CHAPTER 4

SPECsfs97 Run and Disclosure Rules

1.0 Introduction

This document provides the rules to follow for all submitted, reported, published and publicly disclosed runs of the SPEC System File Server (SFS) 2.0 Benchmark according to the norms specified and approved by the SPEC SFS sub Steering Committee (SFSSC). These run rules also form the basis for determining which server hardware and software features are allowed for benchmark execution and result publication.

This document should be considered the complete guide when addressing the issues of benchmark and NFS server configuration requirements for the correct execution of the benchmark. The only other documents that should be considered are potential clarifications or interpretations of these Run and Disclosure Rules. These potential interpretations should only be accepted if they originate from and are approved by the SFSSC.

These run and disclosure rules are meant to provide the standard by which customers can compare and contrast NFS server performance. It is the intent of the SFSSC to set a reasonable standard for benchmark execution and disclosure of results so customers are presented with enough information about the disclosed configuration to potentially reproduce configurations and their corresponding results.

As a requirement of the license of the benchmark, these run and disclosure rules must be followed. If the user of the SFS 2.0 benchmark suite does not adhere to the rules set forth herein, SPEC may choose to terminate the license with the user. Please refer to the SPEC SFS 2.0 Benchmark license for complete details of the user's responsibilities.

For this document, it is assumed the reader is familiar with the SFS 2.0 benchmark through the use of SFS 1.1 and/or the reading of the user documentation for SFS 2.0.

2.0 Definitions

- *Benchmark* refers to the SPEC SFS 2.0 release of the source code and corresponding work loads defined for the measurement of NFS version 2 and NFS version 3 servers.
- *Disclosure or Disclosing* refers to the act of distributing results obtained by the execution of the *benchmark* and its corresponding work loads. This includes but is not limited to the disclosure to SPEC for inclusion in its electronic medium or paper newsletter or the electronic or paper publication by other organizations or individuals. This does not include the disclosure of results between the user of the benchmark and a second party where there exists a confidential disclosure agreement between the two parties relating to the benchmark results.
- *Publication* refers to the use by SPEC for inclusion in its electronic medium or paper newsletter or other SPEC printed content.

3.0 Overview of SPEC SFS Release 2.0 Run Rules

The general philosophy behind this set of rules for benchmark execution is to ensure that benchmark results can be reproduced if desired.

1. All data published must be gathered from benchmark execution conducted according to the SFS Release 2.0 Run and Disclosure Rules.
2. Benchmark execution must complete in its entirety and normally without benchmark failure or benchmark error messages.
3. The complete hardware, software, and network configuration used for the benchmark execution must be published. This includes any special server hardware, client hardware or software features.
4. Use of software features which invoke, generate or use software designed specifically for the benchmark is not allowed. Configuration options chosen for benchmark execution should be options that would be generally recommended for the customer.
5. The system, including all hardware, software, and network components must be available for general customer shipment within **six months** of the date of benchmark result publication. If the system tested was not generally available on date tested, the generally available system's performance must meet or exceed the system tested for the initially reported performance. If generally available system does not meet the reported performance, the result publisher must publish the lower performing results. Lower results are acceptable if the margin of error for throughput is less than one percent and the margin of error for response time is less than five percent or 1 millisecond which ever is greater.

4.0 Benchmark Software Requirements

4.1 Server and Client Software

In addition to the base operating system, the server will need either the NFS Version 2 or NFS Version 3 software. The clients used for testing will need an ANSI-conformant C compiler (if benchmark compilation is required), a bourne shell, a remote shell, a copy of the benchmark and a network interface.

All of the server software components are required to be generally available within six months of result publication. Use of benchmark specific software components on either the clients or server are not allowed.

4.2 Vendor Makefile Wrappers

Included in this benchmark release are pre-compiled versions of the benchmark for various operating systems at various levels. If it becomes necessary for the user to compile a version of the benchmark source for testing, generic makefiles are provided in the benchmark source directories.

Typically a vendor makefile wrapper (M.vendor) is used in conjunction with the generic makefile for benchmark compilation. The makefiles may be modified or supplemented in a performance neutral fashion to facilitate the compilation and execution of the benchmark on operating systems not included within the benchmark distribution.

It should be noted that as of SFS 2.0, the client no longer needs NFS client software present or configured for successful execution of the benchmark.

The following is a list of the vendors and their respective operating system levels for which the benchmark has been pre-compiled and included with the benchmark distribution.

- Digital Equipment Corporation

Digital UNIX 3.0 and later

- Hewlet-Packard Company
  HP-UX 10.0.1 and later
- IBM Corporation
  AIX version 4.1 and later
- Silicon Graphics
  IRIX (?)
- Sun Microsystems, Inc.
  Solaris 2.4 and later

4.3 Benchmark Source Code Changes

SPEC permits minimal performance-neutral portability changes of the benchmark source. When benchmark source changes are made, an enumeration of the modifications and the specific source changes must be submitted to SPEC prior to result publication. All modifications must be reviewed and deemed *performance neutral* by the SFSSC. Results requiring such modifications can not be published until such time that the SFSSC accepts the modifications as performance neutral.

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

Portability changes will generally be allowed if, without the modification, the:

1. Benchmark source will not compile,
2. Benchmark does not execute, or,
3. Benchmark produces results which are marked INVALID

5.0 Protocol and Server Configuration and Network Requirements

For a benchmark result to be eligible for disclosure, all items identified in the following sections must be true.

5.1 NFS protocol requirements

1. For NFS Version 2, the server adheres to the protocol specification and in particular the requirement that for NFS write operations the NFS server must not reply to the NFS client before any modified file system data or metadata are written to stable storage.
2. For NFS Version 3, the server adheres to the protocol specification. In particular the requirement that for *STABLE* write requests and *COMMIT* operations the NFS server must not reply to the NFS client before any modified file system data or metadata are written to stable storage for that specific or related operation. See RFC 1813, NFSv3 protocol specification for a definition of STABLE and COMMIT for NFS write requests.
3. For NFS Version 3, operations which are specified to return wcc data must, in all cases, return TRUE and the correct attribute data. Those operations are:
   a. SETATTR
   b. READLINK

c. CREATE
d. MKDIR
e. SYMLINK
f. MKNOD
g. REMOVE
h. RMDIR
i. RENAME
j. LINK

4. The server must pass the benchmark validation for the NFS protocol being tested.
5. When UDP is the network transport, UDP checksums must be calculated and verified for all NFS request and reply messages. In other words, checksums must be enabled on both the client and server.

5.2 Server configuration requirements

1. The server does not use any type of RAM disk or other type of file system which does not survive server failure and reboot.
2. The server configuration follows the uniform access rules for the clients' access to the server file systems.

5.3 SPEC's Description of Stable Storage for SFS 2.0

In section "NFS protocol requirements" on page 49, the term *stable storage* is used. For clarification, the following references and further definition is provided and must be followed for results to be disclosed.

5.3.1 Protocol definition of stable storage and its use

RFC 1094, NFS: Network File System, of March 1989, page 3 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.

5.3.2 Stable storage further defined

SPEC has further clarification of this definition to resolve any potential ambiguity. For the purposes of the 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). At any point where the data can be cached, after response to the client, there must be a

mechanism to ensure the cached data survives server failure.

5.3.3 Examples of stable storage

1. Media commit of data, i.e., the modified data has been successfully written to the disk media, for example, the disk platter.
2. An immediate reply disk drive with battery-backed on-drive intermediate storage or uninterruptible power system. (UPS)
3. Server commit of data with battery-backed intermediate storage and recovery software.
4. Cache commit with uninterruptible power system (UPS) and recovery software.

5.3.4 Examples which are not considered stable storage

1. An immediate reply disk drive without battery-backed on-drive intermediate storage or uninterruptible power system. (UPS)
2. Cache commit without both uninterruptible power system (UPS) and recovery software.
3. Server commit of data without battery-backed intermediate storage & memory.

5.4 SPEC's Description of Uniform Access for SFS 2.0

In "Server configuration requirements" on page 50 the term *uniform access* is used to define a requirement. This section provides a complete description and examples. The NFS server configuration for the benchmark execution should provide uniform file system access to the clients being used.

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

Uniform access is meant to eliminate potential exploitation of any partionable aspect of the benchmark, particularly when reporting cluster results. It is recognized that servers vary as to exposing elements such as processor, disk controller or disk to load generators remotely accessing file systems. The algorithm presented below is the preferred mechanism when determining file system access for benchmark configuration. This method should prevent biased configurations for benchmark execution.

5.4.1 Uniform access algorithm

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 (1), then (2), then (3), and so on:

1. next network,
2. next cluster processor (if clustered system),
3. other controllers in the path from the network, to the file system,
4. file system.

Note that this list may not be complete for system components which should be considered for uniform access. Some server architectures may have other major components. In general, components should be included so all data paths are included within the system.

5.4.2 Examples of uniform access

1. n-level symmetric multiprocessors (include uniprocessor, i.e. n=1).
   k. Select next load-generating process for a client.
   l. Select next network accessed by that client.
   m. Select next network controller on the network.
   n. Select next disk controller
   o. Select next file system.
2. 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.
3. 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.

5.5 Network configuration requirements

The network(s) used for valid benchmark execution must be isolated networks. Results obtained on production networks are invalid as they will most likely not be reproducible. Furthermore, the benchmark may fail to correctly converge to the requested load rate and behave erratically due to varying ambient load on the network.

6.0 Benchmark Execution Requirements

This section details the requirements governing how the benchmark is to be executed for the purpose of generating results for disclosure.

6.1 Server File System Creation and Configuration

As stated in section 5.3, ”SPEC’s Description of Stable Storage for SFS 2.0”, on page 50, the NFS server’s target file systems, their configuration and underlying physical medium used for benchmark execution must follow the stable storage requirements.

At the start of each benchmark run, before the first in a series of requested NFS load levels is generated, the NFS server’s target filesystems 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 target filesystem. For non-UNIX-based systems, a semantic equivalent to the **mkfs** or **newfs** command must be used.

6.2 Data Point Specification for Results Disclosure

The result of benchmark execution is a set of NFS throughput / response time data points for the server under test which defines a performance curve. The measurement of all data points used to define this performance curve must be made within a single benchmark run, starting with the lowest requested NFS load level and proceeding to the highest requested NFS load level.

Published benchmark results must include at least 10 data points uniformly distributed between zero and the maximum achieved throughput (excluding zero ops/sec and including the maximum measured throughput), except as noted below. For example in a 10-point run the first uniformly spaced data points would be at 10%, 20%, and 30% of the maximum throughput. In a 25-point run the first uniformly spaced data points would be at 4%, 8%, and 12% of the maximum throughput.

Additional data points may also be included within the above range: some of these data points may be omitted in the disclosure. However, the omitted data points must correspond to the right-most or highest requested throughput. Data points may not be omitted from the beginning or middle of the requested data points.

Any invalid data points will invalidate the entire run unless they are at or below 25% of the maximum measured throughput. All data points at or below the maximum reported throughput must be reported. Invalid data points must be submitted but will not appear on the disclosure page graph. (The requested load associated with the invalid points will appear on the disclosure reporting table, however, the throughput and response time will be omitted.)

No server or testbed configuration changes, server reboots, or file system initialization (e.g., “newfs”) are allowed during the execution of the benchmark or between data point collection.

If any requested NFS load level or data point must be rerun for any reason, the entire benchmark execution must be restarted, i.e., the server’s filesystems must be initialized and the series of requested NFS load levels repeated in whole.

6.3 Maximum response time for Results Disclosure

For each data point measured, there will be the throughput and corresponding response time. For a data point to be eligible for results disclosure the response time reported by the benchmark must not exceed 40 milliseconds.

6.4 Over all response time calculation

The overall response time is an indicator of how quickly the system under test responds to NFS operations over the entire range of the tested load. The overall response time is a measure of how the system will respond under an average load. Mathematically, the value is derived by calculating the area under the curve divided by the peak throughput. Below the first valid data point is assumed to have a constant response time equal to that of the first data point.

6.5 Benchmark Modifiable Parameters

The benchmark has a number of parameters which are configurable. This parameter modification is specified with the use of the **RC** file on the prime client. For benchmark execution for results to be disclosed, there is a subset of parameters which may be modified. Parameters outside of the set specified below may not be modified for a publishable benchmark result.

Parameters which may be modified for benchmark execution:

6.5.1 LOAD

Used to specify the data points to be collected by the benchmark. List must increase in value and must represent a uniform distribution.

6.5.2 INCR_LOAD

If the **LOAD** has a single value, this parameter is used to specify the increment to increase the load for successive data points.

6.5.3 NUM_RUNS

If **INCR_LOAD** is used, this parameter is used to specify the number of data points to gather. For a valid benchmark execution, this value must be greater than or equal to 10.

6.5.4 PROCS

This parameter specifies the number of load generating processes to be used on each load generating client. There is a minimum number of eight processes for each network used in the benchmark configuration. For example, if the server being measured has two network interfaces and there are two clients on each network, then each client would require a minimum of four processes to be used and this parameter would have a value of 4.

6.5.5 CLIENTS

CLIENTS is used to specify the host names of the clients used for generating the NFS load points.

6.5.6 MNT_POINTS

List of file systems to be used for the benchmark execution. This list should be generated to comply to the uniform access requirements defined in "SPEC's Description of Uniform Access for SFS 2.0" on page 51.

6.5.7 BIOD_MAX_WRITES

Specifies the number of outstanding or async writes that the benchmark will generate per benchmark process. The minimum number is two and there is no maximum number.

6.5.8 BIOD_MAX_READS

Specifies the number of outstanding or async reads that the benchmark will generate per benchmark process. The minimum number is two and there is no maximum number.

6.5.9 TCP

Specifies if TCP should be used as the transport mechanism to contact the NFS server for all generated transactions. Default is to use UDP, if this option is set to "on" then TCP will be used.

6.5.10 NFS_VERSION

Specifies the version of the NFS protocol to use for benchmark execution. The default is version 2 and if "3" is specified, NFS version 3 will be used for the benchmark execution.

6.5.11 SFS_USER

The user account name which is configured on all clients to be used for the benchmark execution. Each client should be configured to allow this user execution of the benchmark.

6.5.12 SFS_DIR

Path name which specifies the location of the benchmark executables. Each client should be configured to use the same path.

6.5.13 WORK_DIR

Path name where all benchmark results are placed. Each client should be configured to have this path available.

6.5.14 PRIME_MON_SCRIPT

Name of a shell script or other executable program which will be invoked to control any external programs. These external programs must be performance neutral. If this option is used, the executable used must be disclosed.

6.5.15 PRIME_MON_ARGS

Arguments which are passed to the executable specified in **PRIME_MON_SCRIPT**.

6.5.16 RSH

The default for this option is the *rsh* command. For those operating environments which do not use *rsh* for remote execution, this option should be set to the appropriate remote execution program. This value applies to the prime client.

6.6 Valid methods for benchmark execution

There are two mechanisms which can be used for obtaining valid benchmark executions.

The first is the use of the ***sfs_mgr*** script. For those familiar with the benchmark, this shell script can be used in combination with an **RC** file for benchmark execution.

The second is to use the ***runsfs*** script. This script is a menu based utility that will provide a helping hand to the user that is somewhat unfamiliar with the benchmark and its execution.

7.0 Results Disclosure

Since it is the intent of these run and disclosure rules to provide the standard by which customers can compare and contrast NFS server performance, it is important to provide all the pertinent information about the system tested so this intent can be met. The following describes what is required for disclosure of benchmark results. It is recognized that all of the following information can not be provided with each reference to benchmark results. Because of this, there is a minimum amount of information that must be always be present and upon request, the party responsible for disclosing the benchmark results must provide a *full* disclosure of the benchmark configuration. Note that SPEC publication requires a full disclosure.

7.1 Benchmark metric or minimum disclosure

The following are the minimum allowable disclosure of benchmark results

1. "XXX SPECsfs97.v2 ops per second with an overall response time of YYY ms"
2. "XXX SPECsfs97.v3 ops per second with an overall response time of YYY ms"

The XXX would be replaced with the throughput value obtain from the right most data point of the throughput / response time curve generated by the benchmark. The YYY would be replaced with the overall response time value as generated by the benchmark.

7.2 Full disclosure of benchmark results

The information described in the following sections should be sufficient for reproduction of the disclosed benchmark results. If additional information is needed, the party disclosing the results should provide the information as a note or additional disclosure. All product names and model numbers and configurations should be complete such that the information provided could be used to order the disclosed products.

7.2.1 Server hardware configuration

7.2.1.1 Server CPU configuration

1. Model Number
2. CPU (Name and Mhz or other identification)
3. Number of CPUs
4. Primary CPU Cache
5. Secondary CPU Cache
6. Other Cache
7. Memory

7.2.1.2 Server stable storage configuration

1. Number and type of disk controllers
2. Number and type of disks
3. Special disk or NVRAM products and brief description of their functionality

7.2.1.3 Server network configuration

1. Number and type of network controllers
2. Number of networks (potentially different if switch network involved or if network controller has more than one physical connection)

7.2.1.4 Other server hardware configuration

1. UPS
2. Other special hardware employed for tested configuration

7.2.2 Server software configuration

1. Operating system
2. Other software (i.e. device drivers, NFS products or software patches)
3. Buffer Cache size
4. Number of NFS daemons
5. Number of file systems
6. File system configuration and corresponding physical disks
7. Options used for file system creation/initialization

7.2.3 Client hardware configuration

1. Vendor name
2. Model number
3. Number and type of CPUs
4. Memory size
5. Type of network controller

7.2.4 Client software configuration

1. Operating system used
2. Compiler and version
3. Any non-standard compiler options

7.2.5 Network hardware configuration

These apply for configuration which use network components to build test configuration.

1. Switches and model numbers and option configurations
2. Bridges and model numbers
3. Hubs and model numbers

7.2.6 Benchmark configuration

1. File set size
2. Number of clients
3. Processes per client
4. biod_max_read parameter setting
5. biod_max_write parameter setting
6. Configuration of file systems as they are used by the clients

7. UDP or TCP transport selection

7.2.7 Benchmark results

1. Throughput number and average response time for each data point used
2. Overall response time metric generated by the benchmark

7.2.8 Miscellaneous information

1. Benchmark license number
2. Licensee name who generate results
3. Location of licensee
4. Date tested
5. Date of hardware availability
6. Date of software availability