NFS Sensitivity to High Performance Networks

Richard P. Martin and David E. Culler
Computer Science Division
University of California
Berkeley, CA 94720

Abstract

This paper examines NFS sensitivity to performance characteristics of emerging networks. We adopt an unusual method of inserting controlled delays into live systems to measure sensitivity to basic network parameters. We develop a simple queuing model of an NFS server and show that it reasonably characterizes our two live systems running the SPECsfs benchmark. Using the techniques in this work, we can infer the structure of servers from published SPEC results. Our results show that NFS servers are most sensitive to processor overhead; it can be the limiting factor with even a modest number of disks. Continued reductions in processor overhead will be necessary to realize performance gains from future multi-gigabit networks. NFS can tolerate network latency in the regime of newer LANs and IP switches. Due to NFS's historic high mix of small metadata operations, NFS is quite insensitive to network bandwidth. Finally, we find that the protocol enhancements in NFS version 3 tolerate high latencies better than version 2 of the protocol.

1 Introduction

Local and system area networks have made rapid performance advances in recent years [2, 3, 7, 11, 24], including increases in per-port bandwidth (e.g. from 10, to 100, to 1000 Mb/s), huge increases in aggregate bandwidth, plus reductions in network latency and software overhead. With switching, aggregate bandwidth scales with number of nodes in network. Routing ASICs can forward small packets at the line rate. Cut-through routing reduces the switch latency into the microsecond range. Such performance in Local Area Networks (LANs) was unheard of even 5 years ago—these networks represent a fundamental advance over their predecessors, 10 Mb Ethernet and FDDI.

The recent performance gains of these networks motivate the question: "How much do improvements in network performance translate into improvements in application performance?" Although the application space is enormous, previous work shows that 60%-70% of LAN traffic is filesystem related [8, 18]. We thus focus our study on the effects these recent networks have on the Network File System (NFS) protocol. In particular, we want to understand to what aspects of the communication subsystem (e.g. bandwidth, latency, overhead) is NFS most sensitive. Instead of attempting to answer these questions in a piecemeal fashion for specific point technologies, we take a systematic, parameterized approach so we can draw general conclusions.

We view the NFS system under investigation as a "grey-box". From this perspective, the clients and server comprise a system for which we can measure how certain outputs, e.g., the time to read a file, change in response to various parametric inputs. One class of inputs are the traditional characteristics of NFS workloads, e.g. the mix of reads/writes/lookups and the data set size. These inputs are governed by the SPECsfs mix and scaling rules. The second class of inputs are novel to this study. We vary each of the performance characteristics of the communication subsystem, including the bandwidth, latency and overhead. The output of the system is a curve of response time vs. throughput for a fixed set of communication characteristics. We can then understand the effect of the network parameters on the response time vs. throughput curve.

Our approach is two-phased. First, we build a simple model using standard queuing theory techniques. Second, we perform experiments on a live system. Using the model, we can make predictions about what will happen in a real system. Our goal in this work, however, is not to develop highly accurate models. Rather, the purpose of the model is to conceptualize how the system should behave as we change the networking parameters. The model's value lies in its ability to identify why the system responds as it does.

The second phase of our approach is experimental: we construct a live testbed on which we can change input parameters and measure the system's response. We begin with a network which is higher-performance than what is generally available, and then scale back the performance in a controlled manner. As we scale back the performance, we observe the "slowdown" of the NFS system as a function of different network parameters. From the slowdown, we derive the sensitivity of the system to each parameter. A high sensitivity indicates where future work will yield the largest improvements. Our choice of workload, SPECsfs, allows us to compare our results to industry published data. More importantly, using the techniques in this work we can infer the structure of the NFS servers from the data published by SPEC.

The combined use of a model and experimental data forms a syn-

This work was supported in part by the Defense Advanced Research Projects Agency (F30602-95-C-0014), the National Science Foundation (CDA 9401156), Sun Microsystems, Fujitsu and California MICRO. The authors can be contacted at {rmartin, culler}@cs.berkeley.edu.

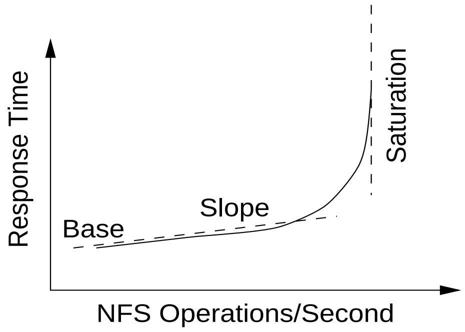


Figure 1: **Important Characteristics of the SFS Curve.** *This figure shows three important characteristics of all SFS curves: the* **base** *(i.e. minimum) response time, the* **slope**, *which determines the rate of increase in response time as load increases for a linear region of the curve and, the* **saturation point** *at the peak operations sustainable.*

ergy which is much more powerful than either alone. With only a model, we can predict the responses but the results are suspect. Measured data alone often lacks the simple conceptual interpretations that models provide. Using both a model and measurement we can explain the measured results in terms of the model. Points where the data deviates from model predictions expose weaknesses in our understanding of the system.

The contributions of this work are: (1) the first systematic characterization of NFS sensitivity to network parameters on a full-scale benchmark, (2) a comparison of simple queuing theory models against a live system across a large region of the network design space, and (3) a characterization of the SPECsfs benchmark which will aid those interpreting industrial results.

The remainder of the paper is organized as follows. After providing the necessary background and related work in Section 2, Section 3 describes the experimental setup and our methodology for emulating designs with a range of communication performance. Section 4 introduces a simple queuing-theoretic model of an NFS system. Section 5 documents the measured sensitivities to network parameters on two live systems and describes the accuracy of the predictions made by the model. Section 6 outlines some implications of our results and analyzes industrial data in the context of our methods. Finally, we conclude in Section 7.

2 Background

In this section we first describe the SPECsfs benchmark which forms the workload for our study. Next, we outline the LogP network model, which we use to parameterize the network inputs to the NFS "grey-box". Finally, we describe the similarities and differences between this and previous work done to quantify NFS performance.

2.1 SPECsfs Workload

SPEC, while widely known for its CPU benchmarks, also produces an NFS benchmark, SPECsfs (formerly known as LADDIS) [26]. SPEC released the latest version, SFS 2.0, in 1997 [22]. Version 2.0 adds several enhancements. First is the addition of version 3 of the NFS protocol [20] while retaining NFS version 2. In addition, TCP can be used as a transport layer instead of UDP. The combination of these two variants results in four possible configurations (e.g. NFS version 3 on UDP and NFS version 2 on TCP). We focus on NFS version 2 running over UDP because this will comprise a large number of installed systems. Unless otherwise reported, all results are for systems running SFS 2.0, NFS version 2 using UDP. We do examine some TCP vs. UDP tradeoffs in Section 5.4.

SPECsfs, as a synthetic benchmark, must define both the operation mix and scaling rules. The mix has been derived from much observation of production systems [22, 26]. Qualitatively, the SFS 2.0 operation mix is mostly small metadata operations and reads, followed by writes. The mix represents a challenge to latency tolerating techniques because of the small and synchronous nature of most operations.

Learning from past benchmarking errors, SPECsfs also defines scaling rules for the data set. In order for a vendor to report a large number of operations per second, the server must also handle a large data-set. For every NFS op/sec, the clients create an aggregate of 10 MB of data. The amount of data accessed similarly increases; for each op/sec 1 MB of data is touched.

Unlike the SPEC CPU benchmarks which report point-values, SPECsfs reports a curve of response time vs. throughput. The reported response time is the weighted average of different operations' response times, where the weights are determined by the percentage of each operation in the mix. Figure 1 shows an abstract SFS results curve. The *signature* of the curve contains three key features: the *base response time*, the *slope* of the curve in the primary operating regime, and the *saturation point*.

At low throughput there will be an average *base* minimum response time. The base represents the best average response time obtainable from the system. The base will be determined by several factors, including the network, the speed of the server processor, the size of the file cache, the amount of Non-Volatile RAM (NVRAM), and the speed of the disks.

As load on the server increases, there will be a region where there is a linear relationship between throughput and response time. The *slope* signifies how well the server responds to increasing load; a low slope implies the clients cannot perceive a more loaded server, while a high slope implies noticeable delays as we add load. The slope will be affected by queuing effects in the network, at the server CPU, the server disks and the client CPU. However, a much more important role in the determination of the slope is the changing miss rate in the server file cache.

As the load further increases, a bottleneck in the system will limit the maximum throughput at some *saturation point*. The nature of the bottleneck will determine if the point is reached suddenly, resulting in a pronounced inflection, or gradually. Example bottlenecks include an insufficient number of clients, lack of network bandwidth, the speed and number of CPUs on the server, and an insufficient number of server disks.

From our "grey-box" perspective, SPECsfs is quite useful because it fixes all the NFS parametric inputs but one: the server load. We can thus restrict the parameter space primarily to the network. At the same time, our choice of SPECsfs clearly limits our understanding of NFS in several ways; it is attribute intensive and it does not model the client well. Practically, however, our choice allows us to interpret industrial data published on SPECsfs in the framework presented in this paper. We perform a brief analysis of industrial data in Section 6.

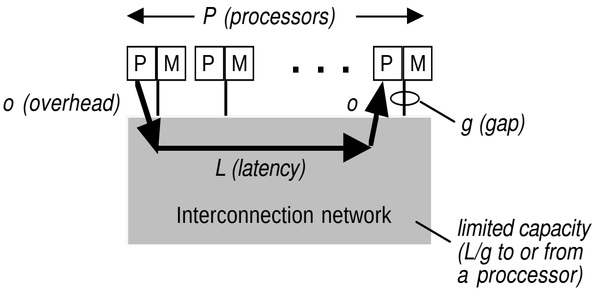


Figure 2: **LogP Abstraction.** *The LogP model describes an abstract configuration in terms of four performance parameters: L, the latency experienced in each communication event, o, the overhead experienced by the sending and receiving processors, g, the gap between successive sends or successive receives by a processor, and P, the number of processors/memory modules.*

2.2 LogGP Model

When investigating trade-offs in communication architectures, it is important to recognize that the time per communication operation breaks down into portions that involve different machine resources: the processor, the network interface, and the actual network. However, it is also important that the communication cost model not be too deeply wedded to a specific implementation. The LogP model [6] provides such a middle ground by characterizing the performance of the key resources, but not their structure. An environment in which processors communicate by point-to-point messages is characterized by four parameters illustrated in Figure 2.

L: the *latency*, or delay, incurred in communicating a message containing a small number of words from its source processor/memory module to its target.

o: the *overhead*, defined as the length of time that a processor is engaged in the transmission or reception of each message; during this time, the processor cannot perform other operations.

g: the *gap*, defined as the minimum time interval between consecutive message transmissions or consecutive message receptions at a module; this is the time it takes for a message to cross through the bandwidth bottleneck in the system.

P: the number of processor/memory modules.

L, o, and g are specified in units of time. It is assumed that the network has a finite capacity, such that at most $\lceil L/g \rceil$ messages can be in transit from any processor or to any processor at any time. If a processor attempts to transmit a message that would exceed this limit, it stalls until the message can be sent without exceeding the capacity limit.

The simplest communication operation, sending a single packet from one machine to another, requires a time of $L + 2o$. Thus, the latency may include the time spent in the network interfaces and the actual transit time through the network, which are indistinguishable to the processor. A request-response operation, such as a read or blocking write, takes time $2L + 4o$. The processor issuing the request and the one serving the response both are involved for time $2o$. The remainder of the time can be overlapped with computation or sending additional messages.

The available per-processor message bandwidth, or communication rate (messages per unit time) is $1/g$. Depending on the machine, this limit might be imposed by the available network bandwidth or by other facets of the design. In many machines, the limit is imposed by the message processing rate of the network interface, rather than the network itself. Because many machines have separate mechanisms for long messages(e.g. DMA), it is useful to extend the model with an additional gap parameter, G, which specifies the time-per-byte, or the reciprocal of the bulk transfer bandwidth [1].

2.3 Previous Work

Due to its ubiquity as a distributed filesystem there is vast body of work on NFS. Fortunately, [20] contains an excellent bibliography and summary. This section does not try to document all previous NFS work; rather we categorize related work and introduce papers which describe previous results upon which our work builds.

NFS studies fall into roughly three categories: protocol changes, client/server enhancements, and performance evaluation. Although papers contain some element of all three, often they focus in a single area. Our work clearly falls into the performance analysis category, using network performance as the dependent variable.

Although [20] deals with differences between NFS version 2 and version 3, it performs much performance analysis to justify the changes. It found that a server running NFS version 3 is roughly comparable to the same server running version 2 with NVRAM. However, little exploration of the impact of the version 3 protocol changes is made with relation to network performance.

An extensive bottleneck analysis is presented in [27]. The book examines the peak throughputs of real-world components (e.g. CPU, disks, network) and characterizes where the saturation point will be for different configurations. An interesting result of the work is that most servers in the SPEC results are over-provisioned with disks.

Perhaps closest in spirit to our own study is [4]. That work was primarily concerned with NFS performance over congested ATM networks. They found that a high L, in the 10's of milliseconds, was quite detrimental. A trace-fed simulation was used rather than a live system. Moreover, their custom workloads make a quantitative comparison to our work difficult.

The impact of specific networking technologies (ATM, Autonet, FDDI) was examined in [9]. Their conclusions are quite similar to ours: CPU overhead is a dominant factor in NFS performance. We compare their results to ours in greater detail in Section 6. Their workload was the precursor to SPECsfs, NFSstone [21]. Some of their most interesting data, however, came from their measurements of a large production system.

3 Methodology

In this section we describe our experimental apparatus and methodology. We first describe the machines used, followed by two different disk sub-systems, which result in two distinct server platforms. Next, we describe the network. Finally, we detail our technique for independently scaling the network parameters.

3.1 Experimental Setup

All of the machines in our experiments consist of Sun Ultra-1 workstations (167 MHz CPU, 512K L2 cache, a single 60 MB/s S-Bus and Solaris 2.5.1). Attached to the S-Bus is an internal narrow SCSI bus and local disk that holds the operating system and swap space.

All the clients have 128 MB of main memory. We use a total of 4 clients: 3 load-generators and 1 master control station.

The primary difference between our two servers is the disk subsystem. The "SCSI" system contains 128MB of main memory and 24 7200 RPM 9GB IBM drives. The drives are evenly divided between two SCSI buses. The S-bus interfaces used are the fast-wide Sun ''FAS" controller cards. In contrast, the "RAID" system contains 448 MB of main memory. The 28 7200 RPM 9GB Seagate disks are contained in a Sun "A3000" RAID. The disks are divided into 5 RAID level-0 (striping) groups; 4 groups have 6 disks and the last group contains 4 disks. The striping size is 64 Kb. The A3000 contains 64 MB of battery-backed NVRAM which can absorb some writes that may otherwise have gone to disk.

There are two reasons for investigating different systems. First, they allow us to draw conclusions about the effects of different hardware and software, e.g., drivers, main memory, and NVRAM. Second, having two systems serves as a check on our model; inaccuracies may show in one system but not the other. In addition, the RAID is closer in spirit to servers found in published SPEC data.

We use a Myrinet [3] to connect the load generators and server. The Myrinet network interface contains a "LANai" embedded processor which plays a key role in our ability to modify the network parameters. We have developed a custom device driver which sits under the TCP/IP stack. Our choice of a Maximum Transmission Unit (MTU) of 3.5 KB allows for the maximum bandwidth between pairs of nodes while balancing for the congestion effects of large packets. Although the links can sustain a rate of 160MB/s, store-and-forward delays between the S-bus and LANai limit the sustainable bandwidth to 26 MB/s (208 Mb/s) for any MTU size. The control station and other machines are also connected via a switched 10Mb/s Ethernet. The Ethernet is used to start and stop the benchmark as well as to monitor results.

3.2 Technique

The key experimental innovation is to modify the communication layer so that it can emulate a system with arbitrary overhead, latency, gap or bulk bandwidth. The basic idea is to introduce controlled delays in system components (e.g. the network interface) that correspond to LogP network parameters. The approach is similar in spirit to [17].

Varying the overhead, o, is straightforward. Before each send and before each receive, our custom device driver loops for a specific period of time before actually writing or reading the message. The processor is "stuck" in the device driver for the specified period and thus cannot perform other operations.

The gap is dominated by the message handling loop within the LANai network processor. To vary the gap, g, we insert a delay loop into the LANai's firmware code path after the message is transferred onto the wire and before it attempts to service the next message. Since the stall is done after the message is actually sent the network latency is unaffected. Also, since the host processor can write and read messages to or from the network interface at its normal speed, overhead should not be affected. We modify G in a similar manner, except that the delay is a function of the packet size.

The latency, L, requires care to vary without affecting the other LogGP characteristics. An obvious place to increase L is in the network switches. The "hard-wired" Myrinet switches, however, cannot be reprogrammed. We are thus forced to modify L at the edge of the network, either using the CPU or LANai processor. While simply stalling the send or receive path increases L, it also would have the side effect of increasing g. Our approach involves adding an extra delay queue inside the LANai. When a message is received, the interface processor deposits the message into the normal receive queue, but defers setting the flag that indicates the presence of the message. The time that the message "would have" arrived in the face of increased latency is entered into a delay queue. The receive loop inside the LANai checks the delay queue for messages ready to be marked as valid in the standard receive queue. Modifying the effective arrival time in this fashion ensures that network latency can be increased without modifying o, g or G.

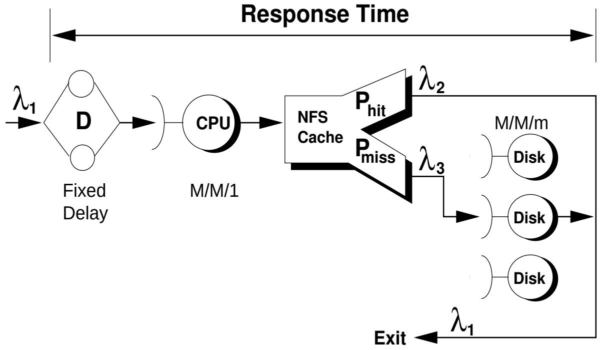


Figure 3: **Analytic Model.** *This figure shows the simple analytic model used to validate the results.*

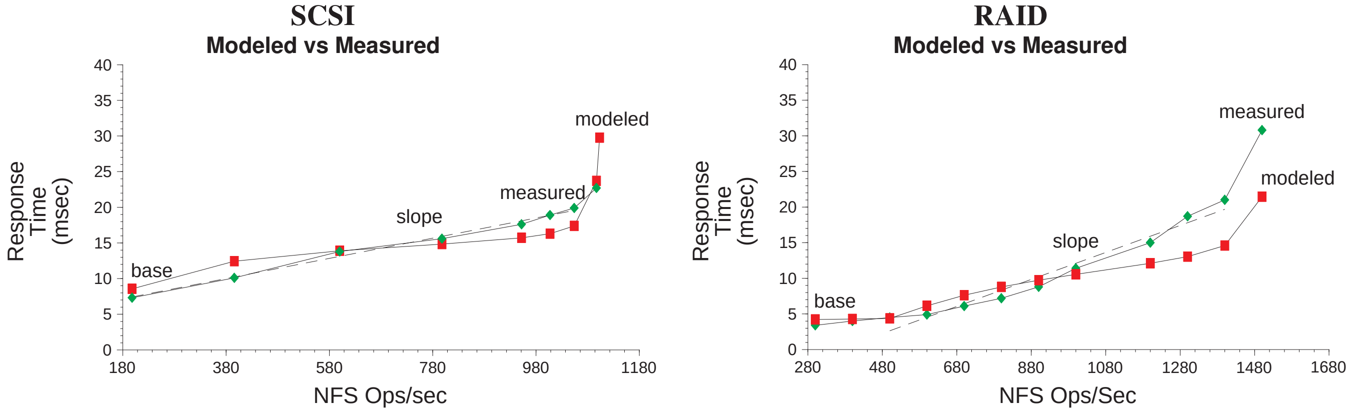
Once we have a working apparatus, we can derive the sensitivity of the SPECsfs benchmark to each parameter by scaling the parameters in turn and empirically measuring changes to the signature of the SFS curve. For example, we can measure changes to the base response time as a function of L. The strong correspondence of the LogP model to network-system components allows us to identify which components are most critical to performance. However, in order to explain these results or make any predictions we need a conceptual model of the NFS system. The next section introduces a such a model.

4 An NFS System Model

In this section we describe a simple analytic model of the entire system, but focused on the server portion of the system. The goal of the model is to provide a framework for understanding the effects of changes in L, o and G on the SFS results curve. We compare the predictions of the model against two measured SFS curves. Agreement between the model and experimental data builds confidence in both. Significant differences between the two show where either the model fails to describe the system, or where the system is misconfigured and thus not operating "correctly". In either case, more investigation may be needed to resolve the discrepancy. We end the section with the model's predictions on the sensitivity of the system to network parameters.

4.1 Model Construction

Figure 3 shows the queuing network we use to model an NFS server, adopting the simple techniques described in [13, 15]. The model consists of a CPU, disks, NFS cache, and a delay center. Our model ignores queuing delays in the controllers and I/O bus; they can easily support the load placed on them given the small nature of most requests.



(a) (b)

Figure 4: **Modeled vs. Measured Baseline Performance.** *This figure plots the modeled as well as baseline SFS curves for the SCSI system on the left as well as for the RAID based system on the right.*

We assume that the clients behave as Poisson processes with a sum mean arrival rate of λ_1 requests per second. Because the departure rate must equal the arrival rate, the departure rate is also λ_1.

The server CPU is the simplest component of the system. We model it as an M/M/1 queue. We derived the average service time, including all sub-systems (e.g. TCP/IP protocol stacks, and the local filesystem, UFS) from experimental measurement. For the SCSI based system, the measured average service time was 900 μsec per operation. The RAID system has a lower average service time of 650 μsec. See Section 5.2.3 for a detailed investigation of the components of the service time.

Most NFS operations have the potential to be satisfied by an in-memory cache. Only 7% of the SFS 2.0 mix are writes and these must bypass the cache—NFS version 2 semantics require that they exist in stable storage before the write completes. The 64 MB of NVRAM in the RAID can cache writes, however. The file cache size, and corresponding miss rate, are critical to determining the base response time as well as the slope of the SFS curve. However, the SFS strategy of increasing the data set size per op/sec places an upper limit on the effectiveness of a cache.

We model the NFS caches (both in-memory and NVRAM) as a splitter. The probability of a hit is given as P_{hit} and of a miss as $P_{miss} = 1 - P_{hit}$. On a hit, the request is satisfied and leaves the system. Because the data set accessed by SFS increases with the load, P_{hit} is a function of λ_1. We use a simple approach to computing P_{hit}. We take the main memory size plus the NVRAM size and divide it by the accessed data set size. In terms of our model, the splitting a Poisson stream results in two Poisson streams, λ_2 and λ_3. The rate of requests going to the disks is easily derived as $\lambda_3 = P_{miss}\lambda_1$

The disks are modeled by an M/M/m queue where m is equal to the number of disks. We have empirically observed using the `iostat` command an unloaded average service time of 12 ms for the IBM drives. We use the same value to model the Seagate drives.

We fold the remaining components into a fixed delay center with a delay of D. These components include the overhead in the client operating system, fixed costs in the server, and the network latency. The use of fixed delay greatly simplifies the model, allowing us to focus on the important elements of the server. We can still obtain reasonable accuracy using a fixed delay center, however. We empirically observed a D of 3.4 msec. This fixed delay parameter was obtained by observing a small request rate of 300 op/sec on the RAID system. At that rate, the entire workload fits into memory, so nearly all disk requests have been eliminated.

	SCSI		RAID	
Ops/sec	200-1050		500-1400	
	Q-model	measured	Q-model	measured
Slope μsec per op/s	8.5	14.3	10.4	18.9
Y-intercept	8.0	4.52	-6.8	-0.04
Base	8.6	7.3	4.3	4.5
r^2	0.93	0.99	0.98	0.96

Table 1: **Linear Regression Models & Accuracy.** *This table demonstrates linear regressions of the SFS queuing-theoretic models and measured data. The table shows the slope of the SFS curve, (increase in response time vs load), the Y-intercept, the base performance at 200 and 500 ops/sec, and the coefficient of determination (r^2).*

4.2 Baseline Model Accuracy

Figure 4 shows the accuracy of our simple queuing model compared to the measured data for our baseline systems. The baseline systems have the minimum L, o, and G, and thus maximum performance in all dimensions. In order to measure the slope of the SFS curves, we performed a linear regression on a range of measured data (200-1050 for the SCSI and 500-1400 for the RAID). Table 1 shows that within these ranges a linear model is quite accurate; the r^2 values are 0.99 (SCSI) and 0.96 (RAID).

At a qualitative level, we can see that the NFS cache sizes have a significant impact on the shapes of both the measured and modeled systems. Below 500 ops/sec for the RAID, the SFS curve is fairly flat because the cache is absorbing most of the requests. The SCSI system, with its small cache, has a continuously rising curve. The slope of the RAID is much steeper than the SCSI system for exactly the same reason—differences in cache size. In the RAID, the miss rate increases much more rapidly than in the SCSI system, which already has a high miss rate.

At a more quantitative level, across the entire range of throughputs the relative error of the queuing model is at worst 24% for the SCSI and 30% for the RAID. This is reasonably accurate considering the simplicity of the model, e.g., we do not model writes bypassing the file cache. Unfortunately, the queuing model consistently under predicts the slopes of the SFS curve. Linear regressions of the queuing model predict slopes of 8.5 (SCSI) and 10.4 (RAID) μsec per op/sec. These are substantially lower than the 14.3 and 18.9 μsec per op/sec for the measured slopes.

The shape of the inflection point is a second inaccuracy of the model. In the SCSI system, the measured inflection point is quite muted compared to the modeled curve. The last point of the modeled SCSI curve, which has no measured counterpart, shows a rapid rise in response time in the 99+% utilization regime. The real system, however, will not enter into that regime. We explore the effects of high utilizations in Section 5.2.1.

In spite of the inaccurate slopes and inflection point, the queuing model is reasonably accurate for most of the operating regime. Only at very high utilizations does it deviate much from the measured values. Interestingly, the r^2 values in Table 1 show that the live system behaves in a linear fashion across almost all of the operating regime, more so than the model would predict.

For the purposes of capacity planning, the queuing model may be quite acceptable because operating at the extremes of the performance ranges is undesirable. A lightly loaded system wastes resources, while a system operating near capacity results in unacceptable response times.

4.3 Expected Sensitivity

Figure 4 showed that the queuing model performs reasonably well in the baseline network case across a range of loads. Given that we have reasonable confidence in our model, we can use it predict the impact of changing each LogGP parameter.

The delay center captures the network latency term. We thus model an increase in L as a linear increase in D, thereby changing the base response time. Each μsec of added L should add 2 μsec to the base response time, because each operation is a synchronous request-response pair. The model also predicts that an increase in L should have no effect on the slope or saturation point. In the next section, we see that our model predictions for the slope and saturation point are accurate for a wide range of L values, but not for extreme ranges. We also see that the model consistently under-predicts the sensitivity of the base response time to L.

Increasing o we expect changes to all three components of the SFS signature. The response time should increase because of the client overhead encapsulated in the D parameter and increased service time on the CPU. We expect the slope to increase due to queuing delays at the CPU. The most important effect of o however, is that the saturation point may be reached at lower load. Because of the increased service time on the CPU, it will reach maximum utilization sooner. If some other component of the system were the bottleneck however, we may see a region where the saturation point is insensitive to o. Our model, however, predicts the CPU will be the bottleneck. We model the relationship between the saturation point and overhead as:

$$Saturation = \frac{1}{Serv + 2.4o}$$

Where $Serv$ is the average CPU service time per operation previous measured in Section 4.1. The coefficient of 2.4 is the average number of messages per NFS operation. We model two messages per operation: a request and a reply. However, as o is incurred on every message, we also model two extra fragments per read or write due to MTU effects. Given the frequency and size of reads and writes, the MTU effects raise the constant to 2.4. The next section will show our model of sensitivity to o to be quite accurate.

The bandwidth, $\frac{1}{G}$, is not captured well by any single parameter of the model. If we assume that requests are uniformly distributed in time, G will have no effect until $\lambda_1 > \frac{1}{G}$. Indeed, this is a good test to see if requests are bursty or not. If requests are bursty then we expect that the NFS system would be quite sensitive to changes in G.

5 Sensitivity

Given our methodology and NFS characterization, we now empirically quantify the effect of varying LogGP parameters. To this end, we independently vary each of the parameters in turn and observe the resulting SFS curves. For each parameter, we attempt to explain any observed changes to the SFS signatures based on the model developed previously. In addition, we isolate the most significant sensitivity for each parameter. For latency this is change in base response time as a function of L. For overhead, it is the change in saturation point as a function of o.

5.1 Latency

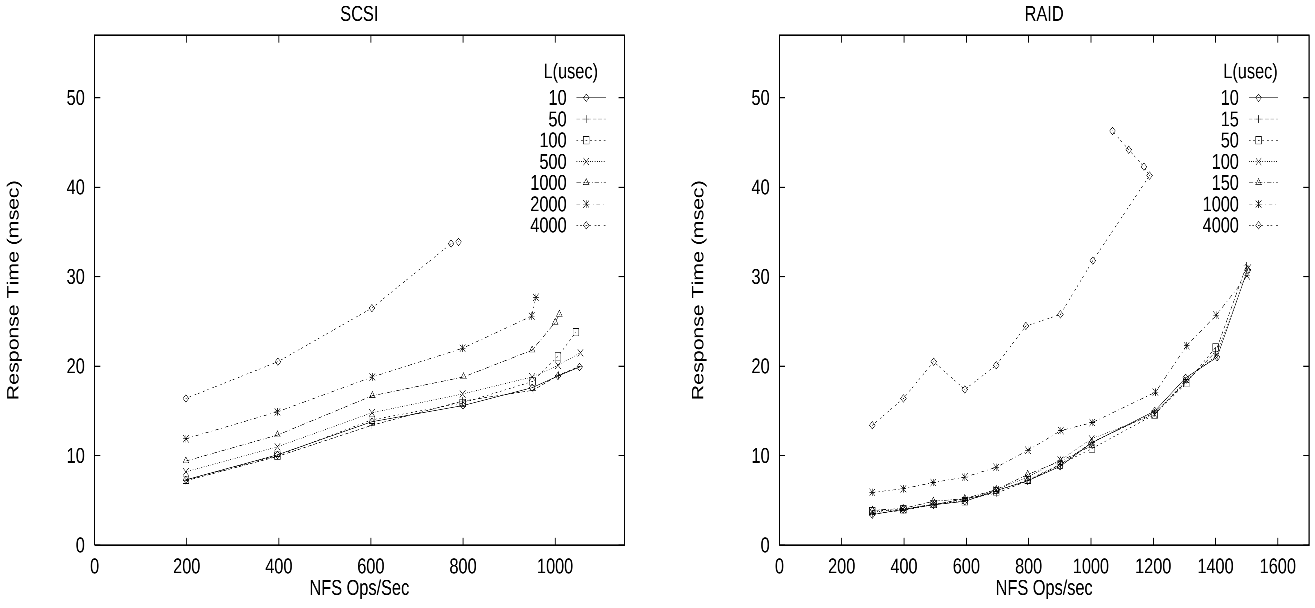
Historically end-to-end latency is often thought of as the critical parameter for NFS performance [4, 19]. In terms of the LogP model, the typical definition of latency includes **both** L and o. In this section, we examine solely the L term. By focusing on latency alone, we can better quantify the effects of the network itself, rather than mixing the effects of the network and end-system.

5.1.1 Response to Latency

Figure 5(a) shows a range of SFS curves resulting from increasing L for the SCSI system. Likewise, Figure 5(b) shows the results for the RAID. The range of L has been scaled up from a baseline of 10 μsec to 4 msec. For comparison, most LAN switches have latencies in the 10's of μsec. Most IP routers, which would be used to form a campus-wide network, have latencies of about a millisecond. Thus the range explored in Figure 5 is most likely what one might find in an actual NFS network. We will explore the effect of very high WAN-class latencies in section 5.4.

All of the SCSI curves "double back" beyond the saturation point forming a "hook", like the 4 ms RAID curve in Figure 5(b). We have truncated the SCSI curves at the saturation point to increase readability, but present the full RAID data. Because the SFS benchmark reports the response time vs. delivered throughput, as opposed to offered load, attempts to exceed the saturation point can result in *fewer* operations per second than attempted. We will explore this effect in greater detail in the discussion of overhead.

As predicted by the queuing model, the measured data shows the primary effect of increased L is to raise the base response time. Also, as predicted by the model, the slope does not change. Modest changes in L do not affect the saturation point. However, a high L can cause the saturation point to fall, as shown by both the 4 msec curves. The reason for the drop is that insufficient parallelism exists due to lack of client processes. We have tested this hypothesis by increasing the number of load generator processes on the client. An unusual side effect of increasing the number of load generators is a modest increase in the slope of the SFS curve. We therefore use the minimum number of load generators that can saturate the system in the baseline case even if it results in lower saturation points as we scale L.



(a) (b)

Figure 5: **Sensitivity to Latency.** *This figure plots the SFS curves as a function of latency in microseconds. Measurements for the graph on the left were taken on the SCSI system, while measurements for the graph on the right were taken on the RAID system.*

Returning to the base response time, a key question is what the rate of increase with respect to L is. That is, for each μsec of L added, what is the corresponding increase in response time? The next section explores this question in greater detail.

5.1.2 Sensitivity to Latency

Figure 6 shows the sensitivity of response time as a function of L for a range of throughputs, i.e., each curve is a vertical slice through Figure 5(a). Two distinct sensitivity regions are observable. Figure 6(a) shows the first region has a constant sensitivity of 3 μsec of response time for each μsec of added L between 150 - 4000 μsec. This is quite a bit higher than the 2 predicted by the model in Section 4.3. Zooming in, Figure 6(b) shows a completely insensitive region between 10 and 150 μsec. For the same range of L, the same result applies to the RAID as well.

An important result is that in the sensitive region, all the sensitivity curves are a constant 3 across an order magnitude change in L. Given that the system is responding in a linear fashion, there may be an accurate way to model it. However, the constant of 3 is quite a bit higher than the constant 2 predicted by our simple model. A more complex model is needed to account for the discrepancy.

The insensitive region has important implications for switch and interface designers as these operate in the 10's of μsec region. From an NFS perspective, a LAN switch adding 10 μsec of delay per hop have little observable impact on NFS performance.

5.2 Overhead

Software overhead, much neglected in networking analysis, permeates the design space because it affects all aspects of the SFS signature curve. We focus our analysis efforts, however, on its effect on the saturation point. Not only are these effects likely to be the most pronounced, but they also will greatly impact the machine size needed to sustain a given load.

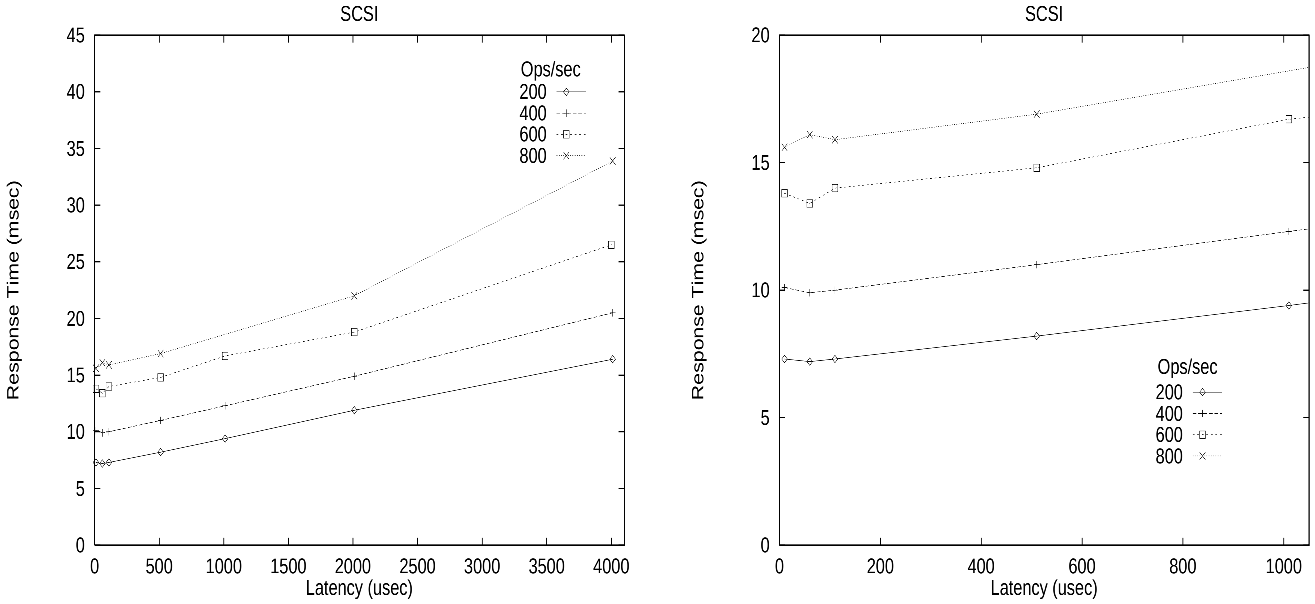
5.2.1 Response to Overhead

Figure 7 shows the SPECsfs curves for both the SCSI and RAID systems while scaling overhead from a baseline of 80 μsec. For the SCSI we have truncated the results at the saturation point to make the graph more readable.

Figure 7 shows that the base response time increases as we scale o, and the measured results are close to the model predictions. The slope of the SFS curve is fairly insensitive to o until throughput is near the saturation point. A queuing model gives us nearly the same result; the slope of the SFS curve will not change drastically with respect to o. The most dramatic effect of o, however, is on the saturation point.

Figure 7(b) shows what happens to the saturation point in the RAID when system capacity is exceeded; both response time and throughput degrade slightly as offered load exceeds the saturation point. A lesser version of this effect was observable as we scaled L. An interesting open question is how well the system responds to these extreme conditions, i.e., how much of the peak performance is obtainable when the offered load is 150% of peak? Queuing theoretic models tell us that response time should increase to infinity as offered load nears 100%. Figure 7(b) shows that in a real system (which is a closed system) the response time hovers around an overhead-dependent maximum while the delivered throughput slowly decreases. Feedback loops built into the RPC layer, based on algorithms in [12], keep the system out of the realm of very high response times, instead forcing the entire system towards lower throughput. The algorithms are quite effective; rather than a complete system breakdown we observe small degradations in throughput and response time. A full investigation of these effects, however, is beyond the scope of this work.

Because we are scaling a processor resource, the lower saturation point must be due to the higher service time of the CPU. In the next sections we will explore the nature of the saturation point. We first derive the sensitivity curve and then examine the components of the



(a) (b)

Figure 6: **Latency vs. Response Time.** *This figure plots response time as a function of latency in microseconds. Measurements were taken on the SCSI system. The graph on the left shows a range of L up to 4000 μsec. The graph on the right shows that up to 150 μsec there is little sensitivity to L.*

service time for both the SCSI and RAID systems.

5.2.2 Sensitivity to Overhead

Figure 8 shows the relationship between overhead and throughput. The modeled line shows where the CPU reaches 100% utilization in the model presented in Section 4, while the measured line is derived from the results in Figure 7. The most interesting aspect of both systems is that the peak performance drops immediately as we add overhead; unlike the response to latency, there is no insensitive region. Therefore, we can easily conclude that the CPU is the bottleneck in the baseline system. Also for both curves, the response to overhead is non-linear, i.e., for each μsec of added overhead, the peak drops off quickly and then tapers out.

To determine the accuracy of the model, we performed a curvilinear regression against the overhead model in Section 4.3. The r^2 values of .99 for the SCSI and .93 for the RAID show that our model is fairly accurate. The sensitivity to o agrees well with the model.

5.2.3 Examining Overhead

The SCSI and RAID system use the same CPU, operating system, network, and nearly the same number of disks. Yet RAID's saturation point is much higher. An obvious question is the reason for the lower performance of the SCSI system. We examined the kernel times with the `kgmon` utility, which instruments running kernels by statistical sampling. Figure 9 shows the observed percentage of time in each kernel sub-system for a 300 second period. Most of the performance difference between the two systems was due to the device drivers. The FAS SCSI drivers spend an average of 150 μsec per NFS operation while the RAID drivers spend an average of only 36 μsec. A second interesting result is that a significant amount of the service time (20% and 26%) are general kernel procedures which do not fall into any specific category. There are a myriad of these small routines in the kernel code. Getting an order of magnitude reduction in the service time would require reducing the time of many sub-systems. Much as was found in [5, 14] there is no single system accounting for an overwhelming fraction of the service time.

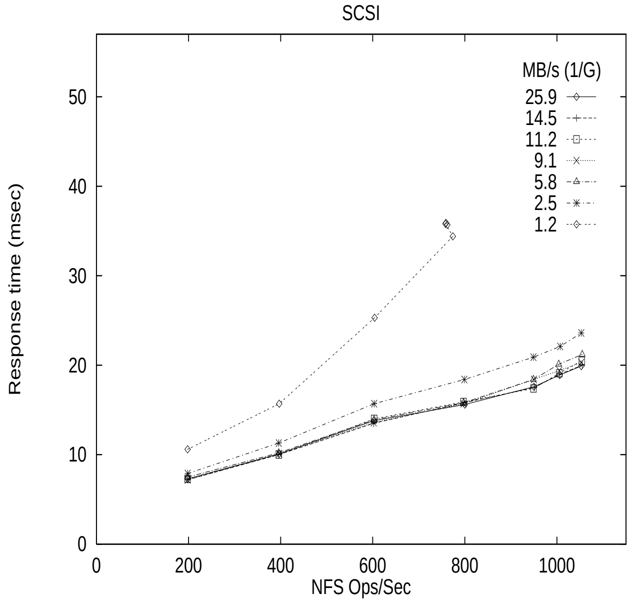
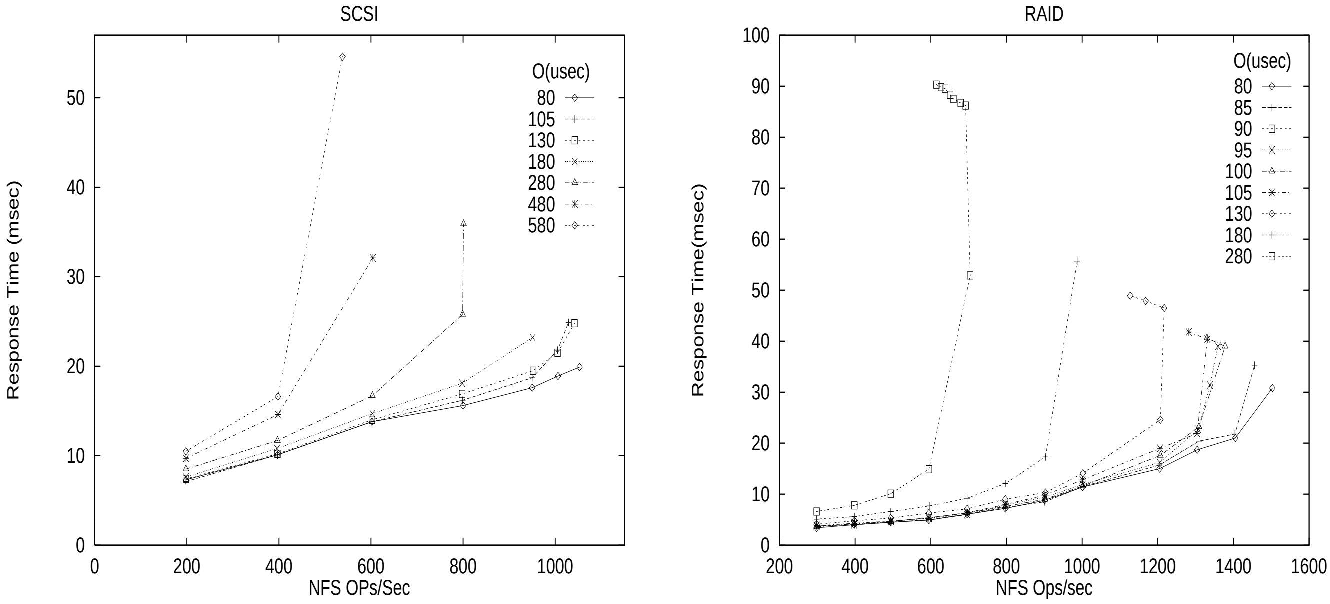


Figure 10: **Sensitivity to Gap.** *This figure plots the SFS curves as a function of Gap in microseconds. Measurements for the graph were taken on the SCSI system.*

5.3 Bulk Gap

We choose to examine sensitivity to bulk Gap, G, as opposed to the per-message rate g. First, networking vendors often tout per-byte bandwidth as the most important metric in comparing networks.



(a) (b)

Figure 7: **Sensitivity to Overhead.** *This figure plots the SFS curves as a function of overhead in microseconds. Measurements for the graph on the left were taken on the SCSI system, while measurements for the graph on the right were taken on the RAID system.*

Using our apparatus we can quantify its sensitivity (and thus importance). Secondly, for the SPECsfs benchmark, g is quite low (in the 1000's msg/sec range) and is easily handled by most networks.

Unlike overhead, which is incurred on every message, sensitivity to Gap is incurred only if the data rate exceeds the Gap. Only if the processor sends data in an interval smaller than that specified by G will it stall. The clients and server could potentially ignore G entirely. At one extreme, if all data is sent at a uniform rate that is less than G we will not observe any sensitivity to Gap. At the other extreme, if all data is sent in bursts then we would observe maximum sensitivity to Gap.

Because SPECsfs sends messages at a controlled rate, we would expect that message intervals are not bursty and the benchmark should be quite insensitive to Gap. Figure 10 shows that this is indeed the case. Only when the bandwidth ($\frac{1}{G}$) falls from a baseline of 26 MB/s to a mere 2.5 MB/s do we observe any sensitivity to G. We are thus assured that the SFS benchmark is not bursty.

Measured production environments however, are quite bursty [9, 16]. Our measured sensitivity to G is thus lower than what one might expect in a production environment. We explore the implications of bursty networks to the sensitivity of G in more detail in Section 6.

Figure 10 shows queuing delays at very low bandwidths that are not captured by the model. There is a slight increase in slope at a bandwidth of 2.5 MB/s, and at 10 Mb/s Ethernet speeds (1.25 MB/s) there is a noticeable increase in slope. Replacing the simple delay center with a more sophisticated queuing network would capture these effects. However, given the low bandwidths at which these effects occur, we have made a quite reasonable tradeoff between model simplicity and accuracy.

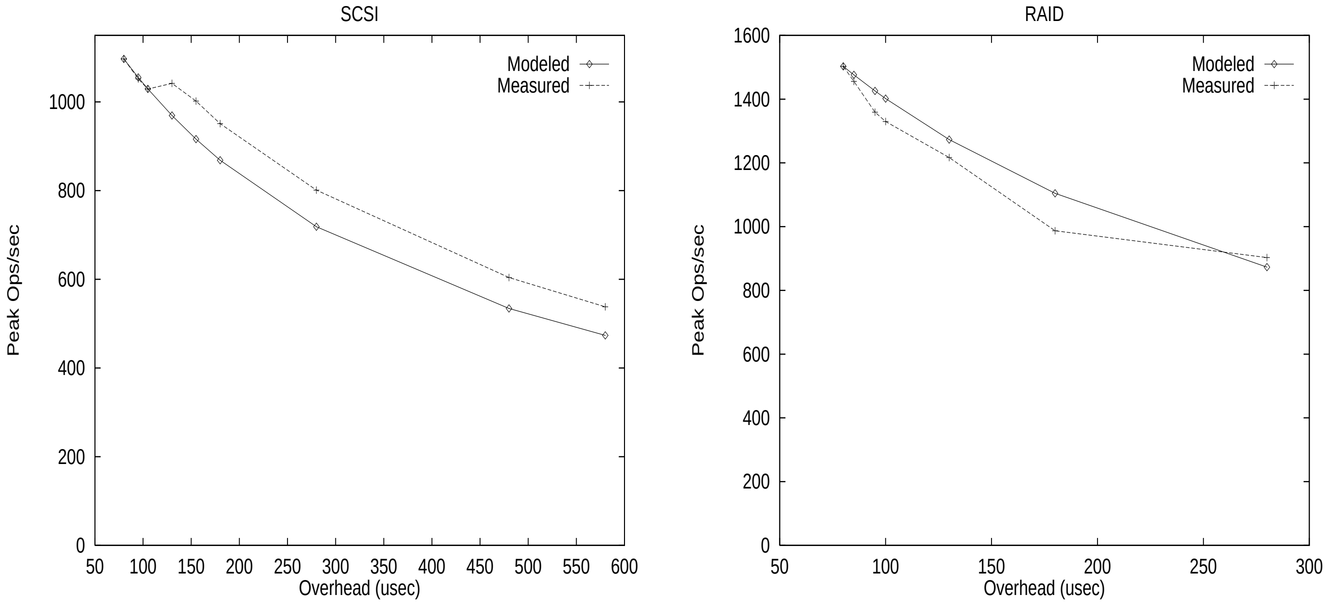
5.4 High Latency

The original NFS protocol was not designed to operate in environments with very high L. NFS version 3 added several latency tolerating techniques, most notably asynchronous writes [20]. In this section, we examine the effects of very high L, in the 10's of millisecond range. For example WANs typically have an L ranging from 10's to 100's of milliseconds.

Figure 11 compares the relative effectiveness of NFS version 2 running on UDP, a typical configuration, to version 3 running on TCP for networks with high L. The experiment varies both the NFS version and network transport at once to better understand the total impact of an upgrade. Typically, operating systems that ship with version 3 also allow TCP as a transport layer. Both the throughput and response times between NFS version 2 and version 3 are not comparable; thus we examine the percentage of performance loss as we scale L.

Figure 11(a) shows, as expected, that the "classic" NFS V2/UDP performance over WAN latencies is dismal. First, the base response time is hyper-sensitive to high latency in that it is much greater than one would predict from the simple model. Second, very little of the peak load is obtainable. NFS Version 3 over TCP is able to handle long latencies much better than version 2 over UDP. Figure 11(b) shows that even at an L of 40 msec (a round trip of 80 msec), Version 3/TCP can sustain 50% of the peak throughput without adding extra clients.

A notable effect on both versions is that average response time *decreases* as the load increases. This could be caused by a number of effects. One possible effect could be the interaction of a light workload with the RPC and TCP transport protocols. These algorithms constantly probe the network looking for more bandwidth. Under a light workload however, an insufficient number of packets may be sent for the protocol to reach a stabilization point in its time-out/re-try algorithm. As both curves are consistent with this theory, it begs the question as to the performance of these algorithms [12] under a very light load.



(a) (b)

Figure 8: **Peak Throughput vs. Overhead.** *This figure plots the saturation point as a function of overhead in microseconds. Measurements for the graph on the left were taken on the SCSI system, while measurements for the graph on the right were taken on the RAID system.*

6 Discussion

Given that we can isolate the sensitivity of NFS to network performance, we may consider the implications of our sensitivity results to L, o and G for emerging networks, the utility of the queuing model, and the longer term implications of our study. To make our results concrete, we perform a short analysis of two servers using data in the SPEC website framed in the context of this work.

Our results show that for typical SAN and switched LAN environments, the latency is quite satisfactory for NFS. Latencies under 150 μsec are easily obtainable, even when cascading switches, and further reductions will provide little benefit. For campus-wide networks with several routers, obtaining this level may be difficult in the short term.

In the WAN range, we have seen that the changes to NFS version 3 indeed improve throughput. However, such latencies are still a significant performance drag, as was also found in [4]. Qualitatively, the changes in Version 3 have raised the level of NFS performance over WANs from "unusable" to merely "slow". Even with these enhancements however, it may not be economically viable to use NFS over WAN ranges. Given the low cost of disk storage compared with WAN links, it may make more sense to replicate the entire data-set. Even for large amounts of data, the storage requirements are cheap compared with the recurring costs of WAN links.

Overhead continues to be the performance limiter and this is where significant performance improvements could be made. Although the networking overhead was only 20% of the entire service time, that does not mean that attempts to reduce o will yield marginal results. Indeed, networking overhead is one of the primary components of the service time. However, a number of subsystems must be improved at once for significant progress to be made. For example, a combination of novel overhead reducing interfaces between major OS sub-systems, disks and network interfaces might yield significant improvements. A similar conclusion as was found in [9] as well. The study examined 3 points in the networking space (ATM, Autonet and FDDI), rather than systematically varying overhead. However, it is encouraging that two different studies have come to the same conclusions by much different methods.

The SPECsfs workload has minimal bandwidth needs and is quite regular; generating traffic on the order of single MB/s. However, real networks exhibit quite bursty behavior and thus bandwidth requirements would be higher, but not into the gigabit range. Network technologies such as switched 100Mb Ethernet and 155 Mb ATM provide plenty of bandwidth for NFS workloads. Given that most NFS packets are quite small, overhead, rather than bandwidth, will still be the dominant factor facing future network designers.

In the longer term, the latency reductions from IP switches will have a large impact on NFS. The order of magnitude drop in L from the millisecond to the 10-20 μsec region [24, 25] will expand the range of NFS to a much wider area. Recent switches also offer increased port densities, ranging to 100's of ports at 100 Mb Ethernet speeds. A network composed of these low-latency, high-density IP switches would expand the range of NFS service to a whole campus, even multiple campuses, instead of its traditional domain, a building. The implications of such an expansion are interesting; NFS service could reach a much larger number of machines than previously possible.

Simple queuing models are quite effective in analyzing the behavior of an NFS server. We were able to model changes in response time, slope and saturation point for a variety of parameters. However, more investigation is needed to better describe the effect of latency on response time.

We empirically measured and validated the inputs to the model. One could, however, obtain close to the same inputs for a specific configuration by looking at the published data [23]. Using the SPEC data and assuming our model framework, it is relatively straightforward to deduce the parameters of the queuing model for a specific configuration from the published SFS curves.

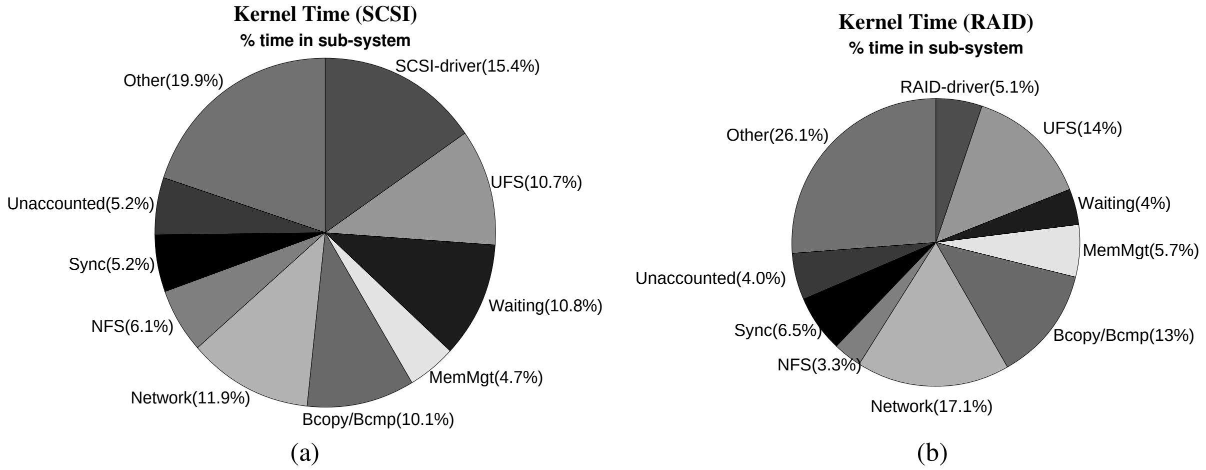


Figure 9: **Time Breakdown Near Peak Op/sec.** *These charts show the percentage of time spent in each sub-system when operating near the saturation point. Measurements for the graph on the left were taken on the SCSI system at 1000 ops/sec, while measurements for the graph on the right were taken on the RAID system at 1400 ops/sec. The area of each chart shows the relative time per operation, including idle time (i.e. waiting on I/O), of 1 msec for the SCSI and 714 μsec for the RAID.*

Examining the NetApp F630 and AlphaServer 4000 5/466 SPECsfs results is instructive. They have roughly the same CPU (500 MHz Alpha), but the Alphaserver has twice the main memory and disks as the NetApp box. The NetApp box however, has half the base response time, a much lower slope, and a higher saturation point. Putting the results into the context of this work, we can conclude that Network Appliance was quite successful in their bid to reduce overhead via a specialized operating system [10]. Another approach to obtaining a higher saturation point is to add processors, demonstrated by the 18 CPU Sun system. Such an approach would not reduce the base response time, however, unless the operating system can parallelize a single NFS operation.

7 Conclusions

We have developed a simple model and empirical method for exploring the sensitivity of NFS to various aspects of communication performance. Simple queuing models are reasonably accurate in characterizing NFS server performance. Our parameterized approach was quite effective in exploring the network design space as well. By varying each LogP component in turn we can pinpoint the sensitivity to each software/hardware component in isolation.

Our results show that NFS is quite sensitive to processor overhead. Significant reductions in overheads will be critical for future servers to utilize emerging multi-gigabit networks. Overhead reductions will have to come from two places: improvements to networking stacks and the local filesystem. However, a significant component of the overhead in our testbeds came from general kernel code with no easily identifiable source sub-system.

We found that NFS is quite insensitive to network bandwidth. Our choice of benchmark offers a very controlled load on the server. This fact combined with the small size of most NFS operations accounts for the low sensitivity to bandwidth. Given that real traffic is bursty our sensitivity result to bandwidth is lower than what would be seen in a production environment. Even assuming the worst-case behavior, however, gigabit LANs will not bandwidth-limit NFS.

NFS is quite insensitive for latencies of a typical high-performance LAN. Newer LANs will have even lower latencies as switch degrees increase. However, when crossing into the current IP routing regime of milliseconds latency does have an impact. It will be interesting to see if NFS is used over a wider area as IP switching becomes more common.

Our results show that NFS version 3 exhibits better tolerance to high latencies than version 2. In addition, at very high latencies our model breaks down; the system exhibits a hyper-sensitive response at WAN-class latencies on the order of 10's of milliseconds.

Acknowledgments

We would like to thank John Ousterhout for motivating this study. We would also like to thank Brian Wong, Drew Rosselli and Randy Rettburg for their comments.

References

[1] ALEXANDROV, A., IONESCU, M., SCHAUSER, K. E., AND SCHEIMAN, C. LogGP: Incorporating Long Messages into the LogP model - One step closer towards a realistic model for parallel computation. In *7th Annual Symposium on Parallel Algorithms and Architectures* (May 1995).

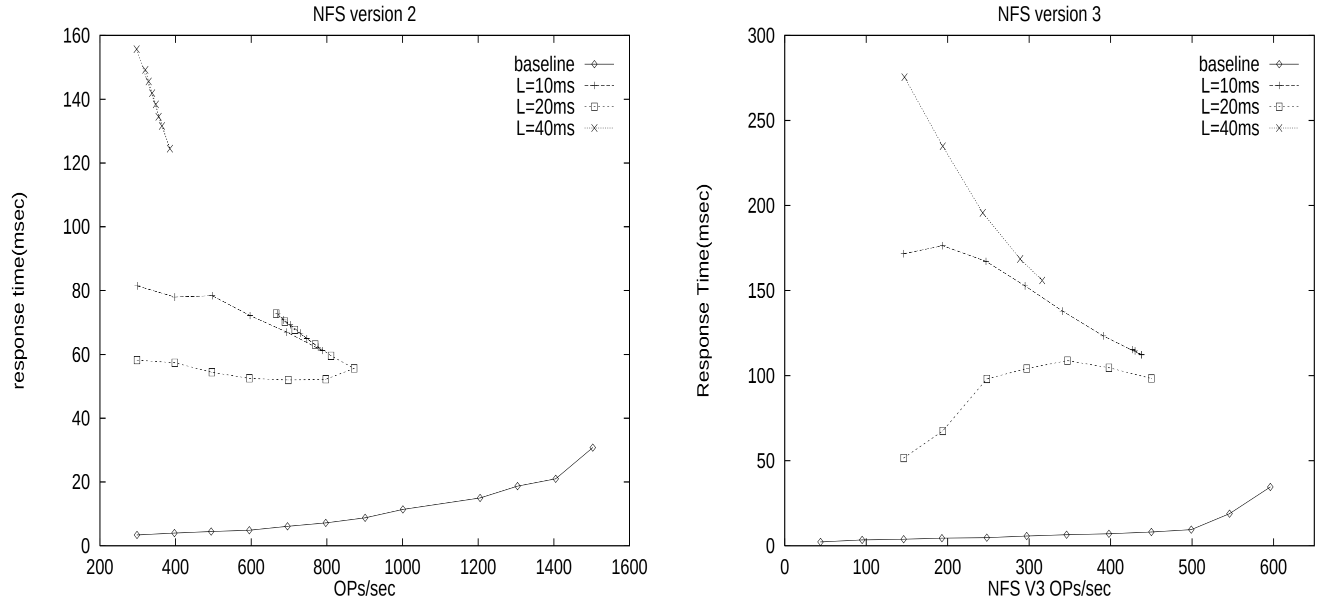
[2] BLACK, R., LESLIE, I., AND MCAULEY, D. Experience of Building an ATM switch for the Local Area. In *Proceedings of the ACM SIGCOMM '94 Conference on Communications Architectures and Protocols* (London, UK, Sept. 1994), pp. 158–167.

[3] BODEN, N. J., COHEN, D., FELDERMAN, R. E., KULAWIK, A. E., SEITZ, C. L., SEIZOVIC, J. N., AND SU, W.-K. Myrinet—A Gigabet-per-Second Local-Area Network. *IEEE Micro 15*, 1 (Feb. 1995), 29–38.

[4] CHANG, K., MORRIS, R., AND KUNG, H. T. NFS Dynamics Over Flow-Controlled Wide Area Networks. In *Proceeding of the 1997 INFOCOMM* (Kobe, Japan, Apri 1997), pp. 619–625.

[5] CLARK, D. D., JACOBSON, V., ROMKEY, J., AND SALWEN, H. An Analysis of TCP Processing Overhead. *IEEE Communications Magazine 6* (June 1989), 23–29.

[6] CULLER, D. E., KARP, R. M., PATTERSON, D. A., SAHAY, A., SCHAUSER, K. E., SANTOS, E., SUBRAMONIAN, R., AND VON EICKEN, T. LogP: Towards a Realistic Model of Parallel Computation. In *Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (1993), pp. 262–273.



(a) (b)

Figure 11: **Effects of Very Long Latency.** *This figure plots the SFS curves as a function of very high latencies on the RAID. Measurements for the graph on the left were taken using NFS Version 2 over UDP, while measurements for the graph on the right were taken using NFS Version 3 running over TCP. The figure is designed to show the relative performance degradation for each version as neither the operations/sec or the response times between versions is comparable.*

[7] GILLETT, R. B. Memory Channel Network for PCI. In *IEEE Micro* (Feb. 1996), vol. 16, pp. 12–18.

[8] GUSELLA, R. A Measurement Study of Diskless Workstation Traffic on an Ethernet. *IEEE Transactions on Communications 38*, 9 (Sept. 1990), 1557–1568.

[9] HALL, J., SABATINO, R., CROSBY, S., LESLIE, I., AND BLACK, R. Counting the Cycles: a Comparative Study of NFS Performance Over High Speed Networks. In *Proceedings of the 22nd Annual Conference on Local Computer Networks (LCN'97)* (Minneapolis, MN, Nov. 1997), pp. 8–19.

[10] HITZ, D., LAU, J., AND MALCOLM, M. File system design for an NFS file server appliance. In *Proceedings of the Winter 1994 USENIX Conference* (San Francisco, CA, Jan. 1994), pp. 235–246.

[11] HORST, R. TNet: A Reliable System Area Nework. *IEEE Micro 15*, 1 (Feb. 1995), 37–45.

[12] JACOBSON, V. Congestion avoidance and control. In *Proceedings of the ACM SIGCOMM '88 Conference on Communications Architectures and Protocols* (Stanford, CA, Aug. 1988), pp. 314–329.

[13] JAIN, R. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[14] KAY, J., AND PASQUALE, J. The Importance of Non-Data-Touching Overheads in TCP/IP. In *Proceedings of the 1993 SIGCOMM* (San Francisco, CA, September 1993), pp. 259–268.

[15] LAZOWSKA, E. D., ZAHORIAN, J., GRAHAM, G. S., AND SEVCIK, K. C. *Quantitative System Performance : Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Englewood Cliffs, N.J, 1984.

[16] LELAND, W. E., TAQQU, M. S., WILLINGER, W., AND WILSON, D. V. On the Self-Similar Nature of Ethernet Traffic. *IEEE Transactions on Networking 2*, 1 (Feb. 1994), 1–15.

[17] MARTIN, R. P., VAHDAT, A. M., CULLER, D. E., AND ANDERSON, T. P. The Effects of Latency, Overhead and Bandwidth in a Cluster of Workstations. In *Proceedings of the 24th International Symposium on Computer Architecture* (Denver, CO, June 1997).

[18] MOGUL, J. C. Network locality at the scale of processes. *ACM Transactions on Computer Systems 10*, 2 (May 1992), 81–109.

[19] OUSTERHOUT, J. K. Personal communication, Jan. 1997.

[20] PAWLOWSKI, B., JUSZCZAK, C., STAUBACH, P., SMITH, C., LEBEL, D., AND HITZ, D. NFS Version 3 Design and Implementation. In *Proceedings of the Summer 1994 USENIX Conference* (Boston, MA, June 1994), pp. 137–152.

[21] SHEIN, B., CALLAHAN, M., AND WOODBURY, P. NFSSTONE - A Network File Server Performance Benchmark. In *Proceedings of the 1989 USENIX Summer Conference* (Baltimore, MD, June 1989), pp. 269–274.

[22] STANDARD PERFORMANCE EVALUATION CORP. *SPEC SFS97 Benchmarks*, 1997. http://www.specbench.org/osg/sfs97.

[23] STANDARD PERFORMANCE EVALUATION CORP. *SPECsfs97 Press Release Results*, 1997. http://www.specbench.org/osg/sfs97/results.

[24] STRATEGIC NETWORKS CONSULTING. *Fore Systems Intelligent Gigabit Routing Switch Custom Test*, Oct. 1998. http://www.snci.com/-reports/ESX-4800.pdf.

[25] STRATEGIC NETWORKS CONSULTING. *Packet Engines PowerRail 5200 Enterprise Routing Switch Custom Test*, Apr. 1998. http://www.-snci.com/reports/packetengines.pdf.

[26] WITTLE, M., AND KEITH, B. E. LADDIS: the Next Generation in NFS File Server Benchmarking. In *Summer 1993 USENIX Conference* (Cincinnati, OH, June 1993), pp. 111–128.

[27] WONG, B. *Configuration and Capacity Planning for Solaris Servers*. Prentice-Hall, 1997.