SPEC SFS® 2014 SP2 User's Guide

Standard Performance Evaluation Corporation (SPEC)
7001 Heritage Village Plaza
Suite 225
Gainesville, VA 20155

Phone: 1-703-579-8460
Fax: 1-703-579-8463
E-Mail: info@spec.org

Copyright (c) 2014, 2017 by Standard Performance Evaluation Corporation (SPEC)
All rights reserved

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

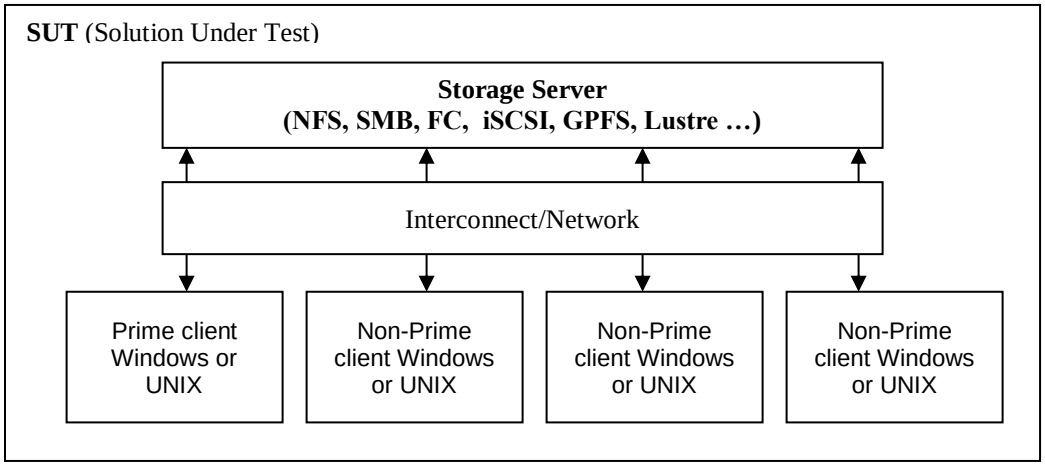
Table of Contents

1 Quick Start Guide

The SPEC SFS 2014 benchmark is used to measure the maximum sustainable throughput that a storage solution can deliver. The benchmark is protocol independent. It will run over any version of NFS or SMB, clustered file systems, object oriented file systems, local file systems, or any other POSIX compatible file system. Because this tool runs at the application system call level, it is file system type agnostic. This provides strong portability across operating systems, and storage solutions. The SPEC SFS 2014 benchmark already runs on Linux, Windows 7, Windows 8, Windows 10, Windows Server 2008, Windows Server 2012, Windows Server 2016, Mac OS X, BSD, Solaris, and AIX, and can be used to test any of the file system types that these systems offer.

The SPEC SFS 2014 benchmark consists of multiple workloads which represent real data processing file system environments. Each of these workloads may be run independently. A SPEC SFS 2014 benchmark publication may use any one of these workloads. The SfsManager allows the user to input parameters, run the benchmark, and review the results for all workloads.

The benchmark runs on a group of workstations and measures the performance of the storage solution that is providing files to the application layer. Because the benchmark runs at the application system call level, all components of the storage solution impact the performance of the solution – this includes the load generators themselves as well as any physical or virtual hardware between the load generators and where the data ultimately rests on stable storage.



Example topology

The minimal configuration consists of one load generating client, and one NFS, SMB, iSCSI, FCP, GPFS, or Luster server. For Windows-based environments, a separate prime client is required for a minimal configuration.

The steps to produce a SPEC SFS 2014 result are:

- Install the SPEC SFS 2014 benchmark on the load generators
- Edit the configuration file on the prime client
- Configure the solution for testing
- Start the benchmark

1.1 Prerequisites

- Python version 2.7 or later must be installed on the Prime Client system.
- Matplotlib must be installed on the Prime Client system.
  See: http://www.Matplotlib.org
- Matplotlib dependencies (dateutil , numpy , pyparsing & six) must also be installed.
  dateutil …… pip install python-dateutil
  Numpy …… http://www.Numpy.org
  Pyparsing … easy_install pyparsing
  Six ……... https://pypi.python.org/pypi/six
- The test file systems must have the permissions set correctly in order to allow access by the clients.
- The test file systems must be mounted or mapped prior to execution of the benchmark. For Windows shares one may use UNC paths without mapping the shares.
- There must be interconnect/network connectivity between any storage system and clients, and between the clients and the Prime Client. The Prime Client is simply the system on which the benchmark run is started, and could be one of the clients. The prime client may also present load, in some configurations. When running on Windows, the prime client cannot also generate load, so at least one additional client is required.
- The contents of the SPEC SFS 2014 benchmark distribution must be accessible on all the systems where the benchmark will be installed.
- Using these quick start procedures assumes that the pre-compiled C code binaries, shipped with the benchmark, will be used.

1.2 Installing the SPEC SFS 2014 Benchmark

The SPEC SFS 2014 benchmark can be installed on client machines running either a UNIX-based or Windows operating system. Each of these require slightly different configuration and are described separately below.

UNIX client installation and configuration:

- Ensure that DNS is correctly configured.
  - Ensure that the client's hostname does not appear in the hosts file on either the 127.0.0.1 or ::1 line!
- Ensure that any/all forms of firewalls are disabled on all of the clients. (SELinux, iptables…)
- Ensure that all clients have ssh installed and configured such that clients can start commands on each other without any password challenges. (Example: ssh hostname command) Setup the ssh keys, and permit empty passwords.
- Install Python 2.7, and ensure that python is in the user's search path. Python may be downloaded from http://www.python.org.
- Install the SPEC SFS 2014 benchmark using the following steps:
  - Login to the client (as root)
  - Download or copy the SPEC SFS 2014 ISO or archive to the client
  - Loop mount the ISO or expand the archive
  - cd to the top level directory containing the SPEC SFS 2014 contents
  - Enter 'python SfsManager --install-dir="*destination_directory*"' (where "*destination_directory*", without the double-quotes, is the full path where you wish to have the benchmark installed)

- It is strongly recommended to apply the recommended tuning for somaxconn in 11.1 below for Linux hosts

Windows client installation and configuration:

- Ensure that DNS is correctly configured.
  - Ensure that the client's hostname does not appear in the hosts file on either the 127.0.0.1 or ::1 line!
- Ensure that all Windows clients are properly configured in the Active Directory Domain.
- Install Python 2.7 or later, and ensure that python is in the user's search path. Python may be downloaded from http://www.python.org
- Install the SPEC SFS 2014 benchmark
  - Start a command prompt window. This can be done using the 'Start' button, choosing 'Run…' and entering 'cmd'.
  - Download or copy the SPEC SFS 2014 ISO or archive to the client
  - Attach the ISO as a virtual optical drive or expand the archive
  - chdir to the top level directory containing the SPEC SFS 2014 directory
  - Enter 'python SfsManager --install-dir="*destination_directory"*' (where "*destination_directory"*, enclosed by double-quotes, is where you wish to have the benchmark installed)
    Note: Please use a destination_directory that is *not* in the user's home directory. The actual path to a User's home directory location varies widely across versions of Windows and can make heterogeneous clients problematic.

1.3 Editing the configuration file on the prime client

On the prime client, the default configuration file is called sfs_rc. The user must edit the sfs_rc configuration file, and only needs to edit it on the prime client. The user does not need to edit, or even have, a configuration file on the other load generating clients. On the prime client, edit the values for:

- **BENCHMARK**
  Name of the benchmark to run. Valid values are: **DATABASE**, **EDA**, **SWBUILD**, **VDA**, or **VDI**.

- **LOAD**
  Each workload has an associated business metric as a unit of workload. The magnitude of the workload to run is specified with the **LOAD** parameter in units of the workload's business metric. Valid values for **LOAD** are either a starting number or a list of values, all positive integers. If a single value is specified, it is interpreted as a starting value and used in conjunction with **INCR_LOAD** and **NUM_RUNS**. The following table shows the name for the business metric corresponding to each workload type.

Workload	Business Metric (LOAD parameter)
DATABASE	DATABASES
EDA	JOB_SETS
SWBUILD	BUILDS
VDA	STREAMS
VDI	DESKTOPS

If a list of values if specified, at least 10 uniformly spaced data points must be specified for valid benchmark execution. For more detail on the requirements for uniformly spaced data points, see section 5.3 "Data Point Specification for Results Disclosure" in the SPEC SFS® 2014 SP2 Run and Reporting Rules.

As a helpful guideline, some rules of thumb for the resources required per business metric for the different workloads:
Capacity requirements per business metric:

DATABASE	= 24 GiB per DATABASE
EDA	= 11 GiB per JOB_SET
SWBUILD	= 5 GiB per BUILD
VDA	= 24 GiB per STREAM
VDI	= 12 GiB per DESKTOP

Client memory requirements per business metric:

DATABASE	= 55 MiB per DATABASE
EDA	= 12 MiB per JOB_SET
SWBUILD	= 400 MiB per BUILD
VDA	= 10 MiB per STREAM
VDI	= 8 MiB per DESKTOP

- **INCR_LOAD**
  Incremental increase in load for successive data points in a 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.

- **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.

- **CLIENT_MOUNTPOINTS**
  The list of local mount points, local directories, or shares, to use in the testing. The value of CLIENT_MOUNTPOINTS can take several different forms:
  - UNIX style: client:/exportfs1 client:/exportfs2 …
    - Used for local storage or mounted network shares
  - Windows style: client:\\server\exportfs1 client:\\server\exportfs2 …
  - Use a file that contains the mount points: mountpoints_file.txt
    - For more detail, see section 3.3 "*Configuring the Required Benchmark Parameters*" below

The CLIENT_MOUNTPOINTS list is used to assign both a load generator as well as a storage location to a single business metric. The order of entries in the CLIENT_MOUNTPOINTS list matters, as the list is consumed sequentially to assign a client and a storage location to each business metric as the load scales up. Once the list of CLIENT_MOUNTPOINTS is exhausted, the list will be reused from the start as many times as necessary.

The order of the CLIENT_MOUNTPOINTS list will affect how the load scales up in the SUT – be sure to order this list to avoid artificial bottlenecks. For example, if the SUT has many load generators and many file systems accessible by all load generators, it may be desirable to order the CLIENT_MOUNTPOINTS list so that a unique load generator and unique file system is used with each CLIENT_MOUNTPOINTS entry until re-use is required to exhaust all possible combinations. By spreading the load as evenly as possible, it is possible to avoid one load generator or one file system becoming a hot-spot.

The ideal way to order the CLIENT_MOUNTPOINTS list for a given SUT will depend heavily on its architecture.

- **EXEC_PATH**
  The full path to the SPEC SFS 2014 executable. Currently the executable is called *netmist* for POSIX systems and *netmist.exe* for Windows systems.

- **USER**
  The user account name, which must be configured on all clients, to be used for the benchmark execution. To specify a domain, prefix the user with the domain name, separated by a backslash. E.g. DOMAIN\User33

- **WARMUP_TIME**
  The amount of time, in seconds, that the benchmark will spend in WARMUP before initiating the measurement ("RUN") phase. The minimum for publishable submissions is 300 seconds (five minutes). The maximum value for a publishable submission is 604,800 seconds (one week). If this value is modified, the value used must be noted in the Other Solution Notes (otherSutNotes) field of the disclosure.

- **IPV6_ENABLE**
  Set to "1" or "Yes" when the benchmark should use IPv6 to communicate with other benchmark processes.

- **PRIME_MON_SCRIPT**
  The 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 and their actions must comply with the SPEC SFS® 2014 SP2 Run and Reporting Rules. If this option is used, the executable or script used must be disclosed. If this option is used, the script must be noted in the Other Solution Notes (otherSutNotes) field of the disclosure. Scripts may also be attached as config diagrams so they are not inline with the submission text. If doing this, the script attached as a config diagram must be referenced from the Other Solution Notes (otherSutNotes) field.
  An example of how to write a script to be invoked by PRIME_MON_SCRIPT, see the sfs_ext_mon example script included with the benchmark.

- **PRIME_MON_ARGS**
  Arguments which are passed to the executable specified in PRIME_MON_SCRIPT. If this option is used, the values used must be noted in the Other Solution Notes (otherSutNotes) field of the disclosure. Each argument passed via PRIME_MON_ARGS appears in a separate command line argument to PRIME_MON_SCRIPT – there is no escaping, etc. to encapsulate all PRIME_MON_ARGS into one argument to PRIME_MON_SCRIPT.

- **NETMIST_LOGS**
  Set the path to the directory in which to store log files from the load generators. The same path will be used on all clients. If this path is not set, /tmp/ or c:\tmp\ will be used.

- **PASSWORD**
  The password for the user specified in USER. (Only applicable when running on Windows platforms)

1.4 Installing the license key

- Obtain your license number from SPEC. Create the ***netmist_license_key*** file in either /tmp or in the current working directory where you will run the benchmark. This file should be a simple text file that contains:
  LICENSE KEY #####

Where the ##### is the license number that you have received from SPEC.

You must create this file before running the benchmark.

1.5 Configuring the storage solution for testing

- Mount all working directories on the clients (POSIX only). The path names must match the values specified in the CLIENT_MOUNTPOINTS parameter in the SPEC SFS 2014 configuration file. Mapping the shares is not needed if running on Windows and using UNC paths.
- Ensure the exported file systems have read/write permissions.
- Ensure access is permitted for username, password, and domain. (SMB testing only)

1.6 Starting the benchmark

Note that the SfsManager must be run under the same user id (UID) on the all of the clients, including the prime client.

- Change directories to the *destination_directory* specified during install.
- On the Prime client:
  - Enter 'python SfsManager -r sfs_config_file -s output_files_suffix

1.7 Monitoring the benchmark execution

- On the Prime client, change directories to the *destination_directory* from the installation step above, then 'cd result'

The user may now examine the benchmark logs, as well as the results. As the benchmark runs, the results are stored in the files with names like:

sfssum_*.{txt,xml}	Summary file used in the submission process described later.
sfslog_*.log	Log file of the current activity.

After all load points are complete, the results from each client are collected into the result directory on prime client. The client logs are files with names like: sfsc001.*

sfsc*.*	The client log files.

More detailed client logs can be found on each client in the path specified by the NETMIST_LOGS RC file parameter, or /tmp/ or C:\tmp\ by default. It is recommended that these log files be purged from each client between each run of the benchmark – you may wish to save these with the other log files from the run before deleting them.

1.8 Examining the results after the benchmark execution has completed

The results of the benchmark are summarized in the sfssum.* files in the result directory on the prime client. These may be examined with any text editing software package. The sfssum_*.xml is the XML summary file that will be used for the submission process, described later in this document.

2 Introduction

The SPEC SFS 2014 benchmark is the latest version of the Standard Performance Evaluation Corporation benchmark that measures storage solution throughput and response time. It provides a standardized method for comparing performance across different vendor platforms.

This document specifies how the SPEC SFS 2014 benchmark is to be run for measuring and publicly reporting performance results, and includes a guide to using the SFS tools. The SPEC SFS® 2014 SP2 Run and Reporting Rules (included in a separate companion document that is included in the SPEC SFS 2014 distribution) have been established by the SPEC Storage 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. Per the SPEC license agreement, all results publicly disclosed must adhere to these Run and Reporting Rules.

SPEC requires that any public use of results from this benchmark follow the SPEC OSG Fair Use Policy at https://www.spec.org/fairuse.html. In the case where it appears that these guidelines have not been adhered to, SPEC may investigate and request that the published material be corrected.

The initial SPEC SFS 2014 release of the benchmark included major workload and functionality changes, as well as clarification of run rules. The code changes were NOT performance neutral, therefore **comparing SPEC SFS 2014 results with results from previous major versions of the SFS benchmark (e.g. SFS 2008, SFS 97_R1) is NOT allowed**.

The SPEC SFS 2014 SP2 benchmark release includes existing workloads and one new workload. For the existing workloads (DATABASE, SWBUILD, VDA, VDI), the code changes between SP2 and SP1 are performance-neutral. This release also adds the new EDA workload.

2.1 What is the SPEC SFS 2014 Benchmark

The SPEC SFS 2014 benchmark is used to measure the maximum sustainable throughput that a storage solution can deliver. The benchmark is protocol independent. It will run over any version of NFS or SMB, clustered file systems, object oriented file systems, local file systems, or any other POSIX compatible file system. Because this tool runs at the application system call level, it is file system type agnostic. This provides strong portability across operating systems, and storage solutions. The SPEC SFS 2014 benchmark already runs on Linux, Windows Vista, Windows 7, Windows 8, Windows 10, Windows Server 2008, Windows Server 2012, Windows Server 2016, Mac OS X, BSD, Solaris, and AIX, and can be used to test any of the file system types that these systems offer.

The SPEC SFS 2014 workloads are a mixture of file meta-data and data oriented operations. The SPEC SFS 2014 benchmark is fully multi-client aware, and is a distributed application that coordinates and conducts the testing across all of the client nodes that are used to test a storage solution.

The benchmark runs on a group of workstations and measures the performance of the storage solution that is providing files to the application layer on the workstations. Each workload consists of several typical file operations. The following is the current set of operations that are measured.

read()	Read file data sequentially
read_file()	Read an entire file sequentially
mmap_read()	Read file data using the mmap() API.
read_random()	Read file data at random offsets in the files.
write()	Write file data sequentially

write_file()	Write an entire file sequentially.
mmap_write()	Write a file using the mmap() API.
write_random()	Write file data at random offsets in the files.
rmw()	Read+modify+write file data at random offsets in files.
mkdir()	Create a directory
rmdir()	Removes a directory
unlink()	Unlink/remove an empty file.
unlink2()	Unlink/remove a non-empty file.
append()	Append to the end of an existing file.
lock()	Lock a file.
unlock()	Unlock a file.
access()	Perform the access() system call on a file.
stat()	Perform the stat() system call on a file.
chmod()	Perform the chmod() system call on a file.
create()	Create a new file.
readdir()	Perform a readdir() system call on a directory.
statfs()	Perform the statfs() system call on a filesystem.
copyfile()	Copy a file.
rename()	Rename a file
pathconf()	Perform the pathconf() system call.

The read() and write() operations are performing sequential I/O to the data files. The read_random() and write_random() perform I/O at random offsets within the files. The read_file and write_file operate in whole files. Rmw is a read_modify_write operation.

The results of the benchmark are:

1. Maximum workload-specific Business Metric achieved.
2. Aggregate Ops/sec that the storage solution can sustain at requested or peak load.
3. Average file operation latency in milliseconds.
4. Aggregate KiB/sec that the storage solution can sustain at requested or peak load.

The SPEC SFS 2014 benchmark includes multiple workloads:

DATABASE	Transactional SQL database
EDA	Electronic design automation
SWBUILD	Software build
VDA	Video data acquisition
VDI	Virtual desktop infrastructure

The user has the option to submit results using any or all of the above workloads.
The SfsManager accepts benchmark run configuration information, starts benchmark execution, and collects results from the SPEC SFS 2014 benchmark.

2.2 SPEC SFS 2014 Benchmark Overview

The SPEC SFS 2014 benchmark is used to test the performance capability of storage solutions. Performance is measured in metrics related to the specific workload tested.

In a typical SPEC SFS test configuration, a series of load generating clients are directed through a network at file systems shared or exported from a file server. The SUT consists of the load generators, the network, the file server, and finally the stable backend storage. The SUT is the entire solution under test. More details of the SUT may be found in section 6.4 of the SPEC SFS® 2014 SP2 Run and Reporting Rules document.

2.2.1 Benchmark Manager

2.2.1.1 Introduction

The benchmark manager is called the SfsManager. It is a Python program that requires Python version 2.7 or higher. It is recommended to get Python from
http://www.python.org/

You can get the syntax by running "python SfsManager -h"

```
Usage: python SfsManager [options]

Command line option:
Required for Benchmark Execution:
[-r <file>] or [--rc-file=<file>]..... Specify rc file
Required for Benchmark Installation:
[--install-dir=<directory>]..... Specify an installation directory (must not exist)
Optional:
[-s <suffix>] or [--suffix=<suffix>]..... Suffix to be used in log and summary files
(default=sfs2014_SP2)
[-b <file>] or [--benchmark-file=<file>]..... benchmark definition file
[-d <dir>] or [--results-dir=<dir>]..... Results directory, use full path
[-i] or [--ignore-override]..... Bypass override of official workload parameters
[-e <filename>] or [--export=<filename>]..... Export workload definitions to a file
[-a]..... Auto mode (max): finds maximum passing load value
[-A <scaling>]..... Auto mode (curve): generate 10 point curve based on result of -a
(if used) or LOAD
[-A] (continued)..... <scaling> is a percentage between 0-100 to scale the maximum
LOAD.
[-A] (continued)..... If -A and -a are used together a '<suffix>.auto' is used for
finding the maximum.
[--save-config=<file>]..... Save the token config file and executeable command line args
[--test-only]..... Simulate a set of load points without actually running.
[-h]..... Show usage info
[-v]..... Show version number
[--debug]..... Detailed output
```

The only required parameter is an rc file. The -a and -A options to SfsManager are not valid for publication runs. These options are included as a convenience only. The base directory has an example called sfs_rc, whose contents are

```
########################################################################################
#
#      sfs_rc
#
# Specify netmist parameters for generic runs in this file.
#
# The following parameters are configurable within the SFS run and
# reporting rules.
#
# Official BENCHMARK values are
#      -SWBUILD
#      -VDA
#      -VDI
```

```
#      -DATABASE
#      -EDA
#
################################################################################
BENCHMARK=VDI
LOAD=10
INCR_LOAD=10
NUM_RUNS=10
CLIENT_MOUNTPOINTS=
EXEC_PATH=/usr/local/bin/netmist
USER=root
WARMUP_TIME=300
IPV6_ENABLE=0
PRIME_MON_SCRIPT=
PRIME_MON_ARGS=
NETMIST_LOGS=
INIT_RATE=0
################################################################################
#
# Specifying a password is only required for Windows clients
#
################################################################################
PASSWORD=
################################################################################
#
#  DO NOT EDIT BELOW THIS LINE FOR AN OFFICIAL BENCHMARK SUBMISSION
#
#  Constraints and overrides on the values below this line can be found in the
#  benchmark XML file (default is benchmarks.xml).  To bypass all overrides
#  use the --ignore-overrides flag in SfsManager.  Using the flag will make
#  the results invalid for formal submission.
#
################################################################################
RUNTIME=300
WORKLOAD_FILE=
OPRATE_MULTIPLIER=
CLIENT_MEM=1g
AGGR_CAP=1g
FILE_SIZE=
DIR_COUNT=10
FILES_PER_DIR=100
UNLINK_FILES=0
LATENCY_GRAPH=1
HEARTBEAT_NOTIFICATIONS=1
DISABLE_FSYNCS=0
USE_RSHRCP=0
BYTE_OFFSET=0
MAX_FD=
PIT_SERVER=
PIT_PORT=
LOCAL_ONLY=0
FILE_ACCESS_LIST=0
SHARING_MODE=0
SOCK_DEBUG=0
```

TRACEDEBUG=0
NO_OP_VALIDATE=0
NO_SHARED_BUCKETS=0
UNLINK2_NO_RECREATE=0

Some things to keep in mind:

- You never need single or double quotes around anything
- The only parameters that must be explicitly specified are BENCHMARK, LOAD, CLIENT_MOUNTPOINTS, USER, EXEC_PATH, and PASSWORD (Windows only)
- The binary parameters all get translated to 1 or 0, but you can also use "yes", "no", "Y", "N", "on", or "off"

2.2.1.2 The Summary Files

Each run will produce 2 summary files: sfssum_<suffix>.txt and sfssum_<suffix>.xml. The txt file is a human readable summary file with the following fields

Field	Value
1	Business metric (may be blank for custom workloads)
2	Requested op rate (blank if not set)
3	Achieved op rate in Ops/s
4	Average latency in milliseconds
5	Total throughput in KiB/s
6	Read Throughput in KiB/s
7	Write Throughput in KiB/s
8	Run time in seconds
9	# of clients
10	# of procs per client
11	Average file size in KiB
12	Client data set size in MiB
13	Total starting data set size in MiB
14	Total initial file set space in MiB
15	Max file space in MiB
16	Workload name
17	Run validity field (blank if valid or no validation criteria exist in custom workload)

2.2.1.3 Example 1: Official Benchmark Run - VDI

In this example there are 2 clients, client1 and client2, each with a single mountpoint /mnt. The user starts with 50 DESKTOPS and generates 10 load points while incrementing the number of DESKTOPS by 50 for each load point. The RC file, vdi_rc, contains the following

Contents of vdi_rc

```
BENCHMARK=VDI
LOAD=50
INCR_LOAD=50
NUM_RUNS=10
CLIENT_MOUNTPOINTS=client1:/mnt client2:/mnt
EXEC_PATH=/usr/local/bin/netmist
USERNAME=sfsuser
```

From the prime client, which may or may not be one of the 2 clients listed in CLIENT_MOUNTPOINTS, the user starts the benchmark with:

```
[prime client]$ python SfsManager -r vdi_rc -s vditest
```

After the benchmark completes, the results directory will contain the following files:

File	Description
sfslog_vditest.log	Overall log file
sfssum_vditest.txt	Summary file in text format
sfssum_vditest.xml	Summary file in XML format
sfsc001.vditest	Detailed results for client1
sfsc002.vditest	Detailed results for client2

2.2.2 Database (DATABASE) Benchmark

2.2.2.1 Workload description

This workload represents the typical behavior of a transactional SQL database. The complete workload is a mixture of DB_TABLE and DB_LOG workloads. The DB_TABLE workload is the database component, and DB_LOG represents the log writer component of a database operation.

2.2.2.2 Workload characteristics

2.2.2.2.1 DATABASE File Operation Distribution

2.2.2.2.1.1 DB_TABLE File Operation Distribution

File Operation Distribution	Operation	%	Operation	%
	read	1	read file	0
	mmap read	0	rand read	79
	write	0	write file	0
	mmap write	0	rand write	20
	rmw	0	append	0
	mkdir	0	rmdir	0
	readdir	0	create	0
	unlink	0	unlink2	0
	stat	0	access	0
	rename	0	copyfile	0
	locking	0	chmod	0
	statfs	0	pathconf	0

2.2.2.2.1.2 DB_LOG File Operation Distribution

File Operation Distribution	Operation	%	Operation	%
	read	0	read file	0
	mmap read	0	rand read	0
	write	80	write file	0
	mmap write	0	rand write	20
	rmw	0	append	0
	mkdir	0	rmdir	0
	readdir	0	create	0
	unlink	0	unlink2	0
	stat	0	access	0
	rename	0	copyfile	0
	locking	0	chmod	0
	statfs	0	pathconf	0

2.2.2.2.2 DATABASE Read Transfer Size Distribution

2.2.2.2.2.1 DB_TABLE Read Transfer Size Distribution

Read Transfer Size Distribution	Slot	Start	End	%
	0	1	511	0
	1	512	1023	0
	2	1024	2047	0
	3	2048	4095	0
	4	4096	4096	0
	5	4097	8191	0
	6	8192	8192	99
	7	8193	16383	0
	8	16384	16384	0
	9	16385	32767	0
	10	32768	32768	0
	11	65536	65536	0
	12	98304	98304	0
	13	131072	131072	0
	14	262144	262144	0
	15	1048576	1048576	1

2.2.2.2.2.2 DB_LOG Read Transfer Size Distribution

Read Transfer Size Distribution	Slot	Start	End	%
	0	1	511	0
	1	512	1023	0
	2	1024	2047	0
	3	2048	4095	0
	4	4096	4096	0
	5	4097	8191	0
	6	8192	8192	99
	7	8193	16383	0
	8	16384	16384	0
	9	16385	32767	0
	10	32768	32768	0
	11	65536	65536	0
	12	98304	98304	0
	13	131072	131072	0
	14	262144	262144	0
	15	1048576	1048576	1

2.2.2.2.3 DATABASE Write Transfer Size Distribution

2.2.2.2.3.1 DB_TABLE Write Transfer Size Distribution

Write Transfer Size Distribution	Slot	Start	End	%
	0	512	512	0
	1	1024	1024	0
	2	1536	1536	0
	3	2048	2048	0
	4	2560	2560	0
	5	3072	3072	0
	6	3584	3584	0
	7	4096	4096	0
	8	4608	4608	0
	9	5120	5120	0
	10	8192	8192	100
	11	12288	12288	0
	12	16384	16384	0
	13	20480	20480	0
	14	24576	24576	0
	15	32768	32768	0

2.2.2.2.3.2 DB_LOG Write Transfer Size Distribution

Write Transfer Size Distribution	Slot	Start	End	%
	0	512	512	5
	1	1024	1024	5
	2	1536	1536	5
	3	2048	2048	5
	4	2560	2560	5
	5	3072	3072	5
	6	3584	3584	5
	7	4096	4096	5
	8	4608	4608	5
	9	5120	5120	5
	10	8192	8192	10
	11	12288	12288	10
	12	16384	16384	10
	13	20480	20480	10
	14	24576	24576	10
	15	32768	32768	0

2.2.2.2.4 DATABASE Miscellaneous Characteristics

2.2.2.2.4.1 DB_TABLE Miscellaneous Characteristics

	Option	Value	Option	Value
Miscellaneous	write commit %	100	background	0
	% direct	100	sharemode	1
	% osync	0	uniform size dist	1
	% notification	0	init rate throttle	0
	LRU	0	init read flag	1
	release version	1		

2.2.2.2.4.2 DB_LOG Miscellaneous Characteristics

	Option	Value	Option	Value
Miscellaneous	write commit %	100	background	0
	% direct	100	sharemode	1
	% osync	0	uniform size dist	1
	% notification	0	init rate throttle	0
	LRU	0	init read flag	1
	release version	1		

2.2.2.2.5 DATABASE Access Pattern Characteristics

2.2.2.2.5.1 DB_TABLE Access Pattern Characteristics

	Option	Value	Option	Value
Access Patterns	rand dist behavior	2	% per spot	5
	min acc per spot	1000	access mult spot	5
	affinity %	20	spot shape	0
	geometric %	2	align	0

2.2.2.2.5.2 DB_LOG Access Pattern Characteristics

	Option	Value	Option	Value
Access Patterns	rand dist behavior	2	% per spot	5
	min acc per spot	1000	access mult spot	5
	affinity %	20	spot shape	0
	geometric %	2	align	0

2.2.2.2.6 DATABASE Content Pattern Characteristics

2.2.2.2.6.1 DB_TABLE Access Pattern Characteristics

Content Patterns	Option	Value	Option	Value
	dedup %	0	dedup within %	0
	dedup across %	0	dedup group count	1
	dedup granule size	4096	dedup gran rep limit	100
	compress %	50	comp granule size	8192
	cipher flag	0	pattern version	1

2.2.2.2.6.2 DB_LOG Access Pattern Characteristics

Content Patterns	Option	Value	Option	Value
	dedup %	0	dedup within %	0
	dedup across %	0	dedup group count	1
	dedup granule size	4096	dedup gran rep limit	100
	compress %	50	comp granule size	8192
	cipher flag	0	pattern version	1

2.2.2.2.7 DATABASE Execution Parameters

2.2.2.2.7.1 DB_TABLE Execution Parameters

Execution Parameters	Parameter	Value	Parameter	Value
	Procs	10	Dirs per proc	1
	Oprate per proc	16	Files per dir	5
	Avg file size			200 MiB

2.2.2.2.7.2 DB_LOG Execution Parameters

Execution Parameters	Parameter	Value	Parameter	Value
	Procs	1	Dirs per proc	1
	Oprate per proc	32	Files per dir	5
	Avg file size			200 MiB

2.2.2.2.8 DATABASE Validation Criteria

Threshold	Value
Per proc oprate	>= 75% of requested
Global average oprate	>= 95% of requested
Per proc maximum acceptable latency	N/A
Global average maximum acceptable latency	N/A
Workload oprate ratio variance	<= +/- 5%

2.2.3 Electronic Design Automation (EDA) Benchmark

2.2.3.1 Workload description

This workload represents the typical behavior of a mixture of EDA applications. The complete workload is a mixture of EDA_FRONTEND and EDA_BACKEND workloads. The EDA_FRONTEND workload is the EDA frontend processing applications, and EDA_BACKEND represents the EDA backend applications that generate the final output files.

2.2.3.2 Workload characteristics

2.2.3.2.1 EDA File Operation Distribution

2.2.3.2.1.1 EDA_FRONTEND File Operation Distribution

File Operation Distribution	Operation	%	Operation	%
	read	0	read file	7
	mmap read	0	rand read	8
	write	0	write file	10
	mmap write	0	rand write	15
	rmw	0	append	0
	mkdir	1	rmdir	0
	readdir	0	create	2
	unlink	1	unlink2	1
	stat	39	access	15
	rename	0	copyfile	0
	locking	0	chmod	1
	statfs	0	pathconf	0

2.2.3.2.1.2 EDA_BACKEND File Operation Distribution

File Operation Distribution	Operation	%	Operation	%
	read	50	read file	0
	mmap read	0	rand read	0
	write	50	write file	0
	mmap write	0	rand write	0
	rmw	0	append	0
	mkdir	0	rmdir	0
	readdir	0	create	0
	unlink	0	unlink2	0
	stat	0	access	0
	rename	0	copyfile	0
	locking	0	chmod	0
	statfs	0	pathconf	0

2.2.3.2.2 EDA Read Transfer Size Distribution

2.2.3.2.2.1 EDA_FRONTEND Read Transfer Size Distribution

Read Transfer Size Distribution	Slot	Start	End	%
	0	1	511	4
	1	512	1023	0
	2	1024	2047	0
	3	2048	4095	2
	4	4096	8191	43
	5	8192	16383	30
	6	16384	32767	21
	7	32768	65535	0
	8	65536	65536	0
	9	131072	131072	0
	10	1	1	0
	11	1	1	0
	12	1	1	0
	13	1	1	0
	14	1	1	0
	15	1	1	0

2.2.3.2.2.2 EDA_BACKEND Read Transfer Size Distribution

Read Transfer Size Distribution	Slot	Start	End	%
	0	1	511	0
	1	512	1023	0
	2	1024	2047	0
	3	2048	4095	0
	4	4096	8191	0
	5	8192	16383	0
	6	16384	32767	0
	7	32768	65535	49
	8	65536	65536	51
	9	131072	131072	0
	10	1	1	0
	11	1	1	0
	12	1	1	0
	13	1	1	0
	14	1	1	0
	15	1	1	0

2.2.3.2.3 EDA Write Transfer Size Distribution

2.2.3.2.3.1 EDA_FRONTEND Write Transfer Size Distribution

Write Transfer Size Distribution	Slot	Start	End	%
	0	1	511	25
	1	512	1023	10
	2	1024	2047	15
	3	2048	4095	18
	4	4096	8191	27
	5	8292	16383	3
	6	16384	32767	2
	7	32768	65535	0
	8	65536	65536	0
	9	131072	131072	0
	10	1	1	0
	11	1	1	0
	12	1	1	0
	13	1	1	0
	14	1	1	0
	15	1	1	0

2.2.3.2.3.2 EDA_BACKEND Write Transfer Size Distribution

Write Transfer Size Distribution	Slot	Start	End	%
	0	1	511	0
	1	512	1023	0
	2	1024	2047	0
	3	2048	4095	0
	4	4096	8191	0
	5	8292	16383	0
	6	16384	32767	0
	7	32768	65535	45
	8	65536	131072	55
	9	131072	131072	0
	10	1	1	0
	11	1	1	0
	12	1	1	0
	13	1	1	0
	14	1	1	0
	15	1	1	0

2.2.3.2.4 EDA Miscellaneous Characteristics

2.2.3.2.4.1 EDA_FRONTEND Miscellaneous Characteristics

	Option	Value	Option	Value
Miscellaneous	write commit %	15	background	0
	% direct	0	sharemode	0
	% osync	0	uniform size dist	0
	% notification	0	init rate throttle	1
	LRU	1	init read flag	0
	release version	2		

2.2.3.2.4.2 EDA_BACKEND Miscellaneous Characcteristics

	Option	Value	Option	Value
Miscellaneous	write commit %	15	background	0
	% direct	50	sharemode	0
	% osync	5	uniform size dist	0
	% notification	0	init rate throttle	1
	LRU	1	init read flag	0
	release version	2		

2.2.3.2.5 EDA Access Pattern Characteristics

2.2.3.2.5.1 EDA_FRONTEND Access Pattern Characteristics

	Option	Value	Option	Value
Access Patterns	rand dist behavior	0	% per spot	0
	min acc per spot	0	access mult spot	5
	affinity %	0	spot shape	0
	geometric %	50	align	0

2.2.3.2.5.2 EDA_BACKEND Access Pattern Characteristics

	Option	Value	Option	Value
Access Patterns	rand dist behavior	0	% per spot	0
	min acc per spot	0	access mult spot	5
	affinity %	0	spot shape	0
	geometric %	50	align	0

2.2.3.2.6 EDA Content Pattern Characteristics

2.2.3.2.6.1 EDA_FRONTEND Content Pattern Characteristics

Content Patterns	Option	Value	Option	Value
	dedup %	50	dedup within %	0
	dedup across %	0	dedup group count	1
	dedup granule size	4096	dedup gran rep limit	100
	compress %	50	comp granule size	8192
	cipher flag	0	pattern version	2

2.2.3.2.6.2 EDA_BACKEND Content Pattern Characteristics

Content Patterns	Option	Value	Option	Value
	dedup %	40	dedup within %	0
	dedup across %	0	dedup group count	1
	dedup granule size	4096	dedup gran rep limit	100
	compress %	20	comp granule size	8192
	cipher flag	0	pattern version	2

2.2.3.2.7 EDA Execution Parameters

2.2.3.2.7.1 EDA_FRONTEND Execution Parameters

Execution Parameters	Parameter	Value	Parameter	Value
	Procs	3	Dirs per proc	10
	Oprate per proc	100	Files per dir	10
	Avg file size			16 KiB

2.2.3.2.7.2 EDA_BACKEND Execution Parameters

Execution Parameters	Parameter	Value	Parameter	Value
	Procs	2	Dirs per proc	5
	Oprate per proc	75	Files per dir	10
	Avg file size			10 MiB

2.2.3.2.8 EDA Validation Criteria

Threshold	Value
Per proc oprate	>= 75% of requested
Global average oprate	>= 95% of requested
Per proc maximum acceptable latency	N/A
Global average maximum acceptable latency	N/A
Workload oprate ratio variance	<= +/- 5%

2.2.4 Software Build (SWBUILD) Benchmark

2.2.4.1 Benchmark Description

The software build type workload is a classic meta-data intensive build workload. This workload was derived from analysis of software builds, and traces collected on systems in the software build arena. Conceptually, these tests are similar to running unix 'make' against several tens of thousands of files. The file attributes are checked (metadata operations) and if necessary, the file is read, compiled, then data is written back out to storage.

2.2.4.1.1 SWBUILD File Operation Distribution

File Operation Distribution	Operation	%	Operation	%
	read	0	read file	6
	mmap read	0	rand read	0
	write	0	write file	7
	mmap write	0	rand write	0
	rmw	0	append	0
	mkdir	1	rmdir	0
	readdir	2	create	1
	unlink	2	unlink2	0
	stat	70	access	6
	rename	0	copyfile	0
	locking	0	chmod	5
	statfs	0	pathconf	0

2.2.4.1.2 SWBUILD Read Transfer Size Distribution

Read Transfer Size Distribution	Slot	Start	End	%
	0	1	511	1
	1	512	1023	5
	2	1024	2047	7
	3	2048	4095	7
	4	4096	4096	0
	5	4096	8191	45
	6	8192	8192	0
	7	8192	16383	13
	8	16384	16384	0
	9	16384	32767	3
	10	32768	65535	2
	11	65536	65536	0
	12	98304	98304	0
	13	65536	131072	17
	14	262144	262144	0
	15	524288	524288	0

2.2.4.1.3 SWBUILD Write Transfer Size Distribution

Write Transfer Size Distribution	Slot	Start	End	%
	0	1	511	5
	1	512	1023	3
	2	1024	2047	10
	3	2048	4095	15
	4	4096	4096	0
	5	4096	8191	14
	6	8192	8192	0
	7	8192	16383	7
	8	16384	16384	0
	9	16384	32767	6
	10	32768	65535	4
	11	65536	131072	36
	12	98304	98304	0
	13	131072	131072	0
	14	262144	262144	0
	15	524288	524288	0

2.2.4.1.4 SWBUILD Miscellaneous Characteristics

	Option	Value	Option	Value
Miscellaneous	write commit %	33	background	0
	% direct	0	sharemode	0
	% osync	0	uniform size dist	0
	% notification	0	init rate throttle	0
	LRU	0	init read flag	1
	release version	1		

2.2.4.1.5 SWBUILD Access Pattern Characteristics

	Option	Value	Option	Value
Access Patterns	rand dist behavior	0	% per spot	0
	min acc per spot	0	access mult spot	5
	affinity %	0	spot shape	0
	geometric %	10	align	0

2.2.4.1.6 SWBUILD Content Pattern Characteristics

	Option	Value	Option	Value
Content Patterns	dedup %	0	dedup within %	0
	dedup across %	0	dedup group count	1
	dedup granule size	4096	dedup gran rep limit	100
	compress %	80	comp granule size	8192
	cipher flag	0	pattern version	1

2.2.4.1.7 SWBUILD Execution Parameters

	Parameter	Value	Parameter	Value
Execution Parameters	Procs	5	Dirs per proc	50
	Oprate per proc	100	Files per dir	100
	Avg file size			16 KiB

2.2.4.1.8 SWBUILD Validation Criteria

Threshold	Value
Per proc oprate	>= 75% of requested
Global average oprate	>= 95% of requested
Per proc maximum acceptable latency	N/A
Global average maximum acceptable latency	N/A
Workload oprate ratio variance	N/A

2.2.5 Video Data Acquisition (VDA) Benchmark

2.2.5.1 Workload description

The workload generally simulates applications that store data acquired from a temporally volatile source (e.g. surveillance cameras). A stream refers to an instance of the application storing data from a single source (e.g. one video feed). The storage admin is concerned primarily about maintaining a minimum fixed bit rate per stream and secondarily about maintaining the fidelity of the stream. The goal of the storage admin is to provide as many simultaneous streams as possible while meeting the bit rate and fidelity constraints.
The business metric for the benchmark is STREAMS. The benchmark consists of two workload objects: VDA1 (data stream) and VDA2 (companion applications). Each stream corresponds to a roughly 36 Mb/s bit rate, which is in the upper range of high definition video.

2.2.5.2 Workload characteristics

2.2.5.2.1 VDA File Operation Distribution

2.2.5.2.1.1 VDA1 File Operation Distribution

File Operation Distribution	Operation	%	Operation	%
	read	0	read file	0
	mmap read	0	rand read	0
	write	100	write file	0
	mmap write	0	rand write	0
	rmw	0	append	0
	mkdir	0	rmdir	0
	readdir	0	create	0
	unlink	0	unlink2	0
	stat	0	access	0
	rename	0	copyfile	0
	locking	0	chmod	0
	statfs	0	pathconf	0

2.2.5.2.1.2 VDA2 File Operation Distribution

File Operation Distribution	Operation	%	Operation	%
	read	5	read file	0
	mmap read	0	rand read	84
	write	0	write file	0
	mmap write	0	rand write	0
	rmw	2	append	0
	mkdir	0	rmdir	0
	readdir	3	create	1
	unlink	1	unlink2	0
	stat	2	access	2
	rename	0	copyfile	0
	locking	0	chmod	0
	statfs	0	pathconf	0

2.2.5.2.2 VDA Read Transfer Size Distribution

2.2.5.2.2.1 VDA1 Read Transfer Size Distribution

Read Transfer Size Distribution	Slot	Start	End	%
	0	1	511	0
	1	512	1023	0
	2	1024	2047	0
	3	2048	4095	0
	4	4096	4096	0
	5	4097	8191	0
	6	8192	8192	0
	7	8193	16383	0
	8	16384	16384	0
	9	16385	32767	0
	10	32768	32768	0
	11	65536	65536	15
	12	131072	131072	10
	13	262144	262144	20
	14	524288	524288	35
	15	1048576	1048576	20

2.2.5.2.2.2 VDA2 Read Transfer Size Distribution

Read Transfer Size Distribution	Slot	Start	End	%
	0	1	511	0
	1	512	1023	0
	2	1024	2047	0
	3	2048	4095	0
	4	4096	4096	0
	5	4097	8191	0
	6	8192	8192	0
	7	8193	16383	0
	8	16384	16384	0
	9	16385	32767	0
	10	32768	32768	0
	11	65536	65536	15
	12	131072	131072	10
	13	262144	262144	20
	14	524288	524288	35
	15	1048576	1048576	20

2.2.5.2.3 VDA Write Transfer Size Distribution

2.2.5.2.3.1 VDA1 Write Transfer Size Distribution

Write Transfer Size Distribution	Slot	Start	End	%
	0	1	511	0
	1	512	1023	0
	2	1024	2047	0
	3	2048	4095	0
	4	4096	4096	0
	5	4097	8191	0
	6	8192	8192	0
	7	8193	16383	0
	8	16384	16384	0
	9	16385	32767	0
	10	32768	32768	5
	11	65536	65536	10
	12	131072	131072	10
	13	262144	262144	25
	14	524288	524288	25
	15	1048576	1048576	25

2.2.5.2.3.2 VDA2 Write Transfer Size Distribution

Write Transfer Size Distribution	Slot	Start	End	%
	0	1	511	0
	1	512	1023	0
	2	1024	2047	0
	3	2048	4095	0
	4	4096	4096	0
	5	4097	8191	0
	6	8192	8192	0
	7	8193	16383	0
	8	16384	16384	0
	9	16385	32767	0
	10	32768	32768	5
	11	65536	65536	10
	12	131072	131072	10
	13	262144	262144	25
	14	524288	524288	25
	15	1048576	1048576	25

2.2.5.2.4 VDA Miscellaneous Characteristics

2.2.5.2.4.1 VDA1 Miscellaneous Characteristics

Miscellaneous	Option	Value	Option	Value
	write commit %	5	background	0
	% direct	0	sharemode	0
	% osync	0	uniform size dist	0
	% notification	0	init rate throttle	0
	LRU	0	init read flag	1
	release version	1		

2.2.5.2.4.2 VDA2 Miscellaneous Characteristics

Miscellaneous	Option	Value	Option	Value
	write commit %	0	background	0
	% direct	0	sharemode	0
	% osync	0	uniform size dist	0
	% notification	0	init rate throttle	0
	LRU	0	init read flag	1
	release version	1		

2.2.5.2.5 VDA Access Pattern Characteristics

2.2.5.2.5.1 VDA1 Access Pattern Characteristics

Access Patterns	Option	Value	Option	Value
	rand dist behavior	0	% per spot	0
	min acc per spot	0	access mult spot	5
	affinity %	0	spot shape	0
	geometric %	0	align	0

2.2.5.2.5.2 VDA2 Access Pattern Characteristics

Access Patterns	Option	Value	Option	Value
	rand dist behavior	0	% per spot	0
	min acc per spot	0	access mult spot	5
	affinity %	0	spot shape	0
	geometric %	0	align	0

2.2.5.2.6 VDA Content Pattern Characteristics

2.2.5.2.6.1 VDA1 Content Pattern Characteristics

Content Patterns	Option	Value	Option	Value
	dedup %	0	dedup within %	0
	dedup across %	0	dedup group count	1
	dedup granule size	4096	dedup gran rep limit	100
	compress %	0	comp granule size	8192
	cipher flag	0	pattern version	1

2.2.5.2.6.2 VDA2 Content Pattern Characteristics

Content Patterns	Option	Value	Option	Value
	dedup %	0	dedup within %	0
	dedup across %	0	dedup group count	1
	dedup granule size	4096	dedup gran rep limit	100
	compress %	0	comp granule size	8192
	cipher flag	0	pattern version	1

2.2.5.2.7 VDA Execution Parameters

2.2.5.2.7.1 VDA1 Content Pattern Characteristics

Execution Parameters	Parameter	Value	Parameter	Value
	Procs	1	Dirs per proc	1
	Oprate per proc	9	Files per dir	1
	Avg file size			1 GiB

2.2.5.2.7.2 VDA2 Content Pattern Characteristics

	Parameter	Value	Parameter	Value
Execution Parameters	Procs	1	Dirs per proc	1
	Oprate per proc	1	Files per dir	1
	Avg file size			1 GiB

2.2.5.2.8 VDA Validation Criteria

Threshold	Value
Per proc oprate	>= 75% of requested
Global average oprate	>= 95% of requested
Per proc maximum acceptable latency	N/A
Global average maximum acceptable latency	N/A
Workload oprate ratio variance	<= +/- 5%

2.2.6 Virtual Desktop Infrastructure (VDI) Benchmark

2.2.6.1 Workload description

This workload simulates a steady-state high-intensity knowledge worker in a VDI environment that uses full clones. This workload does not simulate a linked-clone environment. This is the behavior that was seen in traces between the hypervisor and storage when the VM's were running on ESXi, Hyper-V, KVM and Xen environments.

2.2.6.2 Workload characteristics

2.2.6.2.1 VDI File Operation Distribution

	Operation	%	Operation	%
File Operation Distribution	read	6	read file	0
	mmap read	0	rand read	20
	write	9	write file	0
	mmap write	0	rand write	64
	rmw	0	append	0
	mkdir	0	rmdir	0
	readdir	0	create	0
	unlink	0	unlink2	0
	stat	0	access	1
	rename	0	copyfile	0
	locking	0	chmod	0
	statfs	0	pathconf	0

2.2.6.2.2 VDI Read Transfer Distribution

Read Transfer Size Distribution

Slot	Start	End	%
0	512	512	1
1	2048	2048	1
2	2560	3584	1
3	4096	4096	20
4	4608	7680	1
5	8192	8192	4
6	8704	15872	4
7	16384	16384	42
8	16896	32256	3
9	32768	32768	14
10	33280	64024	1
11	65536	65536	6
12	66048	126976	1
13	131072	131072	1
14	262144	262144	0
15	524288	524288	0

2.2.6.2.3 VDI Write Transfer Distribution

Write Transfer Size Distribution

Slot	Start	End	%
0	512	512	21
1	1024	1024	2
2	2048	2048	1
3	4096	4096	47
4	8192	8192	6
5	8704	15360	3
6	16384	16384	5
7	16896	30720	5
8	32768	32768	1
9	35328	64000	3
10	65536	65536	2
11	69632	126976	1
12	131072	131072	3
13	262144	262144	0
14	524288	524288	0
15	1048576	1048576	0

2.2.6.2.4 VDI Miscellaneous Characteristics

	Option	Value	Option	Value
Miscellaneous	write commit %	100	background	0
	% direct	100	sharemode	0
	% osync	0	uniform size dist	0
	% notification	0	init rate throttle	0
	LRU	0	init read flag	1
	release version	1		

2.2.6.2.5 VDI Access Pattern Characteristics

	Option	Value	Option	Value
Access Patterns	rand dist behavior	0	% per spot	0
	min acc per spot	0	access mult spot	5
	affinity %	0	spot shape	0
	geometric %	90	align	0

2.2.6.2.6 VDI Content Pattern Characteristics

	Option	Value	Option	Value
Content Patterns	dedup %	0	dedup within %	0
	dedup across %	0	dedup group count	1
	dedup granule size	4096	dedup gran rep limit	100
	compress %	60	comp granule size	8192
	cipher flag	0	pattern version	1

2.2.6.2.7 VDI Execution Parameters

	Parameter	Value	Parameter	Value
Execution Parameters	Procs	2	Dirs per proc	1
	Oprate per proc	100	Files per dir	1
	Avg file size			500 MiB

2.2.6.2.8 VDI Validation Criteria

Threshold	Value
Per proc oprate	>= 75% of requested
Global average oprate	>= 90% of requested
Per proc maximum acceptable latency	N/A
Global average maximum acceptable latency	N/A
Workload oprate ratio variance	N/A

3 Installing and Configuring the Benchmark Environment

This section provides information on hardware/software configuration requirements for the load generators and the storage solutions. It also includes installation instructions for the benchmark on the load generators for each of the supported operating systems.

3.1 Setting up the Solution Under Test (SUT)

There are several things you must set up on your storage solution before you can successfully execute a benchmark run.

1. Configure enough disk space. You may mount your test disks anywhere in your server's file name space that is convenient for you. The maximum ops/sec a storage solution can process is often limited by the number of independent disk drives configured on the server. In the past, a disk drive could generally sustain on the order of 100-200 NFS or SMB ops/sec. This was only a rule of thumb, and this value will change as new technologies become available. However, you will need to ensure you have sufficient disks configured to sustain the load you intend to measure.
   Space requirements scale with the requested load.
   DATABASE = 24 GiB per DATABASE
   EDA = 11 GiB per JOB_SET
   SWBUILD = 5 GiB per BUILD
   VDA = 24 GiB per STREAM
   VDI = 12 GiB per DESKTOP

2. Initialize (if necessary or desired) and mount all file systems. According to the Run and Reporting Rules, it is not necessary to (re-)initialize the solution under test prior to a benchmark run. However, in the full disclosure report for a benchmark run, any configuration steps or actions taken since the last (re-)initialization must be documented for each component of the solution. Therefore, it may be desirable to re-initialize the solution between runs, depending on the tester's objective. See section 5.2 "Solution File System Creation and Configuration" in the SPEC SFS® 2014 SP2 Run and Reporting Rules for more detail.

3. Export or share all file systems to all clients. This gives the clients permission to mount/map, read, and write to your test storage. The benchmark program will fail without this permission.

4. Verify that all RPC services work. The clients may use port mapping, mount, and NFS services, or Microsoft name services, and file sharing, provided by the server. The benchmark will fail if these services do not work for all clients on all networks. If your client systems have NFS client software installed, one easy way to do this is to attempt mounting one or more of the server's exported file systems on the client. On a Windows client one may try mapping the shares to ensure that the services are correctly configured on the SMB server.

5. Ensure your solution is idle. Any other work being performed by your solution is likely to perturb the measured throughput and response time. The only safe way to make a repeatable measurement is to stop all non-benchmark related processing on your solution during the benchmark run.

6. Ensure that your test network is idle. Any extra traffic on your network will make it difficult to reproduce your results, and will probably make your solution look slower. The easiest thing to do is to have a separate, isolated network for all components that comprise the solution. Results obtained on production networks may 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. **Please do not run this benchmark over a corporate LAN. It can present heavy loads and adversely affect others on the same shared network**.

At this point, your solution should be ready for a benchmark measurement. You must now set up a few things on your client systems so they can run the benchmark programs.

3.2 Setting up the Load Generators

Running the SfsManager requires that the Python 2.7 be installed.

On UNIX systems, create "spec" user. SPEC SFS 2014 benchmark runs should be done as a non-root user. The SPEC SFS 2014 binary must be installed on clients.

To install the SPEC SFS 2014 binary:
On all the clients:

1. Login as "root"
2. Change directory to the top level directory containing the SPEC SFS 2014 benchmark files
3. Enter 'python SfsManager –install-dir="destination_directory"'

Alternately, just copy the binary file to all clients using any feasible method.

Verify that all clients, including prime, have the SFS 2014 binary at the same location with the same name and that this matches the EXEC_PATH parameter in the RC file.

Additional client setup validation:

1. Configure and verify network connectivity between all clients and server. Clients must be able to send IP packets to each other and to the server. How you configure this is system-specific and is not described in this document. Two easy ways to verify network connectivity are to use a "ping" program or the netperf benchmark *(http://www.netperf.org)*.
2. Before starting the benchmark, ensure that the prime client can execute commands on the remote clients using ssh with no password challenges. Refer to Appendix B for an example of how to do this. (On Unix-based systems)
3. Ensure that the file systems specified in the CLIENT_MOUNTPOINTS string are mounted and accessible on all clients. Where possible, it may help to mount all test file systems to a path under a tmpfs directory. This avoids filling local disks if a mount fails. If using Windows clients with UNC paths in CLIENT_MOUNTPOINTS, verify that the clients can access those UNC paths with the USER and PASSWORD specified in the RC file.
4. Clients must have free space for logs generated during the course of the run. In general, logs can vary between 500 KiB to 100 MiB or more. If necessary, a separate log path can be specified in the RC file.

*** IMPORTANT * –** If Windows Firewall is turned on; each program will need to be added to the exceptions list. Either open the Windows Firewall control panel and add the applications manually, or wait for the pop-up to appear after the first execution of each application. Other locally-based firewall applications may require a similar allowance. Note that on Windows systems additional firewall options appear, and may default to enabled, upon joining a domain. After joining a domain, verify that your firewall settings are as expected.

*** IMPORTANT * –** Windows client load generator configurations must have one additional client that is used as the Prime client and this client cannot be used to generate load. This constraint is due to Windows security mechanisms that prevent a client from logging into itself. You may use a single client on non-Windows clients, but it is recommended that the prime client not generate load, which would require at least two clients.

3.2.1 Configuring SPEC SFS 2014 Windows Clients for Auto-Startup

The following are the steps to follow to configure Windows clients in order to allow the Prime Client to communicate with them directly and remotely start the SfsManager process when a benchmark run is started.

Granting DCOM Remote Launch permissions:

1. Click Start, click Run, type DCOMCNFG, and then click OK.
2. In the Component Services dialog box, expand Component Services, expand Computers.
3. Right mouse click on My Computer and select properties.
   The My Computer dialog box appears.
4. In the My Computer dialog box, click the COM Security tab.
5. Under Launch and Activate Permissions, click Edit Limits.
6. In the Launch Permission dialog box, follow these steps if your name or your group does not appear in the Groups or user names list:
   a. In the Launch Permission dialog box, click Add.
   b. In the Select Users, Computers, or Groups dialog box, add your name and the group in the Enter the object names to select box, and then click OK.
7. In the Launch Permission dialog box, select your user and group in the Group or user names box. In the Allow column under Permissions for User, select Remote Launch, and then click OK.

3.3 Configuring Benchmark Parameters

Once you have the clients and server configured, you must set some parameters for the benchmark itself, which you do in a file called the "sfs_rc file". The actual name of the file is a prefix picked by you, and the suffix "_rc". The default version shipped with the benchmark is delivered as "sfs_rc" in the benchmark source directory. One may use any text editor to modify parameters in the rc files.

There are several parameters you must set, and several others you may change to suit your needs while performing a disclosable run. There are also many other parameters you may change which change the benchmark behavior, but lead to a non-disclosable run (for example, turning on the PIT_SERVER). See the SPEC SFS 2014 Run Rules for the classification of all the parameters.

The parameters you must set are:

1. **BENCHMARK:** The name of the workload to run. DATABASE, EDA, SWBUILD, VDA, or VDI.

2. **CLIENT_MOUNTPOINTS**: This parameter specifies the names of the file systems the clients will use when testing the storage solution. The business metric values are spread among the client mount points in the following way. If the number of items N in the CLIENT_MOUNTPOINTS list is greater than the business metric value L (the current value for LOAD), then the first L items from the list are used, one business metric value per client/mountpoint. If L>N, then the N+1 business metric value will wrap around to the beginning of the list and allocation proceeds until all L business metrics have been allocated, wrapping around to the beginning of the list as many times as is necessary.

Examples:

For an NFS configuration:
client_name:/mount_point_path client_name:/mount_point_path

For a SMB configuration: (client_name followed by UNC path)
client_name:\\server\path client_name:\\server\path …

When using an external file: (CLIENT_MOUNTPOINTS=mountpoints.txt), the syntax for each line in the file is "client_name path". Multiple mountpoints for a single load generator can be specified on the same line, separated by spaces. However, because the CLIENT_MOUNTPOINTS list is used in order, this may create artificial bottlenecks if not done carefully. The lines do not need to be unique. For example:

```
client1 /mnt
client1 /mnt
client2 /mnt
client3 /mnt1
client3 /mnt2
```

Reminder: If using Windows load generators, the Prime client must not be listed in the CLIENT_MOUNTPOINTS list.

3. **LOAD**, **INCR_LOAD**, and **NUM_RUNS**: These parameters specify the aggregate load the clients will generate. To test a set of evenly spaced load points, set all three parameters. Set LOAD to the lowest load level, set INCR_LOAD the amount you would like to increase the load for each measured run, and set NUM_RUNS to the number of times you would like to increment the load. This is the easiest way to configure a disclosable run. For example, if you would like to measure 10 evenly spaced points ending at 2000, you would set LOAD to 200, INCR_LOAD to 200 and NUM_RUNS to 10.
   When choosing LOAD values please take into consideration the amount of RAM that these workloads will consume on the client.

DATABASE	=	55 MiB per DATABASE
EDA	=	12 MiB per JOB_SET
SWBUILD	=	400 MiB per BUILD
VDA	=	10 MiB per STREAM
VDI	=	8 MiB per DESKTOP

   Also, please consider the capacity requirements for each LOAD increment.

DATABASE	=	24 GiB per DATABASE
EDA	=	11 GiB per JOB_SET
SWBUILD	=	5 GiB per BUILD
VDA	=	24 GiB per STREAM
VDI	=	12 GiB per DESKTOP

4. **EXEC_PATH**: Set this to the absolute path to the benchmark executable. The same path will be used on all clients, so the executable must be at the same path on all clients.
5. **USER**: Set this to the User ID for launching the benchmark on all clients. (On Windows systems this includes the Domain/User) E.g. DOMAIN\User33
6. **PASSWORD**: Set this to the account password for running the benchmark. (Windows clients only)

3.3.1 Other Variables in the RC File

In addition to the parameters required to be changed for a run, the following parameters are optionally adjustable in the RC file – note that some may not be changed or set for a publishable run:

1. **PRIME_MON_SCRIPT** and **PRIME_MON_ARGS**: This is the name (and argument list) of a program which the SPEC SFS 2014 benchmark will execute during the various phases of the benchmark. This is often used to start some performance measurement program while the benchmark is running to assist with debugging and tuning your system.
   An example monitor script – sfs_ext_mon – is provided in the SPEC SFS 2014 source directory. For a disclosable run, this program/script must be performance neutral and its actions must comply with the SPEC SFS® 2014 SP2 Run and Reporting Rules. If this option is used, the script used, as well as the contents of the PRIME_MON_ARGS parameter, must be disclosed in the Other Solution Notes (otherSutNotes) field. Longer scripts may be attached as a configuration diagram and referenced in the Other Solution Notes (otherSutNotes) field.
2. **WARMUP_TIME**: Sets the duration of the WARMUP phase in seconds. This may be set to a value between 300 seconds and one week (604,800 seconds.)
3. **RUNTIME***: Sets the duration of the RUN (measurement) phase of the benchmark.
4. **WORKLOAD_FILE*:** Used to import custom workload objects.
5. **IPV6_ENABLE:** Flag to set for when the benchmark should use IPv6 to communicate with other benchmark processes.
6. **INIT_RATE:** Set dataset creation rate throttle such that no proc will exceed this rate in MiB/s during the INIT phase
7. **CLIENT_MEM*:** Used to tell the benchmark the amount of RAM in the client. This is used to set the file size the benchmark will use if the FILE_SIZE parameter is not set in the RC or benchmarks.xml file. This parameter has no effect for any publishable workloads.
8. **AGGR_CAP*:** Used to tell the benchmark the maximum aggregate data set size. This is used to set the file size the benchmark will use if the FILE_SIZE parameter is not set in the RC or benchmarks.xml file. This parameter has no effect for any publishable workloads.
9. **FILE_SIZE*:** Used to tell the benchmark to create files of a particular size size. This overrides the effects of CLIENT_MEM and AGGR_CAP, both of which are ignored if this is set. This is overridden by the value set in benchmarks.xml for the requested workload.
10. **DIR_COUNT*:** Used to specify the number of subdirectories at the leaf level. This is overridden by the value set in benchmarks.xml for the requested workload.
11. **FILES_PER_DIR*:** Used to specify the number of files in each leaf subdirectory. This is overridden by the value set in benchmarks.xml for the requested workload.
12. **UNLINK_FILES*:** Specify whether to delete the dataset between load points. This is not recommended for common test cases.
13. **LATENCY_GRAPH*:** Specify whether to or not to enable timers to keep track of latency per op type. Enabling this prints out a table of op type latencies and overall latency histogram for each iteration of the benchmark. If this value is not present in the RC file, the behavior will be enabled by SfsManager.
14. **HEARTBEAT_NOTIFICATIONS*:** Specify whether or not to display heartbeat notifications during all phases of benchmark execution to indicate ongoing test activity. If this value is not present in the RC file, the behavior will be enabled by SfsManager.
15. **DISABLE_FSYNCS*:** Disables fsync() calls in the load generator.
16. **USE_RSHRCP*:** Use RSH instead of SSH to start remote processes on UNIX platforms.
17. **BYTE_OFFSET*:** Over-ride beginning offset of files – adds a fixed-length offset to change the beginning offset of every file.
18. **MAX_FD*:** Sets the maximum number of file descriptors each proc can use.
19. **PIT_SERVER*:** Name of the server running the Programmable Interval Timer server.
20. **PIT_PORT*:** TCP port number to access the PIT on the remote PIT server.
21. **LOCAL_ONLY*:** Use sh instead of ssh/rsh – confines the benchmark to run only on the prime client. This works on UNIX and Windows systems. This option is for benchmark development

testing purposes only and not recommended for use. This option may be deprecated in future releases.

22. **FILE_ACCESS_LIST*:** If enabled, the benchmark will dump a list of all files accessed during each load point.
23. **SHARING_MODE*:** If enabled, the dataset for a single business metric is shared between all procs in that business metric.
24. **SOCK_DEBUG*:** Enables TCP socket debug logging.
25. **TRACEDEBUG*:** Enables tracing of file system operations for each proc.
26. **NO_SHARED_BUCKETS*:** Disable creation of dataset for unused op types and sharing of dataset between op types.
27. **NO_OP_VALIDATE*:** Disable validation that all op types can succeed on file systems under test before each load point.
28. **UNLINK2_NO_RECREATE*:** Suppress recreation of files (with non-zero size) after unlink2.
29. **NETMIST_LOGS:** Used to set a non-default location for the netmist_C*.log files. If this isn't set either /tmp/ or c:\tmp\ will be used.

***This parameter may not be changed or set to a non-default value for a publishable run.**

4 Running the Benchmark and Interpreting Results

This section contains information on the SPEC SFS 2014 benchmark directory structure, running the benchmark, and interpreting the benchmark metrics output generated in the summary results file.

4.1 SFS Benchmark Directory Structure

The following is a quick overview of the benchmark's directory structure. Please note that the variable "$SPEC" used below represents the full path to the install_directory, where the benchmark is installed.

1. $SPEC
   The directory contains the SPEC SFS 2014 benchmark Makefile. The makefile is used to build tools, compile the benchmark source into executables, and to clean directories of all executables. Pre-built binaries are provided for many operating systems, therefore compilation is probably not required. The top-level directory also contains the SfsManager and SpecReport tools as well as the example sfs_rc and sfs_ext_mon files.
2. $SPEC/bin
   The benchmark binaries for the specific environment being used are located in the "$SPEC/bin" directory if the user has built the binaries using the Makefile provided.
3. $SPEC/binaries
   Contains the pre-built binaries for various operating systems.
4. $SPEC/docs
   Contains documentation for the SPEC SFS 2014 benchmark.
5. $SPEC/msbuild
   Contains the Microsoft Visual Studio solution file used to compile the benchmark on Windows.
6. $SPEC/netmist
   Contains the source files for the netmist load generator.
7. $SPEC/redistributable_sources
   This directory contains tools relevant to the execution or analysis of SPEC SFS 2014 benchmark runs licensed under compatible terms.
8. $SPEC/win32lib
   Contains compatibility libraries for building the benchmark under Microsoft Visual Studio.
9. $SPEC/results
   Contains benchmark log and results files created during a benchmark run. This directory is created upon successful start of benchmark execution if it does not exist.

4.2 Pre-Compiled SPEC SFS 2014 Benchmark Binaries

The SPEC SFS 2014 benchmark includes pre-compiled binaries for a large number of supported platforms and architectures. If it is necessary or desired for the user to compile a version of the benchmark source for testing, a generic UNIX makefile is provided in the benchmark top level directory ($SPEC).
The makefile 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. To build the software simply type: make

The Visual Studio solution file is also provided to compile the Windows executables. The solution file is located in the $SPEC/msbuild subdirectory. The SPEC SFS 2014 benchmark can be built with Visual Studio C++ 2015 Express.

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

IBM Corporation
: AIX 7.2

FreeBSD
: FreeBSD 11

Oracle Corporation
: Solaris 11.1, Solaris 11.3

Red Hat, Inc.
: RHEL 6, RHEL 7

Apple Inc.
: Mac OS X

Microsoft Corporation
: Windows 7, Windows 8, Windows 10, Windows Server 2008R2, Windows Server 2012, Windows Server 2012R2, Windows Server 2016

4.3 Using the SFS Manager

The SfsManager is used to run the benchmark. The results obtained from multiple data points within a run are also collected in a form suitable for use with other result formatting tools.

4.3.1 Example of SUT Validation

By default, and during a publication run, the client validates that it can perform all of the POSIX level operations that will be used during the benchmark before starting benchmark execution. If the validation fails, then the benchmark will terminate, with errors collected in the log files.

4.3.2 Example of a Benchmark Run

```
<<< Thu Oct 12 06:57:11 2017: Starting VDI run 1 of 1: DESKTOPS=1 >>

     SPEC SFS2014_SP2 Release         $Revision: 1688 $

     This product contains benchmarks acquired from several sources who
     understand and agree with SPEC's goal of creating fair and objective
     benchmarks to measure computer performance.

     This copyright notice is placed here only to protect SPEC in the
     event the source is misused in any manner that is contrary to the
     spirit, the goals and the intent of SPEC.

     The source code is provided to the user or company under the license
     agreement for the SPEC Benchmark Suite for this product.


    ------------------------------------------------------------------------
    This program contain a 64-bit version of Mersenne Twister pseudorandom
    number generator.

    Copyright (C) 2004, Makoto Matsumoto and Takuji Nishimura,
    All rights reserved.

    Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions
    are met:

    1. Redistributions of source code must retain the above copyright
       notice, this list of conditions and the following disclaimer.
    2. Redistributions in binary form must reproduce the above copyright
       notice, this list of conditions and the following disclaimer in
       the documentation and/or other materials provided with the
       distribution.
```

```
    3. The names of its contributors may not be used to endorse or
       promote products derived from this software without specific prior
       written permission.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
    "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
    LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
    A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT
    OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
    SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
    LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
    DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
    OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
    ------------------------------------------------------------------------
     Test run time = 300 seconds, Warmup = 300 seconds.
     Op latency reporting activated
     Files per directory set to 1
     Directories per proc set to 1
     Using custom file size 51200 Kbytes
     Running 2 copies of the test on 1 clients
     Results directory: /home/capps/Downloads/trunk/bin/results
     Clients have a total of 1024 MiBytes of memory
     Clients have 512 MiBytes of memory size per process
     Clients each have 2 processes
     Adjustable aggregate data set value set to 1024 MiBytes
     Each process file size = 51200 kbytes
     Client data set size       = 1100 MiBytes
     Total starting data set size = 1100 MiBytes
     Total initial file space   = 1100 MiBytes
     Total max file space       = 1200 MiBytes

[INFO][Thu Oct 12 06:57:11 2017]Exec validation successful

     SPEC SFS2014_SP2 Release         $Revision: 1689 $

     This product contains benchmarks acquired from several sources who
     understand and agree with SPEC's goal of creating fair and objective
     benchmarks to measure computer performance.

     This copyright notice is placed here only to protect SPEC in the
     event the source is misused in any manner that is contrary to the
     spirit, the goals and the intent of SPEC.

     The source code is provided to the user or company under the license
     agreement for the SPEC Benchmark Suite for this product.


    ------------------------------------------------------------------------
    This program contain a 64-bit version of Mersenne Twister pseudorandom
    number generator.

    Copyright (C) 2004, Makoto Matsumoto and Takuji Nishimura,
    All rights reserved.

    Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions
    are met:

    1. Redistributions of source code must retain the above copyright
       notice, this list of conditions and the following disclaimer.
    2. Redistributions in binary form must reproduce the above copyright
       notice, this list of conditions and the following disclaimer in
       the documentation and/or other materials provided with the
       distribution.
    3. The names of its contributors may not be used to endorse or
       promote products derived from this software without specific prior
       written permission.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
    "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
    LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
    A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT
    OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
    SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
    LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
    DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
```

```
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
-------------------------------------------------------------------------
 Test run time = 300 seconds, Warmup = 300 seconds.
 Op latency reporting activated
 Files per directory set to 1
 Directories per proc set to 1
 Using custom file size 51200 Kbytes
 Running 2 copies of the test on 1 clients
 Results directory: /home/capps/Downloads/trunk/bin/results
 Clients have a total of 1024 MiBytes of memory
 Clients have 512 MiBytes of memory size per process
 Clients each have 2 processes
 Adjustable aggregate data set value set to 1024 MiBytes
 Each process file size = 51200 kbytes
 Client data set size       = 1100 MiBytes
 Total starting data set size = 1100 MiBytes
 Total initial file space   = 1100 MiBytes
 Total max file space       = 1200 MiBytes

 Starting tests: Thu Oct 12 06:57:11 2017

     Launching 2 processes.
     Starting test client:   0 Host:   centos7v    Workload:            VDI  Location: /tmp
     Starting test client:   1 Host:   centos7v    Workload:            VDI  Location: /tmp
     Thu Oct 12 06:57:19 2017 Prime's GO 'start comm' average message latency 6 ms
     Waiting to finish initialization. Thu Oct 12 06:57:19 2017
     Thu Oct 12 06:57:22 2017 Starting INIT phase
     Thu Oct 12 06:57:28 2017 Init 30 percent complete from client 1
     Thu Oct 12 06:57:28 2017 Init 50 percent complete from client 1
     Thu Oct 12 06:57:36 2017 Init 80 percent complete from client 1
     Thu Oct 12 06:57:36 2017 Init 100 percent complete from client 1
     Initialization finished: Thu Oct 12 06:57:37 2017
     Testing begins: Thu Oct 12 06:57:37 2017
     Thu Oct 12 06:57:37 2017 Actual average warmup GO latency: 7 ms
     Waiting for tests to finish. Thu Oct 12 06:57:37 2017
     Thu Oct 12 06:57:42 2017 Starting WARM phase
     Thu Oct 12 06:58:10 2017 Warm-up 10 percent complete from client 1
     Thu Oct 12 06:58:44 2017 Warm-up 20 percent complete from client 1
     Thu Oct 12 06:59:10 2017 Warm-up 30 percent complete from client 1
     Thu Oct 12 06:59:22 2017 Warm Heartbeat Client 1:  99.861 Ops/sec
     Thu Oct 12 06:59:40 2017 Warm-up 40 percent complete from client 1
     Thu Oct 12 07:00:10 2017 Warm-up 50 percent complete from client 1
     Thu Oct 12 07:00:22 2017 Warm Heartbeat Client 1:  99.774 Ops/sec
     Thu Oct 12 07:00:40 2017 Warm-up 60 percent complete from client 1
     Thu Oct 12 07:01:10 2017 Warm-up 70 percent complete from client 1
     Thu Oct 12 07:01:24 2017 Warm Heartbeat Client 1:  99.883 Ops/sec
     Thu Oct 12 07:01:40 2017 Warm-up 80 percent complete from client 1
     Thu Oct 12 07:02:10 2017 Warm-up 90 percent complete from client 1
     Thu Oct 12 07:02:43 2017 Warm-up 100 percent complete from client 1
     Thu Oct 12 07:02:43 2017 Starting RUN phase
     Thu Oct 12 07:03:11 2017 Run 10 percent complete from client 1
     Thu Oct 12 07:03:27 2017 Run  Heartbeat Client 1:  99.730 Ops/sec
     Thu Oct 12 07:03:41 2017 Run 20 percent complete from client 1
     Thu Oct 12 07:04:11 2017 Run 30 percent complete from client 1
     Thu Oct 12 07:04:41 2017 Run 40 percent complete from client 1
     Thu Oct 12 07:05:11 2017 Run 50 percent complete from client 1
     Thu Oct 12 07:05:22 2017 Run  Heartbeat Client 1: 100.580 Ops/sec
     Thu Oct 12 07:05:42 2017 Run 60 percent complete from client 1
     Thu Oct 12 07:06:12 2017 Run 70 percent complete from client 1
     Thu Oct 12 07:06:22 2017 Run  Heartbeat Client 1: 100.527 Ops/sec
     Thu Oct 12 07:06:42 2017 Run 80 percent complete from client 1
     Thu Oct 12 07:07:12 2017 Run 90 percent complete from client 1
     Thu Oct 12 07:07:22 2017 Run  Heartbeat Client 1: 100.538 Ops/sec
     Thu Oct 12 07:07:42 2017 Run 100 percent complete from client 1
     Thu Oct 12 07:07:43 2017 Prime receiving results from child 1
     Tests finished: Thu Oct 12 07:07:43 2017
     ----------------------------------------------------------------
     Overall average latency              2.322 Milli-seconds
     Overall SPEC SFS2014_SP2           200.004 Ops/sec
     Overall Read_throughput        ~   1089.864 Kbytes/sec
     Overall Write_throughput       ~   1864.093 Kbytes/sec
     Overall throughput             ~   2953.956 Kbytes/sec
     Public Finger Print             24293
     ----------------------------------------------------------------
     Band 1:  20us:7        40us:257      60us:163      80us:96      100us:33
     Band 2: 200us:75      400us:319     600us:2973    800us:4387      1ms:3491
     Band 3:   2ms:15518     4ms:26123     6ms:5918      8ms:230      10ms:103
     Band 4:  12ms:60       14ms:56       16ms:53       18ms:30       20ms:17
```

```
        Band 5:  40ms:19       60ms:0        80ms:0       100ms:0
        Band 6: 200ms:0       400ms:0       600ms:0       800ms:0          1s:0
        Band 7:    2s:0         4s:0          6s:0          8s:0         10s:0
        Band 8:   20s:0        40s:0         60s:0         80s:0        120s:0
        Band 9: 120+s:0
        ---------------------------------------------------------------
netmist completed successfully, summarizing.
```

Reminder: The benchmark run may take many hours to complete depending upon the requested load and how many data points were requested. Also, some failures may take more than an hour to manifest.

5 Submission and Review Process

The SPEC SFS 2014 benchmark release includes a tool for collecting benchmark results in a format that can be submitted by email to the SPEC SFS 2014 results processing facility at SPEC. This facility will automatically process these results and distribute them to the SPEC Storage subcommittee for review. This section describes how you can use these tools to generate a file for each result that you wish to submit to SPEC. It also describes the review process that occurs once the results are submitted. At this point, it is expected that you have become familiar with the SPEC SFS® 2014 SP2 Run and Reporting Rules. See the SPEC SFS® SP2 Run and Reporting Rules documentation that is included in the distribution.

5.1 Creating Reports

Once a benchmark run is completed, the configuration file, results file and additional information are combined into a submission file that is used for submitting runs to SPEC for review using the SpecReport command. Descriptions of the fields that need to be filled out in the submission file are included in Section 6.1 in the SPEC SFS® 2014 SP2 Run and Reporting Rules. This same submission file can be used to generate reports in the form presented on the SPEC web site using the SpecReport command. Each command is documented below.

$ python SpecReport -h

Usage: python SpecReport [options]

Command Line Option	Description	Required/Optional
[-i <file>] or [--submission-file=<file>]	Specify XML submission file	Required
[-r <file>] or [--rc-file=<file>]	Specify RC file	Required for initial package creation
[-s <suffix>] or [--suffix=<suffix>]	Suffix used in log and summary files, similar to SfsManager	Required for initial package creation
[-p <prefix>] or [--prefix=<prefix>]	Prefix common to all submission files that get created. Default during initial submission package creation: sfs2014-YYYYmmdd-HHMM. This parameter is required for renaming existing submissions.	Optional
[-u] or [--update]	Update an existing submission. This option gets the prefix from the submission file (-i <file>) filename. The RC file, suffix, and results directory flags will be ignored. Use with -p <prefix> for a renamed and updated version of a submission.	Optional
[-d <dir>] or	Results directory, default is "results" in the	Optional

[--results-dir=<dir>]	current working directory.	
[-o <file>] or [--output=<dir>]	Output ZIP archive for full disclosure.	Optional
[-a <file1,file2,...>] or [--attachments=<file1,file2,...>]	List of extra files to attach to the submission (e.g. client/mountpoint file)	Optional
[--validate-only]	Validate the submission without creating the full disclosure package.	Optional
[-h]	Show usage info.	Optional

5.1.1 Creating the Submission Package

To create a submission file one must first create an XML document based on the submission_template.xml example found in the base directory. The template document has the correct XML structure expected by SpecReport. Valid entries for each field can be found in the SPEC SFS® 2014 SP2 Run and Reporting Rules. Edit a copy of the template and fill in each field according to the run rules with specific information pertaining to the SUT.

Once the XML submission document is complete a formal submission package can be created with SpecReport. The tool has 3 required arguments: the RC file, the XML submission file, and the suffix used during the test, the same suffix used with SfsManager. To test the submission files for correctness, issue the command

$ python SpecReport -r <RC file> -i <XML file> -s <suffix> --validate-only

The tool will check for the existence of all the necessary files and check the format of the XML document. If the command returns without reporting any errors, repeat the command without the "--validate-only" flag to create the submission package. The package will be a zip archive containing the following files: The RC file, the run summary files, the submission file, an HTML version of the report, a text version of the report, a SUB version of the report, and any configuration diagrams.

The syntax for updating an existing set of submission files is

$ python SpecReport -u -i <XML file>

5.2 Submitting Results

Once you have generated a submission file as described in the Creating the Submission Package section above, you may submit your run for review by the SPEC Storage subcommittee by emailing the ZIP file to `subsfs2014@spec.org`. Upon receipt, the SPEC results processing facility will parse the submission file and validate the formats. If the check passes, an email reply is returned to the sender including a submission number assigned to the result. This submission number is used to track the result during the review and publishing process. If there are any formatting errors, the parser will respond with a failure message indicating where in the file the parsing failed. You may then either correct the error and resubmit or contact the SPEC office for further assistance.

Every results submission goes through a minimum two-week review process, starting on a scheduled SPEC Storage sub-committee conference call. During the review, members of the committee may contact the submitter and request additional information or clarification of the submission. Once the result has been reviewed and accepted by the committee, it is displayed on the SPEC web site at http://www.spec.org/.

6 FAQ

6.1 SPEC SFS 2014 Benchmark Press Release

Question 1: What is the SPEC SFS 2014 benchmark and how does it compare to other storage solution benchmarks?

Answer: The SPEC SFS 2014 benchmark is the latest version of the Standard Performance Evaluation Corp.'s benchmark that measures a storage solution throughput and response time. It differs from other file server benchmarks in that it provides a standardized method for comparing performance across different vendor platforms. The benchmark was written to be solution independent and vendor-neutral. Results are validated through peer review before publication on SPEC's public website <http:// www.spec.org/SPECsfs2014/>

Question 2: What improvements have been made to the SPEC SFS 2014 benchmark?

Answer: See the "What's New in SPEC SFS 2014?" document included with the benchmark or on the SPEC SFS 2014 website: http://www.spec.org/sfs2014/

Question 3: How were the SPEC SFS 2014 workloads determined?

Answer: The SPEC SFS 2014 workloads are based on network traces collected from real environments and input from domain experts in and published documentation of the real-world implementation of the application types being simulated.

Question 4: What are the metrics for the SPEC SFS 2014 benchmark?

Answer: The SPEC SFS 2014 benchmark has multiple performance measurement metrics:

- DATABASE = DATABASES
- EDA = JOB_SETS
- SWBUILD = BUILDS
- VDA = STREAMS
- VDI = DESKTOPS

Question 5: What is the correlation between the SPEC SFS 2014 benchmark and the TPC (Transaction Processing Council) and SPC (Storage Performance Council) benchmarks?

Answer: There is no correlation; the benchmarks present very different workloads on the solutions under test and measure different aspects of solution performance.

Question 6: Is the SPEC SFS 2014 benchmark a CPU-intensive or I/O-intensive benchmark?

Answer: The SPEC SFS 2014 benchmark is an application-level benchmark that heavily exercises CPU, mass storage and network components. The greatest emphasis is on I/O, especially as it relates to operating and file system software. To obtain the best performance for a system running the SPEC SFS 2014 benchmark, the vendor will typically add additional hardware -- such as memory, disk controllers, disks, network controllers and buffer cache – as needed in order to help alleviate I/O bottlenecks and to ensure that server CPUs are used fully.

Question 7: For what computing environment is the SPEC SFS 2014 benchmark designed?

Answer: The benchmark was developed for load-generating clients running UNIX or Windows. The SPEC SFS 2014 benchmark can be used to evaluate the performance of any storage solution, regardless of the underlying environment.

Question 8: Can users measure performance for workloads other than the ones provided within the SPEC SFS 2014 benchmark?

Answer: Yes, users can measure their own workloads by making changes to the SPEC SFS 2014 benchmark workload objects file. The SPEC SFS® 2014 User's Guide details how this can be done. Workloads created by users cannot, however, be compared with SPEC SFS 2014 results, nor can they be published in any form, as specified within the SPEC SFS 2014 license.

Question 9: To what extent is the server's measured performance within the SPEC SFS 2014 benchmark affected by the client's performance?

Answer: SPEC has written the SPEC SFS 2014 benchmark to include the effect of client performance on SPEC SFS 2014 results. This is a storage solution benchmark, not a component level benchmark. The aggregate data set sweeps a range that covers the in cache and out of cache cases for the solution. This provides coverage for the real world situations.

Question 10: How does SPEC validate numbers that it publishes?

Answer: Results published on the SPEC Web site have been reviewed by SPEC members for compliance with the SPEC SFS® 2014 SP2 Run and Reporting Rules, but there is no monitoring beyond that compliance check. The vendors that performed the tests and submitted the performance numbers have sole responsibility for the results. SPEC is not responsible for any measurement or publication errors.

Question 11: Are the reported SPEC SFS 2014 configurations typical of systems sold by vendors?

Answer: Yes and no. They are similar to large server configurations, but the workload is heavier than that found on smaller server configurations. SPEC has learned from experience that today's heavy workload is tomorrow's light workload. For some vendors, the configurations are typical of what they see in real customer environments, particularly those incorporating high-end servers. For other vendors, SPEC SFS 2014 configurations might not be typical.

Question 12: Do the SPEC SFS® 2014 SP2 Run and Reporting Rules allow results for a clustered server?

Answer: Yes, cluster configurations are allowed as long as they conform to the SPEC SFS® 2014 SP2 Run and Reporting Rules.

Question 13: What resources are needed to run the SPEC SFS 2014 benchmark?

Answer: In addition to a server, a test bed includes several clients and an appropriate number of networks. Ideally, the server should have enough memory, disks and network hardware to saturate the CPU. The test bed requires at least one network. Examples of typical load-generating configurations can be found on the SPEC Web site: < http://www.spec.org/SPECsfs2014/>.

Question 14: What is the estimated time needed to set up and run the SPEC SFS 2014 benchmark?

Answer: Hardware setup and software installation time depend on the size of the server and the complexity of the test beds. Many servers require large and complex test beds. The SPEC SFS 2014 software installs relatively quickly. A SPEC SFS 2014 submission from a vendor includes at least 10 data points, with each data point taking from ~30 to ~90 minutes to complete. The performance of the storage solution is a factor in the time it takes to setup and run each load point.

Question 15: What shared resources does the SPEC SFS 2014 benchmark use that might limit performance?
Answer: Shared resources that might limit performance include CPU, memory, disk controllers, disks, network controllers, network concentrators, network switches, clients, etc.

Question 16: Does the SPEC SFS 2014 benchmark permit tuning parameters?
Answer: When submitting results for SPEC review, vendors are required to supply a description of all tuning parameters for all cases where non-default values were used for all components in the SUT. This information is displayed in the appropriate sections in published SPEC SFS 2014 results.

Question 17: Can a RAM disk be used within a SPEC SFS 2014 configuration?
Answer: SPEC enforces strict storage rules for stability. Generally, RAM disks do not meet these rules, since they often cannot survive cascading failure-recovery requirements unless an uninterruptible power supply (UPS) with long survival times is used.

Question 18: How will the choice of networks affect SPEC SFS 2014 results?
Answer: Different link types and even different implementations of the same link type might affect the measured performance -- for better or worse -- of a particular server. Consequently, the results measured by clients in these situations might vary as well.

Question 19: Is the SPEC SFS 2014 benchmark scalable with respect to CPU, cache, memory, disks, controllers and faster transport media?
Answer: Yes the benchmark is scalable as users migrate to faster technologies.

Question 20: What is the price of a SPEC SFS 2014 license and when will it be available?
Answer: The SPEC SFS 2014 benchmark is available now from the SPEC download site for US$2,000. A discounted price is available for non-profit and academic licensees. Contact the SPEC office: (See www.spec.org for any updates)

Standard Performance Evaluation Corporation (SPEC)
7001 Heritage Village Plaza
Suite 225
Gainesville, VA 20155
Phone: 1-703-579-8460
Fax: 1-703-579-8463
E-Mail: info@spec.org

Question 21: Can users get help in understanding how to run the SPEC SFS 2014 benchmark?
Answer: The majority of questions should be answered in the SPEC SFS® 2014 User's Guide. There is also useful information on the SPEC Web site: http://www.spec.org/SPECsfs2014/

Question 22: Do I need to measure every workload?
Answer: No. Each workload has a separate metric that can be published independently.

Question 23: How do I get started running the SPEC SFS 2014 benchmark?
Answer: Please read the SPEC SFS® 2014 User's Guide and SPEC SFS® 2014 SP2 Run and Reporting Rules in their entirety.

Question 24: I am running into problems setting up and running the benchmark. What can I do?
Answer: The most common problem is usually that file server file systems are not being correctly mounted on the clients. Most of the problems relating to the SPEC SFS 2014 benchmark can be resolved by referring to appropriate sections of the User's Guide, including this FAQ. Other common problems include: hosts file/DNS configuration, firewalls not being

disabled, password-less SSH not configured, or attempting to run outside a domain in a Windows environment.

Question 25: I have read the SPEC SFS® 2014 User's Guide. But I am still running into problems. What can I do next?
Answer: Looking at the sfslog.* and the sfsc* files can give you an idea as to what may have gone wrong. Inspecting the client logs in /tmp/, c:\tmp\, or at NETMIST_LOGS on each load generator can also provide more details on errors. And, as a last resort, you can contact SPEC at support@spec.org. It is assumed that such calls/emails are from people who have read the SPEC SFS® 2014 User's Guide completely, and have met all the prerequisites for setting up and running the benchmark.

Question 26: How does one abort a run?
Answer: The benchmark can be aborted by simply stopping the SfsManager. This will terminate all SFS related processes on all clients and on the prime client.

Question 27: For a valid run, which parameters are required to be unchanged?
Answer: Information is provided in the SPEC SFS® 2014 SP2 Run and Reporting Rules and in the sfs_rc file, and this is enforced by the benchmark. If invalid parameter values are selected, the benchmark reports an invalid run.

Question 28: Is there a quick way to debug a testbed?
Answer: Read the SPEC SFS® 2014 User's Guide, ping the server from the client, ping from the prime client to the other clients and vice versa, validate that the paths in CLIENT_MOUNTPOINTS exist on all load generators and are writeable, validate password-less SSH works from the prime client to all load generators, run the benchmark with one client and one file system.

6.2 Tuning the Solution

Question 29: What are a reasonable set of parameters for running the benchmark?
Answer: Study existing results pages with configuration information similar to your system configuration.

Question 30: How do I increase the performance of our solution?
Answer: One may need to add, as necessary, one or more of the following: processors, memory, disks, controllers, etc.

6.3 Submission of Results

Question 31: We have a valid set of results. How do we submit these results to SPEC?
Answer: See the Submission and Review Process section above. The new submission tool documentation is in that section.

7 Trademarks

IBM and AIX are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Iozone is a trademark of Iozone.org, in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

MacOS is a trademark of Apple, Inc. in the United States, other countries, or both.

Microsoft(R), Windows, Windows NT(R), Visual Studio, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Solaris is a trademark of Oracle, in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

8 Research corner

8.1 Custom changes to SfsManager to add new workloads.

Each workload has an XML description in the benchmarks.xml file. When adding or changing workloads within the SfsManager, edit this file to define or change the definition of a workload.

Example :

```
<benchmark name="NEW" business_metric="NEWMETRIC">
    <workload name="WORKLOAD1">
        <oprate>100</oprate>
        <instances>1</instances>
    </workload>
    <workload name="WORKLOAD2">
        <oprate>10</oprate>
        <instances>4</instances>
        <shared_files/>
    </workload>
    <dedicated_subdirectory/>
    <override_parm name="RUNTIME">300</override_parm>
    <override_parm name="FILE_SIZE">500m</override_parm>
    <override_parm name="DIR_COUNT">10</override_parm>
    <override_parm name="FILES_PER_DIR">100</override_parm>
    <override_parm name="SHARING_MODE">1</override_parm>
    <threshold type="proc oprate">75</threshold>
    <threshold type="global oprate">90</threshold>
    <threshold type="workload variance">10</threshold>
</benchmark>
```

8.2 Custom workload objects

The SPEC SFS 2014 benchmark is capable of running user defined workloads. These are not publishable via SPEC, however consumers may find it useful to create a model of their specific application load and then to be able to run the benchmark while presenting their specific application's workload.

8.2.1 Export current workload definitions

The existing SPEC SFS 2014 workload definitions may be exported to a flat text file by executing the command:

```
netmist -E > workload.txt
```

8.2.2 Workload object attribute definitions

Each workload object contains the following attributes:

Workload name	< Name of the Workload object >
Percent Read	< Percent of the Op mix that is sequential read transfers from files >
Percent Read_file	< Percent of the Op mix that is sequential read whoe files >
Percent Mmap_read	< Percent of the Op mix that is using Mmap to read files >
Percent Random_read	< Percent of the Op mix that is reading at random locations >
Percent Write	< Percent of the Op mix that is sequential write transfers to files >
Percent Write_file	< Percent of the Op mix that is sequential write of whole files >
Percent Mmap_write	< Percent of the Op mix that is using Mmap to write files >
Percent Random_write	< Percent of the Op mix that is writing at random locations >

Percent Read_modify_write	< Percent of the Op mix that is read/modify/write at location in a file >
Percent Mkdir	< Percent of the Op mix that is creating new directories >
Percent Rmdir	< Percent of the Op mix that is removing existing directories >
Percent Unlink	< Percent of the Op mix that is removing empty files >
Percent Unlink2	< percent of the Op mix that is removing files that have size >
Percent Create	< Percent of the Op mix that is creating new files >
Percent Append	< Percent of the Op mix that is appending writes to existing files >
Percent Lock	< Percent of the Op mix that is locking files >
Percent Access	< Percent of the Op mix that is calling access() on files >
Percent Stat	< Percent of the Op mix that is calling stat() on files >
Percent Chmod	< Percent of the Op mix that is calling chmod() on files >
Percent Readdir	< Percent of the Op mix that is calling readdir() on directories >
Percent Copyfile	< Percent of the Op mix that is copying whole files >
Percent Rename	< Percent of the Op mix that is renaming files >
Percent Statfs	< Percent of the Op mix that is calling statfs() >
Percent Pathconf	< Percent of the Op mix that is calling pathconf() >
Percent Custom1	< Reserved for future use >
Percent Custom2	< Reserved for future use >
Read elem 0 xfer min size	< Minimum transfer size for this read slot >
Read elem 0 xfer max size	< Maximum transfer size for this read slot >
Read elem 0 xfer percent	< Percent of reads that are of this slot definition >
Read elem 1 xfer min size	< Minimum transfer size for this read slot >
Read elem 1 xfer max size	< Maximum transfer size for this read slot >
Read elem 1 xfer percent	< Percent of reads that are of this slot definition >
Read elem 2 xfer min size	< Minimum transfer size for this read slot >
Read elem 2 xfer max size	< Maximum transfer size for this read slot >
Read elem 2 xfer percent	< Percent of reads that are of this slot definition >
Read elem 3 xfer min size	< Minimum transfer size for this read slot >
Read elem 3 xfer max size	< Maximum transfer size for this read slot >
Read elem 3 xfer percent	< Percent of reads that are of this slot definition >
Read elem 4 xfer min size	< Minimum transfer size for this read slot >
Read elem 4 xfer max size	< Maximum transfer size for this read slot >
Read elem 4 xfer percent	< Percent of reads that are of this slot definition >
Read elem 5 xfer min size	< Minimum transfer size for this read slot >
Read elem 5 xfer max size	< Maximum transfer size for this read slot >
Read elem 5 xfer percent	< Percent of reads that are of this slot definition>
Read elem 6 xfer min size	< Minimum transfer size for this read slot >
Read elem 6 xfer max size	< Maximum transfer size for this read slot >
Read elem 6 xfer percent	< Percent of reads that are of this slot definition>
Read elem 7 xfer min size	< Minimum transfer size for this read slot >
Read elem 7 xfer max size	< Maximum transfer size for this read slot >
Read elem 7 xfer percent	< Percent of reads that are of this slot definition>
Read elem 8 xfer min size	< Minimum transfer size for this read slot >
Read elem 8 xfer max size	< Maximum transfer size for this read slot >
Read elem 8 xfer percent	< Percent of reads that are of this slot definition>
Read elem 9 xfer min size	< Minimum transfer size for this read slot >
Read elem 9 xfer max size	< Maximum transfer size for this read slot >
Read elem 9 xfer percent	< Percent of reads that are of this slot definition>
Read elem 10 xfer min size	< Minimum transfer size for this read slot >
Read elem 10 xfer max size	< Maximum transfer size for this read slot >
Read elem 10 xfer percent	< Percent of reads that are of this slot definition>
Read elem 11 xfer min size	< Minimum transfer size for this read slot >
Read elem 11 xfer max size	< Maximum transfer size for this read slot >
Read elem 11 xfer percent	< Percent of reads that are of this slot definition>

Read elem 12 xfer min size	< Minimum transfer size for this read slot >
Read elem 12 xfer max size	< Maximum transfer size for this read slot >
Read elem 12 xfer percent	< Percent of reads that are of this slot definition >
Read elem 13 xfer min size	< Minimum transfer size for this read slot >
Read elem 13 xfer max size	< Maximum transfer size for this read slot >
Read elem 13 xfer percent	< Percent of reads that are of this slot definition >
Read elem 14 xfer min size	< Minimum transfer size for this read slot >
Read elem 14 xfer max size	< Maximum transfer size for this read slot >
Read elem 14 xfer percent	< Percent of reads that are of this slot definition >
Read elem 15 xfer min size	< Minimum transfer size for this read slot >
Read elem 15 xfer max size	< Maximum transfer size for this read slot >
Read elem 15 xfer percent	< Percent of reads that are of this slot definition >
Write elem 0 xfer min size	< Minimum transfer size for this write slot >
Write elem 0 xfer max size	< Maximum transfer size for this write slot >
Write elem 0 xfer percent	< Percent of writes that are of this slot definition >
Write elem 1 xfer min size	< Minimum transfer size for this write slot >
Write elem 1 xfer max size	< Maximum transfer size for this write slot >
Write elem 1 xfer percent	< Percent of writes that are of this slot definition >
Write elem 2 xfer min size	< Minimum transfer size for this write slot >
Write elem 2 xfer max size	< Maximum transfer size for this write slot >
Write elem 2 xfer percent	< Percent of writes that are of this slot definition >
Write elem 3 xfer min size	< Minimum transfer size for this write slot >
Write elem 3 xfer max size	< Maximum transfer size for this write slot >
Write elem 3 xfer percent	< Percent of writes that are of this slot definition >
Write elem 4 xfer min size	< Minimum transfer size for this write slot >
Write elem 4 xfer max size	< Maximum transfer size for this write slot >
Write elem 4 xfer percent	< Percent of writes that are of this slot definition >
Write elem 5 xfer min size	< Minimum transfer size for this write slot >
Write elem 5 xfer max size	< Maximum transfer size for this write slot >
Write elem 5 xfer percent	< Percent of writes that are of this slot definition >
Write elem 6 xfer min size	< Minimum transfer size for this write slot >
Write elem 6 xfer max size	< Maximum transfer size for this write slot >
Write elem 6 xfer percent	< Percent of writes that are of this slot definition >
Write elem 7 xfer min size	< Minimum transfer size for this write slot >
Write elem 7 xfer max size	< Maximum transfer size for this write slot >
Write elem 7 xfer percent	< Percent of writes that are of this slot definition >
Write elem 8 xfer min size	< Minimum transfer size for this write slot >
Write elem 8 xfer max size	< Maximum transfer size for this write slot >
Write elem 8 xfer percent	< Percent of writes that are of this slot definition >
Write elem 9 xfer min size	< Minimum transfer size for this write slot >
Write elem 9 xfer max size	< Maximum transfer size for this write slot >
Write elem 9 xfer percent	< Percent of writes that are of this slot definition >
Write elem 10 xfer min size	< Minimum transfer size for this write slot >
Write elem 10 xfer max size	< Maximum transfer size for this write slot >
Write elem 10 xfer percent	< Percent of writes that are of this slot definition >
Write elem 11 xfer min size	< Minimum transfer size for this write slot >
Write elem 11 xfer max size	< Maximum transfer size for this write slot >
Write elem 11 xfer percent	< Percent of writes that are of this slot definition >
Write elem 12 xfer min size	< Minimum transfer size for this write slot >
Write elem 12 xfer max size	< Maximum transfer size for this write slot >
Write elem 12 xfer percent	< Percent of writes that are of this slot definition >
Write elem 13 xfer min size	< Minimum transfer size for this write slot >
Write elem 13 xfer max size	< Maximum transfer size for this write slot >
Write elem 13 xfer percent	< Percent of writes that are of this slot definition >

Write elem 14 xfer min size	< Minimum transfer size for this write slot >
Write elem 14 xfer max size	< Maximum transfer size for this write slot >
Write elem 14 xfer percent	< Percent of writes that are of this slot definition >
Write elem 15 xfer min size	< Minimum transfer size for this write slot >
Write elem 15 xfer max size	< Maximum transfer size for this write slot >
Write elem 15 xfer percent	< Percent of writes that are of this slot definition >
Percent write commit	< Percent of writes that will be flushed/committed >
Percent direct	< Percent of I/O reads and writes that use O_DIRECT >
Align	< Align file accesses to this size boundary for reads and writes >
Percent osync	< Percent of I/O operations that use O_SYNC >
Percent geometric	< Percent of geometric distribution of file accesses >
Percent compress	< Percent of compressibility of file data >
Percent dedup	< Percent of dedupability of the file data >
Percent dedup_within	< Percent of dedupability within any given file >
Percent dedup_across	< Percent of dedupability across files >
Dedupe group count	< Number of dedupe groups that can share across files >
Percent per_spot	< Percent of file size that is used to set the hot spot size >
Min acc_per_spot	< Minimum accesses per hot spot before re-selecting >
Acc mult_spot	< Scaled hits within a hot spot >
Percent affinity	< Affinity to move away from a hot spot >
Spot shape	< Shape of hot spot. 0 = Uniform Rand, 1 = Geometric rand dist >
Dedup Granule size	< Granule size for Dedup. Minimum dedupable unit size >
Dedup gran rep limit	< Maximum number of dedup replicants >
Comp Granule size	< Granule size for compression. Minimum compressible unit size >
Background	< When set does not include in measurement >
Sharemode	< When set, all files and directories are shared across procs >
Uniform size dist	< When set, all files are the same size, else Gaussian dist >
Rand dist behavior	< Distribution pattern for workload within files: 0 = uniform random; 1 = geometric distribution; 2 = geometric across hot spots >
Cipher behavior	< Encrypt all files to create flat character frequency distribution data >
Notification percent	< Percent of writes that will also generate notification events >
LRU	< Enable use of LRU cache for file descriptors >
Pattern version	< Set data fill pattern characteristics: 1 = SP1, 2 = SP2 >
Init rate throttle	< Limit init rate per proc to this value in MiB/s >
Init read flag	< Enable read of existing dataset during INIT; 1 = enable >
Release version	< Identify SFS 2014 SP workload version; 1 = SP1; 2 = SP2 >
FS type POSIX	< Default is POSIX. Used for non-POSIX implementations >

8.2.3 Edit workload definitions

The existing workload definitions must be exported to a file to modify the workload definitions to ensure the file format is correct. The contents are in flat ASCII and can be edited with your favorite editor. Do not remove or add any lines to this file. You may only edit the existing lines, and save the file in the same format.

There are currently nine defined workloads: DB_LOG, DB_TABLE, EDA_FRONTEND, EDA_BACKEND, VDA1, VDA2, VDI, SWBUILD, and USERDEF.

The values of each parameter may be modified to create a new workload. Once the modifications are complete, this new workload definition may be imported and used to run the user defined workload.

NOTE: The values for the existing workload descriptions may be modified, but the number of workload objects may not be changed.

8.2.4 Import workload definitions

To import a user defined workload, edit the sfs_rc file and modify the value for the WORKLOAD_FILE= parameter to contain the name of the workload definition file that was modified.
Example:

```
WORKLOAD_FILE=workload_file
```

The custom workload file need only be present on the prime client, just like the rc file.

NOTE: once you define a custom workload, you must define a workload object in the benchmarks.xml file to use the workload with the SfsManager framework. See section 8.1 above for more detail.

8.2.5 Client mountpoint file format

When specifying a separate file for the CLIENT_MOUNTPOINT option in the SPEC SFS 2014 rc file, use the following format (note, additional options are shown for example purposes, but a real mountpoints file would likely be more consistent):

```
client1 /mnt1
client2 /mnt1
client3 /mnt1 /mnt2 filesize=8192
client4 /mnt1 dircount=5 filesperdir=10
client6 /mnt3 logpath=/mnt1
```

The file format consists of one load generator hostname or IP followed by one or more mountpoints with optional extra options that are passed to netmist. The file is space-delimited. The available extra options are:

filesize	set size of files in the dataset (size in KiB)
dircount	set the number of directories in the dataset
filesperdir	set the number of files per directory in the dataset
logpath	set a separate log path for this proc

Note: all these options apply for any business metrics (group of procs) assigned to the particular entry in the client mountpoints list. Therefore, in the example above, any business metric assigned to the last entry (client6 /mnt3) would log to the /mnt1 directory on client6.

9 Appendix A – Building SFS Benchmark Components

9.1 Building the SPEC SFS 2014 benchmark for UNIX

Note that it is a requirement that you have a GNU-compatible build environment (e.g., a 'gmake' compatible 'make') in order to build SFS on a UNIX system.

To build SFS, you need to simply:

- *cd* to the top-level spec directory ($SPEC)
- Type *make*

9.2 Building the SPEC SFS 2014 benchmark for Windows

The SPEC SFS 2014 benchmark for WIN32 was developed using Microsoft Visual Studio 2015 Community. Other versions of the development environment may work for building the benchmark, but these have not been tested.

9.2.1 Update Visual Studio Libraries/Includes

If not already installed, download and install Microsoft Visual Studio 2015 and the Windows SDK. The installation and configuration of Visual Studio for proper operation is beyond the scope of this document.

9.2.2 Open the Visual Studio solution file

Obtain the SPEC SFS 2014 source base and place it in a convenient location on the system to be used for building. In $SPEC/msbuild there is a Visual Studio solution file that can be used to build the benchmark executables for x86 (32 bit) as well as x64 (64 bit) based systems.

9.2.3 Build the individual project files

If you are familiar with building projects in Visual Studio, you may build the projects as you are accustomed to doing and skip this section.

To build all the projects at once, from the *Build* menu select *Batch Build*. With all projects and configurations selected, click *Build* or *Rebuild All* and the build process will begin. The binaries will be built in *bin\dist\windows\Win32* and/or *bin\dist\windows\x64*.

To build the individual projects, from the *Project* menu select a project under the *Select Active Project* submenu. The selected project will appear in bold in the workspace window. From the *Build* menu, select *Rebuild All* to force a rebuild of all modules in that project, or select *Build [project name]* to build only those modules that have changed or are missing.

Paths and directories within the project settings are all relative to the project file locations, so building the benchmark files should be possible regardless of where the source base is placed on the system.

10 Appendix B – Setting up password-less SSH

Here is a sample script that can be used to set up password-less SSH on Linux clients.

```
# Define the hosts to be involved in the trust here
# DO NOT include the host you are running, it is added by default

hosts="s2 s3 s4 s5 s6"

echo ""
echo ""
echo "This script will generate SSH keys for the specified machines,"
echo " and set up password-less authentication between them."
echo " You will be prompted for passwords several times during this
process."
echo ""

# Get current user
user=`who -m | awk {'print $1'}`
echo "Trust will be configured for user $user"
echo ""
echo "If this is not correct, stop and login as the appropriate user"
echo -n "(RETURN to continue, CTRL-C to exit) "

read continue

# Configure keys on current host
cd $HOME
ssh-keygen -t rsa
cat .ssh/id_rsa.pub >> .ssh/authorized_keys
chmod 700 .ssh
chmod 600 .ssh/*

for host in $hosts
do
   ssh $user@$host 'ssh-keygen -t rsa'
   ssh $user@$host 'cat .ssh/id_rsa.pub' | cat - >>
~/.ssh/authorized_keys
done

for host in $hosts
do
   scp .ssh/authorized_keys $host:.ssh
   ssh $user@$host 'chmod 700 .ssh ; chmod 600 .ssh/*'
done

exit
```

11 Appendix C – Tunes for the load generators

11.1 Linux tunes

Some or all of the following tuning parameters may be helpful in tuning Linux load generators for optimal performance. The net.core.somaxconn tuning, if available, is highly recommended for any Linux environment running SPEC SFS 2014. Tuning this will avoid excessive delays during benchmark startup/shutdown between iterations.

To make these tunings persistent, use /etc/sysctl.conf – see the documentation for sysctl.conf for more information.

```
if  your NIC type == 10 GigE
  echo "300000" > /proc/sys/net/core/netdev_max_backlog
fi
echo "131071" > /proc/sys/net/core/rmem_default
echo "131071" > /proc/sys/net/core/rmem_max
echo "131071" > /proc/sys/net/core/wmem_default
echo "131071" > /proc/sys/net/core/wmem_max
echo "4096 87380 8388608" > /proc/sys/net/ipv4/tcp_rmem
echo "4096 87380 8388608" > /proc/sys/net/ipv4/tcp_wmem
echo "128"    > /proc/sys/sunrpc/tcp_slot_table_entries
echo "65536"  > /proc/sys/net/core/somaxconn
echo "5"      > /proc/sys/net/ipv4/tcp_fin_timeout
```

Power management, and it's impacts on performance: (From an Operating system perspective)
https://docs-old.fedoraproject.org/en-US/Fedora/20/html/Power_Management_Guide/index.html

Limiting RAM in the client
There may be situations where one wishes to limit the amount of RAM that is used in the client. This can be accomplished via kernel boot options either at the boot prompt, or by editing /boot/grub/grub.conf. See the mem=xxxM option in the Linux vendor's installation guide.

11.2 Windows tunes

- Disable power management, or set every possible value to "High performance" mode. https://docs.microsoft.com/en-us/windows-server/administration/performance-tuning/hardware/power/power-performance-tuning
- Disable the Windows firewall on all clients and servers.
    - Note that upon joining a domain, a new firewall setting may appear for domain networks that defaults to enabled
- Disable the Windows Defender, and any other antivirus software, on the clients and servers.
- Disable all slide-show wallpapers, and screen savers.
- Disable automatic updates.
- Disable any interrupt throttling.

Limiting RAM in the client.

- There may be situations where one wishes to limit the amount of RAM that is used in the client. See: http://msdn.microsoft.com/en-us/library/ff542202(v=vs.85).aspx
  The section on "remove memory" provides information on how to reduce memory.

12 Appendix D – Workload Definition Tables

DB_TABLE

File Operation Distribution

Operation	%	Operation	%
read	1	read file	0
mmap read	0	rand read	79
write	0	write file	0
mmap write	0	rand write	20
rmw	0	append	0
mkdir	0	rmdir	0
readdir	0	create	0
unlink	0	unlink2	0
stat	0	access	0
rename	0	copyfile	0
locking	0	chmod	0
statfs	0	pathconf	0

Thresholds

Threshold	%	Threshold	Value
proc oprate	75	proc latency	n/a
global oprate	95	global latency	n/a
workload variance	5		

Execution Parameters

Parameter	Value	Parameter	Value
Procs	10	Dirs per proc	1
Oprate per proc	16	Files per dir	5
Avg file size			200 MiB

Miscellaneous

Option	Value	Option	Value
write commit %	100	background	0
% direct	100	sharemode	1
% osync	0	uniform size dist	1
% notification	0	init rate throttle	0
LRU	0	init read flag	1
release version	1		

Access Patterns

Option	Value	Option	Value
rand dist behavior	2	% per spot	5
min acc per spot	1000	access mult spot	5
affinity %	20	spot shape	0
geometric %	2	align	0

Content Patterns

Option	Value	Option	Value
dedup %	0	dedup within %	0
dedup across %	0	dedup group count	1
dedup granule size	4096	dedup gran rep limit	100
compress %	50	comp granule size	8192
cipher flag	0	pattern version	1

Read Transfer Size Distribution

Slot	Start	End	%
0	1	511	0
1	512	1023	0
2	1024	2047	0
3	2048	4095	0
4	4096	4096	0
5	4097	8191	0
6	8192	8192	99
7	8193	16383	0
8	16384	16384	0
9	16385	32767	0
10	32768	32768	0
11	65536	65536	0
12	98304	98304	0
13	131072	131072	0
14	262144	262144	0
15	1048576	1048576	1

Write Transfer Size Distribution

Slot	Start	End	%
0	512	512	0
1	1024	1024	0
2	1536	1536	0
3	2048	2048	0
4	2560	2560	0
5	3072	3072	0
6	3584	3584	0
7	4096	4096	0
8	4608	4608	0
9	5120	5120	0
10	8192	8192	100
11	12288	12288	0
12	16384	16384	0
13	20480	20480	0
14	24576	24576	0
15	32768	32768	0

DB_LOG

File Operation Distribution

Operation	%	Operation	%
read	0	read file	0
mmap read	0	rand read	0
write	80	write file	0
mmap write	0	rand write	20
rmw	0	append	0
mkdir	0	rmdir	0
readdir	0	create	0
unlink	0	unlink2	0
stat	0	access	0
rename	0	copyfile	0
locking	0	chmod	0
statfs	0	pathconf	0

Thresholds

Threshold	%	Threshold	Value
proc oprate	75	proc latency	
global oprate	95	global latency	
workload variance	5		

Execution Parameters

Parameter	Value	Parameter	Value
Procs	1	Dirs per proc	1
Oprate per proc	32	Files per dir	5
Avg file size	200 MiB		

Miscellaneous

Option	Value	Option	Value
write commit %	100	background	0
% direct	100	sharemode	1
% osync	0	uniform size dist	1
% notification	0	init rate throttle	0
LRU	0	init read flag	1
release version	1		

Access Patterns

Option	Value	Option	Value
rand dist behavior	2	% per spot	5
min acc per spot	1000	access mult spot	5
affinity %	20	spot shape	0
geometric %	2	align	0

Content Patterns

Option	Value	Option	Value
dedup %	0	dedup within %	0
dedup across %	0	dedup group count	1
dedup granule size	4096	dedup gran rep limit	100
compress %	50	comp granule size	8192
cipher flag	0	pattern version	1

Read Transfer Size Distribution

Slot	Start	End	%
0	1	511	0
1	512	1023	0
2	1024	2047	0
3	2048	4095	0
4	4096	4096	0
5	4097	8191	0
6	8192	8192	99
7	8193	16383	0
8	16384	16384	0
9	16385	32767	0
10	32768	32768	0
11	65536	65536	0
12	98304	98304	0
13	131072	131072	0
14	262144	262144	0
15	1048576	1048576	1

Write Transfer Size Distribution

Slot	Start	End	%
0	512	512	5
1	1024	1024	5
2	1536	1536	5
3	2048	2048	5
4	2560	2560	5
5	3072	3072	5
6	3584	3584	5
7	4096	4096	5
8	4608	4608	5
9	5120	5120	5
10	8192	8192	10
11	12288	12288	10
12	16384	16384	10
13	20480	20480	10
14	24576	24576	10
15	32768	32768	0

EDA_FRONTEND

File Operation Distribution

Operation	%	Operation	%
read	0	read file	7
mmap read	0	rand read	8
write	0	write file	10
mmap write	0	rand write	15
rmw	0	append	0
mkdir	1	rmdir	0
readdir	0	create	2
unlink	1	unlink2	1
stat	39	access	15
rename	0	copyfile	0
locking	0	chmod	1
statfs	0	pathconf	0

Thresholds

Threshold	%	Threshold	Value
proc oprate	75	proc latency	n/a
global oprate	95	global latency	n/a
workload variance	5		

Execution Parameters

Parameter	Value	Parameter	Value
Procs	3	Dirs per proc	10
Oprate per proc	100	Files per dir	10
Avg file size			16 KiB

Miscellaneous

Option	Value	Option	Value
write commit %	15	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate throttle	1
LRU	1	init read flag	0
release version	2		

Access Patterns

Option	Value	Option	Value
rand dist behavior	0	% per spot	0
min acc per spot	0	access mult spot	5
affinity %	0	spot shape	0
geometric %	50	align	0

Content Patterns

Option	Value	Option	Value
dedup %	50	dedup within %	0
dedup across %	0	dedup group count	1
dedup granule size	4096	dedup gran rep limit	100
compress %	50	comp granule size	8192
cipher flag	0	pattern version	2

Read Transfer Size Distribution

Slot	Start	End	%
0	1	511	4
1	512	1023	0
2	1024	2047	0
3	2048	4095	2
4	4096	8191	43
5	8192	16383	30
6	16384	32767	21
7	32768	65535	0
8	65536	65536	0
9	131072	131072	0
10	1	1	0
11	1	1	0
12	1	1	0
13	1	1	0
14	1	1	0
15	1	1	0

Write Transfer Size Distribution

Slot	Start	End	%
0	1	511	25
1	512	1023	10
2	1024	2047	15
3	2048	4095	18
4	4096	8191	27
5	8292	16383	3
6	16384	32767	2
7	32768	65535	0
8	65536	65536	0
9	131072	131072	0
10	1	1	0
11	1	1	0
12	1	1	0
13	1	1	0
14	1	1	0
15	1	1	0

EDA_BACKEND

File Operation Distribution

Operation	%	Operation	%
read	50	read file	0
mmap read	0	rand read	0
write	50	write file	0
mmap write	0	rand write	0
rmw	0	append	0
mkdir	0	rmdir	0
readdir	0	create	0
unlink	0	unlink2	0
stat	0	access	0
rename	0	copyfile	0
locking	0	chmod	0
statfs	0	pathconf	0

Thresholds

Threshold	%	Threshold	Value
proc oprate	75	proc latency	n/a
global oprate	95	global latency	na
workload variance	5		

Execution Parameters

Parameter	Value	Parameter	Value
Procs	2	Dirs per proc	5
Oprate per proc	75	Files per dir	10
Avg file size			10 MiB

Miscellaneous

Option	Value	Option	Value
write commit %	15	background	0
% direct	50	sharemode	0
% osync	5	uniform size dist	0
% notification	0	init rate throttle	1
LRU	1	init read flag	0
release version	2		

Access Patterns

Option	Value	Option	Value
rand dist behavior	0	% per spot	0
min acc per spot	0	access mult spot	5
affinity %	0	spot shape	0
geometric %	50	align	0

Content Patterns

Option	Value	Option	Value
dedup %	40	dedup within %	0
dedup across %	0	dedup group count	1
dedup granule size	4096	dedup gran rep limit	100
compress %	20	comp granule size	8192
cipher flag	0	pattern version	2

Read Transfer Size Distribution

Slot	Start	End	%
0	1	511	0
1	512	1023	0
2	1024	2047	0
3	2048	4095	0
4	4096	8191	0
5	8192	16383	0
6	16384	32767	0
7	32768	65535	49
8	65536	65536	51
9	131072	131072	0
10	1	1	0
11	1	1	0
12	1	1	0
13	1	1	0
14	1	1	0
15	1	1	0

Write Transfer Size Distribution

Slot	Start	End	%
0	1	511	0
1	512	1023	0
2	1024	2047	0
3	2048	4095	0
4	4096	8191	0
5	8292	16383	0
6	16384	32767	0
7	32768	65535	45
8	65536	131072	55
9	131072	131072	0
10	1	1	0
11	1	1	0
12	1	1	0
13	1	1	0
14	1	1	0
15	1	1	0

SWBUILD

File Operation Distribution

Operation	%	Operation	%
read	0	read file	6
mmap read	0	rand read	0
write	0	write file	7
mmap write	0	rand write	0
rmw	0	append	0
mkdir	1	rmdir	0
readdir	2	create	1
unlink	2	unlink2	0
stat	70	access	6
rename	0	copyfile	0
locking	0	chmod	5
statfs	0	pathconf	0

Thresholds

Threshold	%	Threshold	Value
proc oprate	75	proc latency	n/a
global oprate	95	global latency	n/a
workload variance	n/a		

Execution Parameters

Parameter	Value	Parameter	Value
Procs	5	Dirs per proc	50
Oprate per proc	100	Files per dir	100
Avg file size			16 KiB

Miscellaneous

Option	Value	Option	Value
write commit %	33	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate throttle	0
LRU	0	init read flag	1
release version	1		

Access Patterns

Option	Value	Option	Value
rand dist behavior	0	% per spot	0
min acc per spot	0	access mult spot	5
affinity %	0	spot shape	0
geometric %	10	align	0

Content Patterns

Option	Value	Option	Value
dedup %	0	dedup within %	0
dedup across %	0	dedup group count	1
dedup granule size	4096	dedup gran rep limit	100
compress %	80	comp granule size	8192
cipher flag	0	pattern version	1

Read Transfer Size Distribution

Slot	Start	End	%
0	1	511	1
1	512	1023	5
2	1024	2047	7
3	2048	4095	7
4	4096	4096	0
5	4096	8191	45
6	8192	8192	0
7	8192	16383	13
8	16384	16384	0
9	16384	32767	3
10	32768	65535	2
11	65536	65536	0
12	98304	98304	0
13	65536	131072	17
14	262144	262144	0
15	524288	524288	0

Write Transfer Size Distribution

Slot	Start	End	%
0	1	511	5
1	512	1023	3
2	1024	2047	10
3	2048	4095	15
4	4096	4096	0
5	4096	8191	14
6	8192	8192	0
7	8192	16383	7
8	16384	16384	0
9	16384	32767	6
10	32768	65535	4
11	65536	131072	36
12	98304	98304	0
13	131072	131072	0
14	262144	262144	0
15	524288	524288	0

VDA1

File Operation Distribution

Operation	%	Operation	%
read	0	read file	0
mmap read	0	rand read	0
write	100	write file	0
mmap write	0	rand write	0
rmw	0	append	0
mkdir	0	rmdir	0
readdir	0	create	0
unlink	0	unlink2	0
stat	0	access	0
rename	0	copyfile	0
locking	0	chmod	0
statfs	0	pathconf	0

Thresholds

Threshold	%	Threshold	Value
proc oprate	75	proc latency	n/a
global oprate	95	global latency	n/a
workload variance	5		

Execution Parameters

Parameter	Value	Parameter	Value
Procs	1	Dirs per proc	1
Oprate per proc	9	Files per dir	1
Avg file size			1 GiB

Miscellaneous

Option	Value	Option	Value
write commit %	5	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate throttle	0
LRU	0	init read flag	1
release version	1		

Access Patterns

Option	Value	Option	Value
rand dist behavior	0	% per spot	0
min acc per spot	0	access mult spot	5
affinity %	0	spot shape	0
geometric %	0	align	0

Content Patterns

Option	Value	Option	Value
dedup %	0	dedup within %	0
dedup across %	0	dedup group count	1
dedup granule size	4096	dedup gran rep limit	100
compress %	0	comp granule size	8192
cipher flag	0	pattern version	1

Read Transfer Size Distribution

Slot	Start	End	%
0	1	511	0
1	512	1023	0
2	1024	2047	0
3	2048	4095	0
4	4096	4096	0
5	4097	8191	0
6	8192	8192	0
7	8193	16383	0
8	16384	16384	0
9	16385	32767	0
10	32768	32768	0
11	65536	65536	15
12	131072	131072	10
13	262144	262144	20
14	524288	524288	35
15	1048576	1048576	20

Write Transfer Size Distribution

Slot	Start	End	%
0	1	511	0
1	512	1023	0
2	1024	2047	0
3	2048	4095	0
4	4096	4096	0
5	4097	8191	0
6	8192	8192	0
7	8193	16383	0
8	16384	16384	0
9	16385	32767	0
10	32768	32768	5
11	65536	65536	10
12	131072	131072	10
13	262144	262144	25
14	524288	524288	25
15	1048576	1048576	25

VDA2

File Operation Distribution

Operation	%	Operation	%
read	5	read file	0
mmap read	0	rand read	84
write	0	write file	0
mmap write	0	rand write	0
rmw	2	append	0
mkdir	0	rmdir	0
readdir	3	create	1
unlink	1	unlink2	0
stat	2	access	2
rename	0	copyfile	0
locking	0	chmod	0
statfs	0	pathconf	0

Thresholds

Threshold	%	Threshold	Value
proc oprate	75	proc latency	n/a
global oprate	95	global latency	n/a
workload variance	5		

Execution Parameters

Parameter	Value	Parameter	Value
Procs	1	Dirs per proc	1
Oprate per proc	1	Files per dir	1
Avg file size			1 GiB

Miscellaneous

Option	Value	Option	Value
write commit %	0	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate throttle	0
LRU	0	init read flag	1
release version	1		

Access Patterns

Option	Value	Option	Value
rand dist behavior	0	% per spot	0
min acc per spot	0	access mult spot	5
affinity %	0	spot shape	0
geometric %	0	align	0

Content Patterns

Option	Value	Option	Value
dedup %	0	dedup within %	0
dedup across %	0	dedup group count	1
dedup granule size	4096	dedup gran rep limit	100
compress %	0	comp granule size	8192
cipher flag	0	pattern version	1

Read Transfer Size Distribution

Slot	Start	End	%
0	1	511	0
1	512	1023	0
2	1024	2047	0
3	2048	4095	0
4	4096	4096	0
5	4097	8191	0
6	8192	8192	0
7	8193	16383	0
8	16384	16384	0
9	16385	32767	0
10	32768	32768	0
11	65536	65536	15
12	131072	131072	10
13	262144	262144	20
14	524288	524288	35
15	1048576	1048576	20

Write Transfer Size Distribution

Slot	Start	End	%
0	1	511	0
1	512	1023	0
2	1024	2047	0
3	2048	4095	0
4	4096	4096	0
5	4097	8191	0
6	8192	8192	0
7	8193	16383	0
8	16384	16384	0
9	16385	32767	0
10	32768	32768	5
11	65536	65536	10
12	131072	131072	10
13	262144	262144	25
14	524288	524288	25
15	1048576	1048576	25

VDI

File Operation Distribution

Operation	%	Operation	%
read	6	read file	0
mmap read	0	rand read	20
write	9	write file	0
mmap write	0	rand write	64
rmw	0	append	0
mkdir	0	rmdir	0
readdir	0	create	0
unlink	0	unlink2	0
stat	0	access	1
rename	0	copyfile	0
locking	0	chmod	0
statfs	0	pathconf	0

Thresholds

Threshold	%	Threshold	Value
proc oprate	75	proc latency	n/a
global oprate	90	global latency	n/a
workload variance	n/a		

Execution Parameters

Parameter	Value	Parameter	Value
Procs	2	Dirs per proc	1
Oprate per proc	100	Files per dir	1
Avg file size			500 MiB

Miscellaneous

Option	Value	Option	Value
write commit %	100	background	0
% direct	100	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate throttle	0
LRU	0	init read flag	1
release version	1		

Access Patterns

Option	Value	Option	Value
rand dist behavior	0	% per spot	0
min acc per spot	0	access mult spot	5
affinity %	0	spot shape	0
geometric %	90	align	0

Content Patterns

Option	Value	Option	Value
dedup %	0	dedup within %	0
dedup across %	0	dedup group count	1
dedup granule size	4096	dedup gran rep limit	100
compress %	60	comp granule size	8192
cipher flag	0	pattern version	1

Read Transfer Size Distribution

Slot	Start	End	%
0	512	512	1
1	2048	2048	1
2	2560	3584	1
3	4096	4096	20
4	4608	7680	1
5	8192	8192	4
6	8704	15872	4
7	16384	16384	42
8	16896	32256	3
9	32768	32768	14
10	33280	64024	1
11	65536	65536	6
12	66048	126976	1
13	131072	131072	1
14	262144	262144	0
15	524288	524288	0

Write Transfer Size Distribution

Slot	Start	End	%
0	512	512	21
1	1024	1024	2
2	2048	2048	1
3	4096	4096	47
4	8192	8192	6
5	8704	15360	3
6	16384	16384	5
7	16896	30720	5
8	32768	32768	1
9	35328	64000	3
10	65536	65536	2
11	69632	126976	1
12	131072	131072	3
13	262144	262144	0
14	524288	524288	0
15	1048576	1048576	0

WORKLOAD NAME

File Operation Distribution

Operation	%	Operation	%
read		read file	
mmap read		rand read	
write		write file	
mmap write		rand write	
rmw		append	
mkdir		rmdir	
readdir		create	
unlink		unlink2	
stat		access	
rename		copyfile	
locking		chmod	
statfs		pathconf	

Thresholds

Threshold	%	Threshold	Value
proc oprate		proc latency	
global oprate		global latency	
workload variance			

Execution Parameters

Parameter	Value	Parameter	Value
Procs		Dirs per proc	
Oprate per proc		Files per dir	
Avg file size			

Miscellaneous

Option	Value	Option	Value
write commit %		background	
% direct		sharemode	
% osync		uniform size dist	
% notification		init rate throttle	
LRU		init read flag	
release version			

Access Patterns

Option	Value	Option	Value
rand dist behavior		% per spot	
min acc per spot		access mult spot	
affinity %		spot shape	
geometric %		align	

Content Patterns

Option	Value	Option	Value
dedup %		dedup within %	
dedup across %		dedup group count	
dedup granule size		dedup gran rep limit	
compress %		comp granule size	
cipher flag		pattern version	

Read Transfer Size Distribution

Slot	Start	End	%
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

Write Transfer Size Distribution

Slot	Start	End	%
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			