The Advancement of NFS Benchmarking: SFS 2.0

David Robinson – Sun Microsystems, Inc.

ABSTRACT

With the release of the Standard Performance Evaluation Corporation's (SPEC) System File Server (SFS) Release 1.0 benchmark suite in April 1993, the characterization of NFS server performance grew from a small and not widely accepted set of benchmarks, to a single industry standard benchmark. This paper will provide a historical look at the success of SFS 1.0[1] and how it has driven server capabilities, describe SFS 1.0's shortcomings, and detail the design and rationale behind the development of SFS 2.0.

Terminology

The SFS benchmark suite has been referred to by a number of different names that has lead to confusion. For this paper, the term SFS shall refer to the suite and in particular the framework used to drive the workloads, SFS 1.0 or SFS 2.0. Each suite is composed of one or more workloads which reflect the version of protocol used. SFS 1.0 contains the 097.LADDIS workload and SFS 2.0 contains the 162.nfsv2 and 163.nfsv3 workload. For this paper, the 097.LADDIS workload will be referred to as LADDIS, and 162.nfsv2 and 163.nfsv3 as V2 and V3 respectively.

SFS 1.0

Background

SFS is a synthetic benchmark used to measure the throughput and response time of an NFS server over a variety of load levels. The benchmark uses multiple physical clients, called load generators, each containing multiple load generating processes. Each load generating process is designed to represent multiple real world clients. A load generating process contains its own NFS and RPC protocol stacks, eliminating any effects due to different NFS client implementations, making it possible to use a variety of different hardware and operating system platforms and still have comparable results. The load generators send a controlled stream of time-stamped NFS requests to the server and measure the precise response times producing a detailed report of each type of operation, the overall throughput and average response time. The server is treated as a black box and the benchmark relies on no services on the server beyond the standard NFS protocol.

The SFS workload does not model any specific user application or any specific environment, but is designed to present the server with a series of requests to simulate the aggregation of many different applications and clients. The workload was designed through a series of studies of actual NFS traffic to servers used in a wide variety of environments.

The basic framework of SFS 2.0 has not significantly changed from the original SFS 1.0 release described in detail in the 1993 Usenix paper by Whittle and Keith [Whittle93]. The primary focus of SFS 2.0 was to update the accuracy of the workload and this paper will focus on those changes.

Historical Perspective

The SPEC SFS 1.0 benchmark suite and its sole component, the 097.LADDIS workload, has had a significant impact of the NFS server market since its introduction in April 1993. Prior to the release, evaluation and specification of NFS servers were done based on a number of small benchmarks that did not adequately represent real world NFS servers. Some of the problems included results that were greatly influenced by the effects of the client operating system, the lack of a defensible workload, and no common agreed upon standards for testing and reporting of results. The creation of the informal industry wide LADDIS group to address the technical issues and the subsequent incorporation with the SPEC standards body resulted in a benchmark that was accepted by the industry as fair and vendor neutral. The conventional wisdom that if you measure it, it will improve, is validated by the publication of results. Figure 1 shows a thirty fold increase in throughput over the five years results were published. This dramatic increase greatly exceeds the increase in processor performance over the same period. Using the often quoted figure of integer performance increasing every 12 to 18 months, the actual growth of LADDIS results exceeds this by a factor of 2 to 4. This growth can be attributed primarily to the enhancement of system software in efficiency and scalability on multiprocessor systems.

Another measure for the success of SFS has been the demand for results. In addition to the purpose built NFS servers, most vendors of general purpose servers now include SFS as one of the standard metrics marketed at their initial product announcements.

[1] SPEC SFS 1.1 was a bug fix release in 1994 which made no measurable changes to the workload. Both will be referred to as SFS 1.0 in this paper.

Customers are also including the SFS metrics in their minimum specification for both NFS and general purpose server requests for proposals.

Although SFS was designed primarily to produce competitive benchmarking results, the designed in flexibility and tuneability of a wide variety of parameters has enabled it to be used in the sizing of servers and creating application specific workloads. This capability has allowed both customers to evaluate their specific environments and server vendors to tune their systems for a multitude of applications.

Deficiencies

While SFS 1.0 has been extremely popular and a success in the industry, a number of deficiencies were known when it was released or subsequently discovered during its lifetime.

Operations Mix

The basis of the default percentages of each NFS operation (or operations mix) in the LADDIS workload was an unpublished 1986 study of the Sun Microsystems engineering network and also validated by a survey of customer nfsstat data from software development and other similar technical computing environments. The clients in the study were primarily a homogeneous set of workstations which may not reflect the differences in client implementations. Many of the workstations in the study were diskless and had relatively small amounts of physical memory resulting in small caches with high paging and swapping rates over NFS. Very few modern workstations are diskless and now contain large physical memories, resulting in large caches that reduce the need to swap over the network. Casual observation of current server loads appeared to indicate that the LADDIS operations mix was too heavy on I/O operations.

Version 3 and TCP

Version 3 of the NFS protocol was finalized shortly after the initial release of SFS 1.0. It provided some compelling new features, including better file system semantics and performance, resulting in it being chosen as the default protocol version on most major NFS vendor platforms. The benchmark, which measured only NFS version 2, needed to be updated to reflect what real customers were running in their environment.

Simultaneous with the release of NFS version 3, most vendors also introduced TCP as an available transport. While NFS was designed to be transport independent, there is some performance impact when compared to UDP due to the added complexity of managing a reliable transport compared to unreliable transport. With many vendors also making TCP the default transport, there was an increasing demand to characterize that impact.

File Size

The size of files created in SFS 1.0 is uniformly 136 kilobytes (KB) in length. The original LADDIS paper called this "unrealistic" but stated that it would

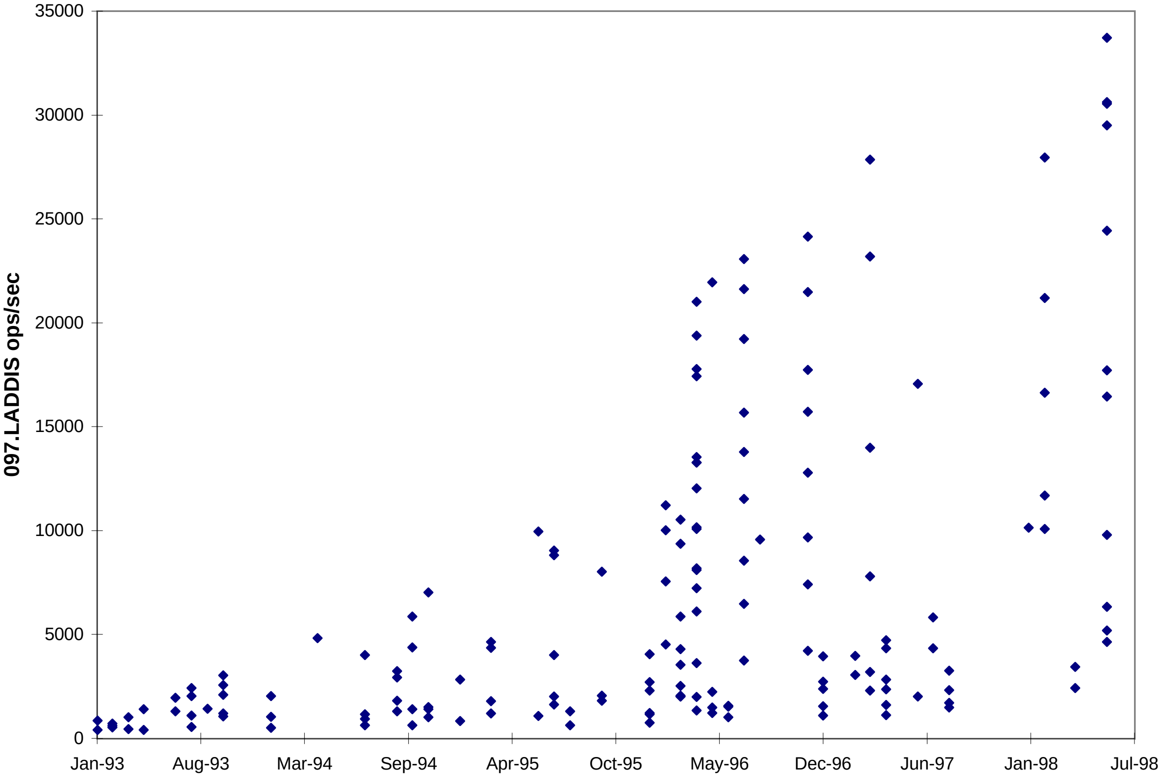


Figure 1: SPEC SFS 1.0 results published 1993-1998

have little impact on the performance model. With the appearance of servers that were specifically optimized for this size of file, the assertion was proven false and needed to be addressed.

Latency Metric

Even though the SFS 1.0 reporting pages include the response time at the peak throughput, the benchmark does not contain a metric to allow a fair comparison of the overall responsiveness of servers. The primary competition between vendors at the time of the release of SFS 1.0 was capacity, as no server was large enough to handle even a moderate sized work group. As server capacity (i.e., aggregate throughput as well as large storage components) has grown to be able to serve very large work groups, if not entire enterprises, a secondary market arose competing for small to medium size servers, using low response times as the key differentiator. The lack of a response time metric in SFS has limited customers ability to make fair comparisons in this new market.

Benchmark Portability

Great effort was taken to insure that SFS 1.0 could be run on a wide variety of client operating systems. While the source code was reasonably portable, it still contained a number of dependencies on the client's native RPC implementation and used parts of the client's native NFS implementation to initialize the file set. With the creation of 64-bit operating systems this reliance on the client NFS and RPC became problematic. Better portability was essential to the continued success of SFS.

Design of SFS 2.0

The design and motivation for SFS 2.0 was driven by two primary factors: creating a workload that more accurately reflects real world servers and updating it to support the recently released NFS version 3. The philosophy of SPEC is to provide standard benchmarks that reflect real world usage [SPEC], unfortunately every synthetic benchmark is a compromise. SFS 1.0 has a number of known deficiencies that were intentionally not addressed before the benchmark was released, and others that were discovered through use. By addressing some of the more significant compromises, customers will have an increased confidence in the reported results.

The changes required to update SFS 1.0 were primarily in the workload generated by the benchmark and the only minor changes were made to the framework and test harness. This is a credit of the work of the original designers.

NFS V2 Operations Mix

The LADDIS operations mix is roughly half file name and attribute operations, one third I/O operations, and the remaining one-sixth spread among the other operations. Throughout the years that SFS 1.0 was being used, the configuration of clients and the implementation of operating systems has steadily evolved. In the 1987 study, many clients were diskless, which is now relatively rare. Diskless clients must page in applications and swap over the network, resulting in a substantial number of I/O operations. This was reflected in the mix by 37% of the operations being reads or writes. Additionally, the average amount of memory configured on a client has risen from 16 or 32 megabytes (MB) of memory, to 128 MB or more. The additional memory allows clients to aggressively cache more data on the client, further reducing I/O operations.

Based on the understanding of the change in the nature of clients and some informal validation based on actual observations, it was believed that the SFS 1.0 workload was too heavy with respect to I/O operations. A formal study was conducted to collect the actual operations mixes of over 750 Auspex servers running NFS V2.

A cluster analysis of the data collected was performed using the SAS statistical tools to determine if there were any significant common mixes present. Only two dominant clusters emerged, the primary cluster (75%) contained an operations mix that was very similar to the existing LADDIS workload, but as predicted, there was a significant reduction in the number of I/O operations. It contained approximately half the read operations and one third the write operations. The secondary cluster (15%) was very I/O intensive and contained a significantly higher percentage of read operations and a slightly higher percentage of write operations than the existing workload. The remaining 10% of the data was spread across a wide variety of mixes, none of which were statistically interesting.

Operation	Cluster 1	Cluster2
null	1%	0%
getattr	25	15
setattr	1	1
lookup	40	23
readlink	7	2
read	11	32
write	5	17
create	2	1
remove	1	0
readdir	7	5
fsstat	1	2

Table 1: NFS V2 Mix Cluster Analysis.

The goal of SPEC benchmarks is to provide customers with a workload that represents a common real world environment. The cluster analysis clearly showed one dominant operations mix, but from the raw data it was not known if this represented one or many types of environments. Each of the servers were further categorized by the type of environment (e.g., software development, office, CAD) and a statistical correlation was run against the clusters. The primary cluster surprisingly showed no significant correlation to any specific type of environment. The secondary cluster showed a correlation to the computer aided design (CAD) environments. CAD is known to be very I/O intensive as it reads and writes large design data sets, confirming the correlation.

The default V2 workload was chosen to match the primary cluster based on it representing the vast majority of servers studied and the lack of correlation to any one environment. A second workload matching the secondary cluster was considered but ultimately rejected. Although this cluster showed a strong correlation to a CAD environment, the workload typically consists of large sequential reads and writes which can be more easily simulated by a simple copy benchmark.

Operation	097.laddis	162.nfsv2	163.nfsv3
getattr	13%	26%	11%
setattr	1	1	1
lookup	34	36	27
readlink	8	7	7
read	22	14	18
write	15	7	9
create	2	1	1
remove	1	1	1
readdir	3	6	2
fsstat	1	1	1
access	-	-	7
commit	-	-	5
fsinfo	-	-	1
readdirplus	-	-	9

Table 2: SFS Operation Mix

NFS V3 Operations Mix

At the time of the development of SFS 2.0, the deployment of NFS V3 within the industry was not yet widespread. Major operating system vendors had not yet, or had only just recently, released support for V3. The detailed statistical analysis used to derived the V2 workload was not possible due to the lack of a significant installed base. Because support of V3 was a key requirement for the release of SFS 2.0, a reasonable approximation of a V3 workload was required.

The operations mix presented to the server represents the aggregation of the file system workload generated by applications running on clients and then converted into NFS operations by the client operating system. In moving from V2 to V3, the application workload remains constant, so a transformation from V2 to V3 is possible if a typical client implementation is known. Using the data from a published comparison of a V2 and V3 client running the Andrew benchmark [Pawlowski94] the V3 operations mix was partially derived. The resulting mix was confirmed through a survey of servers within the Sun engineering network that served both V2 and V3 clients.

The V3 workload appears to be more data than attribute intensive when compared to the V2 workload. This is a result of the the changes in the underlying protocol, in particular, most V3 operations return the attributes of a file, thus reducing the number of getattr operations requested by the client. Through the addition of the readdirplus operation that returns the attributes with each directory entry, the number of lookup operations is also reduced. The result of the heavier V3 operations is a numerically smaller throughput value even though the server can handle more clients.

The commit operation had to be treated specially, as the number of commits is derived from the number of write operations and not measured experimentally. A commit is generated by a client to request that a server flush to stable storage any data written by previous asynchronous write operations. The most common scenario resulting in a commit operation is the closing of a file, when the client wants to free up the resources it may have been caching.

Within SFS, the write requests are typically the same size as the file being written to simulate the common practice of writing an entire file. These requests are then broken down into one or more write operations of the transfer size (8192 bytes) length. For requests less than or equal to the transfer size, only one write operation is generated and it is done synchronously. For all other requests, the commit operation is generated after the final write to simulate the close. With just over half of all write requests generating more than one write operation, 4% of the total workload are commit operations.

Length (KB)	Read	Write
1-7	0%	49%
8-15	85	36
16-23	8	8
32-39	4	4
64-71	2	2
128-135	1	1

Table 3: SFS 2.0 Percentage of File I/O Operations.

File Set

The fundamental properties of the SFS 2.0 file set has not changed from SFS 1.0. A large static set of test files is created consisting primarily of regular data files and directories and a small fixed number of symbolic links. No files are shared between load generating processes which reflects the absence of sharing in real world environments.

The number of data files and directories scales with the requested load like SFS 1.0, but at a ratio of 10 MB of data for each NFS operation/second (op/sec) instead of 5 MB per op/sec. The increase reflects the actual growth in disk capacity and that disk usage closely tracks the capacity growth. A larger file set causes servers to further stress their caching algorithms and disk seek times. The intention behind this was to emulate real world server conditions.

Working Set Size

From the large file set created, a smaller working set is chosen for actual operations. In SFS 1.0 the working set size was 20% of file set size or 1 MB per op/sec. With the doubling of the file set size in SFS 2.0, the working set was cut in half to 10% to maintain the same working set size. Although the amount of disk storage grows at a rapid rate, the amount of that storage actually being accessed grows at a much slower rate. In the IBM study, discussed below, it was found that only 12% of the storage was accessed within the last five days. A 10% working set size may still be too large. Further research in this area is needed.

Variable Sized Files

The size of files created by SFS 1.0 was uniformly fixed at 136 KB. The fixed size allows for a simpler implementation and a less complex selection algorithm to determine which I/O file to use. The size of 136 KB was chosen to be large enough to handle a variety of starting file offsets for read and write operations, while ensuring that the file system on the server allocates and references the first-level indirect file index blocks. At the time, it was believed that this compromise would have little impact on performance. In addition, other studies performed showed that most write operations are sequential, resulting in a high percentage (70%) of operations appending to a file. Unfortunately the combination of fixed size files and the high append ratio allowed servers to significantly increase their results by optimizing the handling of the first-level indirect file index blocks. Because most files in the real world are small and do not utilize indirect blocks, the increase in the benchmarking result from this optimization did not translate into a real world increase in performance. To solve this problem the SFS 2.0 file set uses data files that are of varying size.

A study was performed to determine a typical distribution of file sizes. The IBM engineering network containing over 6 million files and directories with a total storage of over 210 billion bytes was used as the basis of the study. For each file, the size, the number of fragments, the last access time, the last modification time, the length of the name, and the distribution of characters in the name, were recorded.

A simple statistical analysis showed that the median file size was just under 2048 bytes with 76.7% of the files under 8192 bytes. However, these files represented only 4.8% of the space consumed. Only 0.5% of the files were over one megabyte, yet represented 48.8% of the space consumed. The data very closely reflects a study performed three years earlier [Irlam94] with a slight trend towards more files with larger sizes.

When the IBM study's distribution is compared to the SFS 1.0 file distribution, I/O performed on 136 KB files only represents less than 4% of the files, confirming the concerns about using fixed length files. To address this problem SFS 2.0 creates a file set that is composed of files varying in size from 1 KB to 1024 KB, a range that reflects the experimental data. The read and write size distribution (Table 3) has not changed from SFS 1.0, as a result the large 1024 KB files are never accessed. Because these large files are

File size (max. bytes)	Number Files	Percent Files	Percent Space
0K	101264	1.69%	0.00%
0.5K	1102362	18.36	0.49
1K	746062	12.43	0.33
2K	1252118	20.86	1.11
4K	776916	12.94	1.18
8K	622675	10.37	1.71
16K	461333	7.68	2.43
32K	317392	5.29	3.25
64K	226765	3.78	4.62
128K	159661	2.66	6.57
256K	104109	1.73	8.19
512K	77396	1.29	12.87
1024K	26553	0.44	8.47
2048K	14396	0.24	9.05
4096K	8659	0.14	10.75
8192K	3410	0.06	8.46
16384K	1420	0.02	7.03
32768K	616	0.01	5.93
> 65536K	231	0.00	7.56

Table 4: IBM File Size Distribution.

such a low percentage of the total, no change in performance is expected.

While the distribution of file sizes is correct, the average file size is only 27 KB instead of the 37 KB average found in the IBM experiment. This was not noticed until after the release of the benchmark. A better distribution would be to replace the one percent of 1024 KB files with one percent 2048 KB files.

Size (Kbytes)	Percentage
1	33%
2	21
4	13
8	10
16	8
32	5
64	4
128	3
256	2
1024	1

Table 5: SFS 2.0 File Size Distribution.

File Selection

With all files in SFS 1.0 being fixed in size, the selection of a file for an I/O operation within the working set is done randomly with a uniform distribution. A serious problem resulting from this approach was the common occurrence of a small read request starting from the beginning or middle of a 136 KB file. Many servers and disk subsystems are tuned through read ahead algorithms for the common case of an application reading a file from start to end. The fixed size selection algorithm resulted in some servers demonstrating worse benchmark than real world performance.

The random selection of I/O files will not work with the new variable file size distribution, as some files may not be large enough to satisfy the request. Both a first fit and a best fit algorithm would allow all requests to be satisfied, but a best fit was chosen because it would also maximize the number of whole file reads and writes.

The variable file size distribution results in 67% of all files being less than the 8 KB transfer size. However, most modern NFS clients use an I/O subsystem that is integrated with the virtual memory subsystem, resulting in all I/O being rounded up to page size requests. To simulate this common style of client, SFS does not request the actual size of a small file, but instead issues a full 8 KB read. When a request for a read of less than 8 KB is made, any file less than 8 KB is chosen on a first fit basis. This greatly simplifies the selection algorithm while not materially changing the load on the server.

Append ratio

Research at the time of the original SFS 1.0 release indicated that the ratio of writes appending to files was as high as 90-95% in some environments [Hartman92]. A simple explanation of this is the fact

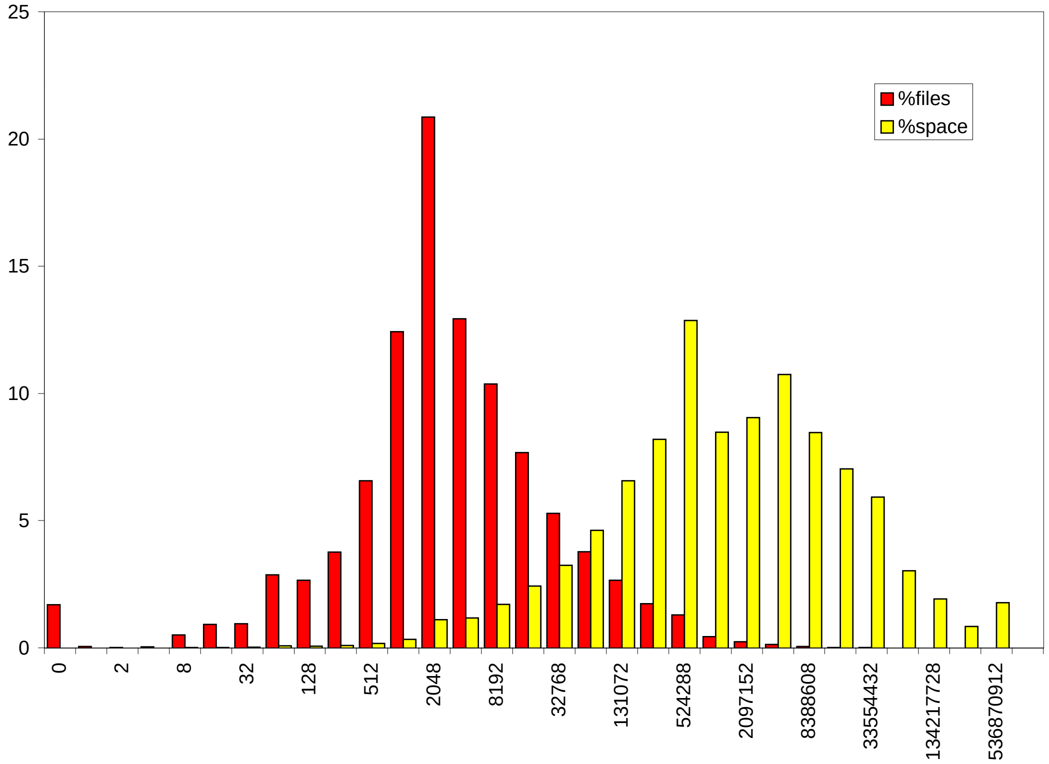


Figure 2: IBM distribution of files by number and space.

that most applications write a file from start to end. Unfortunately it was found that the benchmark became unstable due to uncontrolled growth of the file set when append ratios higher than 70% where chosen. SFS 2.0 maintains the same explicit append ratio but the actual append ratio is higher through the combination of variable sized files, a best fit selection algorithm, and most requests resulting in multiple sequential operations to the server. Although this better simulates whole file writes, the append ratio is probably still not high enough.

Multiple Directories

Each client in SFS 1.0 runs one or more load generating processes, each with its own directory of files to operate on. The directory is large and flat, containing all of the files and empty directories. While easy to implement in the benchmark, the main directory became unrealistically large under heavy loads causing an over-emphasis on the server's ability to perform directory operations. SFS 2.0 solves this problem by creating a new subdirectory for every 30 files created in the working set.

```
int clnt_poll(CLIENT *cl,
              uint32_t usecs);

bool_t clnt_getreply(CLIENT *cl,
   xdrproc_t xres, void *resp,
   int cnt, uint32_t *xids,
   uint32_t *xid,
   struct timeval *tv);
```

Table 6: New client interfaces.

The IBM study showed that the mean number of directory entries was 10 but the median was only 3, significantly less than the 30 entries created by SFS. To accurately implement this distribution would have been fairly complex. The performance impact of this compromise is considered to be low. The operations most affected by directory size are lookup, readdir, and readdirplus. Virtually all servers implement some form of directory name lookup cache which minimizes the effects of directory size on lookup operations. Conversely, using a small number of entries in a directory would help the performance of readdir operations but would not reflect some significant environments. By choosing a fixed size of 30 entries, lookup operations are not greatly effected by directory size and there are enough entries to adequately test readdir performance.

The names of the files created in the file set were changed to reflect the type of the file. This was done to assist in trouble shooting benchmark problems as well as to invalidate any potential server optimizations based on the SFS 1.0 names. The average length of a file name was set to 13 to reflect the results of the IBM study which showed that over 50% of the files had a name length between 8 and 12 characters. The IBM study also provided the average occurrence of a particular character within a file name. Although no attempt was made to set the selection of characters in a file name to match this distribution, the most common character, the period, was added to the names. The current simple fixed naming scheme is still vulnerable to servers optimizing caches for the benchmark based on the names of files.

TCP

The NFS protocol is designed to be transport independent, however until the early 1990 most implementations chose to only support the UDP transport. Concurrent with the release of V3 implementations, most vendors also introduced support of both V2 and V3 over the TCP transport and many made TCP the default. To provide customers with the ability to compare servers running in these environments, support for TCP was added to SFS 2.0.

Connection Management

There is no formal specification of how many TCP connections should be made between a client and a server. Some clients choose to have just one connection, others may have one per mount point, while some user level clients may have one per file. SFS is designed to present the server with the appearance of many clients but only using a small number of actual load generators, each generating a pseudo random set of operations on a set of files. This is very effective at hiding the number of real clients when using a protocol like UDP that has a single end point. However, with a connection oriented transport like TCP, the choice of how to model connections is quite important.

Although there are multiple choices on how a client should connect to a server, the majority of the clients have only one connection to each server. This is fortunate, because the aggregation of the operations of many client processes over multiple mount points can still be reasonably modeled using SFS' pseudo random operation selection. If one connection per mount point, or worse one per file, were common, it would have required SFS to introduce temporal effects into the operation selection, a formidable task.

To simulate multiple connections, the existing SFS 1.0 framework that supports multiple load generating processes on each load generator was used. Each load generating process creates one connection to the server. While the number of load generating processes used on tests of very large systems is likely to be considerably less than would occur in a real world environment, the effects were not considered to be significant in the results.

Implementing TCP

The addition of TCP to the SFS 2.0 code base proved to be one of the more difficult challenges. A primary feature of SFS is the ability to simulate client asynchronous read-aheads and write-behinds, otherwise known as BIOD emulation. Because of the request/response nature of an RPC protocol, a client waiting for the result of an I/O operation to return before issuing the next request, would suffer from poor sequential performance. To minimize this problem, most clients break large I/O requests into multiple concurrent requests and perform read-ahead or write-behind.

To minimize the effects of the client operating system and increase portability, SFS provides a full NFS and RPC user level client. Unfortunately, for portability reasons the client is single-threaded, making BIOD emulation a challenge. To further compound the problem, no freely available RPC library provided the capability to send RPCs asynchronously and gather replies. Ironically the existing libraries contain a minimal attempt at sending asynchronous RPCs, but fail to provide a reply gathering mechanism.

BIOD emulation was built using the freely available Sun RPCSRC 4.0. The standard *clnt_call* function was split into three pieces, the synchronous case and two new asynchronous functions *clnt_getreply* and *clnt_poll*. The standard *clnt_call* semantics were extended to place the RPC's transaction id in the result buffer if a timeout of zero is specified, allowing the caller to track multiple requests. The client can then call the new *clnt_poll* with the list of transaction ids to wait on a reply. When a reply is available *clnt_getreply* is called to process one request. The client provides as arguments an array of transaction ids being waited upon and if the *clnt_getreply* is successful, it returns the transaction id of the reply that was processed. The client can then repeat the process until all outstanding requests are processed.

Transport Specific Metrics

The choice of using TCP or UDP as the transport layer will have an impact on the performance of NFS. It is generally believed that UDP as a lighter weight protocol will offer better performance on a local network where packet loss is (nowadays) extremely rare, while TCP, with its reliable nature, will offer better performance on a wide area network, where packet loss is not uncommon.

The interesting question was whether to create an additional transport specific metric for both V2 and V3 or not. Because there is a measurable difference between transports, some considered it unfair to compare a TCP result against a UDP result. However, the workload presented to the server is independent of the transport. If the performance effects caused by choice of transport layer are compared to the performance effects caused by choice of network media, many similarities exist. An FDDI ring will have lower latencies than a 10BaseT Ethernet and a network adapter that does on board protocol processing will have higher throughput than one without. In these cases no distinction in the metric is made and therefore the choice of transport is also not reflected in the metric. The transport is clearly presented in the full results report allowing an informed comparison to be made.

Overall Response Time Metric

The official abbreviated form of an SFS 1.0 results report is the throughput in operations per second at a peak response time in milliseconds. While the primary throughput metric is easily understood and comparable between systems, the response time is poorly understood and is occasionally misused in the marketing or interpretation of results. The response time reported is simply the response time at the maximum throughput with the only condition that it can not exceed 50 milliseconds. To compare the peak response time of two results only shows how the system degrades under maximum load and is no reflection on how the system response during typical loads experienced by most customers.

As the capacity of most NFS servers has grown dramatically, the aggregate throughput of the server has become a secondary consideration for many environments. The responsiveness of the server becomes more important as it contributes to the throughput of sequential operations as well as the subjective "feel" experienced by end-users sitting at NFS clients.

In comparing two SFS results, the shape of the curve is an indication of how the server responds over the entire load range. A curve that is flat and close to the x-axis indicates a server that is consistently fast over a wide range of loads. The challenge was to create a meaningful single metric that captured the quality of the graph. The area under the curve divided by the peak throughput was chosen as the new SFS 2.0 metric called the Overall Response Time, calculated by using the Riemann sum:

$$O.R.T. = \frac{\sum_{0}^{n}\left[\frac{R_i + R_{i-1}}{2} * (T_i - T_{i-1})\right]}{T_n}$$

Run Rules

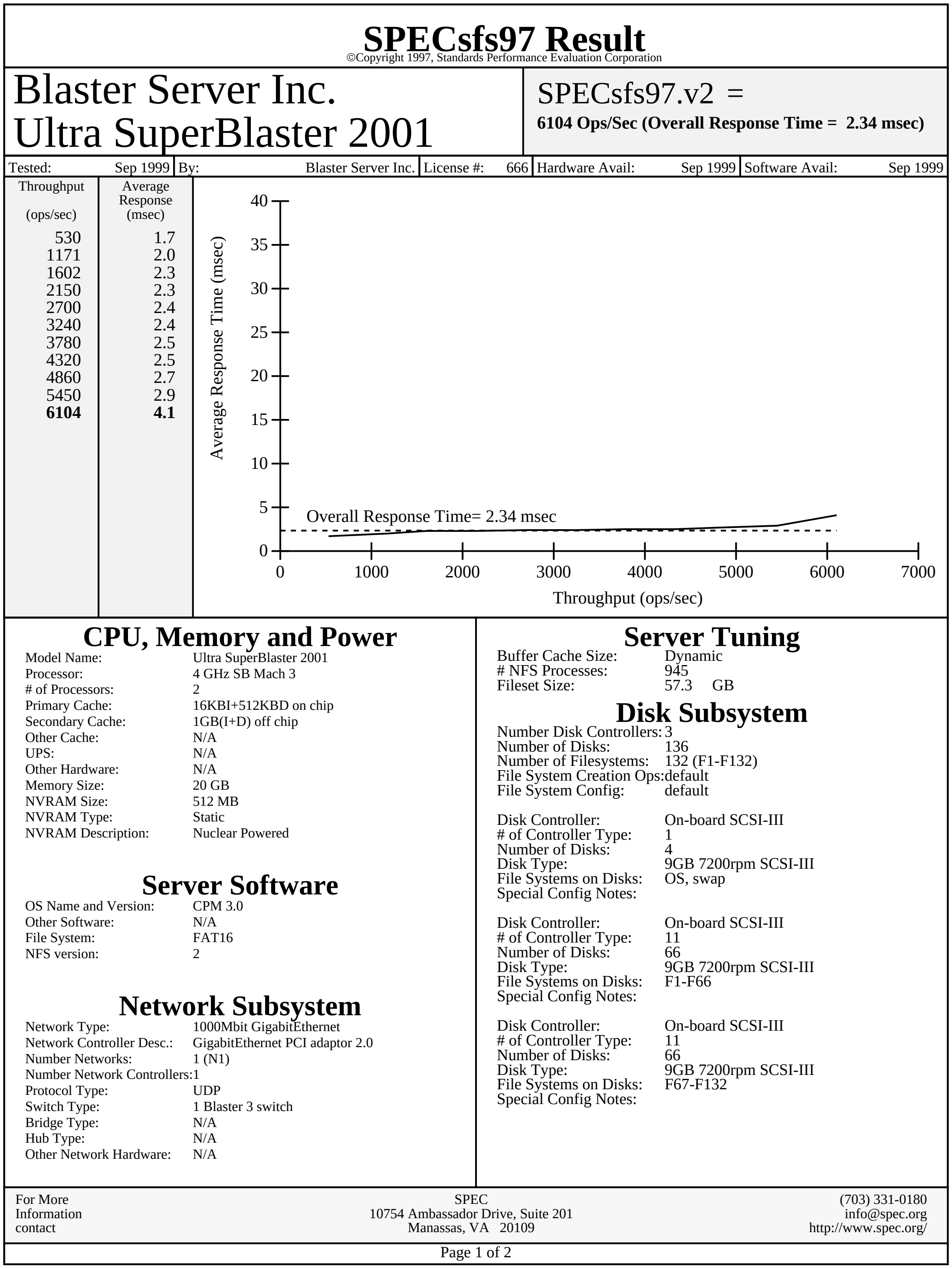
A critical component of any SPEC benchmark are the run and disclosure rules which allow customers to make fair comparisons between competing vendors. To the credit of the original SFS 1.0 benchmark, only a few significant changes were required for SFS 2.0.

Response Time

The throughput of a system is the primary metric for comparison between two competing vendors, but at some point additional throughput is overwhelmed by the degradation in response time. The maximum reportable response time was set in SFS 1.0 to the some what arbitrary value of 50 milliseconds. The choice was based primarily on the human sensitivity of response time. Following the decrease in both network and disk latency, the maximum reportable response time was decreased to 40 milliseconds in SFS 2.0.

Warmup and Runtime

SFS load generation is split into two phases, warmup and runtime. The warmup phase is required to allow the benchmark to make initial requests and adjust the inter-request sleep interval to achieve the requested load level. As vendors gained experience with SFS 1.0, it was observed that in some configurations the benchmark would not converge on a steady request rate within the one minute warmup time resulting in an undesirable variance in reported results. For SFS 2.0, the warmup time was increased to five minutes to ensure stability of the request rate. Additionally it was observed that during the last five minutes of the ten minute runtime, the request rate was extremely stable and the results were unaffected by decreasing the runtime to five minutes. The increase in the warmup time may cause large servers to better cache the working set, it is hoped that the decrease in



SPECsfs97 Result

©Copyright 1997, Standards Performance Evaluation Corporation

Blaster Server Inc.
Ultra SuperBlaster 2001

SPECsfs97.v2 =
6104 Ops/Sec (Overall Response Time = 2.34 msec)

Tested:	Sep 1999	By:	Blaster Server Inc.	License #:	666	Hardware Avail:	Sep 1999	Software Avail:	Sep 1999

Throughput (ops/sec)	Average Response (msec)
530	1.7
1171	2.0
1602	2.3
2150	2.3
2700	2.4
3240	2.4
3780	2.5
4320	2.5
4860	2.7
5450	2.9
6104	**4.1**

CPU, Memory and Power

Model Name:	Ultra SuperBlaster 2001
Processor:	4 GHz SB Mach 3
# of Processors:	2
Primary Cache:	16KBI+512KBD on chip
Secondary Cache:	1GB(I+D) off chip
Other Cache:	N/A
UPS:	N/A
Other Hardware:	N/A
Memory Size:	20 GB
NVRAM Size:	512 MB
NVRAM Type:	Static
NVRAM Description:	Nuclear Powered

Server Software

OS Name and Version:	CPM 3.0
Other Software:	N/A
File System:	FAT16
NFS version:	2

Network Subsystem

Network Type:	1000Mbit GigabitEthernet
Network Controller Desc.:	GigabitEthernet PCI adaptor 2.0
Number Networks:	1 (N1)
Number Network Controllers:	1
Protocol Type:	UDP
Switch Type:	1 Blaster 3 switch
Bridge Type:	N/A
Hub Type:	N/A
Other Network Hardware:	N/A

Server Tuning

Buffer Cache Size:	Dynamic
# NFS Processes:	945
Fileset Size:	57.3 GB

Disk Subsystem

Number Disk Controllers:	3
Number of Disks:	136
Number of Filesystems:	132 (F1-F132)
File System Creation Ops:	default
File System Config:	default

Disk Controller:	On-board SCSI-III
# of Controller Type:	1
Number of Disks:	4
Disk Type:	9GB 7200rpm SCSI-III
File Systems on Disks:	OS, swap
Special Config Notes:	

Disk Controller:	On-board SCSI-III
# of Controller Type:	11
Number of Disks:	66
Disk Type:	9GB 7200rpm SCSI-III
File Systems on Disks:	F1-F66
Special Config Notes:	

Disk Controller:	On-board SCSI-III
# of Controller Type:	11
Number of Disks:	66
Disk Type:	9GB 7200rpm SCSI-III
File Systems on Disks:	F67-F132
Special Config Notes:	

For More
Information
contact

SPEC
10754 Ambassador Drive, Suite 201
Manassas, VA 20109

(703) 331-0180
info@spec.org
http://www.spec.org/

Page 1 of 2

Figure 4: SFS 2.0 Reporting Page

the runtime will lessen any opportunity to exploit the cache.

Clients per Network

The SFS 1.0 run rules require that there be at least two clients per network used in the test configuration. The initial intent was to ensure that clients would realistically experience collisions on ethernet networks. A few factors have emerged to make this requirement unnecessary. The wide spread shift from fat yellow ethernet cable and thin-net to switching twisted pair intelligent hubs have greatly reduced the chances that a client on a heavily loaded ethernet will experience a significant number of collisions. Even on a network with collisions, the effects of the physical layer arbitration does not change the maximum capacity of a server. When a client backs off from a collision, it increases the response time for a given operation but does not decrease the capacity of the server, as the back off is handled in hardware without intervention of the main processor. The collision does decrease the available network bandwidth, but with the addition of another network interface the same server can handle the same capacity whether the network is switched or not. With the goal of SFS to measure server capacity, the mandating of network topology has proven to be unnecessary.

Future Work

SFS 2.0 was able to solve a number of the flaws in the original SFS 1.0 benchmark and update the workload to more closely reflect real world usage. However there are still a number of significant areas for future work. Most important is another study of V3 servers, similar to what was performed for V2 servers, to validate the operations mix. In the time since the release of the benchmark, it appears that the mix of directory reads may need to be adjusted. The algorithms used by clients to determine when to issue a light weight readdir request, versus a readdirplus request, is still evolving and moving towards minimizing the number of readdirplus operations.

The SFS framework is designed to provide the correct operations mix and I/O sizes based on studies that represent the cumulative history of the server. The random selection does not take into account the temporal nature of file access. The next file accessed is more likely to be the same file, or a file in the same directory, than a file randomly selected from the working set. Two possible solutions would be to study the sequences of requests made to servers to determine any significant patterns, or to switch from a random selection with a Poisson distribution to a Markoff distribution.

SFS began as a benchmark that used the client system calls to generate network traffic. It has become client independent, generating network traffic directly. The reduction in client sensitivity allows for very fair comparisons of two servers using the same protocol. But customers are asking for comparisons of the same server using different protocols. To accomplish this a protocol independent benchmark that either calls the client system calls directly or drives real or pseudo applications. The challenge of such a benchmark is to determine what a representative workload is and to manage client side effects so the server is being fairly tested. The Andrew benchmark was an early attempt at a protocol independent benchmark.

Acknowledgments

The author would like to acknowledge the members of the SFS steering committee who spent many hours collecting data, analyzing it, testing the code, and most importantly, arguing about it. In particular, Kamesh Gargeya, Sherry Hoffman, Joan Lawler, Tony Lukes, Brian Pawlowski, Judy Piantedosi, Spencer Shepler, Pete Smoot, and Andy Watson.

This paper is dedicated to the memory of Terry Flynn who provided most of the statistical analysis which is the basis of SFS 2.0.

Author Information

David Robinson is a Senior Staff Engineer in the Solaris Networking Technology Group at Sun Microsystems, Inc. He has been the primary author of the SFS 2.0 benchmark suite, and founder of the SPEC SFS steering committee. He has been active in the NFS development community for since 1986. Prior to Sun, he worked at the Jet Propulsion Laboratory where he developed one of the first NFS servers for VMS to allow workstations to access large satellite images. Reach him via U.S. Mail at Sun Microsystems, Inc., MS MPK17-201, 901 San Antonio Rd, Palo Alto, CA 94303. Reach him electronically at <robinson@eng.sun.com>.

References

[Baker91] Baker, Mary G., "Measurement of a Distributed File System," *Proceedings of the 13th Symposium on Operating System Principles,* pp. 198-212, October 1991.

[Callaghan95] Callaghan, Brent, et al., "RFC1813, NFS Version 3 Protocol Specification," June 1995.

[Hartman92] Hartman, John, "File Append vs. Overwrite in a Sprite Cluster," Sprite Project, University of California at Berkeley, Presentation to the LADDIS Group, Jan 21, 1992.

[Irlam94] Irlam, Gordan, http://www.base.com/gordoni/ufs93.html, *An informal study of Unix file sizes from data gather via Usenet*, Sept, 1994.

[Keith90] Keith, Bruce, "Perspectives on NFS File Server Performance Characterization," *Proceedings of the Summer 1990 USENIX Conference*, pp. 267-277, June, 1990.

[Pawlowski94] "NFS Version 3 Design and Implementation," *Proceedings of the Summer 1994 USENIX Conference*, June, 1994.

[Sandberg85] Sandberg, Russell, et al., "Design and Implementation of the Sun Network File System," *Proceedings of the Summer 1985 USENIX Conference*, pp. 119-130, June, 1985.

[SPEC] Standard Performance Evaluation Corporation (SPEC) web site, http://www.spec.org/ .

[Sun89] Sun Microsystems, "RFC1094, NFS," March, 1989.

[Watson92] Watson, Andy, et al., "LADDIS: Multi-Vendor and Vendor-Neutral SPEC NFS Benchmark," *Proceedings of the LISA VI Conference*, pp. 17-32, October, 1992.

[Whittle93] Whittle, Mark, et al., "LADDIS: The Next Generation In NFS File Server Benchmarking," *Proceedings of the 1993 USENIX Conference, 1993*.