LADDIS: The Next Generation In NFS File Server Benchmarking

Mark Wittle
Data General Corporation

Bruce E. Keith
Digital Equipment Corporation

April 1993

Abstract

The ability to compare the performance of various NFS[1] file server configurations from several vendors is critically important to a computing facility when selecting an NFS file server. To date, nhfsstone[2] has been a popular means of characterizing NFS file server performance. However, several deficiencies have been found in nhfsstone. The LADDIS NFS file server benchmark has been developed to resolve nhfsstone's shortcomings and provide new functionality. The Standard Performance Evaluation Corporation (SPEC[3]) released the System File Server (SFS) Release 1.0 benchmark suite, which contains 097.LADDIS, as an industry-standard file server benchmark in April 1993. This paper describes the major technical issues involved in developing the benchmark and the rationale used to establish default 097.LADDIS workload parameter values. Where appropriate, areas for further research are identified and encouraged.

Contents

Figures

1 LADDIS Overview

LADDIS is a synthetic benchmark used to measure the NFS [Sandberg85] request response time and throughput capacity of an NFS file server. The benchmark produces response time versus throughput measurements for various NFS load levels. LADDIS is executed concurrently on one or more NFS client systems, located on one or more network segments connected to the NFS file server being measured. The NFS client systems, called LADDIS load generators, send a controlled stream of NFS requests to the server, according to a specific operation mix and file access distribution, allowing precise measurement of the server's response time for each request. Other than basic server configuration requirements, LADDIS is unconcerned with how the file server provides NFS service—the benchmark measures the server as if it were a black box.

This paper describes the major technical issues involved in developing the benchmark and the rationale used to establish the default LADDIS workload parameter values, reflected in the 097.LADDIS benchmark released in the SPEC SFS Release 1.0 benchmark suite [SPEC93].

1.1 Nhfsstone Deficiencies

To date, nhfsstone has been a popular means of characterizing NFS file server performance [Legato89; Shein89]. Nhfsstone synthetically duplicates average NFS file server resource utilization levels attained with a real user-level workload running on a collection of NFS clients [Keith90]. Despite its popularity, nhfsstone has several shortcomings.

Nhfsstone is sensitive to differences in NFS client kernel implementations across vendors' NFS clients. For a given logical file operation, nhfsstone can produce a different NFS protocol request sequence depending on the vendor's NFS client kernel and hardware configuration. As the NFS request load level produced by nhfsstone is increased, the impact of the client kernel implementation is amplified. This results in considerable inconsistency in the mix of operations generated at high NFS request load levels.

Nhfsstone attempts to overcome these difficulties through specific algorithms designed to by-pass NFS client kernel performance factors such as file name, attribute, and data caching. These work-arounds are operating-system specific and are not equally effective across different vendor platforms. The resulting variations can skew the performance results obtained with nhfsstone.

The use of operating-system specific algorithms and the need to access kernel data structures to monitor the delivery rate of requests to the NFS file server make nhfsstone difficult to port to a broad set of NFS client platforms.

Nhfsstone's load-generating capacity is confined to a single NFS client system, which eliminates the performance impact of all network contention effects observed when servicing multiple NFS clients [Stern92]. Nhfsstone is limited in its ability to characterize NFS file servers that support multiple networks. An adequate solution to coordinating multiple nhfsstone instantiations requires the creation of additional control and results-consolidation software. The development of such control and results-consolidation software introduces second-order issues, primarily file set size control when using multiple clients to generate an aggregate load.

Finally, nhfsstone lacks a standardized approach to running the benchmark and reporting performance results that ensures fair comparisons of NFS file server performance across a range of configurations and vendor platforms.

1.2 LADDIS Improvements

To resolve nhfsstone's shortcomings, the LADDIS NFS file server benchmark was developed by a small group of engineers from NFS file server vendors: Legato Systems, Auspex Systems, Data General, Digital Equipment, Interphase, and Sun Microsystems. The benchmark was adopted and further refined by the Standard Performance Evaluation Corporation (SPEC) as an industry-standard NFS file server benchmark. SPEC released its System File Server benchmark suite, SFS Release 1.0, which comprises the LADDIS NFS file server benchmark, designated 097.LADDIS, in April 1993. All specific algorithms and parameter settings described herein apply to the 097.LADDIS benchmark.

The LADDIS NFS file server benchmark overcomes nhfsstone's shortcomings:

- NFS client kernel sensitivities were reduced significantly by implementing the NFS protocol within the LADDIS benchmark.
- NFS load generation algorithms were improved to produce a highly accurate mix of NFS operations.
- Control and result-consolidation functionality now supports coordinated, simultaneous use of multiple NFS client systems on multiple networks to generate an aggregate NFS load on the server under test.
- SPEC's SFS Release 1.0 Run and Report Rules for LADDIS provide a consistent and platform-independent method for running the benchmark and reporting NFS file server performance results, thereby eliminating confusion in interpreting NFS performance results within the NFS community.
- LADDIS has been ported to a variety of BSD-, SVR3-, SVR4-, and OSF/1[4]-based systems as part of SPEC's benchmark adoption process.

LADDIS also provides new capabilities. The NFS server file set targeted by LADDIS scales with requested NFS load level. This scaling allows LADDIS to mimic real NFS server environments where more server files are accessed as NFS load increases due to additional users or more NFS-intensive applications.

LADDIS provides new parameters, which further refine nhfsstone's NFS workload abstraction of an NFS operation mix and an NFS operation request rate. NFS request packet sizes, data stream I/O lengths, file working set size, and file update patterns are under parameterized control. Users and developers can tailor the values of these parameters enabling LADDIS to produce a synthetic workload that emulates the load on NFS file servers in their own specific computing environment.

1.3 Dual-Purpose Tool

LADDIS is both an NFS server benchmark and an NFS performance characterization tool. As an industry-standard benchmark, SPEC SFS Release 1.0 Run and Report rules ensure that NFS server performance is measured consistently throughout the NFS community using standard, required workload abstractions that establish specific values for LADDIS's numerous workload parameters. Furthermore, SPEC SFS Release 1.0 Run and Report Rules ensure that NFS server performance results are reported in a thorough, consistent, results format, which specifies all of the hardware and software configuration parameters used to produce the results.

As a performance characterization tool, the breadth of LADDIS's workload parameters coupled with its multiple NFS client and network load generation capability allow NFS servers to be stressed under a variety of workloads. These capabilities facilitate the investigation of server behavior and performance in a number of computing environments.

The default LADDIS workload is based on an intensive software development environment. The parameters have been established based on the findings of several independent NFS file server studies. Future work is encouraged to develop additional LADDIS work load parameter sets that will allow LADDIS to generate synthetic workloads mimicking computing environments such as CAD, imaging, animation, and commercial NFS applications.

1.4 How LADDIS Measures NFS Server Performance

LADDIS measures the response time of the NFS server at the load generator using client operating system clock routines. Nearly all configuration and runtime parameters associated with the clients and networks are controlled explicitly by the benchmark. The goal is to isolate and measure the NFS server's response time by eliminating, minimizing, or controlling outside factors, including the client platform, whenever possible. With few exceptions, LADDIS is unconcerned with the file server's configuration parameters except to require that all relevant hardware and software details are reported with the performance results.

LADDIS implements the NFS protocol within the benchmark, rather than using the client kernel's NFS implementation. The benchmark formulates Open Network Computing (ONC[1]) remote procedure call (RPC) packets directly, and measures server response time on entry and exit from the user-space RPC library calls used to send and receive network packets. This approach removes the client kernel's NFS attribute and data caching implementation from the benchmark's code path, yet maintains the RPC library as a standard base of software portability.

Some client operating system overhead is inherent in the process of generating NFS load and measuring response times. However, implementing the NFS protocol within the benchmark reduces operating system overhead, which has a positive impact on the accuracy of the benchmark. First, LADDIS gains precise control over the NFS request stream actually sent to the server. Second, much of the client kernel code path execution is removed from server response time measurements.

1.5 What LADDIS Does Not Measure

LADDIS is not an NFS client or NFS client-server benchmark. LADDIS uses NFS client systems to generate carefully controlled NFS load. Unlike normal client-side implementations of NFS, the NFS protocol is implemented in user space without attribute or data caching. The benchmark is meant to generate NFS load as it is seen by the NFS file server, not as it is indirectly generated by NFS client applications. This design choice makes LADDIS an inappropriate indicator of NFS client performance.

LADDIS is not an I/O bandwidth benchmark. The NFS protocol supports 17 different RPC request types. NFS READ and WRITE requests make up only about a third of the requests performed when using the benchmark's default workload parameters. Although LADDIS can be configured to perform a mix of operations dominated by READ and WRITE requests, other benchmarks tailored for sequential I/O bandwidth measurement are better suited to that task.

LADDIS does not model any specific user application. Executed with the default parameter settings, LADDIS generates a workload that represents an intensive software development environment, not any particular application or class of applications. LADDIS is also a powerful performance characterization tool that provides a wide range of workload parameters that can be adjusted to represent a specific environment. However, LADDIS is not a workload profiling tool. If users desire a non-standard workload, they must determine an appropriate LADDIS parameter set (workload abstraction) to represent their specific environment.

2 LADDIS Architecture

LADDIS benchmark components consist of a set of LADDIS load generators that create the NFS request load, a central LADDIS manager which coordinates execution of the benchmark, and the NFS file server under test. The LADDIS components execute on systems configured as NFS clients of the file server. Figure 2-1 shows an example benchmark testbed configuration with six LADDIS load generators distributed over two LAN segments, accessing an NFS file server. The LADDIS manager may be executed on a separate system, or on one of the LADDIS load generators. Actual LADDIS measurements can be made using a wide range of load generator and network configurations.

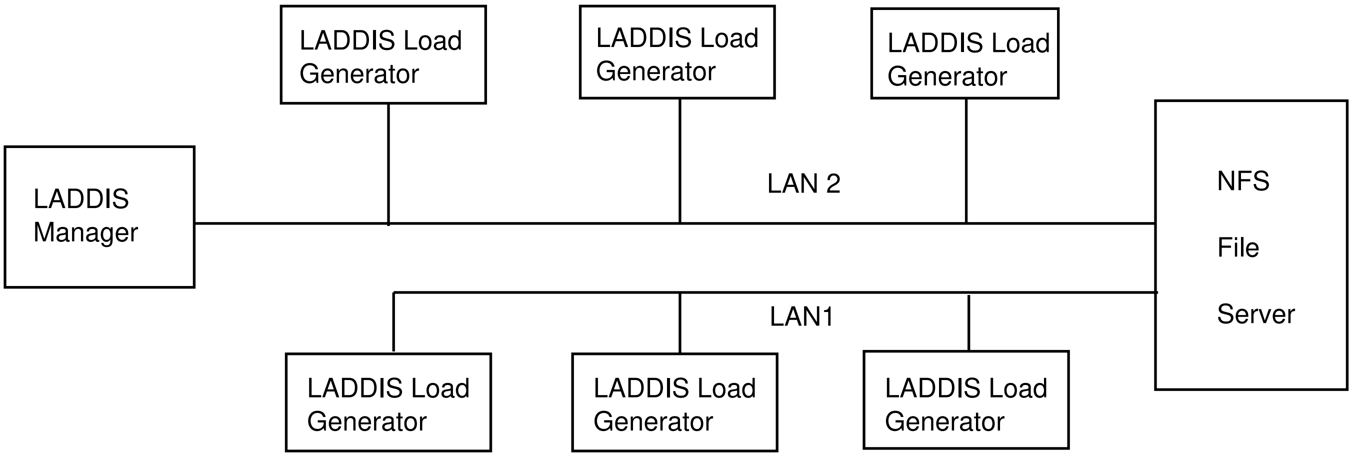

Figure 2-1 LADDIS Testbed Scenario

2.1 LADDIS Components

LADDIS executes as a collection of distributed, cooperating processes. A master shell script (laddis_mgr) reads the benchmark input parameter file (laddis_rc), and then spawns a remote shell script (laddis_mcr) on each of the load generators participating in the benchmark run. These shell scripts handle benchmark output, create log and error files, and spawn the various distributed processes involved in executing the benchmark.

The central LADDIS manager process (laddis_prime) controls the various phases of the benchmark and consolidates performance results from each LADDIS load generator. LADDIS synchronization daemons (laddis_syncd) execute on each load generator system and use an RPC-based message protocol to synchronize load generators while the benchmark is running.

During the load generation phase of the benchmark, multiple load-generating processes (LADDIS) executing on each load generator send NFS requests to the file server. A parent communicates with the synchronization daemon and central LADDIS manager to synchronize remote execution and report results. Figure 2-2 shows the communication paths of the LADDIS processes that execute on each load generating system.

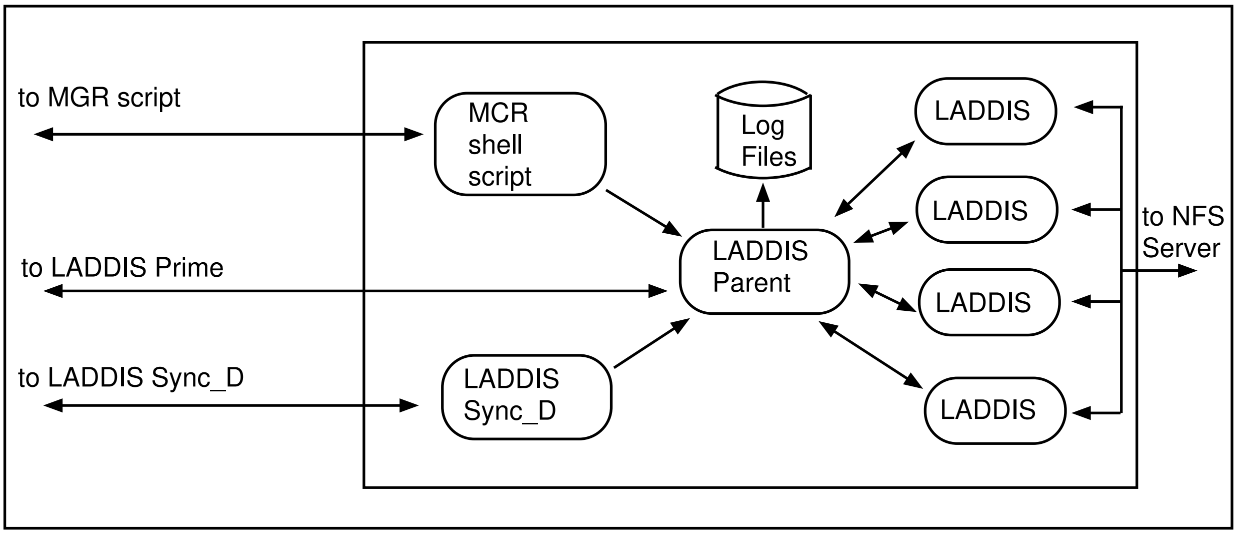

Figure 2-2 LADDIS Load Generator Processes

2.2 User Interface

LADDIS includes an ASCII-based, menu-driven, user interface for managing benchmark configuration files, executing test runs, and viewing results files. The interface is similar to other SPEC benchmark suites [Dronamraju93].

2.3 Protocol Requirements

LADDIS implements the NFS protocol within the benchmark. This design enables the benchmark to maintain an accurate count of NFS requests sent to the file server. Correctness of the server's NFS implementation is verified during each benchmark run by a validation phase before beginning performance measurements.

LADDIS uses standard ONC/RPC library interfaces, which provide the base of portability for the benchmark. The RPC library isolates LADDIS from the network transport layer implementation. To make use of asynchronous RPC calls, some standard RPC library functions have been re-implemented within LADDIS.

LADDIS uses standard TCP/IP transports for internal communications between distributed LADDIS processes and UDP/IP for sending requests to the NFS file server. SPEC SFS Release 1.0 Run and Report Rules require the use of UDP checksums on all network packets generated and received during benchmark execution.

3 Testbed Parameters

A LADDIS test run consists of a series of benchmark executions, each at increasing NFS load levels, generated by a set of LADDIS load generators and targeted at the NFS file server being tested. Most benchmark configuration parameters have a broad range of valid settings, but once chosen, they must remain constant across the LADDIS test run. The configuration parameters that are fixed across a test run are:

- all aspects of the server software and hardware configuration
- the number and type of LADDIS load generators used to create NFS load
- the number and type of network segments connecting the load generators to the server
- the arrangement of the load generators on the network segments
- the number of LADDIS load-generating processes executing on each load generator
- all LADDIS workload parameters except the requested load level

3.1 Server Configuration

SPEC SFS Release 1.0 Run and Report Rules specify service characteristics that the NFS server is required to support (e.g., correctness of operations and reliability of data). NFS Version 2 protocol conformance is verified during the test run, just before beginning the series of performance measurements.

During test execution, LADDIS is not concerned with the details of the server configuration. However, once chosen, the server configuration must remain constant across the entire LADDIS test run. All relevant server hardware and software configuration information is reported with the performance results to ensure reproducibility.

3.2 Client Configuration

LADDIS can be run using one or more NFS load generator systems. The load generators need not be equipped identically, be the same model, or even be from the same vendor. The only requirement is to have enough load generators to saturate the file server's ability to service NFS requests. SPEC SFS Release 1.0 Run and Report Rules require that at least two NFS load-generating systems and at least eight load-generating processes be configured on each network segment. However, throughput saturation of an NFS file server may require a greater number of load generators, each producing many NFS requests concurrently.

A primary LADDIS design goal was to reduce client sensitivity in NFS performance measurement, and LADDIS has achieved that goal to a significant degree. Typically, each load generator system executes multiple LADDIS load-generating processes (the LADDIS parent and synchronization daemon present minimal overhead during the performance measurement phase of the benchmark). Because these processes share the client system's CPU, memory, network device, and kernel software resources, the timesharing efficiency of these system elements can impact LADDIS response time measurements. Reducing the number of load-generating processes can improve NFS response time. However, reducing the number of load-generating processes may also reduce the client system's ability to generate load. If so, then additional load generator systems will be required to saturate the file server.

In addition to the cost of timesharing between processes, the load-generating platform introduces some overhead into the measurement of server response time. The CPU speed and efficiency of the network controller affect the measurement. In addition, the efficiency of the platform's compiler and RPC library code may influence response time results. LADDIS load generation algorithms have been designed to reduce client platform sensitivity

The sensitivity of LADDIS to client-based factors could be eliminated entirely by requiring all LADDIS performance measurements to be made using the same type of client system. This approach was rejected since few NFS environments are equipped with identical client configurations. Rather, SPEC advises that fast, efficient load generators be used when reporting LADDIS results. SPEC experience indicates that by employing powerful NFS client platforms as load generators, the effects of client sensitivity on results are minimized.

Before starting a LADDIS test run, the load-generating capacity of each client system type should be determined [Watson92]. This can be done by running LADDIS between a single load generator system and the NFS server in isolation at increasing load levels until a maximum throughput is reached. The number of load-generating processes can be adjusted, and response time and throughput results can be compared. The default value of four processes is a good starting point. This procedure should yield an upper bound on the load generator's ability to produce NFS load (relative to the targeted server) and determine the appropriate number of load-generating processes to run on the load-generating platform. Load generators should be added to the testbed configuration until the NFS server's saturation point is determined; this will ensure that LADDIS is measuring server saturation, rather than the NFS request limit of the load generators.

3.3 Network Configuration

LADDIS can be run using one or more network segments. The networks do not need to be the same type. The goal is to provide enough network bandwidth to allow the NFS load generators to saturate the server's ability to service NFS requests. However, depending on the NFS file server, simply adding additional load generators to a single network may result in measuring the network bandwidth rather than the NFS file server's throughput capacity.

An NFS file server may require more load than a single network can supply, regardless of the number of LADDIS load generators connected to it. Two questions must be answered: how many load generators are required to saturate a network segment, and how many network segments are required to saturate the NFS file server. Load generators can be added to a network segment until either the server or the network segment is saturated. Determining which is the bottleneck may require independent measurements of the network and server. For instance, a network analyzer can determine the network utilization level, and system utilities on the server, such as sar(1), or iostat(1), can measure the critical components of server performance. Experience within SPEC indicates that a single Ethernet[5] is capable of supporting up to approximately 300 SPECnfs_A93[3] operations/second.

Adding LADDIS load generators to a network segment may increase contention for network resources. Additional load generators may, in turn, introduce additional network collisions and retransmissions, which can affect the response time measured by LADDIS. Some network contention is normal. SPEC SFS Release 1.0 Run and Report Rules require that at least two load generators be configured on each network segment. Reducing the number of load generators will reduce contention and may improve NFS response time. However, reducing the number of load generators might not make efficient use of the network segment and may introduce the need for additional network segments to saturate the server. In turn, additional network interfaces will be required on the server, and these may have associated performance costs on the server. The common practice is to configure enough load generators to saturate each network segment and no more.

3.4 Load Level

Having determined the appropriate setting for the benchmark testbed parameters, a series of LADDIS executions is run at increasing NFS request load levels. The LADDIS manager process distributes the load equally among the load-generating systems, which, in turn, distributes the load equally among that system's load-generating processes. Subsequent executions of the benchmark increase the load level. Eventually, the NFS file server becomes completely saturated with requests, and the measured load level no longer increases with additional requested load. Requesting additional load beyond saturation may produce a rapid increase in server response time, or may result in reduced NFS throughput on the server.

Each execution of LADDIS produces an average response time versus load level pair, and these are combined to form a LADDIS performance graph. The graph plots average server response time (on the y-axis) versus NFS request throughput (on the x-axis). The resulting response time/throughput curve has a number of interesting features.

Figure 3-1 shows an example LADDIS performance curve. The general shape of the curve is flat or slowly rising followed by a steeper rise at higher load levels. The 50-millisecond response time level serves as an arbitrary reference for maximum average RPC response time. This level is sufficient for most NFS environments. The SPEC single figure-of-merit performance value is derived from server throughput at 50 milliseconds average response time.

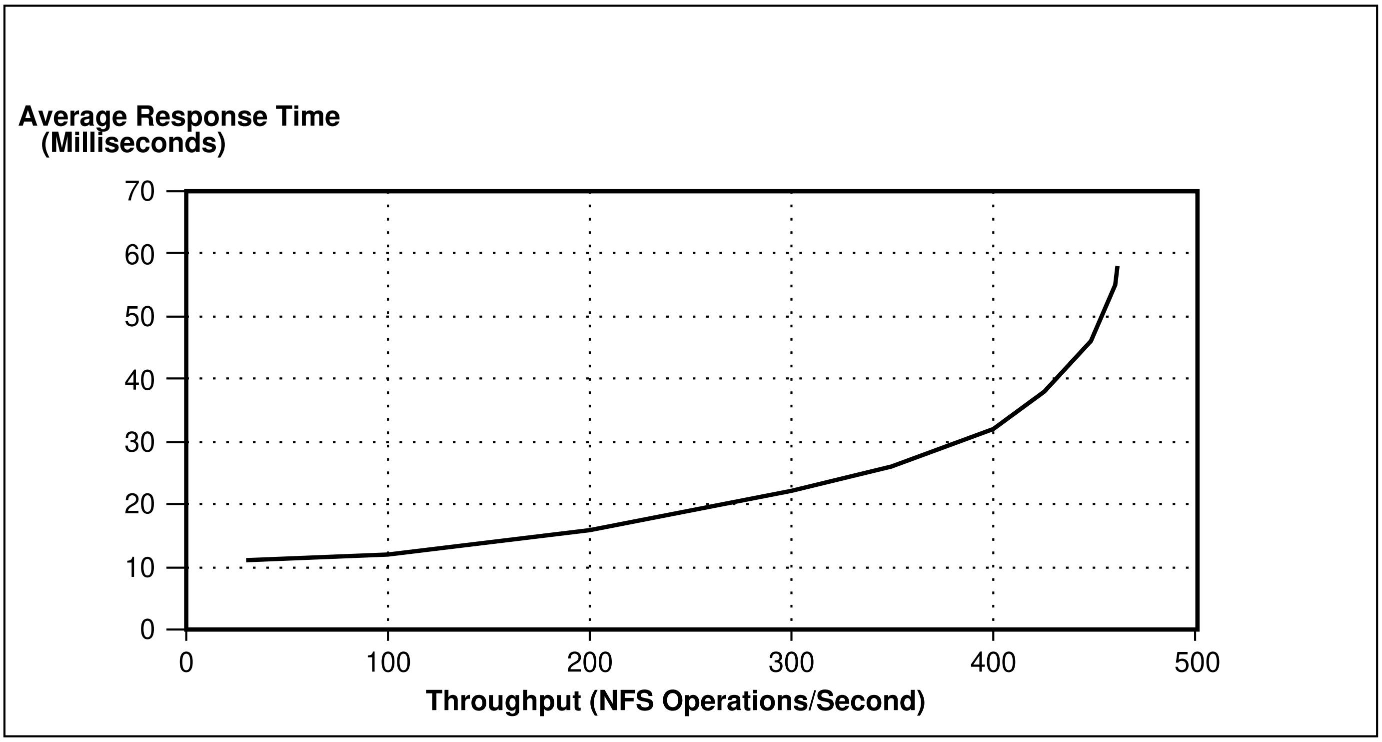

Figure 3-1 LADDIS Performance Graph

4 Workload Parameters

The goal of the LADDIS workload is to produce a synthetic approximation of an intensive software development environment. A number of workload studies have been applied, however no single study had sufficient breadth to establish all LADDIS parameter values simultaneously. Thus, a number of compromises, rules of thumb, and heuristics have been applied.

4.1 Operation Mix

LADDIS supports all operation types defined by the NFS Version 2 protocol. The default LADDIS operation mix is the same as the operation mix used in the nhfsstone benchmark. The mix is based on unpublished NFS client workload studies performed at Sun Microsystems during 1987 [Lyon92]. Standard LADDIS parameters specify an operation mix consisting of about half file name and attribute operations (LOOKUP 34% and GETATTR 13%), roughly one third I/O operations (READ 22% and WRITE 15%), with the remaining one-sixth spread among six other operations. Figure 4-1 presents the default operation mix percentages.

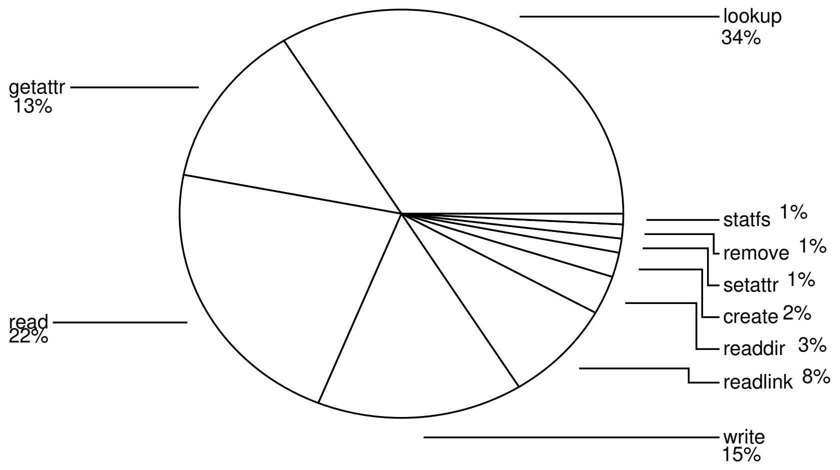

Figure 4-1 NFS Operation Mix

4.2 File Set

The set of test files accessed by LADDIS operations consists of a number of regular data files that scales with the requested load level, and a small fixed set of directories and symbolic links used for directory and symbolic link operations.

The test files (and the data they contain) are partitioned equally among all of the LADDIS load-generating processes; each process populates and accesses its own subdirectory of files on the NFS file server. There are no shared files, and hence, no file access contention between load-generating processes. For most applications, lack of contention is a reasonable assumption and has little bearing on overall NFS file server performance. For applications that do access shared data files and directories, the major performance impact is on client-based cache consistency policies which cause additional requests to be sent to the server. Because LADDIS is modeled on the operation mix actually delivered to the server, client cache hit rates do not affect the performance model.

Within each LADDIS process' private subdirectory, the test data files exist in a single, flat directory structure. One large directory per load-generating process is unrealistic for most NFS file server environments where files are generally distributed throughout hierarchical directory trees. Normally, creating a large number of files in a single shared directory can cause a greater number of directory LOOKUP requests to be performed. Because LADDIS does not share its private subdirectory, there is no impact on the mix of operations. However, NFS server algorithms may be affected by the number of entries in each directory. Distribution of the test file population into subdirectory hierarchies would be more realistic. The effect of this change should be studied and possibly incorporated into a future version of LADDIS.

All LADDIS test file names are formed from a common initial sequence of nine characters followed by a three digit unique number, e.g. "dir_entry001." LADDIS limits file name length to provide support for UNIX[6] systems that limit file names to 14 characters. The length limit and the similarity between file names may not be realistic for most NFS environments and could affect the NFS server's file name storage algorithms. The LADDIS file name space could be improved by introducing some randomization into the file names.

4.2.1 File Set and Working Set

There are two fundamental properties of the LADDIS file set. First, a large, static file set is created. Second, a smaller working set of files is chosen from the larger file set. All NFS requests generated by LADDIS are targeted at the working set. The initial size of the file set and the growth and access characteristics of the working set are critical components of the LADDIS workload model.

LADDIS creates the file set during the benchmark initialization phase. The amount of data in the file set is scaled to the target load level specified for the benchmark run at the rate of 5 megabytes (MB) of data for each NFS operation/second (op/sec) of load. This scaling ensures that the NFS file server storage capacity scales with the NFS throughput offered by the server. The 5 MB per op/sec scaling factor recognizes the fact that NFS file servers store significant amounts of data, and should have the storage capacity to scale as larger and more active user populations are served. The choice of 5 MB per op/sec of load is meant to represent the static file set residing on today's medium to high-end NFS servers. For example, the file set for a server providing NFS throughput of 500 ops/sec would be initialized to about 2.5 gigabytes (GB); a 2000 ops/sec NFS server would have a 10 GB file set.

Each file is initialized and filled with 136 kilobytes (KB) of data. The file data is written to the file server during the benchmark initialization phase to ensure that storage space for the file is allocated. A file size of 136 KB results in approximately 40 files for each 5 MB of data and each NFS op/sec of requested load.

Using a uniform file size allows the number of files created by LADDIS to scale along with the data set size and server load level. Initializing each file to 136 KB provides a number of convenient properties:

- For UNIX operating system-based NFS file servers, a 136 KB file is large enough to ensure that the server allocates and references first-level, indirect, file index blocks.

- It allows a single read or write operation to produce a stream of up to 17 NFS 8-KB READ or WRITE transfers that access a contiguous file region, resulting in file accesses that provide for both single-threaded, spatial locality and multi-threaded, temporal locality (through BIOD simulation).

- It provides a wide range of starting file offsets for read and write operations.
- It simplifies file selection criteria, hastening the file selection process during the benchmark's performance measurement phase, thereby minimizing client sensitivity effects.

Despite the numerous advantages, a uniform file size is unrealistic. However, LADDIS file access length is not based directly on file size, and has little impact on the performance model. File size distribution could be revisited in a future LADDIS release, but probably is not necessary.

The working set of files is chosen randomly from the total file set. The working set provides the set of files that LADDIS accesses when performing NFS requests. There are a number of important advantages to using a working set. It models real-world environments where not all files stored on the NFS file server are accessed on a regular basis. It forces the NFS server to be configured to support more storage space than is needed to execute the benchmark. Also, by creating a file set and then choosing a random subset to be the working set, LADDIS ensures that the files being accessed are distributed within a larger set of files. Further, using a subset of files should create more realistic file caching and disk access patterns on the NFS file server.

The working set comprises 20% of the total file set. Thus, the amount of data in the working set and the number of files in the working set both scale with the requested load level at a rate of 1 MB per op/sec and 8 files per op/sec, respectively. No firm scientific basis exists for establishing the working set to be 20% of the total file set. Based on collective experience within SPEC, applying a 80/20 rule-of-thumb to the ratio of static to dynamic files seems appropriate.

4.2.2 Directories, Symbolic Links, and Non-working Set Files

For operations that create and remove files, LADDIS incorporates a number of simplifying design assumptions. First, LADDIS does not initiate operations that are expected to fail. For instance, remove operations are only performed on files that already exists. Total error avoidance is unrealistic, but greatly simplifies the description of what is being measured by LADDIS—successful NFS requests only. To avoid performing failed operations, LADDIS explicitly manages the file set name space and maintains existence state information for each file.

Directory and symbolic link operations are performed on a set of 20 directories and 20 symbolic links on each load generator, divided equally among the load-generating processes. These fixed values are a gross simplification justified by the modest overall NFS performance impact of these operations. The NFS MKDIR, RMDIR, READDIR, SYMLINK, and READLINK requests together account for only 11% of the default LADDIS mix of operations. Although LADDIS provides parameters to set the number of directories and symbolic links, future versions should consider scaling these values along with the requested load level.

NFS REMOVE, RENAME and LINK requests, and most CREATE requests are performed on a small set of zero-length files that are not a part of the file working set. The number of these non-working set files is fixed at 100 on each load generator, divided equally among the load-generating processes executing on the load generator. Maintaining a distinct group of non-working set files simplifies LADDIS data set size management by isolating requests that, as a side-effect, impact the number of bytes of data residing on the server. LINK and RENAME requests are restricted to non-working set files to maintain a strict separation of the working set and non-working set name spaces. The size of the non-working set is 100 files, half of which are initialized during the benchmark initialization phase. This value is adjustable via a benchmark parameter.

Normally, NFS CREATE requests are performed on the non-working set files, but they are also used to help manage the data set size.

4.3 Data Set Size

Given an initial working data set size of 8 136-KB files for each op/sec of requested load (or approximately 1 MB of data per op/sec), LADDIS controls the fluctuation in the data set size as the benchmark executes. Two factors are involved: file write append operations cause the data set to grow, and file truncation operations cause it to shrink.

LADDIS performs two distinct types of write operations: writes that overwrite existing data in a file, and writes that append new data to the end of a file. These operations exercise different parts of the server's file system implementation. Depending on data buffer alignment, over-writing data may require pre-fetching the old data before overwriting part of it, while appending new data may require allocating space from the file system's pool of free storage space. Default LADDIS parameters specify that 30% of write operations over-write existing data and 70% of write operations append new data. Appending new data increases the total amount of data in the file set. If unchecked, append operations would cause uncontrolled growth of the data set size as each benchmark execution proceeded.

LADDIS limits data set growth to a 10% increase. The limit is implemented by using some of the NFS SETATTR and CREATE requests specified by the operation mix to perform file truncations (by setting the file length argument to 0). These special truncation operations are performed whenever a write append operation would cause data set size growth to exceed 10%. The truncation is performed just before the append. These truncation operations are included in the calculation of the operation mix percentages for SETATTR and CREATE.

Using SETATTR and CREATE requests to perform file truncation models real-world, UNIX operating open/truncate sequences, which are performed when updating a file from a temporary copy of the file. This approach produces a bi-modal distribution of response times for these operations; some SETATTR requests update a file's attributes, but others perform file storage space deallocation.

Recent research into file write access patterns [Hartman92] indicates that append ratios as high as 90 - 95% may be common in some environments. Using the default LADDIS settings, a stable data set size cannot be maintained with a 90% append ratio—there are not enough SETATTR and CREATE requests to counter-balance the append operations. Maintaining a stable data set size depends on the relative percentages of WRITE, SETATTR, and CREATE requests in the operation mix, the average file size, the average length of an append operation, and the data set fluctuation limit. The 70% append ratio maintains a stable data set size within the default LADDIS parameters.

The values for the write append percentage and data set growth limit parameters are interdependent. The 10% fluctuation value was chosen to minimize the number of SETATTR and CREATE requests required to maintain a stable data set size. All file truncations shorten existing files to zero length for the same reason. There is a strong desire to have LADDIS maintain a stable data set size so results produced at different load levels or with different run times will access a known file set and can be compared.

4.4 Read and Write Operations

LADDIS read/write operations impact NFS performance. Depending on how read/write operations are monitored, different characteristics become apparent. Viewed from the network, LADDIS controls the amount of data transferred in each network packet. From the perspective of an individual file access, LADDIS controls the length of each byte stream to and from the file. From the NFS file server's point of view, LADDIS controls the aggregate data transfer rate to and from the server. A number of LADDIS parameters interact to control these characteristics.

4.4.1 Network Packet Size Distribution

The NFS Version 2 protocol allows up to 8 KB of file data to be transferred in each network transfer (lower level transport protocols may subdivide the NFS packet further). Most NFS client kernel implementations provide data caching and some amount of file read-ahead and write-behind activity. Read-ahead and write-behind allow network transfers to be performed independently of an application's access pattern, producing fewer network packets and larger transfer sizes. As a result, most NFS READ/WRITE requests are transferred in network packets containing 8 KB of file data. However, because file lengths are seldom an exact multiple of 8 KB, transfers accessing the end of a file typically contain less than 8 KB of data.

LADDIS controls the amount of data contained in each NFS read/write transfer to produce a known distribution of network data packet sizes. The packet size for any given request is chosen randomly, but the choice is weighted to produce a target packet-size distribution. The default LADDIS network packet distribution produces approximately 90% read packets containing 8 KB of data, and 10% packets containing less than 8 KB. Roughly 50% of the write packets contain 8 KB, and 50% contain less than 8 KB. The distribution of "fragment" packets containing less than 8 KB of data is spread evenly across packets containing multiples of 1 KB. These values are derived from NFS packet transfers observed by one of the authors [Keith90].

The only simplification made by LADDIS is to limit fragment packet sizes to multiples of 1 KB and distribute them equally across 1 - 7 KB sizes, as shown in Figure 4-2.

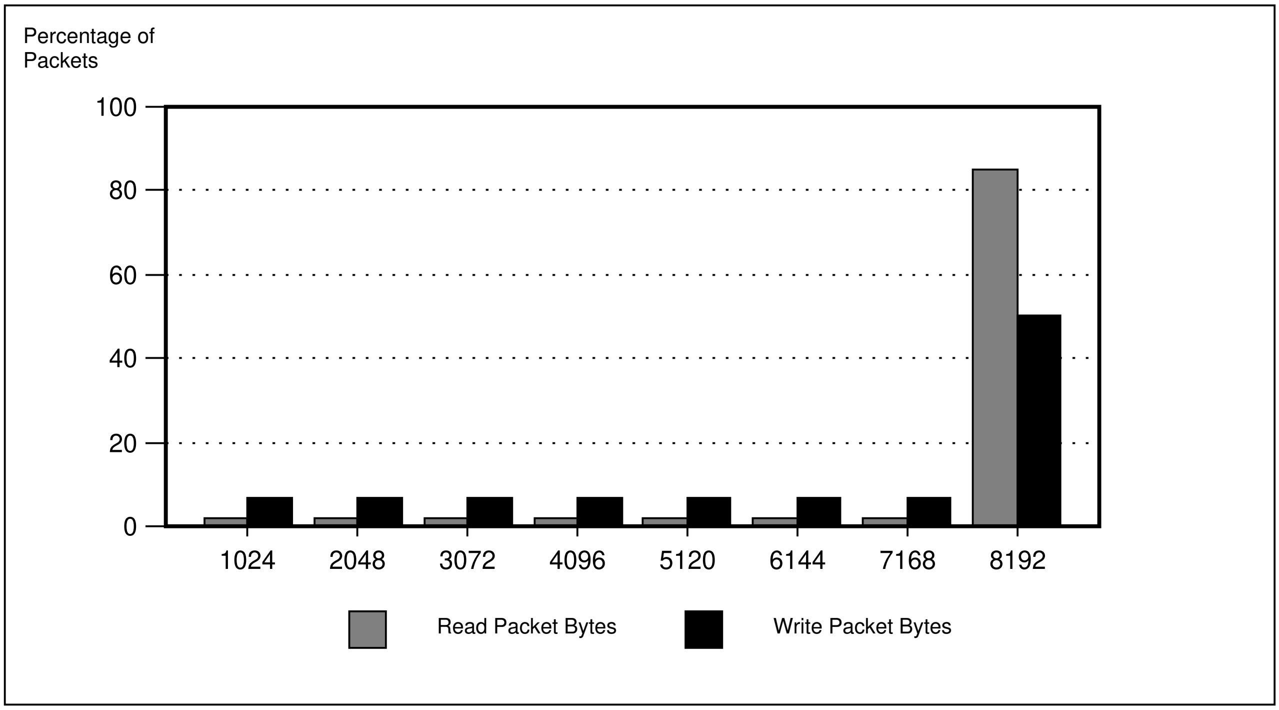

Figure 4-2 Read and Write Packet Size Distribution

4.4.2 File I/O Data Streams

LADDIS file accesses are modeled on byte stream operations that can be larger than the single NFS 8-KB packet size. High-level file read/write operations targeted at a single file can vary from 1 KB to 136 KB in length. The length of the operation is limited to the initial file size of 136 KB to ensure that most LADDIS test files can successfully accommodate the largest possible read operation. This artificial limit helps to simplify file choice overhead during the benchmark performance measurement phase.

Of course, the actual access to the file server is implemented as a stream of NFS 8-KB READ/WRITE requests, possibly followed by a single 1 - 7 KB fragment request. The distribution of high-level file accesses is tailored to produce the required percentage of 8 KB and 1 - 7 KB fragment network packets. This distribution is achieved by weighting the choice of file access length to produce many short accesses of 1 - 15 KB and fewer long accesses of 16 - 135 KB [Baker91].

Beyond packet sizes of 15 KB, the distribution is essentially logarithmic (e.g., half as many 128 KB accesses as 64 KB accesses, etc.), although some variations involving fragment packets are needed to meet the packet size distribution requirements. See Figure 4-3.

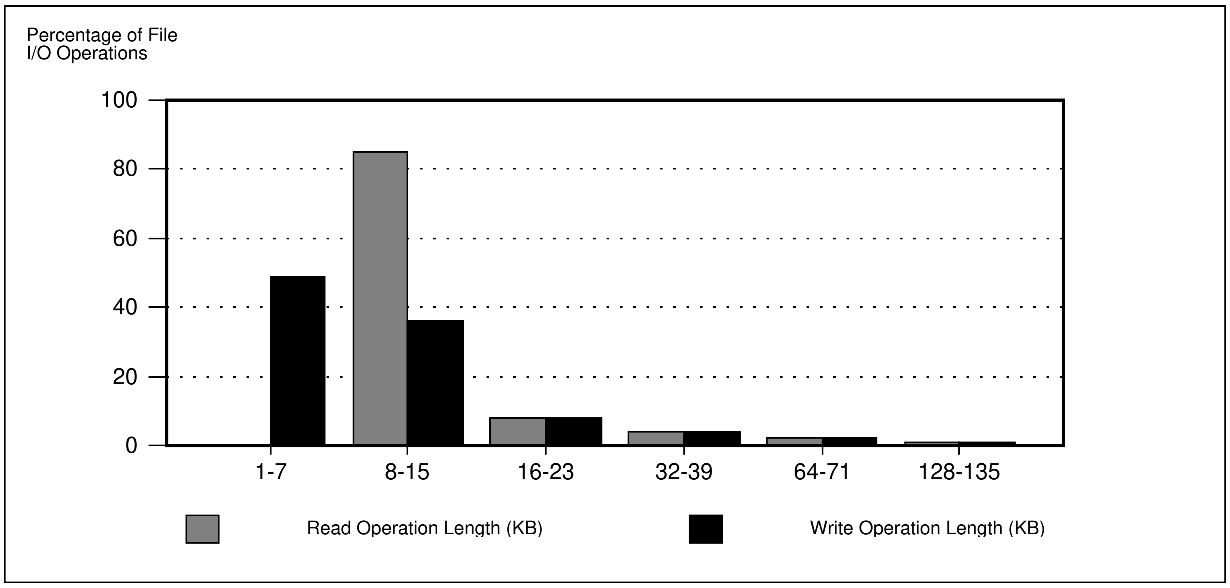

Figure 4-3 File I/O Operation Length Distribution

The starting file offset for each access is chosen to be an exact multiple of 8 KB. Because most NFS implementations perform client-side data caching, 8-KB boundaries are an obvious choice for file offsets in individual packet transfers. For read operations, the offset is chosen randomly from the set of file offsets that will allow the read operation to succeed without exceeding the current end-of-file. For write operations that over-write existing data, the starting offset is chosen as it is for reads. For write operations that append new data to end of the file, the current end-of-file offset is used as the starting offset.

This model of file access synthesizes client cache read-ahead and write-behind implementations and presents some spatial locality of file reference to the NFS server.

4.4.3 BIOD Simulation

LADDIS models temporal locality of file accesses by simulating NFS Block I/O Daemon (BIOD) file access patterns. The simulation is performed using asynchronous RPC requests for multi-packet file read/write operations.

Many NFS client implementations support BIOD threads of control, which perform file read-ahead and write-behind operations between the client kernel's data cache and the NFS server. Normally, a number of BIOD processes are supported, allowing many READ/WRITE requests to be sent to the server concurrently before awaiting a reply to the first request. The default LADDIS BIOD setting requires that whenever a LADDIS load-generating process performs a read/write operation longer than 8 KB, at least two NFS READ or WRITE requests are sent to the server before waiting for a reply.

This setting produces several effects on the NFS file server. First, because multiple READ/WRITE requests to the same file are sent to the server within a short time-span, the server may be able to provide service efficiencies based on the file access locality. Second, the total number of requests submitted to the server concurrently is not limited to the number of LADDIS load-generating processes. Rather, each LADDIS process may send multiple requests, generating a burst of increased load over a short period of time. Such a burst may introduce additional short-term stress on the file server's resources.

4.4.4 Aggregate Data Transfer Rate

A number of random factors affect the LADDIS data transfer rate at any given point during a test run. These factors include which operation is being performed and how much data is transferred by the read/write operations. The average data transfer rate can be calculated from the overall load rate, the percentage of READ/WRITE requests in the operation mix, and the average transfer size for each READ/WRITE request.

For the standard LADDIS distribution, the average transfer size for WRITE requests is 6 KB (50% 8-KB and 50% randomly distributed across 1 - 7 KB), and the size for READ requests is 7.6 KB (90% 8-KB and 10% randomly distributed over 1 - 7 KB). Thus, for a 100 ops/sec load rate, using the default operation mix of 15% WRITE requests and 22% READ requests, the data transfer rate is 100 ops/sec * ((0.15 * 6 KB/op) + (0.22 * 7.6 KB/op)) = 257 KB/sec.

5 Load Generation

LADDIS load generation consists of a paced stream of NFS requests, separated by random delays. Individual operations are chosen randomly but weighted according to the operation mix. The operation is performed on a randomly chosen file, with the choice weighted by a file access pattern distribution function, and constrained by the requirement to chose a file that will satisfy the operation successfully. Each NFS request is timed, and per-operation type statistics are maintained. Periodically, the load rate is adjusted as necessary.

5.1 Load Rate Management

In order for each LADDIS load-generating process to produce NFS requests at the specified load rate, the server's average request response time is estimated periodically.

LADDIS employs a warm-up phase to calibrate the server's average response time before beginning the benchmark performance measurement phase. Every 2 seconds during the warm-up phase, each LADDIS process calculates an average inter-request delay period that allows it to produce the requested load rate. The calculation is based on the server's average response time over the previous 2 seconds. The warm-up phase executes for 60 seconds. The warm-up phase duration was established empirically to produce a steady state of request generation for load levels as low as 10 ops/sec for an individual load-generating process.

At the end of the warm-up phase, all performance counters and statistics are reinitialized and further server recalibrations are performed at 10 second intervals. Between each operation, LADDIS pauses for a random time of from one half to one and a half times the current inter-request delay period before beginning the next operation.

When a shortfall or overage in load rate is observed, LADDIS aggressively adjusts the inter-request delay period to allow it to make up any cumulative request shortage or overage during the next measurement period. One effect of this algorithm is that during a test run where the requested load cannot be achieved, LADDIS quickly eliminates the inter-request delay. For load levels near the file server's capacity, this is the desired behavior. For load levels beyond the server's capacity, LADDIS may tend to overburden the server and introduce CPU inefficiencies on the load generator by attempting to produce more load than the server can handle. In this case, response time may rise significantly as may the number of errors due to RPC time-outs.

5.2 Operation Choice

LADDIS randomly chooses the next operation to perform based on a probability derived from the operation mix. Some operations cause more than one NFS request to be sent to the server. For instance, write operations may involve a data length that causes multiple WRITE request packets to be sent to the server. Based on the packet size distribution table, LADDIS calculates the average number of requests to be generated by each read/write operation and re-weights the operation mix accordingly.

By choosing operations randomly, LADDIS loses the ability to generate common NFS request sequences (e.g., a READDIR followed by a GETATTR, or a LOOKUP followed by a read). Operation context sensitivity would be a nice addition to a future version of the benchmark.

5.3 File Choice

File choice depends on the selected operation type. For directory and symbolic link operations a file is chosen randomly from the small set of files supporting that operation. The choice is constrained only by the existence attribute of the file, so that the operation will succeed. Similarly, file creation, removal, and naming operations access a random file chosen from the non-working set files, according to the same requirements.

If an appropriate file cannot be found (e.g., no directories currently exist, but a RMDIR request is to be performed), then LADDIS randomly chooses another operation to perform. An operation that cannot be performed immediately due to lack of an appropriate file may be successful later. If too many operations cannot be performed for this reason, then the test run will not achieve the target operation mix and will be flagged as invalid. This can result from specifying an operation mix that the benchmark cannot produce, for instance, specifying a very high percentage of remove operations and a very low percentage of create operations.

The remaining operations choose files from the static file working set. Files selected for read/write operations must be long enough to accommodate the operation. The length of each read/write operation is chosen randomly, and weighted according to the file access length distribution. For read and over-write operations, the file chosen must be long enough for the read to complete successfully.

5.4 File Access Patterns

When choosing a file from the working set to be accessed, LADDIS implements a non-uniform, file access distribution algorithm. During the benchmark initialization phase, the working set is partitioned into small groups of files, and each group is assigned a probability of providing the next file to be accessed. The probabilities are based on a Poisson distribution. Each group contains the same number of files, and the files within each group have an equal chance of being selected (constrained by the file length requirements for read and over-write operations).

As the benchmark load level is increased during successive test runs, additional files are created in the working set, and the number of file groups is increased. Scaling the number of file groups with the load level adds more values to the Poisson distribution and smooths the file access distribution curve. The Poisson distribution is applied to groups of files rather than individual files to improve the chances that a file selected for a read/write operation will be long enough to support the operation.

LADDIS employs a non-uniform, file access distribution to increase file access locality. The desired effect is to increase file server buffer cache hit rates. Other non-uniform mechanisms have been suggested to increase locality further, including Markov processes. It is thought that file access choice based on recent file access patterns might provide a more realistic model of file access. The implementation of the LADDIS file selection algorithm has been isolated from the main code path to encourage future experimentation.

5.5 Response Time Measurements

After selecting an operation and a file, LADDIS sends the request or series of requests to the server. The elapsed time of sending each NFS request and waiting to receive a reply is the basis for determining the server's response time. The measurement includes time spent executing in the local RPC library and network protocol code path while sending and receiving packets, processing time required to determine the elapsed time, and any additional client platform operating system overhead or interference from other LADDIS processes executing concurrently on the load generator system. Asynchronous RPC calls are handled similarly, but include a small amount of extra processing required to manage multiple concurrent requests.

The accuracy of each individual measurement depends on the granularity of the client's gettimeofday(1) implementation. Assuming that the error in each measurement is random, an average value derived from a large number of sample measurements produces an unbiased estimate of the actual average response time. For each operation type, the benchmark reports a 95% confidence interval for the mean response time reported.

Only successful requests are included in these results. Failed requests are tallied separately, but are not included in response time measurements or in the server's throughput calculation.

Except for rare, unexpected errors reported by the server, failed requests indicate RPC time-out errors. LADDIS uses a single, non-adaptive time-out mechanism to define the maximum allowable service time for each request. This is a simpler mechanism than is used by most NFS implementations, but reduces the overhead required to generate NFS load. Three RPC time-out classes are defined, and no requests are retried. The three classes are for 1, 2, and 3 seconds and are applied to file attribute read requests, file data read requests, and file data and attribute write requests, respectively.

5.6 Test Duration

The performance measurement phase of each benchmark execution (each performance graph data point) lasts for 10 minutes. The time period was established empirically as the minimum time required for each LADDIS load-generating process to achieve a close approximation of the operation mix, when generating requests at the rate of 10 ops/sec. Normally, test runs are conducted at higher load rates with multiple processes, and they achieve an aggregate operation mix approaching the target mix in only a few minutes.

The duration of the test run can significantly impact the amount of file data that becomes cached in the file server's data buffer caches. If a large percentage of the data set can be accommodated in the server's buffer cache, longer warm-up periods and longer test runs may produce better results. Therefore, only results from test runs of the same duration are comparable.

5.7 Algorithm Trade-offs

Each step in the LADDIS load generation algorithms involves trade-offs between implementing more realistic load generation behavior and reducing the internal overhead required to produce NFS requests. For instance, operation choice could be made sensitive to the recent operations, but this would require additional algorithmic complexity to achieve the desired mix of operations. Non-uniform file access could be applied to symbolic link operations with only a minor increase in processing time, but, for the default LADDIS parameter settings, would have a negligible impact on server performance. RPC time-out handling could be improved, but would complicate reporting procedures.

Most of these issues could be addressed by increasing the overhead involved in generating requests and would reduce LADDIS's load-generating capacity. Extra load generators could be required to measure an NFS server. Also, executing additional code within LADDIS introduces additional client sensitivity. LADDIS attempts to balance the goals of realistic load generation and increased load-generating capacity by focusing on the areas with the greatest impact on server performance and simplifying the approach in less critical areas.

6 Benchmark Results

Each execution of the LADDIS benchmark produces a detailed results report for each load generator, and an aggregate results report combining the results from all load generators involved in the test run. Each report includes detailed information for each NFS operation type, parameter setting and file set information, and a summary description of server throughput and average response time, as shown in Figure 6-1. If more than 1% of the NFS requests failed, including timed-out requests, LADDIS indicates that the test run is invalid.

The throughput and average response time reported by individual test runs are combined into a LADDIS performance graph showing the server's response times over a range of load levels. This graph and all LADDIS configuration information is combined on a standardized, 2-page, SPEC SFS Release 1.0 performance report. This is the highly recommended way to judge overall file server performance results. Results may be reported as "baseline" or "non-baseline" results. Baseline results are for server configurations that conform to all standard NFS service requirements defined by the NFS Version 2 Protocol and the SPEC SFS 1.0 Run and Report Rules [SPEC93].

```
Aggregate Test Parameters:
    Number of processes = 36
    Requested Load (NFS operations/second) = 240
    Maximum number of outstanding biod writes = 2
    Maximum number of outstanding biod reads = 2
    Warm-up time (seconds) = 60
    Run time (seconds) = 600
    File Set = 9612 Files created for I/O operations
               1908 Files accessed for I/O operations
                    (each prefilled to 136 KB)
                612 Files for non-I/O operations
                144 Symlinks
                144 Directories
                    Additional non-I/O files created as necessary

LADDIS Aggregate Results for 6 Client(s), Fri Mar 12 18:54:07 1993
LADDIS NFS Benchmark Version 0.1.22, Creation - 10th December 1992
------------------------------------------------------------------------
NFS      Target Actual   NFS    NFS    Mean     Std Dev  Std Error   Pcnt
Op        NFS    NFS      Op     Op    Response Response of Mean,95%  of
Type      Mix    Mix    Success Error   Time      Time    Confidence  Total
          Pcnt   Pcnt    Count  Count  Msec/Op  Msec/Op  +- Msec/Op  Time
------------------------------------------------------------------------
null       0%    0.0%        0     0     0.00     0.00      0.00     0.0%
getattr  13%    13.1%    18962     0     5.71     5.51      0.03     2.7%
setattr   1%     0.8%     1293     0    34.52    35.71      0.33     1.1%
root       0%    0.0%        0     0     0.00     0.00      0.00     0.0%
lookup   34%    33.6%    48504     0     7.35     7.59      0.02     9.0%
readlink  8%     8.1%    11815     0     5.73     5.56      0.04     1.7%
read     22%    21.7%    31410     0    15.73    18.05      0.05    12.5%
wrcache   0%     0.0%        0     0     0.00     0.00      0.00     0.0%
write    15%    15.5%    22430     0   116.36   127.54      0.15    66.4%
create    2%     1.9%     2857     0    41.03    49.43      0.26     3.0%
remove    1%     0.9%     1350     0    11.88    13.30      0.19     0.3%
rename    0%     0.0%        0     0     0.00     0.00      0.00     0.0%
link      0%     0.0%        0     0     0.00     0.00      0.00     0.0%
symlink   0%     0.0%        0     0     0.00     0.00      0.00     0.0%
mkdir     0%     0.0%        0     0     0.00     0.00      0.00     0.0%
rmdir     0%     0.0%        0     0     0.00     0.00      0.00     0.0%
readdir   3%     2.9%     4243     0    25.15    23.06      0.14     2.7%
fsstat    1%     0.9%     1422     0     5.88     6.34      0.13     0.2%
------------------------------------------------------------------------

          ---------------------------------------------------
          | LADDIS VERSION 0.1.22 AGGREGATE RESULTS SUMMARY |
          ---------------------------------------------------

NFS THROUGHPUT: 240.45 Ops/Sec AVG. RESPONSE TIME: 27.23 Msec/Op
NFS MIXFILE: [ LADDIS default ]
AGGREGATE REQUESTED LOAD: 240 Ops/Sec
TOTAL NFS OPERATIONS: 144286 TEST TIME: 600 Sec
NUMBER OF LADDIS CLIENTS: 6
TOTAL FILE SET SIZE CREATED: 1307232 KB
TOTAL FILE SET SIZE ACCESSED: 259488 - 285390 KB (100% to 109% of Base)
```

Figure 6-1 Detailed Aggregate Results Report

7 Areas for Future Work

LADDIS provides a rich set of workload parameter settings that could be used to create workload abstractions for CASE, CAD, office automation, and other common NFS environments. Second-order parameter sets could be created to model combinations of these application environments. Workload studies are needed to establish the basis for these abstractions. The current default LADDIS workload was derived from a variety of research studies, anecdotal evidence, rules of thumb, and common sense. Future workload characterizations should encompass all key factors affecting NFS server performance within a single study. These factors include NFS operation mix, network packet characteristics, file set size and distribution by file type, data set size and distribution within the file set, working set size, and file access patterns.

The implementation of LADDIS lends itself to adaptation to other benchmark situations. For instance, isolating the actual formulation of NFS RPC request packets within separate subroutines facilitates the creation of a new LADDIS implementation supporting future releases of the NFS protocol. Additionally, the LADDIS workload parameterization and load generation framework can be applied to other distributed file systems such as DCE/DFS. Changing LADDIS to produce local operating system calls (like nhfsstone) instead of network requests (like LADDIS) would greatly impact client sensitivity; however, the existing LADDIS load generation algorithms and workload parameterization would probably still be applicable.

The fundamental LADDIS design goal of reducing client sensitivity has been achieved through a number of design decisions. SPEC experience indicates that using fast, efficient NFS client platforms as LADDIS load generators minimizes the client platform’s impact on performance measurements. However, as LADDIS is deployed throughout the industry and additional experience is gained on a broad set of client platforms, there will be an increased need for careful quantification of LADDIS client sensitivity effects.

There are also a number of technical improvements that could be made to LADDIS including:

- shared files and directories, among multiple LADDIS processes,
- test files distributed across a hierarchical directory structure,
- randomization factors included in file name generation,
- variable file data lengths,
- the number of directories, symbolic links, and non-working set files scaled with load level,
- improved RPC time-out and error mechanisms,
- separate response time reporting for truncation operations,
- operation choices sensitive to recent request streams,

- additional file access distribution mechanisms.

However, the specific benefit that each of these changes might introduce into the performance measurement must be weighed carefully against client sensitivity effects introduced by the algorithmic changes. Further, the potential impact that the change will have on the benchmark's accuracy and ability to produce the desired workload abstraction must be considered.

8 Acknowledgments

The LADDIS benchmark is the result of the efforts of many individuals over the past three years: Ken Teelucksingh (Digital) provided the multi-client support, John Corbin (Sun) contributed the asynchronous RPC code, Santa Wiryaman (Digital) wrote the validation suite, and Richard Bean (Data General) did the original port from nhfsstone.

The LADDIS Group was critical to the technical evolution of the benchmark: Bruce Nelson (Auspex) inspired and chaired the LADDIS effort, Bob Lyon (Legato) provided an important historical perspective, Maneesh Dhir (Sun) and Peter Olenick (Interphase) debugged and characterized early versions of the benchmark, Brian Pawlowski (Sun) contributed significant technical input, Janet Johnson (NCSU) helped out with statistical analysis, and a number of others discussed requirement and design issues over the course of two years.

The SPEC SFS subcommittee played a crucial role in turning the benchmark into a product: Krishna Dronamraju (AT&T/NCR) and Jeff Reilly (Intel) integrated the SPEC tools and user interface, Tom Spuhler (HP), Rick Jones (HP), and Andy Watson (Auspex) were key contributors to workload discussions, the SPEC membership ported LADDIS to a broad range of platforms, and the SPEC steering committee and board members helped resolve a number of key issues and focused efforts toward releasing the benchmark.

9 Author Information

Mark Wittle is a Principal Software Engineer in the DG/UX[7] Development Group at Data General Corporation. Mark has been the primary technical contributor to the design and implementation of LADDIS load generation functionality. Mark holds a bachelors degree in Mathematics from Knox College in Galesburg, IL, and attended the University of Illinois graduate Computer Science program for two years. He can be reached at wittle@dg-rtp.dg.com, or, Data General Corp, 62 Alexander Drive MS-109, Research Triangle Park., NC 27709.

Bruce Keith is the DEC OSF/1[8] AXP[8] NFS performance project leader in the Systems Engineering Characterization Group at Digital Equipment Corporation. He was the chairman of the SPEC SFS subcommittee and was the technical project leader for LADDIS and SFS release manager within SPEC. Bruce received his B. S. degree in Computer Science from Worcester Polytechnic Institute. He can be reached at keith@oldtmr.enet.dec.com, or, Digital Equipment

Corporation, 85 Swanson Road BXB1-1/F11, Boxboro, MA 01719.

10 References

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

[Dronamraju93] Dronamraju, S. Krishna, et al., "Managing the SFS User Interface," SPEC Newsletter, Volume 5, Issue 1, pp. 5 - 10, March 1993.

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

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

[Legato89] "nhfsstone" NFS load-generating program, Legato Systems Inc., Palo Alto, CA 94306

[Lyon92] Lyon, Robert, private communication to the LADDIS group, June 26, 1992.

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

[Shein89] Shein, Barry, et al., "NFSSTONE - A Network File Server Performance Benchmark," Proceedings of the Summer 1989 USENIX Conference, pp. 269-274, June 1989.

[SPEC93] Standard Performance Evaluation Corporation, "SPEC SFS Release 1.0 Run and Report Rules," SPEC SFS 1.0 Release, SPEC c/o NCGA, 2722 Merrilee Drive, Suite 200, Fairfax, VA 22031-4499

[Stern92] Stern, Hal, et al., "NFS Performance and Network Loading," Proceedings of the LISA VI Conference, pp. 33 - 38, October 1992.

[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.

11 Trademarks

[1] NFS and ONC are trademarks of Sun Microsystems, Inc.

[2] Nhfsstone source code is a copyrighted product of Legato Systems, Inc.

[3] SPEC and SPECnfs_A93 are trademarks of the Standard Performance Evaluation Corporation.

[4] OSF/1 is a registered trademark of Open Software Foundation, Inc.

[5] Ethernet is a trademark of Xerox Corporation.

[6] UNIX is a registered trademark of UNIX Systems Laboratories.

[7] DG/UX is a trademark of Data General Corporation.

[8] DEC OSF/1 and AXP are trademarks of Digital Equipment Corporation.