f3be0</td>
<td>ufs_getpage+0x369</td>
</tr>
<tr>
<td>1</td>
<td>2%</td>
<td>100%</td>
<td>0.00</td>
<td>118455</td>
<td>0xffffffff831f3be0</td>
<td>ufs_putpages+0x176</td>
</tr>
</tbody>
</table>
Chasing the hot lock caller...

```
# dtrace -n 'pr_p_lock:entry { @s[stack()]=count() }'
```
dtrace: description 'pr_p_lock:entry ' matched 1 probe
```
^C
```
```
procfs`pr_read_lwpusage_32+0x4f
procfs`prread+0x5d
genunix`fop_read+0x29
genunix`pread+0x217
genunix`pread32+0x26
unix`sys_syscall32+0x101
12266066
```
```
# dtrace -n 'pr_p_lock:entry { @s[execname]=count() }'
```
dtrace: description 'pr_p_lock:entry ' matched 1 probe
```
^C
```
```
filebench                                                   8439499
```
```
# pgrep filebench
4485
```
```
# dtrace -n 'pid4485:libc:pread:entry { @us[ustack()]=count() }'
```
dtrace: description 'pid4485:libc:pread:entry ' matched 1 probe
```
^C
```
```
libc.so.1`pread
filebench`flowop_endop+0x5b
filebench`flowoplib_read+0x238
filebench`flowop_start+0x2b1
libc.so.1`_thr_setup+0x51
libc.so.1`_lwp_start
2084651
```
```
libc.so.1`pread
filebench`flowop_beginop+0x6a
filebench`flowoplib_read+0x200
filebench`flowop_start+0x2b1
libc.so.1`_thr_setup+0x51
libc.so.1`_lwp_start
2084651
```
Icing on the cake...

# dtrace -q -n 'ufs_read:entry { printf("UFS Read: %s\n",stringof(args[0]->v_path)); }'
UFS Read: /ufs/largefile1
UFS Read: /ufs/largefile1
UFS Read: /ufs/largefile1
UFS Read: /ufs/largefile1
UFS Read: /ufs/largefile1
UFS Read: /ufs/largefile1
UFS Read: /ufs/largefile1
UFS Read: /ufs/largefile1
UFS Read: /ufs/largefile1
^c

#
#
# dtrace -q -n 'ufs_read:entry { @[execname,stringof(args[0]->v_path)]=count() }'
^C

filebench                                /ufs/largefile1
                                       864609
Session 2 - Memory
Virtual Memory

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

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Space</td>
<td>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 its own address space.</td>
</tr>
<tr>
<td>Virtual Memory</td>
<td>Illusion of real memory within an address space.</td>
</tr>
<tr>
<td>Physical Memory</td>
<td>Real memory (e.g. RAM)</td>
</tr>
<tr>
<td>Mapping</td>
<td>A memory relationship between the address space and an object managed by the virtual memory system.</td>
</tr>
<tr>
<td>Segment</td>
<td>A co-managed set of similar mappings within an address space.</td>
</tr>
<tr>
<td>Text Mapping</td>
<td>The mapping containing the program's instructions and read-only objects.</td>
</tr>
<tr>
<td>Data Mapping</td>
<td>The mapping containing the program's initialized data</td>
</tr>
<tr>
<td>Heap</td>
<td>A mapping used to contain the program's heap (malloc'd) space</td>
</tr>
<tr>
<td>Stack</td>
<td>A mapping used to hold the program's stack</td>
</tr>
<tr>
<td>Page</td>
<td>A linear chunk of memory managed by the virtual memory system</td>
</tr>
<tr>
<td>VNODE</td>
<td>A file-system independent file object within the Solaris kernel</td>
</tr>
<tr>
<td>Backing Store</td>
<td>The storage medium used to hold a page of virtual memory while it is not backed by physical memory</td>
</tr>
<tr>
<td>Paging</td>
<td>The action of moving a page to or from its backing store</td>
</tr>
<tr>
<td>Swapping</td>
<td>The action of swapping an entire address space to/from the swap device</td>
</tr>
<tr>
<td>Swap Space</td>
<td>A storage device used as the backing store for anonymous pages.</td>
</tr>
</tbody>
</table>
**Solaris Virtual Memory Glossary (cont)**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanning</td>
<td>The action of the virtual memory system takes when looking for memory which can be freed up for use by other subsystems.</td>
</tr>
<tr>
<td>Named Pages</td>
<td>Pages which are mappings of an object in the file system.</td>
</tr>
<tr>
<td>Anonymous Memory</td>
<td>Pages which do not have a named backing store.</td>
</tr>
<tr>
<td>Protection</td>
<td>A set of booleans to describe if a program is allowed to read, write or execute instructions within a page or mapping.</td>
</tr>
<tr>
<td>ISM</td>
<td>Intimate Shared Memory - A type of System V shared memory optimized for sharing between many processes</td>
</tr>
<tr>
<td>DISM</td>
<td>Pageable ISM</td>
</tr>
<tr>
<td>NUMA</td>
<td>Non-uniform memory architecture - a term used to describe a machine with differing processor-memory latencies.</td>
</tr>
<tr>
<td>Lgroup</td>
<td>A locality group - a grouping of processors and physical memory which share similar memory latencies.</td>
</tr>
<tr>
<td>MMU</td>
<td>The hardware functional unit in the microprocessor used to dynamically translate virtual addresses into physical addresses.</td>
</tr>
<tr>
<td>HAT</td>
<td>The Hardware Address Translation Layer - the Solaris layer which manages the translation of virtual addresses to physical addresses.</td>
</tr>
<tr>
<td>TTE</td>
<td>Translation Table Entry - The UltraSPARC hardware's table entry which holds the data for virtual to physical translation.</td>
</tr>
<tr>
<td>TLB</td>
<td>Translation Lookaside Buffer - the hardware's cache of virtual address translations.</td>
</tr>
<tr>
<td>Page Size</td>
<td>The translation size for each entry in the TLB.</td>
</tr>
<tr>
<td>TSB</td>
<td>Translation Software Buffer - UltraSPARC's software cache of TTEs, used for lookup when a translation is not found in the TLB.</td>
</tr>
</tbody>
</table>
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
Physical Memory Management
Memory Allocation Transitions

- Page Scanner (Bilge Pump)
- Process Allocations
- Process exit

- Segmap File Cache
  - Mapped Files
    - File delete, fs unmount

- Cache-List
  - Inactive File Pages (named files)
  - File delete, fs unmount, memcntl

- Free-List
  - Unused Memory
  - reclaim

- Kernel Internals
  - Kernel reap (low freemem)

- vmstat "free"
  - Pageout Steal
  - Reclaim

- Free
  - Allocation
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

Diagram:
- Write to backing store
- Memory Page
- Clearing bits
Clock or Callout Thread

schedpaging()
- how many pages
- how much CPU

Wake up the scanner

Page Scanner Thread

queue_io_request()

modified?

N

Free Page

checkpage()

Y

Page-out Thread

Dirty Page push list

page-out()

file system or specfs
vop_putpage() routine

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

\[
\text{scanrate} = \left( \frac{\text{lotsfree} - \text{freemem}}{\text{lotsfree}} \times \text{fastscan} \right) + \left( \text{slowscan} \times \frac{\text{freemem}}{\text{lotsfree}} \right)
\]
## Scanning Parameters

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

1 GB Example

Scan Rate

fastscan
8192

# pages scanned / second

slowscan
100

Scan Rate

0 MB
4 MB
8 MB
16 MB

Amount of Free Memory

minfree
throttlefree
desfree
lotsfree

LISA '09 Baltimore, Md.
The Solaris Page 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
- UFS
  - segmap kernel address space segment
  - Starting in Solaris 10 3/05, segkpm integration (SPARC)
- ZFS
  - Uses kernel memory (kmem_alloc) for ARC cache
The Solaris UFS Cache - segmap

Sol 8 (and beyond) segmap

Kernel
Memory

segmap

process memory
heap, data, stack

cachelist

freelist

reclaim
The Solaris Cache

• Now `vmstat` reports a useful `free`
• Throw away your old `/etc/system pager` configuration parameters
  • `lotsfree`, `desfree`, `minfree`
  • `fastscan`, `slowscan`
  • `priority_paging`, `cachefree`
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 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 0 238 825 451 2 1 98
0 0 0 415200 123416 0 0 0 0 0 0 0 0 0 0 0 0 0 219 788 427 1 1 98
Memory Summary

nv70b> kstat unix:0:system_pages:pp_kernel
module: unix instance: 0
name: system_pages class: pages
pp_kernel 53610

nv70b> pagesize
4096
nv70b> bc -l
4096*53610
219586560
nv70b> su
Password:
# mdb -k
Loading modules: [ unix genunix specfs dtrace uppc pcplusmp scsi_vhci ufs mpt ip ]
> ::memstat

<table>
<thead>
<tr>
<th>Page Summary</th>
<th>Pages</th>
<th>MB</th>
<th>%Tot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>50377</td>
<td>196</td>
<td>19%</td>
</tr>
<tr>
<td>Anon</td>
<td>63005</td>
<td>246</td>
<td>24%</td>
</tr>
<tr>
<td>Exec and libs</td>
<td>13689</td>
<td>53</td>
<td>5%</td>
</tr>
<tr>
<td>Page cache</td>
<td>8871</td>
<td>34</td>
<td>3%</td>
</tr>
<tr>
<td>Free (cachelist)</td>
<td>52344</td>
<td>204</td>
<td>20%</td>
</tr>
<tr>
<td>Free (freelist)</td>
<td>71696</td>
<td>280</td>
<td>28%</td>
</tr>
<tr>
<td>Total</td>
<td>259982</td>
<td>1015</td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td>259981</td>
<td>1015</td>
<td></td>
</tr>
</tbody>
</table>

LISA ’09 Baltimore, Md.
```
# vmstat 5 5

<table>
<thead>
<tr>
<th>procs</th>
<th>memory</th>
<th>page</th>
<th>disk</th>
<th>faults</th>
<th>cpu</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- **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
- **in** = interrupts / second
- **sy** = system calls / second
- **cs** = context switches / second
- **us** = user cpu time
- **sy** = kernel cpu time
- **id** = idle + wait cpu time

---

LISA ’09 Baltimore, Md.
### vmstat -p

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

- **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

<table>
<thead>
<tr>
<th>swap</th>
<th>free</th>
<th>re</th>
<th>mf</th>
<th>fr</th>
<th>de</th>
<th>sr</th>
<th>executable</th>
<th>anonymous</th>
<th>filesystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>46715224</td>
<td>891296</td>
<td>24</td>
<td>350</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>46304792</td>
<td>897312</td>
<td>151</td>
<td>761</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>45886168</td>
<td>899808</td>
<td>118</td>
<td>339</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>46723376</td>
<td>899440</td>
<td>29</td>
<td>197</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Executable pages**: kilobytes in - out - freed
- **Anonymous pages**: kilobytes in - out - freed
- **Filesystem**: kilobytes in - out - freed

LISA ’09 Baltimore, Md.
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 \texttt{maxslp}

- **Hard swapping:**
  - Run queue >= 2 (waiting for CPU)
  - Avg. freemem below desfree for 30 sec.
  - Excessive paging, \((\text{pageout} + \text{pagein}) > \text{maxpgio}\)
  - 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"
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 usage and sharing
- MMU/Page Size Performance
  - Use trapstat to observe time spent in TLB misses
Memory Kstats – via kstat(1m)

```
sol8# kstat -n system_pages
module: unix
    name: system_pages
    availrmem
    crtime
    desfree
    desscan
    econtig
    fastscan
    freemem
    kernelbase
    lotsfree
    minfree
    nalloc
    nalloc_calls
    nfree
    nfree_calls
    nscan
    pagesfree
    pageslocked
    pagetotal
    physmem
    pp_kernel
    slowscan
    snaptime

instance: 0
    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
    pagetotal 512136
    physmem 522272
    pp_kernel 64102
    slowscan 100
    snaptime 6573953.83957897
```
Memory Kstats – via kstat Perl API

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

```
so18# wget http://www.solarisinternals.com/si/downloads/prtmem.pl
so18# prtmem.pl 10
prtmem started on 04/01/2005 15:46:13 on devnull, sample interval 5 seconds

<table>
<thead>
<tr>
<th>Time</th>
<th>Total</th>
<th>Kernel</th>
<th>Delta</th>
<th>Free</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:46:18</td>
<td>2040</td>
<td>250</td>
<td>0</td>
<td>972</td>
<td>-12</td>
</tr>
<tr>
<td>15:46:23</td>
<td>2040</td>
<td>250</td>
<td>0</td>
<td>968</td>
<td>-3</td>
</tr>
<tr>
<td>15:46:28</td>
<td>2040</td>
<td>250</td>
<td>0</td>
<td>968</td>
<td>0</td>
</tr>
<tr>
<td>15:46:33</td>
<td>2040</td>
<td>250</td>
<td>0</td>
<td>970</td>
<td>1</td>
</tr>
</tbody>
</table>
```
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 < 1/16\textsuperscript{th} physical)
  - Anonymous page-in/page-outs (vmstat: api, apo > 0)

```bash
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 28739 28804  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:

```sh
sol8$ prstat -mL

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

- The “vminfo” provider has probes at 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 64 bit 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
```
dtrace pagefaults

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

#pragma D option quiet

fbt:unix:pagefault:entry
{
    @st[execname] = count();
    self->pfst[execname] = timestamp
}

fbt:unix:pagefault:return
/ self->pfst[execname] /
{
    @pft[execname] = sum(timestamp - self->pfst[execname]);
    self->pfst[execname] = 0;
}

tick-10s
{
    printf("Pagefault counts by execname ...
    printa(@st);

    printf("\nPagefault times (in nano's) by execname...
    printa(@pft);

    clear(@st);
    clear(@pft);
}
```

tracking pagefault entry and returns for counts and times
dtrace pagefaults

```
# ./pf.d
Pagefault counts by execname ... 

<table>
<thead>
<tr>
<th>Execname</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtrace</td>
<td>93</td>
</tr>
<tr>
<td>java</td>
<td>1257</td>
</tr>
<tr>
<td>kstat</td>
<td>1588</td>
</tr>
</tbody>
</table>

Pagefault times (in nano's) by execname...

<table>
<thead>
<tr>
<th>Execname</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtrace</td>
<td>798535</td>
</tr>
<tr>
<td>kstat</td>
<td>17576367</td>
</tr>
<tr>
<td>java</td>
<td>85760822</td>
</tr>
</tbody>
</table>

Pagefault counts by execname ... 

<table>
<thead>
<tr>
<th>Execname</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtrace</td>
<td>2</td>
</tr>
<tr>
<td>java</td>
<td>1272</td>
</tr>
<tr>
<td>kstat</td>
<td>1588</td>
</tr>
</tbody>
</table>

Pagefault times (in nano's) by execname...

<table>
<thead>
<tr>
<th>Execname</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtrace</td>
<td>80192</td>
</tr>
<tr>
<td>kstat</td>
<td>18227212</td>
</tr>
<tr>
<td>java</td>
<td>75422709</td>
</tr>
</tbody>
</table>

^C
```
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
64-bit Solaris

- LP64 Data Model
- 32-bit or 64-bit kernel, with 32-bit & 64-bit application support
  - 64-bit kernel only on SPARC
    - 32-bit apps no problem
  - Solaris 10 64-bit on AMD64 and Intel
- Comprehensive 32-bit application compatibility
Why 64-bit for large memory?

- Extends the existing programming model to large memory
  - Beyond 4GB limit imposed by 32 bits
- Existing POSIX APIs extend to large data types (e.g. file offsets, 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 Overhead

- 64-bit on AMD64
  - Native 64-bit integer arithmetic
  - 16 general purpose registers (instead of 8)
  - Optimized function call interface – register based arg passing
  - Other instruction set optimizations
Large Memory Configs
Configuring Solaris

- fsflush uses 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(1m)
- Estimate dump as 20% of memory size
- Configure separate dump device
  - Reliable dumps
  - Asynchronous saves during boot (savecore)
- Configure a fast dump device
  - If possible, a HW RAID stripe dump device
Databases

- Exploit 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
    - For every 1000 IOs, 50 are going to disk
  - 99% means 1 out of 100
    - For every 1000 IOs, 10 are going to disk
  - That's a **5X (500%) reduction is physical disk IOs!**
- Use memory for caching
- Write-mostly I/O pattern results
  - Reads satisfied from cache
Multiple Pagesize Support (MPSS) aka Large Pages

- Leverage hardware MMU support for multiple page sizes
- Supported page sizes will vary across different processors
  - pagesize(1)
- Functionality has been an ongoing effort, evolving over time
- Intended to improve performance through more efficient use of hardware TLB
- Be aware of cache effects of large pages (page coloring)
- For DR-capable systems, an interesting dynamic between kernel cage and large pages
  - cage-on: good for LP, may be not good for performance
  - cage-off: more memory fragmentation, not good for LP, but sometimes helps performance
Why Large Pages?

512 8k pages for a 4MB segment, versus one 4MB page
Large Pages – A Brief History

- Solaris 2.6 – Solaris 8
  - SPARC: 4MB pages for ISM
  - SPARC: 4MB pages for initial kernel text and data segments
- Solaris 9
  - SPARC: 8k, 64k, 512k, 4M for user process anon, heap and stack via ppgsz(1), memcntl(2), mpss.so
  - SPARC: 4M for ISM / DISM
- Solaris 10 1/05
  - SPARC: Same as above
  - AMD64: 4k, 2M pages - same constraints as Solaris 9 SPARC
- Solaris 10 1/06 (Update 1)
  - SPARC: Added MPSS for regular file mappings (VMPSS) – enabled by default, 8k & 4M for sun4u, 8k, 64, 4M for sun4v
  - SPARC: Added Large Pages Out-Of-The-Box (LPOOB) for user process anon, stack and heap
  - SPARC: KPR integrated
  - AMD64: 2M for text can be enabled via /etc/system
Large Pages – A Brief History (continued)

- Solaris 10 6/06 (Update 2)
  - SPARC: Large page support for kernel heap
  - SPARC: sun4v 8k, 64k, 512k, 4M, 32M, 256M
- Solaris 10 11/06 (Update 3)
  - SPARCV9 (OPL): 8k, 64k, 512k, 4M, 32M, 256M
- Solaris 10 8/07 (Update 4)
  - SPARC: MPSS Extended to MAP_SHARED anon mappings and non-ISM/DISM SysV Shared Segments. 8k and 4M defaults on all sun4u
  - SPARC: LPOOB changed to use smaller page sizes on N1
  - SPARC: Now have large page support for all types of user segments
  - AMD64: 4k, 2M
  - X64:
Do I need Large Pages?

- Is the application memory intensive?
- How large is the address space?
- 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
### Trapstat Introduction

```
sol9# trapstat -t 1 111

<table>
<thead>
<tr>
<th></th>
<th>cpu m</th>
<th>itlb-miss %tim</th>
<th>itsb-miss %tim</th>
<th>dtlb-miss %tim</th>
<th>dtsb-miss %tim</th>
<th>%tim</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 u</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>2171237</td>
<td>45.7</td>
<td>0.0</td>
</tr>
<tr>
<td>0 k</td>
<td>2</td>
<td>0.0</td>
<td>0.0</td>
<td>3751</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>ttl</td>
<td>3</td>
<td>0.0</td>
<td>0.0</td>
<td>2192238</td>
<td>46.2</td>
<td>0.0</td>
</tr>
</tbody>
</table>
```

- This application *might* run almost 2x faster!
Observing MMU traps

```
[traps]
```

```
sol9# trapstat -T 1 111

<table>
<thead>
<tr>
<th>cpu m size</th>
<th>itlb-miss %tim</th>
<th>itsb-miss %tim</th>
<th>dtlb-miss %tim</th>
<th>dtsb-miss %tim</th>
<th>%tim</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 u 8k</td>
<td>30 0.0</td>
<td>0 0.0</td>
<td>2170236 46.1</td>
<td>0 0.0</td>
<td>46.1</td>
</tr>
<tr>
<td>0 u 64k</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0 u 512k</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0 u 4m</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>------</td>
</tr>
<tr>
<td>0 k 8k</td>
<td>1 0.0</td>
<td>0 0.0</td>
<td>4174 0.1</td>
<td>10 0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>0 k 64k</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0 k 512k</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0 k 4m</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0 0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>------</td>
</tr>
<tr>
<td>ttl</td>
<td>31 0.0</td>
<td>0 0.0</td>
<td>2174410 46.2</td>
<td>10 0.0</td>
<td>46.2</td>
</tr>
</tbody>
</table>

sol9# trapstat -t 1 111

<table>
<thead>
<tr>
<th>cpu m</th>
<th>itlb-miss %tim</th>
<th>itsb-miss %tim</th>
<th>dtlb-miss %tim</th>
<th>dtsb-miss %tim</th>
<th>%tim</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 u</td>
<td>1 0.0</td>
<td>0 0.0</td>
<td>2171237 45.7</td>
<td>0 0.0</td>
<td>45.7</td>
</tr>
<tr>
<td>0 k</td>
<td>2 0.0</td>
<td>0 0.0</td>
<td>3751 0.1</td>
<td>7 0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>ttl</td>
<td>3 0.0</td>
<td>0 0.0</td>
<td>2192238 46.2</td>
<td>7 0.0</td>
<td>46.2</td>
</tr>
</tbody>
</table>
```
Setting Page Sizes

- Solution: `ppgsz(1)`, or `mpss.so.1`
  - Sets page size preference
  - Doesn't persist across `exec()`
  - Beginning with Solaris 10 1/06, Large Pages Out Of the Box (LPOOB) is enabled, so you don't need to do this...
    - You really want to be at Solaris 10 Update 4...

```bash
sol9# ppgsz -o heap=4M ./testprog
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 specified 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
```
## Checking Allocated Page Sizes

```bash
Sol9# pmap -sx `pgrep testprog`
```
2953: ./testprog

<table>
<thead>
<tr>
<th>Address</th>
<th>Kbytes</th>
<th>RSS</th>
<th>Anon</th>
<th>Locked</th>
<th>Pgsz</th>
<th>Mode</th>
<th>Mapped File</th>
</tr>
</thead>
<tbody>
<tr>
<td>00010000</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>dev:277,83 ino:114875</td>
</tr>
<tr>
<td>00020000</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>8K</td>
<td>rwx--</td>
<td>dev:277,83 ino:114875</td>
</tr>
<tr>
<td>00022000</td>
<td>3960</td>
<td>3960</td>
<td>3960</td>
<td>-</td>
<td>8K</td>
<td>rwx--</td>
<td>[ heap ]</td>
</tr>
<tr>
<td>00400000</td>
<td>131072</td>
<td>131072</td>
<td>131072</td>
<td>-</td>
<td>4M</td>
<td>rwx--</td>
<td>[ heap ]</td>
</tr>
<tr>
<td>FF280000</td>
<td>120</td>
<td>120</td>
<td>-</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF340000</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>8K</td>
<td>rwx--</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF390000</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>libc_psr.so.1</td>
</tr>
<tr>
<td>FF3A0000</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>libdl.so.1</td>
</tr>
<tr>
<td>FF3B0000</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>8K</td>
<td>rwx--</td>
<td>[ anon ]</td>
</tr>
<tr>
<td>FF3C0000</td>
<td>152</td>
<td>152</td>
<td>-</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>ld.so.1</td>
</tr>
<tr>
<td>FF3F6000</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>8K</td>
<td>rwx--</td>
<td>ld.so.1</td>
</tr>
<tr>
<td>FFBFA000</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>8K</td>
<td>rwx--</td>
<td>[ stack ]</td>
</tr>
</tbody>
</table>
```

----- ------ ------ ------- ------- -------

| total Kb | 135968 | 135944 | 135112 | -      |
### TLB traps eliminated

```
$ trapstat -T 1 111
```

<table>
<thead>
<tr>
<th>cpu</th>
<th>m</th>
<th>size</th>
<th>itlb-miss</th>
<th>itsb-miss</th>
<th>%tim</th>
<th>dtlb-miss</th>
<th>dtsb-miss</th>
<th>%tim</th>
<th>%tim</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 u</td>
<td>8k</td>
<td></td>
<td>30 0.0</td>
<td>0 0.0</td>
<td></td>
<td>36 0.1</td>
<td>0 0.0</td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>0 u</td>
<td>64k</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>0 u</td>
<td>512k</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>0 u</td>
<td>4m</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>0 k</td>
<td>8k</td>
<td></td>
<td>1 0.0</td>
<td>0 0.0</td>
<td></td>
<td>4174 0.1</td>
<td>10 0.0</td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>0 k</td>
<td>64k</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>0 k</td>
<td>512k</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>0 k</td>
<td>4m</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0 0.0</td>
<td>0 0.0</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>ttl</td>
<td></td>
<td></td>
<td>31 0.0</td>
<td>0 0.0</td>
<td></td>
<td>4200 0.2</td>
<td>10 0.0</td>
<td></td>
<td>0.2</td>
</tr>
</tbody>
</table>
Address Spaces: A Deeper Dive
Example Program

```c
#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

Stack
Libs
Heap
Data
Text

MMU
V
P

LISA ’09 Baltimore, Md.
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
    - ARC for ZFS mapped to kernel heap
So18# /usr/proc/bin/pmap -x $$

18084: csh

<table>
<thead>
<tr>
<th>Address</th>
<th>Kbytes</th>
<th>Resident</th>
<th>Shared</th>
<th>Private</th>
<th>Permissions</th>
<th>Mapped File</th>
</tr>
</thead>
<tbody>
<tr>
<td>000100000</td>
<td>144</td>
<td>144</td>
<td>136</td>
<td>8</td>
<td>read/exec</td>
<td>csh</td>
</tr>
<tr>
<td>000440000</td>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
<td>16 read/write/exec</td>
<td>csh</td>
</tr>
<tr>
<td>000480000</td>
<td>120</td>
<td>104</td>
<td></td>
<td>104</td>
<td>read/write/exec</td>
<td>[ heap ]</td>
</tr>
<tr>
<td>FF2000000</td>
<td>672</td>
<td>624</td>
<td>600</td>
<td>24</td>
<td>read/exec</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF2B80000</td>
<td>24</td>
<td>24</td>
<td></td>
<td></td>
<td>24 read/write/exec</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF3000000</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>read/exec</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF3200000</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>8 read/exec</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF3320000</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>8 read/write/exec</td>
<td>libmapmalloc.so.1</td>
</tr>
<tr>
<td>FF3400000</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>8 read/write/exec</td>
<td>[ anon ]</td>
</tr>
<tr>
<td>FF3500000</td>
<td>168</td>
<td>112</td>
<td>88</td>
<td>24</td>
<td>read/exec</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF38A0000</td>
<td>32</td>
<td>32</td>
<td></td>
<td>32</td>
<td>read/write/exec</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF3920000</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>8 read/write/exec</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF3A00000</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>8 read/write/exec</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF3B00000</td>
<td>136</td>
<td>136</td>
<td>128</td>
<td>8</td>
<td>read/exec</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FF3E20000</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>8 read/write/exec</td>
<td>libc.so.1</td>
</tr>
<tr>
<td>FFBE60000</td>
<td>40</td>
<td>40</td>
<td></td>
<td></td>
<td>40 read/write/exec</td>
<td>libmapmalloc.so.1</td>
</tr>
</tbody>
</table>

----------  ------  ------  ------  -----
total Kb  1424  1304  960  344
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];
```

Address Space

Heap

Data

Text

Vnode Segment Driver

seg_fault()

segvn_fault()

vop_getpage()

swapfs

swap space
### 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

<table>
<thead>
<tr>
<th>Memory</th>
<th>Page</th>
<th>Executable</th>
<th>Anonymous</th>
<th>Filesystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>swap</td>
<td>free</td>
<td>re mf fr de sr</td>
<td>epi epo epf</td>
</tr>
<tr>
<td>46715224</td>
<td>24</td>
<td>350</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>46304792</td>
<td>151</td>
<td>761</td>
<td>25 0 0</td>
<td>17 0 0</td>
</tr>
<tr>
<td>45886168</td>
<td>118</td>
<td>339</td>
<td>1 0 0</td>
<td>3 0 0</td>
</tr>
<tr>
<td>46723376</td>
<td>29</td>
<td>197</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>
Examining paging with dtrace VM Provider

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

```
$ kstat -n vm
module: cpu
  name:   vm
table:   module: cpu
description: name
class:    misc
  anonfree 0
  anonpgin 0
  anonpgout 0
  as_fault 3180528
  cow_fault 37280
  ctime 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 â€œpginâ€ 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 responsible 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:

```bash
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:

```d
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

```bash
# dtrace -s ./soffice.d
 dtrace: script ./soffice.d 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

<table>
<thead>
<tr>
<th>Zfod value</th>
<th>Distribution</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>@@@@@@@@@@@</td>
<td>525</td>
</tr>
<tr>
<td>1</td>
<td>@@@@@@@@@@@</td>
<td>605</td>
</tr>
<tr>
<td>2</td>
<td>@@</td>
<td>208</td>
</tr>
<tr>
<td>3</td>
<td>@@@</td>
<td>280</td>
</tr>
<tr>
<td>4</td>
<td>@@</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>@@@</td>
<td>161</td>
</tr>
<tr>
<td>6</td>
<td>@@@</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>@@@</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>@@@</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>@@@</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>@@@</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>@@@</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>@@@</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>@@@</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>@@@</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>@@@</td>
<td>29</td>
</tr>
<tr>
<td>16</td>
<td>@@@@@@@@@@@@@@</td>
<td>1048</td>
</tr>
<tr>
<td>17</td>
<td>@@@@@@@@@@@@@</td>
<td>24</td>
</tr>
<tr>
<td>18</td>
<td>@@@@@@@@@@@@@</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>@@@@@@@@@@@@@</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>@@@@@@@@@@@@@</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>@@@@@@@@@@@@@</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>@@@@@@@@@@@@@</td>
<td>3</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Shared Mapped File

Mapped File

Mapped File

Heap

Data

Text

Heap

Data

Text
Copy-on-write

Libraries
Copy on write remaps pagesize address to anonymous memory (swap space)
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()`
Session 3
Processes, Threads, Scheduling Classes & The Dispatcher
## Process/Threads Glossary

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

- Processes originate 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 (SPARC, x86)
  - 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

```bash
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

```bash
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
```
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

```sh
bornrorun> 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/platform/SUNW,Ultra-60/lib/libmp.so.2
    /usr/platform/SUNW,Ultra-60/lib/libc_psr.so.1

bornrorun> 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
bornrorun>
```
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)
.....
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: calling .init (from sorted order): /lib/libc.so.1
13686: 1: calling .init (done): /lib/libc.so.1
13686: 1: transferring control: /opt/filebench/bin/filebench
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: hardware capabilities - 0x2b [ VIS V8PLUS DIV32 MUL32 ]
15151: configuration file=/var/ld/ld.config: unable to process file
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'
Runtime Linker – Debug

• Explore the options in The Linker and Libraries Guide
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

- proc_t
- vnode_t
- a.out vnode
- as_t
- address space
- cred_t
- credentials
- sess_t
- session
- kthread_t
- kernel thread
- scheduling class data
- environment
- data
- environment
- args
- signals
- rlimits
- file list
- open
- memory pages
- page_t
- seg
- vnode
- /proc support
- signal management
- LWP & kernel thread stuff
- lineage pointers
- kernel process table
- user area
Process Structure

```
# mdb -k
Loading modules: [ unix krt1d genunix specfs dtrace ufs ip scctp usba fctl nca lofs nfs random sppp crypto ptm logindmux cpc ]
```

```
> ::ps

<table>
<thead>
<tr>
<th>S</th>
<th>PID</th>
<th>PPID</th>
<th>PGID</th>
<th>SID</th>
<th>UID</th>
<th>FLAGS</th>
<th>ADDR</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0x00000001</td>
<td>fffffffffbc1ce80 sched</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0x00020001</td>
<td>fffffffff880838f8 fsflush</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0x00020001</td>
<td>fffffffff88084520 pageout</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0x42004000</td>
<td>fffffffff88085148 init</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>21344</td>
<td>1</td>
<td>21343</td>
<td>21280</td>
<td>2234</td>
<td>0x42004000</td>
<td>fffffffff95549938 tcpPerfServer</td>
<td></td>
</tr>
</tbody>
</table>

> fffffffff95549938::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 = 0xfffffffffffffffff
            rlim_max = 0xfffffffffffffffff
        }
    ]
    ...
    fi_nfiles = 0x3f
    fi_list = 0xffffffff8dc44000
    fi_rlist = 0
}
```
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

```bash
borntorun> kstat -n var
module: unix
   instance: 0
ame:   var
class:   misc
crtime:  61.041156087
snaptime:  113918.89449089
v_autoup:  30
v_buf:  100
v_bufhwm:  20312
[vsnip]
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 ... ptm ipc ]
> max_nprocs/D
max_nprocs: 15882
>
```
## System-wide Process View - ps(1)

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

`-daemon`
System-wide Process View - prstat(1)

<table>
<thead>
<tr>
<th>PID</th>
<th>USERNAME</th>
<th>SIZE</th>
<th>RSS</th>
<th>STATE</th>
<th>PRI</th>
<th>NICE</th>
<th>TIME</th>
<th>CPU</th>
<th>PROCESS/NLWP</th>
</tr>
</thead>
<tbody>
<tr>
<td>26292</td>
<td>root</td>
<td>5368K</td>
<td>3080K</td>
<td>run</td>
<td>24</td>
<td>0</td>
<td>0:00:00 1.5%</td>
<td>pkginstall/1</td>
<td></td>
</tr>
<tr>
<td>26188</td>
<td>rmc</td>
<td>4880K</td>
<td>4512K</td>
<td>cpu0</td>
<td>49</td>
<td>0</td>
<td>0:00:00 0.6%</td>
<td>prstat/1</td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>root</td>
<td>3304K</td>
<td>1800K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:07 0.3%</td>
<td>nscd/24</td>
<td></td>
</tr>
<tr>
<td>23078</td>
<td>root</td>
<td>20M</td>
<td>14M</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:56 0.2%</td>
<td>lupi_zones/1</td>
<td></td>
</tr>
<tr>
<td>23860</td>
<td>root</td>
<td>5104K</td>
<td>2328K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:01 0.1%</td>
<td>sshd/1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>365</td>
<td>root</td>
<td>4760K</td>
<td>128K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>zoneadmd/4</td>
<td></td>
</tr>
<tr>
<td>364</td>
<td>root</td>
<td>4776K</td>
<td>128K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>zoneadmd/4</td>
<td></td>
</tr>
<tr>
<td>374</td>
<td>root</td>
<td>0K</td>
<td>0K</td>
<td>sleep</td>
<td>60</td>
<td>-</td>
<td>0:00:00 0.0%</td>
<td>zsched/1</td>
<td></td>
</tr>
<tr>
<td>361</td>
<td>root</td>
<td>2016K</td>
<td>8K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>ttymon/1</td>
<td></td>
</tr>
<tr>
<td>349</td>
<td>root</td>
<td>8600K</td>
<td>616K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:20 0.0%</td>
<td>snmpd/1</td>
<td></td>
</tr>
<tr>
<td>386</td>
<td>root</td>
<td>2096K</td>
<td>360K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>init/1</td>
<td></td>
</tr>
<tr>
<td>345</td>
<td>root</td>
<td>3160K</td>
<td>496K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>sshd/1</td>
<td></td>
</tr>
<tr>
<td>591</td>
<td>root</td>
<td>3824K</td>
<td>184K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>automountd/2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>242</td>
<td>root</td>
<td>1896K</td>
<td>8K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>smcboot/1</td>
<td></td>
</tr>
<tr>
<td>248</td>
<td>smmsp</td>
<td>4736K</td>
<td>696K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:08 0.0%</td>
<td>sendmail/1</td>
<td></td>
</tr>
<tr>
<td>245</td>
<td>root</td>
<td>1888K</td>
<td>0K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>smcboot/1</td>
<td></td>
</tr>
<tr>
<td>824</td>
<td>root</td>
<td>2016K</td>
<td>8K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>ttymon/1</td>
<td></td>
</tr>
<tr>
<td>204</td>
<td>root</td>
<td>2752K</td>
<td>536K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>inetd/1</td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>root</td>
<td>1568K</td>
<td>8K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>powerd/3</td>
<td></td>
</tr>
<tr>
<td>313</td>
<td>root</td>
<td>2336K</td>
<td>216K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00 0.0%</td>
<td>snmpdx/1</td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>root</td>
<td>4312K</td>
<td>872K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:01 0.0%</td>
<td>syslogd/13</td>
<td></td>
</tr>
<tr>
<td>162</td>
<td>daemon</td>
<td>2240K</td>
<td>16K</td>
<td>sleep</td>
<td>60</td>
<td>-20</td>
<td>0:00:00 0.0%</td>
<td>lockd/2</td>
<td></td>
</tr>
</tbody>
</table>

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 parent's address space, until exec()

```c
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
Process create example

C code calling fork()

```c
#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...
");
    } else {
        printf("Child PID: %d\n", ret);
    }
    exit(0);
}
```

D script to generate kernel trace

```d
#!/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);
}
```

C code calling fork()

D script to generate kernel trace
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
            ...
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  -> continuellwps
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  <= forkl
Watching Forks

D script for watching fork(2)

```d
#!/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

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

fork1

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

D script for watching fork(2)
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...

```d
#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

<table>
<thead>
<tr>
<th>WHO</th>
<th>WHAT</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>make.bin</td>
<td>yacc</td>
<td>1</td>
</tr>
<tr>
<td>tcsh</td>
<td>make</td>
<td>1</td>
</tr>
<tr>
<td>make.bin</td>
<td>spec2map</td>
<td>1</td>
</tr>
<tr>
<td>sh</td>
<td>grep</td>
<td>1</td>
</tr>
<tr>
<td>lint</td>
<td>lint2</td>
<td>1</td>
</tr>
<tr>
<td>sh</td>
<td>lint</td>
<td>1</td>
</tr>
<tr>
<td>sh</td>
<td>ln</td>
<td>1</td>
</tr>
<tr>
<td>cc</td>
<td>ld</td>
<td>1</td>
</tr>
<tr>
<td>make.bin</td>
<td>cc</td>
<td>1</td>
</tr>
<tr>
<td>lint</td>
<td>lint1</td>
<td>1</td>
</tr>
</tbody>
</table>
```
In this example the command "man gzip" was executed. The output lets us see what the man command is actually doing,

```
# ./execsnoop

UID   PID  PPID ARGS
100  3064  2656 man gzip
100  3065  3064 sh -c cd /usr/share/man; tbl /usr/share/man/man1/gzip.1 |nroff -u0 -Tlp -man -
100  3067  3066 tbl /usr/share/man/man1/gzip.1
100  3068  3066 nroff -u0 -Tlp -man -
100  3069  3065 col -x
100  3069  3064 sh -c trap " 1 15; /usr/bin/mv -f /tmp/mpoMaa_f /usr/share/man/cat1/gzip.1 2>
100  3070  3069 /usr/bin/mv -f /tmp/mpoMaa_f /usr/share/man/cat1/gzip.1
100  3071  3064 sh -c more -s /tmp/mpoMaa_f
100  3072  3071 more -s /tmp/mpoMaa_f
^C
```
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

<table>
<thead>
<tr>
<th>Process State</th>
<th>Kernel Thread State</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIDL</td>
<td></td>
</tr>
<tr>
<td>SRUN</td>
<td>TS_RUN</td>
</tr>
<tr>
<td>SONPROC</td>
<td>TS_ONPROC</td>
</tr>
<tr>
<td>SSLEEP</td>
<td>TS_SLEEP</td>
</tr>
<tr>
<td>SSTOP</td>
<td>TS_STOPPED</td>
</tr>
<tr>
<td>SZOMB</td>
<td>TS_ZOMB</td>
</tr>
<tr>
<td></td>
<td>TS_FREE</td>
</tr>
</tbody>
</table>
State Transitions

fork()

IDL

RUN

ONPROC

SLEEP

PINNED

intr

swtch()

syscall

preempt

wakeup

prun(1)

STOPPED

pstop(1)

reap

exit()

pthread_exit()

FREE

ZOMBIE
## Watching Process States

<table>
<thead>
<tr>
<th>PID</th>
<th>USERNAME</th>
<th>SIZE</th>
<th>RSS</th>
<th>STATE</th>
<th>PRI</th>
<th>NICE</th>
<th>TIME</th>
<th>CPU</th>
<th>PROCESS/NLWP</th>
</tr>
</thead>
<tbody>
<tr>
<td>27946</td>
<td>root</td>
<td>4880K</td>
<td>4520K</td>
<td>cpu0</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.7%</td>
<td>prstat/1</td>
</tr>
<tr>
<td>28010</td>
<td>root</td>
<td>4928K</td>
<td>2584K</td>
<td>run</td>
<td>29</td>
<td>0</td>
<td>0:00:00</td>
<td>0.7%</td>
<td>pkginstall/1</td>
</tr>
<tr>
<td>23078</td>
<td>root</td>
<td>20M</td>
<td>14M</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:57</td>
<td>0.3%</td>
<td>lupi_zones/1</td>
</tr>
<tr>
<td>25947</td>
<td>root</td>
<td>5160K</td>
<td>2976K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:04</td>
<td>0.3%</td>
<td>sshd/1</td>
</tr>
<tr>
<td>24866</td>
<td>root</td>
<td>5136K</td>
<td>2136K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:01</td>
<td>0.2%</td>
<td>sshd/1</td>
</tr>
<tr>
<td>202</td>
<td>root</td>
<td>3304K</td>
<td>1800K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:09</td>
<td>0.2%</td>
<td>nscd/24</td>
</tr>
<tr>
<td>23001</td>
<td>root</td>
<td>5136K</td>
<td>2176K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:04</td>
<td>0.1%</td>
<td>sshd/1</td>
</tr>
<tr>
<td>23860</td>
<td>root</td>
<td>5248K</td>
<td>2392K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:05</td>
<td>0.1%</td>
<td>sshd/1</td>
</tr>
<tr>
<td>25946</td>
<td>rmc</td>
<td>3008K</td>
<td>2184K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:02</td>
<td>0.1%</td>
<td>ssh/1</td>
</tr>
<tr>
<td>25690</td>
<td>root</td>
<td>1240K</td>
<td>928K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.1%</td>
<td>sh/1</td>
</tr>
<tr>
<td>312</td>
<td>root</td>
<td>4912K</td>
<td>24K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.0%</td>
<td>dtlogin/1</td>
</tr>
<tr>
<td>250</td>
<td>root</td>
<td>4760K</td>
<td>696K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:16</td>
<td>0.0%</td>
<td>sendmail/1</td>
</tr>
<tr>
<td>246</td>
<td>root</td>
<td>1888K</td>
<td>0K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.0%</td>
<td>smcboot/1</td>
</tr>
<tr>
<td>823</td>
<td>root</td>
<td>1936K</td>
<td>224K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.0%</td>
<td>sac/1</td>
</tr>
<tr>
<td>242</td>
<td>root</td>
<td>1896K</td>
<td>8K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.0%</td>
<td>smcboot/1</td>
</tr>
<tr>
<td>248</td>
<td>smmsp</td>
<td>4736K</td>
<td>680K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:08</td>
<td>0.0%</td>
<td>sendmail/1</td>
</tr>
<tr>
<td>245</td>
<td>root</td>
<td>1888K</td>
<td>0K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.0%</td>
<td>smcboot/1</td>
</tr>
<tr>
<td>824</td>
<td>root</td>
<td>2016K</td>
<td>8K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.0%</td>
<td>ttymon/1</td>
</tr>
<tr>
<td>204</td>
<td>root</td>
<td>2752K</td>
<td>520K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.0%</td>
<td>inetd/1</td>
</tr>
<tr>
<td>220</td>
<td>root</td>
<td>1568K</td>
<td>8K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.0%</td>
<td>powerd/3</td>
</tr>
<tr>
<td>313</td>
<td>root</td>
<td>2336K</td>
<td>216K</td>
<td>sleep</td>
<td>59</td>
<td>0</td>
<td>0:00:00</td>
<td>0.0%</td>
<td>snmpdx/1</td>
</tr>
</tbody>
</table>

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

```
# prstat -m

<table>
<thead>
<tr>
<th>PID</th>
<th>USERNAME</th>
<th>USR</th>
<th>SYS</th>
<th>TRP</th>
<th>TFL</th>
<th>DFL</th>
<th>LCK</th>
<th>SLP</th>
<th>LAT</th>
<th>VCX</th>
<th>ICX</th>
<th>SCL</th>
<th>SIG</th>
<th>PROCESS/ NLWP</th>
</tr>
</thead>
<tbody>
<tr>
<td>739</td>
<td>root</td>
<td>0.3</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>99</td>
<td>0.0</td>
<td>126</td>
<td>3</td>
<td>345</td>
<td>5</td>
<td>Xsun/1</td>
</tr>
<tr>
<td>1561</td>
<td>root</td>
<td>0.1</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>23</td>
<td>0</td>
<td>381</td>
<td>0</td>
<td>gnome-panel/1</td>
</tr>
<tr>
<td>1125</td>
<td>tlc</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>29</td>
<td>0</td>
<td>116</td>
<td>0</td>
<td>prstat/1</td>
</tr>
<tr>
<td>15553</td>
<td>rmc</td>
<td>0.1</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>24</td>
<td>0</td>
<td>381</td>
<td>0</td>
<td>prstat/1</td>
</tr>
<tr>
<td>5591</td>
<td>tlc</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>33</td>
<td>66</td>
<td>206</td>
<td>0</td>
<td>1K</td>
<td>0</td>
<td>mozilla-bin/6</td>
</tr>
<tr>
<td>1121</td>
<td>tlc</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.1</td>
<td>50</td>
<td>0</td>
<td>230</td>
<td>0</td>
<td>metacity/1</td>
</tr>
<tr>
<td>2107</td>
<td>rmc</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>25</td>
<td>0</td>
<td>36</td>
<td>0</td>
<td>gnome-termin/1</td>
</tr>
<tr>
<td>478</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>17</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>squid/1</td>
</tr>
<tr>
<td>798</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>11</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>Xsun/1</td>
</tr>
<tr>
<td>1145</td>
<td>tlc</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>25</td>
<td>1</td>
<td>34</td>
<td>0</td>
<td>mixer_applet/1</td>
</tr>
<tr>
<td>1141</td>
<td>rmc</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>25</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>mixer_applet/1</td>
</tr>
<tr>
<td>1119</td>
<td>tlc</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>5</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>gnome-smprox/1</td>
</tr>
<tr>
<td>1127</td>
<td>tlc</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>7</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>nautilus/3</td>
</tr>
<tr>
<td>1105</td>
<td>rmc</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>7</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>nautilus/3</td>
</tr>
<tr>
<td>713</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>85</td>
<td>15</td>
<td>2</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>mibiisa/7</td>
</tr>
<tr>
<td>174</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>5</td>
<td>0</td>
<td>50</td>
<td>5</td>
<td>ipmon/1</td>
</tr>
<tr>
<td>1055</td>
<td>tlc</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>5</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>dsdm/1</td>
</tr>
</tbody>
</table>
```

Total: 163 processes, 275 lwps, load averages: 0.07, 0.07, 0.07
prstat – user summary

```
# 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  kfheisen     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 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/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] ...
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 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", 0xFFBE68C, 0xFFBE6A4) 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(0x0FF332000, 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(0x0FF390000, 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 sysconf(_CONFIG_PAGESIZE) = 8192
0.0121 brk(0x000023F40) = 0
0.0123 brk(0x00002BF40) = 0
0.0127 sigaction(SIGINT, 0xFFBEF470, 0xFFBEF4F0) = 0
0.0129 sigaction(SIGINT, 0xFFBEF470, 0xFFBEF4F0) = 0
0.0134 read(3, "01010101010101010101010101010101", 16384) = 16384
0.0137 write(4, "01010101010101010101010101010101", 16384) = 16384
0.0140 read(3, "01010101010101010101010101010101", 16384) = 16384
0.0143 write(4, "01010101010101010101010101010101", 16384) = 16384
0.0146 read(3, "01010101010101010101010101010101", 16384) = 16384
0.0149 write(4, "01010101010101010101010101010101", 16384) = 16384
0.0152 read(3, "01010101010101010101010101010101", 16384) = 16384
0.0154 write(4, "01010101010101010101010101010101", 16384) = 16384
### Process Tracing – System Call Summary

- Counts total cpu seconds per system call and calls

```bash
# truss -c dd if=500m of=/dev/null bs=16k count=2k
	sySCALL          SECONDS  CALLS
_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=/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", 0xFFFBEF68C, 0xFFFBEF6A4) 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) = 0xFFF3A0000
0.0094 open("/usr/lib/libc.so.1", O_RDONLY) = 4
0.0097 fstat(4, 0xFFFBEF224) = 0
0.0100 mmap(0x00000000, 8192, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFFF390000
0.0102 mmap(0x00000000, 761856, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFFF280000
0.0105 munmap(0xFFF324000, 57344) = 0
0.0107 mmap(0xFFF332000, 25284, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 663552) = 0xFFF332000
0.0113 close(4) = 0
0.0116 open("/usr/lib/libdl.so.1", O_RDONLY) = 4
0.0119 fstat(4, 0xFFFBEF224) = 0
0.0121 mmap(0xFFF390000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0) = 0xFFF390000
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, 0xFFFBEF004) = 0
0.0133 mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFFF380000
0.0135 mmap(0x00000000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFFF370000
0.0138 close(4) = 0
0.2369 close(3) = 0
0.2372 munmap(0xFFF380000, 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, 0xfff6f8c, 0xfff6f6a4, 0x23400)
0.2443 -> libc:setlocale(0x6, 0x12f14, 0x0, 0x0)
0.2585 <- libc:setlocale() = 0xff31f316
```
Library Tracing – `apptrace(1)`

```sh
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 =
                 "RaAdClxmnlogrtucpFbq") = 0xffffffff errno = 0 (Error 0)
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:

```c
#include <pthread.h>
#include <stdio.h>
mutex_t mem_lock;

void childthread(void *argument)
{
    int i;
    for(i = 1; i <= 100; ++i) {
        printf("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);
    printf("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)

- Bound thread
- Unbound user threads
- Libthread run queues & scheduler
- Kernel per-cpu run queues, kernel dispatcher
- Processors

**Diagram Description:**
- The diagram illustrates the multilevel model of the Solaris 8 operating system.
- It shows the relationship between threads and processors, with kernel threads and user threads interacting through run queues.
- The diagram highlights the structure and flow of processes and threads within the operating system.
T1 – Multilevel Model

• Pros:
  • Fast user thread create and destroy
  • Allows many-to-few thread model, to minimize 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

user threads

user

kernel

process

LWP kernel thread

LWP kernel thread

LWP kernel thread

LWP kernel thread

kernel per-cpu run queues, kernel dispatcher

Processors

LISA '09 Baltimore, Md.
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 Behavior
  - 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.
Thread Semantics Added to pstack, truss

```bash
# 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
cceac686 _cond_reltimedwait (ce9f8378, ce9f83a0, afffdea0) + 36
cceac6b4 cond_reltimedwait (ce9f8378, ce9f83a0, afffdea0) + 24
cce9e5902 __aio_waitn (82d1f08, 1000, afffdef2c, afffdef18, 1) + 529
cceaf2a84 aio_waitn64 (82d1f08, 1000, afffdef2c, afffdef18) + 24
08063065 flowoplib_aiowait (b4eb475c, c40f4d54) + 97
08061de1 flowop_start (b4eb475c) + 257
cceab15c0 _thr_setup (ce9a8400) + 50
cceab1780 _lwp_start (ce9a8400, 0, 0, afffdef8, cceab1780, ce9a8400)
```

```bash
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 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 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 U", 1056768)  = 1056768
```
### Thread Microstates

<table>
<thead>
<tr>
<th>PID</th>
<th>USERNAME</th>
<th>USR</th>
<th>SYS</th>
<th>TRP</th>
<th>TFL</th>
<th>DFL</th>
<th>LCK</th>
<th>SLP</th>
<th>LAT</th>
<th>VCX</th>
<th>ICX</th>
<th>SCL</th>
<th>SIG</th>
<th>PROCESS/LWPID</th>
</tr>
</thead>
<tbody>
<tr>
<td>918</td>
<td>rmc</td>
<td>0.2</td>
<td>0.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>99</td>
<td>0.0</td>
<td>27</td>
<td>2</td>
<td>1K</td>
<td>0</td>
<td>prstat/1</td>
</tr>
<tr>
<td>919</td>
<td>mauroj</td>
<td>0.1</td>
<td>0.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>99</td>
<td>0.1</td>
<td>44</td>
<td>12</td>
<td>1K</td>
<td>0</td>
<td>prstat/1</td>
</tr>
<tr>
<td>907</td>
<td>root</td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>97</td>
<td>3.1</td>
<td>121</td>
<td>2</td>
<td>20</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>913</td>
<td>root</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>100</td>
<td>0.0</td>
<td>0.0</td>
<td>15</td>
<td>2</td>
<td>420</td>
<td>0</td>
</tr>
<tr>
<td>866</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>4.1</td>
<td>44</td>
<td>41</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>820</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>95</td>
<td>5.0</td>
<td>43</td>
<td>42</td>
<td>424</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>814</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>95</td>
<td>5.0</td>
<td>43</td>
<td>41</td>
<td>424</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>772</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>3.6</td>
<td>46</td>
<td>39</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>749</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>3.7</td>
<td>45</td>
<td>41</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>744</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>95</td>
<td>4.7</td>
<td>47</td>
<td>39</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>859</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>95</td>
<td>4.9</td>
<td>44</td>
<td>41</td>
<td>424</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>837</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>4.0</td>
<td>43</td>
<td>43</td>
<td>405</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>787</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>95</td>
<td>4.5</td>
<td>43</td>
<td>41</td>
<td>424</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>776</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>95</td>
<td>4.8</td>
<td>43</td>
<td>42</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>774</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>4.2</td>
<td>43</td>
<td>40</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>756</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>3.8</td>
<td>44</td>
<td>41</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>738</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>4.4</td>
<td>43</td>
<td>42</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>735</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>3.9</td>
<td>47</td>
<td>39</td>
<td>405</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>734</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>4.3</td>
<td>44</td>
<td>41</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>727</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>4.4</td>
<td>43</td>
<td>43</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
<tr>
<td>725</td>
<td>root</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>96</td>
<td>4.4</td>
<td>43</td>
<td>43</td>
<td>398</td>
<td>0</td>
<td>filebench/2</td>
</tr>
</tbody>
</table>

Total: 257 processes, 3139 lwps, load averages: 7.71, 2.39, 0.97
## Watching Threads

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

Total: 125 processes, 310 lwps, load averages: 0.50, 0.38, 0.40
Who's Creating Threads?

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

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

• **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 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

- cpu_t
  - cpu Disp
  - cpu_runrun
  - cpu_kprunrun
  - cpu_dispthread
  - ...

- disp_t
  - disp_lock
  - disp_npri
  - disp_q
  - disp_qactmap
  - disp_maxrunpri
  - disp_nrunnable

- dispq_t
  - dq_first
  - dq_last
  - dq_runcnt

- kthread_t
  - kernel thread

- A queue for every global priority

LISA '09 Baltimore, Md.
Timeshare Dispatch Table

- TS and IA class share the same dispatch table
  - \textit{RES}. Defines the granularity of \texttt{ts\_quantum}
  - \texttt{ts\_quantum}. CPU time for next ONPROC state
  - \texttt{ts\_tqexp}. New priority if time quantum expires
  - \texttt{ts\_slpret}. New priority when state change from TS\_SLEEP to TS\_RUN
  - \texttt{ts\_maxwait}. “waited too long” ticks
  - \texttt{ts\_lwait}. New priority if “waited too long”

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

<table>
<thead>
<tr>
<th># ts_quantum</th>
<th>ts_tqexp</th>
<th>ts_slpret</th>
<th>ts_maxwait</th>
<th>ts_lwait</th>
<th>PRIORITY</th>
<th>LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td>#</td>
<td>0</td>
</tr>
<tr>
<td>200</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td>#</td>
<td>1</td>
</tr>
<tr>
<td>..............</td>
<td>0</td>
<td>51</td>
<td>0</td>
<td>51</td>
<td>#</td>
<td>10</td>
</tr>
<tr>
<td>160</td>
<td>1</td>
<td>51</td>
<td>0</td>
<td>51</td>
<td>#</td>
<td>11</td>
</tr>
<tr>
<td>..............</td>
<td>10</td>
<td>52</td>
<td>0</td>
<td>52</td>
<td>#</td>
<td>20</td>
</tr>
<tr>
<td>120</td>
<td>11</td>
<td>52</td>
<td>0</td>
<td>52</td>
<td>#</td>
<td>21</td>
</tr>
<tr>
<td>..............</td>
<td>20</td>
<td>53</td>
<td>0</td>
<td>53</td>
<td>#</td>
<td>30</td>
</tr>
<tr>
<td>80</td>
<td>21</td>
<td>53</td>
<td>0</td>
<td>53</td>
<td>#</td>
<td>31</td>
</tr>
<tr>
<td>..............</td>
<td>30</td>
<td>55</td>
<td>0</td>
<td>55</td>
<td>#</td>
<td>40</td>
</tr>
<tr>
<td>40</td>
<td>31</td>
<td>55</td>
<td>0</td>
<td>55</td>
<td>#</td>
<td>41</td>
</tr>
<tr>
<td>..............</td>
<td>49</td>
<td>59</td>
<td>32000</td>
<td>59</td>
<td>#</td>
<td>59</td>
</tr>
</tbody>
</table>
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 on 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)

```c
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 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 their 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
## Context Switching

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>minf</th>
<th>mjf</th>
<th>xcal</th>
<th>intr</th>
<th>ithr</th>
<th>csw</th>
<th>icsw</th>
<th>migr</th>
<th>smtx</th>
<th>srw</th>
<th>syscl</th>
<th>usr</th>
<th>sys</th>
<th>wt</th>
<th>idl</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>74</td>
<td>2</td>
<td>998</td>
<td>417</td>
<td>302</td>
<td>450</td>
<td>18</td>
<td>45</td>
<td>114</td>
<td>0</td>
<td>1501</td>
<td>56</td>
<td>7</td>
<td>0</td>
<td>37</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>3</td>
<td>797</td>
<td>120</td>
<td>102</td>
<td>1107</td>
<td>16</td>
<td>58</td>
<td>494</td>
<td>0</td>
<td>1631</td>
<td>41</td>
<td>16</td>
<td>0</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>209</td>
<td>2</td>
<td>253</td>
<td>114</td>
<td>100</td>
<td>489</td>
<td>12</td>
<td>45</td>
<td>90</td>
<td>0</td>
<td>1877</td>
<td>56</td>
<td>11</td>
<td>0</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>503</td>
<td>7</td>
<td>2448</td>
<td>122</td>
<td>100</td>
<td>913</td>
<td>21</td>
<td>53</td>
<td>225</td>
<td>0</td>
<td>2626</td>
<td>32</td>
<td>21</td>
<td>0</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>287</td>
<td>3</td>
<td>60</td>
<td>120</td>
<td>100</td>
<td>771</td>
<td>20</td>
<td>35</td>
<td>122</td>
<td>0</td>
<td>1569</td>
<td>50</td>
<td>12</td>
<td>0</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>46</td>
<td>1</td>
<td>51</td>
<td>115</td>
<td>99</td>
<td>671</td>
<td>16</td>
<td>20</td>
<td>787</td>
<td>0</td>
<td>846</td>
<td>81</td>
<td>16</td>
<td>0</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>127</td>
<td>2</td>
<td>177</td>
<td>117</td>
<td>101</td>
<td>674</td>
<td>14</td>
<td>27</td>
<td>481</td>
<td>0</td>
<td>881</td>
<td>74</td>
<td>12</td>
<td>0</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>375</td>
<td>7</td>
<td>658</td>
<td>1325</td>
<td>1302</td>
<td>671</td>
<td>23</td>
<td>49</td>
<td>289</td>
<td>0</td>
<td>1869</td>
<td>48</td>
<td>16</td>
<td>0</td>
<td>37</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU</th>
<th>minf</th>
<th>mjf</th>
<th>xcal</th>
<th>intr</th>
<th>ithr</th>
<th>csw</th>
<th>icsw</th>
<th>migr</th>
<th>smtx</th>
<th>srw</th>
<th>syscl</th>
<th>usr</th>
<th>sys</th>
<th>wt</th>
<th>idl</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>733</td>
<td>399</td>
<td>297</td>
<td>548</td>
<td>10</td>
<td>8</td>
<td>653</td>
<td>0</td>
<td>518</td>
<td>80</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>182</td>
<td>4</td>
<td>45</td>
<td>117</td>
<td>100</td>
<td>412</td>
<td>16</td>
<td>34</td>
<td>49</td>
<td>0</td>
<td>904</td>
<td>54</td>
<td>8</td>
<td>0</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td>156</td>
<td>4</td>
<td>179</td>
<td>108</td>
<td>102</td>
<td>1029</td>
<td>6</td>
<td>46</td>
<td>223</td>
<td>0</td>
<td>1860</td>
<td>15</td>
<td>16</td>
<td>0</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>98</td>
<td>1</td>
<td>53</td>
<td>110</td>
<td>100</td>
<td>568</td>
<td>9</td>
<td>19</td>
<td>338</td>
<td>0</td>
<td>741</td>
<td>60</td>
<td>9</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>8</td>
<td>47</td>
<td>1</td>
<td>96</td>
<td>111</td>
<td>101</td>
<td>630</td>
<td>6</td>
<td>22</td>
<td>712</td>
<td>0</td>
<td>615</td>
<td>56</td>
<td>13</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>9</td>
<td>143</td>
<td>4</td>
<td>127</td>
<td>116</td>
<td>102</td>
<td>1144</td>
<td>11</td>
<td>42</td>
<td>439</td>
<td>0</td>
<td>2443</td>
<td>33</td>
<td>15</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>12</td>
<td>318</td>
<td>0</td>
<td>268</td>
<td>111</td>
<td>100</td>
<td>734</td>
<td>9</td>
<td>30</td>
<td>96</td>
<td>0</td>
<td>1455</td>
<td>19</td>
<td>12</td>
<td>0</td>
<td>69</td>
</tr>
<tr>
<td>13</td>
<td>39</td>
<td>2</td>
<td>16</td>
<td>938</td>
<td>929</td>
<td>374</td>
<td>8</td>
<td>9</td>
<td>103</td>
<td>0</td>
<td>756</td>
<td>69</td>
<td>6</td>
<td>0</td>
<td>25</td>
</tr>
</tbody>
</table>
#!/usr/sbin/dtrace -Zqs
long inv_cnt;  /* all involuntary context switches */
long tqe_cnt;  /* time quantum expiration count */
long hpp_cnt;  /* higher-priority preempt count */
long csw_cnt;  /* total number context switches */

dtrace:::BEGIN
{
    inv_cnt = 0; tqe_cnt = 0; hpp_cnt = 0; csw_cnt = 0;

    printf("%-16s %-16s %-16s %-16s\n", "TOTAL CSW", "ALL INV", "TQE_INV", "HPP_INV");

    printf("====================================================================\n");

    sysinfo:unix:preempt:inv_swtch
    {
        inv_cnt += arg0;
    }

    sysinfo:unix::pswitch
    {
        csw_cnt += arg0;
    }

    fbt:TS:ts_preempt:entry
    / ((tsproc_t *)args[0]->t_cldata)->ts_timeleft <= 1 /
    {
        tqe_cnt++;
    }

    fbt:TS:ts_preempt:entry
    / ((tsproc_t *)args[0]->t_cldata)->ts_timeleft > 1 /
    {
        hpp_cnt++;
    }

    fbt:RT:rt_preempt:entry
    / ((rtproc_t *)args[0]->t_cldata)->rt_timeleft <= 1 /
    {
        tqe_cnt++;
    }

    fbt:RT:rt_preempt:entry
    / ((rtproc_t *)args[0]->t_cldata)->rt_timeleft > 1 /
    {
        hpp_cnt++;
    }

tick-1sec
{
    printf("%-16d %-16d %-16d %-16d\n", csw_cnt, inv_cnt, tqe_cnt, hpp_cnt);

    inv_cnt = 0; tqe_cnt = 0; hpp_cnt = 0; csw_cnt = 0;
}

LISA '09 Baltimore, Md.
```
```

```
solaris10> ./csw.d
TOTAL CSW    ALL INV   TQE_INV   HPP_INV
          ===========  ===========  ===========
 1544        63   24      40
 3667        49   35      14
 4163        59   34      26
 3760        55   29      26
 3839        71   39      32
 3931        48   33      15
^C
```

```
solaris10> ./threads &
[2]   19913
solaris10>
solaris10> ./csw.d
```

```
solaris10> ./csw.d
TOTAL CSW    ALL INV   TQE_INV   HPP_INV
          ===========  ===========  ===========
 3985       1271   125      1149
 5681       1842   199      1648
 5025       1227   151      1080
 9170       520    108      412
 4100       390    84       307
 2487       174    74       99
 1841       113    64       50
 6239       170    74       96
^C
```

```
solaris10> ./csw.d
TOTAL CSW    ALL INV   TQE_INV   HPP_INV
          ===========  ===========  ===========
 1440       155    68       88
```

```
```
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.
- Use `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.
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 that it is not ok to 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
Processors, Processor Controls & Binding
Processor Controls

- Processor controls provide for segregation 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_create**(2), **p_online**(2)
  - system calls to do things programmatically
Processor Sets

- Partition CPU resources for segregating 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 explicitly 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
Session 4
File Systems & Disk I/O Performance
The Solaris File System/IO Stack

- **Application**
- **Files & File Systems**
- **Virtual File System**
  - **VFS**
  - **Virtual Disks**
  - **Virtual Device**
  - **Blocks**
  - **Disks**

- **UFS/VxFS**
- **SVM/VxVM**
- **SCSI/FC**
- **MpxIO/DMP**
- **File System**
- **Volume Manager**
- **Multi-Pathing**
- **Driver Stack**
- **Array**
File System Architecture

FOP Layer

- open()
- close()
- mkdir()
- rmdir()
- rename()
- link()
- unlink()
- seek()
- fsync()
- ioctl()
- create()

UFS
- Paged VNODE VM Core
- (File System Cache)

ZFS
- ZFS ARC

NFS

PROC

Network

Kernel

bdev_strategy() Device Driver Interface

sd

ssd
UFS I/O

Kernel Address

Space

segmap

read() write()

File System

File Segment Driver (seg_map)

VNODE Segment Driver (seg_vn)

Paged VNODE VM Core
(File System Cache & Page Cache)

Process Address

Space

stack

mmap()

text

text
UFS Caching

- File Name Lookups
  - read()
  - write()
  - fread()
  - fwrite()

- Directory Name Cache (ncsize)
- Inode Cache (ufs_ninode)
- Buffer Cache (bufhwm)

Read and write operations can be performed through functions such as read(), write(), fread(), fwrite().

- stdio buffers
- segmap
- Level 1 Page Cache
- Level 2 Page Cache
- Dynamic Page Cache

- Direct Blocks
- mmap()'d files bypass the segmap cache

- Stack
- Heap
- Data
- Text

- Disk Storage

Measure the buffer cache hit rate with kstat -n biostats.
Measure the DNLC hit rate with kstat -n dnlcstats.
Measure the segmap cache hit ratio can be measured with kstat -n segmap.

LISA '09 Baltimore, Md.
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 – IO provider
- DTrace: fsinfo provider
- DTrace: top for files – logical and physical per file
- DTrace: top for fs – logical and physical per filesystem
- DTraceToolkit – iosnoop and iotop
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 its 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

![Diagram showing executing and waiting times]

<table>
<thead>
<tr>
<th>Executing</th>
<th>Waiting</th>
</tr>
</thead>
<tbody>
<tr>
<td>20s</td>
<td>80s</td>
</tr>
</tbody>
</table>
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:

```bash
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
   .
   .
   .
```
Solaris iostat

# iostat -xnz

## Extended Device Statistics

<table>
<thead>
<tr>
<th>r/s</th>
<th>w/s</th>
<th>kr/s</th>
<th>kw/s</th>
<th>wait</th>
<th>actv</th>
<th>wsvec_t</th>
<th>asvc_t</th>
<th>%w</th>
<th>%b</th>
<th>device</th>
</tr>
</thead>
<tbody>
<tr>
<td>687.8</td>
<td>0.0</td>
<td>38015.3</td>
<td>0.0</td>
<td>0.0</td>
<td>1.9</td>
<td>0.0</td>
<td>2.7</td>
<td>0</td>
<td>100</td>
<td>c0d0</td>
</tr>
</tbody>
</table>

- **Queue**: threads queued for I/O
- **Performing I/O**: threads performing I/O

- **Wait**: number of threads queued for I/O
- **Actv**: number of threads performing I/O
- **wsvec_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 utilization – only useful if device can do just 1 I/O at a time (invalid for arrays etc...)

---

LISA '09 Baltimore, Md.
Thread I/O example

sol$ cd labs/disks
sol$ ./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...

sol$ 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
64 Thread I/O example

sol$ cd labs/disks
sol$ ./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...

sol$ 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
 568.0  0.3   571.0  17.3  61.8  2.0  866.5  28.0  100  100  c1d0
Solaris iostat

- 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

```c
#include <stdio.h>
#include <sys/buf.h>
#include <sys/dr.h>

int main() {
    #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 8192
    cmdk0 /opt/Acrobat4/Reader/intelsolaris/bin/acroread R 8192
    cmdk0 /opt/Acrobat4/Reader/intelsolaris/bin/acroread R 8192
```
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$ iotraced.d

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>FILE</th>
<th>RW</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
</tbody>
</table>
# iotop -C

Sampling... Please wait.

2005 Jul 16 00:34:40, load: 1.21, disk_r: 12891 Kb, disk_w: 1087 Kb

<table>
<thead>
<tr>
<th>UID</th>
<th>PID</th>
<th>PPID</th>
<th>CMD</th>
<th>DEVICE</th>
<th>MAJ</th>
<th>MIN</th>
<th>D</th>
<th>BYTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>fsflush</td>
<td>cmdk0</td>
<td>102</td>
<td>4</td>
<td>W</td>
<td>512</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>fsflush</td>
<td>cmdk0</td>
<td>102</td>
<td>0</td>
<td>W</td>
<td>11776</td>
</tr>
<tr>
<td>0</td>
<td>27751</td>
<td>20320</td>
<td>tar</td>
<td>cmdk0</td>
<td>102</td>
<td>16</td>
<td>W</td>
<td>23040</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>fsflush</td>
<td>cmdk0</td>
<td>102</td>
<td>0</td>
<td>R</td>
<td>73728</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>sched</td>
<td>cmdk0</td>
<td>102</td>
<td>0</td>
<td>R</td>
<td>548864</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>sched</td>
<td>cmdk0</td>
<td>102</td>
<td>0</td>
<td>W</td>
<td>1078272</td>
</tr>
<tr>
<td>0</td>
<td>27751</td>
<td>20320</td>
<td>tar</td>
<td>cmdk0</td>
<td>102</td>
<td>16</td>
<td>R</td>
<td>1514496</td>
</tr>
<tr>
<td>0</td>
<td>27751</td>
<td>20320</td>
<td>tar</td>
<td>cmdk0</td>
<td>102</td>
<td>3</td>
<td>R</td>
<td>11767808</td>
</tr>
</tbody>
</table>
**DTraceToolkit - iosnoop**

```
# iosnoop
<table>
<thead>
<tr>
<th>UID</th>
<th>PID</th>
<th>D</th>
<th>BLOCK</th>
<th>SIZE</th>
<th>COMM</th>
<th>PATHNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>3808</td>
<td>8192</td>
<td>tar</td>
<td>/usr/bin/eject</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>35904</td>
<td>6144</td>
<td>tar</td>
<td>/usr/bin/eject</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>39828</td>
<td>6144</td>
<td>tar</td>
<td>/usr/bin/env</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>3872</td>
<td>8192</td>
<td>tar</td>
<td>/usr/bin/expr</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>21120</td>
<td>7168</td>
<td>tar</td>
<td>/usr/bin/expr</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>43680</td>
<td>6144</td>
<td>tar</td>
<td>/usr/bin/false</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>44176</td>
<td>6144</td>
<td>tar</td>
<td>/usr/bin/fdetch</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>3920</td>
<td>8192</td>
<td>tar</td>
<td>/usr/bin/fdformat</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>3936</td>
<td>8192</td>
<td>tar</td>
<td>/usr/bin/fdformat</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>4080</td>
<td>8192</td>
<td>tar</td>
<td>/usr/bin/fdformat</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>9680</td>
<td>3072</td>
<td>tar</td>
<td>/usr/bin/fdformat</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>4096</td>
<td>8192</td>
<td>tar</td>
<td>/usr/bin/fgrep</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>46896</td>
<td>6144</td>
<td>tar</td>
<td>/usr/bin/fgrep</td>
</tr>
<tr>
<td>100</td>
<td>15795</td>
<td>R</td>
<td>4112</td>
<td>8192</td>
<td>tar</td>
<td>/usr/bin/file</td>
</tr>
</tbody>
</table>
```
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 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.
The `vmstat -p` command provides detailed information about the memory usage of a system. The table below shows the output of `vmstat -p` with 5-second intervals over 5 seconds:

<table>
<thead>
<tr>
<th></th>
<th>swap</th>
<th>free</th>
<th>re</th>
<th>mf</th>
<th>fr</th>
<th>de</th>
<th>sr</th>
<th>epi</th>
<th>epo</th>
<th>epf</th>
<th>api</th>
<th>apo</th>
<th>apf</th>
<th>fpi</th>
<th>fpo</th>
<th>fpf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>46715224</td>
<td>891296</td>
<td>24</td>
<td>350</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>46304792</td>
<td>897312</td>
<td>151</td>
<td>761</td>
<td>25</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>280</td>
<td>25</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>45886168</td>
<td>899808</td>
<td>118</td>
<td>339</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>641</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>46723376</td>
<td>899440</td>
<td>29</td>
<td>197</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **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
- **filesystem**: kilobytes in - out - freed
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
2055: 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
   pageio_setup:pgin 40
   pageio_setup:pgpgin 42
   pageio_setup:maj_fault 43
   pageio_setup:fspgin 45
   bdev_strategy:start 52
0  <= pread64
   biodone:done 11599
   pread64 11626
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
2055: 31.425: Starting 1 rand-thread threads
2055: 34.435: Running for 600 seconds...

sol10# ./fspaging.d

<table>
<thead>
<tr>
<th>Event</th>
<th>Device</th>
<th>Path</th>
<th>RW</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>get-page</td>
<td></td>
<td>/filebench/bigfile0</td>
<td></td>
<td>8192</td>
</tr>
<tr>
<td>getpage-io</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>get-page</td>
<td></td>
<td>/filebench/bigfile0</td>
<td></td>
<td>8192</td>
</tr>
<tr>
<td>getpage-io</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>get-page</td>
<td></td>
<td>/filebench/bigfile0</td>
<td></td>
<td>8192</td>
</tr>
<tr>
<td>getpage-io</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>R</td>
<td>8192</td>
</tr>
<tr>
<td>get-page</td>
<td></td>
<td>/filebench/bigfile0</td>
<td></td>
<td>8192</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>Event</th>
<th>Device</th>
<th>Path</th>
<th>RW</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>put-page</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>W</td>
<td>8192</td>
<td>18702224</td>
</tr>
<tr>
<td>other-io</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>W</td>
<td>512</td>
<td>69219</td>
</tr>
<tr>
<td>put-page</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>W</td>
<td>8192</td>
<td>11562912</td>
</tr>
<tr>
<td>other-io</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>W</td>
<td>512</td>
<td>69220</td>
</tr>
<tr>
<td>put-page</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>W</td>
<td>8192</td>
<td>10847040</td>
</tr>
<tr>
<td>other-io</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>W</td>
<td>512</td>
<td>69221</td>
</tr>
<tr>
<td>put-page</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>W</td>
<td>8192</td>
<td>22170752</td>
</tr>
<tr>
<td>other-io</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>W</td>
<td>512</td>
<td>69222</td>
</tr>
<tr>
<td>put-page</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>W</td>
<td>8192</td>
<td>25189616</td>
</tr>
<tr>
<td>other-io</td>
<td>cmdk0</td>
<td>/filebench/bigfile0</td>
<td>W</td>
<td>512</td>
<td>69223</td>
</tr>
</tbody>
</table>
### fsinfo(1)

```bash
# fsstat ufs 1

<table>
<thead>
<tr>
<th>new</th>
<th>name</th>
<th>name</th>
<th>attr</th>
<th>attr</th>
<th>lookup</th>
<th>rddir</th>
<th>read</th>
<th>read</th>
<th>write</th>
<th>write</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>file</td>
<td>remov</td>
<td>chng</td>
<td>get</td>
<td>set</td>
<td>ops</td>
<td>ops</td>
<td>ops</td>
<td>bytes</td>
<td>bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>96.0K</td>
<td>91.7K</td>
<td>48</td>
<td>10.6M</td>
<td>46.0K</td>
<td>40.6M</td>
<td>429K</td>
<td>362K</td>
<td>57.2G</td>
<td>8.46M</td>
<td>67.3G</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>352</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>176</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>176</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>176</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2.47K</td>
<td>3</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>176</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>176</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>505</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>176</td>
</tr>
</tbody>
</table>
```
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.
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 UFS Cache

Sol 8 (and beyond) segmap

Kernel Memory

segmap

process memory
heap, data, stack

cachelist

reclaim

freelist
Tuning segmap (UFS L1 cache)

- By default, on SPARC, 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%
  - segkpm in Solaris 10 and OpenSolaris
- On Solaris 10 on X64, segmap is 64MB by default
  - Tune with segmapsize in /etc/system or eeprom
    - set segmapsize = 1073741824 (1 GB)
  - On 32-bit X64, max segmapsize is 128MB
Tuning segmap_percent

- There are kstat statistics for segmap hit rates
- Estimate hit rate as $(\text{get\_reclaim} + \text{get\_use}) / \text{getmap}$

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

crtype                          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 use this extensively
  • See aio_read(), aio_write() etc...

• Slightly different variants 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
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
- 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
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
- What is the I/O size in our example?
  - \( \frac{38015}{687} = 56k \)
  - Too small for best sequential performance

```bash
# iostat -xnz

extended device statistics

<table>
<thead>
<tr>
<th>r/s</th>
<th>w/s</th>
<th>kr/s</th>
<th>kw/s</th>
<th>wait</th>
<th>actv</th>
<th>wsvc_t</th>
<th>asvc_t</th>
<th>%w</th>
<th>%b</th>
<th>device</th>
</tr>
</thead>
<tbody>
<tr>
<td>687.8</td>
<td>0.0</td>
<td>38015.3</td>
<td>0.0</td>
<td>0.0</td>
<td>1.9</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>100</td>
<td>c0d0</td>
</tr>
</tbody>
</table>
```
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/...
```
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_readAhead=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 many 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 \( ufs_{HW} \leq <\text{bytes}> \)
    - Resume when 256k of unwritten cache
      - Set \( ufs_{HW} \leq <\text{bytes}> \)
- Solaris 9+ Defaults
  - Block when >16MB of unwritten cache
  - Resume when <8MB of unwritten cache
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
- Remember, it's a system-wide setting
- Use sparingly, only applications which don't want caching will benefit
  - 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
ZFS

- Started from scratch with today's problems in mind
- Pooled Storage
  - Do for storage what VM does for RAM
- End-to-End Data integrity
  - Block-level checksum
  - Self-correcting when redundant data available
  - No more silent data corruption
- Transaction Model
  - COW updates – no changes to on-disk data
  - FS on-disk integrity maintained
  - Many opportunities for performance optimizations (IO scheduler and transaction reordering)
- Massive Scale – 128 bits
FS/Volume Model vs. Pooled Storage

**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
- All storage in the pool is shared
ZFS Data Integrity Model

- Copy-on-write, transactional design
- Everything is checksummed
- RAID-Z/Mirroring protection
- Ditto Blocks
- Disk Scrubbing
- Write Failure Handling
Copy-on-Write and Transactional

- Uber-block
- Initial block tree
- Writes a copy of some changes
- Original Data
- New Data
- Original Pointers
- New Pointers
- Copy-on-write of indirect blocks
- Rewrites the Uber-block
- New Uber-block

LISA '09 Baltimore, Md.
Measurements at CERN

• Wrote a simple application to write/verify 1GB file
  • Write 1MB, sleep 1 second, etc. until 1GB has been written
  • Read 1MB, verify, sleep 1 second, etc.
• Ran on 3000 rack servers with HW RAID card
• After 3 weeks, found 152 instances of silent data corruption
  • Previously thought “everything was fine”
• HW RAID only detected “noisy” data errors
• Need end-to-end verification to catch silent data corruption
End-to-End Checksums

Checksums are separated from the data

Entire I/O path is self-validating (uber-block)

Prevents:
- Silent data corruption
- Panics from corrupted metadata
- Phantom writes
- Misdirected reads and writes
- DMA parity errors
- Errors from driver bugs
- Accidental overwrites
Disk Scrubbing

- Uses checksums to verify the integrity of all the data
- Traverses metadata to read every copy of every block
- Finds latent errors while they're still correctable
- It's like ECC memory scrubbing – but for disks
- Provides fast and reliable re-silvering of mirrors
Copy-on-Write Design
Multiple Block Sizes
Pipelined I/O
Dynamic Striping
Intelligent Pre-Fetch
Separate Intent Log
Dedicated cache device

Architected for Speed
Variable Block Size

- No single block size is optimal for everything
  - Large blocks: less metadata, higher bandwidth
  - Small blocks: more space-efficient for small objects
  - Record-structured files (e.g. databases) have natural granularity; filesystem must match it to avoid read/modify/write

- Why not arbitrary extents?
  - Extents don't COW or checksum nicely (too big)
  - Large blocks suffice to run disks at platter speed

- Per-object granularity
  - A 37k file consumes 37k – no wasted space

- Enables transparent block-based compression
ZFS Intent Log (ZIL)

- Satisfies synchronous writes semantics
  - O_SYNC/O_DSYNCH
- Blocks are allocated from the main pool
  - Guaranteed to be written to stable storage before system call returns
- Examples:
  - Database often utilize synchronous writes to ensure transactions are on stable storage
  - NFS and other applications can issue fsync() to commit prior to writes
Separate Intent Log (slog)

- Leverages high speed devices for dedicated intent log processing
  - Low latency devices such as SSDs (aka Logzilla)
- Can be mirrored and striped
- Blocks are allocated from dedicated log device
  - Failure reverts back to general pool

**Example:** Create a pool with a dedicated log device

```
# zpool create tank mirror c0d0 c1d0 log c2d0
```
Adaptive Replacement Cache (ARC)

- Scan-resistant LRU (least recently used)
- Cache size divided into two:
  - Used once
  - Used multiple times
- Automatically adjust to memory pressure and workload
  - Data which is not being referenced is evicted
  - Ratio of once/multiple adjust dynamically based on workload
L2ARC – cache device

- Provides a level of caching between main memory and disk
  - Utilizes specialized read-biased SSDs to extend the cache (aka “Readzilla”)
- Asynchronously populates the cache
  - Moves blocks from the ARC to L2ARC cache device

Example: Create a pool with a cache device

```bash
# zpool create tank mirror c0d0 c1d0 cache c2d0
```
Typical way to Improve Performance

- Buy lots of RAM
  - Cache as much as possible
  - Use DRAM to compensate for slower disks
- Use lots of spindles
  - Spread the load across as many devices as possible
  - Use the outer most cylinders of the disk (make sure the disks don't seek)
  - Use NVRAM
- Throw $$$ at the problem
How to get terrible performance

- Run against storage array that flush caches
- Run simple benchmarks without decoding the numbers
  - compare write to cache vs write to disk
- Run the pool at 95% disk full
- Do random reads from widest raid-z
- Run a very large DB without tuning the recordsize
- Don't provision enough CPU
- Don't configure swap space
- Don't read the ZFS Best Practices
How to get Great performance

- small files (<128K)
  - ufs allocates 1 inode per MB
  - netapps 1 / 32K
  - ZFS uses 1.2K to store 1K files !!!
  - Create 10s of files per single I/Os
  - $ miss reads == single disk I/O

- ZFS does constant time snapshot
  - it's basically a noop to take a snapshot
  - snap deletions proportional to changes
  - snapshots helps simplify your business
How to get Great performance

- Run ZFS in the storage back end (7000 Storage)
- Or provision for CPU usage.
- Configure enough RPM
  - 2 Mirrored 7.2 K RPM vs 1 x 15 K RPM in Raid-5
- Move Spindle Constrained setup to ZFS
  - write streaming + I/O aggregation
    - efficient use of spindles on writes,
    - 100% full stripes in storage
  - free spindles for reads
  - use a separate intent log (NVRAM or SSD or just N separate spindles) for an extra boost
Solaris 10 Update 6

- Finally got write throttling, ZFS won't eat all of memory
  - Grows and shrink dance now as designed
  - Capping the ARC seems commonly done
  - ZFS reports accurate freemem, others cache data in freemem
- Cache flushes to SAN array partially solved
  - HDS, EMC with recent firmware are ok.
  - Can be tuned per array
  - Others? set zfs_nocacheflush (cf evil tuning guide)
- Vdev level prefetching is auto tuning
  - no problems there
Solaris 10 Update 6

- We have the separate intent log
  - one or a few disks, but preferably SSD or NVRAM/DRAM device

Upcoming

- L2 ARC
  - on/off per dataset
- ARC
  - on/off per dataset, ~directio
- Storage 7000
  - Tracks Nevada
Tuning is Evil

• Leave a trace, explain motivation
  • zfs_nocacheflush (on storage arrays that do)
  • capping the ARC (to preserve large pages)
  • zfs_prefetch_disable (zfextract consuming cpus)
  • zfs_vdev_max_pending (default 35, 10-16 for DB)
  • zil_disable (NO!!! don't or face application corruptions)

• No tuning required
  • vdev prefetch (issue now fixed)
ZFS Best Practices

- Tune recordsize only on fixed records DB files
- Mirror for performance
- 64-bit kernel (allows greater ZFS caches)
- configure swap (don't be scared by low memory)
- Don't slice up devices (confuses I/O scheduler)
- For raid-z[2] : don't go two wide (for random reads)
- Isolate DB log writer if that is critical (use few devices)
- Separate Root pool (system's identify) and data pools (system's function)
ZFS Best Practices

• Don't mix legacy and non-legacy shares (it's confusing)
• 1 FS per user (1 quota/reserv; user quota are coming)
• Rolling Snapshots (smf service)
• Instruct backup tool to skip .zfs
• Keep pool below 80% full (helps COW)
MySQL Best Practices

- Match Recordsize with DB (16K)
- Use a separate intent log device within main zpool
- Find creative use of Snapshot/Clones send/recv
  - backups
  - master & slave architecture
- Use the ARC and L2ARC instead of disk RPM
  - a caching 7000 series serving masters & slaves
- NFS Directio and Jumbo Frames
  - save CPU cycles and memory for application
  - Set innodb_doublewrite=0
- Linux
  - innodb_flush_method = O_DIRECT
  - echo noop > /sys/block/sde/queue/scheduler
Tuning & Best Practices

- Tuning and BP wikis

- ZFS Dynamics : In-Depth view

- Blue Prints
  - http://wikis.sun.com/display/BluePrints/Main

- Performance Savvy Bloggers
  - joyent (Ben Rockwood), Smugmug (Don Mcaskill), Neel (Oracle and MySQL), Media Temple
Networks
Some Useful Numbers

- 1Gbit – theoretical 134MB/sec (megaBYTES)
- 10Gbit – theoretical 1.3GB/sec (gigaBYTES)

<table>
<thead>
<tr>
<th>Bus</th>
<th>Technology</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI</td>
<td>32bit/33Mhz</td>
<td>133MB/sec</td>
</tr>
<tr>
<td>PCI</td>
<td>32bit/66Mhz</td>
<td>266MB/sec</td>
</tr>
<tr>
<td>PCI</td>
<td>64bit/66Mhz</td>
<td>530MB/sec</td>
</tr>
<tr>
<td>PCI-X</td>
<td>64bit/133Mhz</td>
<td>1GB/sec</td>
</tr>
<tr>
<td>PCI Express</td>
<td>1 lane, v1.X</td>
<td>250MB/sec</td>
</tr>
<tr>
<td>PCI Express</td>
<td>2 lanes, v1.X</td>
<td>500MB/sec</td>
</tr>
<tr>
<td>PCI Express</td>
<td>4 lanes, v1.X</td>
<td>1GB/sec</td>
</tr>
<tr>
<td>PCI Express</td>
<td>8 lanes, v1.x</td>
<td>2GB/sec (1.7GB/sec)</td>
</tr>
</tbody>
</table>
Networks...

- Network performance and tuning is probably the most difficult to setup and measure
  - Hardware choices (NICs, PCI-Express attributes, platform-specific (SPARC versus X64) attributes), network switches
  - Software layers – TCP/IP stack, platform-specific device stack, platform-independent device stack...
  - OS releases (Solaris 10 updates versus NV)
    - software churn
    - Resource allocation – CPU to support load
    - Tuning methods - /etc/system and ndd(1M)
- Bandwidth is often the quoted performance metric
  - And it's important, but...
  - Many workloads care more about packets-per-second and latency
NICs and Drivers

- The device name (ifconfig -a) is the driver
  - It's possible for multiple drivers to be available for the same hardware, i.e. configuring T2000 NICs with either e1000g or ipge (note: e1000g is better)
NICs and Drivers
NIC Tuneables
Networks

• Key Observables
  • Link Utilization
  • Transmission, framing, checksum errors
  • Upstream software congestion
  • Routing
  • Over the wire latency

• What to measure
  • Link Bandwidth: nicstat
  • Link Speed: checkcable
  • Dropped upstream packets (nocanput)
Networking - Tools

- netstat – kstat based, packet rates, errors, etc
- kstat – raw counters for NICs and TCP/UDP/IP
- nx.se – SE toolkit utility for bandwidth
- nicstat – NIC utilization
- snmpnetstat – network stats from SNMP
- checkcable – NIC status
- ping – host status
- traceroute – path to host, latency and hops
- snoop – network packets
- TTCP – workload generator
- pathchar – path to host analysis
- ntop – network traffic sniffer
- tcptop – DTrace tool, per process network usage
- tcpsnoop – DTrace tool, network packets by process
- dtrace – TCP, UDP, IP, ICMP, NIC drivers, etc.
# netstat -i 1

<table>
<thead>
<tr>
<th>input packets</th>
<th>bge0 packets</th>
<th>output packets</th>
<th>input (Total) packets</th>
<th>output packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>15381402</td>
<td>1280618</td>
<td>0</td>
<td>15384170</td>
<td>0</td>
</tr>
<tr>
<td>160</td>
<td>0</td>
<td>177</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>213</td>
<td>0</td>
<td>205</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>138</td>
<td>0</td>
<td>159</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>159</td>
<td>0</td>
<td>172</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>215</td>
<td>0</td>
<td>213</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

# kstat -p bge:0:bge0:*bytes64

- bge:0:bge0:obytes64: 969276250
- bge:0:bge0:rbytes64: 1917373531

#
TCP

# netstat -s | grep Bytes

tcpOutDataSegs    =862707  tcpOutDataBytes     =879539866
tcpRetransSegs   =  1189  tcpRetransBytes     =1159401
tcpInAckSegs     =473250  tcpInAckBytes       =879385959
tcpInInorderSegs =694607  tcpInInorderBytes =623233594
tcpInUnorderSegs =  3926  tcpInUnorderBytes =4877730
tcpInDupSegs     =   187  tcpInDupBytes       = 75281
tcpInPartDupSegs =     6  tcpInPartDupBytes =   7320
tcpInPastWinSegs =     0  tcpInPastWinBytes =     0

# kstat -n tcp
module: tcp                         instance: 0
name: tcp                           class: mib2
activeOpens                     4809
attemptFails                    22
connTableSize                   56
connTableSize6                  84
crttime                        237.364807266
...
$ nicstat 1

<table>
<thead>
<tr>
<th>Time</th>
<th>Int</th>
<th>rKb/s</th>
<th>wKb/s</th>
<th>rPk/s</th>
<th>wPk/s</th>
<th>rAvs</th>
<th>wAvs</th>
<th>%Util</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:33:04</td>
<td>hme0</td>
<td>1.51</td>
<td>4.84</td>
<td>7.26</td>
<td>10.32</td>
<td>213.03</td>
<td>480.04</td>
<td>0.05</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:05</td>
<td>hme0</td>
<td>0.20</td>
<td>0.26</td>
<td>3.00</td>
<td>3.00</td>
<td>68.67</td>
<td>90.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:06</td>
<td>hme0</td>
<td>0.14</td>
<td>0.26</td>
<td>2.00</td>
<td>3.00</td>
<td>73.00</td>
<td>90.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:07</td>
<td>hme0</td>
<td>0.14</td>
<td>0.52</td>
<td>2.00</td>
<td>6.00</td>
<td>73.00</td>
<td>88.00</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:08</td>
<td>hme0</td>
<td>0.24</td>
<td>0.36</td>
<td>3.00</td>
<td>4.00</td>
<td>81.33</td>
<td>92.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:09</td>
<td>hme0</td>
<td>2.20</td>
<td>1.77</td>
<td>16.00</td>
<td>18.00</td>
<td>140.62</td>
<td>100.72</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:10</td>
<td>hme0</td>
<td>0.49</td>
<td>0.58</td>
<td>8.00</td>
<td>9.00</td>
<td>63.25</td>
<td>66.00</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:11</td>
<td>hme0</td>
<td>12.16</td>
<td>1830.38</td>
<td>185.06</td>
<td>1326.42</td>
<td>67.26</td>
<td>1413.06</td>
<td>15.09</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:12</td>
<td>hme0</td>
<td>19.03</td>
<td>3094.19</td>
<td>292.88</td>
<td>2229.11</td>
<td>66.53</td>
<td>1421.40</td>
<td>25.50</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:13</td>
<td>hme0</td>
<td>19.55</td>
<td>3151.87</td>
<td>301.00</td>
<td>2270.98</td>
<td>66.50</td>
<td>1421.20</td>
<td>25.98</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:14</td>
<td>hme0</td>
<td>11.99</td>
<td>1471.67</td>
<td>161.07</td>
<td>1081.45</td>
<td>76.25</td>
<td>1393.49</td>
<td>12.15</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:15</td>
<td>hme0</td>
<td>0.14</td>
<td>0.26</td>
<td>2.00</td>
<td>3.00</td>
<td>73.00</td>
<td>90.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:16</td>
<td>hme0</td>
<td>0.14</td>
<td>0.26</td>
<td>2.00</td>
<td>3.00</td>
<td>73.00</td>
<td>90.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>12:33:17</td>
<td>hme0</td>
<td>0.14</td>
<td>0.26</td>
<td>2.00</td>
<td>3.00</td>
<td>73.00</td>
<td>90.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

http://blogs.sun.com/timc
http://www.brendangregg.com
**tcptop**

http://www.brendangregg.com/dtrace.html#DTraceToolkit

```
# tcptop -C 10

Sampling... Please wait.

2005 Jul  5 04:55:25,  load: 1.11,  TCPin: 2 Kb,  TCPout: 110 Kb

<table>
<thead>
<tr>
<th>UID</th>
<th>PID</th>
<th>LADDR</th>
<th>LPORT</th>
<th>FADDR</th>
<th>FPORT</th>
<th>SIZE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>20876</td>
<td>192.168.1.5</td>
<td>36396</td>
<td>192.168.1.1</td>
<td>79</td>
<td>1160</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20875</td>
<td>192.168.1.5</td>
<td>36395</td>
<td>192.168.1.1</td>
<td>79</td>
<td>1160</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20878</td>
<td>192.168.1.5</td>
<td>36397</td>
<td>192.168.1.1</td>
<td>23</td>
<td>1303</td>
<td>telnet</td>
</tr>
<tr>
<td>100</td>
<td>20877</td>
<td>192.168.1.5</td>
<td>859</td>
<td>192.168.1.1</td>
<td>514</td>
<td>115712</td>
<td>rcp</td>
</tr>
</tbody>
</table>

2005 Jul  5 04:55:35,  load: 1.10,  TCPin: 0 Kb,  TCPout: 0 Kb

<table>
<thead>
<tr>
<th>UID</th>
<th>PID</th>
<th>LADDR</th>
<th>LPORT</th>
<th>FADDR</th>
<th>FPORT</th>
<th>SIZE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>242</td>
<td>192.168.1.5</td>
<td>79</td>
<td>192.168.1.1</td>
<td>54220</td>
<td>272</td>
<td>inetd</td>
</tr>
<tr>
<td>0</td>
<td>20879</td>
<td>192.168.1.5</td>
<td>79</td>
<td>192.168.1.1</td>
<td>54220</td>
<td>714</td>
<td>in.fingerd</td>
</tr>
</tbody>
</table>

[...]
# tcpsnoop.d

<table>
<thead>
<tr>
<th>UID</th>
<th>PID</th>
<th>LADDR</th>
<th>LPORT</th>
<th>DR</th>
<th>RADDR</th>
<th>RPORT</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>20892</td>
<td>192.168.1.5</td>
<td>36398</td>
<td>-&gt;</td>
<td>192.168.1.1</td>
<td>79</td>
<td>54</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20892</td>
<td>192.168.1.5</td>
<td>36398</td>
<td>&lt;-</td>
<td>192.168.1.1</td>
<td>79</td>
<td>66</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20892</td>
<td>192.168.1.5</td>
<td>36398</td>
<td>-&gt;</td>
<td>192.168.1.1</td>
<td>79</td>
<td>54</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20892</td>
<td>192.168.1.5</td>
<td>36398</td>
<td>-&gt;</td>
<td>192.168.1.1</td>
<td>79</td>
<td>56</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20892</td>
<td>192.168.1.5</td>
<td>36398</td>
<td>&lt;-</td>
<td>192.168.1.1</td>
<td>79</td>
<td>54</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20892</td>
<td>192.168.1.5</td>
<td>36398</td>
<td>&lt;-</td>
<td>192.168.1.1</td>
<td>79</td>
<td>606</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20892</td>
<td>192.168.1.5</td>
<td>36398</td>
<td>-&gt;</td>
<td>192.168.1.1</td>
<td>79</td>
<td>54</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20892</td>
<td>192.168.1.5</td>
<td>36398</td>
<td>&lt;-</td>
<td>192.168.1.1</td>
<td>79</td>
<td>54</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20892</td>
<td>192.168.1.5</td>
<td>36398</td>
<td>-&gt;</td>
<td>192.168.1.1</td>
<td>79</td>
<td>54</td>
<td>finger</td>
</tr>
<tr>
<td>100</td>
<td>20892</td>
<td>192.168.1.5</td>
<td>36398</td>
<td>&lt;-</td>
<td>192.168.1.1</td>
<td>79</td>
<td>54</td>
<td>finger</td>
</tr>
<tr>
<td>0</td>
<td>242</td>
<td>192.168.1.5</td>
<td>23</td>
<td>&lt;-</td>
<td>192.168.1.1</td>
<td>54224</td>
<td>54</td>
<td>inetd</td>
</tr>
<tr>
<td>0</td>
<td>242</td>
<td>192.168.1.5</td>
<td>23</td>
<td>-&gt;</td>
<td>192.168.1.1</td>
<td>54224</td>
<td>54</td>
<td>inetd</td>
</tr>
</tbody>
</table>

...
dtrace

# dtrace -n 'fbt:ip::entry { @[probefunc] = count(); }'
dtrace: description 'fbt:ip::entry' matched 1875 probes
^C

... tcp_set_rto 2
tcp_timeout_cancel 2
tcp_timer_free 2
tcp_wput_data 2
ip_input 3
ip_loopback_src_or_dst 3
ip_tcp_input 3
ipcl_classify_v4 3
ire_cache_lookup 3
squeue_enter_chain 3
tcp_find_pktinfo 3
tcp_input 3
```
# dtrace -n 'fbt:bge::entry { @[probefunc] = count(); }'
dtrace: description 'fbt:bge::entry ' matched 164 probes
^C
  bge_atomic_renounce                                   1
  bge_atomic_claim                                     2
  bge_atomic_reserve                                   2
  bge_send                                             2
  bge_m_tx                                             3
  bge_atomic_shl32                                     6
  bge_chip_factotum                                    6
  bge_factotum_link_check                               6
  bge_factotum_stall_check                             6
  bge_mbx_put                                          10
  bge_intr                                             11
  bge_receive                                          11
  bge_recycle                                          11
  bge_chip_cyclic                                      12
...```
Summary

• Systems (hardware + software) are extremely complex
  • Understanding behavior requires a variety of skills
• Experience is your friend
  • Dive in!
• The Good News
  • Solaris/OpenSolaris is SECOND TO NONE when it comes to tools, utilities and observability
    • ...and some other things to
  • With few exceptions, typing the wrong thing won't hurt
    • exceptions include truss(1), dtrace with thousands of probes, dtrace PID provider with tens-of-thousands of probes
• With Solaris 10 and OpenSolaris, All questions can and will be answered
  • The Dark Ages are behind us...
  • Come to the light...
Solaris 10 & OpenSolaris Performance, Observability & Debugging (POD)

Jim Mauro  
Sun Microsystems, Inc  
james.mauro@sun.com  

Richard McDougall  
VMware  
rmc@vmware.com
Supplemental Material
Memory Corruption & Memory Leak Detection
The Problem

- Memory leaks
  - allocated memory is not freed when the program no longer needs it
  - Process address space size continues to grow
    - 32-bit processes can run out of address space
    - 64-bit processes can consume a LOT of memory
- Memory corruption
  - Overwriting a segment boundary
  - Unsynchronized writes into a mapped segment
Memory Leak/Corruption Tools

- **DTrace**
  - Not the best tool for this job
  - pid provider can instrument malloc/free calls
    - track addresses and call stacks

- **dbx**
  - Has facilities for memory access errors and memory leak detection

- **watchmalloc(3MALLOCl**
  - binary replacement for libc malloc, free, etc
  - Environmental variable settings for debugging

- **libumem(3LIB)**
  - Extended feature set for debugging memory leaks and corruption
  - Used in conjunction with mdb(1)
Memory Leak - DTrace

- Use the pid provider to instrument malloc/free calls
  - Post process to align mallocs & frees
- Grab a stack frame

```bash
#!/usr/sbin/dtrace -s
pid$target:$1:malloc:entry
{
    ustack();
}
pid$target:$1:malloc:return
{
    printf("%s: %x\n", probefunc, arg1);
}
pid$target:$1:free:entry
{
    printf("%s: %x\n", probefunc, arg0);
}
```
which is 184 bytes above the current stack pointer

Variable is 'j'

stopped in access_error at line 26 in file "hello.c"

26       i = j;

(dbx) cont

hello world

Checking for memory leaks...

Actual leaks report   (actual leaks:  1 total size:    32 bytes)

Memory Leak (mel):
Found leaked block of size 32 bytes at address 0x100101768
At time of allocation, the call stack was:
   [1] memory_leak() at line 19 in "hello.c"
   [2] main() at line 31 in "hello.c"

Possible leaks report  (possible leaks: 0 total size:     0 bytes)

Checking for memory use...

Blocks in use report    (blocks in use:  1 total size:    12 bytes)

Block in use (biu):
Found block of size 12 bytes at address 0x100103778 (100.00% of

At time of allocation, the call stack was:
   [1] memory_use() at line 11 in "hello.c"
   [2] main() at line 32 in "hello.c"

execution completed, exit code is 0
watchmalloc(3MALLOC)

- Binary replacement for malloc, etc
  - Access via LD_PRELOAD=watchmalloc.so.1
- Uses watchpoint facility of procfs (/proc)
- Imposes some constraints
  - See the man page
- Enabled via environmental variables
  - MALLOC_DEBUG=WATCH,RW,STOP
    - WATCH – WA_WRITE SIGTRAP
    - RW – enables WA_READ & WA_WRITE
    - STOP – Stop process instead of SIGTRAP
watchmalloc(3MALLOCA)

buf[8] 8060df0
buf[9] 8060df1
buf[10] 8060df2
buf[12] 8060df4
buf[13] 8060df5
buf[14] 8060df6
buf[15] 8060df7
buf[16] 8060df8

Trace/Breakpoint Trap (core dumped)
opensolaris> pstack core
core 'core' of 841: ./mover 8
 08050b25 main (2, 80479c8, 80479d4, 80479bc) + c5
 080509cd _start (2, 8047ae8, 8047af0, 0, 8047af2, 8047b08) + 7d
opensolaris> dbx mover core
For information about new features see `help changes'
To remove this message, put `dbxenv suppress_start_up_message 7.7' in your .dbxrc
Reading mover
core file header read successfully
Reading ld.so.1
Reading watchmalloc.so.1
Reading libc.so.1
program terminated by signal TRAP (write access watchpoint trap)
0x08050b25: main+0x00c5: movb %al,0x00000000(%edx)
(dbx)
libumem.so

- Binary replacement for malloc, etc
  - enable via LD_PRELOAD
- Designed after the kernel heap allocator
  - Scalable. Less lock contention with threaded apps
- Also enables debug features for memory leak/access issues
  - special buffer management scheme for detecting memory issues

http://developers.sun.com/solaris/articles/libumem_library.html
libumem.so

- Enable with LD_PRELOAD
- Set UMEM_DEBUG & UMEM_LOGGING
  - UMEM_DEBUG=default
  - UMEM_LOGGING=transaction
- Records Thread ID, Timestamp and stack trace for every memory transaction
- Fills allocated & freed segments within the buffer with special patterns
  - Detect use of uninitialized data and previously freed buffers
- Check redzone
- Debug metadata for buffer audit information
libumem.so

• Detects
  • Buffer overruns
  • Multiple frees
  • Use of uninitialized data
  • Use of freed buffers

• Leak detection
  • Most useful when `UMEM_DEBUG` is set to at least default
  • Can use `mdb(1)'s ::findleaks dcmod to search for memory leaks in core files and running processes
  • Get leaked memory summary, breakdown by stack trace
SunStudio
SunStudio 12

- Compilers
  - C, C++, FORTRAN
- Thread Analyzer
- Performance Analyzer
- NetBeans IDE
- Add-on performance libraries
- Misc Tools
  - dbx
  - lint
  - cscope
  - etc...
Thread Analyzer

- Detects data races and deadlocks in a multithreaded application
  - Points to non-deterministic or incorrect execution
  - Bugs are notoriously difficult to detect by examination
  - Points out actual and potential deadlock situations

- Process
  - Instrument the code with `-xinstrument=datarace`
  - Detect runtime condition with `collect -r all [or race, detection]`
  - Use graphical analyzer to identify conflicts and critical regions
Performance Analyzer

- Thread analyzer integrated into performance analyzer
  - Extensions to the .er files to accommodate THA data
  - collect command extensions
  - er_print command extensions
- More extensive data collection
  - function, instruction count, dataspace profiling
  - attach to PID and collect data
- Probe effect can be mitigated
  - Reduce sampling rates when a lot of threads, or long-running collection

http://blogs.sun.com/d/entry/analyzer_probe_effect
http://blogs.sun.com/d/entry/analyzer_probe_effects_part_2
Performance Analyzer Probe Effect

Graph courtesy of Darryl Gove, from http://blogs.sun.com/d/entry/analyzer_probe_effect
Performance Analyzer

- **Collector Tool**
  - `collect(1)`
  - profiles code and traces function calls
  - call stacks, microstates, hardware counters, memory allocation data, summary information

- **Analyzer Tool**
  - `er_print(1), analyzer(1)`
  - Clock profiling metrics
  - Hardware counter metrics
  - Synchronization delay metrics
  - Memory allocation metrics
  - MPI tracing metrics
Performance Analyzer

- Clock-based profiling
  - Thread state sampled/stored at regular intervals (SIGPROF)
  - Default resolution of 10 milliseconds
  - High-res of 1ms possible
  - Low-res of 100ms for longer collections

User CPU time: LWP time spent running in user mode on the CPU.
Wall time LWP: time spent in LWP 1. This is usually the “wall clock time”
Total LWP time: Sum of all LWP times.
System CPU time: LWP time spent running in kernel mode on the CPU or in a trap state.
Wait CPU time: LWP time spent waiting for a CPU.
User lock time: LWP time spent waiting for a lock.
Text page fault time: LWP time spent waiting for a text page.
Data page fault time: LWP time spent waiting for a data page.
Other wait time: LWP time spent waiting for a kernel page, or time spent sleeping or stopped.
Performance Analyzer

- Hardware Counter Overflow Profiling Data
  -
Hardware Counters

• PIC – Programmable Interval Counters
• Can be programmed to count hardware events (e.g. L2 cache miss, TLB miss, etc)
  • VERY processor-specific
  • Need to reference processor's PRM
• cpustat, cputrack
  • Solaris commands to setting PICs and tracking events
• Perf Analyzer & DTrace cpc provider
  • Use overflow profiling
cpustat from a X4600

Use cputrack(1) to monitor per-process statistics.

CPU performance counter interface: AMD Opteron & Athlon64

event specification syntax:
[picn=]<eventn>[,attr[n][=<val>]][,[picn=]<eventn>[,attr[n][=<val>]],...]

Generic Events:

event[0-3]: PAPI_br_ins PAPI_br_msp PAPI_br_tkn PAPI_fp_ops
    PAPI_fad_ins PAPI_fml_ins PAPI_fpu_idl PAPI_tot_cyc
    PAPI_tot_ins PAPI_l1_dca PAPI_l1_dcm PAPI_l1_idm
    PAPI_l1_stm PAPI_l1_ica PAPI_l1_icm PAPI_l1_icr
    PAPI_l2_dch PAPI_l2_dcm PAPI_l2_dcr PAPI_l2_dcw
    PAPI_l2_ich PAPI_l2_icm PAPI_l2_idm PAPI_l2_stm
    PAPI_res_stl PAPI_stl_icy PAPI_hw_int PAPI_tlb_dm
    PAPI_tlb_im PAPI_fp_ins PAPI_vec_ins

See generic_events(3CPC) for descriptions of these events.
Platform Specific Events:

- FP_dispatched_fpu_ops
- FP_cycles_no_fpu_ops_retired
- LS_seg_reg_load
- LS_uarch_resync_self_modify
- LS_uarch_resync_snoop
- LS_buffer_2_full
- LS_retired_cflush
- LS_retired_cpuid
- DC_access
- DC_miss
- DC_refill_from_L2
- DC_refill_from_system
- BU_memory_requests
- BU_data_prefetch
- BU_cpu_clk_unhalted
- IC_fetch
- IC_miss

- IC_refill_from_L2
- IC_refill_from_system
- IC_itlb_L1_miss_L2_hit
- IC_uarch_resync_snoop
- IC_instr_fetch_stall
- IC_return_stack_hit
- IC_return_stack_overflow
- FR_retired_x86_instr_w_excp_intr

- DC_copyback
- DC_dtlb_L1_miss_L2_hit
- DC_dtlb_L1_miss_L2_miss
- DC_1bit_ecc_error_found
- BU_system_read_responses
- BU_quadwords_written_to_system
- BU_internal_L2_req
- BU_fill_req_missed_L2
- BU_fill_into_L2
- IC_itlb_L1_miss_L2_miss
- FR_retired_fpu_instr

- NB_mem_ctrlr_page_access
- NB_mem_ctrlr_page_table_overflow
- NB_mem_ctrlr_turnaround
- NB_ECC_errors
- NB_sized_commands
- NB_probe_result
- NB_gart_events
- NB_ht_bus0_bandwidth
- NB_ht_bus1_bandwidth
- NB_ht_bus2_bandwidth
- NB_sized_blocks
- NB_cpu_io_to_mem_io
- NB_cache_block_commands

 attributes: edge pc inv cmask umask nouser sys

See Chapter 10 of the "BIOS and Kernel Developer's Guide for the
AMD Athlon 64 and AMD Opteron Processors," AMD publication #26094
Well-known HW counters available for profiling:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Information and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cycles[0</td>
<td>1</td>
<td>2</td>
<td>3],9999991</td>
</tr>
<tr>
<td>insts[0</td>
<td>1</td>
<td>2</td>
<td>3],9999991</td>
</tr>
<tr>
<td>ic[0</td>
<td>1</td>
<td>2</td>
<td>3],100003</td>
</tr>
<tr>
<td>icm[0</td>
<td>1</td>
<td>2</td>
<td>3],100003</td>
</tr>
<tr>
<td>itlbh[0</td>
<td>1</td>
<td>2</td>
<td>3],100003</td>
</tr>
<tr>
<td>itlbm[0</td>
<td>1</td>
<td>2</td>
<td>3],100003</td>
</tr>
<tr>
<td>eci[0</td>
<td>1</td>
<td>2</td>
<td>3],1000003</td>
</tr>
<tr>
<td>ecim[0</td>
<td>1</td>
<td>2</td>
<td>3],1000003</td>
</tr>
<tr>
<td>dc[0</td>
<td>1</td>
<td>2</td>
<td>3],1000003</td>
</tr>
<tr>
<td>dcm[0</td>
<td>1</td>
<td>2</td>
<td>3],1000003</td>
</tr>
<tr>
<td>dtlbh[0</td>
<td>1</td>
<td>2</td>
<td>3],1000003</td>
</tr>
<tr>
<td>dtlbm[0</td>
<td>1</td>
<td>2</td>
<td>3],1000003</td>
</tr>
<tr>
<td>ecd[0</td>
<td>1</td>
<td>2</td>
<td>3],10000003</td>
</tr>
<tr>
<td>ecdm[0</td>
<td>1</td>
<td>2</td>
<td>3],10000003</td>
</tr>
<tr>
<td>fpadd[0</td>
<td>1</td>
<td>2</td>
<td>3],1000003</td>
</tr>
<tr>
<td>fpmul[0</td>
<td>1</td>
<td>2</td>
<td>3],1000003</td>
</tr>
<tr>
<td>fpustall[0</td>
<td>1</td>
<td>2</td>
<td>3],1000003</td>
</tr>
<tr>
<td>memstall[0</td>
<td>1</td>
<td>2</td>
<td>3],1000003</td>
</tr>
</tbody>
</table>
Function Metrics

- **Exclusive metrics** – events inside the function itself, excluding calls to other functions
  - Use exclusive metrics to locate functions with high metric values
- **Inclusive metrics** – events inside the function and any functions it calls
  - Use inclusive metrics to determine which call sequence in your program was responsible for high metric values
- **Attributed metrics** – how much of an inclusive metric came from calls from/to another function; they attribute metrics to another function
Using the Performance Analyzer...

```bash
# collect -p lo -d exper ./ldr 8 1 /zp/space

# collect -p lo -s all -d exper ./ldr 8 1 /zp/space

# collect -p lo -s all -t 10 -o synct.er -d exper ./ldr 8 1 /zp/space
```
Run Time Checking (RTC)

- Detects memory access errors
- Detects memory leaks
- Collects data on memory use
- Works with all languages
- Works with multithreaded code
- Requires no recompiling, relinking or makefile changes