SPECstorage™ Solution 2020 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

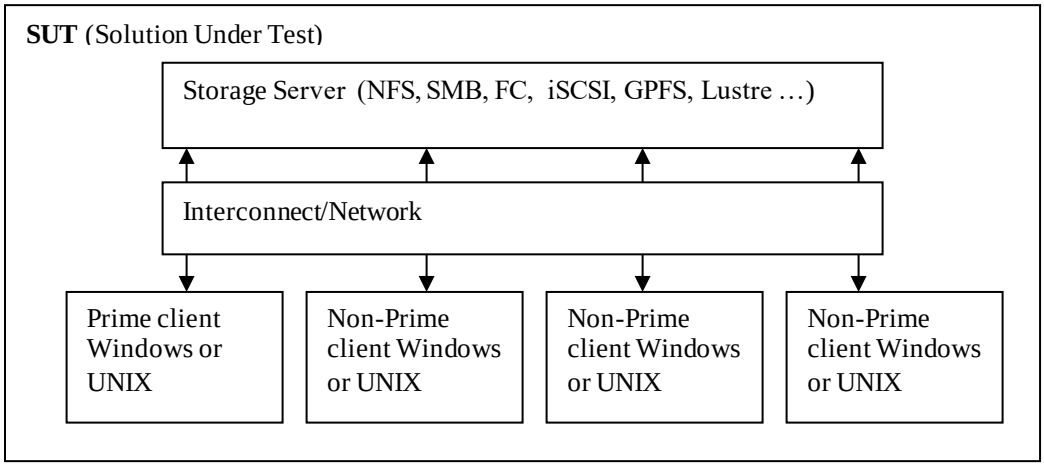
Table of Contents

Overview of Benchmark

The SPECstorage Solution 2020 benchmark is used to measure the maximum sustainable throughput that a storage solution can deliver. The benchmark consists of multiple workloads which represent real data processing file system environments. Each of these workloads are run independently to generate a benchmark publication. In a typical test configuration, a group of load generating clients are directed through a network at file systems shared or exported from a file server, or a cluster of servers. 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. But typically, the benchmark is run using multiple load generators to measure network storage performance. Because this tool runs at the application system call level, it is file system type agnostic and provides strong portability across operating systems and storage solutions. The SPECstorage Solution 2020 benchmark provides pre-compiled binaries for many popular operating systems including Linux, Windows 10, Windows Server 2012R2, Windows Server 2016, Windows Server 2019, Mac OS X, BSD, Solaris, and AIX, and can be used to test any of the file system types that these systems offer that are POSIX compatible.

Many improvements have been made when compared to the previous version of this benchmark. It scales better in distributing load generator work, has more efficient logging capabilities, and faster start-up, and shutdown times.

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

1 Quick Start Guide

These quick start procedures use the pre-compiled C code binaries shipped with the benchmark and assume a homogeneous network of all UNIX-compatible clients or all Windows clients.

1.1 Running the Benchmark for the First Time

We will describe how to set up a simple benchmark run similar to this example in the following sections. This quick start section is intended to get the new user acquainted with the basic configuration of the benchmark.

The minimal configuration consists of:

- One management client, referred to as the Prime Client
- One load generating client

- One POSIX-compliant file system (for example a local file system, NFS, SMB, iSCSI, FCP, GPFS, or Lustre server)

For UNIX-compatible systems the Prime Client may also be a load generator. For Windows-based environments, a separate prime client is required for a minimal configuration.

The steps to produce a SPECstorage Solution 2020 result are:

- Install the SPECstorage Solution 2020 benchmark on the load generators and the prime client
- Edit the sfs_rc configuration file on the prime client
- Configure the solution for testing
- Start the benchmark
- Monitor execution
- Inspect results
- Produce report

Compared to prior releases, the sfs_rc file is now focused on the user and site specific variables that the user needs to modify as part of the setup for their environment. Workloads and their parameters are defined in a separate file (storage2020.yml)

1.2 Example: Official Benchmark Run – SWBUILD

In this example there are 2 clients, each running CentOS7 with a single mountpoint /mnt/Raid5/test. The user starts with 1 BUILDS and generates 10 load points while incrementing the number of BUILDS by 1 for each load point. This creates a curve with 10 data points uniformly distributed. The sfs_rc minimally contains the following:

```
BENCHMARK=SWBUILD          # Choose your workload - SWBUILD is the lightest weight workload
LOAD=1                     # Set your starting LOAD at the lowest value '1'
INCR_LOAD=1                # Set the load increment between points to the LOAD value
NUM_RUNS=10                # of '1' and set number of points to '10' to get ten point curve
# The client names appear before the ':' telling the prime client which to use as load generators
# The directory path to use follws the ':'
CLIENT_MOUNTPOINTS=client1:/mnt/Raid5/test client2:/mnt/Raid5/test
EXEC_PATH=/usr/local/bin/netmist
USER=spec
NETMIST_LICENSE_KEY=12345
NETMIST_LICENSE_KEY_PATH=/tmp/netmist_license_key
```

You do not need to specify any other values in the sfs_rc file to make a benchmark run.

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]$ python3 SM2020 -r sfs_rc -s swbuild
```

or

```
[prime client]$./SM2020 -r sfs_rc -s swbuild
```

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

File	Description
sfslog_swbuild.log	Overall log file
sfssum_swbuild.txt	Summary file in text format
sfssum_swbuild.xml	Summary file in XML format
sfsc001.swbuild	Detailed results for client

1.3 Prerequisites

- Python version 3.8.2 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
- PyYAML must be installed on the Prime client.
- 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 on UNIX systems may also present load. When running on Windows, the prime client cannot also generate load, so at least one additional client is required.
- If one is going to run with a heterogeneous set of clients, for example Prime is Linux and remote nodes are Windows, then one must install and configure Openssh on all of the Windows nodes.

The contents of the SPECstorage Solution 2020 benchmark distribution must be accessible on all the systems where the benchmark will be installed.

1.4 Installing the SPECstorage Solution 2020 Benchmark

The SPECstorage Solution 2020 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.

1.4.1 *Platform common 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!
- Install Python 3.8.2 or later and ensure that python is in the user's search path. Python may be downloaded from http://www.python.org.
- PyYAML installation on the prime client. On some systems the yum install will not work properly with python3. If this happens, use `pip3 install PyYAML` and possibly `pip3 install libyaml-dev`

1.4.2 *UNIX client installation and configuration:*

- Ensure that any/all forms of firewalls are disabled on all of the clients. For example:

```
systemctl stop firewalld
systemctl disable firewalld
setenforce 0
sed -i s/^SELINUX=.*$/SELINUX=disabled/ /etc/selinux/config
```

  You may have to disable iptables also.
- All clients must have a /etc/security/limits.conf entry to increase the maximum number of files per user. Enter these values into the file:

```
root            -       nproc           10000
root            -       nofile          10000
```

- 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. This can be exceptionally challenging in Windows versions that support Openssh.
- Install the SPECstorage Solution 2020 benchmark using the following steps:
  - Login to the client (as root)
  - Download or copy the SPECstorage Solution 2020 ISO or archive to the client
  - Loop mount the ISO or expand the archive
  - cd to the top level directory containing the SPECstorage Solution 2020 contents
- Enter `python3 SM2020 --install-dir=`*`destination_directory`* (where *`destination_directory`* is the full path where you wish to have the benchmark installed)

1.4.3 *Windows client installation and configuration:*

- Ensure that all Windows clients are properly configured in the Active Directory Domain.
- Install the SPECstorage Solution 2020 benchmark
  - Start a command prompt window. This can be done using the 'Start' button, choosing 'Run…' and entering 'cmd'.
  - Download or copy the SPECstorage Solution 2020 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 SPECstorage Solution 2020 directory
  - Enter `python SM2020 --install-dir=`*`destination_directory`* (where *`destination_directory`* 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> be problematic**.**

1.5 Editing the configuration file on the prime client

There is one rc file for every benchmark (or workload) that you wish to run. The default configuration file is called sfs_rc. You should copy and edit this file for your benchmark run. These rc files must reside on the prime client. The user does not need to edit, or even have, a configuration file on the other load generating clients. You may want to name your rc files descriptively. The user must edit the rc configuration file as the defaults must be edited to represent your environment

For a quick start run, on the prime client, the following values **must** be specified:

- **BENCHMARK=<benchmark name>**
  Name of the benchmark to run. Valid values are: SWBUILD, VDA, EDA_BLENDED, AI_IMAGE, and GENOMICS
- **LOAD=<integer starting number | integer list of load points>**
  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 space separated list of values, increasing 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**. If a list of values if specified, at least 10 uniformly spaced data points must be specified for a benchmark result intended for submission. For more detail on the requirements for uniformly spaced data points, see section 5.3 "Data Point Specification for Results Disclosure" in the SPECstorage® Solution 2020 Run and Reporting Rules.
- **INCR_LOAD=<integer>**
  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=<integer>**
  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=<mountpoint mountpoint etc>**
  The list of local mount points, local directories, or shares, to use in the testing. 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 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. The file by default is found in the same directory as the sfs_rc file, unless specified with a full pathname.
- **EXEC_PATH=<pathname>**
  The full path to the SPECstorage Solution 2020 executable. Such as /usr/local/spec/netmist Currently the executable is called *netmist* for POSIX systems and *netmist.exe* for Windows systems.
- **USER=<user name>**
  The user account name, which must be configured on all clients, to be used for the benchmark execution. In homogeneous configurations this identifies the user account use for running the benchmark. In a UNIX environment use a simple common username for all load generators (user33). In a Windows environment, prefix the user with the domain name separated by a backslash (DOMAIN\User33).
- **IPV6_ENABLE=<0 or 1>**
  Set to "1" or "Yes" when the benchmark should use IPv6 to communicate with other benchmark processes. It defaults to '0'.
- **PASSWORD=<password>**
  The password for the user specified in USER. Used for Windows configurations, whether homogeneous or heterogeneous.
- **NETMIST_LICENSE_KEY=<integer value>**
  License code obtained from SPEC office at purchase time, a simple number.
- **NETMIST_LICENSE_KEY_PATH= <pathname>**
  This file gets initialized with the value specified in the NETMIST_LICENSE_KEY variable when the benchmark runs. The path to the license key file. Example: /tmp/netmist_license_key or C:\tmp\netmist_license_key.

Pro Tip: If this is your first time running the benchmark use one load generating client (which means two clients in the case of Windows as the Prime Client cannot be a load generator). And run the SWBUILD workload regardless of what workload you will eventually run. This will simplify debugging your test setup configuration. Other workloads are more demanding on the client and require much more space per business metric to run.

1.6 Configuring the storage solution for testing

- Mount all working directories on the clients (UNIX only). The path names must match the values specified in the CLIENT_MOUNTPOINTS parameter in the SPECstorage Solution 2020 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)

1.7 Starting the benchmark

The SM2020 python script is run on the Prime Client and allows the user to input parameters, run the benchmark, and review the results for all workloads. The script is executable and may or may not work without typing python3 SM2020.

- Change directories to the *destination_directory* specified during install.
- On the Prime client enter '`python3 SM2020 -r sfs_config_file -s output_files_suffix`'

1.8 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:

sfsc*.*	The client log files.

Pro tip:

If the benchmark terminates abnormally, there are also logs on each client in /tmp/netmist_* that contain additional information about any errors that were found.

1.9 Examining the results after the benchmark execution has completed

The results of the benchmark are summarized in the sfssum.* files in the results 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.

1.10 Where Do You Go From Here?

For any reasonable storage solution used in an enterprise data center, the number of load generators required to measure peak load of the storage server numbers in the tens of clients. The benchmark efficiently uses clients to generate a load. But the principles above apply to a larger test setup.

Refer to Appendix A – Building the SPECstorage 2020 Benchmark Components for detailed information on how to build the benchmark, set advanced capability parameters, customize a run and configure a heterogeneous setup of UNIX and Windows clients. See section 3.3 Configuring Required Benchmark Parameters and section 5. Submission and Review Process details configuring the benchmark for and generating an official results submission.

2 SPECstorage Solution 2020 Benchmark

The SPECstorage Solution 2020 benchmark is used to measure the maximum sustainable throughput that a storage solution can deliver. As mentioned before, the benchmark is protocol independent and will generate load over any version of NFS or SMB, clustered file systems, object oriented file systems, local file systems, or any other POSIX compatible file system. This provides strong portability across operating systems, and storage solutions.

2.1 Simulation of Workloads

The SPECstorage Solution 2020 workloads are a mixture of file meta-data and data oriented operations. The SPECstorage Solution 2020 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 measure 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 can be measured in a workload:

Netmist Operation	Description
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
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
neg_stat	Perform a stat() on non-existent files
truncate	Truncate a file to a new size.

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() calls perform a whole file operation.

The results of the benchmark are: <should this section be expanded, with example?>

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.

These are the primary metrics and the minimum that must be stated in public disclosures:

1. Peak aggregate Ops/sec
2. Overall Response Time (ORT)

2.2 SPECstorage Solution 2020 Workloads and Business Metrics

The SPECstorage Solution 2020 benchmark includes multiple workloads each with a business metric for reported results as shown in the following table:

Workload	Description	Business Metric
SWBUILD	Software build	BUILDS
VDA	Video data acquisition	STREAMS
EDA_BLENDED	Electronic design automation	JOBS
AI_IMAGE	AI Image processing	JOBS
GENOMICS	Genomic processing	JOBS

The user has the option to submit results using any or all of the above workloads.

The SM2020 Python script accepts benchmark run configuration information, starts benchmark execution, and collects results from the SPECstorage Solution 2020 benchmark.

2.3 Resource Requirements

As a helpful guideline, here are some rules of thumb for the resources required per business metric for the different workloads.

Storage target capacity requirements per business metric	
SWBUILD	5 GiB per BUILD
VDA	24 GiB per STREAM
EDA_BLENDED	11 GiB per JOB
AI_IMAGE	100 GiB per JOB
GENOMICS	3.5 GiB per JOB

Client memory requirements per business metric:	
SWBUILD	650 MiB per BUILD
VDA	100 MiB per STREAM
EDA_BLENDED	520 MiB per JOB
AI_IMAGE	1.7 GiB per JOB
GENOMICS	416 MiB per JOB

2.4 Scale

Scaling in SPECstorage Storage 2020 is expected to be somewhere around 4 Million load generating processes globally, where the load generators may be geographically distributed around the planet.

There is a sophisticated synchronization mechanism that keeps all of the geographically distributed load generating processes in sync, at a sub milli-second resolution.

2.5 SM2020 – the Benchmark Manager

The benchmark manager is called SM2020. It is a Python program that requires Python version 3.8.2 or higher. It is recommended to get Python from http://www.python.org/

You can get the syntax by running:

```
% python3 ./bin.in/dist_pro/SM2020 -h
```

```
        Usage: python SM2020 [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=sfs2020)
        [-b <file>] or [--yaml-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 SM2020 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.
#
########################################################################

########################################################################
# Legacy settings for current sfs_rc files
# Used for Unix only, or Windows only environments.
########################################################################
#
# Example for all Unix:
# CLIENT_MOUNTPOINTS=clientname:/mnt/testdir
# USER=spec
# PASSWORD=MYpa55w0rd
# EXEC_PATH=/usr/local/bin/netmist
# NETMIST_LOGS=
# NETMIST_LICENSE_KEY= <License number>
# NETMIST_LICENSE_KEY_PATH=/tmp/netmist_license_key
# PDSM_MODE=
# PDSM_INTERVAL=
# UNIX_PDSM_LOG=
# UNIX_PDSM_CONTROL=
# EXEC_PATH=/usr/local/bin/netmist
# ------------------------------------------------------------------------
# Example for all Windows:
# CLIENT_MOUNTPOINTS=clientname:\\servername\share\testdir
# USER=mydomain\spec
# PASSWORD=MYpa55w0rd
```

```
# EXEC_PATH=C:\\tmp\\netmist.exe
# NETMIST_WINDOWS_LOGS=
# NETMIST_LICENSE_KEY= <License number>
# NETMIST_LICENSE_KEY_PATH=C:\tmp\netmist_license_key
# PDSM_MODE=
# PDSM_INTERVAL=
# WINDOWS_PDSM_LOG=
# WINDOWS_PDSM_CONTROL=
# EXEC_PATH=C:\\tmp\\netmist.exe
# USER=domain\account
# PASSWORD=MyPa55w0rd
################################################################################

CLIENT_MOUNTPOINTS=
USER=
PASSWORD=

################################################################################
# In Legacy Unix only mode with a Unix prime this must be set to a Unix path.
# In Legacy Windows only mode, EXEC_PATH must be set to a Windows path.
# In heterogeneous modes EXEC_PATH must be set to a Unix path.
#  **** This field cannot be empty ****
################################################################################
EXEC_PATH=

################################################################################
# Legacy mode + heterogeneous client types:
# These client lists are used in the legacy mode to provide heterogeneous
# client support.
# This allows heterogeneous clients sharing same legacy workload.
# This mode ignores the platform_type field in the YAML config file.
# For these lists:
# 1) Both lists have to be populated
# 2) Corresponding *_EXEC_PATH, *_USER, *_PASSWORD need to be present
################################################################################

UNIX_CLIENT_LIST=
WINDOWS_CLIENT_LIST=

################################################################################
# Heterogeneous clients with their heterogeneous OS types specified in the YAML
# "platform_type" fields.
################################################################################
# For mixed OS workload tests:
# 1) change "platform_type" in YAML to the appropriate OS in each workload
#    component
# 2) Provide both Windows and Unix mount-points
#    < WINDOWS_CLIENT_MOUNTPOINTS= and UNIX_CLIENT_MOUNTPOINTS= >
#    Important: Both OSs should have exactly same number of mount-points
# 3) Set Prime path "EXEC_PATH" to UNIX path. Prime in this mode must be
#    a UNIX system.
#------------------------------------------
# UNIX Settings for heterogeneous client pool
#
# Example:
#
# UNIX_CLIENT_MOUNTPOINTS=client1:/mnt/testdir
# UNIX_EXEC_PATH=/usr/local/bin/netmist
# UNIX_USER=spec
# NETMIST_LOGS=
# NETMIST_LICENSE_KEY= <License number>
# NETMIST_LICENSE_KEY_PATH=/tmp/netmist_license_key
# PDSM_MODE=
# PDSM_INTERVAL=
# UNIX_PDSM_LOG=
# UNIX_PDSM_CONTROL=
#------------------------------------------
UNIX_CLIENT_MOUNTPOINTS=
UNIX_EXEC_PATH=
```

```
UNIX_USER=
#-----------------------------------------
#
# Windows Settings for heterogeneous client pool
#
# Example:
# WINDOWS_CLIENT_MOUNTPOINTS=WIN10M1:\\smbserver\test
# WINDOWS_EXEC_PATH=C:\\tmp\\netmist.exe
# WINDOWS_USER=labnet\\administrator
# WINDOWS_PASSWORD=MYpa55w0rd
# NETMIST_WINDOWS_LOGS=
# NETMIST_LICENSE_KEY= <License number>
# NETMIST_LICENSE_KEY_PATH=C:\tmp\netmist_license_key
# PDSM_MODE=
# PDSM_INTERVAL=
# WINDOWS_PDSM_LOG=
# WINDOWS_PDSM_CONTROL=
#-----------------------------------------
WINDOWS_CLIENT_MOUNTPOINTS=
WINDOWS_EXEC_PATH=
WINDOWS_USER=
WINDOWS_PASSWORD=
###############################################################################


###############################################################################
#
# Common Settings for all modes
#
# Official BENCHMARK values are
#       -SWBUILD
#       -VDA
#       -EDA_BLENDED
#       -AI_IMAGE
#       -GENOMICS
#
###############################################################################

BENCHMARK=SWBUILD
LOAD=1
INCR_LOAD=1
NUM_RUNS=1

IPV6_ENABLE=0
PRIME_MON_SCRIPT=
PRIME_MON_ARGS=
NETMIST_LOGS=
NETMIST_WINDOWS_LOGS=
NETMIST_LICENSE_KEY=
NETMIST_LICENSE_KEY_PATH=/tmp/netmist_license_key

###############################################################################
#
#  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.
#
###############################################################################
MAX_FD=
LOCAL_ONLY=0
FILE_ACCESS_LIST=0

# PDSM_MODE of 0 create shared PDSM log file with overwrites.
# PDSM_MODE of 1 create shared PDSM log file with open append.
PDSM_MODE=
# PDSM_INTERVAL # Interval in seconds
```

```
PDSM_INTERVAL=
# <PLATFORM>_PDSM_LOG filename  Full pathname to the PDSM log file. Used to
# log the details of every proc's activities.
UNIX_PDSM_LOG=
WINDOWS_PDSM_LOG=
# <PLATFORM>_PDSM_CONTROL filename  Full pathname to the PDSM control file. Used
# for dynamically changing workloads.
UNIX_PDSM_CONTROL=
WINDOWS_PDSM_CONTROL=
```

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, INCR_LOAD, NUM_RUNS, 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"

2.5.1 *The Summary Files*

Each run will produce two 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)

3 Installing and Configuring the Benchmark Environment

This section provides detailed information on hardware/software configuration requirements for the load generators and the storage solutions. It also includes detailed 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)

The Solution Under Test (SUT) typically consists of the load generators, the network, the file server(s), and finally the stable backend storage. Results are reported and published for the entire configuration. More details of the SUT may be found in section 6.4 of the SPECstorage Solution 2020 Run and Reporting Rules document.

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. Refer to section 3.3 Resource Requirements for rules of thumb. 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 hard 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.
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 SPECstorage® Solution 2020 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 function correctly. 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. Your solution or Solution Under Test (SUT) consists of the load generating clients, the network and the storage being measured. 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, dedicating the components of the solution for the duration of the test.
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**. **Before doing this, be sure your resume is up to date**.

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 SM2020 Python script requires that the Python 3.8.2 and PyYAML be installed.

SPECstorage Solution 2020 benchmark runs should be done as a non-root user. On UNIX systems, for example, create a "spec" user.

The SPECstorage Solution 2020 binaries must be installed on clients. There are a couple of methods to install the SPECstorage Solution 2020 binaries:

On all the clients:

1. Login as "root"
2. Change directory to the top level directory containing the SPECstorage Solution 2020 benchmark files
3. Enter 'python3 SM2020 –install-dir="destination_directory"'

Alternately, just copy the SPECstorage Solution 2020 kit to all clients using any feasible method.

Verify that all clients, including prime, have the SPECstorage Solution 2020 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. See: NETMIST_LOGS= in the sfs_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.

Also, one should disable the Windows Defender's Virus and thread protection → "Real time protection" checking. Failure to do this will result in very high CPU usage on the client during the measurement, and potentially lower results.

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.

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. For more details, see section 3.3 "Configuring the Required Benchmark Parameters" below.

While the ideal way to order the CLIENT_MOUNTPOINTS list for a given SUT will depend heavily on its architecture, a first approach as mentioned above in the Running the Benchmark for the First Time Running the Benchmark for the First Time section would be to uniformly (round robin) specify the clients and the mount points.

3.2.1 *Configuring SPECstorage Solution 2020 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 netmist 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 Required 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 SPECstorage Solution 2020 Run Rules document for the classification of all the parameters.

The parameters you must set are:

1. **BENCHMARK=<name>**
   The name of the workload to run. SWBUILD, VDA, EDA_BLENDED , AI_IMAGE, or GENOMICS.
2. **CLIENT_MOUNTPOINTS=<mountpoint mountpoint etc | file pathname containing list of 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" (note there is no ':' in the "client_name path" pairs). 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.
4. **EXEC_PATH=<pathname>**
   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.
   UNIX_EXEC_PATH=<pathname>
   The absolute path to the benchmark executable used for remote load generators that are Unix based when running in a heterogeneous configuration of Windows and Unix client.

   E.g. /usr/local/bin/netmist

   WINDOWS_EXEC_PATH=<pathname>
   The absolute path to the benchmark executable used for remote load generators that are Windows based. Only used for heterogeneous configurations of Windows and Unix clients.

   E.g. C:\\tmp\\netmist.exe

   In heterogeneous mode, where there are a mixture of Unix and Windows clients, then the EXEC_PATH is used only by the Prime Client (and must be set), the client load generators will use the appropriate UNIX_EXEC_PATH or WINDOWS_EXEC_PATH variables.
5. **USER=<string>**
   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.
   UNIX_USER=<string>
   In heterogeneous mode, where this is a mixture of Unix and windows clients, then the UNIX_USER and WINDOWS_USER variables are used.
   WINDOWS_USER=<string>
   The user domain and account for the Windows account to be used to execute the benchmark. Only used for heterogeneous configurations of Windows and Unix clients. E.g. domain\administrator
6. **PASSWORD=<string>**
   Set this to the account password for running the benchmark (Windows clients only). In heterogeneous mode, the UNIX_PASSWORD and WINDOWS_PASSWORD variables are used.

3.3.1 *Configuring Other Parameters in the RC File*

In addition to the parameters required to be changed for a run (described in section Quick Start Guide), 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 SPECstorage Solution 2020 benchmark will execute during the various phases of the benchmark. It only runs on the Prime Client. 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 SPECstorage Solution 2020 source directory. For a disclosable run, this program/script must be performance neutral and its actions must comply with the SPECstorage Solution 2020 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. *The script must complete quickly, or be non-block (run processes in the background) for the benchmark to execute correctly.*

a. **PRIME_MON_SCRIPT=<pathname>**
   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 SPECstorage Solution 2020 Run and Reporting Rules. If this option is used, the executable or script used must be disclosed and 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. For 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.
b. **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.

2. **NETMIST_LOGS**
   Used to set a non-default location for the netmist_C*.log files. /tmp/ or c:\tmp\ are used by default.
   a. **NETMIST_LOGS=<directory pathname for log files>**
      Set the path to the directory in which to store log files from the load generators. The same path will be used on all Unix clients. If this path is not set, /tmp/ will be used.
   b. **NETMIST_WINDOWS_LOGS=<directory pathname for log files>**
      Set the path to the directory in which to store the log files from the load generators. The same path will be used by all of the Windows clients. If this path is not set, c:\tmp\ will be used.
3. **IPV6_ENABLE**
   Flag to set for when the benchmark should use IPv6 to communicate with other benchmark processes.
4. **MAX_FD***
   Sets the maximum number of file descriptors each proc can use.
5. **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.
6. **FILE_ACCESS_LIST***
   If enabled, the benchmark will dump a list of all files accessed during each load point.

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

Note: For previous SPECsfs2014 benchmark users, the WARMUP_TIME parameter is now in the storage2020.yml file.

3.4 Detailed 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.

4 Running the Benchmark and Interpreting Results

This section contains information on the SPECstorage Solution 2020 benchmark directory structure, running the benchmark, and interpreting the benchmark metrics output generated in the summary results file.

4.1 SPECstorage Solution 2020 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 SPECstorage Solution 2020 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 SM2020 Python script 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 SPECstorage Solution 2020 benchmark.
5. $SPEC/msbuild
   Contains the Microsoft Visual Studio Community 2015 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 SPECstorage Solution 2020 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 SPECstorage Solution 2020 Benchmark Binaries

The SPECstorage Solution 2020 benchmark includes pre-compiled binaries for a large number of supported platforms and architectures.

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, RHEL 8
- Cent OS

  - CentOS 7, CentOS 8.
- Apple Inc.
  - Mac OS X
- Microsoft Corporation
  - Windows 10, Windows Server 2012R2, Windows Server 2016, Windows Server 2019

4.3 Building the SPECstorage Solution 2020 Benchmark

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). For a valid submission, 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. Modifications must be disclosed and reviewed and accepted by the SPEC Storage subcommittee before use in a publication.

There are additional prerequisites to build the benchmark. If one is trying to use the standard "make" described below, then it is up to the developer to install all necessary packages to support building libyaml. (make, gcc, autom4te, automake, autoconf, libtool, m4).

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 SPECstorage Solution 2020 benchmark can be built with Visual Studio C++ 2015 Express. See section 10.2 Building the SPECstorage Solution Benchmark for Windows for the build instructions for Visual Studio builds.

4.4 Using SM2020

The SM2020 Python script 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.

4.4.1 *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.

4.4.2 *Example of a Benchmark Run*

Example of run with only one load point for brevity.

```
<<< Thu Aug 13 12:42:43 2020: Starting SWBUILD run 1 of 1: BUILDS=1 >>

     SPECstorage(TM) Solution 2020 Release        $Revision: 2462 $

     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 contains contributions from Iozone.org.

     Copyright (C) 2002-2020, Don Capps
```

```
     All rights reserved.

     This program contains 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
        notices, this list of conditions and the following disclaimer.
     2. Redistributions in binary form must reproduce the above copyright
        notices, 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.
     Results directory: /home/capps/SPECstorage2020/bin/results
     Op latency reporting activated
     Importing workloads from storage2020.yml
[INFO][Thu Aug 13 12:42:43 2020]Exec validation successful

     SPECstorage(TM) Solution 2020 Release        $Revision: 2462 $

     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 contains contributions from Iozone.org.

     Copyright (C) 2002-2020, Don Capps
     All rights reserved.

     This program contains 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
        notices, this list of conditions and the following disclaimer.
     2. Redistributions in binary form must reproduce the above copyright
        notices, 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.
     Results directory: /home/capps/SPECstorage2020/bin/results
     Op latency reporting activated
     Importing workloads from storage2020.yml
2020-08-13 12:42:43.692: Prime          : Starting tests...
2020-08-13 12:42:43.695: Prime          :    Launching 1 nodeManager processes.
2020-08-13 12:42:43.695: Prime          :    Starting test nodeManager:   0 Host: centos1804m1
2020-08-13 12:42:49.898: Prime          : Version check successful: $Revision: 2462 $
2020-08-13 12:42:49.910: Prime          : Waiting for each NodeManager to spawn client processes
2020-08-13 12:42:52.227: Prime          : Starting INIT phase
2020-08-13 12:42:52.230: Prime          : Sending Init to NodeManager 0
2020-08-13 12:42:52.230: Prime          : Waiting to finish initialization.
2020-08-13 12:43:20.233: Prime          : Init 10 percent complete from client 0
2020-08-13 12:44:24.461: Prime          : Init 30 percent complete from client 4
2020-08-13 12:45:15.246: Prime          : Init heartbeat __/\_/\__ client 0
2020-08-13 12:45:31.010: Prime          : Init 40 percent complete from client 4
2020-08-13 12:46:06.872: Prime          : Init 60 percent complete from client 1
2020-08-13 12:46:15.246: Prime          : Init heartbeat __/\_/\__ client 0
2020-08-13 12:46:37.182: Prime          : Init 80 percent complete from client 2
2020-08-13 12:46:53.446: Prime          : Init 90 percent complete from client 1
2020-08-13 12:46:57.472: Prime          : Initialization finished
2020-08-13 12:46:57.474: Prime          : Sending INIT results to NodeManager 0
2020-08-13 12:46:57.567: Prime          : Starting WARM phase
2020-08-13 12:46:57.569: Prime          : Sending warmup to NodeManager 0
2020-08-13 12:46:57.569: Prime          : Waiting for WARMUP phase to finish
2020-08-13 12:47:55.648: Prime          : Warm-up 20 percent complete from client 0
2020-08-13 12:48:55.648: Prime          : Warm-up 40 percent complete from client 3
2020-08-13 12:49:15.288: Prime          : Warm heartbeat client 0: 99.520 Ops/sec
2020-08-13 12:49:25.659: Prime          : Warm-up 50 percent complete from client 2
2020-08-13 12:49:55.657: Prime          : Warm-up 60 percent complete from client 4
2020-08-13 12:50:15.347: Prime          : Warm heartbeat client 4: 99.576 Ops/sec
2020-08-13 12:50:25.678: Prime          : Warm-up 70 percent complete from client 4
2020-08-13 12:50:55.658: Prime          : Warm-up 80 percent complete from client 0
2020-08-13 12:51:15.310: Prime          : Warm heartbeat client 3: 99.983 Ops/sec
2020-08-13 12:51:25.677: Prime          : Warm-up 90 percent complete from client 2
2020-08-13 12:51:58.648: Prime          : Starting RUN phase
2020-08-13 12:51:58.649: Prime          : Waiting for RUN phase to finish
2020-08-13 12:52:26.529: Prime          : Run 10 percent complete from client 2
2020-08-13 12:52:56.930: Prime          : Run 20 percent complete from client 3
2020-08-13 12:53:27.069: Prime          : Run 30 percent complete from client 2
2020-08-13 12:53:57.629: Prime          : Run 40 percent complete from client 1
2020-08-13 12:54:27.740: Prime          : Run 50 percent complete from client 3
2020-08-13 12:54:57.899: Prime          : Run 60 percent complete from client 4
2020-08-13 12:55:28.099: Prime          : Run 70 percent complete from client 4
2020-08-13 12:56:26.619: Prime          : Run 90 percent complete from client 4
```

```
2020-08-13 12:56:57.079: Prime          : Run 100 percent complete from client 1
2020-08-13 12:56:57.649: Prime          : Tests finished
2020-08-13 12:56:57.650: Prime          : Sending Asking results to NodeManager 0
       ---------------------------------------------------------------
       Overall average latency                       0.451 Milli-seconds
       Overall SPECstorage(TM) Solution 2020       500.017 Ops/sec
       Overall Read_throughput                ~   3227.734 Kbytes/sec
       Overall Write_throughput               ~    837.597 Kbytes/sec
       Overall throughput                     ~   4065.331 Kbytes/sec
       Total file space initialized           ~    4800000 Mbytes
       ---------------------------------------------------------------
       Workload                              MD5
       SWBUILD              0xfb8c79f3ac48f9c99f453c482e0c5442
       VDA1                 0x9cbfa68f9db422f6e02bc223beb7147c
       VDA2                 0xc65b6fbf99d4ae7ac2a049d14992765d
       EDA_FRONTEND         0x5fdd3373676e6463e84f44935ac8c9a9
       EDA_BACKEND          0x2286a2138b5b0a1b36b022e64bda0b72
       AI_SF                0xf51719f24e46718977a87336e15d2119
       AI_TF                0xa181505bbf17e674e3b880de5c1ded38
       AI_TR                0x771bc069d10527f982be855e6f3b1d66
       AI_CP                0x7851f91ddf3e9b83d4cbf074cde9498e
       NGS                  0x62bc6fc32ba443fc84da7c426268bbbb
       Registered Finger Print         2881026523
       ---------------------------------------------------------------
       Latency Bands:
       Band 1:  20us:1758     40us:4914     60us:1549     80us:545     100us:529
       Band 2: 200us:2028    400us:61516   600us:35832   800us:13264     1ms:5908
       Band 3:   2ms:5618      4ms:682       6ms:13        8ms:5        10ms:1
       Band 4:  12ms:4        14ms:1        16ms:1        18ms:3        20ms:1
       Band 5:  40ms:1        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

       ---------------------------------------------------------------
2020-08-13 12:56:57.701: Prime          : Client results ready
2020-08-13 12:56:57.702: Prime          : Sending shutdown to NodeManager 0
2020-08-13 12:56:57.703: Prime          : Shutting down
2020-08-13 12:56:57.703: Prime          : Closing sockets
2020-08-13 12:56:57.703: Prime          : Closing keepalive sockets
2020-08-13 12:56:57.703: Prime          : Closing file handles
2020-08-13 12:56:57.703: Prime          : Freeing memory
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 SPECstorage Solution 2020 benchmark release includes a tool for collecting benchmark results in a format that can be submitted by email to the SPECstorage Solution 2020 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 SPECstorage Solution 2020 Run and Reporting Rules. See the SPECstorage Solution 2020 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 SPECstorage Solution

2020 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 SM2020 Python script	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: storage2020-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-dir=<dir>]	Results directory, default is "results" in the current working directory.	Optional
[-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 SPECstorage Solution 2020 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 th e test, the same suffix used with SM2020 Python script. To test the submission files for correctness, issue the command

```
$ python3 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

```
$ python3 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 `substoragesolution2020@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 https://www.spec.org/.

6 Workload Definitions

The following sections summarize the important characteristics of each workload available in the SPECstorage Solution 2020 benchmark. For a complete and definitive definition of each workload please refer to the storage2020.yml file in the distribution.

Some benchmarks are a composite of two or more subcomponent workloads. There are global parameters per workload that apply to all subcomponents. If there are multiple subcomponents to a benchmark, then the global parameters are shown at the "Benchmark_name" level in the storage2020.yml file structure and apply to the entire workload.

6.1 Software Build (SWBUILD) Benchmark

6.1.1 *SWBUILD Workload 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 a 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.

The business metric for the SWBUILD benchmark is BUILDS.

6.1.2 *SWBUILD Workload Definition*

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
lock	0	chmod	5
statfs	0	pathconf	0
trunc	0	neg_stat	0

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		

Global Parameters

Threshold	%	Threshold	Value
proc oprate	75	proc latency	0
global oprate	95	global latency	0
workload variance	0	Dedicated subdir	0
Warmup secs	300	Metric	BUILDS

Miscellaneous

Option	Value	Option	Value
write commit %	33	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate speed	0
LRU	1	init read flag	1
release version	3	fadvise seq %	0
fadvise rand %	0	fadvise don't need %	0

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	2

SWBUILD Part 2

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	8191	45
5	8192	16383	13
6	16384	32767	3
7	32768	65535	2
8	65536	131072	17
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	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	8191	14
5	8192	16383	7
6	16384	32767	6
7	32768	65535	4
8	65536	131072	36
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

6.2 Video Data Acquisition (VDA) Benchmark

6.2.1 *VDA 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 VDA benchmark is STREAMS. The benchmark consists of two subcomponents: VDA1 (data stream) and VDA2 (companion applications). Each stream corresponds to a roughly 36 Mib/s bit rate, which is in the upper range of high definition video.

6.2.2 *VDA1 Workload Definition (subcomponent)*

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
lock	0	chmod	0
statfs	0	pathconf	0
trunc	0	neg_stat	0

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		

Global Parameters

Threshold	%	Threshold	Value
proc oprate	75	proc latency	0
global oprate	95	global latency	0
workload variance	5	Dedicated subdir	0
Warmup secs	300	Metric	STREAMS

Miscellaneous

Option	Value	Option	Value
write commit %	5	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate speed	0
LRU	1	init read flag	1
release version	3	fadvise seq %	0
fadvise rand %	0	fadvise don't need %	0

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	2

VDA1 Part 2

Read Transfer Size Distribution

Slot	Start	End	%
0	65536	65536	15
1	131072	131072	10
2	262144	262144	20
3	524288	524288	35
4	1048576	1048576	20
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

Write Transfer Size Distribution

Slot	Start	End	%
0	32768	32768	5
1	65536	65536	10
2	131072	131072	10
3	262144	262144	25
4	524288	524288	25
5	1048576	1048576	25
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

6.2.3 *VDA2 Workload Definition (subcomponent)*

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
lock	0	chmod	0
statfs	0	pathconf	0

Miscellaneous

Option	Value	Option	Value
write commit %	0	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate speed	0
LRU	1	init read flag	1
release version	3	fadvise seq %	0
fadvise rand %	0	fadvise don't need %	0

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

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		

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	2

VDA2 Part 2

Read Transfer Size Distribution

Slot	Start	End	%
0	65536	65536	15
1	131072	131072	10
2	262144	262144	20
3	524288	524288	35
4	1048576	1048576	20
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

Write Transfer Size Distribution

Slot	Start	End	%
0	32768	32768	5
1	65536	65536	10
2	131072	131072	10
3	262144	262144	25
4	524288	524288	25
5	1048576	1048576	25
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

6.3 Electronic Design Automation (EDA_BLENDED) Benchmark

6.3.1 *EDA_BLENDED Workload Description*

This workload represents the typical behavior of a mixture of EDA applications. EDA applications represent software tools and workflows for designing semiconductor chips. Describes dozen of software tools used to design a chip from specification to fabrication. Represent very compute-heavy processes with high concurrency. Storage is often the performance bottleneck. The benchmark comprise of large numbers of small files with a low percent of large files. Mixed random and sequential IO of metadata operations representing two high level design phases: Frontend design and Backend design. The complete workload is a mixture of two subcomponents: the 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. The business metric for the EDA_BLENDED benchmark is JOBS

6.3.2 *EDA_FRONTEND Workload Definition (subcomponent)*

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
lock	0	chmod	1
statfs	0	pathconf	0
trunc	0	neg_stat	0

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		

Global Parameters

Threshold	%	Threshold	Value
proc oprate	75	proc latency	0
global oprate	95	global latency	0
workload variance	5	Dedicated subdir	0
Warmup secs	300	Metric	JOBS

Miscellaneous

Option	Value	Option	Value
write commit %	15	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate speed	0
LRU	1	init read flag	0
release version	3	fadvise seq %	0
fadvise rand %	0	fadvise don't need %	0

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

EDA_FRONTEND Part Two

Read Transfer Size Distribution

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

6.3.3 *EDA_BACKEND Workload Definition (subcomponent)*

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
lock	0	chmod	0
statfs	0	pathconf	0
trunc	0	neg_stat	0

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 speed	0
LRU	1	init read flag	0
release version	3	fadvise seq %	0
fadvise rand %	0	fadvise don't need %	0

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

EDA_BACKEND Part Two

Read Transfer Size Distribution

Slot	Start	End	%
0	32768	65535	49
1	65536	65536	51
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

Write Transfer Size Distribution

Slot	Start	End	%
0	32768	65535	45
1	65536	131072	55
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

6.4 AI_IMAGE (AI_IMAGE) Benchmark

6.4.1 *AI_IMAGE Workload Description*

This workload is representative of AI Tensorflow image processing environments and is expected to be popular as this market continues to expand. The traces used for the basis were collected from Nvidia DGX based systems running COCO, Resnet50, and CityScape processing.

There are four subcomponents that make up the aggregate AI workload. The first two are the Data Preparation Phase, the second two make up the Training Phase:

- AI_SF: small (image) file ingest
- AI_TF: tensor flow record creation
- AI_TR: training consumption of tensor flow records
- AI_CP: Represents the checkpointing functionality and may occur infrequently, if at all, during a typical run.

The business metric for the AI_IMAGE benchmark is JOBS.

6.4.2 *AI_SF Workload Definition (subcomponent)*

AI_SF

File Operation Distribution

Operation	%	Operation	%
read	37	read file	0
mmap read	0	rand read	0
write	0	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	56	access	7
rename	0	copyfile	0
lock	0	chmod	0
statfs	0	pathconf	0
trunc	0	neg_stat	0

Execution Parameters

Parameter	Value	Parameter	Value
Procs	4	Dirs per proc	3
Oprate per proc	100	Files per dir	200
Avg file size	1 MiB		

Global Parameters

Threshold	%	Threshold	Value
proc oprate	75	proc latency	0
global oprate	95	global latency	0
workload variance	5	Dedicated subdir	1
Warmup secs	900	Metric	JOBS

Miscellaneous

Option	Value	Option	Value
write commit %	0	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate speed	0
LRU	1	init read flag	0
release version	3	fadvise seq %	0
fadvise rand %	0	fadvise don't need %	0

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

AI_SF Part 2

Read Transfer Size Distribution

Slot	Start	End	%
0	1	65536	5
1	262144	262144	95
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

Write Transfer Size Distribution

Slot	Start	End	%
0	262144	262144	100
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

6.4.3 *AI_TF Workload Definition (subcomponent)*

AI_TF

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
lock	0	chmod	0
statfs	0	pathconf	0
trunc	0	neg_stat	0

Execution Parameters

Parameter	Value	Parameter	Value
Procs	2	Dirs per proc	2
Oprate per proc	2	Files per dir	10
Avg file size	140 MiB		

Miscellaneous

Option	Value	Option	Value
write commit %	30	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate speed	0
LRU	1	init read flag	0
release version	3	fadvise seq %	0
fadvise rand %	0	fadvise don't need %	0

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

AI_TF Part 2

Read Transfer Size Distribution

Slot	Start	End	%
0	262144	262144	100
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

Write Transfer Size Distribution

Slot	Start	End	%
0	10240	24576	20
1	32768	32768	20
2	65536	65536	20
3	131072	131072	5
4	200704	200704	20
5	262144	262144	5
6	1048576	1048576	5
7	2097152	2621440	5
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

6.4.4 *AI_TR Workload Definition (subcomponent)*

AI_TR

File Operation Distribution

Operation	%	Operation	%
read	95	read file	0
mmap read	0	rand read	0
write	0	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	5	access	0
rename	0	copyfile	0
lock	0	chmod	0
statfs	0	pathconf	0
trunc	0	neg_stat	0

Execution Parameters

Parameter	Value	Parameter	Value
Procs	10	Dirs per proc	2
Oprate per proc	3	Files per dir	10
Avg file size	140 MiB		

Miscellaneous

Option	Value	Option	Value
write commit %	0	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate speed	0
LRU	1	init read flag	0
release version	3	fadvise seq %	0
fadvise rand %	0	fadvise don't need %	0

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

AI_TR Part 2

Read Transfer Size Distribution

Slot	Start	End	%
0	2097152	2097152	100
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

Write Transfer Size Distribution

Slot	Start	End	%
0	262144	262144	100
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

6.4.5 *AI_CP Workload Definition (subcomponent)*

AI_CP

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	5	access	0
rename	0	copyfile	0
lock	0	chmod	0
statfs	0	pathconf	0
trunc	0	neg_stat	0

Execution Parameters

Parameter	Value	Parameter	Value
Procs	1	Dirs per proc	1
Oprate per proc	1	Files per dir	1
Avg file size	30 MiB		

Miscellaneous

Option	Value	Option	Value
write commit %	30	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate speed	0
LRU	1	init read flag	0
release version	3	fadvise seq %	0
fadvise rand %	0	fadvise don't need %	0

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 %	0	comp granule size	8192
cipher flag	0	pattern version	2

AI_CP Part 2

Read Transfer Size Distribution

Slot	Start	End	%
0	1048576	1048756	100
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

Write Transfer Size Distribution

Slot	Start	End	%
0	1	511	80
1	2097152	2097152	20
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

6.5 Genomics (GENOMICS) Benchmark

6.5.1 *Genomics Workload Description*

The Genomics workload models the entire pipeline of the Genomics workflow. The traces used to construct this workload came from commercial and research facilities that perform genetic analysis. The I/O behavior was captured and is synthesized by the benchmark. The data has been sanitized so that it does not contain any of the original genome data. The business metric for GENOMICS benchmark is JOBS.

6.5.2 *GENOMICS Workload Definition*

GENOMICS

File Operation Distribution

Operation	%	Operation	%
read	70	read file	0
mmap read	0	rand read	2
write	8	write file	0
mmap write	0	rand write	1
rmw	0	append	0
mkdir	0	rmdir	0
readdir	0	create	1
unlink	1	unlink2	0
stat	12	access	4
rename	0	copyfile	0
lock	0	chmod	1
statfs	0	pathconf	0
trunc	0	neg_stat	0

Execution Parameters

Parameter	Value	Parameter	Value
Procs	4	Dirs per proc	2
Oprate per proc	250	Files per dir	25
Avg file size	1613 KiB		

Global Parameters

Threshold	%	Threshold	Value
proc oprate	75	proc latency	0
global oprate	95	global latency	0
workload variance	5	Dedicated subdir	0
Warmup secs	300	Metric	JOBS

Miscellaneous

Option	Value	Option	Value
write commit %	0	background	0
% direct	0	sharemode	0
% osync	0	uniform size dist	0
% notification	0	init rate speed	0
LRU	1	init read flag	0
release version	3	fadvise seq %	20
fadvise rand %	0	fadvise don't need %	100

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 %	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	2

GENOMICS Part 2

Read Transfer Size Distribution

Slot	Start	End	%
0	1	4095	1
1	4096	4096	2
2	8192	65536	1
3	131072	131072	96
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

Write Transfer Size Distribution

Slot	Start	End	%
0	1	2048	5
1	2049	8192	4
2	8193	131072	4
3	524288	524288	87
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

File Size Distibution

Slot	Start	End	%
0	1	8191	38
1	8192	131071	43
2	131072	1048575	9
3	1048576	10485760	9
4	104857600	104857600	1
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0

7 FAQ

7.1 SPECstorage Solution 2020 Benchmark Press Release

Question 1: What is the SPECstorage Solution 2020 benchmark and how does it compare to other storage solution benchmarks?

Answer: The SPECstorage Solution 2020 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: https://www.spec.org/storage

Question 2: What improvements have been made to the SPECstorage Solution 2020 benchmark?
Answer: See the "What's New in SPECstorage Solution 2020?" document included with the benchmark or on the SPECstorage Solution 2020 website: https://www.spec.org/storage/

Question 3: How were the SPECstorage Solution 2020 workloads determined?
Answer: The SPECstorage Solution 2020 workloads are based on system call and network traces collected from real environments and input from domain experts and published documentation of the real-world implementation of the application types being simulated.

Question 4: What are the metrics for the SPECstorage Solution 2020 benchmark?
Answer: The SPECstorage Solution 2020 benchmark has multiple performance measurement metrics. Please refer to the table below:

- SWBUILD = BUILDS
- VDA = STREAMS
- EDA_BLENDED = JOBS
- AI_IMAGE = JOBS
- GENOMICS = JOBS

Question 5: What is the correlation between the SPECstorage Solution 2020 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 SPECstorage Solution 2020 benchmark a CPU-intensive or I/O-intensive benchmark?
Answer: The SPECstorage Solution 2020 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 SPECstorage Solution 2020 benchmark, the vendor will typically add additional hardware – such as memory, disk controllers, disks, network controllers and buffer/file/page 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 SPECstorage Solution 2020 benchmark designed?
Answer: The benchmark was developed for load-generating clients running UNIX or Windows. The SPECstorage Solution 2020 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 SPECstorage Solution 2020 benchmark?
Answer: Yes, users can measure their own workloads by making changes to the SPECstorage Solution 2020 benchmark workload objects file.(storage2020.yml) The SPECstorage Solution 2020 User's Guide details how this can be done. Workloads created by users cannot, however, be compared with SPECstorage Solution 2020 results, nor can they be published in any form, as specified within the SPECstorage Solution 2020 license.

Question 9: To what extent is the server's measured performance within the SPECstorage Solution 2020 benchmark affected by the client's performance?
Answer: SPEC has written the SPECstorage Solution 2020 benchmark to include the effect of client performance on SPECstorage Solution 2020 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 SPECstorage Solution 2020 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 SPECstorage Solution 2020 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, SPECstorage Solution 2020 configurations might not be typical.

Question 12: Do the SPECstorage Solution 2020 Run and Reporting Rules allow results for a clustered server?
Answer: Yes, cluster configurations are allowed as long as they conform to the SPECstorage Solution 2020 Run and Reporting Rules.

Question 13: What resources are needed to run the SPECstorage Solution 2020 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: https://www.spec.org/storage/

Question 14: What is the estimated time needed to set up and run the SPECstorage Solution 2020 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 SPECstorage Solution 2020 software installs relatively quickly. A SPECstorage Solution 2020 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 SPECstorage Solution 2020 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 SPECstorage Solution 2020 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 SPECstorage Solution 2020 results.

Question 17: Can a RAM disk be used within a SPECstorage Solution 2020 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 SPECstorage Solution 2020 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 SPECstorage Solution 2020 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 SPECstorage Solution 2020 license and when will it be available?
Answer: The SPECstorage Solution 2020 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 SPECstorage Solution 2020 benchmark?
Answer: The majority of questions should be answered in the SPECstorage Solution 2020 User's Guide. There is also useful information on the SPEC Web site: https://www.spec.org/storage/

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 SPECstorage Solution 2020 benchmark?
Answer: Please read the SPECstorage Solution 2020 User's Guide and SPECstorage Solution 2020 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 SPECstorage Solution 2020 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 SPECstorage Solution 2020 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, on each client, 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 SPECstorage Solution 2020 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 SM2020 Python script. (Typically Control-C) This will terminate all SPECstorage Solution 2020 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 SPECstorage Solution 2020 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 SPECstorage Solution 2020 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.

7.2 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.

7.3 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 SpecReport submission tool documentation is in that section.

8 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.

BSD is a trademark of Berkeley University in the United States.

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

Microsoft[l], Windows, Windows[(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.

Cent OS is a trademark of RedHat, in the United States, other countries, or both.

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

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

9 Research corner

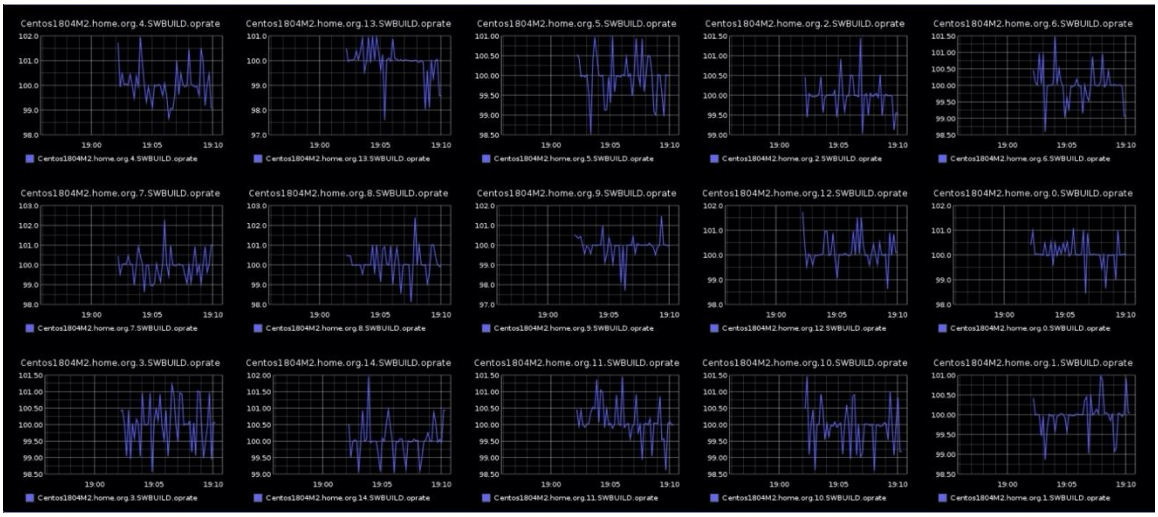
9.1 Custom changes to storage2020.yml file to add new workloads.

Each workload has a description in the storage2020.yml file. One may modify, or add new, definitions to this file for custom workload creation.

9.2 Custom workload objects

The SPECstorage Solution 2020 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.

9.3 Statistics Collection via PDSM



Within the SPECstorage(TM) 2020 benchmark there is an internal mechanism for collecting statistical information while the benchmark is running. This internal subsystem is called Portable Distributed Shared Memory (PDSM). This is an internal mechanism that synthesizes a distributed shared memory mechanism that is used for collecting statistical information from all of the clients, and client processes, throughout the data center that are participating in the benchmark. This mechanism can also be used for dynamic updates to a running benchmark. This document covers the statistical collection mechanism. The dynamic control will be covered by a separate document, in the future.

9.3.1 *Configuring PDSM*

In the SPECstorage 2020 benchmark there is a configuration file that contains information about the setup of the PDSM mechanism. The entries in the sfs_rc file are:

```
# PDSM_MODE of 0 create shared PDSM log file with overwrites.
# PDSM_MODE of 1 create shared PDSM log file with open append.
PDSM_MODE=
# PDSM_INTERVAL # Interval in seconds
PDSM_INTERVAL=
# <PLATFORM>_PDSM_LOG filename  Full pathname to the PDSM log file. Used to
# log the details of 'very proc's activities.
```

```
UNIX_PDSM_LOG=
WINDOWS_PDSM_LOG=
# <PLATFORM>_PDSM_CONTROL filename  Full pathname to the PDSM control file. Used
# for dynamically changing workloads.
UNIX_PDSM_CONTROL=
WINDOWS_PDSM_CONTROL=
```

As the benchmark is running, statistical information will be written to the PDSM_LOG file. Each of the participating clients mounts the /mnt/logdir directory from a shared filesystem, such as NFS or SMB. Updates to this log file are made by all of the processes running on all of the clients participating in the benchmark.

- **PDSM_MODE=integer value**
  Specifies the access behavior of the clients. PDSM_MODE=0 tells the client processes to update only their entries in the log file and to overwrite their entry every PDSM_INTERVAL. This presents a single view of all of the statistical data that is updating every PDSM_INTERVAL seconds. If the PDSM_MODE is 1, then each client process will append new statistical data to the end of the file. This provides data over a continuum of time.
- **PDSM_INTERVAL**= integer value
  The duration of time in seconds between sample collection, that is, how often the client processes will update the log.
- **UNIX_PDSM_LOG**=filename
  The path to the Unix PDSM_LOG file. Example: /tmp/pdsm.log
- **WINDOWS_PDSM_LOG= filename**
  The path to the Windows PDSM log file. Example: C:\tmp\pdsm.log
- **UNIX_PDSM_CONTROL=filename**
  The Unix control file used for dynamic manipulation of the benchmark while it is running.
- **WINDOWS_PDSM_CONTROL=filename**
  The Windows control file used for dynamic manipulation of the benchmark while it is running, used for dynamically changing workloads.

The PDSM control file can be placed in a unified shared name space or defined to be client local.

9.3.2 *Configuring Carbon*

9.3.2.1 The pipeline of processing

In order to visualize the statistical data collected above one needs to have a mechanism to transport and display this information. The pipeline of processing is:

- PDSM Collection to a log file
- Reading, transposing, and sending the statistical data to a "Carbon" server
- Using a "Graphite" server to compose the graphical representations
- Using a web browser to view the graphical visualizations.

9.3.2.2 Prerequisite environment

The first step was accomplished by the method described earlier. The next step involves sending data to a Carbon server. This will require setting up and installing:

- Dockers
  See: https://docs.docker.com/get-started/
- Carbon & Graphite servers
  See: https://graphite.readthedocs.io/en/latest/install.html

Once these services are setup and running then the process to visualize begins with using the "pump_carbon" utility, provided with the SPECstorage 2020 benchmark in the bin directory.

The pump_carbon utility reads the PDSM_LOG file, transposes this into a Carbon format, and sends the information to the Carbon server.

pump_carbon:

-h..... Help screen
-f..... pdsm_log_file_name
-k..... Ignore STOP. Used for multiple load point runs with PDSM_MODE=1
-s..... Carbon server
-i..... Interval in seconds
-t..... One pass flag. Use with append mode collection.
-v..... Display version information.

Example: pump_carbon -f /mnt/logdir/pdsm.log -s hostname_of_carboI_server -i 1

The command line above tells Pump_carbon to read the PDSM log file and send the data to the Carbon server every 1 seconds.

The Carbon server will connect the arriving information and store it in a Whisper database on the Carbon server. The information is encoded by pump_carbon and sent to the Carbon server in the format described below and to this socket for processing by a "Carbon" server that is listening on its port (2003).

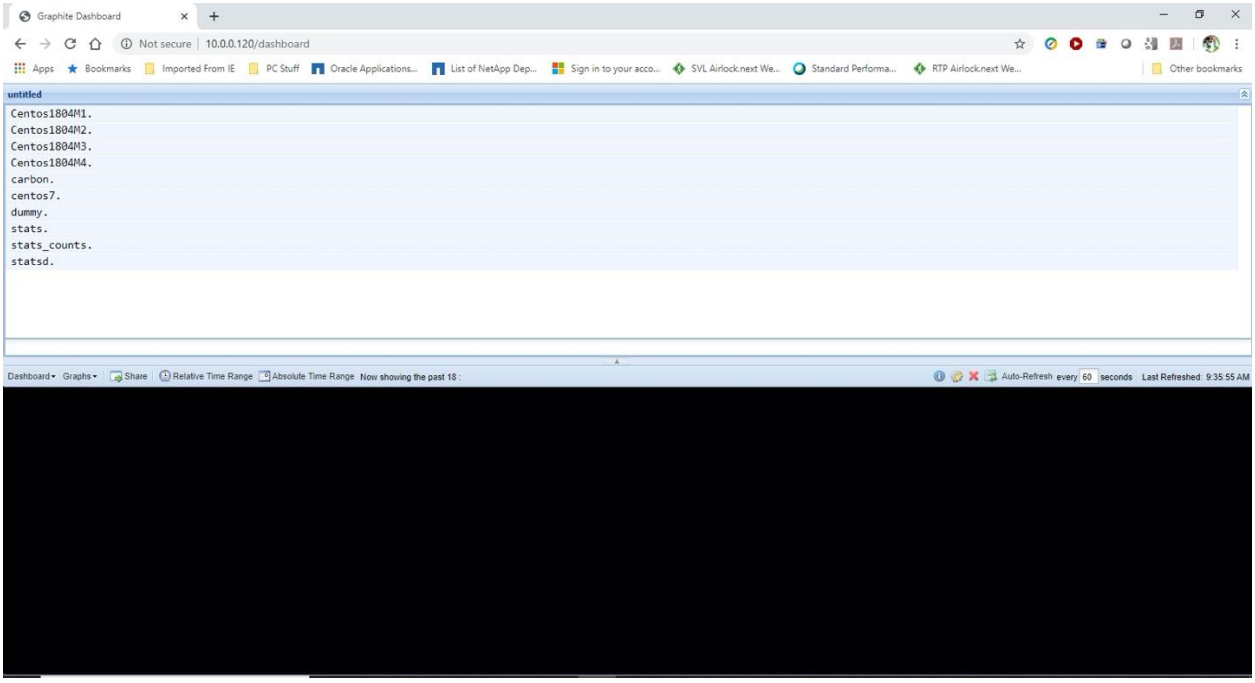
```
Pump_carbon encoding
---------
String . Integer . WorkLoadName . ResultString    Float  Timestamp
-------------------------------------------------------------------------------------------------------------
%s.%d.%s.oprate %10.6f %lld (hostname, client_id, workload_name, op_rate, TimeStamp)
%s.%d.%s.read_latency %10.6f %lld (hostname, client_id, workload_name, read_latency, TimeStamp)
%s.%d.%s.read_file_latency %10.6f %lld (hostname, client_id, workload_name, read_file_latency, TimeStamp)
%s.%d.%s.read_rand_latency %10.6f %lld (hostname, client_id, workload_name, read_rand_latency, TimeStamp)
%s.%d.%s.mmap_read_latency %10.6f %lld (hostname, client_id, workload_name, mmap_read_latency, TimeStamp)
%s.%d.%s.mmap_write_latency %10.6f %lld (hostname, client_id, workload_name, mmap_write_latency, TimeStamp)
%s.%d.%s.write_latency %10.6f %lld (hostname, client_id, workload_name, write_latency, TimeStamp)
%s.%d.%s.write_file_latency %10.6f %lld (hostname, client_id, workload_name, write_file_latency, TimeStamp)
%s.%d.%s.write_rand_latency %10.6f %lld (hostname, client_id, workload_name, write_rand_latency, TimeStamp)
%s.%d.%s.rmw_latency %10.6f %lld (hostname, client_id, workload_name, rmw_latency, TimeStamp)
%s.%d.%s.mkdir_latency %10.6f %lld (hostname, client_id, workload_name, mkdir_latency, TimeStamp)
%s.%d.%s.rmdir_latency %10.6f %lld (hostname, client_id, workload_name, rmdir_latency, TimeStamp)
%s.%d.%s.create_latency %10.6f %lld (hostname, client_id, workload_name, create_latency, TimeStamp)
%s.%d.%s.unlink_latency %10.6f %lld (hostname, client_id, workload_name, unlink_latency, TimeStamp)
%s.%d.%s.unlink2_latency %10.6f %lld (hostname, client_id, workload_name, unlink2_latency, TimeStamp)
%s.%d.%s.append_latency %10.6f %lld (hostname, client_id, workload_name, append_latency, TimeStamp)
%s.%d.%s.lock_latency %10.6f %lld (hostname, client_id, workload_name, lock_latency, TimeStamp )
%s.%d.%s.access_latency %10.6f %lld (hostname, client_id, workload_name, access_latency, TimeStamp)
%s.%d.%s.chmod_latency %10.6f %lld (hostname, client_id, workload_name, chmod_latency, TimeStamp)
%s.%d.%s.readdir_latency, %10.6f %lld          (hostname, client_id, workload_name, readdir_latency, TimeStamp)
%s.%d.%s.stat_latency, %10.6f, %lld (hostname, client_id, workload_name, stat_latency, TimeStamp)
%s.%d.%s.neg_stat_latency, %10.6f, %lld (hostname, client_id, workload_name, neg_stat_latency, TimeStamp)
%s.%d.%s.copyfile_latency %10.6f %lld          (hostname, client_id, workload_name, copyfile_latency, TimeStamp)
%s.%d.%s.rename_latency %10.6f %lld (hostname, client_id, workload_name, rename_latency, TimeStamp)
%s.%d.%s.statfs_latency %10.6f %lld (hostname, client_id, workload_name, statfs_latency, TimeStamp)
%s.%d.%s.pathconf_latency %10.6f %lld          (hostname, client_id, workload_name, pathconf_latency, TimeStamp)
%s.%d.%s.trunc_latency %10.6f %lld (hostname, client_id, workload_name, trunc_latency, TimeStamp)

Examples:
Hostname.client_id.Workload.oprate     value     Seconds past Epoch
-------------------------------------------------------------------------------
Centos7.0.VDI.oprate                 25.010000    28723421
Centos7.1.VDI.read_rand_latency       0.001000    28723422
```

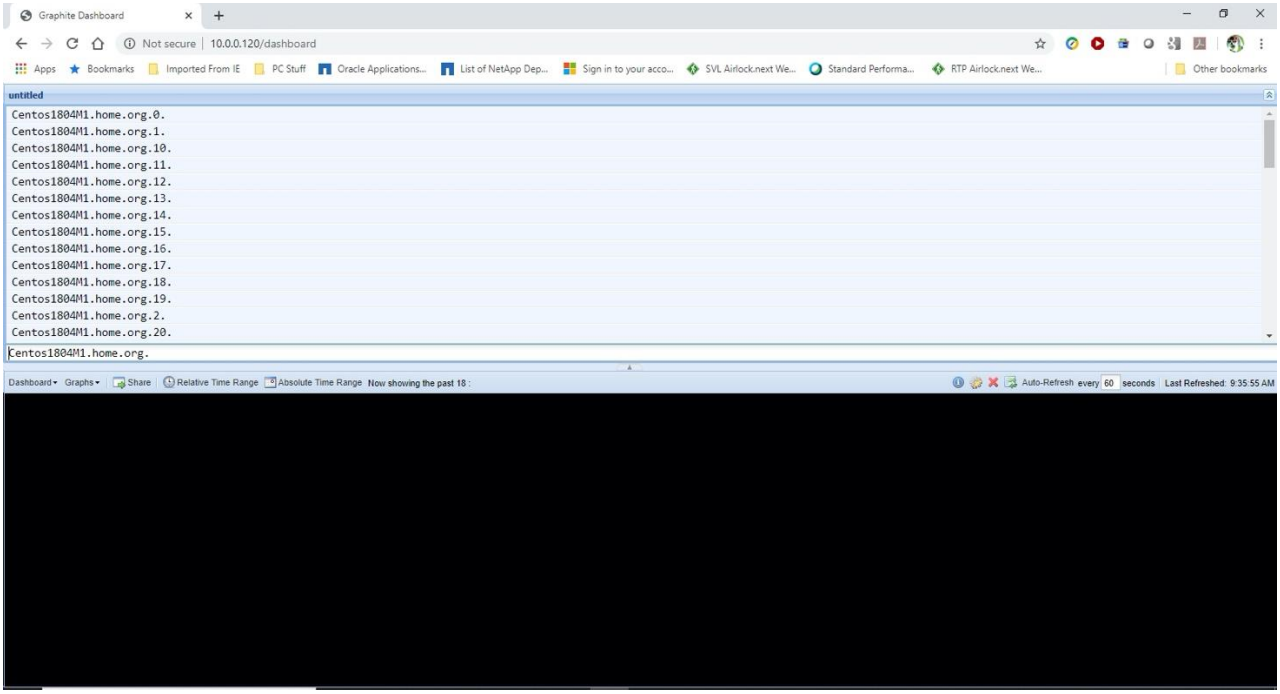
9.3.3 *Visualizing the data*

To view the graphs, one will open a web browser and enter the URL for the Graphite server. (Same host as the Carbon server) e.g. http://carbonserver or http://graphiteserver

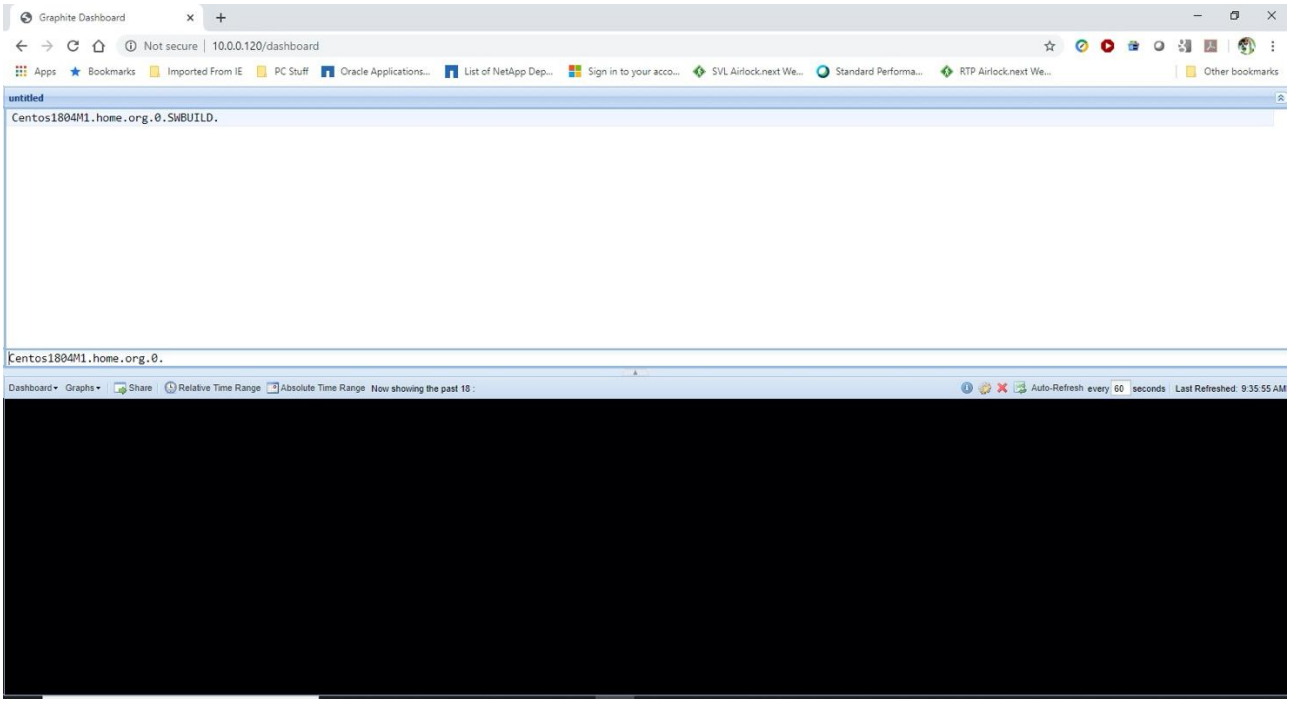
At the Graphite top level: http://graphiteserver switch to DashBoard view and one will see a screen similar to what is below. In this example "centos1804M1" is one of the clients that was running the benchmark and sending statistical data.



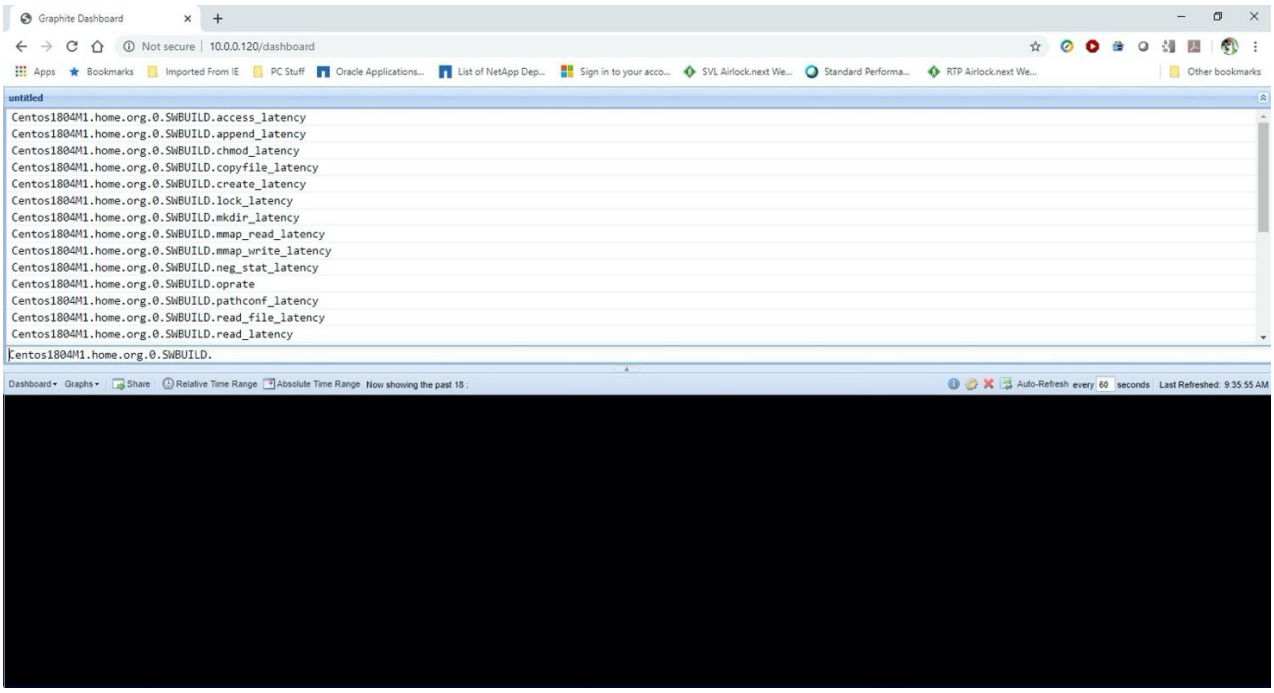
Select the centos1804M1 hostname and one will be presented with the next level; client_id within the SPECstorage processes: (Each process has a unique Client_ID, starting with zero)



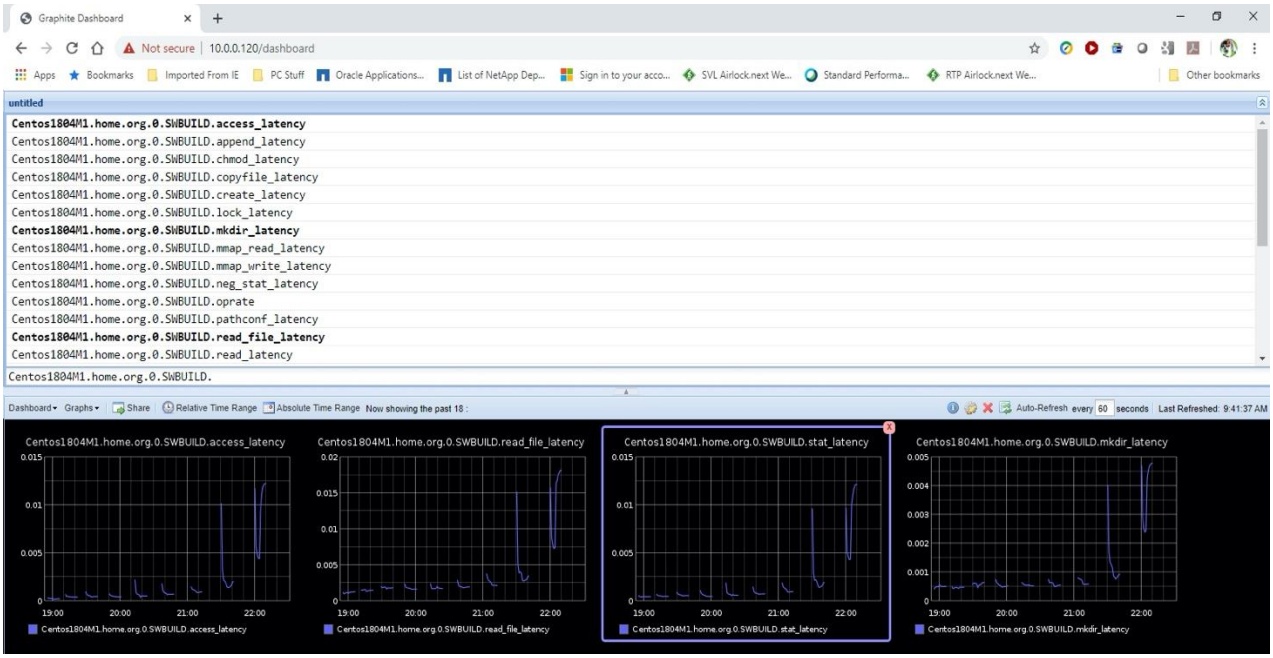
Selecting client_id of zero, one now sees there are statistics for the SWBUILD workload:



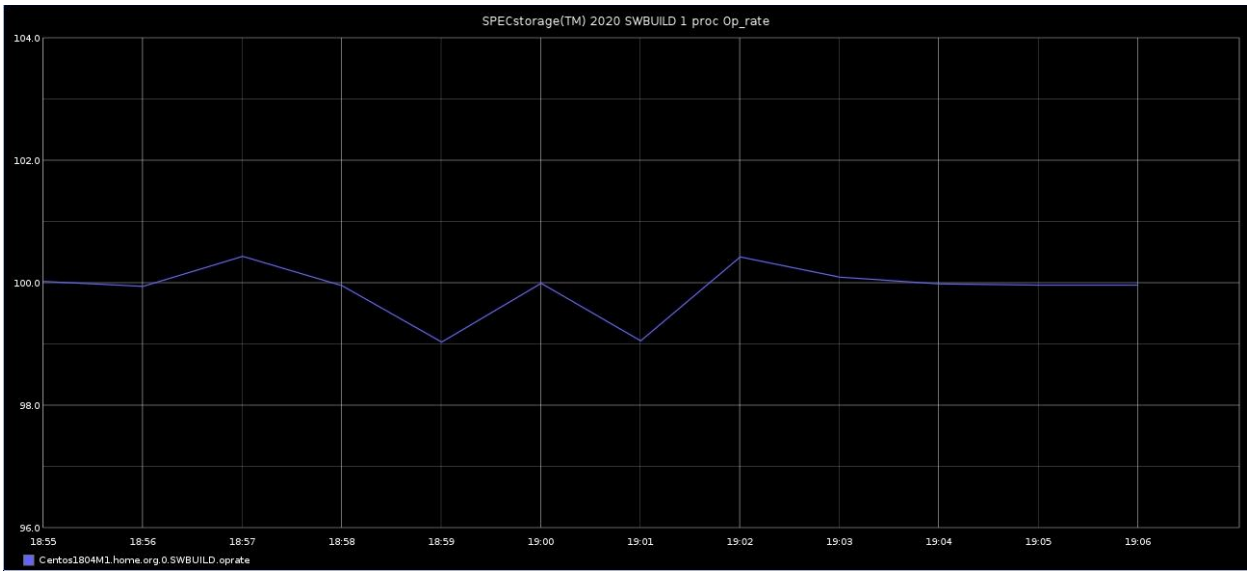
Selecting the SWBUILD workload, one can now see the statistics collected for this client process:



Clicking on the various statistics will produce graphs in the bottom window. Be sure you select a relative, or absolute, time frame that covers when the samples were collected.

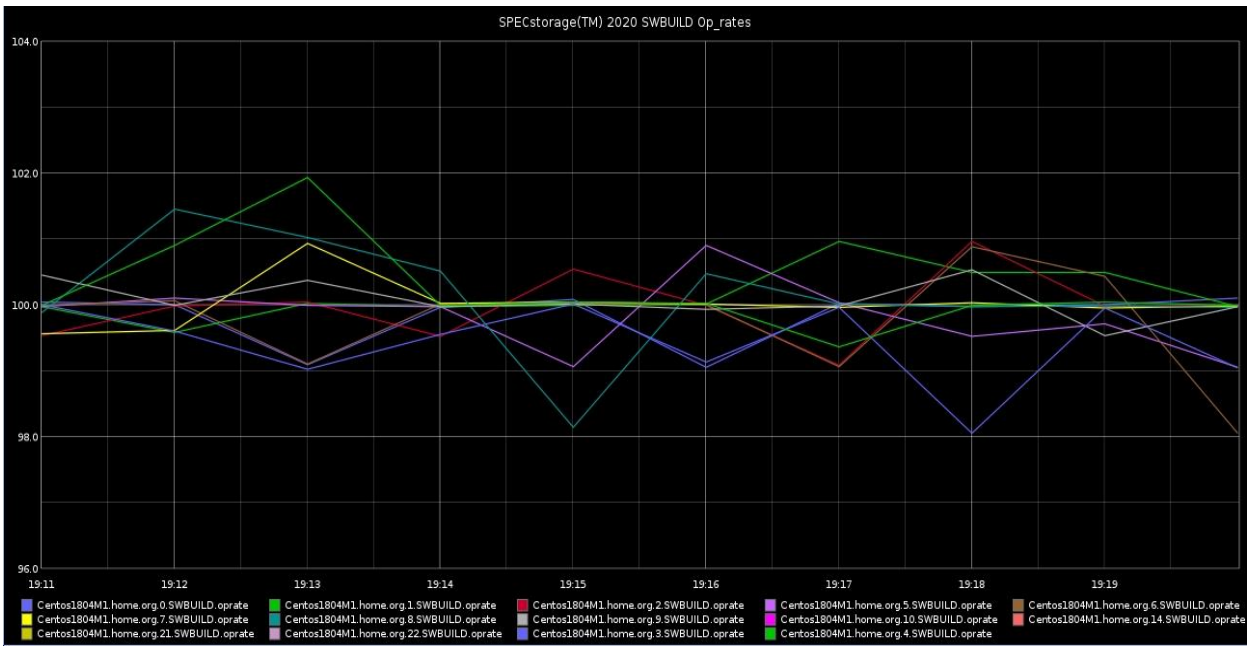


Close up view of op_rate: This is the op_rate of process/client_ID 0, on the client "centos1804M1", over the time_of_day when the samples were collected.

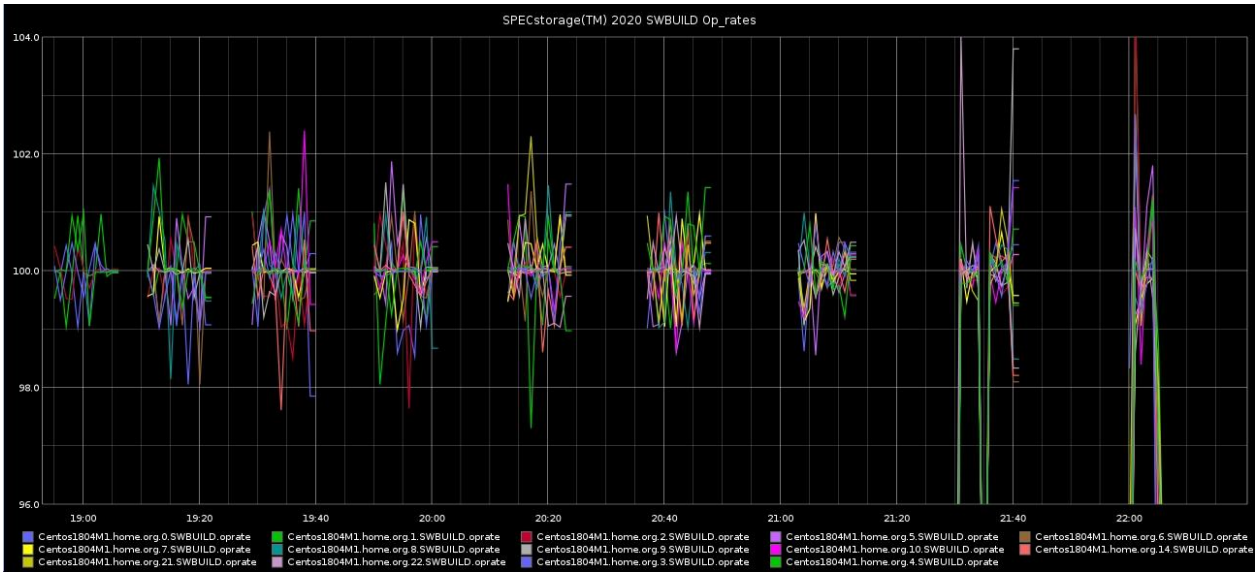


Other examples of visualizations one can produce:

The op_rate for all client_IDs running SWBUILD



The op_rate for all client_ids running SWBUILD over a 9 load point run:



10 Appendix A – Building the Benchmark Components

10.1 Building the SPECstorage Solution 2020 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 SPECstorage Solution 2020 on a UNIX system.

10.1.1 *How to build SPECstorage2020 on Unix. i.e. Linux, FreeBSD, MacOS, Solaris*

If one is trying to use the standard "make" described below, then it is up to the developer to install all necessary packages to support building libyaml. (**make**, **gcc**, **autom4te**, **automake**, **autoconf**, **libtool, m4**) This method has been tested on AIX, Linux, MacOS, FreeBSD, and Solaris. With these tools installed, then the build is as below:

1. Install SPECstorage Solution 2020 kit. (the top level of the installed kit will be called $top)

2. cd $top

3. Type "make"

In the event that the libyaml kit doesn't compile on your platform then one may download the libyaml port for these platforms from:

BSD: https://www.freshports.org/textproc/libyaml/
AIX: https://repology.org/project/libyaml/versions
MAC OS: https://pecl.php.net/package/yaml
Solaris: https://www.opencsw.org/packages/libyaml_dev/

Place the kit in $top/redistributable_sources, build it, and modify the makefiles so that it will skip doing the unzip, bootstrap, and configure. Just use the libyaml.a, or libyaml, that you just built manually. This method has been tested for BSD, MacOS, and Solaris.

10.2 Building the SPECstorage Solution 2020 benchmark for Windows

How to build SPECstorage2020 on Windows (from scratch).

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.

The instructions below assume one already has Visual Studio 2015 Community installed.

1. Install SPECstorage2020 kit. (the top level of the installed kit will be called $top)
2. Download and install 'cmake' See: https://cmake.org/download/
3. cd to $top/redistributable_sources and unzip libyaml-master.zip into $top/redistributable_sources.
   a. The unzip will create a new libyaml-master directory at the same level in the directory structure as the libyaml-master.zip file.
4. Run Cmake and specify the source folder as:
   $top/redistributable_sources/libyaml-master
   where to place the binaries:
   $top/redistributable_sources/libymal-master
   click on the "Configure" button, set the "Optional Platform for generator" to x64, then click "Configure" then select "YAML_STATIC_LIB_NAME" then click on the "Generate" button. Then exit Cmake.
5. In the Windows file manager, double click on the solution file:
   $top/msbuild/netmist_dist.sln. This will launch the Visual Studio environment.

6. Inside of Visual studio, select the "Solution Configurations" to be "Release" and the "Solution platforms" to be x64.
7. Right mouse click on "Solution 'netmist_dist'" and scroll down to "Add -> Existing project" and then select
   "$top/redistributable_sources/libyaml-master/yaml.vcxproj"
8. Right mouse click on the 'Solution netmist_dist' and scroll down to "Project dependencies" and select "yaml"
9. Right mouse click on the yaml project and modify its 'C' code generation properties such that the Runtime library is set to /MT (Multi-Threaded **not** /MD)
10. Right mouse click on the yaml project, and scroll down to "Build", and select this.
11. Right mouse click on project "netmist" and scroll down to "Add -> Existing item" and then select $top/redistributable_sources/libyaml-master/Release/ON.lib
12. Select the netmist project, modify its properties to add YAML_DECLARE_STATIC to the 'C' preprocessor directives.
13. Select the "Build" tab and scroll down to "Build Solution" then click on this or press F7.
14. Be sure to install the SPECstorage2020 binaries, that you just built, onto all of the participating clients. The new binaries are in $top/build/dist/windows/x64:
    netmist.exe, netmist_monitor.exe, netmist_modify.exe, pump_carbon.exe, pump_csv.exe

Note: If building for x86 target, then modify step 4 to select Win32, and also select Win32 in step 11.

10.2.1 *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.

11 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
```

11.1 Setting up Openssh on Windows

See:

https://docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_install_firstuse

https://winaero.com/blog/enable-openssh-server-windows-10/

Pro tip: *If one is setting up Openssh on a Windows system and you are using an account name that has administrative privileges, you will need to take additional steps to configure Openssh. For non-administrative accounts one normally sets up items in the .ssh directory. However, for administrative accounts there are additional files in other directories that must also be setup. Read the above links carefully and take heed of these extra steps for administrative accounts or it will be a very frustrating day.*

12 Appendix C – Tunes for the load generators

12.1 Linux tunes

Some or all of the following tuning parameters may be helpful in tuning Linux load generators for optimal performance. If you have clients with large amounts of RAM, and 40GigE or 100GigE you may need to set some values even higher than what is listed below.

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

```
#
# Recommended client tunes for running SPECstorage Solution 2020
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
# Network parameters. In unit of bytes
net.core.wmem_max = 16777216
net.core.wmem_default = 1048576
net.core.rmem_max = 16777216
net.core.rmem_default = 1048576
net.ipv4.tcp_rmem = 1048576 8388608 16777216
net.ipv4.tcp_wmem = 1048576 8388608 16777216
net.core.optmem_max = 2048000
net.core.somaxconn = 65535
#
# These settings are in 4 KiB size chunks, in bytes they are:
#    Min = 16MiB, Def=350MiB, Max=16GiB
# In unit of 4k pages
net.ipv4.tcp_mem = 4096 89600 4194304
#
# Misc network options and flags
#
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 1
net.ipv4.tcp_no_metrics_save = 1
net.ipv4.route.flush = 1
net.ipv4.tcp_low_latency = 1
net.ipv4.ip_local_port_range = 1024 65000
net.ipv4.tcp_slow_start_after_idle = 0
net.core.netdev_max_backlog = 300000
#
# Various filesystem / pagecache options
#
vm.dirty_expire_centisecs = 100
vm.dirty_writeback_centisecs = 100
vm.dirty_ratio = 20
vm.dirty_background_ratio = 5
#
# Recommended by: https://cromwell-intl.com/open-source/performance-tuning/tcp.html
#
net.ipv4.tcp_sack = 0
net.ipv4.tcp_dsack = 0
net.ipv4.tcp_fack = 0
```

On all clients. In /etc/security/limits.conf one **MUST** increase the value associated with the maximum number of open files per user.
Example:

```
spec            -       nofile          10000
```

It is also advisable to increase the maximum number of processes per user.
Example:

```
spec            -       nproc           10000
```

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

12.1.1 *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.

12.2 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 the Windows Defender's Virus threat protection (Real time protection) or it will consume massive amounts of CPU.
- Disable all slide-show wallpapers, and screen savers.
- Disable automatic updates.
- Disable any interrupt throttling.

12.2.1 *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.