SPECsfs2008

Run and Reporting Rules

Standard Performance Evaluation Corporation (SPEC)
6585 Merchant Place, Suite 100
Warrenton, VA 20187, USA
Phone: 540-349-7878
Fax: 540-349-5992
E-Mail: info@spec.org
www.spec.org

Copyright (c) 2008 by Standard Performance Evaluation Corporation (SPEC)
All rights reserved

SPEC and SFS are registered trademarks of the Standard Performance Evaluation Corporation

NFS is a registered trademark of Sun Microsystems, Inc.

Table of Contents

1 Overview

This document specifies the guidelines on how SPECsfs2008 is to be run for measuring and publicly reporting performance results. These rules have been established by the SPEC SFS subcommittee and approved by the SPEC Open Systems Steering Committee. They ensure that results generated with this suite are meaningful, comparable to other generated results, and are repeatable (with documentation covering factors pertinent to duplicating the results).

This document provides the rules to follow for all submitted, reported, published and publicly disclosed runs of the SPEC System File Server (SPECsfs2008) Benchmark according to the norms specified and approved by the SPEC SFS subcommittee. 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 file 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 Reporting Rules. These potential interpretations should only be accepted if they originate from and are approved by the SFS subcommittee.

These Run and Reporting Rules are meant to provide the standard by which customers can compare and contrast file server performance. It is the intent of the SFS subcommittee 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 Reporting Rules must be followed. If the user of the SPECsfs2008 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 SPECsfs2008 Benchmark license for complete details of the user's responsibilities.

Per the SPEC license agreement, all results publicly disclosed must adhere to these Run and Reporting Rules.

The general philosophy behind the 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 Run and Reporting Rules described in this chapter.
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 entire SUT, including disks, must be comprised of components that are generally available, or shall be generally available within three months of the first publication of the results. If the system was not generally available on the date tested, the generally available system's performance must meet or exceed that of the system tested for the initially reported performance. If the generally available system does not meet the reported performance, the lower performing results shall be published. Lower results are acceptable if the margin of error for peak throughput is less than one percent (1%) and the margin of error for overall response time is less than five percent (5%) or one millisecond (1 ms), whichever is greater.

Products are considered generally available if they can be ordered by ordinary customers and ship within a reasonable time frame. This time frame is a function of the product size and classification, and common practice. The availability of support and documentation for the products must coincide with the release of the products.

Hardware products that are still supported by their original or primary vendor may be used if their original general availability date was within the last five years. The five-year limit does not apply to the hardware used in client systems - i.e., client systems are simply required to have been generally available at some time in the past.

Software products that are still supported by their original or primary vendor may be used if their original general availability date was within the last three years.

In the disclosure, the submitting vendor must identify any SUT component that can no longer be ordered by ordinary customers.

1.1 Definitions

- *Benchmark* refers to the SPECsfs2008 release of the source code and corresponding work loads defined for the measurement of CIFS 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 on the SPEC web site or in 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 on the SPEC web site or any other SPEC printed content.

1.2 Philosophy

SPEC believes the user community will benefit from an objective series of tests, which can serve as common reference and be considered as part of an evaluation process. SPEC is aware of the importance of optimizations in producing the best system performance. SPEC is also aware that it is sometimes hard to draw an exact line between legitimate optimizations that happen to benefit SPEC benchmarks and optimizations that specifically target the SPEC benchmarks. However, with the list below, SPEC wants to increase awareness of implementers and end users to issues of unwanted benchmark-specific optimizations that would be incompatible with SPEC's goal of fair benchmarking.

SPEC expects that any public use of results from this benchmark suite shall be for Systems Under Test (SUTs) and configurations that are appropriate for public consumption and comparison. Thus, it is also required that:

- Hardware and software used to run this benchmark must provide a suitable environment for supporting the *specific application area addressed by this benchmark using the common accepted standards that help define this application space.*
- Optimizations utilized must improve performance for a larger class of workloads than just the ones defined by this benchmark suite. There must be no benchmark specific optimizations.
- The SUT and configuration is generally available, documented, supported, and encouraged by the providers.

To ensure that results are relevant to end-users, SPEC expects that the hardware and software implementations used for running the SPEC benchmarks adhere to following conventions:

- Proper use of the SPEC benchmark tools as provided.
- Availability of an appropriate full disclosure report.
- Support for all of the appropriate protocols.

1.3 Caveats

SPEC reserves the right to investigate any case where it appears that these guidelines and the associated benchmark run and reporting rules have not been followed for a published SPEC benchmark result. SPEC may request that the result be withdrawn from the public forum in which it appears and that the benchmarker correct any deficiency in product or process before submitting or publishing future results. SPEC reserves the right to adapt the benchmark codes, workloads, and rules of SPECsfs2008 as deemed necessary to preserve the goal of fair benchmarking. SPEC will notify members and licensees if changes are made to the benchmark and will rename the metrics (e.g. from *SPECsfs97_R1* to *SPECsfs2008_nfs.v3 and SPECsfs2008_cifs*).

Relevant standards are cited in these run rules as URL references, and are current as of the date of publication. Changes or updates to these referenced documents or URL's may necessitate repairs to the links and/or amendment of the run rules. The most current run rules will be available at the SPEC web site at http://www.spec.org. SPEC will notify members and licensees whenever it makes changes to the suite.

2 Results Disclosure and Usage

SPEC encourages the submission of results for review by the relevant subcommittee and subsequent publication on SPEC's web site. Vendors may publish compliant results independently, however any SPEC member may request a full disclosure report for that result and the benchmarker must comply within 10 business days. Issues raised concerning a result's compliance to the run and reporting rules will be taken up by the relevant subcommittee regardless of whether or not the result was formally submitted to SPEC.

The SPECsfs2008 result produced in compliance with these run and reporting rules may be publicly disclosed and represented as a valid SPECsfs2008 result. All SPECsfs2008 results that are submitted to SPEC will be reviewed by the SFS subcommittee. The review process ensures that the result is compliant with the run and reporting rules set forth in this document. If the result is compliant then the result will be published on the SPEC web site. If the result is found to be non-compliant then the submitter will be contacted and informed of the specific problem that resulted in the non-compliant component of the submission.

Any test result not in full compliance with the run and reporting rules must not be represented using the SPECsfs2008_nfs.v3 or SPECsfs2008_cifs metric names.

The metrics SPECsfs2008_nfs.v3 and SPECsfs2008_cifs *must not be associated with any estimated results.* This includes adding, multiplying or dividing measured results to create a derived metric.

2.1 Fair Use of SPECsfs2008 Results

Consistency and fairness are guiding principles for SPEC. To assure these principles are sustained, guidelines have been created with the intent that they serve as specific guidance for any organization (or individual) that chooses to make public comparisons using SPEC benchmark results. These guidelines are published at: http://www.spec.org/osg/fair_use-policy.html.

2.2 Research and Academic usage of SPECsfs2008

SPEC encourages use of the SPECsfs2008 benchmark in academic and research environments. It is understood that experiments in such environments may be conducted in a less formal fashion than that required of licensees submitting to the SPEC web site or otherwise disclosing valid SPECsfs2008 results. For example, a research environment may use early prototype hardware that simply cannot be expected to stay up for the length of time required to run the required number of points, or may use research software that are unsupported and are not generally available. Nevertheless, SPEC encourages researchers to obey as many of the run rules as practical, even for informal research. SPEC suggests that following the rules will improve the clarity, reproducibility, and comparability of research results. Where the rules cannot be followed, SPEC requires the results be clearly distinguished from full compliant results such as those officially submitted to SPEC, by disclosing the deviations from the rules and avoiding the use of the SPECsfs2008_nfs.v3 and SPECsfs2008_cifs metric names.

2.3 SPECsfs2008 metrics

The following format must be used when referencing SPECsfs2008 benchmark results:

1. "XXX SPECsfs2008_cifs ops per second with an overall response time of YYY ms"
2. "XXX SPECsfs2008_nfs.v3 ops per second with an overall response time of YYY ms"

The XXX would be replaced with the throughput value obtained 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 reporting tools. Only the NFS or the CIFS metric, not both, need to be disclosed.

A result is only valid for the SPECsfs2008 metric that is stated. One can not compare results of different SPECsfs2008 metrics. The workloads are not comparable across different metrics.

2.4 Full disclosure of benchmark configuration and results

Since it is the intent of these Run and Reporting Rules to provide the standard by which customers can compare and contrast file 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 full 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 always be present (i.e., the SPECsfs2008 metrics as specified in the previous section) 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.

Appendix A defines the fields of a full disclosure. It should be sufficient for reproduction of the disclosed benchmark results.

2.5 Disclosure of Results for Electronically Equivalent Systems

The SPEC SFS subcommittee encourages result submitters to run the benchmark on all systems. However, there may be cases where a vendor may choose to submit the same results for multiple submissions, even though the benchmark run was performed on only one of the systems. This is acceptable if the performance reported is representative of those systems (e.g., just the power supply or chassis is different between the systems). These systems are deemed to be "electronically equivalent". A definition of this term which can be applied during SPEC SFS submission reviews is provided below.

As part of the subcommittee review process, the submitter should expect to be asked to justify why the systems should have the same performance. It may be appropriate for the subcommittee to ask for a rerun on the exact system in situations where the technical criteria are not satisfied. In cases where the subcommittee accepts the submitter's claim of electronic equivalence, the submitter must include a line in the Other Notes section of each of the submissions for systems on which the benchmark was NOT run. For example, if a submitter submits the same results for Model A and Model B, and the benchmark run was performed on Model A, the Model B submission should include a note like the following:

"The benchmark run was performed on a Vendor's Model A system. Vendor's Model A and Vendor's Model B systems are electronically equivalent."

2.5.1 Definition of Electronic Equivalence

For the purpose of SPECsfs2008 benchmarking, the basic characteristic of electronically equivalent systems is that there are no noticeable differences in the behavior of the systems under the same environmental conditions specifically in terms of SPECsfs2008 performance, down to the level of electronic signals.

Examples of when systems are considered to be electronically equivalent include:

- ✓ Packaging - for example a system that is sold as both a desk side system and rack mount system (where the only difference is the casing) would be considered electronically equivalent. Another example is systems that are sold in a large case (to allow installation of disks internally) and a small case (which requires an external case for disks) but which are otherwise identical.
- ✓ Naming - for example a system where the vendor has changed the name and/or model number and face plate without changing the internal hardware is considered electronically equivalent.

Examples of when systems are not considered electronically equivalent include:

- ✓ Different number or types of slots or buses - even if unused, hardware differences such as these may change the behavior of the system at peak performance. These systems are usually referred to as 'functionally equivalent'.
- ✓ Vendor fails to convince the committee on technical merits that the systems are electronically equivalent.

3 Benchmark Software Requirements

3.1 Server and Client Software

In addition to the base operating system, the server will need either the CIFS or NFS Version 3 software. Use of benchmark specific software components on either the clients or server are not allowed.

3.2 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 SFS subcommittee. Results requiring such modifications can not be published until such time that the SFS subcommittee 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

4 Server Configuration, Load Generator Configuration, and Protocol Requirements

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

4.1 NFS protocol requirements

1. 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, with the exception of access times, 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.
2. 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:

NFS Version 3
SETATTR
CREATE
MKDIR

SYMLINK
REMOVE
RMDIR
RENAME
LINK

3. The server must pass the benchmark validation for the NFS workload.
4. The use of UDP as a transport for NFS testing is not permitted.

4.2 CIFS protocol requirements

1. The server adheres to the CIFS protocol as defined in the most recent version of the SNIA CIFS Technical Reference.
2. The server must pass the benchmark validation for the CIFS protocol.
3. The server should not respond to a FLUSH SMB request until the data and file allocation information is written to stable storage. See the SNIA CIFS Technical Reference for a description of the FLUSH SMB.
4. For CIFS protocol file query operations which require an information level to be specified, the server must be capable of returning complete and correct data at the SMB_QUERY_FILE_BASIC (0x101) and SMB_QUERY_FILE_STANDARD (0x102) levels.
5. Servers must advertise the following CIFS capabilities when negotiating connection to the server:
   - CAP_UNICODE (0x0004) – support for UNICODE strings
   - CAP_LARGE_FILES (0x0008) – support for large files with 64-bit offsets

4.3 Server configuration requirements

1. The server may 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 must follow the uniform access rules for the clients' access to the server file systems.
3. The server may not be used as a load generator.
4. The benchmark may use UDP only for communicating with the portmapper. The UDP protocol must be available on the SUT for the benchmark to properly initialize.

4.4 Load Generator configuration requirements

1. All the load generators must be running the same operating system.
2. The server may not be used as a load generator.
3. To be used as a load generator, a system must support a clock resolution of 10 microseconds or better.

4.5 Description of Stable Storage for SPECsfs2008

In the sections "NFS protocol requirements" and "CIFS protocol requirements" above, 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.

4.5.1 NFS protocol definition of stable storage and its use

>From Page 52 in RFC 1813:

> "The definition of stable storage has been historically a point of contention. The following expected properties of stable storage may help in resolving design issues in the implementation. Stable storage is persistent storage that survives:
>
> 1. Repeated power failures.
>
> 2. Hardware failures (of any board, power supply, and so on.).
>
> 3. Repeated software crashes, including reboot cycle.
>
> This definition does not address failure of the stable storage module itself."

>From Pages 101-102 in RFC 1813:

> "4.8 Stable storage
>
> NFS version 3 protocol servers must be able to recover without data loss from multiple power failures including cascading power failures, that is, several power failures in quick succession, operating system

failures, and hardware failure of components other than the storage medium itself (for example, disk, nonvolatile RAM).

Some examples of stable storage that are allowable for an NFS server include:

1. Media commit of data, that is, 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.

Conversely, the following are not examples of 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.

The only exception to this (introduced in this protocol revision) is as described under the WRITE procedure on the handling of the stable bit, and the use of the COMMIT procedure. It is the use of the synchronous COMMIT procedure that provides the necessary semantic support in the NFS version 3 protocol."

4.5.2 CIFS protocol definition of stable storage and its use

The SNIA CIFS Technical Reference specifies when data can be cached in non-stable storage:

> 1.1.3. Safe caching, read-ahead, and write-behind
> The protocol supports caching, read-ahead, and write-behind, even for unlocked files, as long as they are safe. All these optimizations are safe as long as only one client is accessing a file; read caching and read-ahead are safe with many clients accessing a file as long as all are just reading. If many clients are writing a file simultaneously, then none are safe, and all file operations have to go to the server. The protocol notifies all clients accessing a file of changes in the number and access mode of clients accessing the file, so that they can use the most optimized safe access method.

INTERPRETATION: In the SPECsfs2008_cifs workload, files aren't shared by different clients, so clients can cache safely. SPECsfs2008_cifs puts SMB packets directly on the wire, effectively bypassing any client caching. The server can safely cache data written by the SPECsfs2008_cifs benchmark.

Write data can be cached in non-stable storage:

> 4.2.5. WRITE_ANDX: Write Bytes to file or resource
> Client requests a file write, using the SMB fields specified below:
> Client Request Description

=============== ============
UCHAR WordCount; Count of parameter words = 12 or 14
UCHAR AndXCommand; Secondary (X) command; 0xFF = none
UCHAR AndXReserved; Reserved (must be 0)
USHORT AndXOffset; Offset to next command WordCount
USHORT Fid; File handle
ULONG Offset; Offset in file to begin write
ULONG Reserved; Must be 0
USHORT WriteMode; Write mode bits:
0 - write through
USHORT Remaining; Bytes remaining to satisfy request
USHORT DataLengthHigh; High 16 bits of data length if
CAP_LARGE_WRITEX; else MUST BE ZERO
USHORT DataLength; Number of data bytes in buffer (>=0)
USHORT DataOffset; Offset to data bytes
ULONG OffsetHigh; Upper 32 bits of offset (only present if
WordCount = 14)
USHORT ByteCount; Count of data bytes; ignored if
CAP_LARGE_WRITEX
UCHAR Pad[]; Pad to SHORT or LONG
UCHAR Data[DataLength]; Data to write

And, the server response is:
Server Response Description
=============== ============
UCHAR WordCount; Count of parameter words = 6
UCHAR AndXCommand; Secondary (X) command; 0xFF = none
UCHAR AndXReserved; Reserved (must be 0)
USHORT AndXOffset; Offset to next command WordCount
USHORT Count; Number of bytes written
USHORT Remaining; Reserved
ULONG Reserved;
USHORT ByteCount; Count of data bytes = 0

If the file specified by Fid has any portion of the range specified by Offset and MaxCount locked for shared or exclusive use by a client with a different connection or Pid, the request will fail with ERRlock.

A ByteCount of 0 does not truncate the file. Rather a zero length write merely transfers zero bytes of information to the file. A request such as SMB_COM_WRITE must be used to truncate the file.

If WriteMode has bit0 set in the request and Fid refers to a disk file, the response is not sent from the server until the data is on stable storage.

If the negotiated dialect is NT LM 0.12 or later, the 14 word format of this SMB may be used to access portions of files requiring offsets expressed as 64 bits. Otherwise, the OffsetHigh field must be omitted from the request.

If CAP_LARGE_WRITEX was indicated by the server in the negotiate protocol response, the request's DataLength field may exceed the negotiated buffer size if Fid refers to a disk file.
The following are the valid AndXCommand values for this SMB:
SMB_COM_READ SMB_COM_READ_ANDX
SMB_COM_LOCK_AND_READ SMB_COM_WRITE_ANDX

SMB_COM_CLOSE

4.2.5.1. Errors
ERRDOS/ERRnoaccess
ERRDOS/ERRbadfid
ERRDOS/ERRlock
ERRDOS/ERRbadaccess
ERRSRV/ERRinvid
ERRSRV/ERRbaduid

INTERPRETATION: If the WriteMode has bit0 set, the response cannot be returned until the server has put the data on stable storage. SPECsfs2008_cifs does not set bit0 on the WRITE_ANDX call, so the server can cache the write data in non-stable storage.

The FLUSH SMB specifies that file data must be on stable storage before generating a response:

4.2.8. FLUSH: Flush File
The flush SMB is sent to ensure all data and allocation information for the corresponding file has been written to stable storage. When the Fid has a value -1 (hex FFFF), the server performs a flush for all file handles associated with the client and Pid. The response is not sent until the writes are complete.

Client Request Description
========== ==========
UCHAR WordCount; Count of parameter words = 1
USHORT Fid; File handle
USHORT ByteCount; Count of data bytes = 0

This client request is probably expensive to perform at the server, since the server's operating system is generally scheduling disk writes is a way which is optimal for the system's read and write activity integrated over the entire population of clients. This message from a client "interferes" with the server's ability to optimally schedule the disk activity; clients are discouraged from overuse of this SMB request.

Server Response Description
========== ==========
UCHAR WordCount; Count of parameter words = 0
USHORT ByteCount; Count of data bytes = 0

INTERPRETATION: In the SPECsfs2008_cifs benchmark, the flush SMB should pend until the data and file allocation information is written to stable storage. The SPECsfs2008_cifs benchmark does not use a Fid of -1, rather it uses a valid Fid, so only the given file will be flushed.

4.5.3 Definition of terms pertinent to stable storage

In order to help avoid further ambiguity in describing "stable storage", the following terms, which are used in subsequent sections, are defined here:

committed data - as defined in the NFS V3 protocol specification (RFC 1813)

non-volatile intermediate storage - electronic data storage media which requires a power source to ensure retention of the data, and which serves as a staging area for written data whose ultimate destination is auxiliary storage. For the purpose of SPECsfs2008 submissions, NVRAM is non-volatile intermediate storage

auxiliary storage - magnetic (or other) data storage media which can retain data indefinitely without a power source

non-destructive failure - failure which does not directly cause data housed in intermediate or auxiliary storage to be lost or overwritten

transient failure - temporary failure which does not require replacement or upgrade of the failed hardware or software component

system crash - hardware or software failure which causes NFS or CIFS services to no longer be available, at least temporarily, and which requires a reboot of one or more hardware components and/or re-initialization of one or more software components in order for NFS or CIFS services to be restored

SUT (System Under Test) - all of the hardware and software components involved in providing NFS or CIFS services. This excludes the load generators, any network elements between them and the NFS or CIFS service provider, and the external primary power source for the components. It includes the entire data and control path between the NFS or CIFS service provider and the storage media

4.5.4 Stable storage further defined

SPEC has further clarification of the definition of the term "stable storage" to resolve any potential ambiguity. This clarification is necessary since the definition of stable storage has been, and continues to be, a point of contention. Therefore, for the purposes of the SFS benchmark, SPEC defines stable storage in terms of the following operational description:

The SUT must be able to tolerate without loss of committed data:

1. Power failures of the server's primary power source, including cascading power failures, with a total duration of no longer than 72 hours.

2. Non-destructive transient failures of any hardware or software component in the SUT which result in a system crash. Multiple and/or cascading failures are excluded.

3. Manual reset of the entire SUT, or of any of its components involved in providing NFS or CIFS services, if required to recover from transient failures.

If the SUT allows data to be cached in intermediate storage, after a response to the client indicating that the data has been committed, but before the data is flushed to auxiliary storage, then there must be a mechanism to ensure that the cached data survives failures of the types defined above.

There is no intention that committed data must be preserved in the face of unbounded numbers of cascading hardware or software errors that happen to combine to prevent the system from performing any significantly useful work. Many NFS and CIFS servers provide for further protection against some forms of direct damage to the committed data, but such fault-tolerant features are not a prerequisite for SPEC SFS

result publication. Nevertheless, SPECsfs2008 provides a means of characterizing some of these fault-tolerant capabilities of the SUT via the questions listed in the next section.

4.5.5 Specifying fault-tolerance features of the SUT

The following questions can help characterize the SUT in terms of its fault-tolerance capabilities beyond those required for SPECsfs2008 publication. You may consider including answers to these questions in the Other Notes section of the reporting form, however, you are not required to do so.

Can the SUT tolerate without loss of committed data:

1. Destructive hardware failures?
2. Destructive software failures?
3. Multiple concurrent failures of one or more of the above?

4.5.6 SPECsfs2008 submission form fields related to stable storage

The following fields in the SPECsfs2008 result submission form are relevant to an NFS or CIFS server's stable storage implementation, and as such should contain the information described herein:

1. Memory. Specify the size, type, and location of non-volatile intermediate storage used to retain data in the SUT upon loss of primary power. For example: (1) 256 MB battery-backed SDRAM on PCI card in the server, (2) 64 MB battery-backed SRAM in the disk controller, (3) 4 GB on Hard Disk Drive in the Server, (4) 8 GB UPS-Backed Main Memory in the Server, etc.

2. Stable Storage. Describe the stable storage implementation of the SUT. There must be enough detail to explain how data is protected in the event that a power or non-destructive hardware/software failure occurs. The description must at least include the following points where applicable:
    a. Specify the vendor, model number and capacity (VA) of the UPS, if one is used.
    b. Where does committed data reside at the time of a failure?
    c. How does the SUT recover committed data?
    d. What is the life of any UPS and/or batteries used to implement the stable storage strategy?
    e. How is the system protected from cascading power failures?

4.5.7 Stable storage examples

Here are two examples of stable storage disclosure using the above rules. They are hypothetical and are not intentionally based on any current product.

Example #1:

UPS: APC Smart-UPS 1400 (1400VA)

Non-volatile intermediate storage Type: (1) 4 GB on Hard Disk Drive in the Server. (2) 64 MB battery-backed SDRAM in the disk controller.

Non-volatile intermediate storage Description: (a) During normal operation, the server keeps committed data in system memory that is protected by a server UPS. When the UPS indicates a low battery charge, the server copies this data to local SCSI drives. The value of the low battery threshold was chosen to guarantee enough time to flush the data to the local disk several times over. The magnetic media on the disk will hold data indefinitely without any power source. Upon power-up, the server identifies the data on the local drive and retrieves it to resume normal operation. Any hard or soft reset that occurs with power applied to the server will not corrupt committed data in main memory. (b) Committed data is also kept in a DIMM on the disk controller. (c) This DIMM has a 96-hour battery attached to overcome any loss in power. (d) If the disk controller NVRAM battery has less than 72 hours of charge, the disk controller will disable write caching. Reset cycles to the disk controller do not corrupt the data DIMM. Write caching is disabled on all disk drives in the SUT.

Example #2:

UPS: None

Non-volatile intermediate storage Type: 256 MB battery-backed SDRAM on a PCI card.

Non-volatile intermediate storage Description: (a) All data is written to the NVRAM before it is committed to the client and retained until the drive arrays indicate successful transfer to disk. The DIMM on the (c) NVRAM card has a 150-hour battery attached to overcome any loss in power. (b) Upon power-up, the server replays all write commands in the NVRAM before resuming normal operation. (d) The server will flush the data to auxiliary storage and stop serving NFS requests if the charge in the NVRAM battery ever falls below 72 hours. Write caching is disabled on all disk drives in the SUT.

4.6 Description of Uniform Access for SPECsfs2008

This section provides a complete description and examples of the term *uniform access*.

The file 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. Each network must access all of the disk controllers in the SUT to be considered compliant with the uniform access requirement.

Uniform access is meant to eliminate potential exploitation of any partitionable 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, but not the only mechanism, when determining file system access for benchmark configuration. This method should prevent biased configurations for benchmark execution.

4.6.1 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.

4.6.2 Examples of uniform access

1. 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.
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.

4.6.3 Complying with the Uniform Access Rule (UAR)

The most common way to perform a run that will not be accepted by the SPEC SFS subcommittee for publication is to violate the uniform access rule. In some systems, it is possible to complete an NFS, or CIFS, operation especially fast if the request is made through one network interface and the data is stored on just the right file system. The intent of the rule is to prevent the benchmarker (that's you) from taking advantage of these fast paths to get an artificially inflated result.

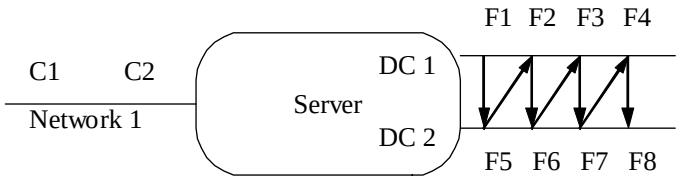
The specific wording of the rule states that “for every network, all file systems should be accessed by all clients uniformly.” The practical implication of the uniform access rule is you must be very careful with the order in which you specify mount points, or shares, in the MNT_POINTS variable.

The fool-proof way to comply with the uniform access rule is to have every client access every file system, evenly spreading the load across the network paths between the client and server. This works pretty well for small systems, but may require more procs per client than you want to use when testing large servers.

If you want to run fewer procs on your clients' than you have file systems, you will need to take some care figuring out the mount points, or shares, for each client.

Below are some examples of generating mount point lists which comply with the uniform access rule. To begin, you must first determine the number of file systems, clients, and load generating processes you will be using. Once you have that, you can start deciding how to assign procs to file systems. As a first example, we will use the following file server:

Clients C1 and C2 are attached to Network1, and the server's address on that net is S1. It has two disk controllers (DC1 and DC2), with four file systems attached to each controller (F1 through F8).



You start by assigning F1 to proc1 on client 1. That was the easy part.
You next switch to DC2 and pick the first unused file system (F5). Assign this to client 1, proc 2.
Continue assigning file systems to client 1, each time switching to a different disk controller and picking the next unused disk on that controller, until client 1 has PROC file systems. In the picture above, you will be following a zig-zag pattern from the top row to the bottom, then up to the top again. If you had three controllers, you would hit the top, then middle, then bottom controller, then move back to the top again. When you run out of file systems on a single controller, go back and start reusing them, starting from the first one.
Now that client 1 has all its file systems, pick the next controller and unused file system (just like before) and assign this to client 2. Keep assigning file systems to client 2 until it also has PROC file systems.
If there were a third client, you would keep assigning it file systems, like you did for client 2.
If you look at the result in tabular form, it looks something like this (assuming 4 procs per client):

C1: S1:F1 S1:F5 S1:F2 S1:F6
C2: S1:F3 S1:F7 S1:F4 S1:F8

The above form is how you would specify the mount points in a file. If you wanted to specify the mount points in the RC file directly, then it would look like this:

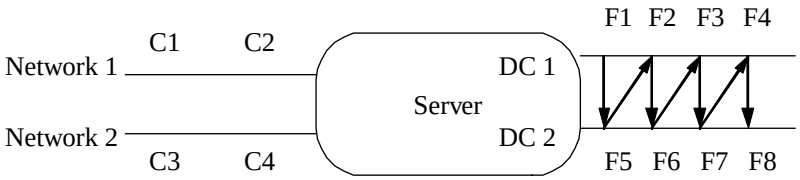
CLIENTS=”C1 C2”
PROCS=4
MNT_POINTS=”S1:F1 S1:F5 S1:F2 S1:F6 S1:F3 S1:F7 S1:F4 S1:F8

If we had 6 procs per client, it would look like this:

C1: S1:F1 S1:F5 S1:F2 S1:F6 S1:F3 S1:F7
C2: S1:F4 S1:F8 S1:F1 S1:F5 S1:F2 S1:F6

Note that file systems F1, F2, F5, and F6 each get loaded by two procs (one from each client) and the remainder get loaded by one proc each. Given the total number of procs, this is as uniform as possible. In a real benchmark configuration, it is rarely useful to have an unequal load on a given disk, but there might be some reasons this makes sense.

The next wrinkle comes if you should have more than one network interface on your server, like so:



Clients C1 and C2 are on Network1, and the server's address is S1. Clients C3 and C4 are on Network2, and the server's address is S2.
We start with the same way, assigning F1 to proc 1 of C1, then assigning file systems to C1 by rotating through the disk controllers and file systems. When C1 has PROC file systems, we then switch to the next client on the same network, and continue assigning file systems. When all clients on that network have file systems, switch to the first client on the next network, and keep going. Assuming two procs per client, the result is:

C1: S1:F1 S1:F5
C2: S1:F2 S1:F6
C3: S2:F3 S2:F7
C4: S2:F4 S2:F8

And the mount point list is:

MNT_POINTS="S1:F1 S1:F5 S1:F3 S1:F7 S2:F2 S2:F6 S2:F4 S2:F8"

The first two mount points are for C1, the second two for C2, and so forth.

Uniform access is a slippery subject. It is much easier to examine a configuration and say whether it is uniform than it is to come up with a perfect algorithm for generating complying mount point lists. There will always be new configurations invented which do not fit any of the examples described below. You must always examine the access patterns and verify there is nothing new and innovative about your system which makes it accidentally violate the uniform access rule.

For instance, when constructing a run of the SPECsfs2008 benchmark for a cluster of servers that present a single namespace (that is, report the number of filesystems as one namespace), special considerations are required to maintain compliance with the Uniform Access Rule. Any server in a single namespace cluster can serve data from all logical disk subsystems attached to or managed by any server in the cluster. Because each server in a traditional cluster only serves data from its locally attached logical disk subsystems, UAR sets a higher standard for single namespace clusters: UAR requires each server in a single namespace cluster to serve the same amount of data traffic for each logical disk subsystem in the cluster.

In the benchmark configuration, UAR is accomplished by specifying mount points that enter the server cluster at specific nodes (via the IP Address of the desired server) with directories that target specific logical disk subsystems. When done properly for single namespace clusters, the number of unique paths is equal to the number of servers multiplied by the number of logical disk subsystems. The point of this stronger requirement is to assure that no architectural or communication shortcuts are made when measuring a cluster of servers that is acting much like a single storage system.

As an example of the UAR for clustered servers claiming a single namespace, consider a 3-node cluster in which each server has one logical disk subsystem associated with it. The MNT_POINTS file might look something like this ...

(NFS example)

test01 10.1.1.1:/disk1 10.1.1.2:/disk1 10.1.1.3:/disk1
test02 10.1.1.1:/disk2 10.1.1.2:/disk2 10.1.1.3:/disk2
test03 10.1.1.1:/disk3 10.1.1.2:/disk3 10.1.1.3:/disk3

or

(CIFS example)

test01 \\10.1.1.1\disk1 \\10.1.1.2\disk1 \\10.1.1.3\\disk1
test02 \\10.1.1.1\disk2 \\10.1.1.2\disk2 \\10.1.1.3\\disk2
test03 \\10.1.1.1\disk3 \\10.1.1.2\disk3 \\10.1.1.3\disk3

These examples are meant to be only that, examples. There are more complicated configurations which will require you to spend some time analyzing the configuration and assuring yourself (and possibly SPEC) that you have achieved uniform access. You need to examine each component in your system and answer the question “is the load seen by this component coming uniformly from all the upstream components, and is it being passed along in a uniform manner to the downstream ones?” If the answer is yes, then you are probably in compliance.

5 Benchmark Execution Requirements

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

5.1 Valid methods for benchmark execution

The benchmark must always be executed by using Java to start the SfsManager on the prime client as well as on all of the load generators.

5.2 Server File System Creation and Configuration

The file server's target file systems, their configuration and underlying physical medium used for benchmark execution must follow the stable storage requirements detailed in the section "Description of Stable Storage for SPECsfs2008".

At the start of each benchmark run, before the first in a series of requested load levels is generated, the file server's target file systems must be initialized to the state of a newly-created, empty file system. For UNIX-based systems, the **mkfs** (make file system) or **newfs** (new file system) command would be used for each target file system. For non-UNIX-based systems, a semantic equivalent to the **mkfs** or **newfs** command must be used. (ie. Format)

5.3 Data Point Specification for Results Disclosure

The result of benchmark execution is a set of 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 load level and proceeding to the highest requested load level.

Published benchmark results must include at least 10 uniformly spaced requested load points (excluding zero ops/sec). The distance between zero and the first requested load point must be the same as the distance between any other consecutive load points that are uniformly spaced. For example, in a submission where only 10 uniformly spaced load points are reported, the first point must be 1/10th of the last point. Note that due to rounding limitations, load points generated automatically via the INCR_LOAD parameter will sometimes exhibit small deviations from strict uniformity (if the desired load points are not evenly divisible by the number of load generators times the number of processes per load generator). These slight deviations due to internal calculations by the benchmark code are expected and allowable.

Two additional non-uniformly spaced requested load points beyond the highest uniformly spaced point may also be included. The achieved throughput of the optional non-uniformly spaced data points should be no more than 5% higher than the highest uniformly spaced achieved throughput data point.

The highest achieved throughput must be within 10% of the requested throughput for it to be considered a valid data point. Any invalid data points will invalidate the entire run unless they are at or below 25% of the maximum measured throughput. All data points in the run series prior to and including the last disclosed data point 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/format") are allowed during the execution of the benchmark or between data point collection. If any requested load level or data point must be rerun for any reason, the entire benchmark execution must be restarted, i.e., the server's file systems must be initialized and the series of requested load levels repeated in whole.

5.4 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 20 milliseconds.

5.5 Overall response time calculation

The overall response time is an indicator of how quickly the system under test responds to NFS or CIFS operations over the entire range of the tested 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 be directly proportional throughput, with zero response-time at zero throughput.

5.6 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 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:

5.6.1 LOAD

Initial value for requested operations/sec, or a complete list of the data points to be collected by the benchmark. The list must increase in value and must represent a uniform distribution. If the list consists of more than a single value, at least 10 uniformly spaced data points must be specified for valid benchmark execution.

5.6.2 INCR_LOAD

Incremental increase in load for successive data points in a benchmark run. This parameter is used only if **LOAD** consists of a single (initial) value. To ensure equally spaced points, the value of **LOAD** and **INCR_LOAD** must be equal.

5.6.3 NUM_RUNS

The number of load points to run and measure (minimum of 10 for a publishable result). This parameter is used only if **INCR_LOAD** is specified.

5.6.4 PROCS

Number of processes per client. Each client load generator may be able to generate more load if the client has sufficient resources to do so. A general rule of thumb is to have the total requested load be divided across all of the clients, and to have sufficient numbers of clients and processes so as to have the operations/sec per process remain below 250 at the highest load point. It is also recommended to have the operations/sec per process remain above 10 at the lowest load point.

At least eight processes must be used for each network 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. If there are less than 8 processes for each network then the result will be non-compliant with the SFS run rules.

5.6.5 CLIENTS

List of clients to use in this test. The Prime client, if listed here, may also be used to generate load. If the Prime client is not listed here then it will only coordinate the testing and will not participate in generating load. The client names in this list are hostnames or IP addresses of the clients that will be participating in generating the load.

5.6.6 MNT_POINTS

List of mount points, or shares, to use in the testing. This list must be generated to comply to the uniform access requirements defined in "Description of Uniform Access for SPECsfs2008". Each of these mount points must be exported by the server so that they may be mounted by the load generating clients. The value MNT_POINTS can take several different forms:

- UNIX style: server:/exportfs1 server:/exportfs2 …
- CIFS style: \\server\exportfs1 \\server\exportfs2 …
- Use a file that contains the mount points: filename
  The use of a file, and its format, is covered later in this document.

The number of mount points in the list must be equal to number of processes specified in the PROCS parameter. Note that a mount point may be repeated in the list.

5.6.7 BIOD_MAX_WRITES

The number of outstanding or async writes that the benchmark will generate per benchmark process. The minimum number is 0 and the maximum number is 32. (Only applicable when running the NFS workload.)

5.6.8 BIOD_MAX_READS

The number of outstanding or async reads that the benchmark will generate per benchmark process. The minimum number is 0 and the maximum number is 32. (Only applicable when running the NFS workload.)

5.6.9 FS_PROTOCOL

The type of server protocol (NFS or CIFS) to test. It may be set to "nfs" or "cifs". Either UNIX or Windows clients can be used to test either NFS or CIFS, however, all clients must be of the same type. ***Note: If this value is set to "nfs" then the MNT_POINTS list must use the UNIX style syntax. If this value is set to "cifs" then the MNT_POINTS list must use the CIFS style syntax.***

5.6.10 USERNAME

The CIFS account name which is configured on all clients to be used for the benchmark execution. (Only applicable when running the CIFS workload.)

5.6.11 PASSWORD

The CIFS password for the user specified in USERNAME. (Only applicable when running the CIFS workload.)

5.6.12 DOMAIN

The CIFS domain name to be used for the benchmark testing. (Only applicable when running the CIFS workload.)

5.6.13 SFS_DIR

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

5.6.14 SUFFIX

The suffix to add to the log file names.

5.6.15 WORK_DIR

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

5.6.16 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.

5.6.17 PRIME_MON_ARGS

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

5.6.18 INIT_TIMEOUT

The maximum time (in seconds) that the benchmark will run during the working set initialization phase for a single data point before timing out. This value may be increased as needed, e.g., when using a slow I/O subsystem, in order to keep the benchmark from timing out during initialization.

5.6.19 BLOCK_SIZE

The maximum block (RPC) size which the load generators will use for network communication with the NFS server. If this value is not set, the load generators will auto-negotiate the block size with the server based on the server's advertised preferred size. (Only applicable when running the NFS workload.)

5.6.20 SFS_NFS_USER_ID

The UID of the user's account on the NFS server for the user who owns the test file system(s), i.e., the ones listed in **MNT_POINTS**. (Only applicable when running the NFS workload using Windows clients.)

5.6.21 SFS_NFS_GROUP_ID

The GID of the user's account on the NFS server for the user who owns the test file system(s), i.e., the ones listed in **MNT_POINTS**. (Only applicable when running the NFS workload using Windows clients.)

6 SFS Submission File and Reporting Form Rules

There are rules associated with the fields in the submission report and in the corresponding sections of the reporting form. Rules for valid and/or required values in the fields are described below. The description for the Processing Elements field is complex enough that it is contained in its own subsection after the table.

6.1 Submission Report Field Descriptions

Tag	Description	Valid Contents
specSFS4_0Info	The entire set of information contained in an info file is covered under this top-level tag.	Does not contain a value.
. productInfo	The information about the product in the report.	Does not contain a value.
. . vendorAndProduct	A collection of vendor, general product, and license info.	Does not contain a value.
. . . testedBy	The name of the SPEC licensee who is publishing this report.	String
. . . productName	The name of the system that was tested for this report.	String
. . . hardwareAvailable	The date all of the product's hardware is available for the public to acquire (by purchase, lease, or other arrangement).	String
. . . softwareAvailable	The date the product's software is available for the public to acquire. Note that this is the latest availability date for the software components described in the Bill of Materials for the SUT.	String
. . . dateTested	The date the product was tested.	String
. . . licenseNumber	The SPEC SFS License number for the company	String
. . . licenseeLocation	A free-form description of the Licensee's location (e.g. city, state, country).	String
. . . otherProductInfo	A free-form description of the product.	String
. . productBomList	The Bill of Materials for the SUT. This list should be sufficient for anyone to purchase and physically configure an identical system to the SUT. (Small components such as power and network cables that would be obviously necessary to complete a configuration may be left out as long as there are no performance-sensitive differences in part choices.) This should include the names and versions of any software used in the SUT.	Does not contain a value.
. . . bomItem	A single record in the productBomList.	Does not contain a value.
. . . . quantity	The number of this item used in the SUT.	Integer

. . . . type	The BOM item type of the associated BOM item - this designates what functionality this item provides to the configuration. Recommended values include but aren't limited to: Disk, Disk Enclosure, Disk Controller, FC Switch, Ethernet Switch, Server, Infiniband Switch, and Software.	String
. . . . vendor	A string that names the supplier of this component.	String
. . . . model	The model number or name that uniquely identifies this item for ordering purposes.	String
. . . . description	A free-form description of the named item.	String
. . serverSoftware	Information about the software running on the server being tested.	Does not contain a value.
. . . operatingSystem	The name and version of the operating system running on the server being tested.	String
. . . filesystemSoftware	The name and version of the software providing the filesystem being used on the test filesystems.	String
. . . otherSoftware	Information about any other software and software versions running on the server being tested that are separate and distinct from the given OS name and version.	String
. . serverTuning	Server tuning information for the system under test.	Does not contain a value.
. . . serverTuningList	A sequence of descriptions of tunings used on the SUT.	Does not contain a value.
. . . . tuningParam	A tuning parameter used in the SUT.	Does not contain a value.
. . . . . name	The name of the tuning parameter that was set. (e.g. NFS Threads)	String
. . . . . value	The value to which the associated tuning parameter was set. (e.g. 256 Threads)	String
. . . . . description	A description of the effect of the associated tuning parameter.	String
. . . serverTuningNotes	A free-form text field for additional server tuning notes for the system under test. Note changes to any configuration files or non-standard options used here if they are not covered in the server tuning table. If any entry in the server tuning table needs further explanation, that explanation should go here.	String
. . configDiagramList	A series of pictures that form a schematic diagram of the system under test. All components described in both the *productBomList* and the *testBomList* and how they are connected should be represented. Repeated components such as disk drives can be indicated with an ellipsis.	Does not contain a value.
. . . configDiagram	A name and file name pair for one member of the list.	Does not contain a value.
. . . . name	The name to appear in a link to a configuration diagram.	String

. . . . ref	The file name of the configuration diagram. Diagrams must be in JPEG format. See http://www.jpeg.org for a specification of JPEG.	String
. . disksAndFilesystems	A collection of information about the disks and filesystems in the SUT.	Does not contain a value.
. . . diskSetList	A list of groups of disks in the SUT. A disk is a memory device that provides durable storage. That is, it holds information including the filesystem data of the benchmark that persists beyond the loss of power to the SUT. All disks in the SUT must be accounted for in the list.	Does not contain a value.
. . . . diskSet	A collection of disks in the SUT.	Does not contain a value.
. . . . . quantityOfDisks	The total number of disks in this set.	Integer
. . . . . usableGb	The total size of the disks in this set in gigabytes of usable space, that is, space that is presented to the file server OS. Usable space may be consumed by data or filesystem metadata. It does not include RAID parity information or any other information. needed to make the group or an individual disk usable. The size specified here must be in GB, if greater than 1 TB the value will be scaled in generated reports.	Decimal
. . . . . description	A free-form description of any important features of the collection such as the type of the individual disks and their RAID organization. For traditional disks this should include their raw size, the rotational speed, and the kind of interconnect if not reported in the Bill of Materials.	String
. . . fsInfo	Information about the filesystem(s) used in the test.	Does not contain a value.
. . . . fsType	The name and version of the filesystem type used in the test.	String
. . . . fsQuantity	The number of filesystems used in the test.	Integer
. . . . totalExportedCapacity	The total filesystem capacity exported from the file server(s). You must specify units - for example, "760 GB" or "4.5 TB".	String
. . . . fsCreation	A free-form description of how the filesystem was created including any specific options. Use "default" in this field if the default options for your product were used when creating the filesystem.	String
. . . . fsConfig	A free-form description of how each of the filesystems maps onto the disks described above.	String
. . . diskAndFsNote	An optional free-form block of additional information about the disk and filesystem configuration.	String
. . networkInterfaceList	A sequence of descriptions of network interfaces contained in the SUT.	Does not contain a value.
. . . networkInterface	A network interface in a component in the SUT.	Does not contain a value.

. . . . networkType	The type of network supported. (e.g. Jumbo Gigabit Ethernet)	String
. . . . portsUsed	The number of ports on this interface used in the test. If the choice of ports is significant, then name the ports used in the notes.	Integer
. . . . networkNotes	A free-form description of additional information about the interface including any special configuration options used.	String
. . networkConfigurationNotes	A free-form description of the network configuration used in the system under test.	String
. . processingElements	Processing elements information for the SUT. See section 6.2 for details.	Does not contain a value.
. . . processingElementList	A list of unique processing elements in the SUT.	Does not contain a value.
. . . . procElement	A unique processing element (general-purpose CPU, ASIC, etc.) in the SUT.	Does not contain a value.
. . . . . quantity	Number of processing elements of the specified type.	Integer
. . . . . type	The type of processing element, e.g., general-purpose CPU, ASIC, etc.	String
. . . . . description	A description of the key technical characteristics of the processing element. A description of the memory contained within the processing elements does not need to be included here, but may be required in the top-level Memory field. Refer to that field for instructions.	String
. . . . . processingFunction	A high-level description of the processing function(s) that the processing element performs, e.g., NFS, CIFS, TCP/IP, RAID, etc.	String
. . . procElementNotes	Any other relevant description of the processing elements, e.g., the location of the elements within the system architecture.	String
. . memory	A collection of information about every unique set of memory in the SUT for which the sum of the memory of that given type is greater than 2 percent of all the memory in the system. This should include such components as storage processors, RAID controllers, gate arrays, and TCP/IP-offload engines. It also includes the main memory, excluding processor caches, of all components of the SUT with the exception of components that only support administrative functions such as a remote console. Other exceptions may be considered by the review committee but should be requested prior to submission. Note that the greater-than-2%-limit applies to the set not an individual. If many individual components sum to greater than 2% of all memory in the system, then they should be included. Do not include processor cache.	Does not contain a value.
. . . memorySetList	A sequence of descriptions of distinct memory groupings in the SUIT.	Does not contain a value.

. . . . memorySet	A distinct grouping of memory in the SUT.	Does not contain a value.
. . . . . sizeGb	The number of gigabytes of usable memory in this group - may be a fractional amount (e.g. 0.5, 1.5).	Decimal
. . . . . quantity	The number of instances of this memory grouping in the SUT	Integer
. . . . . nonVolatile	NV - memory is nonvolatile - or V - memory is volatile. If NV is specified, then the memory group description text should explain how the persistence is accomplished and the time span of the persistence.	String matching pattern: ^N?V$
. . . . . description	A free-form description of this class of memory.	String
. . . memoryNotes	An optional free-form description of additional information about the system or the overall reporting.	String
. . stableStorage	A free-form description of how the SUT conforms to the SFS Stable Storage requirement. (See SPEC's Description of Stable Storage for SFS 2008.)	String
. . sutConfigNotes	A free-form description of additional information needed to reproduce the test using the above equipment. This is a description of the picture of the system.	String
. . otherSutNotes	An optional free-form description of additional information about the SUT.	String
. testInfo	Information about how the test was run.	Does not contain a value.
. . testBomList	The Bill of Materials for the equipment used in the test infrastructure. See bom-item above for a description of the component records. This list should be sufficient for anyone to physically configure an identical system to reproduce the test environment.	Does not contain a value.
. . . bomItem	A single record in the testBomList.	Does not contain a value.
. . . . quantity	The number of this item used in the SUT.	Integer
. . . . vendor	A string that names the supplier of this component. This tag may be omitted if the component supplier is the publication vendor.	String
. . . . model	The model number or name that uniquely identifies this item for ordering purposes.	String
. . . . description	A free-form description of the named item.	String
. . loadGeneratorList	A list of records describing the test client machines.	Does not contain a value.
. . . lgClass	A description of a single class of load generator client machines.	Does not contain a value.
. . . . identifier	A string that identifies this type of load generator. The identifier is used in the testbedConfig record below.	String
. . . . bomNumber	The ordinal number of this system type in the testBomList.	Integer

. . . . processorName	A string that identifies the name of the processor in the load generator.	String
. . . . processorCount	The integer number of processors (chips) in the load generator.	Integer
. . . . processorSpeed	A string that describes the speed of the processor(s) in the load generator.	String
. . . . coresPerChip	The integer number of cores per processor (chip) in the load generator.	Integer
. . . . memorySizeGb	The number of gigabytes of memory.	Decimal
. . . . osVersion	A string describing the name and version of the operating system running on the load generator.	String
. . . . netController	The number and type of network controllers used to connect the load generator to the test network.	String
. . testNetworkConfig	A free-form description of the configuration settings necessary to connect the SUT to the test clients. This description (plus the network components listed in the test-bom) should include sufficient information to allow an outsider to reproduce the test network.	String
. . lgConfig	A collection of information about the load generator and load generator-related benchmark configuration.	Does not contain a value.
. . . nasType	Protocol used by NAS: "NFS V3" or "CIFS"	String. Possible values: • NFS V3 • CIFS
. . . biodMaxRead	BIOD Max Read setting - only relevant for NFS. Set to N/A for CIFS.	String
. . . biodMaxWrite	BIOD Max Write setting - only relevant for NFS. Set to N/A for CIFS.	String
. . . numberOfProcs	The number of SFS sub-processes to generate load per LG.	Integer
. . . blockSize	The block size used by the benchmark if set to a specific value in the rc file, or auto if the block size parameter was left blank (the default value), resulting in auto-negotiation.	String
. . . testbedList	A collection of testbed records that describe the relations between the clients and the SUT and its filesystems.	Does not contain a value.
. . . . testbed	A description of one or more load generators.	Does not contain a value.
. . . . . lgStart	A number that is the ordinal of the first member of the set of LGs defined in this record.	Integer
. . . . . lgEnd	A number that is the ordinal of the last member of the set.	Integer

. . . . . lgType	The identifier attribute from the corresponding lgClass record.	String
. . . . . netName	The name of the network that this testbed is connected to.	String
. . . . . targetFilesystems	A string that lists the names of the filesystems mounted on this client.	String
. . . . . description	An optional free-form text description that further explains how the filesystems were mounted and/or other information about this client or group of clients.	String
. . . lgConfigNotes	Any additional notes about the load generator configuration - this should include any configuration changes such as tuning parameters.	String
. . uniformAccessRule	A freeForm description of how the test conformed to the Uniform Access Rule. [[reference to doc?]]	String
. . otherTestNotes	An optional freeForm description of additional information about the test environment and/or execution.	String
. results		Does not contain a value.
. . result		Does not contain a value.
. . . valid	Whether the given data point in the results is valid. Can be Y or N.	String. Possible values: • Y • N
. . . load	The requested load in ops/sec	Integer
. . . throughput	The achieved throughput in ops/sec	Integer
. . . responseTime	The response time in msec	Decimal
. . . totalOps		Integer
. . . elapsedTime		Integer
. . . protocol		String
. . . transportProtocol		String
. . . ipVersion		Integer
. . . filesetSize		Integer
. . . clientCount		Integer
. . . numProcs		Integer
. . . biodRead		Integer
. . . biodWrite		Integer
. . . version		String
. otherReportNotes	An optional free-form description of additional information about the system or the overall reporting.	String

. resultCompliance	Information detailing the compliance or non-compliance of this result.	Does not contain a value.
. . compliantResult	Whether this SPEC result is compliant with the run and reporting rules. Can be Y or N.	String. Possible values: • Y • N
. . nonComplianceText	A free-form text description with details of why this result is non-compliant.	String
. submissionInfo	Information about the SPEC result submission that is relevant to SPEC submission reviewers. This information is not displayed in the final submission reports.	Does not contain a value.
. . submitterName	The name of the person submitting this SPEC SFS result.	String
. . submitterEmail	The email address of the person submitting this SPEC SFS result.	String
. . reviewersComments	Additional comments about the submission to help with the review process. This might include descriptions of tunables and other information that might be relevant during the review process.	String

6.2 Processing Elements Field Description

The Processing Elements field should include a description of all the major processing elements involved in servicing the NFS/CIFS requests generated by the benchmark, and in producing responses to those requests. This description may include, but is not limited to, those elements involved in processing for the network file system protocol (NFS/CIFS), the networking protocols (TCP/IP), the exported or shared local file systems, and associated drivers. An example of a typical system architecture showing processing elements that should be described is provided in the diagram below. The description does not need to include processing elements used solely for the purpose of system management activities that do not impact the processing of SFS requests in any way (e.g., monitoring control processors).

These sub-fields show up in a table in the submission form and should include the following information about the processing elements:
1) **Item**: Item number, one row for each different type of processing element.
2) **Qty**: Number of processing elements of the specified type.
3) **Type**: The type of processing element, e.g., general-purpose CPU, ASIC, etc.
4) **Description**: A description of the key technical characteristics of the processing element. A description of the memory contained within the processing elements does not need to be included here, but may be required in the top-level **Memory** field. Refer to that field for instructions.
5) **Processing Function**: A high-level description of the processing function(s) that the processing element performs, e.g., NFS, CIFS, TCP/IP, RAID, etc.

If the processing elements are general purpose CPUs (or processors), the **Qty** field should contain the number of processors in the system. As of early 2008, it is assumed that processors can be described as containing one or more "chips", each of which contains some number of "cores", each of which can run

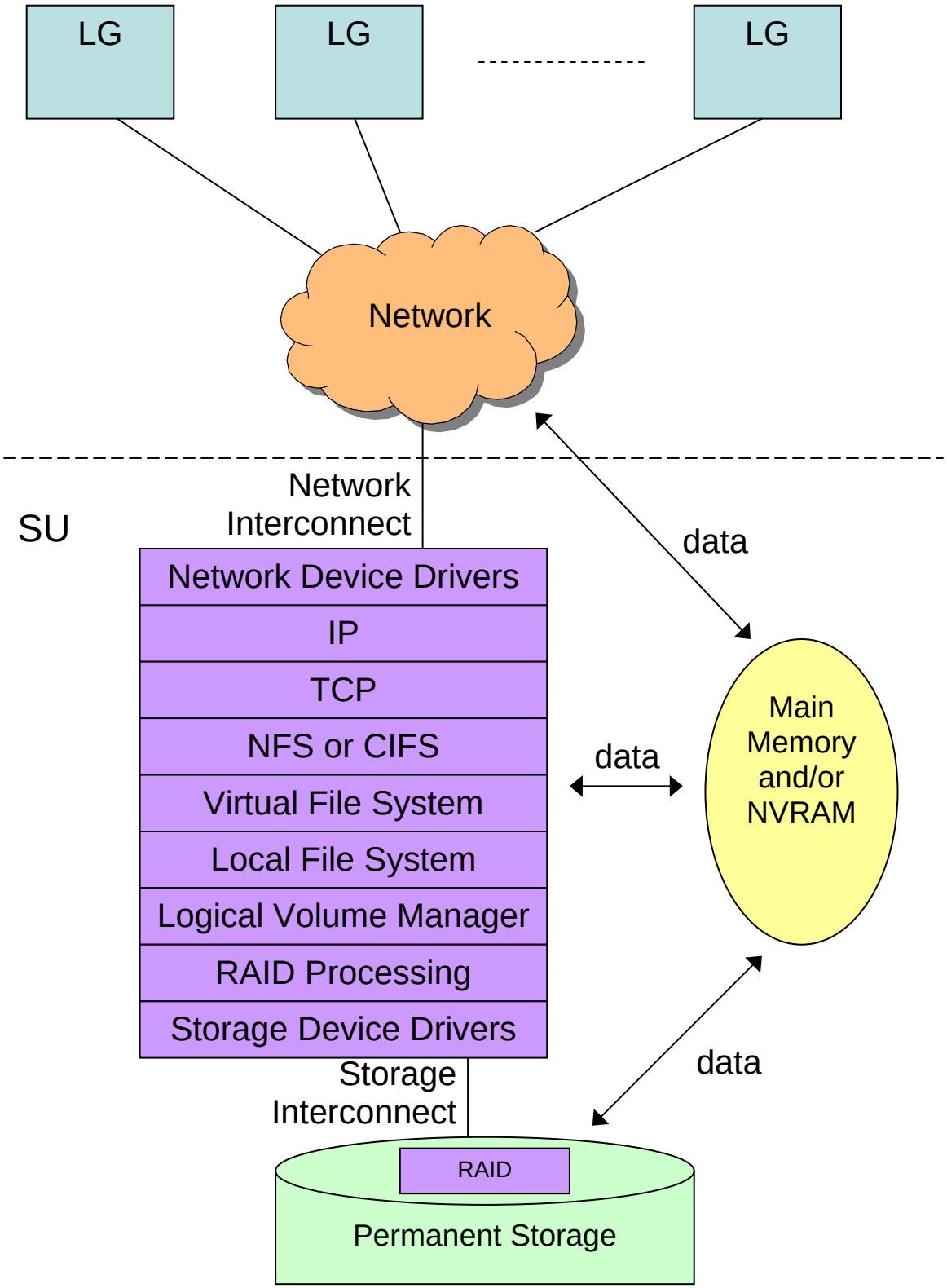
some number of hardware "threads". Therefore, the following the following characteristics of the CPUs should be provided under **Description**:

1) Name: A manufacturer-determined processor formal name.
2) Speed: A numeric value expressed in megahertz or gigahertz. The value here is the speed at which the CPU is run.
3) Cores Enabled. Number of processor cores enabled during the test
4) Chips Enabled. Number of processor chips enabled during the test
5) Cores/Chip. Number of processor cores that are manufactured into a chip (irrespective of whether or not the cores are enabled)
6) HW Threads/Core. Number of hardware threads enabled per core during the test
7) Characteristics: Any other description necessary to disambiguate which processor is used, in case CPU Name and CPU Speed are not sufficient (e.g., L1/L2/L3 cache sizes, etc.)

If the processing elements are not general-purpose CPUs, e.g., application-specific integrated circuits (ASICs) such as FPGAs, the **Qty** field should contain the number of ICs (chips) used during the test. In addition, the **Description** field should include details on meaningful technical characteristics of the processing element, e.g., manufacturer part number, clock speed, etc. It does not have to be as structured a description as it is for general-purpose CPUs.

Any other relevant description of the processing elements, e.g., the location of the elements within the system architecture, may be included under **Processing Elements Notes**.

The diagram below is provided solely as an example to help identify processing elements that should be disclosed. It is not intended to represent the system architecture of any specific product. The purple boxes are examples of the important processing functions to keep in mind when trying to identify the key processing elements in your SUT.


LG
LG
LG
Network
SU
Network
Interconnect
Network Device Drivers
IP
TCP
NFS or CIFS
Virtual File System
Local File System
Logical Volume Manager
RAID Processing
Storage Device Drivers
Storage
Interconnect
RAID
Permanent Storage
data
data
data
Main
Memory
and/or
NVRAM