A Summery of Network Performance Benchmarking of Sun 4/490 File Servers

Technical Staff

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-94-15
April 1994

A Summary of Network Performance Benchmarking of Sun 4/490 File Servers

Technical Staff

Computer Science Department, Brown University

December, 1990

Introduction — Purposes of Testing

The complaints were becoming more and more frequent - the response of our installation was "slow". After some initial evaluation of our systems it became clear that the servers were the bottleneck. The obvious next step was to replace the servers. Unfortunately, life was not as simple as in the old days, when we would have simply bought the next generation of server from Sun. There were several companies with products just arriving on the market which claimed to solve different aspects of the slow NFS service problem. To properly evaluate these solutions, we first had to determine the current performance characteristics of our servers. We then needed to determine what level of performance would suit our needs for the next couple of years. Finally, we had to test the new systems and decide which system or systems to purchase.

The Original Environment

As with any benchmarking, any results we achieved are very dependent on our environment, testing methods, and the execution of the tests. We will first discuss our environment, and later sections will address the testing method and execution. The initial environment consisted of 160 dataless SPARCstation 1's being served by four 4/280 servers. Most of the clients had 12MB of main memory and 100MB of local disk space. Because we are a Computer Science Department, the use of these workstations is probably more varied than at other Sun sites. The systems are used for office automation by administrative staff, systems work by technical staff, and text processing, mail and news reading, programming and computing by just about everyone. The servers had 32MB of memory and 11GB of SMD-4 disk space. Each server had 2 ethernet ports, which were connected to 2 backbone networks. One network served the Department and had 95 systems on it, while the other connected the systems in our 65-seat student lab.

All of these workstations are dataless. All home directories, projects, and most non-sun binaries are stored on the servers, with the kernel, root, swap, and most of */usr* being kept local on the

© Copyright 1990 Brown University Computer Science Department Technical Staff. Reproduction in whole or in part for other than personal use without prior written authorization is prohibited.

workstations to keep the server demands as low as possible. Using the standard Sun "*traffic*" program, and a Network General Sniffer network analyzer, we determined that the slowness users were observing was not due to network congestion, although the networks were very busy. The collision rate was below 5%, the point at which a network is generally considered to be overloaded. The following clues led us to believe the bottleneck was the servers:

- Workstation performance generally does not vary (unless extraneous processes are running on them), but we were seeing occasional delays of 10-30 seconds on file operations
- The servers generally had constant loads of 2 or more
- We had to increase the NFS timeout (timeo) mount option from 7 to 28 tenths of a second to avoid frequent "nfs server not responding" errors
- No other sites we knew of had as large a client/server ratio as we had

Original System Benchmarking

The first problem was to quantify the performance we were currently receiving from our 4/280 servers. Given that information, we could determine both what performance we would want from our new systems and how to determine if the new systems met those goals. We did two types of tests on the 4/280s. The first was to measure the number of Input/Output operations Per Second (IOPS) being delivered to our workstations. This seemed to be the best measure of the "real" performance of the systems. For this test we simply executed the *nfsstat* command on the servers at regular intervals and divided the "server nfs calls" number by the number of seconds between command executions. The peak number of IOPS we saw via this method was 45 per system. The total NFS IOPS being delivered by our 4 4/280s was therefore 180.

We then ran *nhfsstone,* using methods similar to those described below, and attained 65 IOPS per system. Unfortunately, this number is very rough, because the 4/280 is very interesting. When we ran our tests on the on-board ethernet controller (it is built into the CPU board), we saw an astounding 120 IOPS. The mystery was solved when we tried to test the second ethernet, which is its own VME board. This interface *was not able to generate any packets with less than a 70ms response time.* These tests resulted in 0 IOPS! We needed to have **some** value for a 4/280 with 2 ethernet boards, so we decided to run the *nhfsstone* test concurrently on both interfaces (as below) to average the response times, and to measure the number of packets being generated at an average of 70ms. A value of 65 IOPS was derived from this measurement. Therefore, the theoretical maximum total was 260 IOPS from the four systems. An important side note is that the second interface on 4/280 systems should not be expected to provide fast NFS service.

Based on the above information, and with input from the user community and an advisory committee, we determined that doubling the NFS IOPS of our servers would be a reasonable goal for our server upgrade. This would not only assure good performance for our current workstations but would provide excess capacity for the future addition of faster workstations (which make more demands on NFS servers). We also decided to separate the two networks out into seven, to be sure we had enough network bandwidth to carry the traffic we expect to have over the next year or two.

The alternatives

Given a good understanding of our goals and a warm, fuzzy feeling about how we could tell when we'd met them, we turned to the process of selecting equipment which might let us meet these

goals. **Legato** makes the Prestoserve board, which caches disk writes in non-volatile memory until it is convenient for the system to write the blocks to disk. This product only enhances disk write operations, while our NFS operation mix (15% reads and 3% writes) indicated that writes were not a major area of concern. If we'd had diskless workstations, the write percentage would have been 10% or more and Prestoserve would have made more sense.

Auspex, on the other hand, sells a system which is designed to increase the speed of all NFS operations. It is an I/O server, rather than a CPU server or general purpose machine (such as Sun's servers), which suited our purposes well.[1] The Auspex machine can handle 8 ethernet connections and has 10 SCSI disk channels to balance ethernet and disk traffic. Auspex has written a series of very interesting papers about NFS performance and benchmarking, and we used these as "sanity checks" while we were doing our benchmarking. Our numbers generally agreed with those from Auspex about the performance of the 4/280s. To paraphrase Auspex, they claim to achieve 1100 IOPS over 8 ethernets and 10 SCSI disks.

Epoch Systems gained our interest by proposing one of their optical jukebox systems. This would have given us much more storage capacity than we needed (20GB rather than 13GB), but at a reasonable price. They were claiming good NFS performance from their front-end as well. We decided not to consider them as a realistic solution, however, since they only support two ethernets on their front-end, and we were determined to split our network into 7 legs. Nevertheless, Epoch brought in a system for testing since they (and we) were interested in benchmarking it. The results are discussed in the Appendix. In general, the Epoch system seemed to do quite well for a two-ethernet system, especially when compared to a 4/280.

The final option we considered was **Omni Solutions.** Omni sells a set of ethernet/processor boards and kernel modifications. The boards process more of the NFS protocol stack, off-loading a lot of processing from the CPU and using the VME bus less frequently than would normally be the case. This is similar to the Auspex model of handling almost all NFS processing in hardware. Unfortunately, this product was only in beta test and would not have been a complete, tested product by the time we needed our new server installation to be in place. We are still considering testing it during our next semester break to see how much of an IOPS improvement it would give us on our 4/490s.

Give this state of affairs, the choice was between Sun and Auspex. We had no reason to disbelieve the Auspex claims, but Auspex was a very new company at the time (with no fully-loaded systems at customer sites). We were also happy with our **Sun** equipment and our relationship with Sun, and Sun was saying good things about their new 4/490 servers. It was, unfortunately, very difficult to get IOPS benchmarks from Sun. We decided the most reasonable next step was to order 2 4/490s (replacing 2 4/280s with each) and benchmark them to determine if they met our goals.

Limitations of Tests

In hindsight, we regret that our tests were not as thorough and accurate as they could have been. Several factors combined to decrease the quality and accuracy of our benchmarks. For instance, none of the programmers who planned and performed the benchmarks had any particular experience in benchmarking. Also, the second 4/490 arrived just before the servers were scheduled to go

1. We don't allow logins onto our servers to assure that all resources are devoted to NFS service.

into service, which both limited the available time, and forced us to perform some of the benchmarking procedure "live" (while the machine was in place as an NFS server and gateway). Nonetheless, we consider the information we received from these tests to be very valuable, and a far sight better than simply believing vendor claims. This testing took much longer and was more complicated than we hoped, so such a project should not be taken on lightly.

Server & Client Configurations

With one or two exceptions, the clients used in the NFS benchmarks were SPARCstation 1's configured with one 102 MB internal drive, and 12 or 16 Megabytes of memory.

The NFS servers tested were two Sun 4/490s, configured with 32 MB of memory. The first server was tested with 4 IPI disk drives of the 3 MB/second transfer rate variety, the second server was tested with 4 IPI drives rated at 6 MB/second. They were equipped with two and three "3/E" VME-bus Ethernet controllers respectively.

Benchmark Utilities

In an attempt to measure the peak sustained throughput of the servers, we used the *nhfsstone* utility from Legato Systems. This benchmark is run on an NFS client to generate an artificial load with a particular mix of operations at a desired rate. It makes system calls to induce particular NFS operations, and keeps track of their time to complete. The output of the test is a summary of the actual number of NFS operations performed, the rate achieved, and the average latency time in milliseconds per call.

The *nhfsstone* package provides a default mix of NFS operations that simulates a "typical" client load. However, we chose to use a different operation mix, which more closely simulated observations of our 4/280 servers. The operation mix used is as follows: 25% getattr, 1% setattr, 37% lookup, 4% readlink, 15% read, 3% write, 2% create, 1% remove, 11% readdir, and 1% fsstat.

A key element of our testing was to generate NFS traffic on all the network interfaces of a server at once. *Nhfsstone* is intended to run on a single NFS client, so it was necessary to run several instances simultaneously and combine the statistics. The easiest way to synchronize these tests was "by hand" from an X window session. (There was not time enough to develop a real RPC solution.) The target directories were distributed across all the disks as evenly as possible. Most of the disks had been partitioned into one or two large exported filesystems.

Any comprehensive experiment involving *nhfsstone* has too many variables to produce a single numeric rating such as Dhrystones; at the very least, you get a graph of generated load, in operations per second (also known as "IOPS"), against average latency in milliseconds per call. For the purposes of comparison, we varied the IOPS generated by the clients, and noted the value of the performance curve in two places:

• The IOPS load at which the latency exceeded 70 msec., the level at which, according to Auspex, users start noticing "slowness";

• The maximum IOPS load the server could handle in the given configuration, or the point where the latency increased rapidly.

Benchmark Results

The *nhfsstone* benchmarks were organized into "series," which consisted of executions of the program starting at a low request rate (lasting around 100 seconds), increasing until response time degraded seriously or something else happened; repeating for confirmation. The first tests were conducted on "boris", our first 4/490 with one on-board and two VME ethernet ports. *Ie0* was connected to the department backbone (background traffic caused high collision rates at times), and the other two interfaces were connected to lab networks with no other activity.

The first series started with one client generating 25 IOPS, and increased the number of clients (distributed across the 3 networks) each generating 25 IOPS. The clients ran a memory-walker program to reduce the effects of buffer caching. Using this configuration, a large knee appeared in the performance curve between 175 and 200 total IOPS (7-8 clients). It became evident that the number of clients doing the work was a factor in performance. For example, six clients generating a certain load got much better response time than eight clients generating the same total load.[2] Recompiling the kernel with a higher *maxusers* parameter and increasing the number of *nfsd* processes both increased performance somewhat.

The tests continued with *maxusers* = 48 and 36 *nfsds*, and with the number of clients fixed at 6. In this configuration, the load on boris could be increased to 288 ops/second, with the 70 ms. response level occurring at either 283 or >288 ops/second, depending on the day. Response times on the second day of this series were much worse for some unknown reason, so we did not obtain a good measure of the uppermost part of the curve. The results of this series are plotted in Figure 1.

The third series of tests was conducted on "natasha", the second 4/490, with 4 ethernet ports. Natasha was installed as the file server for our 3 lab networks, so it was acting as NFS server as well as NIS server and IP forwarder during the testing period. We attempted to minimize other usage of the networks and server, but could not eliminate it. Since by this time all the clients were running SunOS 4.1, we could use the SysV *mlock*() call to wire down client memory without thrashing.

We began this series of tests using 7 clients spread over the four networks. The problem with trying to use too many clients recurred; when serving 7 clients the server could not deliver more than 220 IOPS. Switching to six clients on four networks, we got a little more than 240 IOPS at the 70 ms level, and latency increased sharply at 250 IOPS.

We then made two modifications and checked for any performance difference. Using 6 clients distributed only on the lab networks (*ie0* not used) increased throughput slightly; maximum throughput was above 250 ops/second. Increasing the number of *nfsds* on the server from 36 to 50 gave further improvement. With this configuration, a load of 270 IOPS could be handled with latency of only 51 msec., but performance "maxed out" when trying to generate 280.

The rest of the test series were done on natasha, after the client kernels had an unofficial patch installed to disable adaptive NFS retransmission, which was shown to be causing throughput problems. At this point we stuck with a "base" configuration of 36 *nfsds* on the server, and 6 cli-

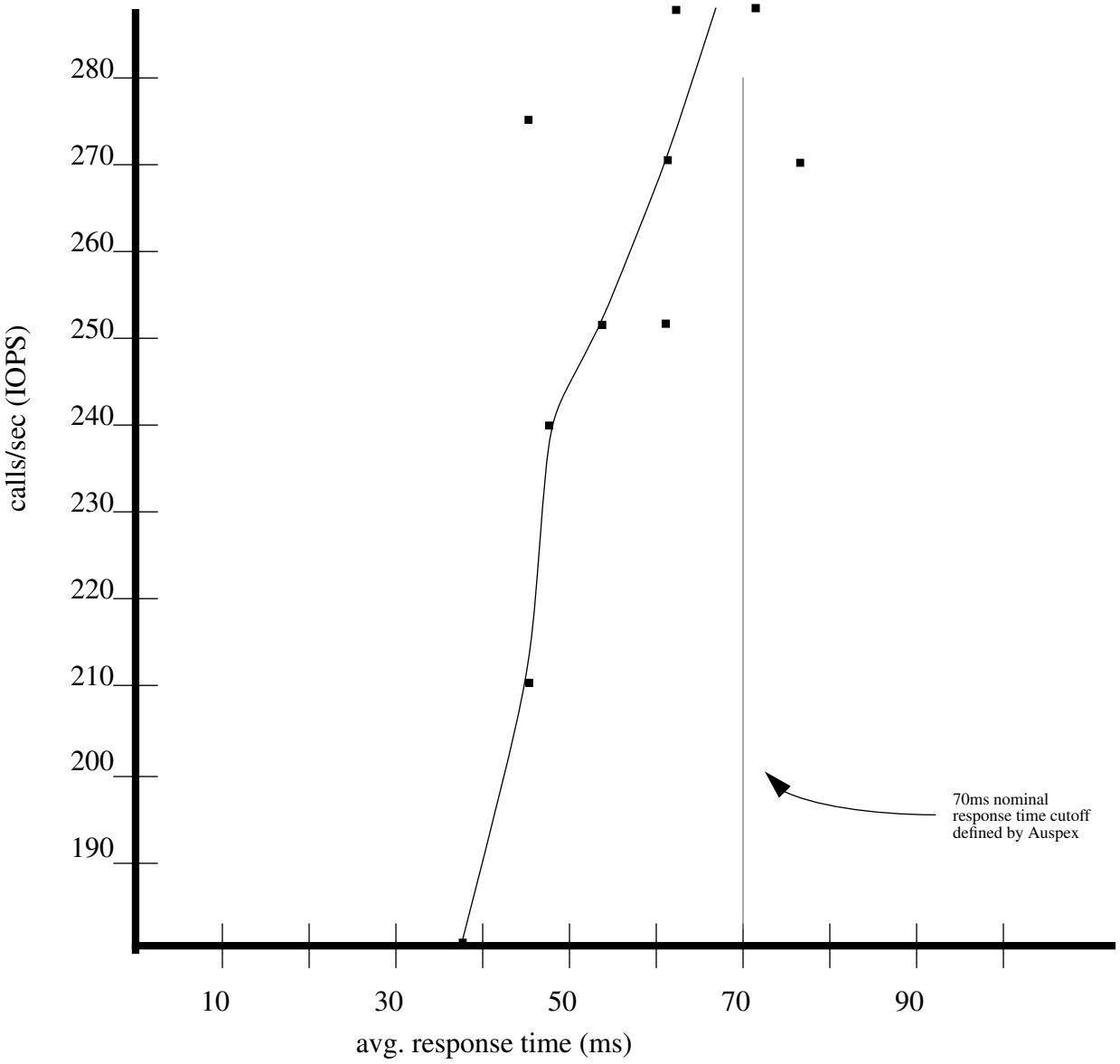
2. One explanation that makes sense is that each client *nhfsstone* process forks 7 load-generating subprocesses, meaning up to 7 NFS requests may be outstanding at once. Adding enough of these clients eventually exhausts a resource, possibly the nfsds themselves. Overflow of the UDP socket receive buffers was not evident, nor were any obvious *netstat -i* problems.

Figure 1

NFS response curve of "boris"

3 Ethernet interfaces, 6 clients

Summary of "reasonable" data points



ents spread across all the networks, so that some sort of valid comparison could be made with the best boris series above.

In this configuration, we reached the 70 ms latency level at between 258 and 275 IOPS (again, some days performance was better than others), and at roughly 110 ms. latency, the curve leveled off and angled downwards (the server evidently could not keep up with the clients' request rate; see Figure 2.).

Follow-up Tests

Repeating the last tests in the "base" configuration, we observed that collisions on the clients were not significant at the highest traffic level, nor were IP fragment timeouts. Another interesting observation was that at the (apparent) saturation level, the server's processing times (from *vmstat*) were 0% user, 80% system and 20% idle.

Follow-up tests showed that both the number of clients, and disk arm contention (accessing multiple partitions on those disks that had them), had been limiting factors. The final test configuration consisted of four clients (over three networks), each accessing a separate disk exclusively. In this somewhat unrealistic setup, the server generated over 380 NFS ops/second with average latency under 60 ms. The server's utilization was 100% in system time.

As a final reality-check, we used the *nfsstat* system of benchmarking on the new systems. We were wondering if the 4/490 servers were providing more packets that the previous systems. Furthermore, we wanted to verify that NFS performance really **was** the bottleneck we should have been solving. It would have been quite disappointing (not to mention embarrassing) if the total demand for NFS packets from the servers was only 160 IOPS, which is what we were getting from our 4/280s. Fortunately, the results were positive. Over the course of a couple of weeks, we saw peak true performance of 130 IOPS from each system, for a total of 260[3]. This leads us to believe that our initial goals of increasing system-wide performance, to eliminate current bottlenecks and leave room for expansion, have been met.

Conclusions - or Lies, Damn Lies, Statistics, and Benchmarks

The testing described herein confirmed the principal hypothesis: that a Sun 4/490 could serve at least 270 NFS operations per second sustained, under conditions roughly approximating our usual client load. Under more "optimized" conditions, a server can handle at least 380 "IOPS" without extreme latency.

Several questions are still unanswered however, due partially to the constraints of our testing situation. In some of the earlier tests the performance curve tended to drop off sharply at a certain point. This level was affected by system parameter adjustment (*maxusers*, *nfsds*) but we could not pin down a specific bottleneck with the system monitoring tools we had. It would have been instructive to be able to "flesh out" the raw data in several ways (i.e., more confirmation of results; more data points at the limits of performance). Time was a constant adversary however, since the machines were on our research and educational networks, which were nearly always in use.

Another result we could not adequately explain was that NFS response latency increased with the number of clients generating the requests, even with the total load fixed. Ethernet collisions were

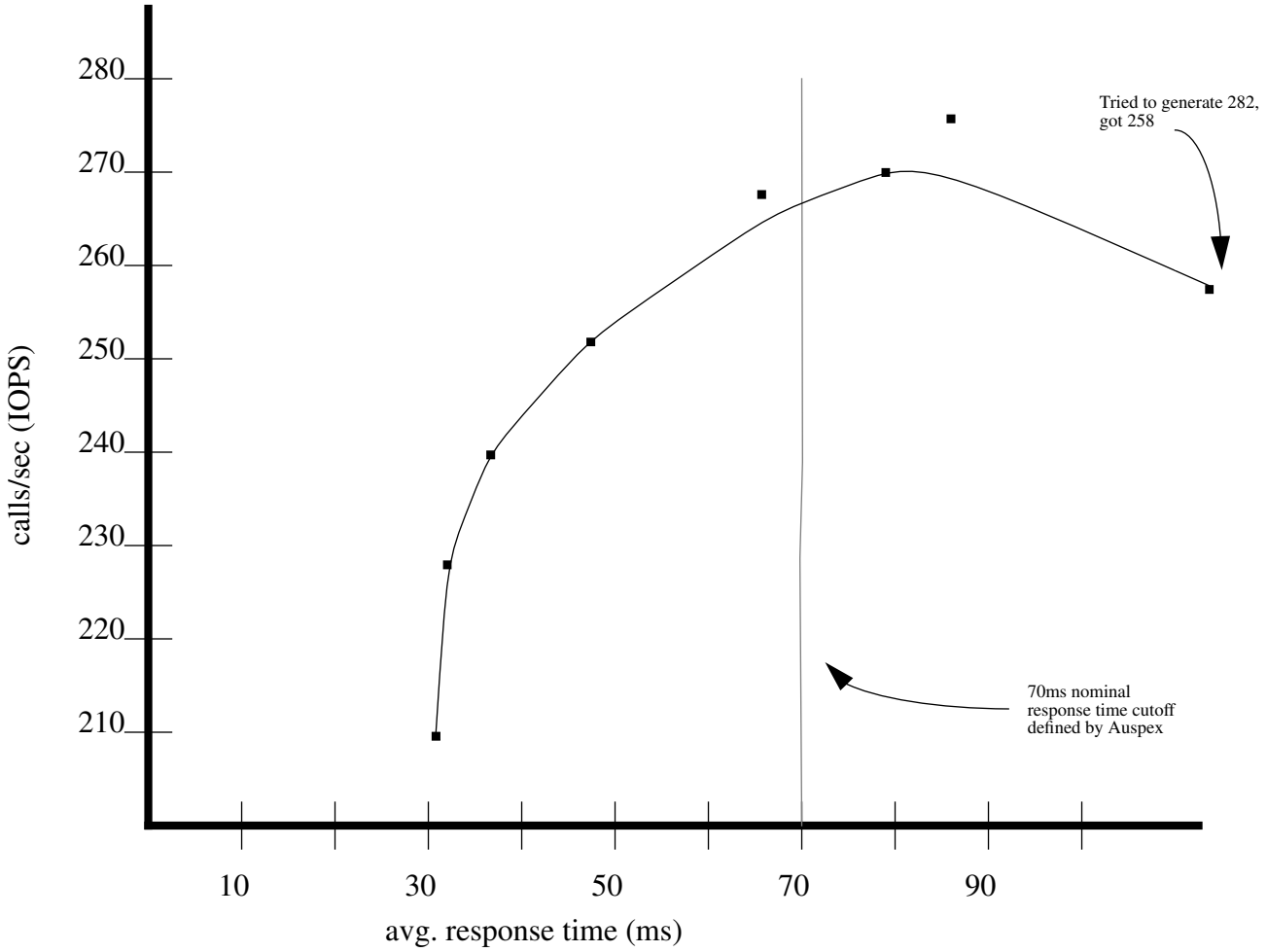
3. And even at these server loads, we heard no complaints of "slowness" from users.

Figure 2

NFS response curve of "natasha"

4 Ethernet interfaces, 6 clients

Composite of several tests



shown not to be the cause; it was at least partially related to the fixed number of *nfsds*, and may also have had to do with the greater total number of files being accessed at one time.

Finally, there seem to be two easy ways to greatly affect NFS benchmark results. The first is to change the number of nfsd daemons servicing NFS requests. Having too few nfsd daemons would result in client requests being queued before being dispatched. The second is to have those requests have exclusive access to the disks or contend for disk access. Exclusive access may be what vendors use when reporting their benchmark results, but the latter is the more "normal" scenario (and therefore the one we used).

Perhaps the most conclusive benchmark of all is that we have had no complaints from users about "slowness" since we installed the new systems!

Appendix – Other Tests

Using the same benchmark strategy, we tested an Epoch Systems file server (without any optical disk equipment), as mentioned previously. The system had two ethernet interfaces, and two 800MB disks that were partitioned roughly in half. The results are summarized in an attached graph.

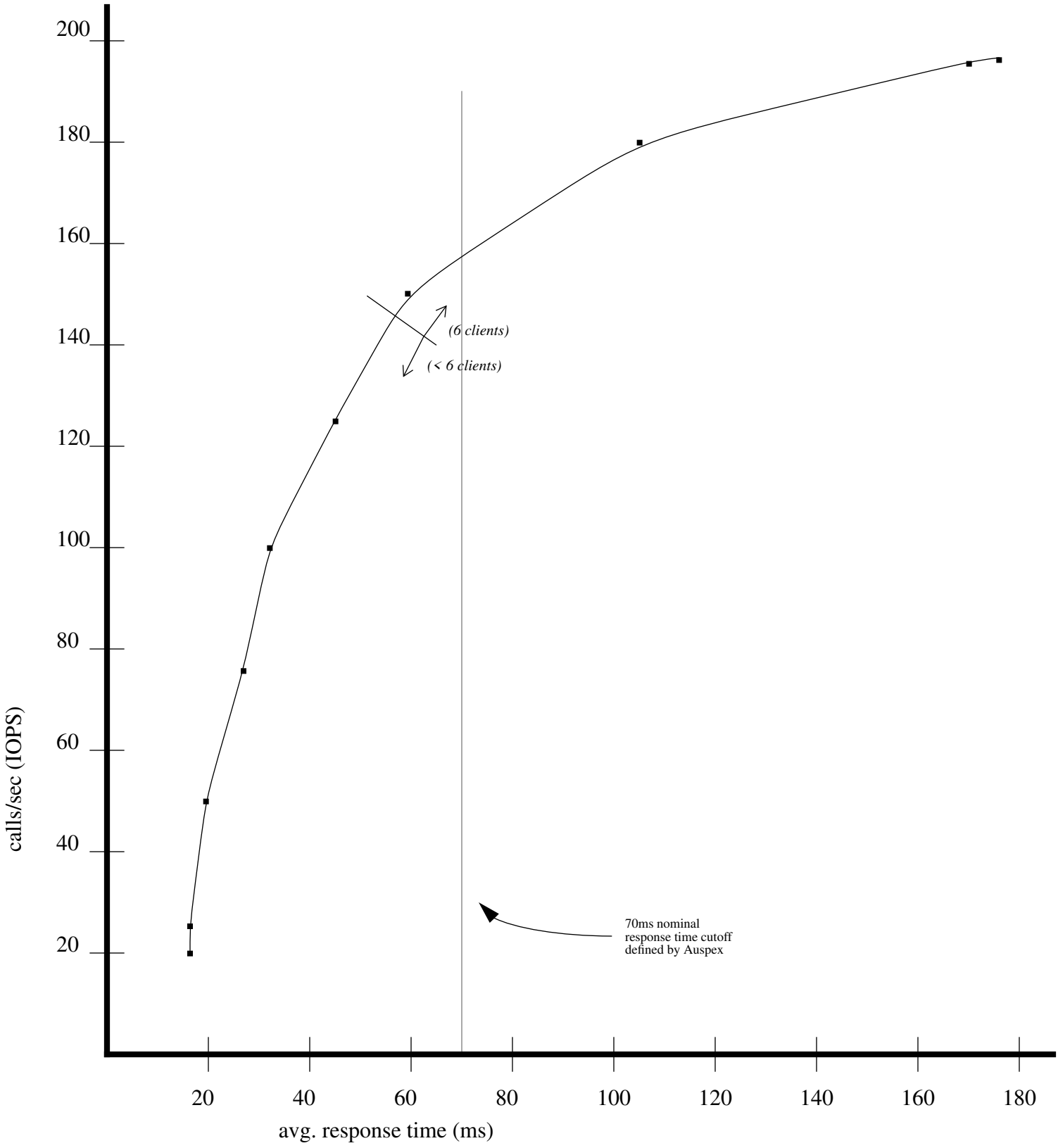
Another staff member developed and ran some tests on boris, the first 4/490, employing a shell loop to run sequences of commands (*find*, *grep*, *cat*,...) which generated the proper NFS operation mix. The goal was to simulate many interactive users more realistically than a synthetic load generator, by employing real commands.[4] Using 6-8 clients on the lab networks, this test generated 275-280 IOPS from the server, which matched our *nhfsstone* results.

Acknowledgments

Many thanks to Bruce Nelson of Auspex and Hal Stern of Sun for their discussions of NFS internals, networking, and benchmarking. Unfortunately, Bruce informs us that a new version of *nhfsstone* is about to be released![5] We do not look forward to redoing all of our tests. Also, thanks to John Wallace of Epoch Systems, Peter Olenick of Omni Solutions, and several system administrators we spoke with in regards to the third party solutions.

4. For an analysis of "real" versus "synthetic" NFS server benchmarks, see *Perspectives on NFS File Server Performance Characterization*, USENIX Summer '90.

5. The *nhfsstone* package is available from Legato's archive server, at the address `<request@legato.com>`.

Summary NFS response graph of Epoch server

2 Ethernet interfaces, 1-6 clients