Tuning the SPARCserver 490 for Optimal NFS Performance

Varun Mehta
Rajiv Khemani

Server Performance Engineering

Sun Microsystems, Inc.

February 7, 1991

1. Introduction

The SPARCserver 490 is Sun's premiere computer system for resource-intensive file server, database management, multi-user, and compute server applications. Like all of Sun's systems, it is available in a wide variety of configurations. Its system software allows many management and tuning strategies.

However, this extraordinary flexibility can be daunting to the prospective or current owner of a SPARCserver 490. The purpose of this paper is to address the usage of the SPARCserver 490 as a file server, by providing the reader with configurations and strategies for achieving maximum performance at several system sizes. The dominant factor in the level of performance of a file server is its ability to handle Network File System (NFS) operations initiated by its client workstations. It must be able to handle the number of operations initiated by the clients, and should process these requests in a timely fashion. When properly configured and managed, the SPARCserver 490 provides powerful and flexible NFS support for large groups of workstations and PCs.

Section 2 of this paper details five specific strategies for increasing the NFS performance of a SPARCserver 490. The effectiveness of these strategies is shown by extensive test results. Section 3 of this paper describes several recommended configurations of the SPARCserver 490, designed to satisfy different levels of customer needs with optimal price/performance. The smallest configuration described can be evolved smoothly into the largest configuration, which allows the customer to preserve investment in hardware and software as user needs grow.

2. Five Steps to Improving NFS Performance

This section describes software tuning actions and optional hardware that can be used to maximize the NFS performance of the SPARCserver 490. Proper software tuning can make a significant difference in the performance of a file server, and users are urged to follow the easy steps described. Optional hardware can provide marked increases in NFS performance at a fraction of the cost of an additional server system, or even an additional IPI disk drive.

In summary, the five steps to be described in this section are:

1. Increasing the *maxusers* SunOS kernel value.
2. Increasing the size of the SunOS buffer cache.
3. Adding a Prestoserve NFS Accelerator[1].
4. Adding one or more Interphase NC 400 Ethernet Co-processors[2].
5. Balancing the loads on disks and networks.

NFS requests are translated on the server to file system calls. Most of these calls result in disk accesses which tend to be randomly spread out on the disk. Contemporary disks are limited to 40 - 50 random I/O's per second (IOPS). Because of this, many NFS servers bottleneck on their disks. One easy solution is to add more disks, but the first three steps listed above are a cheaper way of reducing disk I/O (and in some cases CPU load). Step 4, above, reduces CPU requirements on the server and boosts network processing speed. The final step explains how to use all the I/O resources optimally.

To show the effectiveness of these steps, results from the *nhfsstone* benchmark version 2.0.3 were used. Nhfsstone is a benchmark program that runs on one or more client workstations, and generates NFS requests of various types to the file server. The program allows different rates of NFS operations (NFSops) to be generated, and measures the responsiveness of the server in terms of milliseconds per operation call (abbreviated as ms/call). Nhfsstone also generates a variety of NFS requests -- reads, writes, get attributes, lookups, etc. -- that are indicative of the mix of operations in a typical environment. (The nhfsstone user can vary this mix, but all tests described in this paper use the default mix provided by the program, known as the "standard Legato mix".) See Appendix IV for additional details about nhfsstone.

Like any benchmark, nhfsstone results are only meaningful when compared to other nhfsstone results obtained under similar conditions. Furthermore, it is important to realize that, although the mix of NFS operations used by nhfsstone are typical of real environments, they are generated by the clients at a much greater rate than in a typical environment. The tests described here used two or three clients per network to run nhfsstone. Obviously, a SPARCserver 490 can support vastly more than three clients per net in normal operations, because clients are occupied with many other operations besides generating NFS requests. A typical client generates about 5-20 NFS operations per second during active use. Of course, it will generate much less than this when a user is editing a section of a file, or contemplating the next action to take.

[1] The Prestoserve is available from Sun Microsystems as order number X165A.

[2] The NC 400 is available from Interphase Corporation, 381 E. Evelyn Ave., Mountain View, CA 94041. Phone 415-966-1024.

2.1. Increasing the Value of *maxusers*

Summary

Increasing the *maxusers* value increases the size of various caches in the kernel and improves system performance. For the SS-490 in an NFS server configuration, this value should be increased to 64 for machines with 4 or fewer disks and less than 100 simultaneous processes. For larger workloads *maxusers* should be set to 128.

The value of the *maxusers* field in the SunOS kernel configuration file determines the size of various kernel data structures. Some of the important kernel variables affected by changing *maxusers* are:

nproc The size of the process table.
ninode The maximum size of the inode cache.
ncsize The size of the directory name cache.
nfile Maximum number of open files.
ndquot For maintaining disk quotas.

Maxusers default value for generic SunOS 4.1PSR_A is 8. The value of this field can be changed by reconfiguring SunOS.[3] If sufficient physical memory is available (32MB or more on current server machines), increasing *maxusers* greatly improves certain cache hit rates, which provides significantly better NFS performance.

Table 1 below provides an example of increased NFS performance resulting from an increase in the *maxusers* value. It shows nhfsstone benchmark performance of a SPARCserver 490 with two NC 400 Ethernet interfaces; three clients on each net were used to generate NFS calls using the nhfsstone program. Increasing *maxusers* from 16 to 128 improved nhfsstone performance from 471 NFSops/sec @ 48 msec/call to 527 NFSops/sec @ 43 msec/call. Note that the size of the buffer cache had been increased for this experiment, as discussed in Section 2.2 below. (A Prestoserve board was not used in this test.)

Table 1. Effect of Changing *maxusers* on Nhfsstone Performance

Statistic		MAXUSERS=16	MAXUSERS=128
Dir Name Cache Statistics	Dir cache hits/s	65	168
	% Dir cache hits/s	42%	98%
	Enters into cache/s	97.5	12.6
CPU & Disk Statistics	CPU busy	81%	60%
	Disk busy	85%	75%
	Avg disk xfers/s	41.4	36.3
	Avg disk read/s	9.5	0.3
Nhfsstone Statistics	Max NFSops/sec	471	527
	NFS ms/call	48	43

As can be seen from the table, the larger value of *maxusers* resulted in dramatically lower usage of CPU and disk resources to handle the test load. This was mainly due to much higher *directory name cache* hit rates when *maxusers* was increased. The size of

[3] Chapter 9 of Sun's *System & Network Administration* manual describes kernel configuration procedures.

the directory name cache is calculated as $17*maxusers+90$ at the time of reconfiguring the kernel. A smaller directory name cache means that cache misses will occur frequently, requiring the CPU to spend more time in pathname translation and go to disk frequently. This results in greater disk and CPU usage.

The directory name cache hit rate since the time of last system boot is displayed in the *total name lookups* field using *vmstat -s*. Unfortunately, there is no comprehensive tool that provides all cache statistics in SunOS. However, disk and CPU utilization should be monitored using *iostat* and *vmstat*. Any unusually heavy utilization could be the result of bottlenecking on an OS resource.

2.2. Increasing the Size of the Buffer Cache

Summary

> A small buffer cache increases disk utilization and can saturate disks. The default buffer size on most servers is generally too low. It is recommended that the buffer cache size be increased to 64 if the server has four or fewer disks or to 112 if there are more than four disks.

The buffer cache is a part of SunOS that caches certain data, primarily file inodes and disk cylinder groups. The purpose of the cache is to reduce the frequency of disk accesses. The cache is allocated at boot time, and can expand up to a fixed ceiling or decrease if extra buffers are not required.

For practical purposes the maximum size of the buffer cache is related to a fixed number, minus the size of the kernel at a certain point during boot. Thus, as the kernel size increases (perhaps due to a larger *maxusers* or configuring in new device drivers) the maximum size of the buffer cache falls.

A decrease in the size of the buffer cache leads to increased disk activity, which causes disks to saturate relatively quickly and limit overall performance. Ideally, the size of the buffer cache should be proportional to the amount of physical memory; this was originally the case. But as physical memory on a typical machine surged in recent years, a ceiling based on a predefined kernel constant limited further increases in the buffer cache size. This has led to smaller than desirable buffer caches (especially on fully configured servers). Future SunOS releases will be modified to correct this problem. Fortunately, it is a simple task to increase the buffer cache size on existing kernels. The 3 line script in Appendix II may be used to patch the OS to increase the size of the buffer cache.

To show the effect on NFS performance of increased buffer cache size, nhfsstone was run on a SPARCserver 490 configured with two NC 400 Ethernet boards. (The Prestoserve board was not used in this demonstration, to accentuate the differences due to disk saturation; the *maxusers* value had been increased to 128 per Section 2.1.)

Table 2. Effect of Increased Buffer Cache Size on Nhfsstone Performance

Statistic		Small Buffer Cache	Large Buffer Cache
Buffer Cache Statistics	Buf Cache Size	27	64
	Buffer Hits	49%	94%
	Buffer reads/s	192	231
Disk Statistics	Disk busy	90%	76%
	Avg disk xfers/s	46.7	36.7
	Avg disk read/s	16.2	2.1
Nhfsstone Statistics	Max NFSops/sec	390	521
	NFS ms/call	60	44

Table 2 shows that increasing the size of the buffer cache from the default 27 buffers to 64 buffers resulted in a striking 34% improvement in NFSops/sec performance and a 27% faster response time. The buffer cache hits almost doubled, while the average disk reads/sec and the disk transfers/sec plummeted.

The size of the buffer cache should be tuned according to the number of disk inode structures accessed per second. In practice, this is difficult to measure. As a coarse rule of thumb we recommend that servers with up to 4 heavily utilized disks, have a buffer cache size of 64. For machines with more disks, the buffer cache size should be set to an upper limit of 112.

It is a good idea to monitor buffer cache statistics on a server. The kernel maintains these statistics internally. The second script in Appendix II shows how to output buffer cache statistics using *adb*.

2.3. Adding a Prestoserve NFS Accelerator

Summary

The Prestoserve Accelerator caches synchronous writes to disks. This dramatically reduces disk activity and the number of disks required to achieve excellent NFS throughput with unrivaled response times. A Prestoserve board helps on all but the very largest configuration where the server load exceeds 500 NFSops.

2.3.1. Prestoserve Description and Performance

The VMEbus-based Prestoserve NFS Accelerator significantly improves NFS write performance and random filesystem access performance on the SPARCserver 490. To achieve this, it uses 1 MB of non-volatile memory (NVRAM) as a write cache for synchronous disk I/O. (Prestoserve's data structures are also stored in non-volatile memory, to prevent any possibility of data loss in case of a system crash.) The NVRAM is interposed between the operating system and the disks.

User-selected filesystems are accelerated by the Prestoserve device driver, which intercepts synchronous write requests to disks and stores the data in the NVRAM. The Prestoserve cache is then asynchronously flushed to disks in groups large enough for the disk drivers to optimize the order of writes, thus increasing disk efficiency. Further, since the NFS protocol requires writes to be synchronous on the server, Prestoserve improves NFS write performance by allowing server acknowledgement as soon as data is written to the Prestoserve cache.

Figure 1. Effect of the Prestoserve Accelerator on Nhfsstone Performance

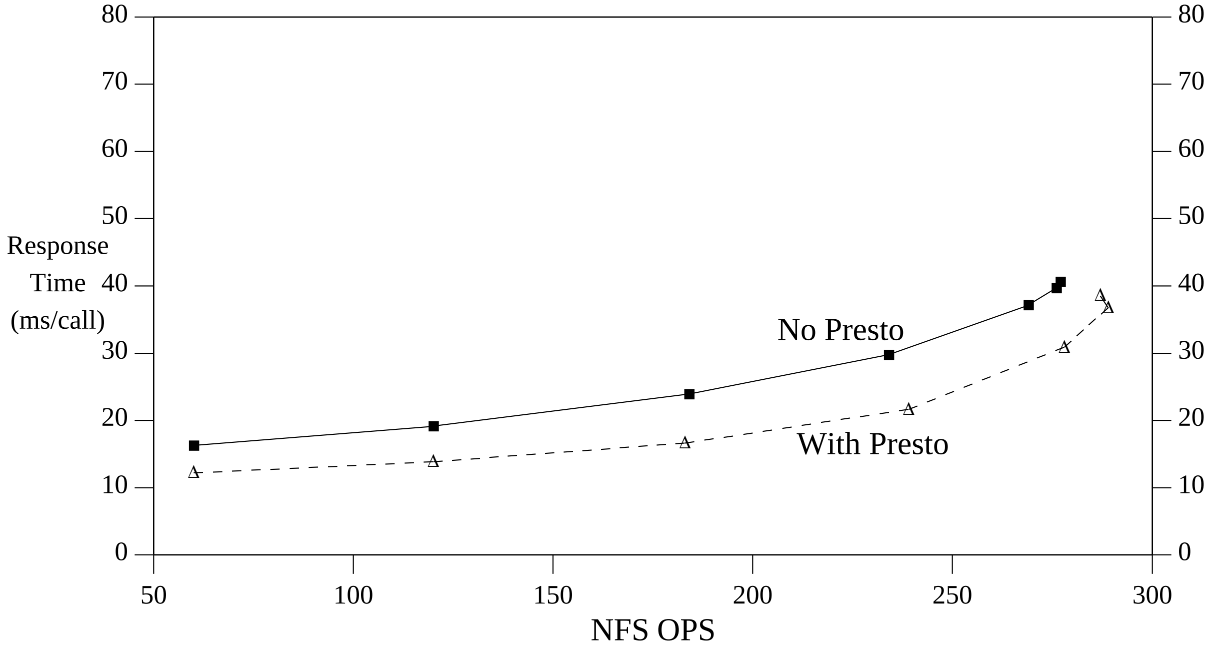
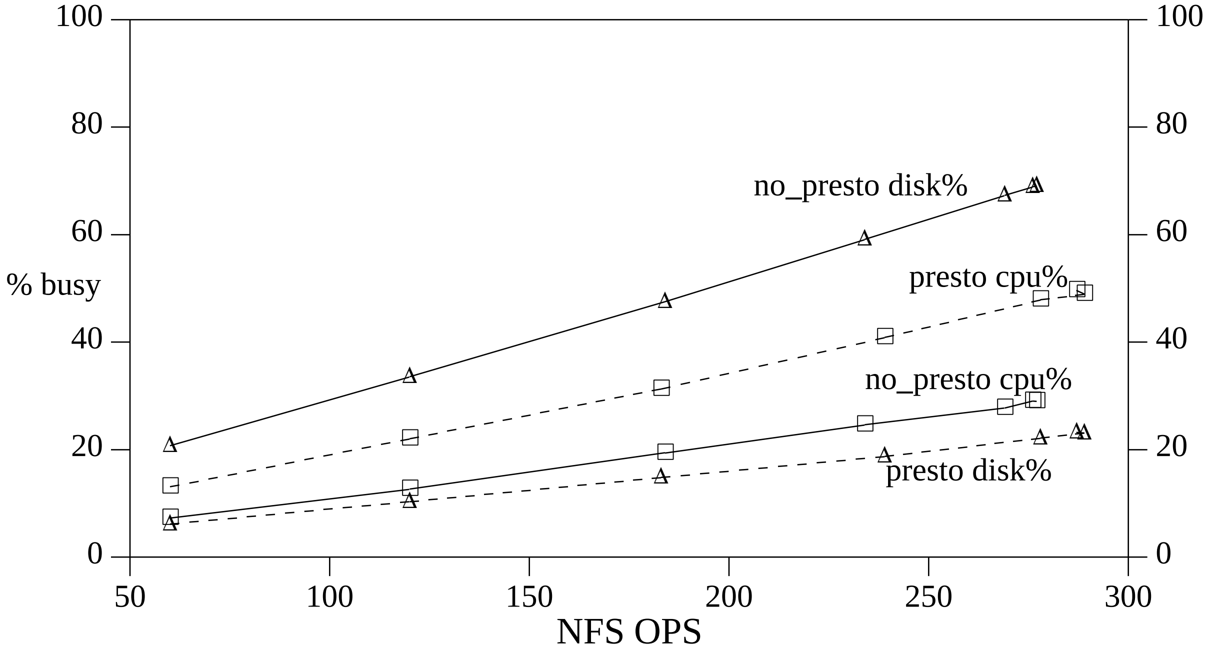


Figure 1 shows the performance of the SPARCserver 490 on the nhfsstone benchmark with and without the Prestoserve Accelerator. A single NC 400 Ethernet interface was used; three clients were used to generate the NFS calls. *maxusers* was set to 128 and the size of the buffer cache was increased, as described in the previous two sections. 277 NFSops/sec @ 41 msec/call were achieved without Prestoserve; this improved to 287 NFSops/sec @ 39 msec/call with a Prestoserve. NFS response time also improved by about 25% with the Prestoserve, at a given level of NFSops load. More significantly, disk utilization using Prestoserve decreased to 23% from 69%, as shown in Figure 2. This much lower disk utilization indicates that a server configuration with a particular amount of disk capacity will be able to adequately support more clients with a Prestoserve than without. This is particularly true if multiple Ethernets are utilized.

2.3.2. Other Performance Issues with Prestoserve

Although the addition of a Prestoserve significantly reduces disk utilization, CPU utilization often increases (see Figure 2). This extra CPU overhead is due to the copying between main memory and the Prestoserve's onboard memory. Some disks suffer a greater CPU penalty than others. This is because Prestoserve's write cache is on the VMEbus and some disk controllers (and most SCSI onboard devices) cannot physically access the VMEbus memory. To handle such devices, Prestoserve copies data buffers from its cache back to kernel memory before transferring them to the disk driver. This mode is called *BOUNCEIO* and involves extra CPU overhead; it also increases physical memory requirements by up to 128 Kbytes. By default, this mode is disabled for Sun IPI disks, on other disks *BOUNCEIO* mode should be disabled if possible because of its performance implications.

Figure 2. Effect of the Prestoserve Accelerator on Disk and CPU Usage



It is also important to realize that, although addition of a Prestoserve generally provides increased client support, this increased support does not apply to *extremely large* SPARCserver 490 configurations. In such a configuration, with loads of more than 500 NFSops/sec the stream of writes to the Prestoserve cache can become the system bottleneck. In such a configuration, a Prestoserve is not recommended. However, the higher disk utilization that will occur without the Prestoserve makes it necessary to add enough disks to support the random I/O needs of all the clients. Naturally, the load must be divided equally among all the disks in the system.
(See Section 2.5 for a discussion of load balancing, and Section 3.2 for examples of similar configurations with and without a Prestoserve.)

2.4. Adding an Interphase NC 400 Network Co-processor

Summary

The Interphase NC 400 Network co-processor and associated software improve NFS performance by offloading network processing from the host CPU. The NC 400 allows an SS-490 to provide fast, scalable and low overhead NFS performance.

The NC 400 Network Co-processor and its associated software can dramatically improve the NFS performance of a SPARCserver 490, relative to its performance when using the onboard Ethernet controller of the 490[4]. It does this by offloading the lower layers of network processing (i.e., IP, UDP, XDR, RPC processing and NFS decode operations) to the specialized NC 400 Ethernet hardware. The NC 400 co-processor communicates with an NC400-specific nfsd daemon process which resides in Sun host memory. This daemon is optimized for efficient network buffer management and low context switching rates. The overall effect of the NC 400 hardware and software combination is vastly improved NFS throughput, with much lower CPU utilization.

Figure 3. Onboard (ie) Ethernet vs. NC 400 (ne) Ethernet Nhfsstone Performance

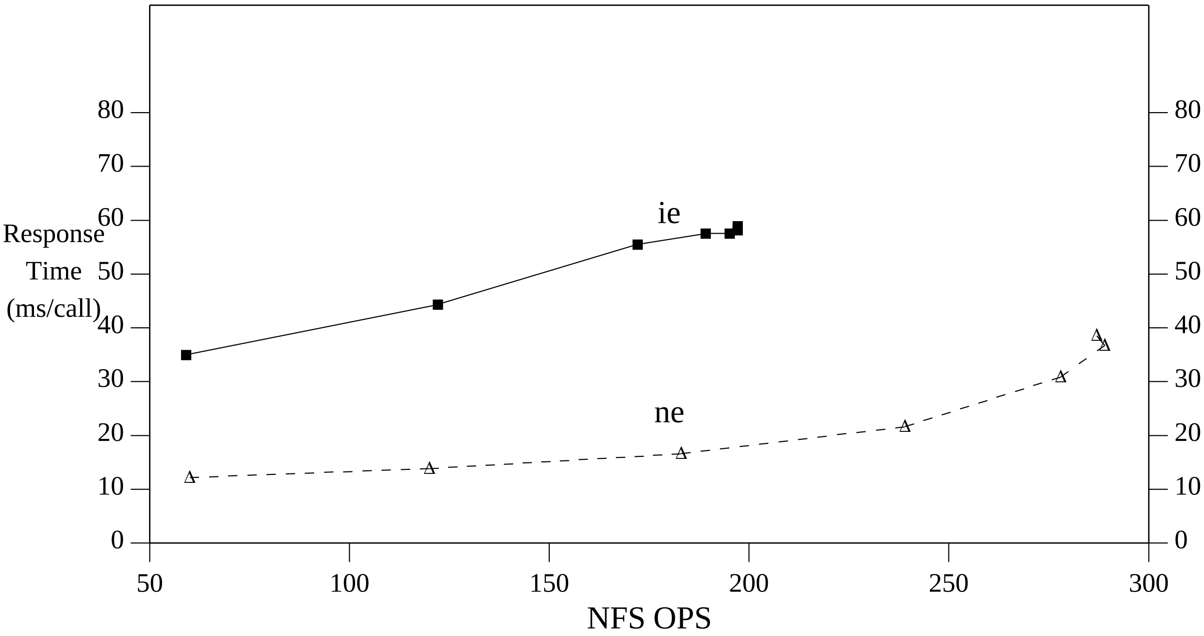


Figure 3 above shows the performance of the NC 400 and onboard Ethernet interfaces on the nhfsstone benchmark. A single Ethernet interface was used for each test and a Prestoserve board was operational; the buffer cache size was increased to 112, *maxusers* was set to 128, and three clients were used to generate the NFS load. Note that the onboard Ethernet interface of the SPARCserver 490 (device name *ie*) provided up to 200 NFSops/sec @ 59 ms/call on the nhfsstone benchmark. By contrast, the NC 400 Ethernet interface (device name *ne*) achieved 290 NFSops/sec @ 37 ms/call. At 290 NFSops/sec the Ethernet media has saturated, so further performance gains can not be achieved without adding a second network interface.

Another significant benefit of the NC 400 Ethernet interface is shown by Figure 4 below, which graphs the CPU and disk resources used by each of the two interfaces. The ie interface uses about twice as many CPU cycles as the ne to achieve equivalent nhfsstone performance. This is partly because most network processing is done on the NC 400 board rather than on the CPU, and partly because of lower context switching on the host CPU when using the NC 400. The ne interface reduces the rate of context switches by an order of magnitude. This occurs because of two reasons:

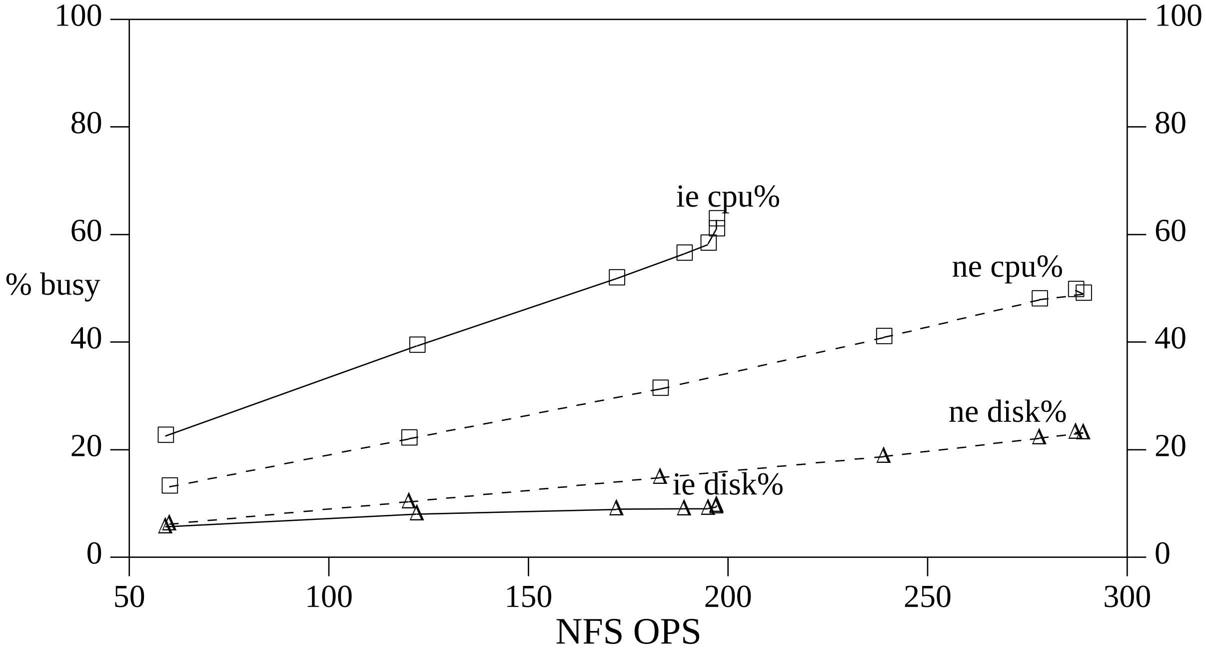
a) On the ie interface, the CPU must be interrupted every time an incoming packet causes an mbuf to be filled. A single 8KB NFS write request requires 6 Ethernet packets. The ie interface interrupts the CPU and causes a context switch 6 times for

Note: Because this report's focus is providing maximum NFS performance, we chose to consider the Interphase NC 400 Ethernet interface rather than the Sun 452A SunNet Ethernet/VME controller. The 452A is substantially less expensive than the NC 400, but it does not offload network processing as does the NC 400. The Sun 452A is best used when there is a need to provide system access to multiple networks for gateway functions, rather than to provide the highest performance interface in a multi-network file server.

each of the incoming packets. On the ne interface, incoming packets are buffered in internal memory until the complete NFS request has been received. A single interrupt and DMA transfer then move the data to the host memory. Transferring the data to the host in a single chunk also avoids subsequent CPU copying when non-contiguous buffer chains have to be concatenated prior to further processing.

b) The NC 400's nfsd, when finished processing an NFS request, checks the request queue instead of sleeping. If a new request is available, it is immediately dequeued and processed. This reduces context switching by not having to wake up a sleeping nfsd to process the new request. (The standard nfsd always sleeps after processing an NFS request.)

Figure 4. Onboard (ie) Ethernet vs. NC 400 (ne) Ethernet Disk and CPU Usage



When considering the benefits of the NC 400 Ethernet, it is important to remember that network processing is only part of what an NFS server must do. All NFS requests are ultimately translated to file system requests. By taking over a substantial portion of the network processing, an NC 400 co-processor improves NFS performance and also frees host CPU cycles to do more file system processing. This in turn means that the CPU can support more NFS clients.

2.5. Load Balancing

Summary

It is important that the load be equally balanced among all disks, controllers and networks in a system. A disk is heavily utilized if it is more than 75% busy or providing in excess of 40 IOPS. In our test environment we found that the Ethernet media is heavily utilized when its collision rate is over 8 packets/sec and I/O rate exceeds 600 packets/sec. If either the disks or ethernets approach saturation after load balancing, more resources should be added.

On any system with replicated resources, it is essential to balance the load among those resources to achieve maximum performance. This is particularly true of disks and disk controllers, but it also applies to network interfaces.

The *iostat* program provides statistics for up to 10 disk drives on a system; these statistics can be used to properly balance disk loads. NFS servers generally do random rather than sequential file accesses; therefore, it is important that the tps (transfers per second) field in *iostat* output for every disk be as similar as possible. A good way to monitor disk activity is to run *iostat* with an interval of about two minutes during peak working hours (e.g., from 10am to 4pm). A simple *awk* script can be used with the *iostat* output to provide hourly and daily averages of the workload per disk.

The system administrator should use this data to balance disk loads. *An IPI disk is heavily loaded when it provides in excess of 40 transfers/sec during normal NFS server operation*. If some disks in a system exhibit this heavy loading while others do not, files should be moved from the loaded disks to the underutilized disks to provide better system performance. When *all* the disks on a system approach the heavy loading level, an administrator should consider either adding another disk or installing a Prestoserve board. Since the Prestoserve requires some CPU bandwidth, an administrator should only consider using it if the CPU is less than 60% busy on average. (CPU utilization numbers can also be derived from *iostat* output.)

To study network interface loading, the command *netstat -i* can be used to get statistics about each interface since the last system boot. On the nhfsstone benchmark in our test environment, we found that the Ethernet interface sustains about 400 I/O packets per second with less than 1 packet collision/sec. This increased to a maximum of a little over 700 I/O packets/sec, but at that point the net sustained about 10 collisions/sec.

The network interfaces can be monitored in the same way as the disks by using a combination of *netstat -i* and *awk*. If the hourly averages show an Ethernet sustaining frequent collisions (> 8/sec) or an I/O rate of more than 600 packets/sec, some of the network load should be moved to a less lightly loaded Ethernet interface, or a new interface should be added to the system.

Network media utilization depends upon a number of factors such as the packet size distribution, length and topography of the network, number of hosts, and type of network protocol. A high I/O packet rate need not mean that the media is saturated, because the packets may be small. The recommendation above is based only upon our experience with the nhfsstone benchmark running in our test environment.

Routing

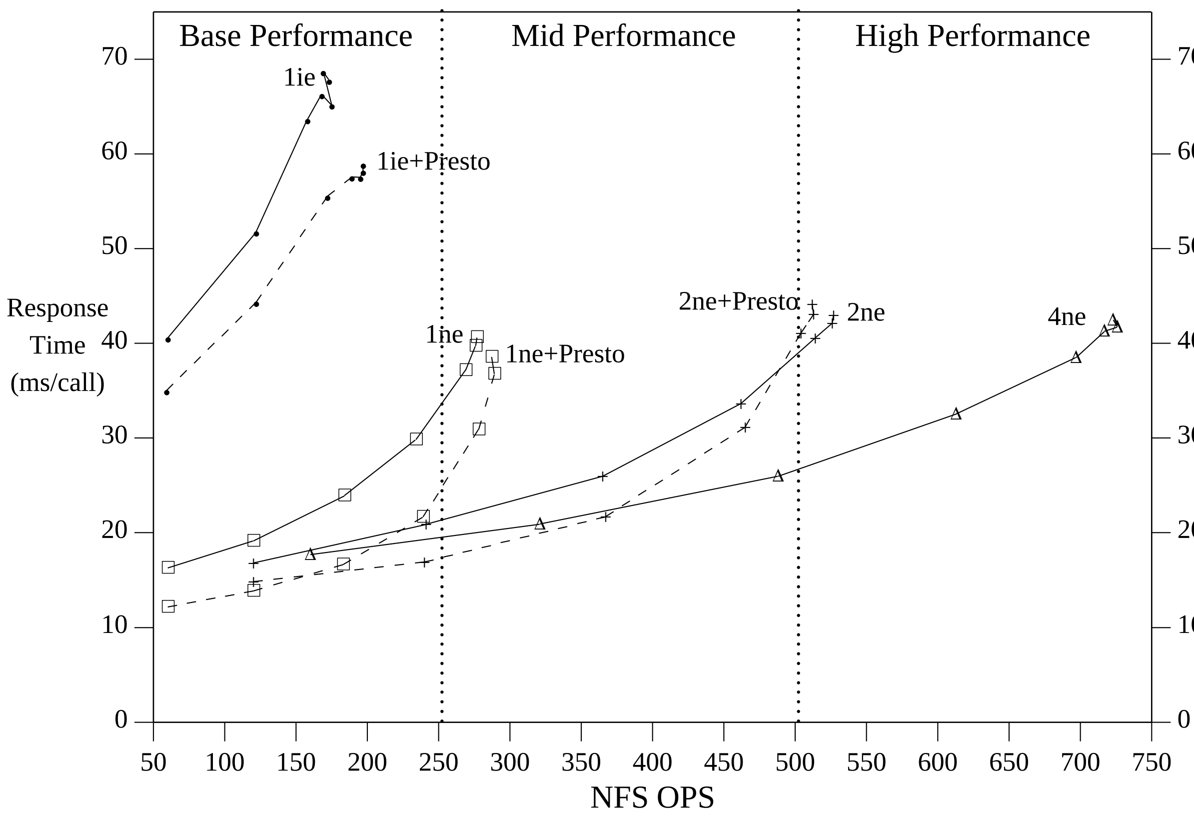
Although we have not run tests on the routing performance of the SPARCserver 490 using the tuning efforts described here, it is likely that the NFS performance of any system will decrease when additional, extraneous load is placed on it. To prevent excessive routing overhead, machines that must communicate with each other should be placed on the same subnet. If a customer's configuration does demand high routing throughput, a dedicated SPARCstation 1 represents a cheap and excellent high capacity routing device.

3. Configuring a SPARCserver 490 for Desired Performance

The SPARCserver 490 is an extremely versatile machine. It balances high compute performance and a large (up to 640MB) main memory with a fast and expandable I/O system. In the role of an NFS server the SPARCserver 490, with the addition of inexpensive I/O processing boards, easily exceeds the performance of similarly configured special purpose machines. This section of the report shows how a SPARCserver 490 can be configured for any of three major performance ranges. The information can aid a system purchaser in deciding how to configure a new system, and can aid the current SPARCserver 490 owner in planning the expansion of a current system to accommodate increased user needs. The configurations described in this report are segregated based on their expected NFS performance, as measured by the nhfsstone benchmark.

Figure 5 is a summary of the full range of NFS performance achieved by the SPARCserver 490 on the nhfsstone benchmark using the standard nhfsstone operation mix. The most efficient usage of the SPARCserver 490's CPU, disk and network subsystems can be achieved by providing the proper mix of system resources for each of the three performance ranges. The following sections describe how to configure the machine depending on the expected NFS performance requirements of the customer's environment.

Figure 5. Nhfsstone Performance According to Configuration

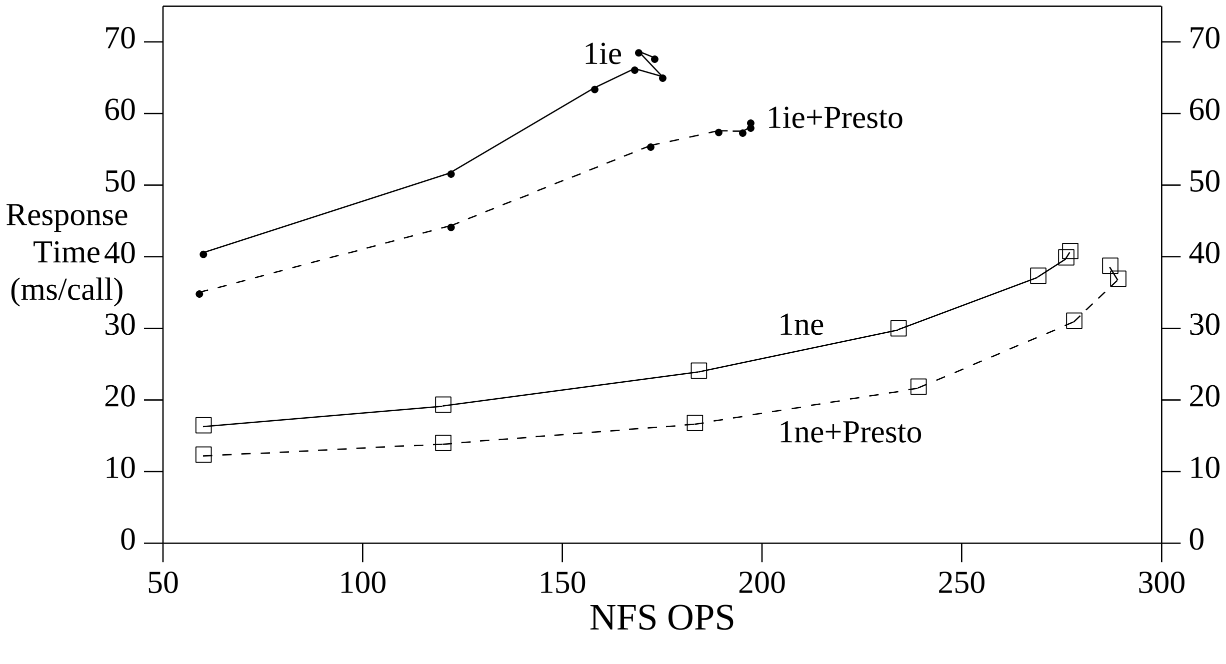


3.1. Base Configuration Performance: Under 250 NFSops/sec

Figure 6 is a summary of the NFS performance of smaller SPARCserver 490 configurations. It shows that the best low-end configuration of a SPARCserver 490 includes a single NC 400 Ethernet co-processor and a Prestoserve board; this configuration provides NFS performance of up to 287 NFSops/sec with a response time of less than 40 ms/call.

It should be noted that all the low-end configurations were tested using a single Ethernet segment on either the onboard ie Ethernet interface or the NC 400 co-processor's ne interface. Three SPARCstation 1 systems generated the load, and each used a single partition on a separate disk on the server. Thus, there were three load disks on the server, plus the server's system disk (which showed negligible load). Although multiple disks were used on the server for these tests, the disk utilization levels discussed below and shown in Figures 7 and 8 indicate that a single disk is sufficient to achieve peak performance on the best low-end configuration.

Figure 6. Base Configuration NFS Performance



Note from Figure 6 that the tested low-end configurations provide a range of NFS performance, and that the smallest and least expensive configuration can be expanded into the highest-performance low-end configuration without sacrificing any already purchased hardware. The standard onboard ie Ethernet interface of the SPARCserver 490 supports up to 173 NFSops/sec. Adding a Prestoserve board boosts performance to almost 200 NFSops/sec and also reduces response time by about 10 ms/call.

Performance can be further improved by using an NC 400 network co-processor instead of or in addition to the onboard interface. The system with a single NC 400 board achieves 277 NFSops/sec and cuts response time by more than half. The Prestoserve board also complements the NC 400 in this performance range. Adding a

Prestoserve to the configuration with the NC 400 improves SS-490 performance to 287 NFSops/sec, and also reduces response time by about 10 ms/call for equivalent NFS throughput.

Figure 7. CPU and Average Disk Utilization for the ie Interface

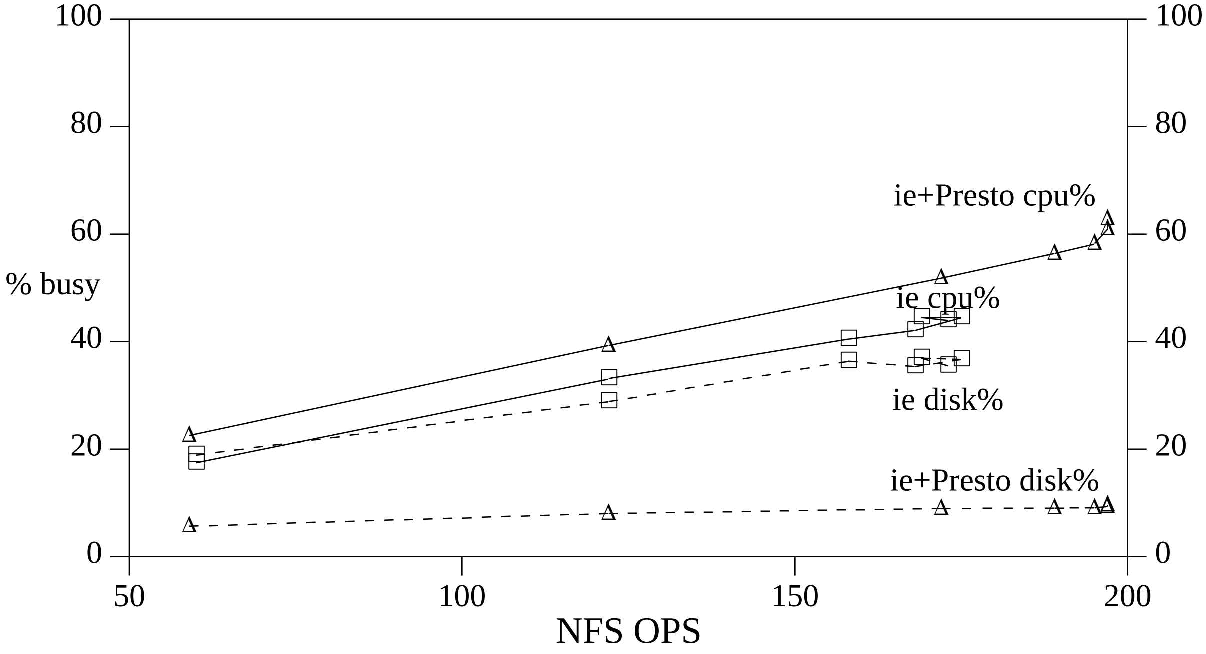


Figure 7 shows the effect of the nhfsstone benchmark on the CPU and disk resources of the SPARCserver 490, when using the onboard ie Ethernet interface. The two curves in the middle show the CPU and average disk utilization without a Prestoserve board. Although NFS performance stops increasing at about 170 NFSops/sec, neither the disks or the CPU appear to be the bottleneck since each is only about 40% utilized. It is probable that the Intel 82586 Ethernet chip on the SPARCserver 490 CPU board is what constrains the performance of this configuration. Adding a Prestoserve board dramatically reduces average disk utilization to less than 10%. This makes it possible for a single disk to provide all the bandwidth required to service the ie interface. Although the Prestoserve board reduces disk utilization, it does result in higher CPU utilization.

Figure 8. CPU and Average Disk Utilization for the NC 400 Interface

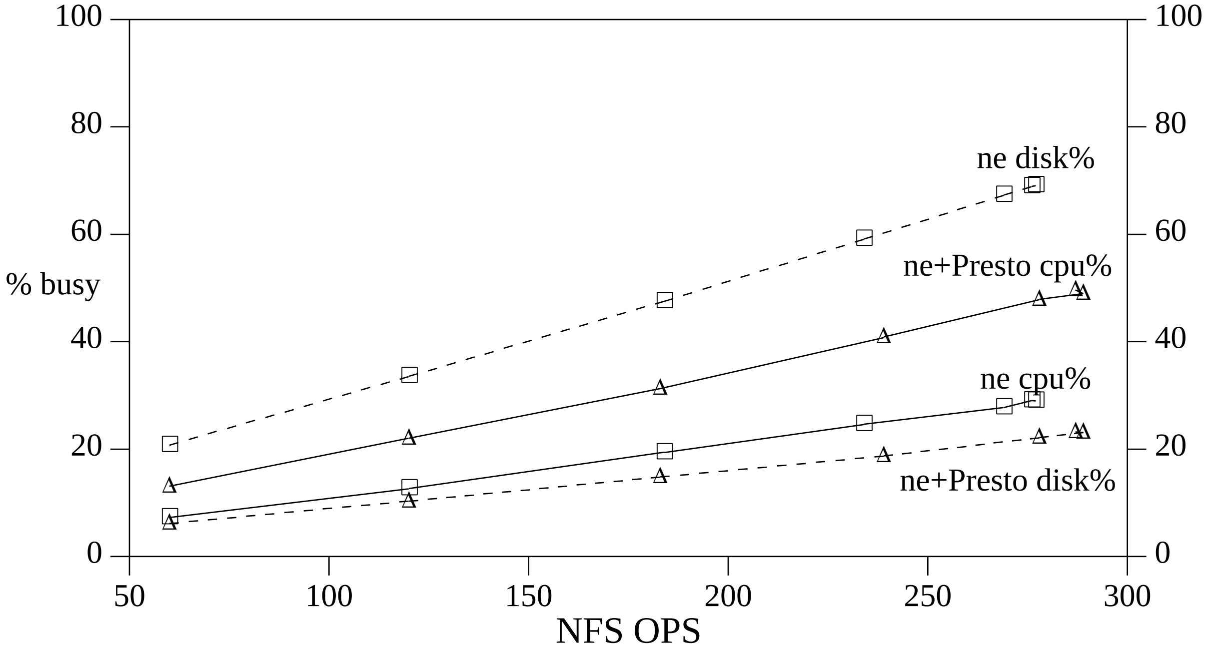


Figure 8 shows CPU and disk utilization in configurations with a single NC 400 Ethernet interface; contrast it to Figure 7. Note that neither the CPU nor disks are saturated even at the peak of 287 NFSops/sec. The bottleneck in this case is the Ethernet, which transferred 730 packets/sec and suffered 9.8 collisions/sec at the peak NFS throughput.

The lower solid curve in Figure 8 shows the CPU utilization using a single NC 400 board. Compare this to the ie CPU utilization curve from Figure 7 at an NFSops/sec rating of 170. The NC 400 ne interface loads the SPARCserver 490 CPU only half as much as the ie interface. However, disk utilization is somewhat higher when using the NC 400 interface. This is because the onboard processing on the NC 400 reduces response times and offloads the CPU, but not the disks. Thus the disks become far busier relative to the CPU. At the peak of 277 NFSops/sec using the NC 400 interface alone, each of the three disks is 70% utilized.

In this situation, the Prestoserve board nicely complements the NC 400 board. The lowest curve in Figure 8 shows that a Prestoserve board causes average disk utilization to plummet from 70% to 23% at 287 NFSops/sec. This again makes it possible for a single disk to serve the requirements of the ne interface. The Prestoserve board almost doubles CPU utilization, but this is still no greater than the ie interface's CPU requirements. With the NC 400 and Prestoserve boards, this low-end SPARCserver 490 configuration uses only half its CPU bandwidth to saturate the Ethernet media and provide unrivaled NFS response times for a single Ethernet configuration.

3.2. Mid-range Performance: 250 to 500 NFSops/sec

We define *mid-range* NFS performance to be 250-500 NFSops per second on the nhfsstone benchmark, with a response time less than 70 msec/call. Based on our experiments on the SPARCserver 490, we recommend the following configurations for mid-range NFS performance:

Configuration A: SPARCserver 490 with 2 NC 400 Ethernet Co-processors; the Prestoserve Accelerator; 2 or more IPI disks.

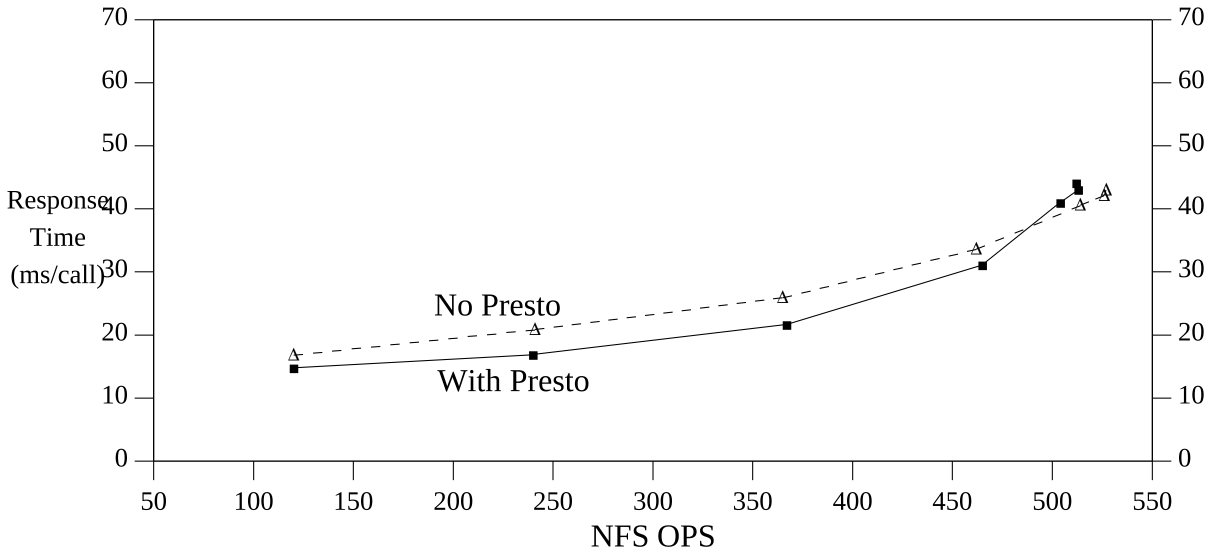
Configuration B: SPARCserver 490 with 2 or 3 NC 400 Ethernet Co-processors; 4 or more IPI disks.

Using the NC 400 Ethernet co-processors was critical in both of these configurations. Prestoserve was useful in Configuration A in lessening the disk bottleneck and thereby achieving the required performance with fewer disks. The Prestoserve also improved response times by about 10% when the load was below 450 NFSops per second. An analysis of both configurations is presented in the rest of this section.

The difference between the two configurations is that one uses the Prestoserve NFS Accelerator and therefore, requires fewer number of disks to obtain slightly better response times. Prestoserve does impose some CPU overhead which is why we also consider a configuration without it.

Figure 9 shows a comparison of the nhfsstone performance with and without the Prestoserve Accelerator. 2 NC 400 Ethernet co-processors and 6 IPI disks were used in both cases. Maximum performance with Prestoserve was 512 NFSops @ 44 msec/call and 527 NFSops @ 43 msec/call without Presto. Three SPARCstation 1 clients per network were used to generate the NFS workload.

Figure 9. Nhfsstone Performance with and without Prestoserve (6 disks)
NFSops and Response Time



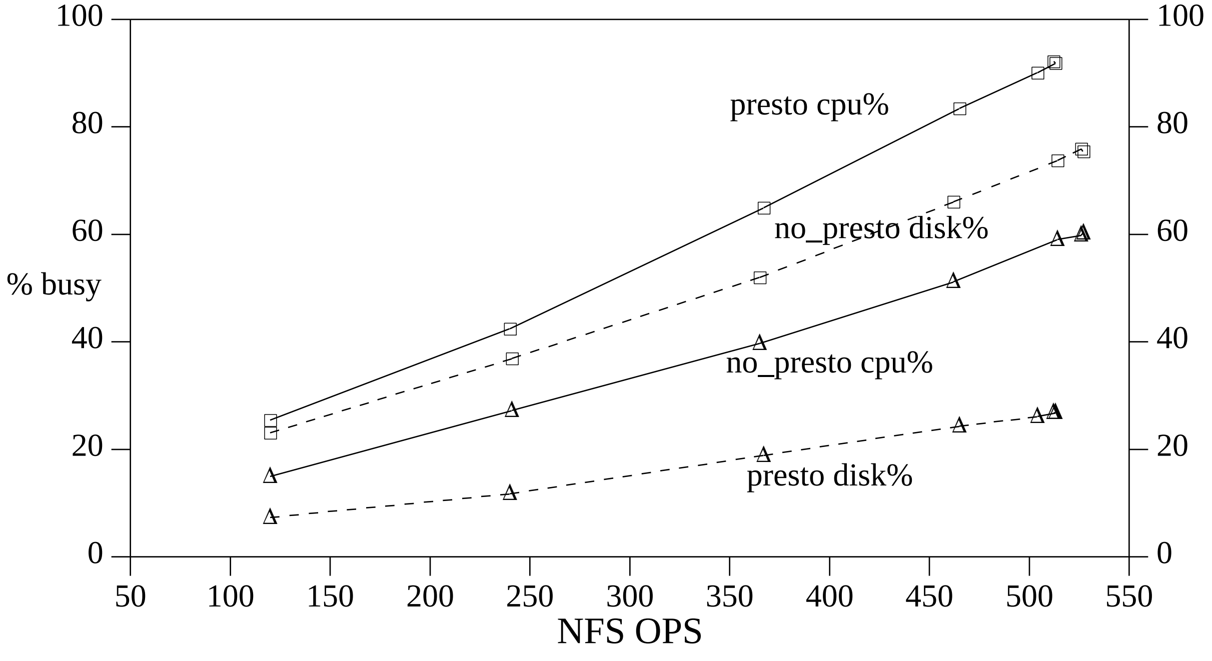
Though the maximum Nhfsstone performance was lower with Prestoserve, the response times were consistently better with it when server load was under 450 NFSops. NFS performance with Prestoserve degrades over 450 NFSops possibly because of its limited memory of 1 MB that does not scale with more clients and disks. Experiments using 4 clients and 4 disks substantiated this claim as the response times and NFS throughputs with Prestoserve were always better than without it.

While Prestoserve is ideal for mid-range NFS configurations with few disks, it does impose a CPU overhead. Figure 10 shows the CPU and disk utilizations with and without the Prestoserve Accelerator. In both cases, 6 IPI disks were used. Average disk utilization was 26% using Prestoserve and 75% without Prestoserve. To demonstrate that a similar NFS performance can be obtained with just 2 IPI disks and the Prestoserve Accelerator, Nhfsstone performance for a base configuration is shown in Appendix III.

The use of Prestoserve causes a CPU overhead as a result of which the server CPU is 92% busy when using Prestoserve and 60% when it is not. Due to this, it is not possible to obtain a higher NFS performance with Prestoserve.

When Prestoserve is not used, the disks become the bottleneck. Adding more disks does not increase NFS throughput for the two network case, as the Ethernet chips are also a bottleneck at this point. At the maximum NFSops level, 650 packets were transferred and 7 collisions occurred per second. Improving performance would require adding more networks and disks with appropriate load balancing.

Figure 10. Nhfsstone Performance with and without Prestoserve
CPU and Disk Utilizations



3.3. High-end Performance: Above 500 NFSops/sec

Figure 11 is a summary of high-end NFS performance. Unlike the mid range options, only one configuration is recommended in this performance range, because it optimally utilizes the CPU, disks and Ethernet interfaces to provide maximum performance. This configuration consists of a SPARCserver 490 with 4 NC 400 Ethernet co-processors, and 8 IPI disks with two disk controllers. Such a configuration can provide in excess of 725 NFSops/sec with a response time of 41 ms/call.

Figure 11. High-end Configuration NFS Performance

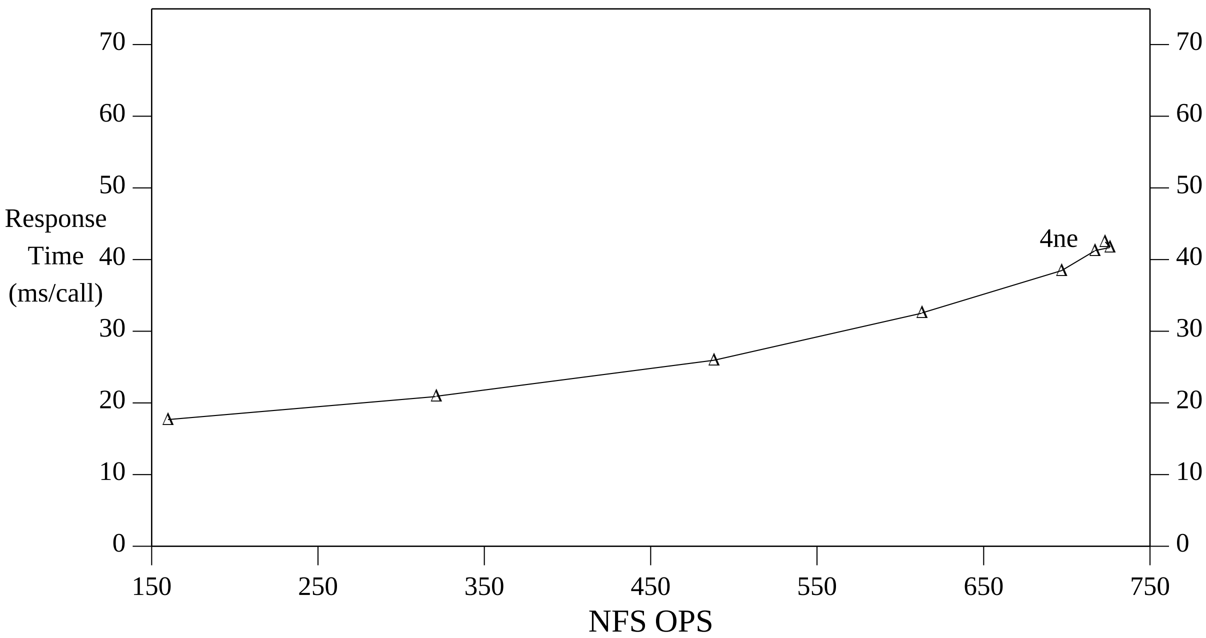
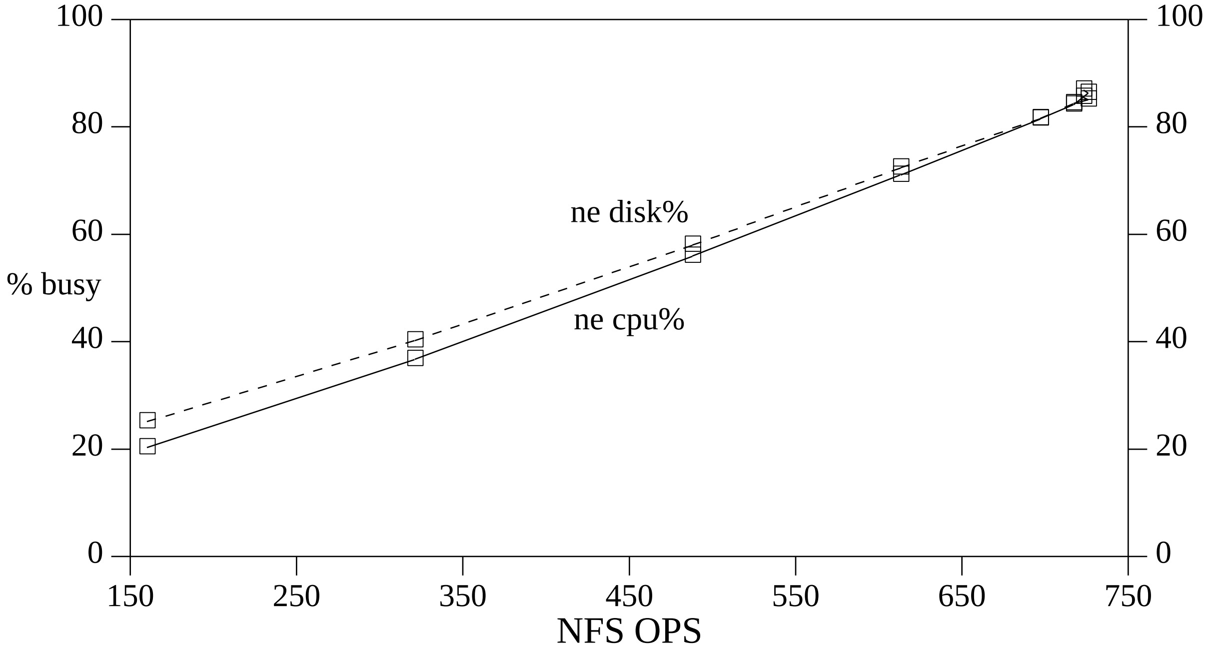


Figure 12 shows CPU and disk utilization figures for this configuration. Both the CPU and the average disk utilization for the 8 load disks peak at 86% utilization. In addition, all four Ethernet segments are near maximum capacity at the peak NFS throughput. A Prestoserve board would not help in this configuration, since there is no CPU bandwidth left to support it. However, examination of Figure 5 shows that the absence of a Prestoserve board does not appreciably hurt response time as long as there are sufficient disk resources. Also, the requirement for 8 disks in this configuration does not seem onerous given that the machine is supporting 4 independent Ethernet segments and NFS client machines that could number in excess of a hundred. Figure 12 shows that all 8 disks are saturated at peak NFS throughputs. Note that the average disk utilization can be reduced by adding more disks.

Figure 12. High-end Configuration CPU and Average Disk Utilization



The tests in this section show that the SPARCserver's CPU is fully utilized only when the four Ethernets are operating close to capacity. However, if a site's NFSops load on the server is lower than in this test case, the SPARCserver 490 can support even more NFS clients by accepting up to 6 NC 400 Ethernet interfaces. Since the SPARCserver 490 is a uniprocessor machine, we cannot add another CPU to avoid CPU saturation at peak performance. However, multi-processor servers available from Sun in the future will be able to overcome the CPU imposed limit, to provide even further growth in NFS server performance.

4. Summary

The basic SPARCserver 490 provides power and flexibility when used as a file server for large work groups. We have shown that proper software configuration and judicious use of optional hardware can yield dramatically higher NFS performance, to provide responsive support for large networks of many client workstations. Configurations of different sizes have been described, each of which makes optimal use of system resources at that configuration's level of performance. Customers are urged to work with their local salespeople and systems engineers to develop configurations tailored to their requirements.

NFS performance is only one aspect of a successful file server system. When choosing a file server, it is also vital to consider such factors as the vendor's commitment to industry standard hardware and software, the range of system offerings, networking expertise, system reliability, and customer-support programs. These factors all contribute to the success of a file server installation and operation.

5. Acknowledgements

A number of people devoted much of their time and effort and volunteered excellent suggestions to help us achieve our goal of tuning the SS-490 for optimal NFS performance. We gratefully acknowledge the assistance of Rich Clewett, Bill Van Loo, Anil Shivalingiah and Larry McVoy. We would also like to thank Neal Nuckolls, Brian Palowski and Hal Stern for their excellent insights while reviewing this paper.

The initial NFS testing effort was led by Maneesh Dhir with assistance from Prasad Wagle. Our task would have been far harder without their pioneering work. Finally we would like to thank Cheena Srinivasan of Server Marketing and our managers, Nhan Chu and Paula Sager for their extraordinary support and encouragement during this project.

6. References

1. *NC 400 Network Coprocessor Installation Manual*, Interphase West, 381 East Evelyn Avenue, Mountain View, CA 94041.

2. *PrestoServe User's Manual, Release 2.0*, Legato Systems, Inc, 260 Sheridan Avenue, Palo Alto CA 94306, 1990.

3. Kirk McKusik, Samuel Leffler, Michael Karels, Luis Cabrera, *Measuring and Improving the Performance of Berkeley UNIX*, Computer Systems Research Group, University of California, Berkeley, Berkeley, CA 94720, Nov. 30 1985, pp 38.

Appendix I

Machine and Benchmark Configurations

I. Recommended Server Configurations

The following list summarizes the server configurations recommended in Section 3 of this paper:

Base Configuration SPARCserver 490

SPARCserver 490 with 32MB memory
One IPI disk
One Interphase NC 400 Network Co-processor
One Sun Prestoserve NFS Accelerator

Mid-range Configuration "A" SPARCserver 490

SPARCserver 490 with 32MB memory
Two or more IPI disks
Two Interphase NC 400 Network Co-processors
One Sun Prestoserve NFS Accelerator

Mid-range Configuration "B" SPARCserver 490

SPARCserver 490 with 32MB memory
Four or more IPI disks
Two or three Interphase NC 400 Network Co-processors

High-End Configuration SPARCserver 490

SPARCserver 490 with 64MB memory
Eight IPI disks, with two IPI controllers
Four Interphase NC 400 Network Co-processors

II. Benchmark Configurations

This section describes the actual benchmark and machine configurations that were used to generate the statistics reported in Section 3.

We ran Nhfsstone version 2.0.3 with the following parameters on each of the diskless SPARCstation 1 load clients:

nhfsstone -p 4 -t 300 -l <load>

This implies that on each client, the benchmark forked off 4 processes that simultaneously made NFS calls to the server. The processes each ran for 300 seconds and together imposed the specified load (given in NFSops) on the server. The total load on the server was computed by summing up the loads generated by the individual clients. We used a simple but effective synchronization mechanism to start multiple clients within 1 second of each other.

Each NFS client machine mounted an individual disk from the server. Since there were 4 simultaneous processes per client, the disk activity was random. Care had to be taken to balance the disk loads properly.

NFS Client Configuration

SPARCstation 1

Multiple (2-10) SPARCstation 1's with 8MB of memory
Diskless
SunOS 4.1 Generic Diskless (GENERIC.DL) kernel
biod daemons increased from 4 to 8

NFS Server Configurations

The SPARCserver 490 ran SunOS 4.1PSR_A which was tuned as described in Section 2. There was 64 MB of swap space for the server. NFS daemons were set to 32, but 16 would have been sufficient for the low and mid range configurations.

Base Configuration SPARCserver 490

SPARCserver 490 with 32MB memory
3 SPARCstation 1 load clients on 1 Ethernet
3 IPI disks, 1 ISP-80 Controller
One Interphase NC 400 Network Co-processor (removed during *ie* tests)
One Sun Prestoserve NFS Accelerator

Mid-range SPARCserver 490

SPARCserver 490 with 32MB memory
6 SPARCstation 1 load clients on 2 NC 400 Ethernets
6 IPI disks, 2 ISP-80 Controllers
Two Interphase NC 400 Network Co-processors
One Sun Prestoserve NFS Accelerator

High-End SPARCserver 490

SPARCserver 490 with 64MB memory
8 SPARCstation 1 load clients on 4 NC 400 Ethernets
8 IPI disks, 2 ISP-80 Controllers
Four Interphase NC 400 Network Co-processors

Appendix II

Scripts for Tuning the Buffer Cache

Modifying Buffer Cache Size

Buffer cache size varies between 20 and 40 for most machines. For NFS servers or any machine doing intensive I/O, its size should be increased to 64 for systems with four or fewer disks and to 112 for machines with more than 4 disks. The script below should be run as root on the vmunix file to be used. Presently, the new value of nbuf (the size of the buffer cache) is set to 64 (0x40). For a cache of size 112 use an nbuf value of 0x70. *The machine must be rebooted for this change to take effect.*

```
#!/bin/sh
adb -w /vmunix << EOF
nbuf?W 0x40
EOF
```

Monitoring Buffer Cache Statistics

Apart from a versatile statistics collection program called statit written by the Server Performance Group, few utilities exist to collect arbitrary kernel statistics. Since statit is not widely available, users can employ the adb script given below to extract buffer cache statistics since the system was last booted.

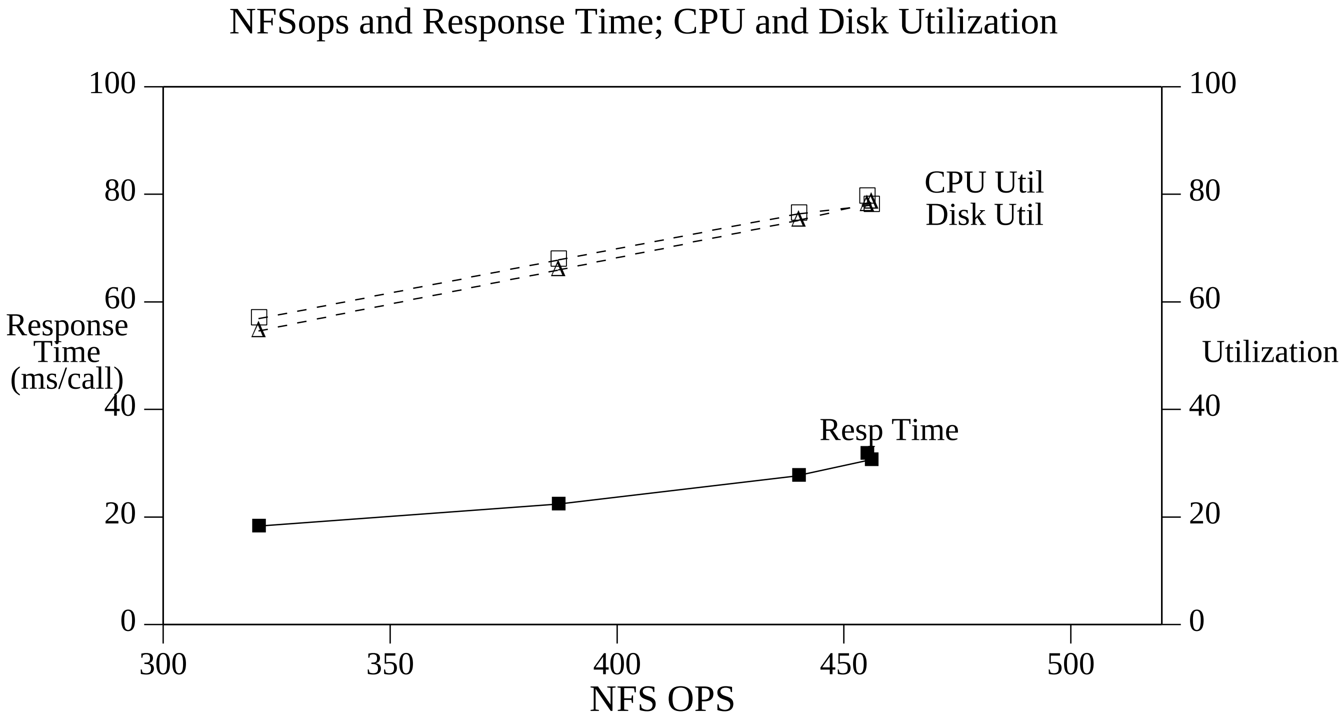
```
#!/bin/sh
adb -k /vmunix /dev/mem << EOF
=n"Buffer Cache Statistics Since Last Boot"
="-------------------------------------"
nbuf/D"Buffer Cache Size"n
bstats/D"total buffer cache read requests"n
+/D"hits in cache"n
+/D"times an aged buf was allocated"n
+/D"times an lru buf was allocated"n
+/D"times had to sleep for a buf"n
EOF
```

Monitoring Directory Name Cache Statistics

Although directory name cache statistics can be output using "vmstat -s", the following adb program, similar to the one above, prints slightly more data than vmstat.

```
#!/bin/sh
adb -k /vmunix /dev/mem << EOF
=n"Directory Name Cache Statistics Since Last Boot"
="---------------------------------------------"
ncsize/D"Directory Name Cache Size"
ncstats/D"cache hits"n
+/D"cache misses"n
+/D"number of enters into cache"n
+/D"number of enters tried when already cached"n
+/D"long names tried to enter"n
+/D"long names tried to look up"n
+/D"times LRU list empty"n
+/D"number of purges of cache"
EOF
```

Appendix III
Nhfsstone Performance for Mid-range Configuration A (Two Disks)



The graph above shows the Nhfsstone performance and resource utilizations for a mid-range configuration to be specially marketed by Sun. The SPARCserver 490 will contain two IPI disks and one ISP-80 IPI controller, two Interphase NC 400 Network Co-processor boards, a Sun Prestoserve board and 32MB of memory. This configuration provides an excellent price/performance point for a mid-range NFS server. The graph shows that the system provides over 450 NFSops/sec at a response time of only 30 msec/call. The CPU and disks are optimally utilized as the load on the server increases. This configuration also provides an inexpensive upgrade path for customers demanding high-end NFS performance. All that needs to be added are more disks and NC 400 Network Co-processor boards.

Appendix IV

Nhfsstone Benchmark Details

As described at the beginning of Section 2, the nhfsstone benchmark (pronounced n-f-s-stone, with a silent "h") is used on an NFS client to generate an artificial server load with a particular mix of NFS operations. The benchmark was developed by Legato Systems, Incorporated[5], and is now used throughout the industry to measure the NFS performance of file server systems. The benchmark allows the user to vary the mix of NFS operations that are generated by the client system; all of the tests reported in this paper used the default mix provided by Legato, which is considered to be a good reflection of a typical NFS network.

Changing the operations mix (e.g., the ratio of reads to writes) can have an enormous influence on the results of the benchmark. Customers currently in an NFS environment can monitor their own NFS mix and use that mix to run nhfsstone. (Nhfsstone accepts the output of the *nfsstat -s* command to describe its mix.) Also, it is vital to know what NFS mix was used when reading reported NFS performance results from any source.

To generate results over a range of NFSops/second loads (as reported in this paper), nhfsstone provides a shell script "nhfsrun" that allows the user to indicate a starting and ending value for NFSops/sec, and an increment between these values. The benchmark is then run at each increment from starting to ending value. If the ending value is set sufficiently high, a point will come above which the server cannot maintain processing of the NFS requests from all clients at the desired rate, and thus the actual *reported* rate of NFSops/second will start to level off at some value even as the *requested* rates increase. This point of stability is the peak NFSops/second value reported throughout this paper.

The tests were all run with version 2.0.3 of nhfsstone on each client workstation. (Realize that although server performance is being measured, the program actually runs on the client.) NFSops/second results described in this paper are always the sum of the NFSops/sec reported by each simultaneously running client, since this is the total NFS load being supported by the server.

The nhfsstone program is available free of charge, provided certain conditions are met, from Legato Systems. For further information, contact Legato at the indicated address or email them at:

nhfsstone-request@legato.com, or
uunet!legato.com!nhfsstone-request.

Be sure to include both your mailing address and your email address.

[5] 260 Sheridan Avenue, Palo Alto, CA 94306.

Tuning the SPARCserver 490 for Optimal NFS Performance

Table of Contents

Tuning the SPARCserver 490 for Optimal NFS Performance

Varun Mehta

Rajiv Khemani

Server Performance Engineering