Quick Tips and Recipes

This book presents too much detail for you to absorb quickly, so, to start you off, I'll first list a few recurring situations and frequently asked questions I have encountered, with references to the rest of the book for in-depth detail. If you are new to Solaris, some of these tips may be confusing. You can follow the cross-references to get a deeper explanation, or read on and return to the tips later.

For situations where you have no idea where to begin, I have also outlined a "cold start" procedure that should help you focus on problem areas. That section is followed by some performance-oriented configuration recipes for common system types.

Quick Reference for Common Tuning Tips

This list focuses primarily, but not exclusively, on servers running Solaris 2. It should help you decide whether you have overloaded the disks, network, available RAM, or CPUs.

The system will usually have a disk bottleneck.

In nearly every case the most serious bottleneck is an overloaded or slow disk. Use `iostat -xn 30`[1] to look for disks that are more than 5 percent busy and have average response times of more than 30 ms. The response time is mislabeled `svc_t`; it is the time between a user process issuing a read and the read completing (for example), so it is often in the critical path for user response times. If many other processes are accessing one disk, a queue can form, and response times of over 1000 ms (not a misprint, over one second!) can occur as you wait to get to the front of the queue. With careful configuration and a disk array controller that includes nonvolatile RAM (NVRAM), you can keep average response time under 10 ms. See "Load Monitoring and Balancing" on page 75 for more details.

Increasing the inode cache size may help reduce the number of disk I/Os required to manage file systems; see "The Inode Cache and File Data Caching" on page 362. If you have a large memory configuration, the inode cache will already be big enough.

1. The `-xn` option is Solaris 2.6 specific; use `-x` in previous releases.

Keep checking `iostat -xn 30` as tuning progresses. When a bottleneck is removed, the system may start to run faster, and as more work is done, some other disk will overload. At some point, you may need to stripe file systems and tablespaces over multiple disks.

Disks that contain UFS file systems will show high average service times when they are idle. This is caused by short bursts of updates from the filesystem flushing process described in “Idle Disks and Long Service Times” on page 186. *This effect can safely be ignored*, hence the 5% busy threshold mentioned above.

Poor NFS response times may be hard to see.

Waiting for a network-mounted file system to respond is not counted in the same way as waiting for a local disk. The system will appear to be idle when it is really in a network I/O wait state. Use `nfsstat -m` or (if you are running Solaris 2.6) `iostat -xn` to find out which NFS® server is likely to be the problem, go to it, and check its disk performance. You should look at the NFS operation mix with `nfsstat` on both the client and server and, if writes are common or the server’s disk is too busy, configure a Prestoserve or NVRAM at the server. A 10-Mbit Ethernet will be overloaded very easily; the network should be replaced with 100-Mbit Ethernet, preferably in switched full-duplex mode. See the *SMCC NFS Server Performance and Tuning Guide* on the Solaris SMCC Hardware AnswerBook® CD, or look for it on http://docs.sun.com.

Avoid the common `vmstat` misconceptions.

When you look at `vmstat`, *please* don’t waste time worrying about where all the RAM has gone. After a while, the free list will stabilize at around 3% of the total memory configured[2]. The system stops bothering to reclaim memory above this level, even when you aren’t running anything. See “Understanding vmstat and sar Output” on page 320. You can also ignore the third “w” column, which is a count of how many *idle* processes are currently swapped out. Of course, you must also remember to ignore the first line output by `vmstat`.

```
% vmstat 5
 procs     memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr f0 s0 s1 s5   in   sy   cs us sy id
 0 0 0   9760 16208   0   4  6  2  5  0  0  0 141 35 19 149  898   99  6  2 92
 0 0 12 212672 1776   0   1  3  0  0  0  0  0  1  0  0  105  140   50  0  0 99
```

2. The actual level depends on the version of the operating system you are running; it may be fixed at a megabyte or less.

Don't panic when you see page-ins and page-outs in `vmstat`.

These activities are normal since all filesystem I/O is done by means of the paging process. Hundreds or thousands of kilobytes paged in and paged out are not a cause for concern, just a sign that the system is working hard.

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

When you really are short of memory, the scanner will be running continuously at a high rate (over 200 pages/second averaged over 30 seconds). If it runs in separated high-level bursts and you are running Solaris 2.5 or earlier, make sure you have a recent kernel patch installed—an updated paging algorithm in Solaris 2.5.1 was back-ported to previous releases. See "Understanding vmstat and sar Output" on page 320.

Look for a long run queue (`vmstat procs r`).

If the run queue or load average is more than four times the number of CPUs, then processes end up waiting too long for a slice of CPU time. This waiting can increase the interactive response time seen by users. Add more CPU power to the system (see "Monitoring Processors" on page 229).

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

A blocked process is a sign of a disk bottleneck. If the number of blocked processes approaches or exceeds the number in the run queue, tune your disk subsystem. Whenever there are any blocked processes, *all CPU idle time is treated as wait for I/O time!* The `vmstat` command correctly includes wait for I/O in its idle value, but it can be viewed with `iostat` or `sar`. If you are running database batch jobs, you should expect to have some blocked processes, but you can increase batch throughput by removing disk bottlenecks.

Check for CPU system time dominating user time.

If there is more system time than user time and the machine is not an NFS server, you may have a problem. NFS service is entirely inside the kernel, so system time will normally dominate on an NFS server. To find out the source of system calls, see "Tracing Applications" on page 155. To look for high interrupt rates and excessive mutex contention, see "Use of mpstat to Monitor Interrupts and Mutexes" on page 236.

Watch out for processes that hog the CPU.

Processes sometimes fail in a way that consumes an entire CPU. This type of failure can make the machine seem sluggish. Watch for processes that are accumulating CPU time rapidly when you don't think they should be. Use `ps` or see "pea.se" on page 488. If you find that the system process `fsflush` is using a lot of CPU power, see the description of the kernel variables "`tune_t_fsflushr` and `autoup`" on page 339.

Cold Start Procedure

I see a lot of questions from users or administrators who have decided that they have a performance problem but don't know where to start or what information to provide when they ask for help. I have seen email from people who just say "my system is slow" and give no additional information at all. I have also seen 10-megabyte email messages with 20 attachments containing days of `vmstat`, `sar`, and `iostat` reports, but with no indication of what application the machine is supposed to be running. In this section, I'll lead you through the initial questions that need to be answered. This may be enough to get you on the right track to solving the problem yourself, and it will make it easier to ask for help effectively.

1. **What is the business function of the system?**
   What is the system used for? What is its primary application? It could be a file server, database server, end-user CAD workstation, internet server, embedded control system.

2. **Who and where are the users?**
   How many users are there, how do they use the system, what kind of work patterns do they have? They might be a classroom full of students, people browsing the Internet from home, data entry clerks, development engineers, real-time data feeds, batch jobs. Are the end users directly connected? From what kind of device?

3. **Who says there is a performance problem, and what is slow?**
   Are the end users complaining, or do you have some objective business measure like batch jobs not completing quickly enough? If there are no complaints, then you should be measuring business-oriented throughput and response times, together with system utilization levels. Don't waste time worrying about obscure kernel measurements. If you have established a baseline of utilization, business throughput, and response times, then it is obvious when there is a problem because the response time will have increased, and that is what drives user perceptions of performance. It is useful to have real measures of response times or a way to derive them. You may get only subjective measures—"it feels sluggish today"—or have to use a stopwatch to time things. See "Collecting Measurements" on page 48.

4. **What is the system configuration?**
   How many machines are involved, what is the CPU, memory, network, and disk setup, what version of Solaris is running, what relevant patches are loaded? A good description of a system might be something like this: an Ultra2/2200, with 512 MB, one 100-Mbit switched duplex Ethernet, two internal 2-GB disks with six external 4-GB disks on their own controller, running Solaris 2.5.1 with the latest kernel, network device, and TCP patches.

5. **What application software is in use?**
   If the system is just running Solaris services, which ones are most significant? If it is an NFS server, is it running NFS V2 or NFS V3 (this depends mostly upon the NFS clients). If it is a web server, is it running Sun's SWS, Netscape, or Apache (and which version)? If it is a database server, which database is it, and are the database tables running on raw disk or in filesystem tables? Has a database vendor specialist checked that the database is configured for good performance and indexed correctly?

6. **What are the busy processes on the system doing?**
   A system becomes busy by running application processes; the most important thing to look at is which processes are busy, who started them, how much CPU they are using, how much memory they are using, how long they have been running. If you may have a lot of short-lived processes, the only way to catch their usage is to use system accounting; see "Using Accounting to Monitor the Workload" on page 48. For long-lived processes, you can use the `ps` command or a tool such as `top`, `proctool`, or `symon`; see "Sun Symon" on page 35. A simple and effective summary is to use the old Berkeley version of `ps` to get a top ten listing, as shown in Figure 1-1. On a large system, there may be a lot more than ten busy processes, so get all that are using significant amounts of CPU so that you have captured 90% or more of the CPU consumption by processes.

Figure 1-1 Example Listing the Busiest Processes on a System

```
% /usr/ucb/ps uaxw | head
USER       PID %CPU %MEM   SZ  RSS TT       S    START  TIME COMMAND
adrianc   2431 17.9 22.63857628568 ?        S   Oct 13  7:38 maker
adrianc    666  3.0 14.913073618848 console  R   Oct 02 12:28 /usr/openwin/bin/X :0
root      6268  0.2  0.9 1120 1072 pts/4    O 17:00:29  0:00 /usr/ucb/ps uaxw
adrianc   2936  0.1  1.8 3672 2248 ??      S   Oct 14  0:04 /usr/openwin/bin/cmdtool
root         3  0.1  0.0    0    0 ?        S   Oct 02  2:17 fsflush
root         0  0.0  0.0    0    0 ?        T   Oct 02  0:00 sched
root         1  0.0  0.1 1664  136 ?        S   Oct 02  0:00 /etc/init -
root         2  0.0  0.0    0    0 ?        S   Oct 02  0:00 pageout
root        93  0.0  0.2 1392  216 ?        S   Oct 02  0:00 /usr/sbin/in.routed -q
```

Unfortunately, some of the numbers above run together: the %MEM field shows the RSS as a percentage of total memory. SZ shows the size of the process virtual address space; for X servers this size includes a memory-mapped frame buffer, and in this case,

for a Creator3D the frame buffer address space adds over 100 megabytes to the total. For normal processes, a large SZ indicates a large swap space usage. The RSS column shows the amount of RAM mapped to that process, including RAM shared with other processes. In this case, PID 2431 has an SZ of 38576 Kbytes and RSS of 28568 Kbytes, 22.6% of the available memory on this 128-Mbyte Ultra. The X server has an SZ of 130736 Kbytes and an RSS of 18848 Kbytes.

7. **What are the CPU and disk utilization levels?**
   How busy is the CPU overall, what's the proportion of user and system CPU time, how busy are the disks, which ones have the highest load? All this information can be seen with `iostat -xc` (`iostat -xPnce` in Solaris 2.6—think of "expense" to remember the new options). Don't collect more than 100 samples, strip out all the idle disks, and set your recording interval to match the time span you need to instrument. For a 24-hour day, 15-minute intervals are fine. For a 10-minute period when the system is busy, 10-second intervals are fine. The shorter the time interval, the more "noisy" the data will be because the peaks are not smoothed out over time. Gathering both a long-term and a short-term peak view helps highlight the problem areas. One way to collect this data is to use the SE toolkit—a script I wrote, called `virtual_adrian.s`, (See "The SymbEL Language" on page 505, and "virtual_adrian.se and /etc/rc2.d/S90va_monitor" on page 498.) writes out to a text-based log whenever it sees part of the system (a disk or whatever) that seems to be slow or overloaded.

8. **What is making the disks busy?**
   If the whole disk subsystem is idle, then you can skip this question. The per-process data does not tell you which disks the processes are accessing. Use the `df` command to list mounted file systems, and use `showmount` to show which ones are exported from an NFS server; then, figure out how the applications are installed to work out which disks are being hit and where raw database tables are located. The `swap -l` command lists swap file locations; watch these carefully in the `iostat` data because they all become very busy with paging activity when there is a memory shortage.

9. **What is the network name service configuration?**
   If the machine is responding slowly but does not seem to be at all busy, it may be waiting for some other system to respond to a request. A surprising number of problems can be caused by badly configured name services. Check `/etc/nsswitch.conf` and `/etc/resolv.conf` to see if DNS, NIS, or NIS+ is in use. Make sure the name servers are all running and responding quickly. Also check that the system is properly routing over the network.

10. How much network activity is there?

You need to look at the packet rate on each interface, the NFS client and server operation rates, and the TCP connection rate, throughput, and retransmission rate. One way is to run this twice, separated by a defined time interval.

```
% netstat -i; nfsstat; netstat -s
```

Another way is to use the SE toolkit's `nx.se` script that monitors the interfaces and TCP data along the lines of `iostat -x`.

```
% se nx.se 10
Current tcp RtoMin is 200, interval 10, start Thu Oct 16 16:52:33 1997
Name  Ipkt/s Opkt/s  Err/s  Coll% NoCP/s Defr/s  tcpIn  tcpOut Conn/s %Retran
hme0   212.0  426.9   0.00   0.00   0.00   0.00     65  593435   0.00    0.00
hme0   176.1  352.6   0.00   0.00   0.00   0.00     53  490379   0.00    0.00
```

11. Is there enough memory?

When an application starts up or grows or reads files, it takes memory from the free list. When the free list gets down to a few megabytes, the kernel decides which files and processes to steal memory from, to replenish the free list. It decides by scanning pages, looking for ones that haven't been used recently and paging out their contents so that the memory can be put on the free list. *If there is no scanning, then you definitely have enough memory.* If there is a lot of scanning and the swap disks are busy at the same time, you need more memory. If the swap disks are more than 50% busy, you should make swap files or partitions on other disks to spread the load and improve performance while waiting for more RAM to be delivered. You can use `vmstat` or `sar -g` to look at the paging system, or `virtual_adrian.se` will watch it for you, using the technique described in "RAM Rule" on page 456.

12. What changed recently and what is on the way?

It is always useful to know what was changed. You might have added a lot more users, or some event might have caused higher user activity than usual. You might have upgraded an application to add features or installed a newer version. Other systems may have been added to the network. Configuration changes or hardware "upgrades" can sometimes impact performance if they are not configured properly. You might have added a hardware RAID controller but forgotten to enable its nonvolatile RAM for fast write capability. It is also useful to know what might happen in the future. How much extra capacity might be needed for the next bunch of additional users or new applications?

Configuration and Tuning Recipes

The rest of this book gives you the information you need in order to understand a lot about how SPARC systems running Solaris work and about the basic principles involved in performance and tuning. Probably all you really want right now is to be told what to do or what to buy and how to set it up. This section just tells you *what* to do, with references to the rest of the book if you want to find out *why*. If you decide that you want to vary the recipes and do something different, you should really read the rest of the book first! For much more information on configuration issues, you should definitely get a copy of the book *Configuration and Capacity Planning for Solaris Servers*, by Brian Wong *(Sun Press)*.

The intention behind these recipes is to provide situation-oriented advice, which gathers together information about how to use a system rather than focusing on a particular subsystem in a generic sense.

Single-User Desktop Workstation Recipe

Local Disks and NFS Mounts

My recommended configuration is to NFS-mount home, mail, and applications programs directories from one or more workgroup servers. Configure a single, local disk to have two partitions, one for the operating system and one for swap space. It is easy to overload a single swap disk, so if you have more than one local disk, split any swap partitions or files evenly across all the disks (keep one clear for cachefs if necessary; see below).

Swap Space

Most application vendors can tell you how much swap space their application needs. If you have no idea how much swap space you will need, configure at least 128 Mbytes of virtual memory to start with. It's easy to add more later, so don't go overboard. With Solaris 2, the swap partition should be sized to top up the RAM size to get to the amount of virtual memory that you need[3]; e.g., 96-Mbyte swap with 32-Mbyte RAM, 64-Mbyte swap with 64-Mbyte RAM, no swap partition at all with 128 or more Mbytes of RAM. If your application vendor says a Solaris 2 application needs 64 Mbytes of RAM and 256 Mbytes of swap, this adds up to 320 Mbytes of virtual memory. You could configure 128 Mbytes of RAM and 192 Mbytes of swap instead. If you run out of swap space, make a swap file (I put them in `/swap`) or add more RAM. Older systems tend to run smaller

3. See "Virtual Memory Address Space Segments" on page 325 for a description of the unique Solaris 2 swap system.

applications, so they can get away with less virtual memory space. Later systems running the Common Desktop Environment (CDE) window system will need a lot more virtual memory space than systems running OpenWindows™.

File Systems and Upgrades

Make the rest of the disk into one big root partition that includes `/usr`, `/opt`, `/var`, and `/swap`. The main reason for doing this is to pool all the free space so that you can easily use `upgrade` or install to move up to the next OS release without running out of space in one of the partitions. It also prevents `/var` from overflowing and makes it easy to have a `/swap` directory to hold extra swap files if they are needed. In Solaris 2, `/tmp` uses the RAM-based `tmpfs` by default; the `mount /tmp` command should be uncommented in `/etc/rc.local` to enable it for SunOS 4.X.

Solaris 2 systems should be automatically installed from a JumpStart™ install server that includes a post-install script to set up all the local customizations. Since the disk can be restored by means of JumpStart and contains no local users files, it is never necessary to back it up over the network. A JumpStart install is much less frequent than a network backup, so its use aids good performance of the network. A useful tip to free up disk space for upgrades is to remove any swap files before you run the upgrade, then re-create the swap file afterwards.

```
# swap -d /swap/swapfile
# rm /swap/swapfile
comment out the swapfile in /etc/vfstab
shutdown and run the upgrade
# mkfile 100M /swap/swapfile
# swap -a /swap/swapfile
add the swapfile back into /etc/vfstab
```

Applications and Cachefs

If possible, NFS-mount the applications read-only to avoid the write-back of file access times, which is unwanted overhead. Configure the cache file system for all application code mount points. First, make `/cache`, then mount all the application directories, using the same cache. If you use large applications, check the application file sizes; if anything you access often is over 3 Mbytes, increase the *maxfilesize* parameter for the cache with `cfsadmin`.

Do not use cachefs for mail directories. It might be useful for home directories if most files are read rather than written—try it with and without to see. Cache loads when a large file is read for the first time can overload the disk. If there is more than one disk on

the system, then don't put any cache on the same disk as any swap space. Swap and cache are often both busy at the same time when a new application is started. The cache works best for files that don't change often and are read many times by the NFS client.

If your application is very data-intensive, reading and writing large files (as often occurs with EDA, MCAD, and Earth Resources applications), you are likely to need an FDDI or 100-Mbit Fast Ethernet interface. If large files are written out a lot, avoid cachefs for that file system. Figure 1-2 shows how to set up and use cachefs.

Figure 1-2 Setting up Cachefs and Checking for Large Files

```
# cfsadmin -c /cache
# find /net/apphost/export/appdir -size +3000k -ls
105849 3408 -rwxr-xr-x  1 root     bin       3474324 Mar  1 13:16
/net/apphost/export/appdir/SUNWwabi/bin/wabiprog
# cfsadmin -u -o maxfilesize=4 /cache
cfsadmin: list cache FS information
   maxblocks     90%
   minblocks      0%
   threshblocks  85%
   maxfiles      90%
   minfiles       0%
   threshfiles   85%
   maxfilesize    4MB
# mount -F cachefs -o backfstype=nfs,cachedir=/cache apphost:/export/appdir
/usr/appdir
```

Example Filesystem Table

The filesystem mount table shown in Figure 1-3 is for a system with a single local disk. Application program code is mounted read-only from *apphost*, using cachefs. Mail is mounted from *mailhost*. Home directories are automounted and so do not appear in this table. This system has a swap partition, and an additional swap file has been added. Direct automount mappings can be used to mount applications (including the cachefs options) and mail.

Figure 1-3 Sample `/etc/vfstab` for Workstation Recipe

#device #to mount	device to fsck	mount point	FS type	fsck pass	mount at boot	mount options
/proc	-	/proc	proc	-	no	-
fd	-	/dev/fd	fd	-	no	-
swap	-	/tmp	tmpfs	-	yes	-
/dev/dsk/c0t3d0s0	/dev/rdsk/c0t3d0s0	/	ufs	1	no	-
/dev/dsk/c0t3d0s1		-	swap	-	no	-

Figure 1-3 Sample `/etc/vfstab` *for Workstation Recipe*

#device #to mount	device to fsck	mount point	FS type	fsck pass	mount at boot	mount options
/swap/swapfile	-	-	swap	-	no	-
apphost:/usr/dist	/cache	/usr/dist	cachefs	3	yes	ro,backfstype=nfs,cachedir=/cache
mailhost:/var/mail	-	/var/mail	nfs	-	yes	rw,bg

Kernel Tuning

Since this setup is the most common, Solaris is already well tuned for it, so you don't need to set anything.

Workgroup Server Recipe

Workgroup servers provide reliable file storage and email and printer support at a departmental level. In a Unix environment, NFS file services are provided; in a PC environment, Microsoft SMB, Novell NetWare, and Apple file service protocols need to be supported, commonly using Syntax TotalNet or Samba to provide the service. It has also become common to provide web services on the home file server, so that it also acts as a home page server for the users and the projects associated with the department. Network Computers such as Sun's JavaStation™ need a boot server and backend application support that can be combined with the other functions of a workgroup server. If a sophisticated proxy caching web server hierarchy is implemented across the enterprise, then the workgroup server may also be the first-level cache.

This sounds like a complex mixture, but over the last few years, individual system performance has increased and several functions have coalesced into a single system. Solaris has several advantages over Windows NT-based workgroup servers in this respect. The stability and availability requirements of a server increase as more and more users and services depend upon it, and Solaris is far more robust than NT. Solaris can also be reconfigured and tuned without requiring rebooting, whereas almost every change to an NT server requires a reboot that adds to the downtime. NT has also picked up a bad reputation for poor performance when several workloads are run together on a system. Solaris not only time-shares many competing workloads effectively, it also scales up to far larger systems, so more work can be combined into a single system. Taking all this into account, a typical new Solaris workgroup server installation is likely to replace several diverse existing servers. Administrative and installation simplicity is important, the hardware can be configured to be resilient to common failures, and Solaris is well tested and reliable, so a single server is appropriate. The complexity and rigorous procedures needed to run a high-availability cluster will rule it out for all but the biggest installations.

Workload Mixture

File service is going to be the largest component of the workload. The basic Solaris email services are efficient enough to support many hundreds of users, and by the use of Solstice™ Internet Mail Server (SIMS), tens of thousands of users can be hosted on a single machine, so a workgroup environment will not stress the email system. The web server load will also be light, with occasional short bursts of activity.

NFS requests will always run in the kernel at a higher priority than that of user-level programs, so a saturated NFS server is unlikely to provide high performance for other uses. However, NFS is now so efficient and system performance is so high that it is, in practice, impossible to saturate even a small server. Serving PC-oriented protocols by using Samba is less efficient than NFS but is still comparable to a dedicated PC file server. A uniprocessor UltraSPARC™ server can saturate a 100-Mbit network with HTTP, NFS and/or SMB traffic if there is enough demand from the clients.

RAM Requirements

Dedicated file and mail servers do not need much RAM. The kernel will be configured to be a little larger than normal, but the main, active user program will be `sendmail`. The rest of RAM is a cache for the file systems. Unix-based NFS clients do a great deal of caching, so they don't usually ask for the same data more than once. Home directories and mail files are specific to a single user, so there will not often be multiple reads of the same data from different machines. Anything that is constantly reread from the server by multiple workstations is a prime candidate for setting up with cachefs on the clients.

NFS clients that are PCs running MS-Windows™ or MacOS™ put much less load on an NFS server than a Unix client does, but they require fast response times for short bursts of activity and do little caching themselves. It may be worth having extra memory on the server to try and cache requests to common files.

Allow 16 Mbytes of RAM for the kernel, printing, and other Solaris daemons, then 64 Mbytes for each fully loaded FDDI or 100-Mbit Ethernet being served.

Write Acceleration

The problem of making writes go fast on file servers merits special consideration. As an example, consider a single NFS write. The server must update the inode block containing the last modified time, the indirect block (and possibly a double indirect block), and the data block itself. The inodes and indirect blocks may not be near the data block, so three or four random seeks may be required on the same disk before the server can confirm that the data is safely written. To accelerate this process, all the filesystem metadata writes (inodes and indirect blocks) are quickly logged somewhere as a sequential stream.

The log is not read back very often because the original data is still in memory and it is eventually flushed to its proper location. If the system crashes, the data in memory is lost, and on reboot, the log is read to quickly restore the filesystem state.

There are several ways to implement the log.

- The oldest implementation is a nonvolatile memory board known as a Prestoserve™ that intercepts synchronous filesystem writes. These boards are still available for use in small systems, but the SBus device driver can only handle systems that have a single SBus, so large servers are not supported.
- The SPARCstation™ 10, 20, SPARCserver™ 1000, and SPARCcenter™ 2000 systems accept a special nonvolatile memory SIMM (NVSIMM) that uses a Prestoserve driver. A problem with this solution is that the data is stored inside a system, so if it is configured as a dual-failover, high-availability system, the logged data cannot be accessed when failover occurs. However, it is the fastest option for a single system.
- The current generation of Ultra Enterprise server products does not have an NVSIMM option, and all the newest machines have a PCIbus rather than an SBus. The log is stored in the disk subsystem so it can be shared in an HA setup. The disk subsystem could include a controller with its own NVRAM (remember to enable fast writes on the SPARCstorage™ Array and similar products), or at the low end, a dedicated log disk should be used with a product such as Solstice DiskSuite™. The log disk can be used to accelerate several file systems, but it should be completely dedicated to its job, with no distractions and a small partition (no more than 100 Mbytes) to make sure that the disk heads never have to seek more than a few cylinders.
- The current 7200rpm disks have very high sequential data rates of about 10 Mbytes/s and handle logging well. You just have to ignore the unused few gigabytes of data space: *don't make a file system on it*.

The recommended disk configuration is to have a single, large, root partition like the workstation recipe with the exception of the `/var` directories. `/var/mail` should be on a separate disk partition because it needs to be accelerated. Mail programs on the NFS clients rewrite the entire mail file when the user saves changes; the time this rewrite takes is very noticeable, and a log disk or Prestoserve speeds it up a great deal. The `/var` file system should be big enough to hold a lot of mail (several megabytes per user account). You will need space in `/var/spool` for outgoing mail and printer jobs (at least 10 or 20 megabytes, sometimes much more). Home directories should be striped over several disks and configured with a log disk.

Network Configurations

A typical setup for a commercial workgroup is based on a 100-Mbit connection from a server to a network switch that feeds multiple, independent 10-Mbit networks with client PCs or Network Computers. Engineering workstations should be using 100-Mbit client connections in a switched full-duplex network infrastructure. Don't skimp on network bandwidth if you care about performance—an Ultra 1/170 is fast enough to saturate a 100-Mbit connection. The server may need to use multiple 100-Mbit networks to feed all its clients. High bandwidth connections may be done by trunking, where the Quadruple Fast Ethernet (QFE) card aggregates its ports to behave like a 400-Mbit duplex connection.

An upcoming option is gigabit Ethernet. Sun's initial product is a switch that takes a gigabit connection from a server and feeds multiple switched 100-Mbit ports to its clients. Some high-end sites are using 155-Mbit ATM cards in the client systems and 622-Mbit cards from the server to an ATM switch, or back-to-back ATM622 cards to provide a high bandwidth link between a pair of servers. A benefit of ATM is that the default maximum IP packet size is 9 Kbytes when used as a LAN. This is far more efficient that the 1500 byte packets used by ethernet, but is only effective if there is an ATM connection all the way from the server to the clients.

CPU Loading for NFS Servers

A SuperSPARC™ can handle four or five 10-Mbit Ethernets. Each fully loaded 100-Mbit Ethernet or FDDI should have two SuperSPARC processors or one UltraSPARC to handle the network, NFS protocol, and disk I/O load.

Disk Configurations

Since an NFS lookup or read can involve two trips over the network from the client as well as a disk I/O, getting good perceived performance from the server requires a low latency disk subsystem that averages better than 40 ms service time. Use Solstice DiskSuite or SPARCstorage Manager (VxVM) to stripe file systems so that the load is evenly balanced across as many independent disks as possible. You will get six times better performance from a stripe of six 4.3-Gbyte disks than you will get from one 23-Gbyte disk. For good performance, configure four to six disks in the stripe for each loaded 100-Mbit network. The data-intensive clients that need faster networks tend to do more sequential accesses and so get more throughput from the disks than the typical random access load. The logging file system supported by Solstice DiskSuite is especially useful with multi-gigabyte file systems because it avoids a time-consuming, full filesystem check.

Setting the Number of NFS Threads

In SunOS™ 4.X, each NFS daemon appears as a separate process, although the NFS daemons do all their work in the kernel. In Solaris 2, a single NFS daemon process and a number of kernel threads do basically the same job. Configure two threads per active client machine, or 32 per Ethernet. The default of 16 is suitable only for casual NFS use, and there is little overhead from having several hundred threads, even on a low-end server. Since kernel threads all use the same context, there is little thread switch overhead in either SunOS 4 or Solaris 2.

For example, a server with two Ethernets running SunOS 4 would need `nfsd 64` to be set in `/etc/rc.local` and, when running Solaris 2, would need `/usr/lib/nfs/nfsd -a 64` to be set in the file `/etc/init.d/nfs.server` (which is hardlinked to `/etc/rc3.d/S15nfs.server`).

Kernel Tuning

NFS servers don't often need a lot of RAM but do need large, name lookup caches, which are sized automatically, based on the RAM size in Solaris 2. The two main changes recommended are to make the inode and directory name lookup caches have at least 8,000 entries. See "Vnodes, Inodes, and Rnodes" on page 360 for more details.

The default size of both caches in Solaris 2 will be (RAM-2)*17+90. For a 64-Mbyte system, this size works out at 1144. If you have more than 512 Mbytes of RAM, the cache is big enough (see Figure 1-4).

Figure 1-4 Sample `/etc/system` Kernel Tuning Parameters for a 128-Mbyte Solaris 2 NFS Server

```
set ncsize=8000           for NFS servers with under 512 MB RAM
set ufs:ufs_ninode=8000   for NFS servers with under 512 MB RAM
```

Database Server Recipe

I am not a database specialist. You should refer to the many good books on database performance tuning for each of the main database vendors. Here, I'll give some general recommendations on system considerations that apply to most databases.

Workload Mixture

System performance scales to much higher levels than most databases require. Whereas in the past we would recommend that a separate system be used for each database, it is now more common to consolidate several workloads onto a single, large system. With a greater dependency on a smaller number of systems, configuring for high availability is more important. Failover and parallel database clusters are becoming much more common.

The two main categories of database usage are online transaction processing, such as order entry typified by the TPC-C benchmark, and complex analytical queries made against a data warehouse, as typified by the TPC-D benchmark. Take the time to read the detailed reports of tested configurations that are available on the TPC web site at http://www.tpc.org. A lot of effort goes into setting up very high performance configurations for those tests, and you can copy some of the techniques yourself.

RAM Requirements

Database servers need a lot of RAM. Each database and application vendor should provide detailed guidelines on how to configure the system. If you have no other guidance, I will suggest a starting point that has been used with Oracle®. For the database back end, allow 64 Mbytes of RAM for the kernel, Solaris daemons, and database backend processes. Then, allow another 2 Mbytes for each user if time-shared, or 512 Kbytes per user on a pure back end. Finally, add memory for the shared global area.

Log -Based and Direct I/O File Systems

Use the log-based UFS file system supported by Solstice DiskSuite if your database tables must be stored in UFS file systems rather than on raw disk. Oracle, Informix, and Sybase work best with raw disk, but other databases such as Ingres and Progress have to use file systems. With Solaris 2.6 there is a direct I/O option that can be used to provide raw access to a file in the file system. This option is not as good as raw but can greatly reduce memory demands.

Network Loading for Database Servers

The network activity caused by SQL is so application dependent that it is hard to give general guidelines. You will need to do some work with the snoop command or a network analyzer on a live system to work out the load. Some applications may work well over a dial-up modem, whereas others will need 100-Mbit networks or may only work well when the client and server are on the same machine.

CPU Loading

CPU loading cannot be estimated easily; any guidelines provided in this book would lead to a mixture of overconfigured and underconfigured systems. Database sizing involves too many variables and is outside the scope of this book. I recommend that you read Brian Wong's book *Configuration and Capacity Planning for Solaris Servers*, which covers the subject in depth. Sizing data for common application and database combinations is being

generated within Sun for use by systems engineers and resellers. Database vendors often place restrictions on the publication of performance-related information about their products.

Disk Configurations

Getting good, perceived performance from the server requires a low-latency disk subsystem. For good write latency, NVRAM is normally configured in the disk controller subsystem. Use Solstice DiskSuite, SPARCstorage Manager or a hardware RAID system such as Sun's RSM2000 to stripe file systems so that the load is evenly balanced across as many independent disks as possible. You will get six times better performance from a stripe of six 4.3-Gbyte disks than you will from one 23-Gbyte disk. Extremely large disk configurations are now common, and several thousand disks may be connected to a single system. Make sure you are familiar with the new `iostat` options in Solaris 2.6, described in "Output Formats and Options for `iostat`" on page 183. You can now get an inventory of the disk configuration and look for error counts on a drive-by-drive basis.

Setting the Shared Memory Size

Shared memory size is often set too small. On a database using raw disks, the size needs to be set higher than on a system with UFS. The effect of UFS is to provide additional data buffering and a duplicate copy of the data in shared memory. This improves read performance and can help sequential table scan rates when UFS prefetching is more aggressive than the database's own prefetching. The drawback is that more RAM and CPU time is used. When running raw, you can choose to run with less caching and use less RAM, or you can go for higher performance by using a much bigger shared memory area and similar total RAM usage. As a first approximation, use half of your total main memory as shared memory, then measure the database to see if it is oversized or too small and adjust as necessary.

Kernel Tuning

Databases tend to use lots of shared memory and semaphore settings. These do not affect performance; as long as shared memory and semaphore settings are big enough, the programs will run. Each database vendor supplies its own guidelines. See "`tune_t_fsflushr` and `autoup`" on page 339 for advice on tuning the `fsflush` daemon. Figure 1-5 presents an example.

Figure 1-5 Example `/etc/system` Entries for a Database Server

```
 * example shared memory settings needed for database
set shmsys:shminfo_shmmax=268435456
set shmsys:shminfo_shmmni=512
set shmsys:shminfo_shmseg=150
```

```
set semsys:seminfo_semmap=350
set semsys:seminfo_semmni=350
set semsys:seminfo_semmns=1000
set semsys:seminfo_semmnu=700
set semsys:seminfo_semume=100
* keep fsflush from hogging a CPU
set autoup=240
```

Multiuser Server with ASCII or X Terminals Recipe

There is little difference in kind between dumb ASCII terminals, proprietary graphics terminals connected over serial ports, IBM 3270 terminals connected over SNA, and X terminals. The terminal understands a fixed, low-level display protocol and has varying amounts of built-in functionality, but all application processing is done on time-shared multiuser servers.

What's the Difference between Client/Server and Time-shared Configurations?

The term *client/server* is sometimes used to describe a time-shared system with X terminals. Personally, I don't like this use of the term because I think it is misleading. The primary extra capability of an X terminal over other types of terminals is that it can make direct connections to many servers at the same time. As an example, consider upgrading a time-shared server by replacing ASCII terminals with X terminals running the same application in a terminal emulator window. There is still no separate client processing going on. An upgrade to client/server would be to have users running part or all of the application on a SPARCstation or PC on their desk, with an application-specific protocol linking them to a database or an NFS server back end.

Performance Is Usually a Problem on Time-shared Systems

From a performance point of view, time-shared systems are usually a source of problems. Part of the problem is that Unix assumes that its users will be well behaved and has few ways to deal with users or programs that intentionally or accidentally take an unfair share of the system. It is sometimes known as a *Denial-of-Service Attack* when a user tries to consume all the CPU, RAM, disk space[4], swap space, or overflow kernel tables. Even unintentional overload can cause serious problems. Instead, if you can configure a client/server system where users get their own SPARCstation or PC to use and abuse, then it is much harder for one user to affect the performance for the rest of the user community.

4. If this is a problem, then you can use the standard BSD Unix disk quotas system in Solaris.

The Softway ShareII resource management system has been ported to Solaris 2. This product was developed to solve the problem by allocating fair shares of system resources to users and groups of users. See http://www.softway.com.au for more details.

Another problem occurs when applications that were developed for use on high-end SPARCstations are installed on a server and used via X terminals. If the application runs well on a low-powered machine like a SPARCstation 5, then sharing a more powerful machine makes sense. If the application is normally used on an Ultra 2, then don't expect many copies to run simultaneously on an X-terminal server.

X terminals work very well in a general office environment where most users spend a small proportion of their time actually working at the X terminal. X terminals don't work well if all the users are active all the time. Try to avoid configuring data entry or telephone sales sweatshops or student classrooms full of X terminals; the backend system needed to support them will often be large and expensive or seriously underpowered. Some software licensing practices can make a single, large system with X terminals cheaper than lots of smaller systems. Hopefully, more software vendors will convert to floating per-user licenses for software.

There is a large movement in the industry to replace all kinds of terminals with Network Computers running Java, so old-style terminals are beginning to die out.

Internet and Java Server Recipes

The whole subject of Internet and intranet web servers, proxy caching web servers, and servers for Java-based applications is covered in detail in Chapter 4, "Internet Servers," and Chapter 5, "Java Application Servers." The simplest sizing recipe is based on two key things. The first is that the minimum baseline for good web server performance is Solaris 2.6 and an up-to-date, efficient web server. The second is that given efficient software, it will be easy to saturate your network, so base all your system sizing on how much network traffic your switches, backbones, and wide-area links can handle.