Sun Performance and Tuning - Java and the Internet

Adrian.Cockcroft@sun.com

Speaker's Notes

Introduction - Adrian Cockcroft - Performance Guru

BSc Applied Physics and Electronics - City Univ London
6 years realtime software engineer R&D consultancy
Sun user and System Administrator since 1984
Joined Sun UK as Systems Engineer 1988
Wrote Sun Performance Tuning Paper for UK SUG 1991
Moved to Sun USA 1993, formed Enterprise Engineering
Wrote Sun Performance and Tuning Book 1994
Put SE toolkit & "virtual_adrian" on www.sun.com 1995
SunWorld Online monthly columnist since 1995
Wrote 2nd edition of book 1997, published April 1998

Speaker's Notes

Introduction

One definition of Guru in the Oxford English Dictionary is "Respected Teacher". Although it is a slightly pretentious title to give myself, it is the most succinct way to describe my role at Sun.

Enterprise Engineering is a group that was formed when we outgrew the definition of Technical Marketing and became part of Sun's Engineering organisation. We are an engineering group with a broad rather than narrow focus and no product development responsibilities. Our job is to foster communications between the field, marketing and engineering, stay a few steps ahead of everyone else, try out products before they are released, work strategically and look at "The Big Picture". We are working to publish a "best practices" documentation series called Sun BluePrints. Our parent organisation, the Strategic Systems Technology Group includes Suns large scale Enterprise Test Center, and is part of the Sun Computer Systems, Enterprise Desktop and Server Systems business.

Contents

This presentation is an overview of the 2nd edition

Read the details in the 600+ page book itself

Speaker's Notes

Contents

Its impossible to cover a 600 page book in any detail in a few hours

This presentation is intended to make sure you know what is in the book, so you can look up the key information and you are alerted to the neat tricks and little known commands that you need.

Quick Tips and Recipes

Common Problems

The system will usually have a disk bottleneck
- Look for unbalanced, busy or slow disks: `iostat -xPnce 30`

Poor NFS response times may be hard to see
- Use `iostat` on a Solaris 2.6 client

Avoid the common `vmstat` misconceptions
- Ignore "free" memory

Don't panic when you see page-ins and page-outs in `vmstat`
- Normal filesystem activity uses paging

Use page scanner "sr" activity as your RAM shortage indicator

Look for a long run queue (`vmstat` procs r) - and add CPUs

Look for processes blocked waiting for disk I/O (`vmstat` procs b)

Check for CPU system time dominating user time

Watch out for processes that hog the CPU

Speaker's Notes

Common Problems

Remember "expense" iostat -xPnce works in Solaris 2.6, use iostat -xc in older releases

Use 30 second interval to avoid spikes in load. Watch svc_t (or asvc_t) the response time.

Look for disks over 5% busy that have response times of more than 30ms as a problem.

If you have NVRAM and hardware RAID, look for times of more than 10ms as a problem.

Ignore large response times on idle disks that have filesystems - its not a problem and the cause is the fsflush process as explained later

NFS client mounts appear in iostat on Solaris 2.6 as if they were disks

Cold Start Procedure - How to ask for help

Information you need to provide

What is the business function of the system?
Who and where are the users?
Who says there is a problem, and what is slow?
What is the system configuration? CPU/RAM/Disk/Net/OS/Patches
What application software is in use?
What are the busy processes on the system doing?

- use `top`, `pea.se` or `/usr/ucb/ps uax | head` to find out

Report CPU and disk utilization levels, `iostat` or `siostat.se 30`
What is making the disks busy?
What is the network name service configuration?

- Slow systems may be waiting for a name server response

How much network activity is there? Use `nx.se` or `netstat -i 30`
Is there enough memory? Check the scan rate with `vmstat 30`
What changed recently and what is on the way?

Speaker's Notes

Cold Start Procedure

More details in the book, also on the web at
http://www.sun.com/sunworldonline/swol-11-1997/swol-11-perf.html

Recipes

Guidelines for setting up and tuning systems
Recipes for desktop, workgroup server, databases
Key new recommendation - mixed workloads

- Systems are now more powerful than a workgroup needs
- So use a single Solaris workgroup server to do everything
- NFS, SMB, email, web home pages, proxy cache, NIS/LDAP/DNS
- Replace several NT systems or older dedicated servers
- This works well with Solaris, but NT can't handle mixes as well
- High dependency on a single system - must be reliable
- Need to configure and tune without needing reboots
- Greatly reduced admin complexity and cost
- Only at the high end use HA clustered pairs of systems
- HA administration procedures must be very rigorous

Speaker's Notes

Recipes

Detailed workgroup server recipes have also been developed as whitepapers.

Performance Management

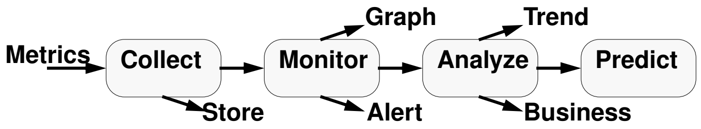
Analogy with financial management

- Use planning, budget allocation and capital & expense controls
- “Capital” - fixed resources like RAM and disk space usage
- “Expenses” - consumables like CPU time, network bandwidth
- Plan and agree resource needs for your workloads in advance
- Monitor resources using system accounting and perf tools
- Control resources on a per workload basis

Speaker’s Notes

Performance Management Sequence

Collect and store configuration and performance
Monitor, alert and display in real time
Analyze into business workload oriented terms
Establish norms and project trends
Predict the effect of workload changes
Predict the effect of configuration changes



Speaker’s Notes

Performance Management Tools

Traditional Unix tools

- Incomplete, inconsistent but common and lightweight
- `sar`, `vmstat`, `iostat`, `netstat` etc.

System Accounting

- Who ran what when, how much resource was used
- Also need to collect long lived process data with `ps`

Sun Symon - Server System Monitor

- Symon 1.5 supplied free for all SPARC servers
- In depth hardware configuration display
- Single system performance monitoring and RAS alerts
- A good demo, but too limited for large scale use
- A placeholder for Symon 2.0, *the real thing....*

Speaker's Notes

Some Commercial Products

Products	Collect	Monitor	Trend	Forecast
AIM SharpShooter		No longer sold		
Amdahl OpenTune/ AUMA	Yes	Yes	No	No
BGS Best/1 for UNIX	Yes	Yes	Yes	Yes
BMC Patrol	Yes	Yes	No	No
Resolute Software RAPS	Yes	Yes	Yes	No
Compuware Ecotools	Yes	Yes	No	No
Datametrics	Yes	Yes	No	No
HP GlancePlus	No	Yes	No	No
HP MeasureWare	Yes	Yes	Yes	No
Landmark PerformanceWorks	Yes	Yes	Yes	Yes
Metron Athene/UNIX	Yes	No	Yes	Yes
SAS IT Service View (CPE)	No	No	Yes	Yes
SES Workbench/Strategizer	No	No	Yes	Yes
Sun Symon 1.5	No	Yes	No	No
TeamQuest Baseline and Model	Yes	Yes	Yes	Yes

Speaker's Notes

This table lists products that I think are interesting. I have omitted network management tools such as SunNet Manager and HP OpenView as they serve a different set of needs with only a minor overlap into performance management. The categorization is based upon my own opinion of their functionality, not the vendors' claims. For example, anyone who can draw a graph over time claims he can do trending, but I want to see some support for fitting a curve to a trend and hitting a threshold in the future or evaluating alternative future growth scenarios

Products that do everything

BGS Best/1 and BMC Patrol Merger

- BMC recently aquired BGS
- Best/1 is the market leader for modelling and prediction
- Patrol is a comprehensive application management monitor
- Good synergy, now they need to do a careful product merger
- Expensive, but the most powerful tools available
- Patrol is easy to setup and use
- Best/1 is complex, quite hard to use
- Both are normally sold with a lot of training and consultancy
- Once setup, they are highly automated “hands-off” tools

Landmark and Metron

- Metron’s Prediction tool has been incorporated by Landmark

Teamquest Baseline and Model

- Less complex and less expensive alternative

Speaker’s Notes

I have used Patrol a little to monitor a web server, and it is easy to install and use, pre-configured knowledge modules know how to manage common applications. Consulting groups such as SunPS have developed additional KMs that monitor Sun products.

I have been using Best/1 for several years, and helped BGS port it to Solaris. Its the most comprehensive performance management tool I have seen, but it comes in a raw state, needs a lot of complex setup, and needs a better GUI.

This is the high end of the market, and a large investment of time and money is needed. The end result should be a very well managed and monitored system that is highly automated and has low running costs and good service levels for the end users.

Landmark have a good GUI, and are working with Metron to add prediction capability. This product is strong on integrating Unix and NT based performance management from both sides.

Teamquest are a good engineering company but have little market prescence. They have a good product at a lower price point than its competition. They also support data collection on supercomputers such as Cray and Convex, and would be the best choice for an HPC oriented environment. They need better sales and marketing support.

Collecting Measurements

Workload types

- Fixed workloads - you must choose the right combination
- Random workloads - you must analyze the data carefully
- Managed workloads - use an RTE to control a mixture

Statistical Design of Experiments

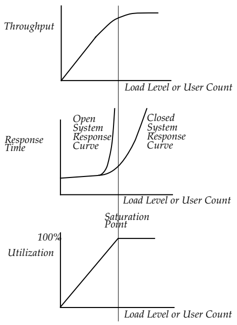
- Choose the factors you want to vary (user count, CPU type etc.)
- Choose the number of levels for each factor
- Six factors with four levels each needs 4^6 = 4096 tests
- Start off with two levels each, still needs 2^6 = 64 tests
- Use statistical methods to start with only 7 tests
- Analysis tells you which are significant and error estimates
- Eliminate second order factors that have little effect
- Add more tests to reduce error bounds

Speaker’s Notes

The best way to approach a large-scale set of tests is to perform an initial small set of tests with just two levels for each factor. Raj Jain, in *The Art of Computer Systems Performance Analysis*, describes a sophisticated statistical technique that drastically reduces the number of tests required to work out the factors that make a significant difference and those that can be ignored. To fully measure every combination of six different factors with four levels for each factor would take 4^6 = 4096 separate measurements. Reducing to two levels for each factor brings this number down to 2^6 = 64 separate measurements. Using the statistical methods to analyze the results, you carefully pick certain combinations of levels to measure and do the analysis on only seven measurements to find out which factors can be ignored for a more complete set of tests.

One extra benefit of doing a small preliminary test is that you start analyzing the results sooner than you would if you did all the measurements in one session. Since you will often find some bug in your measurement methods, you have a chance to fix it before you spend a lot longer taking bad measurements. It is common for a large proportion of the measured data to be useless for some reason, even in the best-designed experiments.

Measurement Terms and Definitions



Bandwidth - gross work per unit time - unattainable
Throughput - net work per unit time
Utilization - busy time relative to elapsed time
Peak throughput - throughput at 100% utilization
Queue length - number of requests waiting
Response time - time to complete a unit of work including waiting
Service time - time to process a unit of work after waiting

$$\text{Mean queue length} = \text{Mean throughput} \times \text{Mean response time}$$

$$\text{Mean service time} = \frac{\text{Mean utilization}}{\text{Mean throughput}}$$

Estimate 95th percentile response time as three times mean response time

Speaker's Notes

Internet Servers - Introduction to TCP

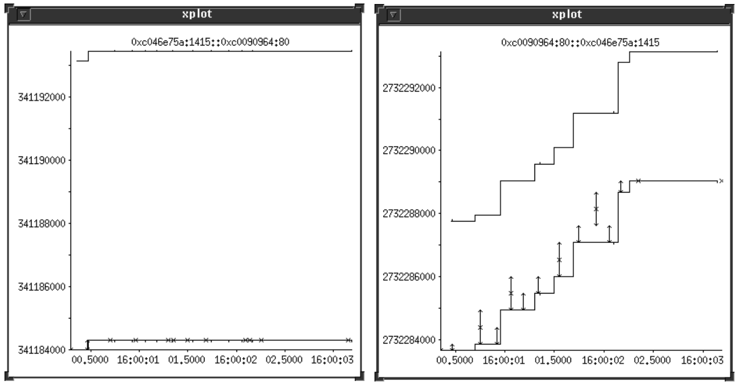
TCP Connections are expensive

- TCP is optimized for reliable data on long lived connections
- Making a connection uses a lot more CPU than moving data
- Connection setup handshake involves several round trip delays
- Pending connections cause "listen queue" issues
- Each new connection goes through a "slow start" ramp up
- TCP windows can limit high latency high speed links
- Lost or delayed data causes timeouts and retransmissions
- Each open connection consumes memory
- HTTP persistent connections carry several ops on one connect

Speaker's Notes

TCP in Pictures

Medium sized HTTP request and response with retransmit congestion



Speaker's Notes

Printouts obtained from the packet shell - see http://playground.sun.com/psh

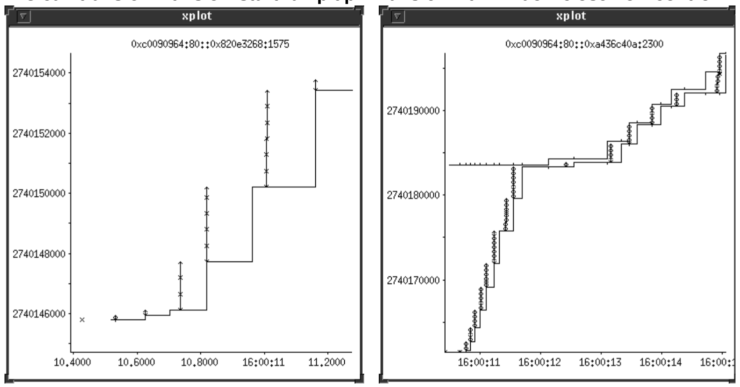
Data mostly taken from a snoop file on www.sun.com several years ago

Across the top:

Medium sized HTTP request and response with retransmit congestion

TCP in Pictures

Clean transfer with slow start ramp up - Transfer with window close flow control



Speaker's Notes

Printouts obtained from the packet shell - see http://playground.sun.com/psh

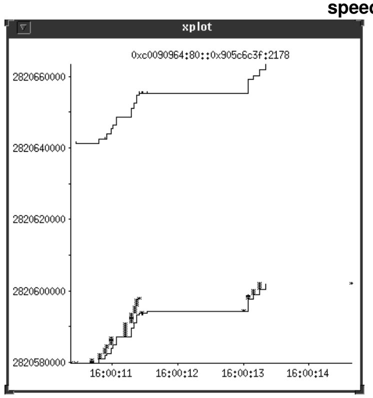
Data mostly taken from a snoop file on www.sun.com several years ago

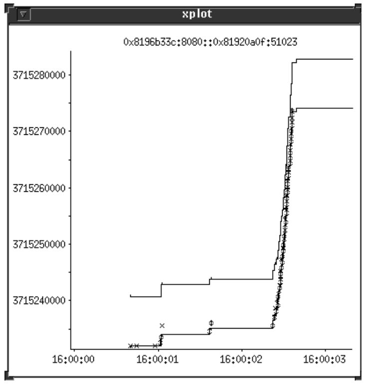
Clean transfer showing slow start ramp up in action - Large transfer with window close flow control

TCP in Pictures

Problem with increased minimum retransmit timeout

Persistent connections showing two small transfers and a large image over high speed net





Speaker's Notes

Printouts obtained from the packet shell - see http://playground.sun.com/psh

Data mostly taken from a snoop file on www.sun.com several years ago

Problem with increased minimum retransmit timeout

Persistent connections showing two small transfers and a large image over high speed net

Tuning TCP

Watch TCP counters

- **netstat -s shows a snapshot**
- **nx.se shows rates in a more convenient form**

Change TCP Tuning Online with ndd (no reboot!)

tcp_close_wait_interval (actually tcp_time_wait_interval)
- **default 240000 ms. Reduced to 60000 for SPECweb96 runs**

tcp_conn_req_max - obsolete per-process listen queue limit

tcp_conn_req_max_q0 - incomplete listen connection limit
- **default 1024 - no need to tune unless tcpListenDropQ0 seen**

tcp_conn_req_max_q - pending completed connection limit
- **default 128 - no need to tune it unless tcpListenDrop seen**

tcp_slow_start_initial - number of initial packets to send
- **default 1 - set to 2 to be the same as other vendors**

tcp_xmit_hiwat and tcp_recv_hiwat - window size
- **default 8192 - try setting to 32768 if using fast WAN links**

Speaker's Notes

Tuning TPC online with no reboot is a *big* benefit of Solaris that few if any other vendors can match. Rebooting to make a change destroys your ability to make changes in a busy system and makes it very hard to measure the effect of a change. Running nx.se in one window, tweaking ndd in another and looking for immediate effects is a very powerful technique.

TCP slowstart standard says that one packet should be acked before more are sent. Most vendors implement a bug that causes two packets to be sent, this makes the initial ramp up a bit quicker. There is an issue due to another implementation feature in W95 and NT that can cause a startup delay when used in a low latency LAN environment with slowstart of 1. By setting to 2, Solaris sends the same as other platforms and avoids the W95 and NT bug. The standard is being changed to allow 2 and Sun will ship with a default of 2 in future OS releases.

Window size depends on the bandwidth x delay product. 1 MB/s at 5ms = 5000 bytes which is less than the 8192 default and is typical of LAN based transfers. Modems at 3KB/s at 1 second = 3000 bytes so they are OK as well. Wide area high speed connections, especially over satellite links can need very large windows. Limit in 2.5.1 is 64KB, window scale of up to 2GB is supported in 2.6 and later.

TCP Rules and the SE Toolkit

Let SE watch your TCP stack for you

- ignore everything if TCP throughput is less than 2 KB/s
- warn if retransmit rate is over 15%, problem if over 25%
- warn if listen drops are seen, problem if over 0.5/s
- warn of SYN service denial attack if tcpHalfOpenDrop over 2/s
- warn if connection refused - sending reset packets at 0.5/s
- possible port scanner attack if resets at over 2/s
- warn if attempted outgoing connection fails over 2/s
- warn if incoming duplicate packets over 15%
- problem if duplicates at 25% - remote is retransmitting at you

Tools that use the TCP rule

- virtual_adrian.se - text log of problems seen
- zoom.se - GUI front end to rules
- percollator.se - long term web server data collector

Speaker's Notes

You can do this yourself or write a script to do it, but the SE toolkit includes example code that implements these rules and uses them in its example performance monitors.

SE is not a supported Sun product, its an experimental development environment

Commercial tools vendors are using its example rules to improve their products

Simple HTTP Web Server Sizing

Network bandwidth is the limiting factor for SPECweb96

Network Connection	Bandwidth (bits/s)	Peak httpops/s	Server
28800baud modem	28.8K	<0.25	Any
ISDN 1 BRI	64K	<0.5	Any
ISDN 2 BRI	128K	<1	Any
T1	1.5M	10	Any
Ethernet	10M	60	Any
T3	45M	300	Ultra 1
FastEthernet	100M	500	Ultra 2
Full Duplex Fast Enet	100+100M	700	Ultra 2
FDDI	100M	600	Ultra 2
ATM (OC3)	155+155M	1000	Ultra 2
ATM (OC12)	622+622M	4000	E450 - E3000

Speaker's Notes

Disk Capacity and HTTP Access Patterns

Cache all content in RAM for peak speeds

- SPECweb96 results are posted with 512MB to 2GB of RAM
- several 100Mbit or 155Mbit networks to clients
- entire data set cached to avoid disk latency
- very unrealistic for most sites

Assume uncached random read accesses instead

- Each http operation matches a random disk lookup and read
- Each individual transfer is slow, so no fast sequential reads
- Approximate by taking 8KB random read rate per disk
- Typical disks sustain ~100 ops/sec flat out
- Sizing guideline: allow 50 httpops/sec per disk (~400KB/s)
- Stripe disks together until you have enough to saturate your net
- RAID5 can work well for read-intensive workloads like this

Speaker's Notes

RAM usage - How many server threads?

Short connection duration over LANs

- round trip time is typically under 1ms
- benchmarks or LAN connections take under 50ms
- even at high connection rates, few are concurrent
- small memory needed for server threads
- e.g. 100 ops/sec @ 50ms/op = 5 threads required
- benchmarks use large memory as file cache

Long connection duration via WAN or Modem

- round trip time typically a second or more
- Internet or WAN transfers take several (or many) seconds
- Big transfers hold a connection for a very long time
- even at low connection rates many are concurrent
- large memory needed for server threads
- e.g. 100 ops/sec @ 5s/op = 500 threads required
- file access is random and slow, file cache needed is small

Speaker's Notes

RAM use - how many instances

Bulk of memory usage can be per-instance.

Benchmarks like SPECweb96 and webstone run over 100Mbit LANs, and bear little relation to the kind of workload seen on the Internet. Each individual transfer sends data at several MB/sec and connection setup and shutdown take milliseconds. Disk transfer times of 20ms or so have a big impact on performance so as much data as possible is cached in RAM.

Round trip times on the Internet are often measured in seconds and vary by seconds from one packet to the next, each connection includes several round trips during setup and shutdown, as well as the time taken to transfer the data (often at a peak rate of a few 100bytes/s to a few KB/s), and the time taken to retransmit dropped segments.

To serve a long latency load, you must be able to dedicate a process or a thread to the connection for a long period of time, so you need a lot of them, and the memory soon adds up.

Since each file transfer runs so slowly, the time taken to get the data from disk (20ms or so) is lost in the propagation delay (several seconds), and it is not worth having a big file cache.

Measurements on www.sun.com indicate that at 15 httpops/sec there were 400 or more open connections. Many of these are due to a buildup of very long-lived transfers of large files and also the use of HTTP 1.1 keepalives causes connections to hang around for a long time waiting for another request from a browser.

Proxy Web Caches

Concentrated Workload due to “funneling”

- Every access from every user goes through the proxy
- High peak loads, single point of failure
- Probably the busiest parts of the web server infrastructure

Much higher load than normal content server

- Read request, search disk cache, issue conditional request
- Likely to need many disks and fast CPU

Adds significant latency for users

- Extra DNS lookups may also increase latency

Three common implementations

- CERN is original and slowest - upgrade!
- Netscape Proxy 2.5 good for small/medium installations
- Harvest/Squid Cache needed for large/complex installations

Speaker’s Notes

Caching Proxies

Normal a population of users all access sites at random, so the load is many to many and is distributed over all systems, with a shared network as the only common bottleneck and failure point.

If all the users go through a caching proxy, they are funneled through a single system for every access, that system sees a high load and is a single point of failure for the users. It must be situated on a high bandwidth backbone, and is likely to have the highest load of any system on a corporate intranet.

Each operation is much more intensive than a normal web server operation. A large cache search is CPU intensive, the outgoing request for new data is common as cache miss rates are high (60-80% misses are common), then the data needs to be saved to disk as well as sent to the client.

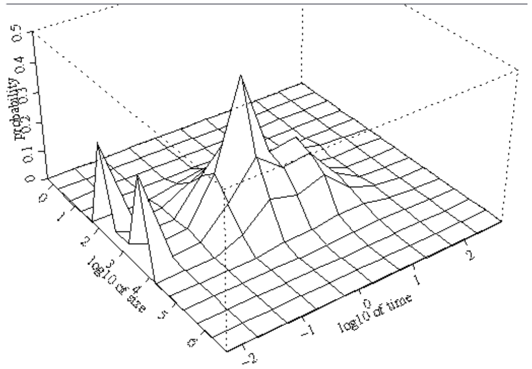
All this slows down the user’s response, but decreases network loads and provides for better security and “screening” of undesirable data. One problem that can occur with multiple levels of cache is that the local cache needs to decide whether to make a direct call to a server, or pass on to another proxy cache for an external domain. The cache rule needs to know the domain name, so a DNS request is issued to turn the IP address back into the “www.somewhere.com” form. This DNS lookup can itself take a long time to complete, and it may be repeated for every level in a cache hierarchy.

The CERN cache is based on the original 1994 CERN server, and forks for every request. If you are still using it, you should upgrade if it is supporting more than trivial levels of use.

The Netscape 2.5 Proxy Cache is a high performance single system cache that seems to be in common use. Sun’s worldwide intranet uses it to provide a multi-level cache, and it is being pushed to its limit. I’d recommend using Netscape in small to medium setups, with one or two levels of cache.

The Harvest cache and the related Squid cache used by Sun’s Netra Proxy Cache use an Inter-cache Protocol (ICP) to cluster together caches so that they can share their cache contents and move data between levels in a multi-level cache more efficiently (and avoiding the extra DNS lookups). Its more complex to setup, but much more powerful and efficient in large installations.

HTTP Response Time Density Distribution



Speaker’s Notes

Java Application Servers

How does an applet read/write files?

- By using a “helper process” on the server
- Authenticated server processes use an applet GUI
- Work can be distributed as appropriate
- Very different to dumb X-terminals

Compare Java with more familiar paradigms

- Relative “size” of client and server
- Network bandwidth requirements
- Ability to tolerate network latency
- Is the network in the interactive loop?
- Java thin client has high latency tolerance and local interactive response

Speaker’s Notes

Distributed Java Applications

A Java applet cannot read or write files at the desktop, and it can only communicate with the server that it was downloaded from. So how can a Java based word processor save your files? And how can a Java based SNA3270 terminal emulator connect to a Mainframe? The answer is that a server-side “helper” process is used. This process must be authenticated, so a user name and password must be provided. An NC asks for these at login time, so they can be provided to a local server automatically. A PC running a browser, or an NC accessing a server that is outside its administrative domain will need to provide the user name and password at the time the application is started.

Once the helper process is running on the server, it has all the normal capabilities of any program, and it can be written in any language. If the helper process is itself written as a stand-alone Java application (as opposed to an applet) then it will be portable to any kind of server that can run Java.

The helper process may provide extra compute power for an applet. It may contain the core of an application that has been ported to Java by merely rewriting the GUI as an applet. It may hold large amounts of data, using the virtual memory capabilities of the server to run applications that are too large to fit in the limited memory of a PC or NC. It may open network connections to other systems using different protocols. It may provide an SQL gateway for database access. All these uses put a load on the Java application server, and on the network, and need to be evaluated as part of the sizing project

In operation, the Java client will access information and Java applications in a location independent manner from systems throughout the intranet and Internet. In contrast, most other computing architectures tie the client system to a local server and are optimized for continuous low latency communications between the client and the server.

Emulating Java Clients

Protocol is application dependent

- Existing Emulation packages can’t handle this
- Much more than HTTP
- Capture at the Java event stream level
- Control HotJava browser with JavaStar
- High emulator load but reliable

HotJava web browser is part of JavaOS

- Also available to run standalone under Unix JVM
- HotJava on a Unix client is similar to NC behavior
- Use to emulate lots of NC users at once
- Using E6000 to drive emulation, 100s of NC’s

Speaker’s Notes

Emulating NC Based Users

In the past, tools have been developed that emulate dumb terminal based users by opening a telnet session from a process that simulates the user, to a server. More recently, these tools have been extended to capture client-server traffic at the SQL level between PCs and database servers. They have also attempted to capture and replay X-terminal traffic (without much success) and the very latest versions can capture and replay simple HTML traffic from web browsers to web servers. We investigated tools such as Pure Performix Empower, Performance Awareness PreVue, and Mercury WebRunner. None of them could capture and replay a Java applet that was connected to a backend process using an arbitrary protocol.

The breakthrough came when we discovered a Java based testing tool developed by a spin-off from Sun’s research labs, called SunTest JavaStar (http://www.suntest.com). This tool can capture and replay any pure Java application. It is itself written in Java, and it captures the users timed actions as a script that is also in Java. We found that it could be used to capture and replay complete sessions by controlling the HotJava browser (which is in itself a Java application) running on a Unix workstation. The sessions included browsing through web pages, starting and operating applets, and using complete Java applications that include a backend server component. Suntest now have a load testing capability as a product.

The HotJava browser is a pure Java application. A customized version also forms the core of the user interface to the JavaOS operating system used by NCs such as Sun’s JavaStation, so running a copy under Unix is as close as we can get to emulation of an NC. JavaStar is oriented to the problems of automated testing of Java programs by developers. One feature they added to support this is a command line interface that allows the testing to be run from a Unix makefile or script as part of an automated build and regression test setup. This includes an “invisible” option, which suppresses all graphical display. The test results are logged, and parameters can be passed into the script. By collecting and parameterizing a series of user sequences we can build a workload mix that can be replicated to any number of users, and does not need any graphical display.

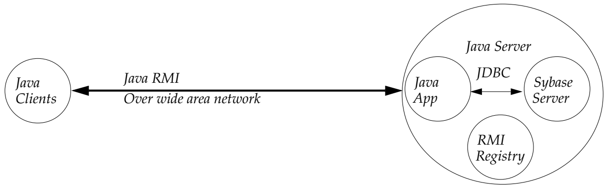
Sizing Test Results - Just One Example

Sun internal sales forecasting application

- **Applets talk over RMI to shared process that uses JDBC to talk to Sybase**

SUN FORECAST EMULATION SIZING

Active Users	Inactive Users	CPU	Network	RAM
10 measured	0	14% of 2 x 200MHz	0.2% of 10Mbit	200MB
100 predicted	0	56% of 6 x 167MHz	3.2% of 10Mbit	300MB
10 predicted	90 predicted	24% of 1 x 200MHz	0.2% of 10Mbit	300MB



Speaker's Notes

Sizing Example - results are provided for several different workloads

Sun internal applications for sales forecasting, order entry, travel expenses management, customer support and others are being migrated to Java. We encouraged the developers to use JavaStar as part of their development testing process. When we came to perform sizing tests, there was no need to debug use with JavaStar, and some usage workloads were already available.

The sales forecasting application was tested in the late stages of development. It will be used by the Sun sales force to forecast opportunities for business and manage their `pipeline' over time. There is also a sales management interface to summarize the total revenue stream. It is based on Java1.1 and uses the Java Remote Method Invocation (RMI) interface to make calls from the applet to a server application written in Java. There are two server processes, a Java RMI registry and a multithreaded application server that handles all the clients (and uses most of the CPU required). The server also runs a Sybase 11 database that uses the JDBC interface to communicate with the Java application server process. The database and RMI registry each account for about 15% of the total CPU load. The database could be located on a separate system if required. A single server will be used to support a large geographic region, so most local sales offices will be communicating over a wide area network. The total number of concurrently connected users could be quite high, but few of them are expected to be active at the same time.We ran a 10 user test using a dual 200MHz Ultra 2 as the server, collected the results with BGS Best/1 and did a basic analysis of the load so that we could predict some larger configurations. The server had 256MB of RAM and 8 4.2GB disks. The total number of sessions we could start was limited by the application setup available at the time. The disk usage was very low, and the network usage is extremely low. The use of high level RMI calls instead of JDBC or the more traditional SQL interfaces over the network seems to minimize network activity. Sybase was configured to use about 100MB of RAM and did not grow when users were added, the RMI registry took 5MB and increased by 0.5MB for 10 users. The application process took 6MB and also increased by about 0.5MB for 10 users. Best/1 decided that about 200MB of RAM was in use overall at 10 users.

NC Booting Storm Analysis

NC's boot their OS over the net

- **Optional flash prom avoids downloading an unchanged JavaOS**
- **Boot transfers ~4MB, in 4000 packets in 40 seconds**
- **TFTP bootstrap and 4MB NFSV2 reads**
- **User login and home page load in 60 packets, 10s**
- **Overall - a bursty but light network load**

Synchronized network booting can be a problem

- **Caused by power outage recovery or shift work**
- **Size using network capacity and SPEC NFS results**
- **Each JavaStation needs 100KB/s and 12.5NFSread/s**
- **Dedicated 10Mbit sustains 8-10 concurrent boots**
- **100Mbit server feeding switched 10Mbit: 80-100 boots**
- **100 boots is ~1250NFSread/s**
- **Ultra1/170 can do 2000+ SPECsfs NFSops**
- **Approx 5min to boot 500 JavaStations on 100Mbit**

Speaker's Notes

Boot Phase Analysis

A simple measurement is the activity during a JavaStation boot to the point where it presents a login screen. This involves the transfer of about 4MB of data using a total of about 4000 packets in about 40 seconds. Apart from a small number of various kinds of name service lookups, the bulk of the traffic consists of a TFTP bootstrap of 158KB, followed by an NFS transfer of the 4MB JavaOS 1.0 image using 8KB NFS V2 reads over UDP. JavaStations that can store a copy of JavaOS in a non-volatile "flash prom" are being developed. They will only download the full 4MB image if a different version is seen on the server, further reducing network loads. Note that subsequent versions of JavaOS may be a different size or use different protocol options. Some other vendors use TFTP to download the entire boot image to their NC. The login process takes about ten seconds and including name service password lookups and a load of the default home page takes only 60 packets.

One NC booting on its own is not going to be a problem. The problems arise when a large population of NCs boot at the same time. Synchronization may occur at certain times of the day if the users are working regular shifts. The bottleneck will be the network infrastructure. Since each boot takes 40 seconds to move about 4MB (500 NFS reads), an average rate of 100KB/s (12.5 NFS read/s) is needed. A 10baseT network can support about 8-10 simultaneous boots before boot time increases significantly (this was tried in practice). A 100baseT server connection feeding ten or more switched 10baseT segments could support 80-100 simultaneous boots. This adds up to 1000 to 1250 NFS read/s. All reads are of the same 4MB file which will be cached in RAM at the server, so there is no need for a high performance disk subsystem. The SPEC LADDIS NFS server benchmark uses a mix that contains a lot of reads and writes. Overall, continuous reads of the same data is likely to be a lighter CPU load than the LADDIS mix, so we can take the LADDIS rating for a server as an over estimate for the CPU power needed during the JavaStation boot process. The LADDIS rating for an uniprocessor UltraSPARC system is over 2000 NFSops/s, so a rough sizing estimate would be that one UltraSPARC CPU can saturate one 100baseT network for a 100 client booting storm. A more realistic aim might be to boot all the users in a five minute period.

Java Source Code Optimization

Java Virtual machine Improvements

- Performance is increasing rapidly
- Competition is driving improvements
- Trade-off startup time for run time with Just-in-time compiler
- Profile feedback optimization at runtime with Hot-Spot
- Direct hardware support for Java is being developed
- Huge investment levels will drive performance up and up

Code Tuning Techniques

- Common problem is inefficient unbuffered I/O
- Code profilers can be used to find slow routines
- Don't over-optimize, next generation JVM will be different
- Take care to use library routines effectively
- Try to write garbage-collection-friendly code
- Don't confuse download class size with runtime memory needs

Speaker's Notes

Java Code

Some useful information has been published at these URL's

http://www.sun.com/sunworldonline/swol-10-1997/swol-10-perf.html - this is the basis of material published in the 2nd edition of my Sun Performance and Tuning book.

http://www.sun.com/solaris/java/wp-java/, especially http://www.sun.com/solaris/java/wp-java/6.html

http://www.cs.cmu.edu/~jch/java/optimization.html

http://www.sun.com/workshop/java/wp-javaio

What does "64bits" mean?

Don't confuse data and addresses

- 64bit data types are already well supported by everyone
- 64bit addresses are the current transition for Sun, HP, IBM, Intel

64bit provides more *capacity* than 32bits

- if you don't need the extra capacity 64bits is not a speedup
- there is higher overhead with 64bit addresses - a slowdown

Solaris 2.7 Support for 64bit addressing

- full user mode support for both 32bit and new 64bit code
- full support for old 32bit systems and 32bit kernel on Ultra
- 64bit kernel on Ultra needed to run any 64bit applications
- full set of both 32 and 64bit Solaris libraries provided
- 64bit kernel needs 64bit device drivers - check 3rd parties
- 64bit applications can only use 64bit libraries - check 3rd parties
- standard LP64 mode - Long and Pointers are 64bit, Int is 32bit

Speaker's Notes

64bits

This presentation is written using the existing 32bit Frame 5.1 release running on an Ultra with a 64bit kernel and 2.7beta. I have not found any programs that fail to work on this system.

Talk to your 3rd party device driver and libraries vendors, they will need to make 64bit versions of their code before you can move to a 64bit version of your applications

The big database vendors are already working with Sun to support greater than 32bit shared memory areas for big databases.

UltraSPARC compiler and linker optimization

Recommendations

- **Use dynamic linking - you get processor specific optimizations**
- **Use SC4.2 at least to get Ultra support**
- **Baseline:** `cc -xO3 -xdepend -xchip=generic -xarch=generic`
- **Portable:** `cc -xO3 -xdepend -xchip=ultra -xarch=generic`
- **Ultra-only:** `cc -fast -xO4 -xdepend -xchip=ultra -xarch=v8plusa`
- **Profiled:** `cc -xprofile=collect -fast -xO4`**... run then compile with**
  `cc -xprofile=use -fast -xO5 -xdepend -xchip=ultra -xarch=v8plusa`
- **64bit using SC5.0:** `cc -xchip=ultra -xarch=v9...`

```
% f77 -o prog -fast -xO4 -depend -xchip=ultra -xarch=v8plus *.f
% file prog
prog:          ELF 32-bit MSB executable SPARC32PLUS Version 1, V8+ Required,
dynamically linked, not stripped
% isalist
sparcv9+vis sparcv9 sparcv8plus+vis sparcv8plus sparcv8 sparcv8-fsmuld sparcv7 sparc
```


Speaker's Notes

The primary interest in UltraSPARC-specific code comes from Fortran end users in the high-performance computing (HPC) marketplace. It is common for HPC end users to have access to the source code of their applications and to be interested in reducing the very long run times associated with large simulation and modeling computations by any means available. There is also relatively less interest in running the code on older, slower SPARC systems. The commonly used SPECfp95 benchmarks contain several examples of this kind of application.

There are also situations where the UltraSPARC system is embedded in a product manufactured by an OEM. Since there is complete control over the hardware and the software combination, it is possible to optimize the two very closely together without concern for backward compatibility.

Using the all-out UltraSPARC compile options, you get access to the 32 bit subset of the SPARC V9 instruction set, and you increase the number of double precision, floating-point registers from 16 to 32. This combination of V8 and V9 is known as the V8plus specification; it is enabled with the `-xarch=v8plus` compiler option. No source code changes will be required, but the code will no longer run on older systems. You can identify the binaries by using the `file` command

The compiler's code generator has many more options. The ones I have described are the ones that usually make a significant difference. In a few cases, I have found that profile feedback can be useful as well. The highest level of optimization is now `-xO5`, and it should only ever be used in conjunction with a collected profile so the code generator knows which loops to optimize aggressively. You simply compile with `xO4` and `xprofile=collect`, run the program, then recompile with `xO5` and `xprofile=use`. This is easy to set up on small benchmarks but is more tricky with a large application.

Tracing and profiling

Tracing Tools

- **truss - shows system calls made by a process**
- **sotruss (Solaris 2.6) - shows shared library calls**
- **prex (Solaris 2.5) - controls TNF tracing for user and kernel code**

Profiling Tools

- **Compiler profile feedback using -xprofile=collect and use**
- **Sampled profile relink using -p and prof/gprof**
- **Function call tree profile recompile using -pg and gprof**
- **Shared library call profiling setenv LD_PROFILE and gprof**
- **Accurate CPU timing using** `/usr/proc/bin/ptime`
- **Microstate process information using pea.se**

```
10:40:16 name lwmx   pid  ppid   uid   usr%   sys% wait% chld%   size   rss   pf
nis_cachemgr     5   176     1     0   1.40   0.19  0.00  0.00  16320 11584  0.0
jre              1 17255  3184  5743  11.80   0.19  0.00  0.00 178112 110336  0.0
sendmail         1 16751     1     0   1.01   0.43  0.00  0.43  18624 16384  0.0
se.sparc.5.6     1 16741  1186  9506   5.90   0.47  0.00  0.00  16320 14976  0.0
imapd            1 16366   198  5710   6.88   1.09  1.02  0.00  34048 29888  0.1
dtmail          10 16364  9070  5710   0.75   1.12  0.00  0.00 102144 94400  0.0
```


Speaker's Notes

Shared library profiling is done by setting LD_PROFILE to the name of the library - see the manpage for ld.so.1, output defaults to /var/tmp and is collected without needing to have any compile options. Data from many runs by different applications is accumulated for the library. Process the result using gprof as usual.

Microstate accounting uses accurate timestamps at state changes rather than sampling at 100Hz to add up CPU usage etc. ptime uses this. See Sunworld Online April 1998 for info on pea.se at

http://www.sun.com/sunworldonline/swol-04-1998/swol-04-perf.html

pea.se is described in the book but is not part of SE3.0, an incremental install tarfile is provided on the SE download web page.

Filesystem issues

ufs - standard, reliable, good for lots of small files

ufs with transaction log - faster writes and recovery

tmpfs - fastest if you have enough RAM, volatile

NFS2 - safe and common, 8KB blocks, slow writes

NFS3 - more readahead and writebehind, faster

- default 32KB block size - fast sequential, may be slow random
- default TCP instead of UDP, more robust over WAN

cachefs - good for read-mostly NFS speedup

Veritas VxFS - option worth buying on large systems

- extent based format much better for large files, high throughput
- hashed directories support large numbers of entries
- transaction log and unbuffered direct access option
- snapshot backup capability

Speaker's Notes

ufs has been improved over the years, in most cases it is good enough when used with the DiskSuite transaction logging option. In 2.6 there is also a directio option for unbuffered file access that can be useful when handling database tables or big datasets. The on-disk format limits sequential throughput for a file to about 30-40 MB/s.

VxFS has been shown to sustain sequential rates up to 1GB/s with careful setup.

Random access is most often disk rotation limited.

File descriptor limits

Processes may want to open lots of files or sockets

Default soft limit is 64, hard limit is 1024

- Applications can call setrlimit to increase to 1024
- shell limit/ulimit commands can preset values
- set rlim_fd_max=8000 in /etc/system for higher default hard limit
- don't ever increase default soft limit above 256

The stdio.h definition of FILE breaks at 256

- for non-stdio, open file and duplicate fd to next free above 256
- tmp = open(...); rawfd = fcntl(tmp, F_DUPFD, 256); close(tmp);
- this keeps low numbered fd's for stdio use
- library calls such as name services secretly use stdio

The select() call bitfield breaks at 1024

- use the underlying poll() system call instead - no limit

Speaker's Notes

Its easy to think your system is working OK with increased default limits, but with a different name service or another library call that reads config files using stdio now and again it could break in the future.

Old versions of select would corrupt memory off the end of the 1024 bit bitfield, later versions check and report errors

Solaris 2.7 for 64bit applications only changes the definition of file to be opaque with no limit, and of select to use 65536 bits in its bitfield. It is still better to use poll - faster and smaller code.

Its impossible to fix this problem in 32bit mode without breaking existing code. The only reason it can be fixed for 64bit is that all these applications must be recompiled in any case, its not a 64 bit issue per-se.

Basic database tuning

Note: I am not a database expert - but I met one once...

Use big shared memory buffers (RAM faster than disk)

Raw disk is lower overhead than filesystems

- raw disk is much faster for write intensive tables
- ufs does extra prefetching so raw may be slower for read scans
- if ufs has enough RAM to cache a whole table it will go very fast
- in most cases with big tables raw and a big shmem area is faster

Wide stripes work well - avoid hot-spots

- Sun's TPC results use VxVM to stripe multiple A3000 RAID5 vols

More indexes reduces load for lookups, slows updates

Big blocksizes speed up sequential scans

Speaker's Notes

When you are tuning databases, it is useful to realize that in many cases the sizing rules that have been developed by database software vendors in the past do not scale well to today's systems. In the mainframe and minicomputer worlds, disk I/O capacity is large, processors are slow, and RAM is expensive. With today's systems, the disk I/O capacity, CPU power, and typical RAM sizes are all huge, but the latency for a single disk read is still very slow in comparison. It is worth trading off a little extra CPU overhead and extra RAM usage for a reduction in I/O requirements, so don't be afraid to experiment with database buffer sizes that are much larger than those recommended in the database vendors' documentation.

Disk workloads and Configurations

Sequential access is ~10 times faster than random

- Sequential rates are now about 10MB/s per disk, 15MB/s soon
- Random rates are 120 operations/sec, 160/sec soon (10000rpm)
- The size of each random read should be as big as possible

Reads should be cached in main memory

- "The only good fast read is the one you didn't have to do"
- Database shared memory or filesystem cache is microseconds
- Disk subsystem cache is milliseconds, plus extra CPU load
- Underlying disk is ~10ms, as its unlikely that data is in cache

Writes should be cached in nonvolatile storage

- Allows write cancellation and coalescing optimizations
- NVRAM inside the system -e.g. Prestoserve
- NVRAM cache inside the disk subsystem - doesn't need a lot
- Dedicated transaction logging disk

Speaker's Notes

Slow idle disks explained

Why do these disks have high svc_t when they are idle?

```
                                  extended disk statistics
disk      r/s  w/s   Kr/s   Kw/s wait actv  svc_t  %w  %b
sd2       1.3  0.3   11.7    3.3  0.1  0.1  146.6   0   3
sd3       0.0  0.1    0.1    0.7  0.0  0.0  131.0   0   0
```

Use prex to turn on kernel TNF probes for disk I/O

- sdstrategy is called when an I/O is started
- biodone is called when it completes
- match the pairs of TNF records to see the time sequences

We find a burst of writes from pid 3 every 30s

- fsflush is updating inodes scattered all over the filesystem
- all writes are issued back to back without waiting to complete
- a long queue forms, each write taking on average ~10ms to service, but response (svc_t) includes a long queue time
- Typically 20 or so writes each 30s is 0% busy, 100-200ms svc_t

Speaker's Notes

Navigating Sun's storage product range

JBOD - just a bunch of disks - high rpm is best

- 7200 rpm SCSI disks at 2.1, 4.2, 9.1GB very fast, 10000rpm soon
- Ultra 5, 10 use 4500rpm IDE disk like a PC - slower but cheap

RSM2000/A3000 and A1000 Hardware RAID5 system

- Dual load balanced 40MB/s SCSI to Symbios controllers
- 64-128MB NVRAM write cache and RAID5 stripe cache
- A1000 is low end package, single controller and 10000 rpm disks

A5000 Fiberchannel Loop Subsystem

- Dual loops at 100MB/s direct to disks, no NVRAM or controller
- Very high throughput for bulk read/write, avoid small writes

Encore SP-40/A7000 “Hands-off” high end storage

- Very high capacity and storage density, phone home service
- Competes directly with EMC storage products, mainframes

Speaker's Notes

An example spec sheet for a common Sun2.1GB disk can be found at

ftp://ftp.seagate.com/techsuppt/scsi/st32171w.txt

The RSM2000 was renamed as the A3000 in early 1998. The A1000 is a small RAID subsystem using a single controller, while the A3000 uses dual controllers with higher capacity packaging. The A3000 design is fully redundant, with no single point of failure and very high performance in RAID5 mode. It has two Ultra SCSI connections to the host system and can sustain 66 Mbytes/s. In the event of a failure, it can continue to work with a single Ultra-SCSI connection.

The A5000 is a building block, used to construct a fiber-based storage fabric, that is very well suited to access from clusters of systems. The package is fully redundant and hot pluggable, with dynamic multipath connections, dual power supplies, and so on. The FC-AL loop goes right through the drives with no separate controller. It also provides the lowest latency for reads and the highest bandwidth for streaming sequential reads and writes. Lack of NVRAM in the basic product can be compensated for by use of dedicated log disks to accelerate writes. It is best used in a mirrored configuration, inasmuch as there is no NVRAM for RAID5 acceleration in the initial product.

In late 1997 Sun aquired the storage business of Encore corporation, and the Encore SP-40 product was renamed the Sun A7000 as the high end of Sun's new StorEdge product line. The A7000 is equally at home connected over an ESCON channel to a mainframe system or connected to Unix systems over SCSI. Uniquely, the same on-disk data can be accessed by both types of system concurrently. The A7000 also supports wide area remote data copy at the storage subsystem level. The A7000 is configured to order by Sun personnel, contains “phone home” modem connections for automated remote diagnostics, and is designed with flexibility and availability as its primary characteristics. With 4 Gbytes of mirrored cache memory and over 1TB per unit the A7000 is a direct competitor to the EMC Symmetrix product line.

All of Sun's disk products will eventually also be supported on HP-UX and NT platforms.

Network interface and NFS metrics

New network interface throughput counters

- In recent `le` and `hme` patches and in Solaris 2.6
- `rbytes`, `obytes` — read and output byte counts
- `multircv`, `multixmt` — multicast byte counts
- `brdcstrcv`, `brdcstxmt` — broadcast byte counts
- `norcvbuf`, `noxmtbuf` — buffer allocation failure counts

Solaris 2.6 iostat shows NFS client mounts

```
crun% iostat -xnP
                                 extended device statistics
  r/s  w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w %b device
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0 0 crun:vold(pid363)
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0 0 servdist:/usr/dist
  0.0  0.5    0.0    7.9  0.0  0.0    0.0   20.7   0 1 servhome:/export/home/adrianc
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0 0 servhome:/var/mail
  0.0  1.3    0.0   10.4  0.0  0.2    0.0  128.0   0 2 c0t2d0s0
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0 0 c0t2d0s2
```


Speaker's Notes

Controlling and Monitoring CPUs

psrinfo - show CPU status and clock rate

psradm - enable/disable CPUs

pbind - bind a process to a CPU

psrset - create sets of CPUs to partition a system

- At least one CPU must remain in the default set, to run kernel services like NFS threads
- All CPUs still take interrupts from their assigned sources
- Processes can be bound to sets

mpstat shows per-CPU counters

```
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0   45   1    0   232    0  780  234  106  201    0   950   72  28   0   0
  1   29   1    0   243    0  810  243  115  186    0  1045   69  31   0   0
  2   27   1    0   235    0  827  243  110  199    0  1000   75  25   0   0
  3   26   0    0   217    0  794  227  120  189    0   925   70  30   0   0
  4    9   0    0   234   92  403   94   84 1157    0   625   66  34   0   0
```


Speaker's Notes

CPU time measurements

CPU time is sampled by the 100Hz clock interrupt

- sampling theory says this is accurate for an unbiased sample
- the sample is very biased, as the clock also schedules the CPU
- daemons that wakeup on the clock timer can hide in the gaps
- problem gets worse as the CPU gets faster

Increase clock interrupt rate?

- set hires_tick=1 sets rate to 1000Hz, good for realtime wakeups
- harder to hide CPU usage, but higher overhead

Use measured CPU time at per-process level

- microstate accounting takes timestamp on each state change
- very accurate and also provides extra information
- still doesn't allow for interrupt overhead
- the pea.se command uses this accurate measurement

Speaker's Notes

Adrian Cockcroft - Sun Microsystems
Sun Performance and Tuning - Java and the Internet
10/20/98
Page 41

Monitoring CPU mutex lock statistics

To fix mutex contention upgrade Solaris

Locking strategies are too complex to be patched

Solaris 2.6 includes the lockstat command

- very powerful and easy to use
- dynamically changes all locks to be instrumented
- displays lots of useful data about which locks are contending

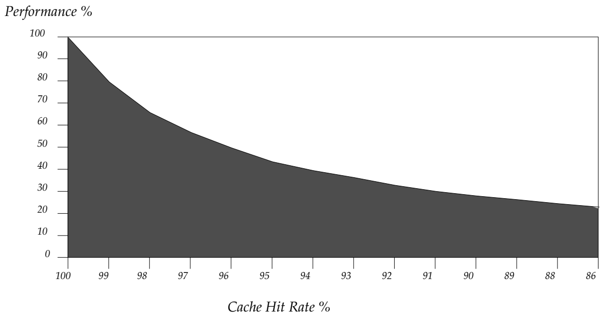
```
# lockstat sleep 5

Adaptive mutex spin: 3318 events

Count indv cuml rcnt     spin Lock                   Caller
-------------------------------------------------------------------------------
  601  18%  18% 1.00        1 flock_lock             cleanlocks+0x10
  302   9%  27% 1.00        7 0xf597aab0             dev_get_dev_info+0x4c
  251   8%  35% 1.00        1 0xf597aab0             mod_rele_dev_by_major+0x2c
  245   7%  42% 1.00        3 0xf597aab0             cdev_size+0x74
  160   5%  47% 1.00        7 0xf5b3c738             ddi_prop_search_common+0x50
... etc and other kinds of locks
```


Speaker's Notes

Adrian Cockcroft - Sun Microsystems
Sun Performance and Tuning - Java and the Internet
10/20/98
Page 42

CPU caches

Application Speed Changes with a 25-Cycle Miss Cost



Speaker's Notes

SPARC Architecture and Implementations

SPARC Version 8 - 32bit SuperSPARC, MicroSPARC

SPARC Version 9 - 64bit UltraSPARC

SPARC V8plus specification of 32bit subset of V9

SPARC V8plusa includes VIS media instructions

Ultra Port Architecture is at system level - sun4u

- common set of system interfaces, one kernel runs any Ultra
- UPA is a switched architecture, not a bus architecture
- Just like switched ethernet is better than shared ethernet

UPA gateways to Gigaplane for larger systems

- Gigaplane is a low latency bus with separate address bus
- Gigaplane XB is a 16 port crossbar switch
- Address "coherency bandwidth" is limiting factor
- Class leading architecture is one reason why Solaris scales

Speaker's Notes

UPA is switched not bussed - think of the difference between shared ethernet and switched ethernet. It has ports of different speeds to match the needs of CPU's RAM, graphics and I/O. Its like an ethernet switch that has gigabit, 100Mbit and 10Mbit ethernet ports, with no collisions on each port.

Older and competing systems use buses that have much more contention and cannot overlap transfers on all buses at once. They also uses their buses to carry address traffic, which is handled outside the UPA datapath via a system controller that sequences all the UPA ports.

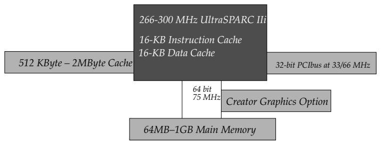
Gigaplane and Gigaplane XB have lower latency and higher bandwidth than their competition, that is a key reason why Solaris can scale up to 30 or 64 CPUs effectively.

Scalability starts with the hardware, then you need a scalable OS, then you need scalable applications, and to get all three you need many years of development effort. Sun started this in 1993 with the SC2000 and has a big lead over the competition.

Ultra 5 and Ultra 10

UltraSPARC IIi Highly Integrated CPU

- Same concept as earlier microSPARC based SPARCstation 5
- RAM control, PCIbus, Graphics interface, and cache
- Very low system cost, almost no “chip set” needed
- Performance a little lower than UltraSPARC II at same clock rate
- Clock rate can go higher, start at 266 and 300MHz
- No UPA as it is all inside the CPU itself, uniprocessor only



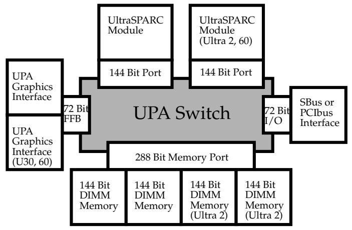
Speaker’s Notes

The UltraSPARC IIi provides a highly integrated design for building low-cost systems in the same way as its microSPARC predecessor. The CPU chip includes a complete memory controller, external cache controller, I/O bus controller (32bit PCIbus this time) and a UPA slave interface for Creator graphics options.

Ultra 1/2/30/60

UltraSPARC I and UltraSPARC II Uni and dual CPUs

- Ultra 1 and Ultra 2 use SBus, Ultra 30 and Ultra 60 use PCIbus
- Common chip set based around UPA ports - ~150-200ns
- Dual CPU systems use wider DRAM banks for lower contention
- CPU clock is a multiple of 83 or 100MHz or 120MHz UPA clock



Speaker’s Notes

Note that only the Ultra 60 has a chip set that can run at up to 120MHz. This will allow divide by three to be maintained above the current 300MHz level for better performance scaling.

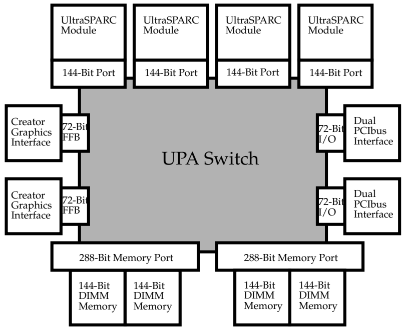
The UPA configuration of these systems is shown. The address connections are omitted, leaving just the data paths through the port. ECC adds to the real width of each port, making the sizes 72, 144, and 288 bits, to carry 64, 128, and 256 bits of data.

Memory is a problem because DRAM is slow to respond, compared to other devices. It is given a double-width port, which is loaded at half the rate (i.e., 50 MHz) from a pair of memory modules. Since a full cache line is 64 bytes, or 576 bits including ECC, a second pair of memory modules is set up to transfer immediately afterward in the dual-CPU, Ultra 2 & 60 version of this design. This interleaving reduces latency in case the other CPU wishes to access the memory port immediately. The Ultra 30 and Ultra 60 support a second UPA graphics interface for Creator or Elite graphics options. These systems benefit from the low latency of the UPA but do not come close to using up its bandwidth. The design is limited by memory bandwidth to around 300–500 Mbytes/s. Main memory latency in this configuration is 28 cycles (168 ns) for a 167 MHz CPU with 83 MHz UPA. Latency in nanoseconds is a little better with a 100 MHz UPA, but the number of clock cycles gets worse as the CPU speed increases—approximately 30 cycles at 200 MHz, 40 at 250 MHz, and 50 at 300 MHz.

The Ultra 30 is packaged as a PCIbus tower system, closer to a PC configuration than to the traditional Sun “pizza box” used in the Ultra 1 and Ultra 2. The Ultra 60 is a dual-CPU version of the Ultra 30. Their architecture is the same as that of the Ultra 2 but with the substitution of a PCIbus controller for the SBus controller used in previous designs. This UPA-to-PCI controller is used in many UltraSPARC systems. It has two PCIbuses: one is a high-speed, 64-bit @ 66 MHz bus with a single slot; the other is a 32/64-bit @ 33 MHz bus that can support several slots.

Enterprise 450 Workgroup server

The biggest UPA based system we could build



Speaker's Notes

To extend the design to cope with the demands of the four-processor Enterprise 450, a more complex switch was developed with twice as many ports, as shown.The configuration includes not only four processor ports but also two independent memory ports, two graphics ports, and two I/O bus ports, each supporting dual PCI buses for a total of four. This design represents the largest balanced configuration that can be built on a single motherboard, yet it still does not saturate the throughput capabilities of the UPA switch. It retains the low latency of the entry level designs, while providing twice the CPU power and twice the memory and I/O bandwidth.

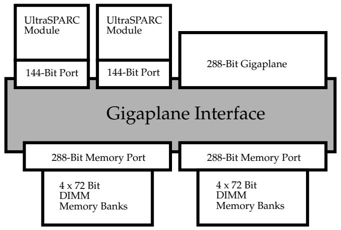
Memory is accessed four DIMMs at a time, and additional performance comes from interleaving the four banks of four DIMMs. The interleaving rules are very restrictive, and for best performance, the simple rule is to always use the same size DIMM throughout. A mixture of DIMM capacities will prevent interleaving. Two banks of identical DIMMS, with two empty banks, will be two-way interleaved. Four banks of identical DIMMs will be four-way interleaved. Any other combination prevents any interleaving. This is unlikely to be an issue on systems with less than four CPUs, but a high-end configuration that is heavily loaded should at least have two-way interleaving, and preferably four.

The low latency of the E450 gives it significantly higher performance than a four-CPU Enterprise Server at substantially lower cost. The scope for expansion is limited, however, so next we will look at the multiboard Ultra Enterprise Server Architecture

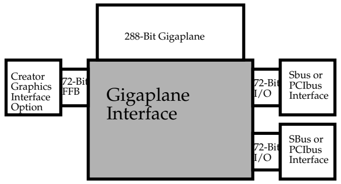
Enterprise 3000-6000 Range

Scalable Gigaplane based Server

- **Uses UPA ports distributed on multiple boards - ~300ns latency**
- **83MHz 256bit data path gives 2.6GB/s sustained throughput**
- **168 to 336MHz UltraSPARC II CPUs with 0.5 to 4MB caches**
- **E3000/4-boards, E4000/8-boards, E6000/16-boards**



CPU/Memory Board



I/O Board

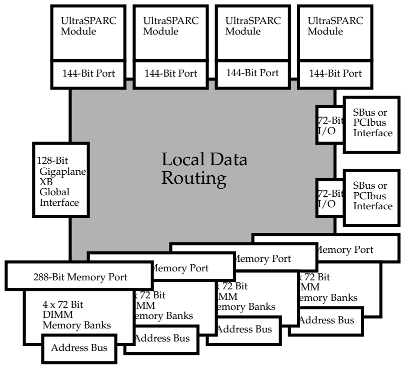
Speaker's Notes

Boards come in two types, CPU+Memory and I/O+Framebuffer. The CPU+Memory board shown allows memory to be added to the system in proportion to the CPU power configured. Some competing designs use separate CPU and memory boards and suffer from having to trade off CPU for memory in big configurations

The three different designs of separate I/O+Framebuffer board shown allow systems to be configured for CPU-intensive or I/O-intensive workloads and also allow a mixture of I/O types (SBus, PCIbus, and SBus+Creator Framebuffer) to be configured as needed

These two board types are combined into three basic packages: the four-board E3000; the eight-board E4000, which is called the E5000 when rack-mounted; and the sixteen-board E6000. The larger systems use a centerplane, with boards plugging in horizontally from both the front and back of the chassis. The E3000 has boards numbered 1, 3, 5, 7 only, the E4000 has boards numbered 0 to 7 inclusive, and the E6000 has boards numbered 0–15.

Enterprise 10000 Mainframe Class Server



16 boards connected via Gigaplane XB

- Four interleaved address buses on each memory bank
- Separate data crossbar
- Good latency (~400ns)
- Much less contention
- Over 10GB/s fully configured
- Needs 6 boards to beat E6000
- Dynamic domain support
- 64 UltraSPARC CPUs
- 64 GB RAM
- 20 TB of disk
- Over 100,000 disk i/o per sec
- Over 3 GB/s of I/O capacity

Speaker’s Notes

The E10000 uses a single, large-format board type, with removable daughter cards that can configure SBus or PCIbus options for I/O. Each board includes four CPU modules, four memory banks, and two I/O buses, as shown. The packaging takes up to sixteen boards, connected as described in “The Gigaplane XB Crossbar” on page 289. More information is available from the “Ultra Enterprise 10000 System Overview White Paper” and other papers at http://www.sun.com/servers/ultra_enterprise/10000/wp

The four memory banks are interleaved. Each memory bank has its own global address bus that maintains coherency with that bank on other boards.

Each board can function independently, and single boards or groups of boards can be configured into domains. (Domain memory bandwidth is described in Table 11-18 on page 291.) The system can boot multiple copies of Solaris, one per domain. The crossbar switch is configured so that data transfers between boards in one domain do not affect the performance of any other domain. There is also very good fault isolation between domains. Hardware failures are not propagated. Domains can be changed on line via dynamic reconfiguration.

Cache principles

Temporal locality - “close in time”

- If you need something frequently, keep it near you
- If you don’t use it for a while, put it back
- If you change it, save the change by putting it back

Spacial locality - “close in space - nearby”

- If you go to get one thing, get other stuff that is nearby
- You may save a trip by prefetching things
- You can waste bandwidth if you fetch too much you don’t use

Caches work well with randomness

- Randomness prevents worst case behaviour
- Deterministic patterns often cause cache busting accesses
- Very careful cache friendly tuning can give great speedups

Speaker’s Notes

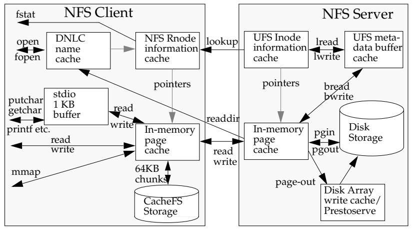
Caches work on two basic principles that should be quite familiar to you from everyday life. The first is that if you spend a long time going to get something and you think you may need it again soon, you keep it nearby. For example, if you are working on your car, you find a 15 mm spanner in your tool box, then crawl under the car. If you then need a 10 mm spanner, you don’t put the 15 mm back in the tool box, you leave it under the car. When you have finished, there is a *cache* of tools in a pile under the car that make up your *working set*. If you allowed only one tool at a time under the car, you would waste a lot of time going back to the tool box. However, if the pile of tools gets too large, you may decide to put away a few that you think you won’t need any more.

The second principle is that when you get something, you can save time by getting a few extra things that you think you might need soon. In the car example, you could grab a handful of spanners each time, thus saving yourself from having to make a few trips. You find that the pile under the car gets bigger much more quickly, you never use some tools, and it takes you longer to pick up and put down a handful of tools on each trip to the tool box.

Many kinds of information can be cached: the value of a memory location, the location of a file, or the network address of another system. When the information is needed, the system first looks for it in the cache. This approach requires that each item in the cache has a unique name or tag associated with the information. When a cache lookup occurs, the first thing that happens is a search during which the system tries to match the name you gave it with one of the tags. If the search succeeds, the associated data is returned; this case is called a read hit. This situation is the best case, as it is both quick and successful. The two measures of interest are the read hit rate and the read hit cost. The read hit rate is the number of read hits that occur per second; the read hit cost is the time it takes to search the cache and return the data.If the search fails and the data is not in the cache, the case is called a read miss. The rate at which this situation occurs is called the read miss rate, and the time taken is called the read miss cost. A read miss corresponds to the work that would be needed if the cache did not exist, plus the added overhead of checking the cache.

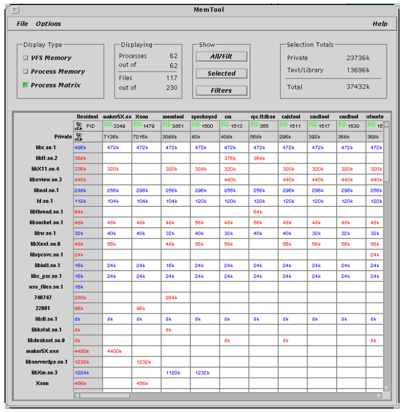
UFS and NFS access caching

A complex set of caches speeds up file access



Speaker's Notes

Memory usage and sizing tools



Speaker's Notes

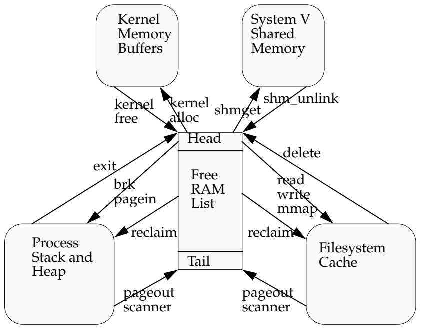
There are four primary consumers of memory: the kernel, processes, filesystem caches, and System V shared memory. Kernel memory usage is described in "Kernel Memory Allocation" on page 365. Process memory usage is described in "Process Memory Usage Information" on page 421. Filesystem caches invisibly consume large amounts of memory; there are no measurements of this usage in Solaris 2. System V Shared Memory can be observed by means of the `ipcs` command.

The Solaris Memory System is a white paper available via http://www.sun.com/sun-on-net/performance.html, written by my colleague Richard McDougall. It explains how to measure and size memory usage. Richard has also written a loadable kernel module that instruments memory usage by processes and the file system in detail and a set of tools that view this data. The `memtool` package has also been known internally at Sun as the `bunyip` tool for the last few years. The tools can be obtained by sending email to memtool-request@chessie.eng.sun.com. The loadable kernel module is unsupported and should not be used in production environments, although it is suitable for use in benchmarks and sizing exercises. There is a significant risk that in the future the module could interact with updated kernel patches and cause a machine to crash. At present (early 1998), modules that work reliably with current patch levels of Solaris 2.5.1 and Solaris 2.6 are provided.

The process memory usage command `/usr/proc/bin/pmap -x` is provided only in Solaris 2.6. It is based on code written by Richard, and his `memtool` package includes this functionality for Solaris 2.5.1. Filesystem cache memory usage is instrumented in detail by `memtool`.

The memory go round

Memory usage flows between subsystems



Speaker's Notes

Memory is one of the main resources in the system. There are four main consumers of memory, and when memory is needed, they all obtain it from the free list. The figure shows these consumers and how they relate to the free list. When memory is needed, it is taken from the head of the free list. When memory is put back on the free list, there are two choices. If the page still contains valid data, it is put on the tail of the list so it will not be reused for as long as possible. If the page has no useful content, it is put at the head of the list for immediate reuse. The kernel keeps track of valid pages in the free list so that they can be reclaimed if their content is requested, thereby saving a disk I/O.

The `vmstat` reclaim counter is two-edged. On one hand, it is good that a page fault was serviced by a reclaim, rather than a page-in that would cause a disk read. On the other hand, you don't want active pages to be stolen and end up on the free list in the first place. The `vmstat free` value is simply the size of the free list, in Kbytes. The way the size varies is what tends to confuse people. The most important value reported by `vmstat` is the scan rate - `sr`. If it is zero or close to zero, then you can be sure that the system does have sufficient memory. If it is always high (hundreds to thousands of pages/second), then adding more memory is likely to help.

Swap space

Swap is very confusing and badly instrumented

```
# se swap.se
ani_max 54814  ani_resv 19429  ani_free 37981  availrmem 13859  swapfs_minfree 1972
ramres 11887  swap_resv 19429  swap_alloc 16833  swap_avail 47272  swap_free 49868

Misleading data printed by swap -s
134664 K allocated + 20768 K reserved = 155432 K used, 378176 K available
Corrected labels:
134664 K allocated + 20768 K unallocated = 155432 K reserved, 378176 K available

Mislabelled sar -r 1
freeswap (really swap available) 756352 blocks

Useful swap data: Total swap 520 M
available 369 M  reserved 151 M  Total disk 428 M  Total RAM 92 M
# swap -s
total: 134056k bytes allocated + 20800k reserved = 154856k used, 378752k available
# sar -r 1
18:40:51 freemem freeswap
18:40:52    4152   756912
```


Speaker's Notes

Swap space is really a misnomer of what really is the paging space. Almost all the accesses are page related rather than being whole-process swapping.

Swap space is allocated from spare RAM and from swap disk. The measures provided are based on two sets of underlying numbers. One set relates to physical swap disk, the other set relates to RAM used as swap space by pages in memory.

Swap space is used in two stages. When memory is requested (for example, via a `malloc` call) swap is reserved and a mapping is made against the `/dev/zero` device. Reservations are made against available disk-based swap to start with. When that is all gone, RAM is reserved instead. When these pages are first accessed, physical pages are obtained from the free list and filled with zeros, and pages of swap become allocated rather than reserved. In effect, reservations are initially taken out of disk-based swap, but allocations are initially taken out of RAM-based swap. When a page of anonymous RAM is stolen by the page scanner, the data is written out to the swap space, i.e., the swap allocation is moved from memory to disk, and the memory is freed.

Memory space that is mapped but never used stays in the reserved state, and the reservation consumes swap space. This behavior is common for large database systems and is the reason why large amounts of swap disk must be configured to run applications like Oracle and SAP R/3, even though they are unlikely to allocate all the reserved space.

The first swap partition allocated is also used as the system dump space to store a kernel crash dump into. It is a good idea to have plenty of disk space set aside in `/var/crash` and to enable `savecore` by uncommenting the commands in `/etc/rc2.d/S20sysetup`. If you forget and think that there may have been an unsaved crash dump, you can try running `savecore` long after the system has rebooted. The crash dump is stored at the very end of the swap partition, and the `savecore` command can tell if it has been overwritten yet.

Kernel values, tunables and defaults

There is far too much emphasis on kernel tweaks

- There really are no “magic bullet” tunables
- It rarely makes a significant difference
- Fix the system configuration or tune the application instead

Solaris 2 is like a modern car

- Very few adjustable components
- “No user serviceable parts inside”
- But Unix has so much history people think it is like a 70’s car

Solaris 2 really is dynamic, adaptive and self-tuning

- DNLC is the last preallocated table, and will be dynamic in 2.7
- Most other “traditional Unix” tunables are just advisory limits

Tweaks may be workarounds for bugs/problems

- Patch or OS release removes the problem - remove the tweak

Speaker’s Notes

Tuning the kernel is a hard subject to deal with. Some tunables are well known and easy to explain. Others are more complex or change from one release to the next. The settings in use are often based on out-of-date folklore. In normal use there is no need to tune the Solaris 2 kernel; it dynamically adapts itself to the hardware configuration and the application workload. If it isn’t working properly, you may have a configuration error, a hardware failure, or a software bug. To fix a problem, you check the configuration, make sure all the hardware is OK, and load the latest software patches.

Some tunable parameters configure the sizes or limits of some data structures. The size of these data structures has no effect on performance, but if they are set too low, an application might not run at all. Configuring shared memory allocations for databases falls into this category. Kernel configuration and tuning variables are normally edited into the `/etc./system` file by hand. Unfortunately, any kernel data that has a symbol can be set via this file at boot time, whether or not it is a documented tunable.

So why is there so much emphasis on kernel tuning? And why are there such high expectations of the performance boost available from kernel tweaks? I think the reasons are historical, and I’ll use a car analogy to explain it. Compare a 1970s car with a 1998 car. The older car has a carburetor, needs regular tune-ups, and is likely to be temperamental at best. The 1998 car has computerized fuel injection, self-adjusting engine components, and is easier to live with, consistent and reliable. If the old car won’t start reliably, you get out the workshop manual and tinker with a large number of fine adjustments. The 1998 car’s computerized ignition and fuel injection systems have no user serviceable components. Unix started out in an environment where the end users had source code and did their own tuning and support. As Unix became a commercial platform for running applications, the end users changed. Tinkering with the operating system is a distraction. SunSoft engineers have put a lot of effort into automating the tuning for Solaris 2. It adaptively scales according to the hardware capabilities and the workload it is running. The self-configuring and tuning nature of Solaris contributes to its ease of use and greatly reduces the gains from tweaking it yourself. Each successive version of Solaris 2 has removed tuning variables by converting hand-adjusted values into adaptively managed limits.

Measurement Data Interfaces

The X/Open UMA Standard

- Universal Measurement Architecture (actually Unix specific)
- Aims to make tools share standardized collected data
- I worked to make the data definitions more useful
- The programming interface is too complex
- Insufficient business case for tools vendors to support UMA

The ARM Standard

- Application Resource Measurement
- HP, Tivoli, BGS implement ARM collectors and analysis
- End-to-end response time and resource usage
- Based on distributed transactions
- Uses library calls in applications to collect data
- Needs support from application vendors

Speaker’s Notes

Reading kernel memory - kvm

Examples

- Older Unix variants use this only - SunOS 4
- Use kernel namelist symbol table and open /dev/kmem
- Solaris wraps up interface in kvm library

Advantages

- Still the only way to get at some kinds of data
- Low overhead, fast bulk data capture

Disadvantages

- Too much intimate implementation detail exposed
- No locking protection to ensure consistent data
- Highly non-portable, unstable over releases and patches
- Breaks when kernel moves to 64bit address support

Speaker's Notes

The book contains detailed code examples for use of this interface

Structured Kernel Statistics - kstat

Examples

- Solaris 2 introduced kstat and extended usage in each release
- Solaris 2 vmstat, iostat, sar, network interface stats, etc.

Advantages

- The recommended and supported metric access API
- Does not require setuid root commands to access for reads
- Individual named metrics stable over releases
- Consistent data using locking, but low overhead
- Unchanged when kernel moves to 64bit address support
- Extensible to add metrics without breaking existing code

Disadvantages

- Somewhat complex hierarchical kstat_chain structure
- State changes (device online/offline) cause kstat_chain rebuild

Speaker's Notes

The book contains detailed code examples for use of this interface

Process based data - /proc

Examples

- Used by ps, proctool and debuggers, pea.se
- proc(1) tools on Solaris in /usr/proc/bin
- Generic UNIX SVR4 and Linux have similar interfaces
- Solaris 2.6 and Linux have /proc/pid/metric hierarchy

Advantages

- The recommended and supported process access API
- Metric data structures reasonably stable over releases
- Consistent data using locking
- Microstate data provides accurate process state timers

Disadvantages

- High overhead for open/ioctl/close for every process
- Solaris 2 microstate data collection adds some overhead

Speaker's Notes

The book contains detailed code examples for use of this interface

Network protocol data - mib and ndd

Examples

- Based on a streams module interface
- Solaris 2 ndd used to configure protocols and interfaces
- Solaris 2 netstat -s and snmpd get TCP stats etc

Advantages

- Individual named metrics reasonably stable over releases
- Consistent data using locking, overhead cut in Solaris 2.6
- Extensible to add metrics without breaking existing code
- ndd can retune TCP online without reboot

Disadvantages

- Underlying API is not supported, SNMP access is preferred

Speaker's Notes

The book contains detailed code examples for use of this interface

Trace data - TNF, prex

Examples

- Solaris 2.5 introduced TNF probes and prex command
- User level probe library for hires tracepoints allows instrumentation of multithreaded applications
- Kernel level probes allow disk I/O and scheduler tracing
- GUI tools available from http://opcom.sun.ca

Advantages

- Low overhead, microsecond resolution, powerful GUI tool
- I/O trace capability is extremely useful

Disadvantages

- Kernel trace buffer can overflow and lose trace records
- Kernel buffer capture is snapshot only, clumsy to use

Speaker's Notes

The book contains detailed examples for use of this interface

Hardware counters

Examples (Sun internal tools)

- time.mxcc for SuperSPARC cache hit rate
- hstat for UltraSPARC pipeline and cache counters
- System wide instrumentation for server backplanes etc.
- Most modern CPUs and systems have counters

Advantages

- See what is really happening, more accurate than kernel stats
- Cache usage useful for tuning code algorithms
- Pipeline usage useful for HPC tuning for megaflops
- Backplane and memory bank usage useful for database servers

Disadvantages

- Not publicly available, and *very* hardware specific
- Data is confusing, lots of architectural background info needed

Speaker's Notes

There are plans to support hardware counter data collection and reporting in a future Solaris release.

Application instrumentation

Examples

- Oracle V$ Tables
- SAP R/3 CCMS performance data
- HP/Tivoli's ARM instrumentation

Advantages

- Focussed application specific information
- High end tools vendors do collect and use data from major applications

Disadvantages

- No common access methods
- ARM is a collection interface only, vendor specific tools, data
- Very few applications are instrumented, even fewer have support from performance tools vendors

Speaker's Notes

Configuration information

Configuration data comes from too many sources

- Device tree displayed by prtconf and prtdiag
- Symon configd adds more information on device usage
- SunVTS component test system has vtsprobe to get config
- SCSI device info using iostat -E in Solaris 2.6
- Logical volume data from vxprint and metastat
- A3000 HW RAID info from lad, drivutil, rdacutil, and raidutil

It is very hard to combine all this data

UMA defines a common config data system

- But no-one has implemented it

Speaker's Notes

SE toolkit Example Tools

A free experimental performance toolkit

- **Written by Rich Pettit with contributions from me**
- **Get from http://www.sun.com/sun-on-net/performance/se3**
- **Support for SPARC & x86 Solaris 2.5, 2.5.1, 2.6**

Function	Example SE Programs		
Rule Monitors	cpg.se	monlog.se	mon_cm.se
	live_test.se	percollator.se	zoom.se
	virtual_adrian.se	virtual_adrian_lite.se	
Disk Monitors	siostat.se	xio.se	xiostat.se
	iomonitor.se	iost.se	xit.se
	disks.se		
CPU Monitors	cpu_meter.se	vmmonitor.se	mpvmstat.se
Process Monitors	msacct.se	pea.se	ps-ax.se
	ps-p.se	pwatch.se	pw.se
Network Monitors	net.se	tcp_monitor.se	netmonitor.se
	netstatx.se	nx.se	nfsmonitor.se
Clones	iostat.se	uname.se	vmstat.se
	nfsstat-m.se	perfmeter.se	xload.se
Data browsers	aw.se	infotool.se	multi_meter.se
Contributed Code	anasa	dfstats	kview
	systune	watch	
Test Programs	syslog.se	cpus.se	pure_test.se
	collisions.se	uptime.se	dumpkstats.se
	net_example	nproc.se	kvmname.se

Speaker's Notes

The toolkit is continuously being developed and upgraded. We decided that it was not a good idea to put a copy on a disk and include it with the book. It is much better to use the Internet to download the packages, which are about 1 megabyte in total. One problem is that Internet addresses change over time, as web sites are reorganized, and the location may change over time. The best way to get the latest information on SE is to read my latest SunWorld Online performance Q&A column. There is always a link to the SE download area at the end in the "Resources" section. SunWorld Online is located at http://www.sun.com/sunworldonline. The SE3.0 release described in this book can be found at http://www.sun.com/sun-on-net/performance/se3.

SE is provided as three packages. Two are largely by Richard Pettit and contain the interpreter and basic examples (`RICHPse`) and the SE Graphical Extension libraries (`RICHPsex`). The third is largely by Adrian Cockcroft and contains rules and related tools (`ANCrules`). There is a dependency on an optional Solaris package SUNWsprot, which contains the `/usr/ccs/lib/cpp` command and is part of the standard Solaris 2 distribution. If you have a full installation of Solaris 2, it will be present, but users may get confused, and think—incorrectly—that this package has something to do with a compiler product. The three packages are provided in compressed tar format. You can use `pkgchk` to see if an SE release is already installed and use `pkgrm` to remove the packages if they are there. The packages must be uncompressed and extracted into their installable format; `pkginfo` is used to check that they are ready to be loaded, then `pkgadd` is used to load them, specifying the current directory. The only question you are asked during installation of `RICHPse` and `RICHPsex` is whether you would like to let us know that you installed SE by sending an email that specifies the organization you work for. During ANCrules installation, you are asked if you want to install scripts in /etc/rc2.d and /etc/rc3.d so that SE monitoring tools are started automatically at boot time. The first time you try this out, you should answer "n". You can then try out the scripts by hand as a regular user to build up your confidence in them, then try them as root. If all seems well, enable the auto-startup next time you add ANCrules via pkgadd on a system.

"virtual adrian" rules explained

Disk Rule for all disks at once

- **Provides overall and per-disk states**
- **Live rule handles floppy and cdrom cases**
- **Looks for slow disks and unbalanced usage**

Network Rule for all networks at once

- **Rule picks out ethernets only**
- **Looks for slow nets and unbalanced usage**

Client RPC Rule

- **Looks for lossy nets and slow NFS servers**

Swap Rule

- **Looks for lack of swap space**

RAM Rule

- **Looks for short page residence times**

Speaker's Notes

"virtual adrian" rules

Kernel Memory Rule

- Looks for allocation failures

CPU Power Rule

- Scales on MP systems
- Looks for long run queue delays

Mutex Rule

- Looks for kernel lock contention

DNLC Rule

- Poor Directory Name Lookup Cache hitrate

Inode Cache Rule

- Poor inode cache hitrate

TCP Rule - described earlier

Speaker's Notes

SE language features

SE is an interpreted dialect of C

- Not a new language to learn from scratch
- Standard /usr/ccs/bin/cpp is used
- Main omissions - pointer types 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

The SE Language is being extended

- better pointer support
- better ways to handle dynamically allocated data

Speaker's Notes

zoom.se GUI based rule front end

Speaker's Notes

Adrian Cockcroft - Sun Microsystems
Sun Performance and Tuning - Java and the Internet
10/20/98
Page 69

Web Pages and Books

Starting point for performance information

- http://www.sun.com/sun-on-net/performance

SunWorld Online Performance Q&A

- new on the 1st of every month, back issues to 1995
- http://www.sunworld.com

Configuration and Capacity Planning for Solaris Servers

- Brian Wong - Sun Press/Prentice Hall, 1996 ISBN 0-13-349952-9

The Practical Performance Analyst

- Dr. Neil Gunther - McGraw-Hill, 1998 ISBN 0-07-912946-3

Speaker's Notes

Adrian Cockcroft - Sun Microsystems
Sun Performance and Tuning - Java and the Internet
10/20/98
Page 70