subject: **SPEC SFS Release 1.1 Run Rules**

date: **November 28, 1994**

from: **SPEC Steering Committee**

*ABSTRACT*

This paper provides rules to follow for all submitted or reported runs of the SPEC System File Server (SFS) Benchmark suite according to the norms laid down by the SPEC SFS subcommittee and approved by the SPEC Open Systems Steering committee. This is a companion paper to "SPEC SFS Release 1.1 Reporting Rules", which provides the rules for reporting results of official runs of the SPEC SFS Release 1.1 Benchmark suite. These papers can be found in files RUN_RULES and RPT_RULES in the $SPEC directory on the release tape.

Memorandum to
Performance Analysts

subject: **SPEC SFS Release 1.1 Run Rules** date: **November 28, 1994**

from: **SPEC Steering Committee**

# *1. SPEC SFS RELEASE 1.1 RUN RULES*

The general philosophy behind this set of rules for running the SPEC SFS Release 1.1 suite is to ensure that a user can reproduce the results reported by a results submitter.

1. All data reported must be gathered from benchmark runs conducted according to the SFS Release 1.1 Run Rules that produce valid results.
2. The complete hardware, software, and network configuration must be documented within the confines of the reporting pages.
3. Use of software features (in pre-processors, compilers, network protocol suites, etc.) that invoke, generate or use software designed specifically for the SPEC SFS Release 1.1 Benchmark is not allowed.
4. The system, including all hardware, software, and network components must be available for general customer shipment within six months of the date of SPEC result publication.

## *1.1 Vendor Makefile Wrappers*

Generic makefiles are provided in the benchmark directories. Typically a vendor makefile wrapper (M.vendor) is used in conjunction with the generic makefile to run the benchmark. The vendor makefile wrappers are used to compile and run the benchmark for a vendor's specific hardware and software platform.

The makefiles may be modified in a "performance-neutral" fashion to facilitate running benchmark on proprietary operating systems.

## *1.2 Software*

In addition to the base operating system you will require the C compiler; the awk, id, make, nroff, ps, and sed system utilities; the System V version (sh5 on BSD-based systems) of the Bourne shell; and the ed line editor. You will also require the software necessary to run the NFS Version 2 protocol on the server being benchmarked and the NFS client systems used as LADDIS load generators.

All of the software components are required to be the standard software components distributed by the vendor. All the rules specified in the beginning of this Section, "SPEC SFS Release 1.1 Run Rules", apply to all software components. Use of benchmark-specific software components is not allowed.

## *1.3 Libraries*

Special libraries may be used as long as:

1. They are products that must be available for general customer shipment within six months of the date of SPEC result publication.

2. They do not require changing source code,
3. They do not replace routines in the benchmark source code,
4. They are not "benchmark-specific".

Libraries can be specified in the makefiles.

*1.4 Pre-Processors*

The use of compiler pre-processors is governed by the following rules:

1. Use of a pre-processor on unmodified source code that is part of the identified released software is allowed and must be documented.
2. Use of benchmark-specific pre-processor features are not allowed.
3. Documented pre-processor options may be used, and must be specified on the results reporting page and in the vendor makefile (M.vendor) wrapper.

*1.5 Source Code Changes*

SPEC permits minimal performance-neutral portability changes. When source code changes are made a diff listing of the changes must be included in the testing report. All changes must be reviewed and deemed "performance-neutral" by the OSSC. These results must be marked with an asterisk ("*") to signify that a portability change was required. The asterisk must accompany such results the first time they are reported in a SPEC Newsletter. If the OSSC accepts these changes as "performance-neutral," the asterisk may be removed from subsequent results using these changes.

Source code changes required for standards compliance should be reported. Appropriate standards documents should be cited. All such changes should be reported to SPEC. SPEC may consider incorporating such changes in future releases. Whenever possible, SPEC will strive to develop and enhance the benchmark to be standards-compliant.

The portability change is allowed if, without the change, the:

1. Benchmark code will not compile,
2. Benchmark code does not execute, or,
3. Benchmark code produces invalid results.

For additional information concerning changes to source code, refer to Section 10, "SPEC SFS Release 1.1 Reporting Rules".

*1.6 Server Features*

*1.6.1 General Features* The following are the requirements for the use of general or special server hardware and software features:

1. If special server hardware or software features are used during the execution of the benchmark, they must be documented.
2. The features must be available for general customer shipment within six months of the date of SPEC results publication.
3. The server hardware or software features used when running the benchmark must meet the requirements for SPEC SFS Release 1.1 results being generated, as defined in Section 9.6.1.1.

*1.6.1.1 SPEC SFS Results Types*

The SPEC SFS Release 1.1 Reporting Rules define which results may may be reported ("reportable results"). SPEC SFS Release 1.1 Reporting Rules form the basis for determining which server hardware and software features are allowed when running the 097.LADDIS benchmark. Furthermore, results for certain types of server configurations may not be reported and are denoted exclusions.

*1.6.1.1.1 Reportable Results*

The reportable result type is defined to be those benchmark results for which the submitter:

1. Asserts that the server adheres to the NFS Version 2 Protocol Specification, particularly the protocol requirement that for NFS write operations, NFS servers must not reply to the NFS client before any modified file system data and metadata are written to stable storage, as quoted in Section 9.6.1.1.3.
2. Asserts that the server adheres to the SPEC description of stable storage specified in Section 9.6.1.1.3.
3. Asserts that the server passes 097.LADDIS validation.
4. Asserts that the server configuration for running the 097.LADDIS benchmark, adheres to the following description, which applies to all architectures:

   > *SPEC intends that for every network, all file systems should be accessed by all clients uniformly.*

   Once the number of load generating processes has been determined, then load generator mount points should distribute file systems in the following manner.

   Using a round-robin assignment, select the next file system to mount by selecting from the following collection, varying first (a), then (b), then (c), and so on:

   a. next network,
   b. next cluster processor (if clustered system),
   c. other controllers in the path from the network, to the file system,
   d. file system.

   Several examples are useful to illustrate this algorithm.

   I. n-level symmetric multiprocessors (include uniprocessor, i.e. n=1).
      a. Select next load-generating process for a client.
      b. Select next network accessed by that client.
      c. Select next network controller on the network.
      d. Select next disk controller
      e. Select next file system.

   II. Cluster system.
      a. Select next load-generating process for a client.
      b. Select next network accessed by that client.

c. Select next cluster processor on the selected network.

d. Select next network controller on cluster controller.

e. Select next disk controller on cluster controller.

f. Select next file system on controller.

III. Functional Multiprocessing.

a. Select next load-generating process for a client.

b. Select next network accessed by that client.

c. Select network processor.

d. Select next file processor.

e. Select next storage processor.

f. Select next file system.

*This foundation for this run rule can be found in a paper entitled "Server Configuration Requirements" in the documents directory, filename "ServerConfiguration.ps". This postscript file can be printed using the command:*

*lp -opostscript ServerConfiguration.ps*

*1.6.1.1.2 SPEC SFS Excluded Results*

SPEC SFS Release 1.1 results may not be reported for NFS servers which incorporate:

1. Volatile memory-based filesystems such as RAM disk.
2. Filesystems that do not survive server reboot.

*1.6.1.1.3 SPEC SFS Description Of Stable Storage*

The Sun Microsystems, Inc. publication "Network Programming," Revision 50, of December 15, 1987, page 174 states the following concerning the NFS protocol:

> *All of the procedures in the NFS protocol are assumed to be synchronous. When a procedure returns to the client, the client can assume that the operation has completed and any data associated with the request is now on stable storage. For example, a client WRITE request may cause the server to update data blocks, filesystem information blocks (such as indirect blocks), and file attribute information (size and modify times). When the WRITE returns to the client, it can assume that the write is safe, even in case of a server crash, and it can discard the data written. This is a very important part of the statelessness of the server. If the server waited to flush data from remote requests, the client would have to save those requests so that it could resend them in case of a server crash.*

Unfortunately, the above excerpt from the NFS protocol specification does not adequately define "stable storage." To resolve this ambiguity for the purposes of the SFS Benchmark, SPEC defines stable storage in terms of the following operational description:

> *NFS servers must be able to recover without data loss from multiple power failures (including cascading power failures, i.e., several power failures in quick succession), operating system failures, and hardware failure of components (e.g., CPU) other than the storage medium itself (e.g., disk, non-volatile RAM).*

This description, coupled with the NFS Version 2 Protocol's requirement as quoted above, that "for NFS write operations, NFS servers do not reply to the NFS client before any modified file system data and metadata are written to stable storage", lead to the following examples of stable storage:

i. Media commit of data, i.e., the modified data has been successfully written to the disk media, for example, the disk platter.

ii. An immediate reply disk drive with battery-backed on-drive intermediate storage or uninterruptible power system. (UPS)

iii. Server commit of data with battery-backed intermediate storage and recovery software.

iv. Cache commit with uninterruptible power system (UPS) and recovery software.

Conversely, the following are not examples of stable storage:

i. An immediate reply disk drive without battery-backed on-drive intermediate storage or uninterruptible power system. (UPS)

ii. Cache commit without both uninterruptible power system (UPS) and recovery software.

iii. Server commit of data without battery-backed intermediate storage & memory.

### *1.6.2 Hardware Features*

The following are the requirements for the use of general or special server hardware features:

1. The use of volatile memory-based filesystems such as RAM disk on the server are prohibited.
2. The use of filesystems that do not survive reboot of the server are prohibited.

*1.6.3 Software Features* The following are the requirements for the use of general or special server software features:

1. User Datagram Protocol (UDP) checksums must be calculated for NFS request and reply messages, hence checksums must be enabled on clients.

## *1.7 Preparing the System for Benchmark Run*

### *1.7.1 General Hardware Requirements*

The 097.LADDIS benchmark exercises the CPU, memory, disk, filesystem, and network subsystems of the server being tested. Thus, experimentation may be necessary to determine the amount of memory, number of disks and disk controllers, and number of networks and network controllers required to achieve "best" performance.

### *1.7.2 General Software Requirements*

In addition to the base UNIX operating system you will require the C compiler; the awk, id, make, nroff, ps, and sed system utilities; the SVR5 version (sh5 on BSD- or OSF-based systems) of the Bourne shell; and the ed line editor. You will also require the software necessary to run the NFS Version 2 protocol on the server being benchmarked and the NFS client systems used as LADDIS load generators.

*1.7.3 Disk Space Requirements For 097.LADDIS*

The SPEC SFS Release 1.1 benchmark uses a collection of Network File System (NFS) clients to generate a load on the NFS server being tested. The NFS clients are called "LADDIS load generators." One LADDIS load generator, denoted the "prime LADDIS load generator," is selected to control the benchmark and consolidate the results and run logs of the individual LADDIS load generators.

The 097.LADDIS benchmark requires disk space on the server being tested and the NFS clients used as LADDIS load generators. The benchmark requires a variable amount of disk space on the server for the target file data generated by the LADDIS load generators. On the LADDIS load generators, the benchmark requires a fixed amount of disk space for benchmark programs and binaries, and a variable amount of disk space for capturing benchmark results and run logs. The prime LADDIS load generator requires additional disk space for the consolidated results of all of the individual LADDIS load generators.

Disk space requirements for the 097.LADDIS benchmark can be summarized as follows:

1. Server Requirements: The server must be configured to provide an amount of disk space equal to five megabytes (5 Mbytes = 5 * 1,048,576 bytes) per NFS operation per second generated by the collection of LADDIS load generators, plus an additional 20% for fileset growth during benchmark execution. For example, at a load level of 100 NFS operations per second, the server must provide 600 Mbytes of disk space for the files generated by the benchmark. The 600 Mbyte disk space requirement for this example can be calculated as follows:

   [600 Mbytes = ((5 Mbytes)*(100 NFS operations per second)) +
   ((0.20)*(5 Mbytes)*(100 NFS operations per second))]

2. LADDIS Load Generator Requirements: Each LADDIS load generator requires approximately 4 Mbytes of disk space for the benchmark source code and binaries. Further, each load generator requires approximately 6 Kbytes (6 Kbytes = 6 * 1024 bytes) of disk space for benchmark results and log files for each load level (test point).
3. Prime LADDIS Load Generator Requirements: In addition to the LADDIS load generator disk space requirements described in item 2 above, the prime LADDIS load generator has a further disk space requirement for consolidated benchmark results and log files from each load generator of 30 Kbytes + ((6 Kbytes)*(the number of 097.LADDIS load generators)) for each load level (test point).

*1.8 Running the SPEC SFS Release 1.1 Benchmark*

*1.8.1 Requirements*

This section details the requirements governing how the SPEC SFS Release 1.1 benchmark is to be run for the purpose of generating reportable SPEC SFS Release 1.1 results.

*1.8.1.1 097.LADDIS Benchmark Execution Requirements*

The following 097.LADDIS benchmark execution requirement must be followed when generating reportable SPEC SFS Release 1.1 results:

1. NFS servers must perform write operations in a manner that is consistent with the requirements for the Reportable SPEC SFS result type, as described in Section 9.6.1.1.1.

Each benchmark run consists of a set of requested NFS load levels for which an average response time measurement is made. The benchmark measures the actual load level achieved and the associated average response time for each of the requested NFS load levels.

At the start of each benchmark run, i.e., before the first in a series of requested NFS load levels is generated, the target

filesystems on the NFS file server being benchmarked must be initialized to the state of a newly-created, empty filesystem. For UNIX-based systems, the "mkfs" (make filesystem) or "newfs" (new filesystem) command would be used for each server filesystem to be used by the benchmark. For non-UNIX-based systems, a semantic equivalent to the "mkfs" or "newfs" command must be used.

The measurement of all data points used to define a performance curve is made within a single benchmark run, starting with the lowest requested NFS load level and proceeding to the highest requested NFS load level. The requested load levels are specified in a list, from lowest to highest, from left to right, respectively, in the LADDIS parameter file laddis_rc.

The "runsfs" script described in Section 8 allows the user to enter the information required for a benchmark run, prompting the user for any missing information. When all pertinent information has been specified by the user, the runsfs script generates a 097.LADDIS parameter file for a benchmark run.

Please refer to "SPEC SFS Release 1.1 Reporting Rules", Section 10, for the required measurement points necessary to report a complete server response/throughput curve.

No server or testbed configuration changes, server reboots, or file system initializations (e.g., "newfs") are allowed between requested NFS load levels.

If any requested NFS load level must be rerun for any reason, the entire benchmark run must be restarted, i.e., the server's filesystems must be initialized and the series of requested NFS load levels repeated.

#### *1.8.1.2 097.LADDIS Benchmark Parameter Requirements*

All 097.LADDIS benchmark default parameter values must be used when generating reportable SPEC SFS results, except as noted in the following list:

1. **Load:** A collection of NFS clients called LADDIS load generators is used to generate an aggregate NFS load on the server being tested. For a testbed composed of 'N' LADDIS load generators, each generator produces 1 'N'-th of the load given the design of the 097.LADDIS benchmark.

2. **Server Fileset:** The size of the fileset generated on the server by the 097.LADDIS benchmark is established as a function of requested NFS throughput. Thus, fileset size is dependent on NFS throughput across the entire results curve. This provides a more realistic server load since more files are being manipulated on the server as the NFS load on the server is increased. This reflects typical server use in actual computing environments.

   The total fileset created by the 097.LADDIS benchmark is sized at 5 MBytes per NFS operation per second. The number of files within the total fileset is established based on the relationship of 8 files per megabyte of fileset size.

   The 097.LADDIS benchmark accesses a subset of the total fileset during load generation. The subset is denoted the working fileset or simply, working set. The working set is sized at 20% of the total fileset size, i.e., 1 Mbyte per NFS operation per second. The files in the working set are selected according to a Poisson distribution of 20% of files in the total fileset.

   The default parameters of the 097.LADDIS benchmark allow the benchmark to automatically create a total fileset and working fileset on the server that adhere to these rules.

3. **Number of Server Filesystems:** The number of server filesystems used to contain the total fileset is determined by the individual executing the benchmark. The selected number of filesystems must allow the total fileset size to be achieved for the requested NFS load levels. The number of server filesystems is specified by the MNT_POINTS parameter in the laddis_rc parameter file.

4. **LADDIS Load Generators:** At least two LADDIS load generators must be used per network supported by the server. For example, if a results submitter claims that a given server supports four Ethernets, then at least eight load generators (two load generators on each of the four Ethernets) must be used in the testbed configuration.

   Note: Bridged Ethernets require that two load generators be on each physical segment, rather than logical segment.

   It is recommended that "diskful" NFS clients (systems with a local disk supporting the operating system, swap space, and user files) be used as LADDIS load generators. This recommendation is made to ensure that performance measurements made by the 097.LADDIS benchmark are not skewed by network traffic not generated by the 097.LADDIS benchmark. For the same reason, it is also recommended that load generator log files be created on the local disk of the diskful NFS client.

   The NFS clients used as LADDIS load generators are defined by the CLIENTS parameter in the laddis_rc parameter file.

5. **Number of Processes per Load Generator:** The number of LADDIS subprocesses per LADDIS load generator is determined by the individual running the benchmark according to the following guideline. The number of subprocesses per load generator is specified by the PROCS parameter in the laddis_rc LADDIS parameter file.

   A minimum of eight LADDIS subprocesses per network supported by the server must be used. For example, for a testbed consisting of two LADDIS load generators targeting a server on one Ethernet, at least eight LADDIS subprocesses must be used on the load generators, hence at least four subprocesses per LADDIS load generator.

6. **Biod Emulation Parameters:** The 097.LADDIS benchmark emulates NFS client block-I/O daemon ("biod") read-ahead and write-behind functionality. A minimum of two outstanding read operations, established by the BIOD_MAX_READS parameter in the laddis_rc file, and a minimum of two outstanding write operations, established by the BIOD_MAX_WRITES parameter in the laddis_rc file, must be used per LADDIS load generator subprocess. This is the default behavior of the benchmark established by the default benchmark parameters.

7. **Time parameters:** RUNTIME, the time of measurement for which results are reported, must be the default 600 seconds for reportable results. The WARMUP_TIME must be set to the default of 60 seconds for reportable results.

8. **Isolated LADDIS network:** LADDIS should be run in an isolated network setup in order to obtain valid results. Results obtained on "production" networks are invalid as they will most likely not be reproducible. Such results are probably meaningless. Further, LADDIS may fail to converge to the requested load rate and behave erratically due to varying ambient load on the network.

### *1.8.2 Method*

The method of using the benchmark tools and its menu driven user interface is described in detail in Section 8, "SFS Tools Interface".

The recommended method for running the SFS suite is described below. The benchmark may be run by any other mechanism that preserves the workload, its required parameters, concurrency, correctness and accuracy verification, and throughput measurements performed by the supplied makefiles. This is most easily accomplished using the supplied scripts.

#### *1.8.2.1 Running 097.LADDIS from the $SPEC Directory*

To run the SPEC SFS Release 1.1 suite you can use the shell script "runsfs", at $SPEC directory level. It calls an interactive menu-driven shell script, $SPEC/bin/runsfs, with the proper environment and shell. Using your responses, runsfs calls $SPEC/bin/laddis_mgr. The runsfs script guides you through the execution of the 097.LADDIS benchmark. The script prompts you for the information necessary to run the benchmark. Further, the script allows you to change only those benchmark parameters that are permitted to be modified by the SPEC Run and Reporting Rules for SFS Release 1.1. The runsfs script allows you to:

1. Display, create, or modify a vendor wrapper file.
2. Display, create, or modify a LADDIS parameter file (laddis_rc).
3. Compile the 097.LADDIS benchmark.
4. Remotely install LADDIS on the clients and mount/unmount target file systems.
5. Run the 097.LADDIS benchmark.
6. Display or archive benchmark results.

Runsfs may be aborted at any time. For more information refer to the runsfs script and the vendor wrapper files inside the 097.LADDIS benchmark directory.

#### *1.8.2.2 Running 097.LADDIS Via the laddis_mgr Script*

Once users gain a high degree of familiarity with the 097.LADDIS benchmark, a second method of running the benchmark may be used. The second method is to run the benchmark via the laddis_mgr script. If the results are to be reported, the only parameters that can be changed, as shown in Section 8.5, are LOAD, BIOD_MAX_WRITES, BIOD_MAX_READS, NUM_RUNS, INCR_LOAD, CLIENTS, MNT_POINTS, SUT_FS_NAMES, SUTNAME, PRIME_SLEEP, and PRIME_MON_SCRIPT.

#### *1.8.2.3 Location of Benchmark Results*

When the runsfs or laddis_mgr script is used at the $SPEC level to run the 097.LADDIS benchmark, then benchmark results are located in $RESULTSDIR.

### *1.9 How to Handle and Report Problems*

If you run the benchmark from the top level directory using the runsfs script, all the messages issued are saved in the $SPEC/output directory under the name of laddislog.id where <id> is a run identifier that you are asked to specify. This is the first place to check for possible causes for errors.

In addition, please refer to Section 11, "General Debug Information", for a list of common pitfalls and how to circumvent those.

If you run into errors, please contact SPEC using the phone number provided at the end of this document. You will be referred to a SPEC technical representative who will assist you in resolving the problem(s).

### *1.10 Recommendations for Reproducibility of Results*

When attempting to reproduce the results reported by a specific results submitter, please ensure that your configurations, parameters, and makefiles are set exactly as specified in the submitter's reported results. These details may be found in the the SPEC newsletter or requested from the submitter.

The "SPEC SFS Release 1.1 Reporting Rules" paper describes the number of measurement points to be made and other significant information.