Solaris Performance Tuning
Introduction, Tools and Rules
Solaris 2.5 and UltraSPARC

Adrian Cockcroft
Staff Engineer
SMCC Server Systems Engineering
Contents

Resources
Papers, Books and Manual Sections To Read

Basic Performance Measurement Concepts
Bandwidth, throughput, latency, utilization
Collect, monitor, analyze, trend, predict

Tuning Overview
How to know what is overloaded
What measurements to ignore

Tools and Rules
virtual_adrian, ruletool and the “se” toolkit
White Papers And Books


“Sun Performance And Tuning” Book - Jan 95
Adrian Cockcroft, SunSoft Press/Prentice Hall

“Panic! UNIX Crash Dump Analysis” Book
May 95, Chris Drake and Kimberley Brown, SunSoft Press/Prentice Hall

“Managing NFS And NIS”, Hal Stern, O’Reilly
Manual Sections

Solaris 2 Security, Performance and Accounting
Generic SVR4 based sysadmin manual
Seriously lacking in both content and accuracy, some fixes for Solaris 2.4, no change for 2.5

SMCC NFS Server Performance Tuning Guide
See the SMCC Solaris 2.4 Hardware CD
Hardware AnswerBook
First release with Solaris 2.3, draft quality only, excellent rewrite for Solaris 2.4. Update due for 2.5.1
Internet Resources

Adrian’s Monthly Performance Column
www.sun.com->Whats New - Columns->Adrian

Adrian’s Monthly Sunworld Online Q&A
www.sun.com->SunWorld Online->Columns->Performance

Solaris Developer Support Center - Opcom
http://opcom.sun.ca/

Sun on the Net - Web Server Performance

Solaris PD and Free Software Resources
http://www.mbp.duke.edu/christensen/sitesG.html
Performance Measurements

*Bandwidth*
- The peak that cannot be exceeded
- Easy to work out and quote

*Throughput*
- What you really get
- Depends on the *Protocol* being used

*Latency*
- How long you have to wait for completion

*Utilization*
- Proportion of peak capacity in use
Solaris Performance Tuning - Introduction, Tools and Rules

Throughput

Response Time

Utilization 100%

Load Level or User Count

Saturation Point

Fast Response Degradation

Gradual Response Degradation

100%
Performance Management

Collect
Obtain metrics from the systems, networks and applications and store them

Monitor
Real time processing, thresholds, rules, alerts, GUI display

Analyze
Derive business oriented measures like transaction response time, and workload mix
Trend

Watch the change in metrics over time and project workload changes into the future

Predict

Model what-if scenarios to predict response time, utilization and throughput
What to Measure

Turn on system accounting!
Records who ran what when
Watch how much CPU and I/O was performed for each application
Memory use accounting fixed in Solaris 2.5
Look for workload trends over time

Enable Utilization Logging
Uncomment entries in the `sys crontab`
Collects one month `sar` history.
Disk Problems

Disk bottleneck

If more than 20-30% busy and more than 30-50 ms service time fix it by striping several disks together with Online: DiskSuite or Volume Manager.

No Problem???

If you are told the disk is no problem don’t believe it! Insist on seeing iostat -x 30.

<table>
<thead>
<tr>
<th></th>
<th>r/s</th>
<th>w/s</th>
<th>Kr/s</th>
<th>Kw/s</th>
<th>wait</th>
<th>actv</th>
<th>svc_t</th>
<th>%w</th>
<th>%b</th>
</tr>
</thead>
<tbody>
<tr>
<td>sd3</td>
<td>1.5</td>
<td>1.8</td>
<td>7.7</td>
<td>58.8</td>
<td>0.0</td>
<td>0.2</td>
<td>73.2</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>sd5</td>
<td>22.9</td>
<td>1.2</td>
<td>154.4</td>
<td>35.3</td>
<td>0.0</td>
<td>0.4</td>
<td>16.9</td>
<td>1</td>
<td>34</td>
</tr>
</tbody>
</table>
Network Problems

NFS clients

They count waiting for the server as idle, not wait for I/O. Use `nfsstat -m` to find the slow (>50ms) NFS server. Fix with NVRAM.

```
/home/username from server:/export/home3/username
Flags: vers=2,hard,intr,down,dynamic,rsize=8192,wsize=8192,retrans=5
Lookups: srtt=7 (17ms), dev=4 (20ms), cur=2 (40ms)
Reads:    srtt=16 (40ms), dev=8 (40ms), cur=6 (120ms)
Writes:   srtt=15 (37ms), dev=3 (15ms), cur=3 (60ms)
All:      srtt=15 (37ms), dev=8 (40ms), cur=5 (100ms)
```

Ethernet

Keep 10Mbit ethernets below 5% collision rate if possible. Use `netstat` to monitor. Use 100Mbit if you care about performance.
Solaris 2.5 - New NFS Features

NFS Version 3 Protocol - double speed of NFS V2
  Two-phase write commit protocol is fast
  Many detailed changes improve performance
  Multiple block read-ahead and large blocks

NFS over TCP/IP - for NFS V2 and NFS V3
  No retransmit timers (nfsstat -m is empty)
  TCP handles dropped segments

Defaults to NFS V3 and TCP/IP with 32KB blocks

Access Control Lists over the network
  Extra protocol works with NFS2, NFS3, UFS
RAM Problems

Don’t Worry About Free RAM Measure and Paging!

Ignore `vmstat free` RAM, as inactive files are cached. Don’t worry about high page in and page out (`pi` `po`) levels with `vmstat`. All filesystem I/O is done by paging.

Scan Rates and Residence Time

*Sustained* high `vmstat scan rates` (30s average `sr` above 200/sec) indicate a RAM shortage.

Keep idle page restime above 20-40 seconds

<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>r b w swap free re mf pi po fr de sr s0 s1 s2 s3 in sy cs us sy id</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59 0 0 234624 10756 97 172 0 1527 1744 0 390 0 0 0 14 507 4582 233 59 41 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**CPU Problems**

Load Average or Run Queue and Blocked Jobs

If the run queue length (\texttt{vmstat r} or \texttt{uptime load average}) is more than 3-5 times the number of CPUs you need more CPU power.

If there are as many blocked processes as runnable (\texttt{vmstat r and b}) check again for a slow disk.

System CPU Time

If system CPU time is more than user (apart from on NFS servers), perhaps tune kernel or work around mutex contention.
Performance Book Chapters

Quick Tips And Recipes
Measurement Techniques
Source Code
Applications
Disks
Networks
Processors
System Architecture
Kernel Algorithms and Memory
Source Code

A Chapter In The Book Covers
Algorithms, programming model, language,
compiler and libraries

References
Expert C Programming by Peter Vanderlinden,
SunSoft Press
High Performance Computing by Keith Dowd,
O’Reilly
SunPro Tuning Manual / You and Your Compiler
by Keith Bierman
Applications

Execution Environment
use `ps` to find which procs are hogs
trace system calls to see what is going on

Filesystem Types
UFS, NFS, tmpfs
Use Cachefs to speedup read-mostly NFS
Use `cachefsstat` in Solaris 2.5
New logging UFS option in Solaris 2.4,
DiskSuite 3.0. Instant fsck. (Don’t presto!)
Synchronous writes? use Prestoserve
Databases

Expert consultancy is worth paying for!

Configure disk for speed not capacity
9 x 1.05GB is eight times faster than 1 x 9GB

Use raw disk to reduce CPU and RAM needs
Much faster for write-intensive workloads
Use `dd | compress` into filesystem for snap backup then `ufsdump` normally

UFS set file “sticky bit” and use Prestoserve
Tune UFS write throttle (`ufs_LW` and `ufs_HW`)

Use large shared memory area (try 25% of RAM)
System Architecture

CPU Caches And Performance Issues
- Caches assume randomness
- Cache Line and Size Effects are important
- Misses are a major performance problem

UltraSPARC Systems
- Cache miss costs minimized, but still high
- UPA crossbar gives much higher throughput
- SBus throughput 3 to 6 times an SS20
- bcopy in libc uses VIS, coherent, non polluting
- Pixel operations typically see 4x VIS speedup
Solaris 2 Releases

Solaris 2.3 - NFS server tuned. X11R5.
Cachefs reduces NFS and network load

Solaris 2.4 - less RAM needed, more efficient
Supports hundreds of telnet users better
Logging UFS option, Intel x86 merge.

Solaris 2.5 - Improved SunOS 4 compatibility
Cachefsstat, accounting fixes, trace probes
Fast pipes, name service cache daemon
Less RAM, maybe 10% more speed overall
NFS V3, Storage Array, UltraSPARC support
Tweaking Solaris 2

Tweaks Set Via /etc/system
Any kernel value can be changed, but only a small number are designed to be tuned. The book explains the important tunables and the algorithm being tuned.

New And Changed Tuning Tools
sar (system activity report) lots of options
new mpstat in 2.3 quite useful, look for more than 200 mutex sleeps/sec per CPU and high system CPU time on the same line.
Kernel Algorithm Topics

Buffer sizes and tuning variables
  maxusers, DNLC, inode cache

Paging and swapping
  How it works and how to tune it

Maxusers
  Set automatically in Solaris 2.2 (up to 128)
  2.3 and later defaults to (~MB of RAM) with maximum of 1024. Limit is 2048.
  No need to change it yourself
Buffer Sizes

Directory Name Lookup Cache (DNLC) & Inodes
Set high (10,000 or more) on NFS servers only
Always make ufs_ninnode as big as ncsize
Check DNLC hit rate with vmstat -s
Watch for DNLC activity with sar -a

UFS Buffer Cache - inodes and indirect blocks
Grows dynamically in kmem - may get too big
set bufhwm=8000 (8MB) to limit size
Not a performance issue, but very big systems
  can run out of kernel memory
File Attribute Information

Name Lookup Rate
namei/s in sar -a

Directory block read rate
dirbk/s in sar -a (DNLC miss)

Inodes Reused That Had Pages
ufs_ipf% in sar -g

Inode Cache Soft Limit
ufs_ninode max entries (2.3)
max inactive entries (2.4)

References To A Cached Vnode For Each Name

Directory Name Lookup Cache
Contains ncszie Entries

Network RPC Call

Inodes Read From Buffer
igets in sar -a
lread/s in sar -b

Inodes Stored In UFS Buffer Cache Pages

Pages Read From Disk
bread/s in sar -b

Inodes Stored In UFS On Disk

Rnode Cache Contains nrnode Entries

Rnodes Read Via NFS Lookup Call From Server

NFS Information Read From Remote UFS Inode
## Vmstat Output - Paging

<table>
<thead>
<tr>
<th>Vmstat Field</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>avm or swap</td>
<td>Active virtual memory is a historical measure that is always set to zero. swap shows the free swap space in Kbytes for Solaris 2</td>
</tr>
<tr>
<td>fre</td>
<td>Free real memory in Kbytes - tends to sit at RAM/16 for 2.3</td>
</tr>
<tr>
<td>Page</td>
<td>Page faults and paging activity. The following values are averaged every five seconds, and given in units per second.</td>
</tr>
<tr>
<td>re</td>
<td>Pages reclaimed from the free list, may be due to scanning too fast.</td>
</tr>
<tr>
<td>at (SunOS 4.X)</td>
<td>Number of attaches to pages already in use by other processes, good.</td>
</tr>
<tr>
<td>mf</td>
<td>Minor faults, pages created without needing page in e.g. copy on write, zero fill on demand, or illegal page access errors.</td>
</tr>
<tr>
<td>pi</td>
<td>Kilobytes per second paged in from disk, nonzero is OK - file reads.</td>
</tr>
<tr>
<td>po</td>
<td>Kilobytes per second paged out, nonzero is OK - fsflush and writes.</td>
</tr>
<tr>
<td>fr</td>
<td>Kilobytes freed per second by pageout or processes exiting.</td>
</tr>
<tr>
<td>de</td>
<td>Artificial memory deficit set during swap outs, ignore.</td>
</tr>
<tr>
<td>sr</td>
<td>Pages scanned by pageout per-second, sustained high implies RAM shortage. Compare vs. slowscan and fastscan.</td>
</tr>
</tbody>
</table>
### Solaris Performance Tuning - Introduction, Tools and Rules

#### What To Expect

**200 User Test Run on 128 MB 4CPU SS1000**

<table>
<thead>
<tr>
<th>% vmstat 5</th>
<th>proc memory page disk faults cpu</th>
</tr>
</thead>
<tbody>
<tr>
<td>re b w swap free re mf pi po fr de sr s0 s1 s2 s3 in sy cs us sy id</td>
<td></td>
</tr>
<tr>
<td>0 0 0 330252 80708 0 2 0 0 0 0 0 0 0 0 1 18 107 113 0 1</td>
<td>99</td>
</tr>
<tr>
<td>0 0 0 330252 80708 0 0 0 0 0 0 0 0 0 0 0 14 87 78 0 0</td>
<td>99</td>
</tr>
</tbody>
</table>

... 4 0 0 320436 71448 0 349 7 0 0 0 0 2 1 0 12 144 4732 316 65 35 0 0 0 0 0 0 |

... 0 0 0 317832 68972 0 275 3 0 0 0 0 1 0 0 1 48 4920 278 64 36 0 0 0 0 0 0 |

... 50 0 0 258716 14040 0 311 2 0 0 0 0 1 0 0 1 447 4822 306 59 41 0 0 0 0 0 0 |

... 50 0 0 258684 12620 0 266 2 0 0 0 0 3 1 0 12 543 3686 341 66 34 0 0 0 0 0 0 |

... 56 0 0 251620 8352 0 321 4 1 1 0 0 1 1 0 1 461 4837 342 57 43 0 0 0 0 0 0 |

... 60 0 0 238280 5340 5 596 1 371 1200 0 4804 0 0 0 6 472 3883 313 48 52 0 0 0 0 0 0 |

... 59 0 0 234624 10756 97 172 0 1527 1744 0 390 0 0 0 14 507 4582 233 59 41 0 0 0 0 0 0 |

... 60 0 0 233668 10660 9 297 2 0 0 0 0 4 2 0 12 539 5223 272 57 43 0 0 0 0 0 0 |

... 61 0 0 232232 8564 2 225 0 75 86 0 87 0 0 0 2 441 3697 217 71 29 0 0 0 0 0 0 |

... 62 0 0 231216 8248 2 334 11 500 547 0 258 1 0 0 7 484 5482 292 52 48 0 0 0 0 0 0 |

... 91 0 0 196868 7836 0 227 8 511 852 0 278 1 7 0 5 504 5278 298 50 50 0 0 0 0 0 0 |

... 91 1 0 196368 8184 1 158 3 1634 2095 0 652 0 37 0 5 674 3930 325 50 50 0 0 0 0 0 0 |

... 92 0 0 200932 14024 0 293 85 496 579 0 42 0 17 0 21 654 4416 435 47 53 0 0 0 0 0 0 |

... 93 0 0 208584 21768 1 329 9 0 0 0 0 0 0 0 3 459 3971 315 62 38 0 0 0 0 0 0 |

... 92 1 0 208388 20964 0 328 12 0 0 0 0 0 3 3 0 14 564 5079 376 53 47 0 0 0 0 0 0 |

... 189 0 0 41136 8816 3 99 32 243 276 0 168 1 1 0 9 500 3804 235 67 33 0 0 0 0 0 0 |

... 190 0 0 40328 8380 6 65 76 0 0 0 0 3 2 0 19 541 3666 178 71 29 0 0 0 0 0 0

---

Idle, lots of free RAM

Users start running

RAM/16 = 8MB page reclaim kicks in hard then stops & stabilizes

process exit frees RAM

stable state during run

Sun Microsystems Computer Corporation.

Adrian Cockroft - December 1995
### Solaris Performance Tuning - Introduction, Tools and Rules

<table>
<thead>
<tr>
<th>procs</th>
<th>memory</th>
<th>page</th>
<th>disk</th>
<th>faults</th>
<th>cpu</th>
<th>swap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>free</td>
</tr>
<tr>
<td>190</td>
<td>0</td>
<td>0</td>
<td>40052</td>
<td>7976</td>
<td>1</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>102</td>
<td>58</td>
<td>65</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>15</td>
<td>457</td>
<td>3415</td>
<td>158</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>28</td>
<td>0</td>
<td>158</td>
<td>128</td>
<td>0</td>
</tr>
</tbody>
</table>

This system has just enough RAM but needs more CPU power to improve response times by reducing the run queue length.

---

stable state during run

users logout back to idle

virtual mem & physical mem

r - jobs waiting for CPU
b - jobs waiting for disk
w - jobs waiting for RAM
MP Control and Monitoring

Processor control and information
psrinfo/psradm - enable/disable CPUs
/usr/platform/sun4d/sbin/prtdiag - config
prtconf/devinfo - show device configuration
sysdef - show software configuration
dispadmin - modify scheduler
mpstat - watch per-cpu interrupts and smtx

Dispadmin tweaks - workload dependent
Increased scheduler time quanta helps
database workloads
Performance Summary

Subjects Covered

Measurement suggestions
Concepts and terminology
Basic rules and thresholds for disk, network, memory and CPU problems
Highlights of the white paper and book
Kernel Tuning
Tools and Rules

Problem - provided tools are too limited
sar doesn’t collect network data
netstat doesn’t show collision % and rates
iostat doesn’t translate sd43 into c1t2d4 etc.
need nasty awk scripts to pick out data

Solution - a flexible extensible toolkit
Dedicated interpreted C dialect
Replaces nasty awk scripts with clean code
Built for this job, small and efficient
Freely available via the internet
SymbEL - A Free Toolkit

Written by Rich Pettit who saw the need
Contributions and encouragement by me

FTP from opcom.sun.ca in /pub/binaries/se2.4 or
http://www.sun.com/950901/columns/adrian/se2.4.html

Support for SPARC Solaris 2.3, 2.4, 2.5beta, 2.4x86

Easy installation based on packages

RICHPse  The SymbEL Interpreter
RICHPsex  The SE eXtensions Package
ANCrules  Adrian's Rules & Tools

# pkgadd -d . RICHPse RICHPsex ANCrules
Introduction to SE

SE is an interpreted dialect of C
Not a new language to learn from scratch
Standard /usr/ccs/bin/cpp is used
Main omissions - pointers and goto
Main additions - classes and “string” type
Dynamic linking to all existing C libraries
Built-in classes access kernel data
Supplied class code hides details
Example scripts improve basic utilities
Example rule based monitors
Example - iostat.se

```c
#!/opt/RICHPse/bin/se

#include <stdio.se>
#include <stdlib.se>
#include <unistd.se>
#include <string.se>
#include <kstat.se>
#include <sysdepend.se>
#include <p_iostat_class.se>
#include <dirent.se>
#include <inst_to_path_class.se>

#define SAMPLE_INTERVAL  5

main(int argc, string argv[2])
{
    p_iostat p_iostat$disk;
    p_iostat tmp_disk;
    int i;
    int interval = SAMPLE_INTERVAL;
    int ndisks;

    switch(argc) {
    case 1:
        break;
    case 2:
```
Solaris Performance Tuning - Introduction, Tools and Rules

```c
interval = atoi(argv[1]);
break;
default:
    printf("use: %s [interval]\n", argv[0]);
    exit(1);
}
ndisks = p_iostat$disk.disk_count;
for(;;) {
    sleep(interval);
    printf("extended disk statistics\n");
    printf("disk r/s w/s Kr/s Kw/s wait actv svc_t %w %b\n");
    for(i=0; i<ndisks; i++) {
        p_iostat$disk.number$ = i;
        tmp_disk = p_iostat$disk;
        printf("%-8.8s %4.1f %4.1f %6.1f %6.1f %4.1f %4.1f %6.1f %3.0f %3.0f\n",
            tmp_disk.name$,
            tmp_disk.reads, tmp_disk.writes,
            tmp_disk.kreads, tmp_disk.kwrites,
            tmp_disk.avg_wait, tmp_disk.avg_run,
            tmp_disk.service,
            tmp_disk.wait_percent, tmp_disk.run_percent);
    }
}
```
Basic Monitor Scripts

Based on standard command output
Simple threshold rules added
Only print anything if they detect a problem

iomonitor.se - based on iostat
  translates sd43 to c2t4d2 etc.
  lists disks over 20% busy and 50ms svc_t

vmmonitor.se - based on vmstat
  looks for swap space and RAM shortage

netmonitor.se - based on extended netstat
  looks for over 5% collisions
Rule Building

Complex rules defined in my tuning book
Appendix A covered Disk, Net, NFS, CPU etc.
Written rules based on experience
Basically the same rules covered by this talk

Rules coded as classes in SE
Pure rules implementation of Appendix A
One bugfix and one minor refinement
Live rules read system data and use pure
Reusable code #included as header files
Trivial to define and use in a script
Code To Use A Live Rule

```c
lr_disk_t lr_disk$dr;
lr_disk_t tmp_dr;
/* use the live disk rule */
tmp_dr = lr_disk$dr;
if ( tmp_dr.state > ST_GREEN) {
    printf("The disks are in the %s state: %s\n",
            state_string(tmp_dr.state), tmp_dr.action);
}
```

All rules are defined as classes containing state code and action string input data for pure rules only derived output data measures to report code that runs when the class is read Data changes each time you read it
The Basic Rule-set

Disk Rule - pr_disk_t and lr_disk_t
- Provides overall and per-disk states
- Live rule handles floppy and cdrom cases
- Looks for slow disks and unbalanced usage

Network Rule - pr_enet_t and lr_net_t
- Pure rule for ethernets only
- Live rule for all network types
- Looks for slow nets and unbalanced usage

Client RPC Rule - pr_rpcclient_t and lr_rpcclient_t
- Looks for bad nets and slow NFS servers
The Basic Ruleset (cont)

Swap Rule - pr_swapspace_t and lr_swapspace_t
Looks for lack of swap space

RAM Rule - pr_ram_t and lr_ram_t
Modified Appendix A rule scales better
Looks for short page residence times

Kernel Memory Rule - pr_kmem_t and lr_kmem_t
Looks for allocation failures

CPU Power Rule - pr_cpu_t and lr_cpu_t
Scales on MP systems
Looks for long run queue delays
The Basic Ruleset (cont)

Mutex Rule - pr_mutex_t and lr_mutex_t
Looks for kernel lock contention

DNLC Rule - pr_dnlc_t and lr_dnlc_t
Poor Directory Name Lookup Cache hitrate

Inode Cache Rule - pr_inode_t and lr_inode_t
Poor inode cache hitrate

More rules?????? - easy to add your own
Copy existing rule classes
Test framework provided
Send them to me for the next release
Example Monitors

pure_test.se and live_test.se
Text based programs for building and running rules under construction

monlog.se - system log monitor
Looks for state changes every 120s
Writes changes only, via syslog

mon_cm.se - calendar monitor
Looks for amber/red/black states
Writes calendar entry every 15 minutes
Multibrowse root@hostname with cm
Virtual_adrian.se

Personalized tuning and monitoring script
Does what I’d do if I was there
Runs as root for full functionality
Somewhat aggressive and annoying...

Static tuning check - runs once at start-up
Knows about OS releases
Checks *and tunes* kernel values if needed

Basic rules and extras - 30 second interval
Looks for slow NFS client mount points
Looks for fsflush taking too much CPU time
Ruletool.se

Graphical View of Rules
Based on the Motif GUI extension library
Uses a lot of RAM but little CPU time

Popup display for more detail - just click on button
All rule threshold values shown
Metric values used as rule inputs visible
Display formats like vmstat, iostat, netstat

Icons for the authors
Pop-ups provide “biographies” :-}
### Ruletool Sample Display

<table>
<thead>
<tr>
<th>cpun</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>Monday June 5, 1995, 11:59:10 AM</td>
</tr>
<tr>
<td>Disk</td>
<td>Move load from busy disks to idle disks</td>
</tr>
<tr>
<td>c0t2d0</td>
<td>c0t3d0</td>
</tr>
<tr>
<td>Network</td>
<td>No problem</td>
</tr>
<tr>
<td>nfs</td>
<td></td>
</tr>
<tr>
<td>NFS Client</td>
<td>No client NFS/RPC activity</td>
</tr>
<tr>
<td>Swap Space</td>
<td>No problem</td>
</tr>
<tr>
<td>RAM Demand</td>
<td>The system is getting short on RAM, perhaps add some more</td>
</tr>
<tr>
<td>Kernel memory</td>
<td>No problem</td>
</tr>
<tr>
<td>CPU Power</td>
<td>There is more CPU power configured than you need right now</td>
</tr>
<tr>
<td>Mutex</td>
<td>No problem</td>
</tr>
<tr>
<td>Directory cache</td>
<td>No activity</td>
</tr>
<tr>
<td>Inode cache</td>
<td>No activity</td>
</tr>
<tr>
<td>Interval</td>
<td></td>
</tr>
</tbody>
</table>

30

RuleTool by Richard Pettit

based on tuning rules by Adrian Cockcroft
Summary

Rules Capture
Existing tools too hard to use
Powerful new toolkit implements rules

Performance Tools
Example tools useful and powerful
Interpreted scripts are easy to extend

Freely Available for Solaris 2
Not a supported product
Documentation of data sources, low level processing and behavior
Conclusion

Sun Performance Tuning Book and SE Scripts
Read these for more details

A Little Knowledge is a Dangerous Thing...
Test Tweaks in Isolation
Measure before and after
Only keep changes that make a difference
Back out changes if problems occur, then call SunService if the problem persists.

Feedback - Let Me Know What Does/Doesn’t Work!
Adrian.Cockcroft@sun.com