



SPC BENCHMARK 1™

FULL DISCLOSURE REPORT

NETAPP, INC.
NETAPP FAS8040 (2-NODE CLUSTER)

SPC-1 V1.14

Submitted for Review: February 19, 2014
Submission Identifier: A00141

First Edition – February 2014

THE INFORMATION CONTAINED IN THIS DOCUMENT IS DISTRIBUTED ON AN AS IS BASIS WITHOUT ANY WARRANTY EITHER EXPRESS OR IMPLIED. The use of this information or the implementation of any of these techniques is the customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by NetApp, Inc. for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

This publication was produced in the United States. NetApp, Inc. may not offer the products, services, or features discussed in this document in other countries, and the information is subject to change with notice. Consult your local NetApp, Inc. representative for information on products and services available in your area.

© Copyright NetApp, Inc. 2014. All rights reserved.

Permission is hereby granted to reproduce this document in whole or in part, provided the copyright notice as printed above is set forth in full text on the title page of each item reproduced.

Trademarks

SPC Benchmark-1, SPC-1, SPC-1 IOPS, SPC-1 LRT and SPC-1 Price-Performance are trademarks of the Storage Performance Council. NetApp, ONTAP and RAID-DP are trademarks or registered trademarks of NetApp, Inc. in the United States and other countries. All other brands, trademarks, and product names are the property of their respective owners.

Table of Contents

Audit Certification.....	vii
Audit Certification (<i>cont.</i>)	viii
Letter of Good Faith	ix
Executive Summary.....	10
Test Sponsor and Contact Information.....	10
Revision Information and Key Dates	10
Tested Storage Product (TSP) Description.....	11
Summary of Results	12
Storage Capacities, Relationships, and Utilization	13
Response Time – Throughput Curve	16
Response Time – Throughput Data.....	16
Priced Storage Configuration Pricing	17
Differences between the Tested Storage Configuration (TSC) and Priced Storage Configuration.....	17
Priced Storage Configuration Diagram.....	18
Priced Storage Configuration Components.....	19
Configuration Information	20
Benchmark Configuration (BC)/Tested Storage Configuration (TSC) Diagram.....	20
Storage Network Configuration	20
Host System(s) and Tested Storage Configuration (TSC) Table of Components	20
Benchmark Configuration/Tested Storage Configuration Diagram.....	21
Host Systems and Tested Storage Configuration Components.....	22
Customer Tunable Parameters and Options	23
Tested Storage Configuration (TSC) Description	23
SPC-1 Workload Generator Storage Configuration	23
ASU Pre-Fill.....	24
SPC-1 Data Repository.....	25
Storage Capacities and Relationships	25
SPC-1 Storage Capacities	25
SPC-1 Storage Hierarchy Ratios	26
SPC-1 Storage Capacity Charts	26
Storage Capacity Utilization	28
Logical Volume Capacity and ASU Mapping	29
SPC-1 Benchmark Execution Results.....	30
SPC-1 Tests, Test Phases, and Test Runs.....	30

“Ramp-Up” Test Runs.....	31
Primary Metrics Test – Sustainability Test Phase	31
SPC-1 Workload Generator Input Parameters	32
Sustainability Test Results File	32
Sustainability – Data Rate Distribution Data (MB/second)	32
Sustainability – Data Rate Distribution Graph	32
Sustainability – I/O Request Throughput Distribution Data.....	33
Sustainability – I/O Request Throughput Distribution Graph	33
Sustainability – Average Response Time (ms) Distribution Data	34
Sustainability – Average Response Time (ms) Distribution Graph	34
Sustainability – Response Time Frequency Distribution Data	35
Sustainability – Response Time Frequency Distribution Graph	35
Sustainability – Measured Intensity Multiplier and Coefficient of Variation.....	36
Primary Metrics Test – IOPS Test Phase.....	37
SPC-1 Workload Generator Input Parameters	37
IOPS Test Results File.....	37
IOPS Test Run – I/O Request Throughput Distribution Data	38
IOPS Test Run – I/O Request Throughput Distribution Graph.....	38
IOPS Test Run – Average Response Time (ms) Distribution Data	39
IOPS Test Run – Average Response Time (ms) Distribution Graph	39
IOPS Test Run – Response Time Frequency Distribution Data	40
IOPS Test Run – Response Time Frequency Distribution Graph.....	40
IOPS Test Run – I/O Request Information.....	41
IOPS Test Run – Measured Intensity Multiplier and Coefficient of Variation	41
Primary Metrics Test – Response Time Ramp Test Phase	42
SPC-1 Workload Generator Input Parameters	42
Response Time Ramp Test Results File.....	42
Response Time Ramp Distribution (IOPS) Data.....	43
Response Time Ramp Distribution (IOPS) Data (<i>continued</i>)	44
Response Time Ramp Distribution (IOPS) Graph	44
SPC-1 LRT™ Average Response Time (ms) Distribution Data.....	45
SPC-1 LRT™ Average Response Time (ms) Distribution Graph	45
SPC-1 LRT™ (10%) – Measured Intensity Multiplier and Coefficient of Variation	46
Repeatability Test	47
SPC-1 Workload Generator Input Parameters	47
Repeatability Test Results File	48
Repeatability 1 LRT – I/O Request Throughput Distribution Data.....	49
Repeatability 1 LRT – I/O Request Throughput Distribution Graph	49
Repeatability 1 LRT –Average Response Time (ms) Distribution Data	50

Repeatability 1 LRT –Average Response Time (ms) Distribution Graph	50
Repeatability 1 IOPS – I/O Request Throughput Distribution Data	51
Repeatability 1 IOPS – I/O Request Throughput Distribution Graph.....	51
Repeatability 1 IOPS –Average Response Time (ms) Distribution Data.....	52
Repeatability 1 IOPS –Average Response Time (ms) Distribution Graph	52
Repeatability 2 LRT – I/O Request Throughput Distribution Data.....	53
Repeatability 2 LRT – I/O Request Throughput Distribution Graph	53
Repeatability 2 LRT –Average Response Time (ms) Distribution Data	54
Repeatability 2 LRT –Average Response Time (ms) Distribution Graph.....	54
Repeatability 2 IOPS – I/O Request Throughput Distribution Data	55
Repeatability 2 IOPS – I/O Request Throughput Distribution Graph.....	55
Repeatability 2 IOPS –Average Response Time (ms) Distribution Data.....	56
Repeatability 2 IOPS –Average Response Time (ms) Distribution Graph	56
Repeatability 1 (LRT) Measured Intensity Multiplier and Coefficient of Variation	57
Repeatability 1 (IOPS) Measured Intensity Multiplier and Coefficient of Variation	57
Repeatability 2 (LRT) Measured Intensity Multiplier and Coefficient of Variation	57
Repeatability 2 (IOPS) Measured Intensity Multiplier and Coefficient of Variation	58
Data Persistence Test.....	59
SPC-1 Workload Generator Input Parameters	59
Data Persistence Test Results File	59
Data Persistence Test Results.....	60
Priced Storage Configuration Availability Date.....	61
Pricing Information.....	61
Tested Storage Configuration (TSC) and Priced Storage Configuration Differences.....	61
Anomalies or Irregularities	61
Appendix A: SPC-1 Glossary	62
“Decimal” (<i>powers of ten</i>) Measurement Units	62
“Binary” (<i>powers of two</i>) Measurement Units.....	62
SPC-1 Data Repository Definitions	62
SPC-1 Data Protection Levels	63
SPC-1 Test Execution Definitions	63
I/O Completion Types.....	65
SPC-1 Test Run Components	65
Appendix B: Customer Tunable Parameters and Options.....	66
Red Hat Enterprise Linux 6.4 (64-bit)	66
Storage Controllers	66

Root Aggregate	66
SPC Data Aggregates.....	66
LUN Options	67
Storage System Options.....	67
Appendix C: Tested Storage Configuration (TSC) Creation	68
Configure the Storage	68
Create and Configure RAID Groups, Aggregates, Volumes and LUNs	68
Create and Configure SPC-1 Logical Volumes	70
Referenced File and Scripts	71
config_full	71
config_storage.py	72
config_hosts.py	77
harness.py.....	82
storage.py.....	88
host.py.....	91
Appendix D: SPC-1 Workload Generator Storage Commands and Parameters	97
ASU Pre-Fill.....	97
Primary Metrics and Repeatability Tests	99
SPC-1 Persistence Test.....	100
Slave JVMs.....	101
Appendix E: SPC-1 Workload Generator Input Parameters	102
spc1.start1.sh	102
start_all_slaves_jvms.sh	103
launch_host1_slaves.sh.....	103
launch_host2_slaves.sh.....	103
launch_host3_slaves.sh.....	104
launch_host4_slaves.sh.....	104
spc1.start2.sh	105

AUDIT CERTIFICATION



Gradient
SYSTEMS

John Elliott
NetApp, Inc.
7301 Kit Creek Road
Building 1, RTP1.3.157
Research Triangle Park, NC 27709

February 19, 2014

The SPC Benchmark 1™ Reported Data listed below for the NetApp FAS8040 (*2-node cluster*) was produced in compliance with the SPC Benchmark 1™ v1.14 Remote Audit requirements.

SPC Benchmark 1™ v1.14 Reported Data	
Tested Storage Product (TSP) Name:	
NetApp FAS8040 (<i>2-node cluster</i>)	
Metric	Reported Result
SPC-1 IOPS™	86,072.33
SPC-1 Price-Performance	\$5.76/SPC-1 IOPS™
Total ASU Capacity	32,219.301 GB
Data Protection Level	Protected 2 (<i>RAID DP™</i>)
Total Price (including three-year maintenance)	\$495,652.43
Currency Used	U.S. Dollars
Target Country for availability, sales and support	USA

The following SPC Benchmark 1™ Remote Audit requirements were reviewed and found compliant with 1.14 of the SPC Benchmark 1™ specification:

- A Letter of Good Faith, signed by a senior executive.
- The following Data Repository storage items were verified by information supplied by NetApp, Inc.:
 - ✓ Physical Storage Capacity and requirements.
 - ✓ Configured Storage Capacity and requirements.
 - ✓ Addressable Storage Capacity and requirements.
 - ✓ Capacity of each Logical Volume and requirements.
 - ✓ Capacity of each Application Storage Unit (ASU) and requirements.
- The total Application Storage Unit (ASU) Capacity was filled with random data, using an auditor approved tool, prior to execution of the SPC-1 Tests.
- An appropriate diagram of the Benchmark Configuration (BC)/Tested Storage Configuration (TSC).

Storage Performance Council
643 Bair Island Road, Suite 103
Redwood City, CA 94062
AuditService@storageperformance.org
650.556.9384

AUDIT CERTIFICATION (CONT.)

NetApp FAS8040 (*2-node cluster*)
SPC-1 Audit Certification

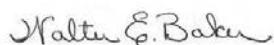
Page 2

- Listings and commands to configure the Benchmark Configuration/Tested Storage Configuration, including customer tunable parameters that were changed from default values.
- SPC-1 Workload Generator commands and parameters used for the audited SPC Test Runs.
- The following Host System requirements were verified by information supplied by NetApp, Inc.:
 - ✓ The type of Host Systems including the number of processors and main memory.
 - ✓ The presence and version number of the SPC-1 Workload Generator on each Host System.
 - ✓ The TSC boundary within each Host System.
- The execution of each Test, Test Phase, and Test Run was found compliant with all of the requirements and constraints of Clauses 4, 5, and 11 of the SPC-1 Benchmark Specification.
- The Test Results Files and resultant Summary Results Files received from NetApp, Inc. for each of following were authentic, accurate, and compliant with all of the requirements and constraints of Clauses 4 and 5 of the SPC-1 Benchmark Specification:
 - ✓ Data Persistence Test
 - ✓ Sustainability Test Phase
 - ✓ IOPS Test Phase
 - ✓ Response Time Ramp Test Phase
 - ✓ Repeatability Test
- There were no differences between the Tested Storage Configuration and Priced Storage Configuration.
- The submitted pricing information met all of the requirements and constraints of Clause 8 of the SPC-1 Benchmark Specification.
- The Full Disclosure Report (FDR) met all of the requirements in Clause 9 of the SPC-1 Benchmark Specification.
- This successfully audited SPC measurement is not subject to an SPC Confidential Review.

Audit Notes:

There were no audit notes or exceptions.

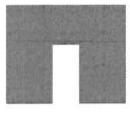
Respectfully,



Walter E. Baker
SPC Auditor

Storage Performance Council
643 Bair Island Road, Suite 103
Redwood City, CA 94062
AuditService@storageperformance.org
650.556.9384

LETTER OF GOOD FAITH



NetApp®

www.netapp.com

919 476 5600 Tel

7301 Kit Creek Road
Research Triangle Park
North Carolina 27709

January 13, 2014

To: Mr. Walter E. Baker, SPC Auditor
Gradient Systems
643 Bair Island Road, Suite 103
Redwood City, CA 94063-2755

Subject: SPC-1 Letter of Good Faith for the NetApp FAS8040 2-Node Cluster

NetApp Inc. is the test sponsor for the above listed product. To the best of our knowledge and belief, the required SPC-1 results and materials we have submitted for that product are complete, accurate, and in full compliance with version 1.14 of the SPC-1 benchmark specification.

Our disclosure of the Benchmark Configuration and execution of the benchmark includes all items that, to the best of our knowledge and belief, materially affect the reported results, regardless of whether such items are explicitly required to be disclosed by the SPC-1 benchmark specification.

Sincerely,

A handwritten signature in black ink, appearing to read "Daniel Morgan".

Daniel Morgan
Vice President, Data Solutions Group Integrations Engineering

EXECUTIVE SUMMARY

Test Sponsor and Contact Information

Test Sponsor and Contact Information	
Test Sponsor Primary Contact	NetApp, Inc. – http://www.netapp.com Don Capps – Don.Capps@netapp.com 7417 Crenshaw Dr. Plano, TX 75025 Phone: (972) 517-9493 FAX: (866) 472-0483
Test Sponsor Alternate Contact	NetApp, Inc. – http://www.netapp.com John Elliott – jelliott@netapp.com 7301 Kit Creek Road Building 1, RTP1.3.157 Research Triangle Park, NC 27709 Phone: (919) 476-5752 FAX: (919) 476-5603
Auditor	Storage Performance Council – http://www.storageperformance.org Walter E. Baker – AuditService@StoragePerformance.org 643 Bair Island Road, Suite 103 Redwood City, CA 94063 Phone: (650) 556-9384 FAX: (650) 556-9385

Revision Information and Key Dates

Revision Information and Key Dates	
SPC-1 Specification revision number	V1.14
SPC-1 Workload Generator revision number	V2.3.0
Date Results were first used publicly	February 19, 2014
Date the FDR was submitted to the SPC	February 19, 2014
Date the Priced Storage Configuration is available for shipment to customers	currently available
Date the TSC completed audit certification	TBD

Tested Storage Product (TSP) Description

Powered by Data ONTAP and optimized for scale out, the FAS8000 series unifies your storage infrastructure and has the flexibility to keep up with changing business needs while delivering on core IT requirements for uptime, scalability and cost-efficiency.

The FAS8000 features a multi-processor Intel chip set and leverages high-performance memory modules, NVRAM to accelerate and optimize writes, and an I/O-tuned PCIe gen3 architecture that maximizes application throughput. Building on a decade of multi-core optimization, Data ONTAP drives the latest cores and increased core counts to keep up with continuous growth in storage demands. The result is a flexible, efficient I/O design capable of supporting large numbers of high-speed network connections and massive capacity scaling.

By delivering more onboard ports to support drive, cluster, and host connectivity, the FAS8000 offers exceptional flexibility and expandability in an extremely dense package. Integrated unified target adapter (UTA) ports support 16Gb Fibre Channel, 10GbE, or FCoE, so your storage is ready on day one for whatever choices the future holds.

Summary of Results

SPC-1 Reported Data	
Tested Storage Product (TSP) Name: NetApp FAS8040 (2-node cluster)	
Metric	Reported Result
SPC-1 IOPS™	86,072.33
SPC-1 Price-Performance™	\$5.76/SPC-1 IOPS™
Total ASU Capacity	32,219.301 GB
Data Protection Level	Protected 2 (<i>RAID DP™</i>)
Total Price	\$495,652.43
Currency Used	U.S. Dollars
Target Country for availability, sales and support	USA

SPC-1 IOPS™ represents the maximum I/O Request Throughput at the 100% load point.

SPC-1 Price-Performance™ is the ratio of **Total Price** to **SPC-1 IOPS™**.

Total ASU (Application Storage Unit) Capacity represents the total storage capacity available to be read and written in the course of executing the SPC-1 benchmark.

A **Data Protection Level** of **Protected 2** using *RAID DP™*, which provides double-parity RAID protection against data loss with negligible performance overhead and no cost penalty compared to single-parity RAID. Additional information is available at the following location: <http://www.netapp.com/us/products/platform-os/raid-dp.aspx>.

Protected 2: *The single point of failure of any component in the configuration will not result in permanent loss of access to or integrity of the SPC-1 Data Repository.*

Total Price includes the cost of the Priced Storage Configuration plus three years of hardware maintenance and software support as detailed on page 17.

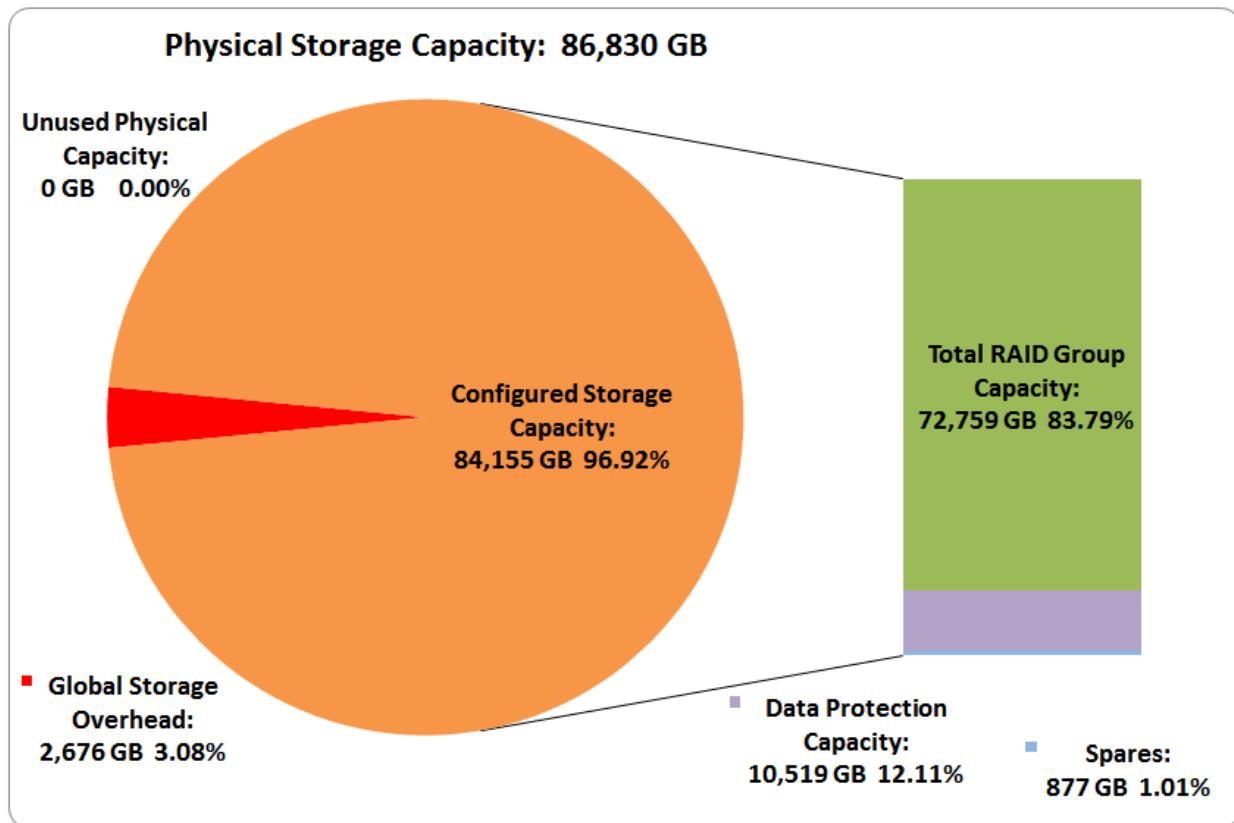
Currency Used is formal name for the currency used in calculating the **Total Price** and **SPC-1 Price-Performance™**. That currency may be the local currency of the **Target Country** or the currency of a difference country (*non-local currency*).

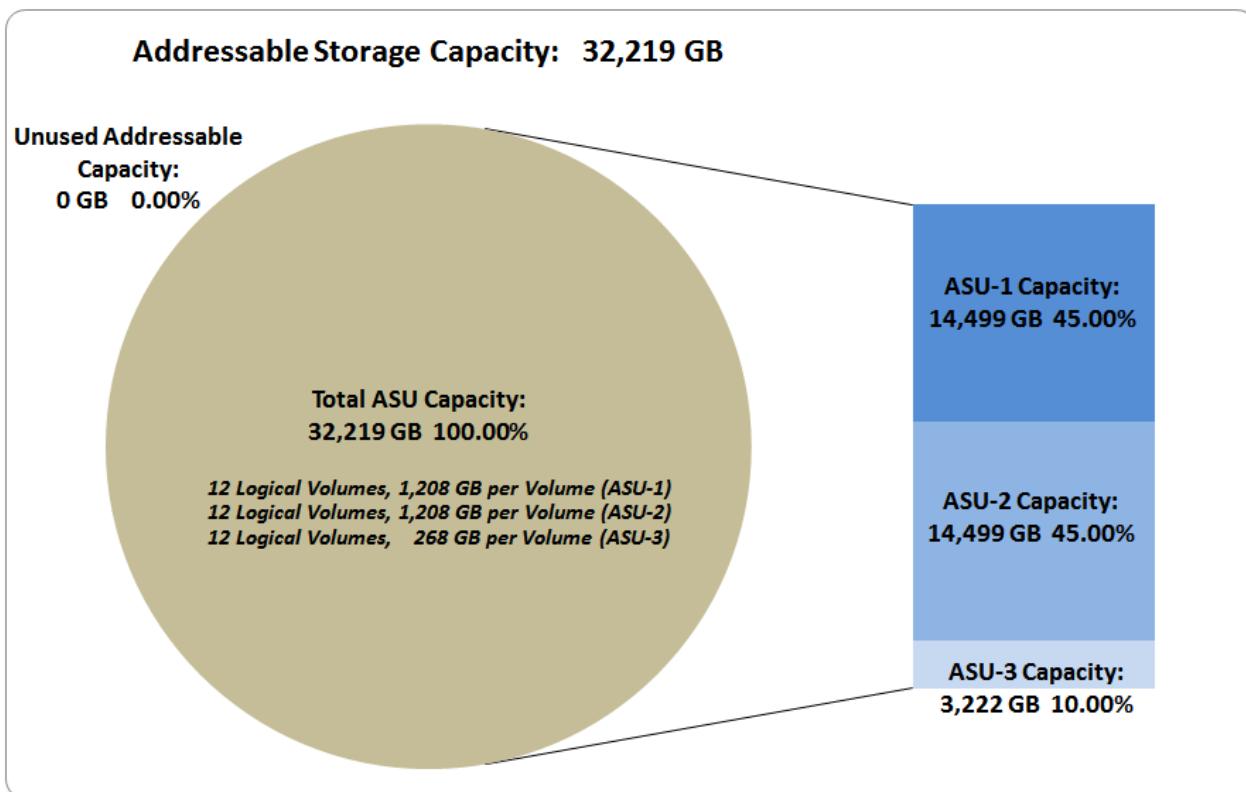
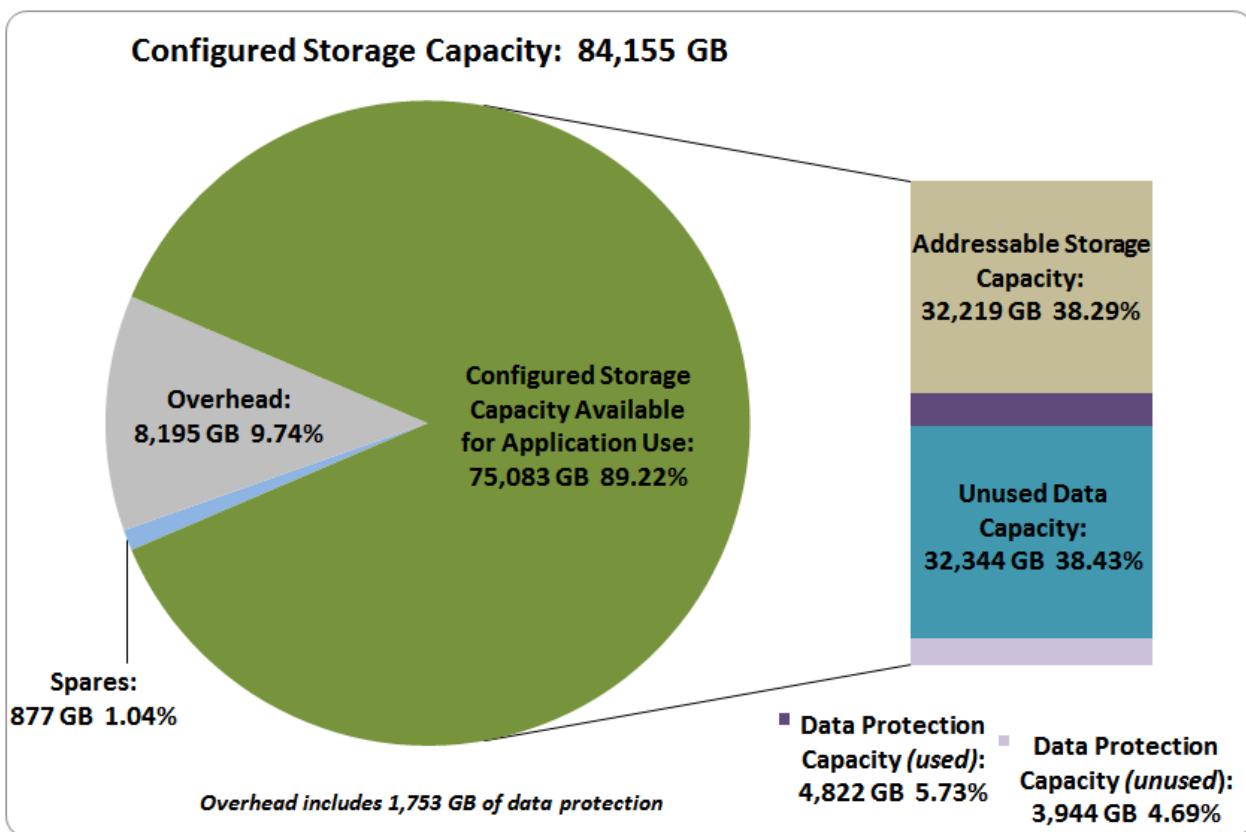
The **Target Country** is the country in which the Priced Storage Configuration is available for sale and in which the required hardware maintenance and software support is provided either directly from the Test Sponsor or indirectly via a third-party supplier.

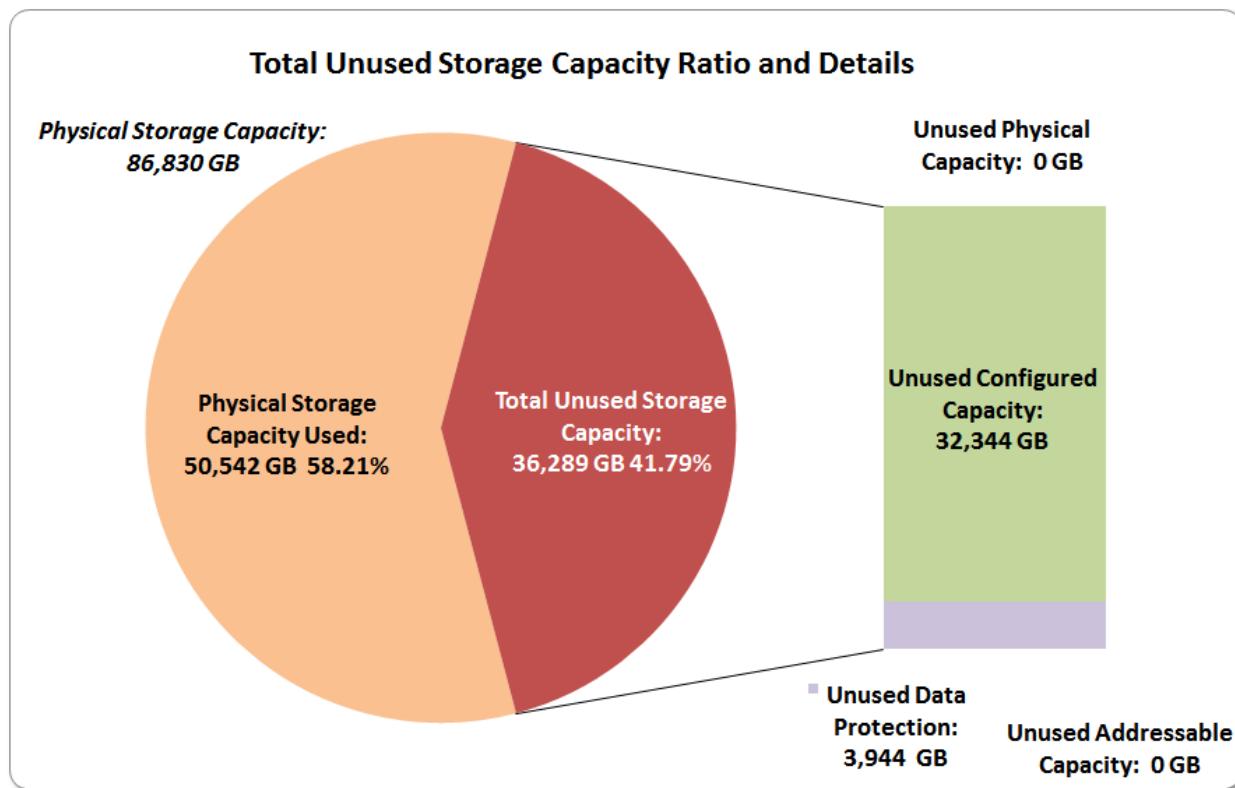
Storage Capacities, Relationships, and Utilization

The following four charts and table document the various storage capacities, used in this benchmark, and their relationships, as well as the storage utilization values required to be reported.

The capacity values in each of the following four charts are listed as integer values, for readability, rather than the decimal values listed elsewhere in this document.







SPC-1 Storage Capacity Utilization	
Application Utilization	37.11%
Protected Application Utilization	44.68%
Unused Storage Ratio	41.79%

Application Utilization: Total ASU Capacity ($32,219.301\text{ GB}$) divided by Physical Storage Capacity ($86.830.090\text{ GB}$).

Protected Application Utilization: Total ASU Capacity ($32,219.301\text{ GB}$) plus total Data Protection Capacity ($10,519.314\text{ GB}$) minus unused Data Protection Capacity ($3,944.408\text{ GB}$) divided by Physical Storage Capacity ($86.830.090\text{ GB}$).

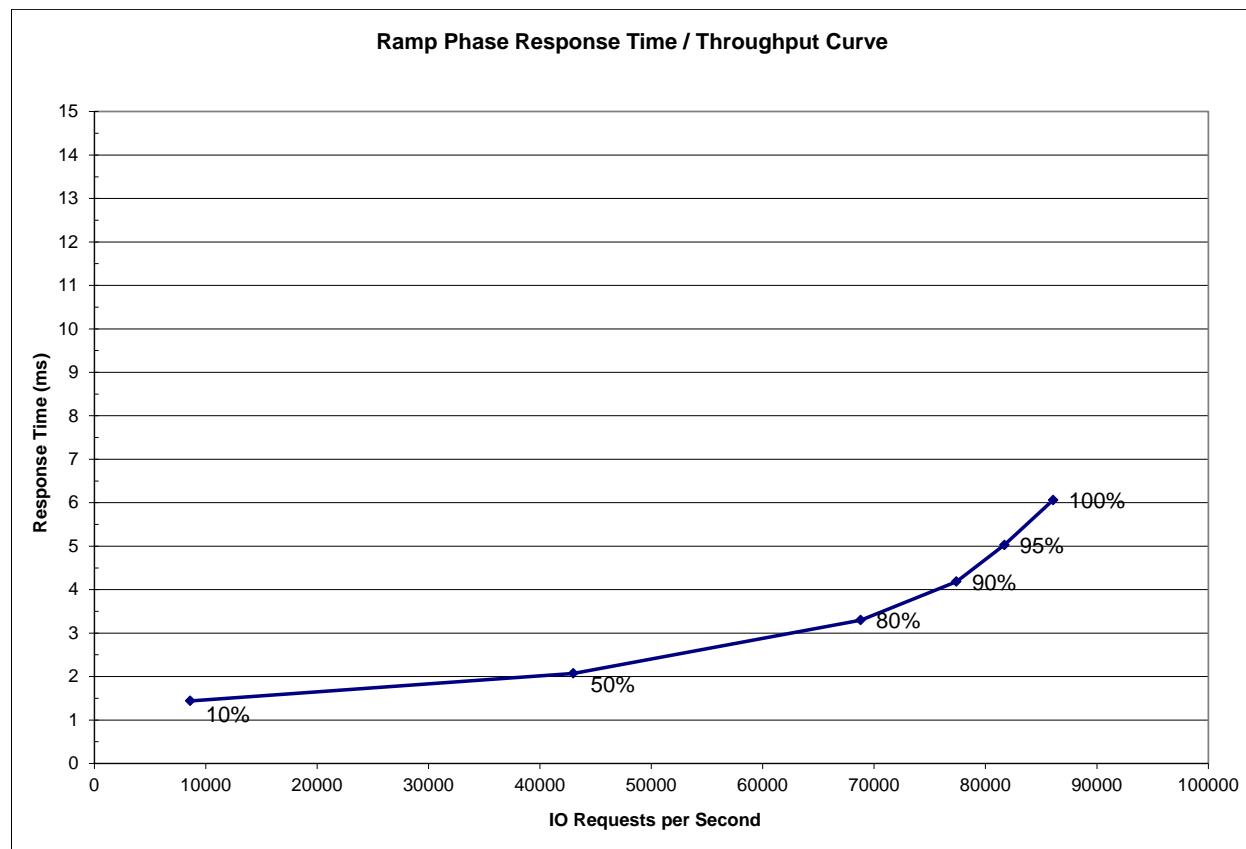
Unused Storage Ratio: Total Unused Capacity ($36,288.553\text{ GB}$) divided by Physical Storage Capacity ($86.830.090\text{ GB}$) and may not exceed 45%.

Detailed information for the various storage capacities and utilizations is available on pages 25-26.

Response Time – Throughput Curve

The Response Time-Throughput Curve illustrates the Average Response Time (milliseconds) and I/O Request Throughput at 100%, 95%, 90%, 80%, 50%, and 10% of the workload level used to generate the SPC-1 IOPS™ metric.

The Average Response Time measured at any of the above load points cannot exceed 30 milliseconds or the benchmark measurement is invalid.



Response Time – Throughput Data

	10% Load	50% Load	80% Load	90% Load	95% Load	100% Load
I/O Request Throughput	8,599.29	43,003.32	68,797.42	77,390.38	81,693.51	86,072.33
Average Response Time (ms):						
All ASUs	1.44	2.07	3.30	4.19	5.03	6.06
ASU-1	1.73	2.48	3.91	4.97	6.05	7.42
ASU-2	2.11	3.05	4.60	5.56	6.32	7.11
ASU-3	0.53	0.78	1.44	1.93	2.30	2.72
Reads	2.71	3.89	5.90	7.28	8.57	10.12
Writes	0.62	0.89	1.60	2.17	2.72	3.42

Priced Storage Configuration Pricing

	Description	QTY	List Price	Extended List
FAS8040A-001-R6	FAS8040 High Availability System	2	\$ 22,350.00	\$ 44,700.00
DS2246-1011-24S-0P-R6-C	DSK SHLF,24x450GB,10K,0P,-C	8	\$ 25,560.00	\$ 204,480.00
OS-ONTAP-CAP2	OS Enable,Per-0.1TB,ONTAP,Perf-Stor	844	\$ 92.00	\$ 77,648.00
SW-2-8040A-FCP-C	SW-2-8040A-FCP-C - FCP protocol	2	\$ 7,000.00	\$ 14,000.00
X-6505-24-16G-1R-R6	Switch,Brocade 6505 24-Pt w/16Gb SWL SFP+ Ent	2	\$ 19,580.00	\$ 39,160.00
X-SFP-H10GB-CU1M-R6-C	Cisco N5020 10GBase Copper SFP+cable, 1m, -C, R6 (Cluster interconnect)	2	\$ 112.00	\$ 224.00
X1973A-R6	Flash Cache 512GB PCIe Module 2	2	\$ 27,050.00	\$ 54,100.00
X6524-R6	Cable,Cntlr-Shelf/Switch,2m,Pair,LC/LC,Op	16	\$ 125.00	\$ 2,000.00
X2065A-EN-R6-C	HBA SAS 4-Port Copper 3/6 Gb QSFP PCIe,EN,-C	2	\$ 1,400.00	\$ 2,800.00
X1095A-R6	HBA Qlogic QLE2562 2-Port 8Gb PCIe	4	\$ 2,005.00	\$ 8,020.00
X6558-R6-C	Cable, SAS Cntlr-Shelf/Shelf-HA, 2m, -C	16	\$ 125.00	\$ 2,000.00
X6557-R6	Cable,SAS Cntlr-Shelf/Shelf-Shelf/HA,0.5m	8	\$ 100.00	\$ 800.00
X8712C-R6-C	PDU, 1-Phase, 24 Outlet, 30A, NEMA, -C, R6	2	\$ 550.00	\$ 1,100.00
X870D-EN-R6-C	Cab,Deep,HeavyDuty,Empty,No PDU,No Rail,EN,-C	1	\$ 3,595.00	\$ 3,595.00
X8778-R6-C	Mounting Bracket, Tie-Down, 32X0, -C, R6	2	\$ 50.00	\$ 100.00
CS-A-INST-4R	SupportEdge Standard Replace 4hr, Hardware Support: 3 years	1	\$ 40,925.43	\$ 40,925.43
Total (\$)				\$ 495,652.43

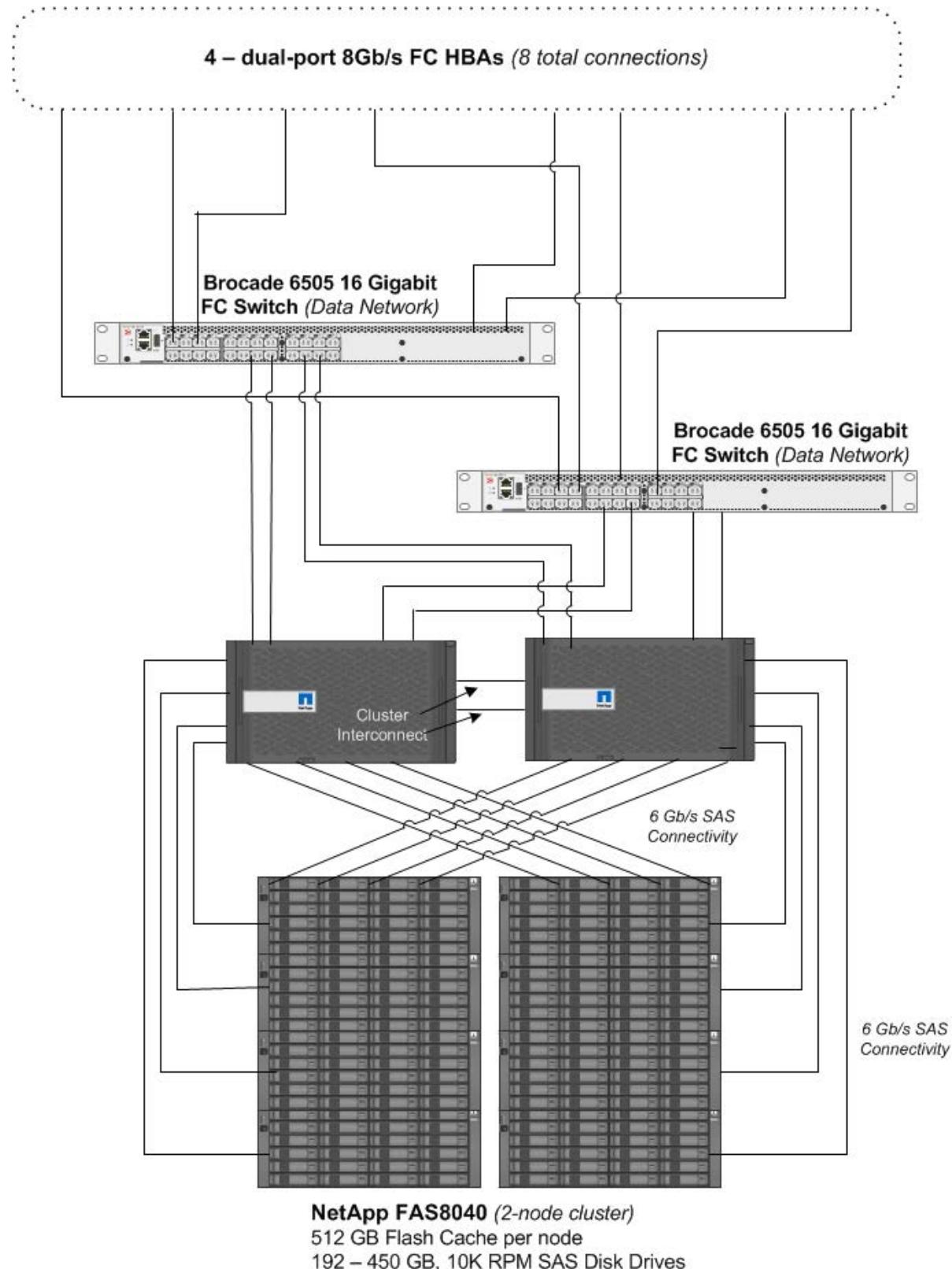
The above pricing includes hardware maintenance and software support for three years, 7 days per week, 24 hours per day. The hardware maintenance and software support provides the following:

- Acknowledgement of new and existing problems with four (4) hours.
- Onsite presence of a qualified maintenance engineer or provision of a customer replaceable part within four (4) hours of the above acknowledgement for any hardware failure that results in an inoperative Price Storage Configuration that can be remedied by the repair or replacement of a Priced Storage Configuration component.

Differences between the Tested Storage Configuration (TSC) and Priced Storage Configuration

There were no differences between the TSC and the Priced Storage Configuration.

Priced Storage Configuration Diagram



Priced Storage Configuration Components

Priced Storage Configuration
4 – QLogic QLE2562 dual-port, 8 Gbps, FC HBAs
2 – Brocade 6505 24 port, 16Gb switches with SFPs
NetApp FAS8040 (2-node cluster)
2 – controller nodes, each with
32 GB memory/cache (<i>64 GB total</i>)
512 GB Flash Cache (<i>1024 GB total</i>)
4 – 16Gb FC front-end connections <i>(8 connections total and used)</i>
2 – 4 port 3/6 Gb SAS adapters <i>(8 ports total and used in a Multipath High Availability (HA) configuration)</i>
2 – Cisco N5020 10GBase Copper SFP + cable (<i>cluster interconnect</i>)
8 – Disk Enclosures (<i>4 enclosures per controller node</i>)
192 – 450 GB, 10K RPM SAS Disk Drives <i>(24 disk drives per disk enclosure)</i>
2 – 24 outlet PDUs
1 – Cabinet

In each of the following sections of this document, the appropriate Full Disclosure Report requirement, from the SPC-1 benchmark specification, is stated in italics followed by the information to fulfill the stated requirement.

CONFIGURATION INFORMATION

Benchmark Configuration (BC)/Tested Storage Configuration (TSC) Diagram

Clause 9.4.3.4.1

A one page Benchmark Configuration (BC)/Tested Storage Configuration (TSC) diagram shall be included in the FDR...

The Benchmark Configuration (BC)/Tested Storage Configuration (TSC) is illustrated on page [21 \(Benchmark Configuration/Tested Storage Configuration Diagram\)](#).

Storage Network Configuration

Clause 9.4.3.4.1

...

5. *If the TSC contains network storage, the diagram will include the network configuration. If a single diagram is not sufficient to illustrate both the Benchmark Configuration and network configuration in sufficient detail, the Benchmark Configuration diagram will include a high-level network illustration as shown in Figure 9-8. In that case, a separate, detailed network configuration diagram will also be included as described in Clause 9.4.3.4.2.*

Clause 9.4.3.4.2

If a storage network was configured as a part of the Tested Storage Configuration and the Benchmark Configuration diagram described in Clause 9.4.3.4.1 contains a high-level illustration of the network configuration, the Executive Summary will contain a one page topology diagram of the storage network as illustrated in Figure 9-9.

The storage network portion of the Benchmark Configuration(BC)/Tested Storage Configuration (TSC) is illustrated on page [21 \(Benchmark Configuration/Tested Storage Configuration Diagram\)](#).

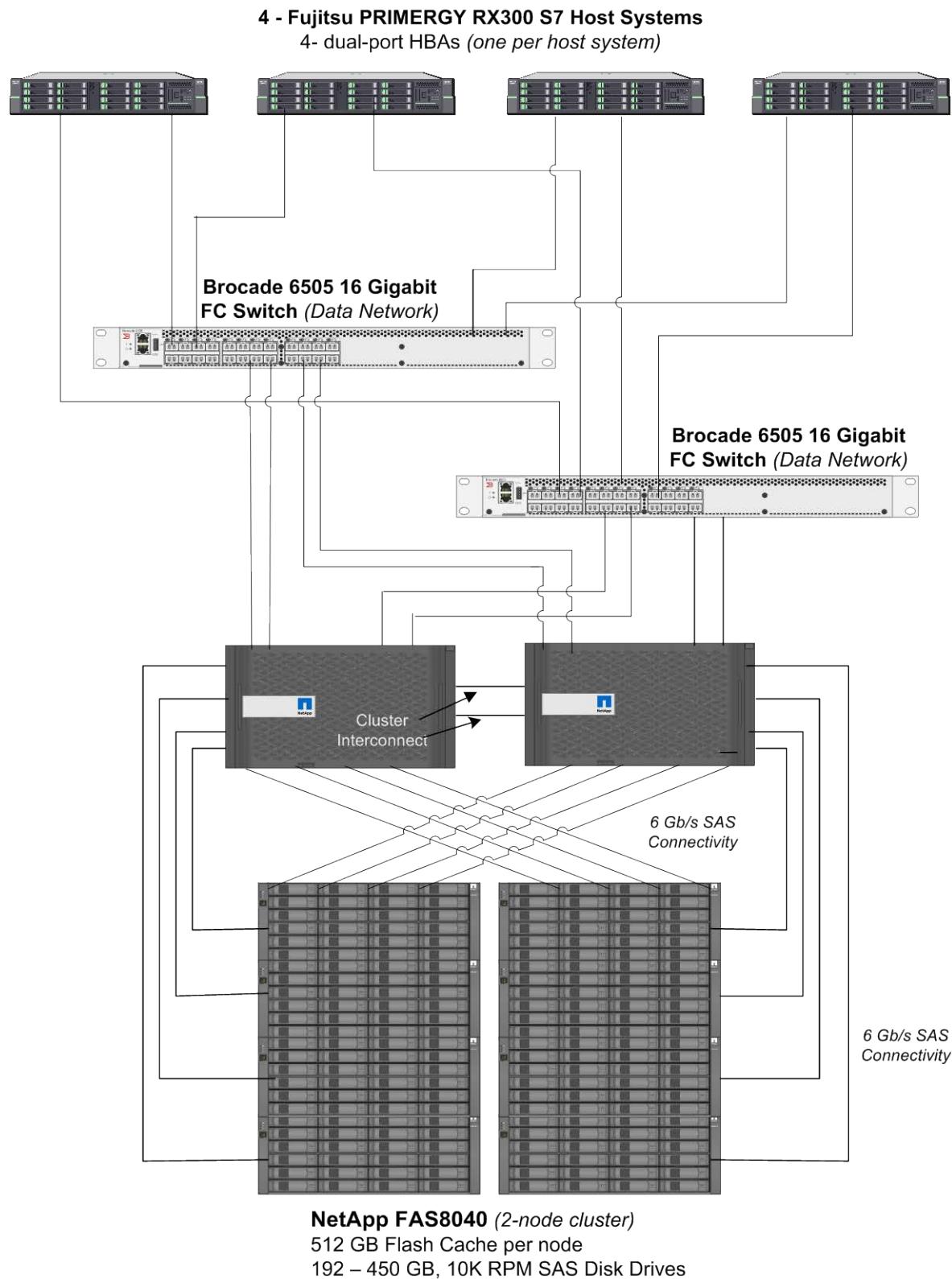
Host System(s) and Tested Storage Configuration (TSC) Table of Components

Clause 9.4.3.4.3

The FDR will contain a table that lists the major components of each Host System and the Tested Storage Configuration (TSC).

The Host System(s) and TSC table of components may be found on page [22 \(Host Systems and Tested Storage Configuration Components\)](#).

Benchmark Configuration/Tested Storage Configuration Diagram



Host Systems and Tested Storage Configuration Components

Host Systems
4 – Fujitsu RX300 S7 servers , each with:
2 – Intel® Xeon® 2.30 GHz E5-2630 processors each with 6 cores, 15 MB shared cache
132,092,332 KB main memory
Red Hat Enterprise Linux Server 6.4 (64-bit)
PCIe 3.0
Priced Storage Configuration
4 – QLogic QLE2562 dual-port, 8 Gbps, FC HBAs
2 – Brocade 6505 24 port, 16Gb switches with SFPs
NetApp FAS8040 (2-node cluster)
2 – controller nodes, each with
32 GB memory/cache (<i>64 GB total</i>)
512 GB Flash Cache (<i>1024 GB total</i>)
4 – 16Gb FC front-end connections (<i>8 connections total and used</i>)
2 – 4 port 3/6 Gb SAS adapters (<i>8 ports total and used in a Multipath High Availability (HA) configuration</i>)
2 – Cisco N5020 10GBase Copper SFP + cable (<i>cluster interconnect</i>)
8 – Disk Enclosures (<i>4 enclosures per controller node</i>)
192 – 450 GB, 10K RPM SAS Disk Drives (<i>24 disk drives per disk enclosure</i>)
2 – 24 outlet PDUs
1 – Cabinet

Customer Tunable Parameters and Options

Clause 9.4.3.5.1

All Benchmark Configuration (BC) components with customer tunable parameter and options that have been altered from their default values must be listed in the FDR. The FDR entry for each of those components must include both the name of the component and the altered value of the parameter or option. If the parameter name is not self-explanatory to a knowledgeable practitioner, a brief description of the parameter's use must also be included in the FDR entry.

[Appendix B: Customer Tunable Parameters and Options](#) on page [66](#) contains the customer tunable parameters and options that have been altered from their default values for this benchmark.

Tested Storage Configuration (TSC) Description

Clause 9.4.3.5.2

The FDR must include sufficient information to recreate the logical representation of the TSC. In addition to customer tunable parameters and options (Clause 4.2.4.5.3), that information must include, at a minimum:

- *A diagram and/or description of the following:*
 - *All physical components that comprise the TSC. Those components are also illustrated in the BC Configuration Diagram in Clause 9.2.4.4.1 and/or the Storage Network Configuration Diagram in Clause 9.2.4.4.2.*
 - *The logical representation of the TSC, configured from the above components that will be presented to the Workload Generator.*
- *Listings of scripts used to create the logical representation of the TSC.*
- *If scripts were not used, a description of the process used with sufficient detail to recreate the logical representation of the TSC.*

[Appendix C: Tested Storage Configuration \(TSC\) Creation](#) on page [68](#) contains the detailed information that describes how to create and configure the logical TSC.

SPC-1 Workload Generator Storage Configuration

Clause 9.4.3.5.3

The FDR must include all SPC-1 Workload Generator storage configuration commands and parameters.

The SPC-1 Workload Generator storage configuration commands and parameters for this measurement appear in [Appendix D: SPC-1 Workload Generator Storage Commands and Parameters](#) on page [97](#).

ASU Pre-Fill

Clause 5.3.3

Each of the three SPC-1 ASUs (ASU-1, ASU-2 and ASU-3) is required to be completely filled with specified content prior to the execution of audited SPC-1 Tests. The content is required to consist of random data pattern such as that produced by an SPC recommended tool.

The configuration file used to complete the required ASU pre-fill appears in [Appendix D: SPC-1 Workload Generator Storage Commands and Parameters](#) on page [97](#).

SPC-1 DATA REPOSITORY

This portion of the Full Disclosure Report presents the detailed information that fully documents the various SPC-1 storage capacities and mappings used in the Tested Storage Configuration. [SPC-1 Data Repository Definitions](#) on page [62](#) contains definitions of terms specific to the SPC-1 Data Repository.

Storage Capacities and Relationships

Clause 9.4.3.6.1

Two tables and four charts documenting the storage capacities and relationships of the SPC-1 Storage Hierarchy (Clause 2.1) shall be included in the FDR. ... The capacity value in each chart may be listed as an integer value, for readability, rather than the decimal value listed in the table below.

SPC-1 Storage Capacities

The Physical Storage Capacity consisted of 86,830.090 GB distributed as follows:

- 43,427.440 GB distributed over 95 disk drives, each with a formatted capacity of 457.131 GB.
- 43,402.650 GB distributed over 97 disk drives, each with a formatted capacity of 447.450 GB.

There was 0.000 GB (0.00%) of Unused Storage within the Physical Storage Capacity. Global Storage Overhead consisted of 2,675.574 GB (3.08%) of the Physical Storage Capacity. There was 32,344.145 GB (38.43%) of Unused Storage within the Configured Storage Capacity. The Total ASU Capacity utilized 100% of the Addressable Storage Capacity resulting in 0.000 GB (0.00%) of Unused Storage within the Addressable Storage Capacity. The Data Protection (*RAID DP™*) capacity was 10,519.314 GB of which 6,574.906 GB was utilized. The total Unused Storage capacity was 36,288.553 GB.

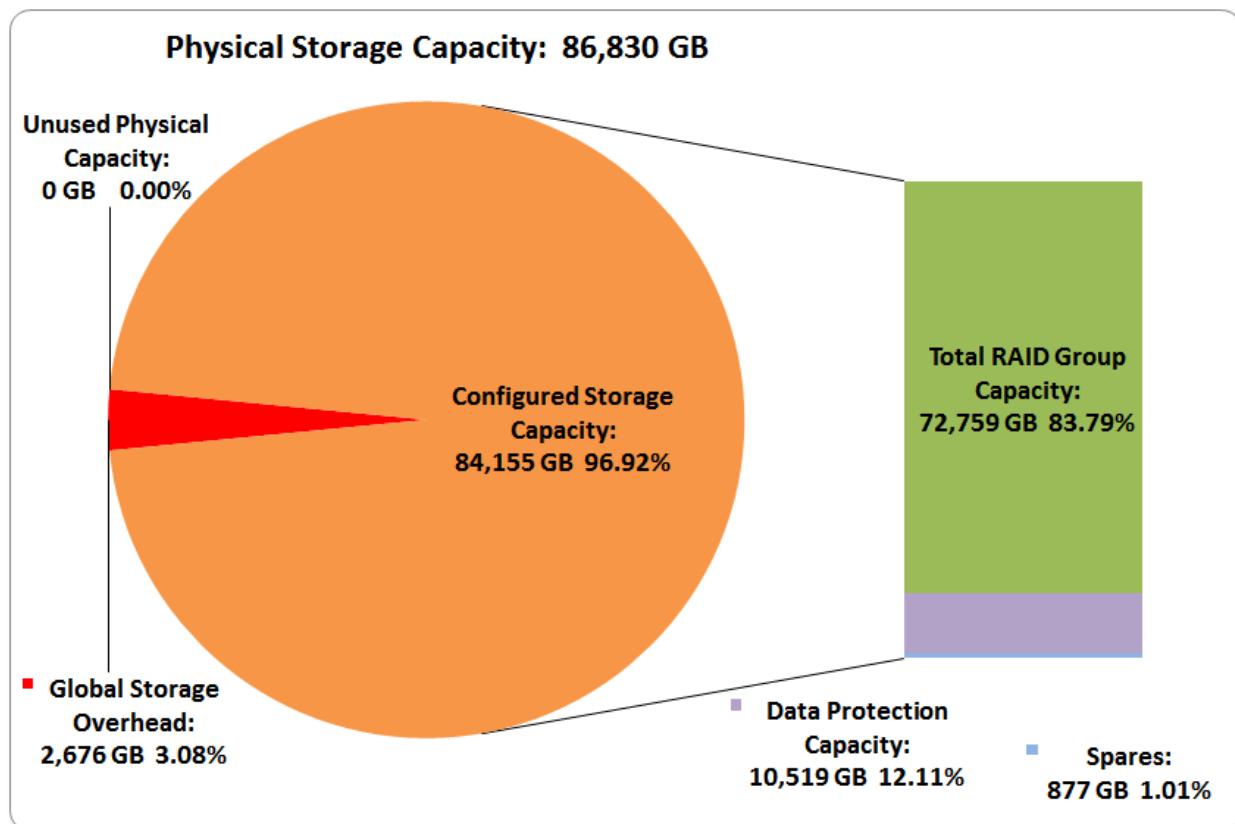
Note: The configured Storage Devices may include additional storage capacity reserved for system overhead, which is not accessible for application use. That storage capacity may not be included in the value presented for Physical Storage Capacity.

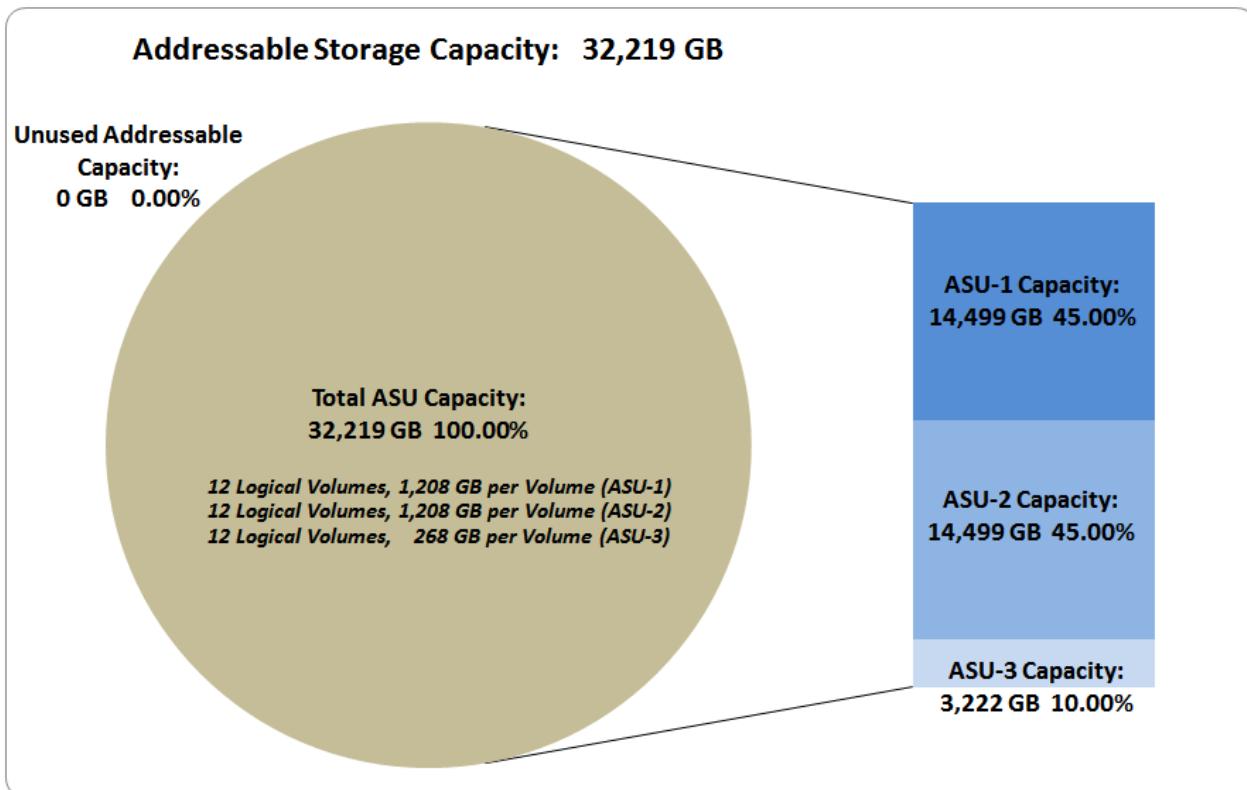
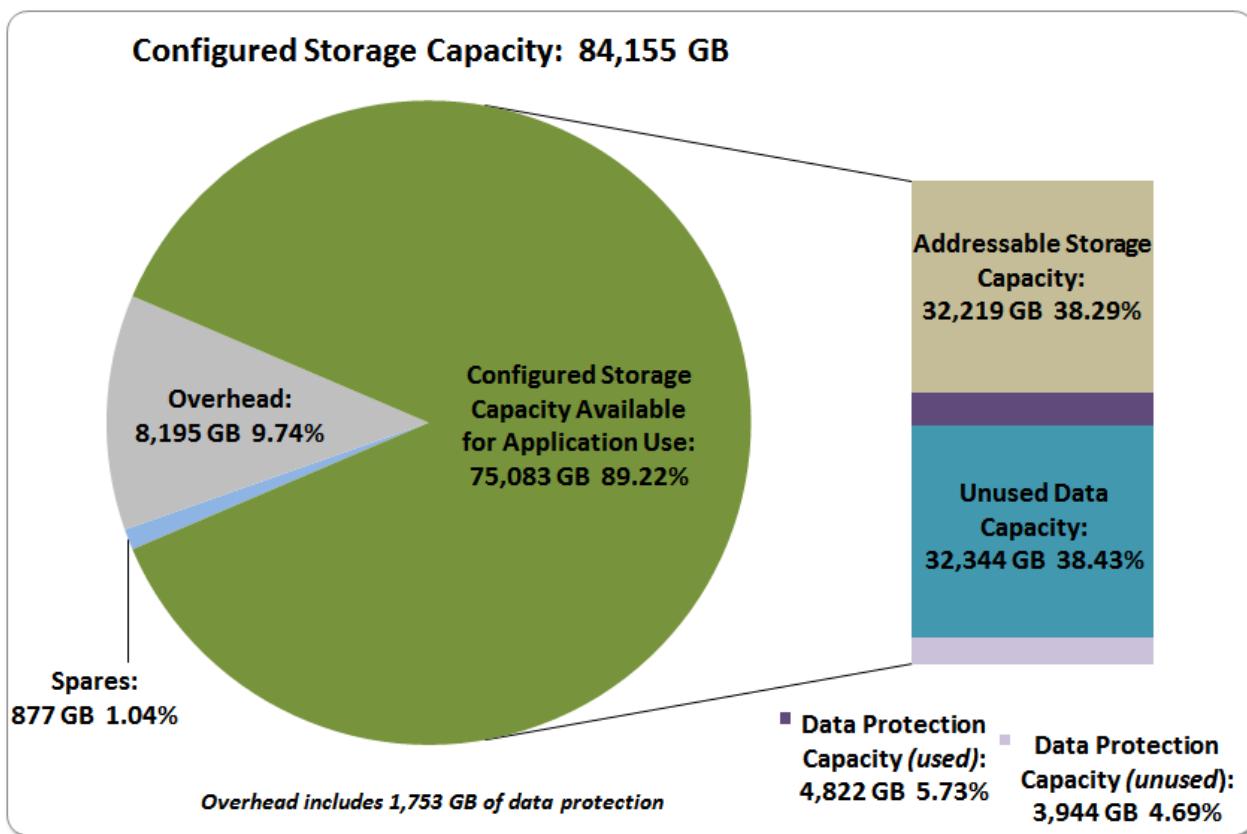
SPC-1 Storage Capacities		
Storage Hierarchy Component	Units	Capacity
Total ASU Capacity	Gigabytes (GB)	32,219.301
Addressable Storage Capacity	Gigabytes (GB)	32,219.301
Configured Storage Capacity	Gigabytes (GB)	84,154.515
Physical Storage Capacity	Gigabytes (GB)	86,830.090
Data Protection (<i>RAID DP™</i>)	Gigabytes (GB)	10,519.314
Required Storage (<i>spares, overhead</i>)	Gigabytes (GB)	9,071.755
Global Storage Overhead	Gigabytes (GB)	2,675.574
Total Unused Storage	Gigabytes (GB)	36,288.553

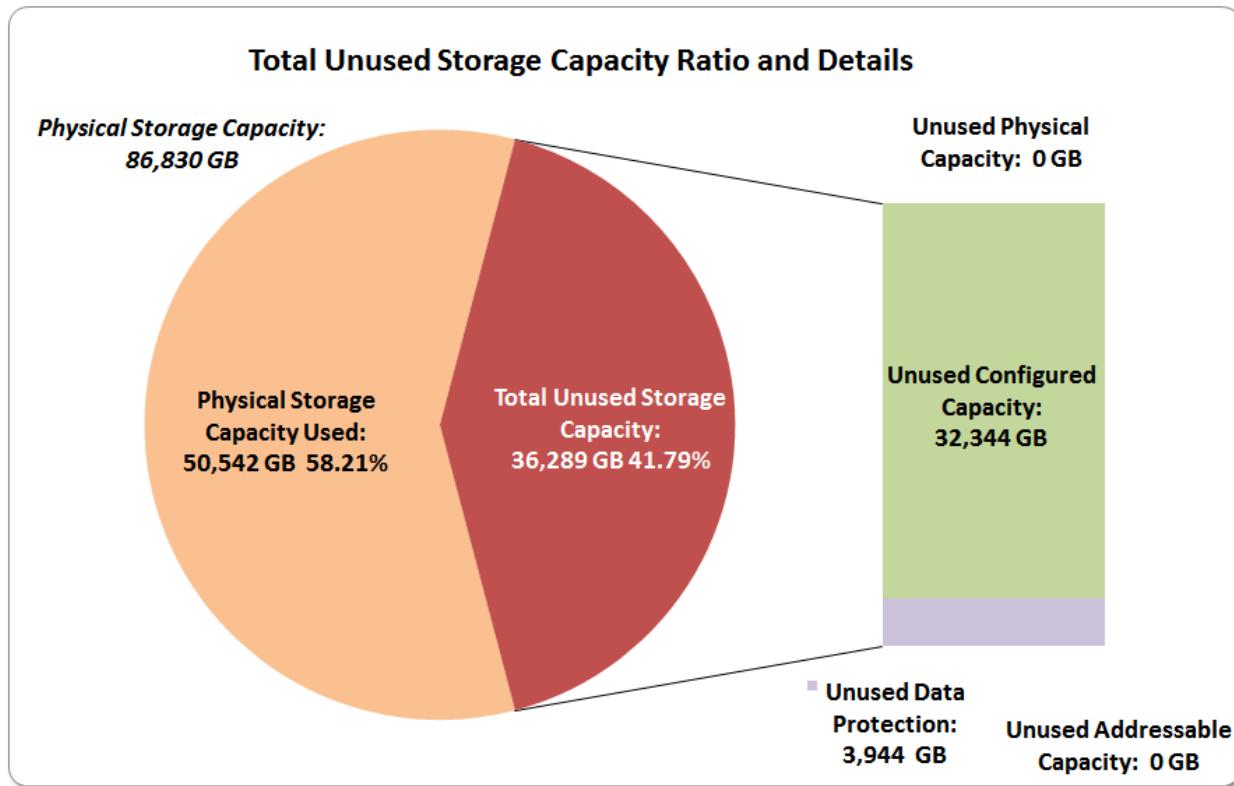
SPC-1 Storage Hierarchy Ratios

	Addressable Storage Capacity	Configured Storage Capacity	Physical Storage Capacity
Total ASU Capacity	100.00%	38.29%	37.11%
Required for Data Protection (RAID DP™)		12.50%	12.11%
Addressable Storage Capacity		38.29%	37.11%
Required Storage (spares, overhead)		10.78%	10.45%
Configured Storage Capacity			96.92%
Global Storage Overhead			3.08%
Unused Storage:			
Addressable	0.00%		
Configured		38.43%	
Physical			0.00%

SPC-1 Storage Capacity Charts







Storage Capacity Utilization

Clause 9.4.3.6.2

The FDR will include a table illustrating the storage capacity utilization values defined for Application Utilization (Clause 2.8.1), Protected Application Utilization (Clause 2.8.2), and Unused Storage Ratio (Clause 2.8.3).

Clause 2.8.1

Application Utilization is defined as Total ASU Capacity divided by Physical Storage Capacity.

Clause 2.8.2

Protected Application Utilization is defined as (Total ASU Capacity plus total Data Protection Capacity minus unused Data Protection Capacity) divided by Physical Storage Capacity.

Clause 2.8.3

Unused Storage Ratio is defined as Total Unused Capacity divided by Physical Storage Capacity and may not exceed 45%.

SPC-1 Storage Capacity Utilization	
Application Utilization	37.11%
Protected Application Utilization	44.68%
Unused Storage Ratio	41.79%

Logical Volume Capacity and ASU Mapping

Clause 9.4.3.6.3

A table illustrating the capacity of each ASU and the mapping of Logical Volumes to ASUs shall be provided in the FDR. ... Logical Volumes shall be sequenced in the table from top to bottom per its position in the contiguous address space of each ASU. The capacity of each Logical Volume shall be stated. ... In conjunction with this table, the Test Sponsor shall provide a complete description of the type of data protection (see Clause 2.4.5) used on each Logical Volume.

Logical Volume Capacity and Mapping		
ASU-1 (14,498.837 GB)	ASU-2 (14,498.837 GB)	ASU-3 (3,221.628 GB)
12 Logical Volumes 1,208.236 GB per Logical Volume (1,208.236 GB used per Logical Volume)	12 Logical Volumes 1,208.236 GB per Logical Volume (1,208.236 GB used per Logical Volume)	12 Logical Volumes 268.469 GB per Logical Volume (268.469 GB used per Logical Volume)

The Data Protection Level used for all Logical Volumes was [Protected 2](#) using **RAID DP™** as described on page [12](#). See “ASU Configuration” in the [IOPS Test Results File](#) for more detailed configuration information.

SPC-1 BENCHMARK EXECUTION RESULTS

This portion of the Full Disclosure Report documents the results of the various SPC-1 Tests, Test Phases, and Test Runs. An [SPC-1 glossary](#) on page 62 contains definitions of terms specific to the SPC-1 Tests, Test Phases, and Test Runs.

Clause 5.4.3

The Tests must be executed in the following sequence: Primary Metrics, Repeatability, and Data Persistence. That required sequence must be uninterrupted from the start of Primary Metrics to the completion of Persistence Test Run 1. Uninterrupted means the Benchmark Configuration shall not be power cycled, restarted, disturbed, altered, or adjusted during the above measurement sequence. If the required sequence is interrupted other than for the Host System/TSC power cycle between the two Persistence Test Runs, the measurement is invalid.

SPC-1 Tests, Test Phases, and Test Runs

The SPC-1 benchmark consists of the following Tests, Test Phases, and Test Runs:

- **Primary Metrics Test**
 - Sustainability Test Phase and Test Run
 - IOPS Test Phase and Test Run
 - Response Time Ramp Test Phase
 - 95% of IOPS Test Run
 - 90% of IOPS Test Run
 - 80% of IOPS Test Run
 - 50% of IOPS Test Run
 - 10% of IOPS Test Run (LRT)
- **Repeatability Test**
 - Repeatability Test Phase 1
 - 10% of IOPS Test Run (LRT)
 - IOPS Test Run
 - Repeatability Test Phase 2
 - 10% of IOPS Test Run (LRT)
 - IOPS Test Run
- **Data Persistence Test**
 - Data Persistence Test Run 1
 - Data Persistence Test Run 2

Each Test is an atomic unit that must be executed from start to finish before any other Test, Test Phase, or Test Run may be executed.

The results from each Test, Test Phase, and Test Run are listed below along with a more detailed explanation of each component.

“Ramp-Up” Test Runs

Clause 5.3.13

In order to warm-up caches or perform the initial ASU data migration in a multi-tier configuration, a Test Sponsor may perform a series of “Ramp-Up” Test Runs as a substitute for an initial, gradual Ramp-Up.

Clause 5.3.13.3

The “Ramp-Up” Test Runs will immediately precede the Primary Metrics Test as part of the uninterrupted SPC-1 measurement sequence.

Clause 9.4.3.7.1

If a series of “Ramp-Up” Test Runs were included in the SPC-1 measurement sequence, the FDR shall report the duration (ramp-up and measurement interval), BSU level, SPC-1 IOPS and average response time for each “Ramp-Up” Test Run in an appropriate table.

There were no “Ramp-Up” Test Runs executed.

Primary Metrics Test – Sustainability Test Phase

Clause 5.4.4.1.1

The Sustainability Test Phase has exactly one Test Run and shall demonstrate the maximum sustainable I/O Request Throughput within at least a continuous eight (8) hour Measurement Interval. This Test Phase also serves to insure that the TSC has reached Steady State prior to reporting the final maximum I/O Request Throughput result (SPC-1 IOPSTM).

Clause 5.4.4.1.2

The computed I/O Request Throughput of the Sustainability Test must be within 5% of the reported SPC-1 IOPSTM result.

Clause 5.4.4.1.4

The Average Response Time, as defined in Clause 5.1.1, will be computed and reported for the Sustainability Test Run and cannot exceed 30 milliseconds. If the Average Response time exceeds that 30-milliseconds constraint, the measurement is invalid.

Clause 9.4.3.7.2

For the Sustainability Test Phase the FDR shall contain:

1. A Data Rate Distribution graph and data table.
2. I/O Request Throughput Distribution graph and data table.
3. A Response Time Frequency Distribution graph and table.
4. An Average Response Time Distribution graph and table.
5. The human readable Test Run Results File produced by the Workload Generator (may be included in an appendix).
6. A listing or screen image of all input parameters supplied to the Workload Generator (may be included in an appendix).
7. The Measured Intensity Multiplier for each I/O stream.
8. The variability of the Measured Intensity Multiplier, as defined in Clause 5.3.13.3.

SPC-1 Workload Generator Input Parameters

The SPC-1 Workload Generator input parameters for the Sustainability, IOPS, Response Time Ramp, Repeatability, and Persistence Test Runs are documented in [Appendix E: SPC-1 Workload Generator Input Parameters](#) on Page [102](#).

Sustainability Test Results File

A link to the test results file generated from the Sustainability Test Run is listed below.

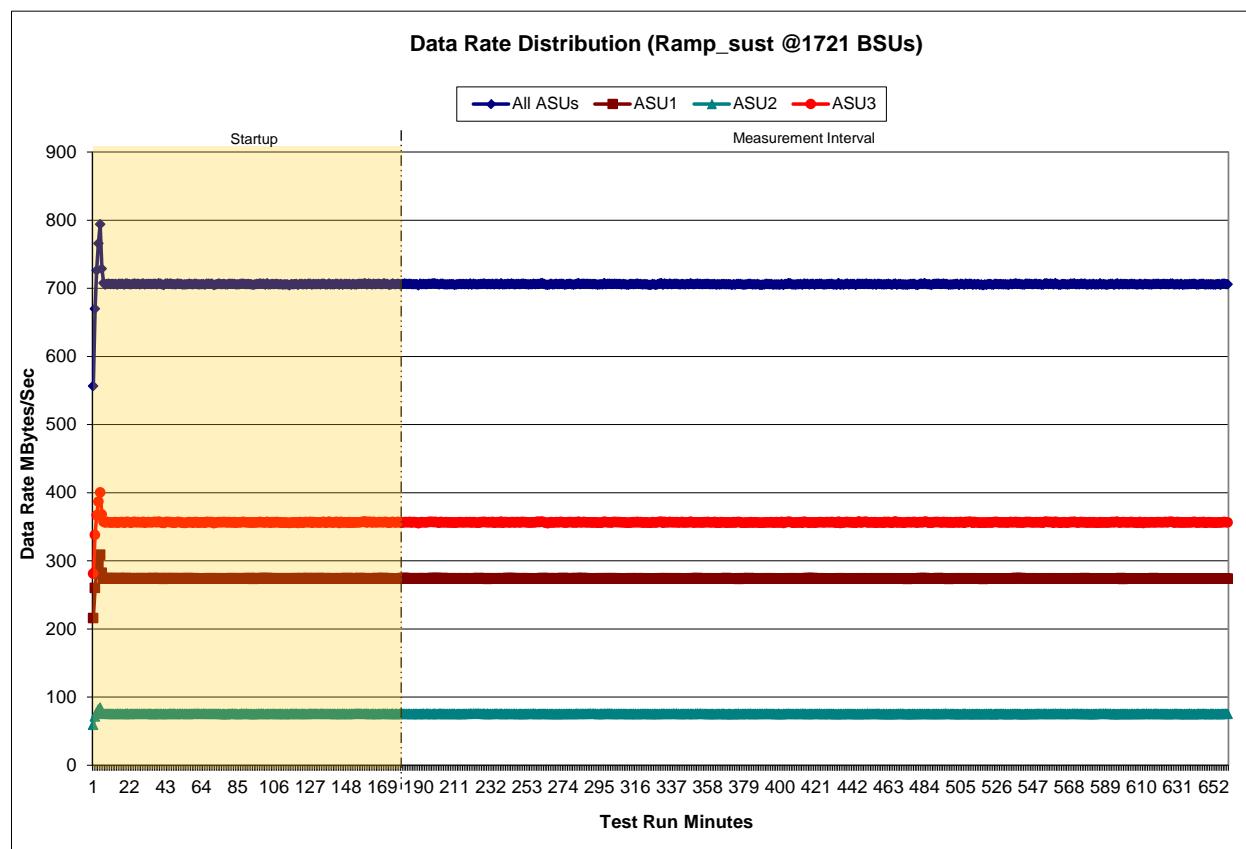
[Sustainability Test Results File](#)

Sustainability – Data Rate Distribution Data (MB/second)

The Sustainability Data Rate table of data is not embedded in this document due to its size. The table is available via the following URL:

[Sustainability Data Rate Table](#)

Sustainability – Data Rate Distribution Graph

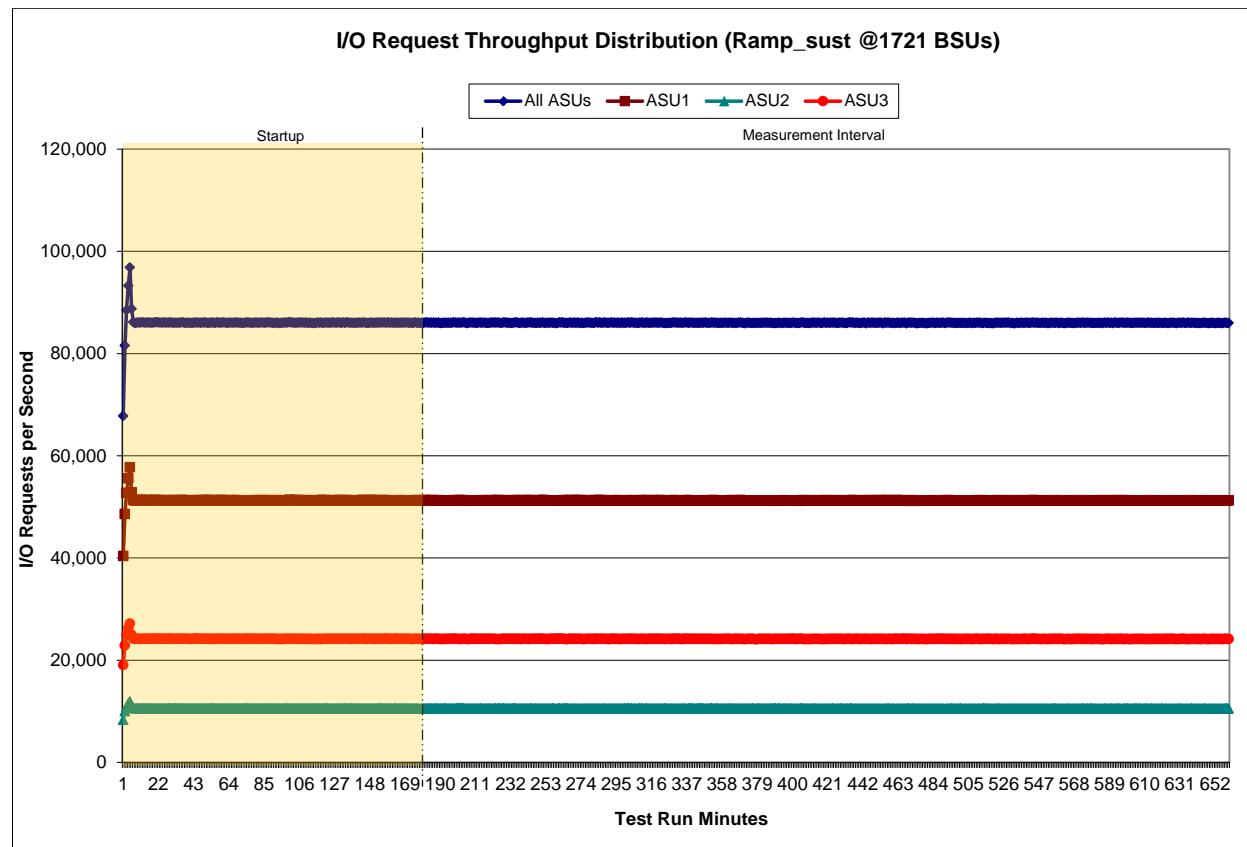


Sustainability – I/O Request Throughput Distribution Data

The Sustainability I/O Request Throughput table of data is not embedded in this document due to its size. The table is available via the following URL:

[Sustainability I/O Request Throughput Table](#)

Sustainability – I/O Request Throughput Distribution Graph

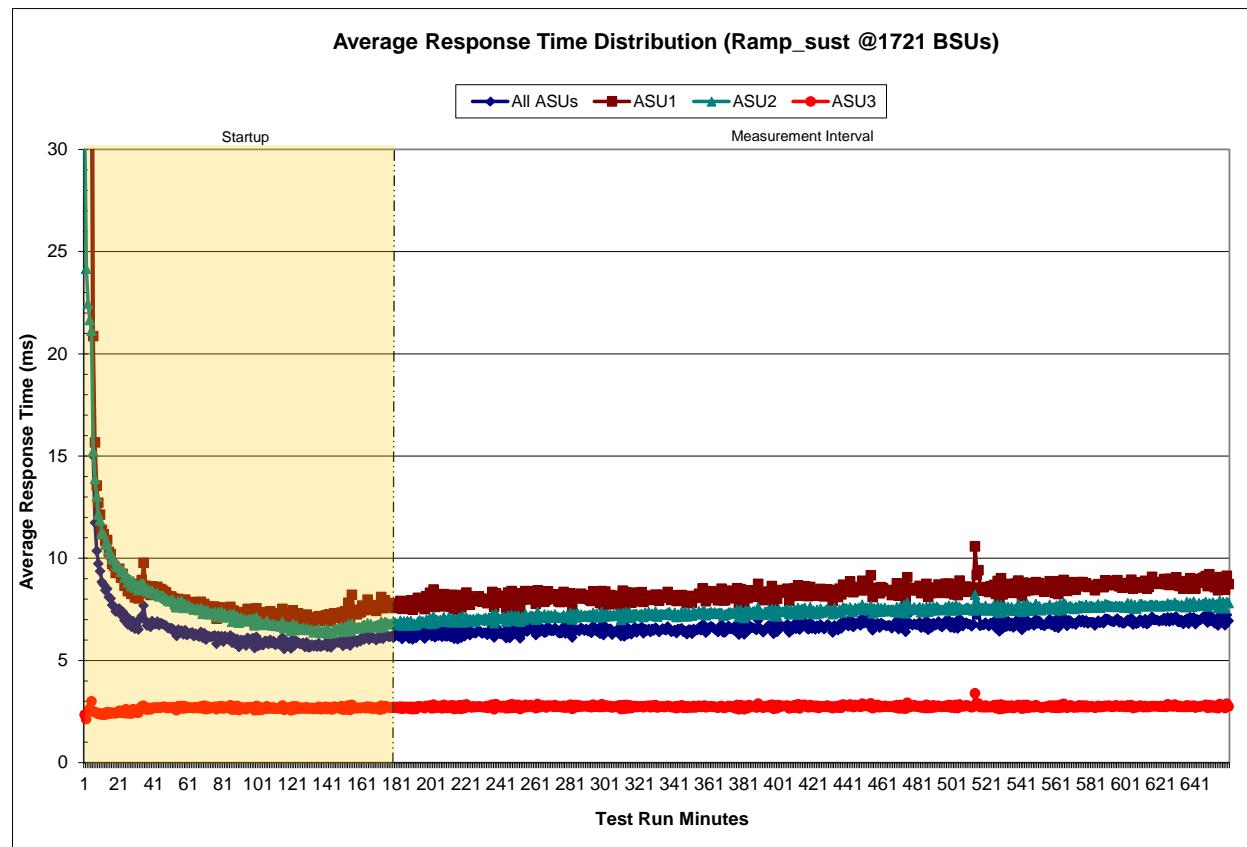


Sustainability – Average Response Time (ms) Distribution Data

The Sustainability Average Response Time table of data is not embedded in this document due to its size. The table is available via the following URL:

[Sustainability Average Response Time Table](#)

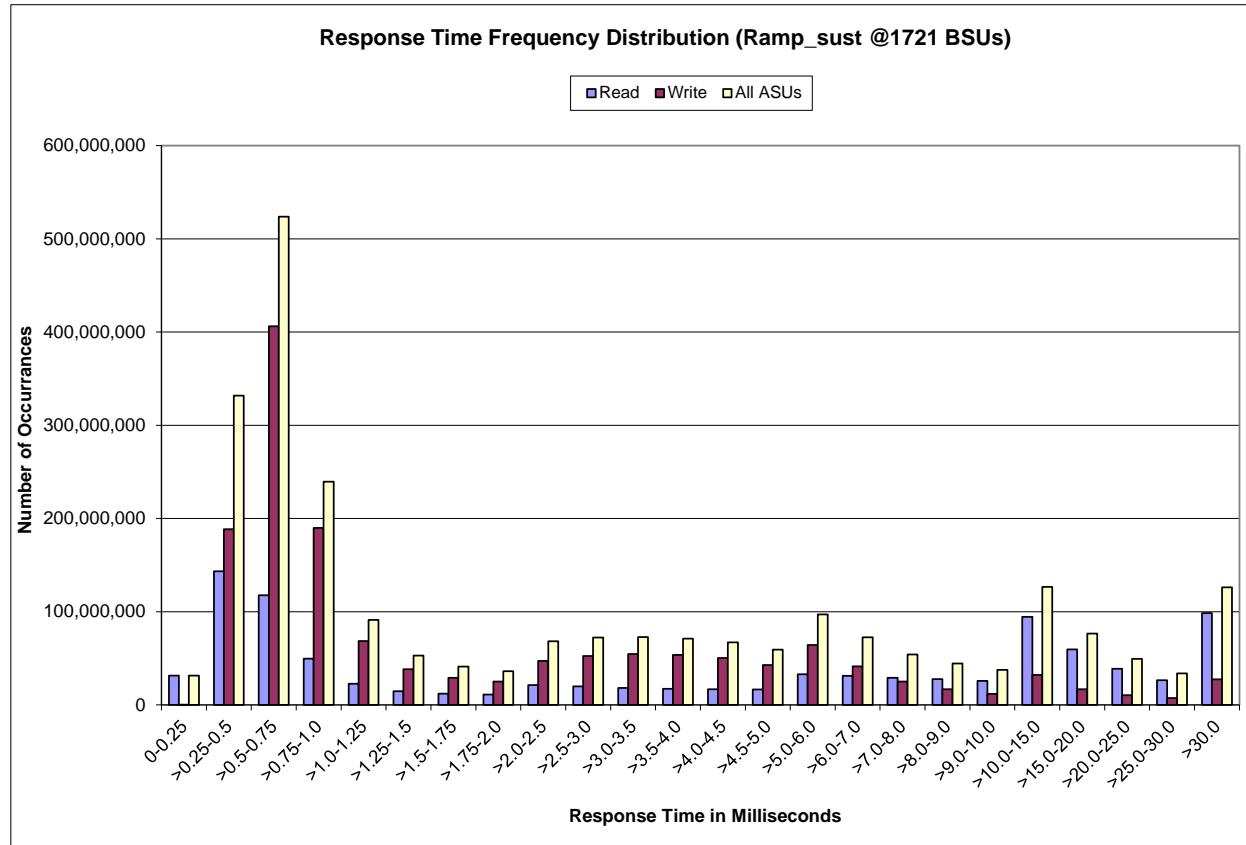
Sustainability – Average Response Time (ms) Distribution Graph



Sustainability – Response Time Frequency Distribution Data

Response Time (ms)	0-0.25	>0.25-0.5	>0.5-0.75	>0.75-1.0	>1.0-1.25	>1.25-1.5	>1.5-1.75	>1.75-2.0
Read	31,379,172	143,279,536	117,672,541	49,713,066	22,622,301	14,701,626	12,005,814	11,064,647
Write	46,168	188,599,644	406,149,813	189,853,603	68,595,078	38,208,128	29,198,740	25,168,910
All ASUs	31,425,340	331,879,180	523,822,354	239,566,669	91,217,379	52,909,754	41,204,554	36,233,557
ASU1	22,524,525	201,847,073	281,562,033	124,809,872	48,668,882	28,808,501	22,535,645	19,890,168
ASU2	8,884,068	45,735,926	49,007,854	24,774,259	10,720,746	6,234,806	4,851,929	4,378,177
ASU3	16,747	84,296,181	193,252,467	89,982,538	31,827,751	17,866,447	13,816,980	11,965,212
Response Time (ms)	>2.0-2.5	>2.5-3.0	>3.0-3.5	>3.5-4.0	>4.0-4.5	>4.5-5.0	>5.0-6.0	>6.0-7.0
Read	21,240,662	19,800,110	18,294,471	17,365,818	16,819,506	16,610,547	32,940,870	31,204,250
Write	47,181,619	52,511,140	54,594,997	53,669,110	50,352,619	42,834,355	64,236,413	41,333,206
All ASUs	68,422,281	72,311,250	72,889,468	71,034,928	67,172,125	59,444,902	97,177,283	72,537,456
ASU1	37,400,715	38,132,152	37,741,783	36,709,403	34,862,146	31,553,498	53,877,271	43,079,755
ASU2	8,533,869	8,846,325	8,657,623	8,183,709	7,474,547	6,523,785	10,927,528	8,838,016
ASU3	22,487,697	25,332,773	26,490,062	26,141,816	24,835,432	21,367,619	32,372,484	20,619,685
Response Time (ms)	>7.0-8.0	>8.0-9.0	>9.0-10.0	>10.0-15.0	>15.0-20.0	>20.0-25.0	>25.0-30.0	>30.0
Read	29,032,260	27,675,010	25,781,105	94,498,894	59,634,134	38,882,280	26,560,006	98,623,055
Write	25,214,242	16,882,683	11,867,996	32,199,398	16,943,983	10,462,149	7,317,371	27,456,268
All ASUs	54,246,502	44,557,693	37,649,101	126,698,292	76,578,117	49,344,429	33,877,377	126,079,323
ASU1	35,045,164	30,636,437	27,041,257	94,192,479	57,476,405	37,176,671	25,787,744	105,626,052
ASU2	7,202,051	6,269,776	5,529,502	19,704,418	12,495,246	8,217,368	5,606,248	17,234,574
ASU3	11,999,287	7,651,480	5,078,342	12,801,395	6,606,466	3,950,390	2,483,385	3,218,697

Sustainability – Response Time Frequency Distribution Graph



Sustainability – Measured Intensity Multiplier and Coefficient of Variation

Clause 3.4.3

IM – Intensity Multiplier: The ratio of I/Os for each I/O stream relative to the total I/Os for all I/O streams (ASU1-1 – ASU3-1) as required by the benchmark specification.

Clauses 5.1.10 and 5.3.15.2

MIM – Measured Intensity Multiplier: The Measured Intensity Multiplier represents the ratio of measured I/Os for each I/O stream relative to the total I/Os measured for all I/O streams (ASU1-1 – ASU3-1). This value may differ from the corresponding Expected Intensity Multiplier by no more than 5%.

Clause 5.3.15.3

COV – Coefficient of Variation: This measure of variation for the Measured Intensity Multiplier cannot exceed 0.2.

	ASU1-1	ASU1-2	ASU1-3	ASU1-4	ASU2-1	ASU2-2	ASU2-3	ASU3-1
IM	0.0350	0.2810	0.0700	0.2100	0.0180	0.0700	0.0350	0.2810
MIM	0.0350	0.2810	0.0700	0.2100	0.0180	0.0700	0.0350	0.2810
COV	0.002	0.001	0.002	0.001	0.003	0.002	0.002	0.001

Primary Metrics Test – IOPS Test Phase

Clause 5.4.4.2

The IOPS Test Phase consists of one Test Run at the 100% load point with a Measurement Interval of ten (10) minutes. The IOPS Test Phase immediately follows the Sustainability Test Phase without any interruption or manual intervention.

The IOPS Test Run generates the SPC-1 IOPSTM primary metric, which is computed as the I/O Request Throughput for the Measurement Interval of the IOPS Test Run.

The Average Response Time is computed for the IOPS Test Run and cannot exceed 30 milliseconds. If the Average Response Time exceeds the 30 millisecond constraint, the measurement is invalid.

Clause 9.4.3.7.3

For the IOPS Test Phase the FDR shall contain:

1. I/O Request Throughput Distribution (data and graph).
2. A Response Time Frequency Distribution.
3. An Average Response Time Distribution.
4. The human readable Test Run Results File produced by the Workload Generator.
5. A listing or screen image of all input parameters supplied to the Workload Generator.
6. The total number of I/O Requests completed in the Measurement Interval as well as the number of I/O Requests with a Response Time less than or equal to 30 milliseconds and the number of I/O Requests with a Response Time greater than 30 milliseconds.

SPC-1 Workload Generator Input Parameters

The SPC-1 Workload Generator input parameters for the Sustainability, IOPS, Response Time Ramp, Repeatability, and Persistence Test Runs are documented in [Appendix E: SPC-1 Workload Generator Input Parameters](#) on Page [102](#).

IOPS Test Results File

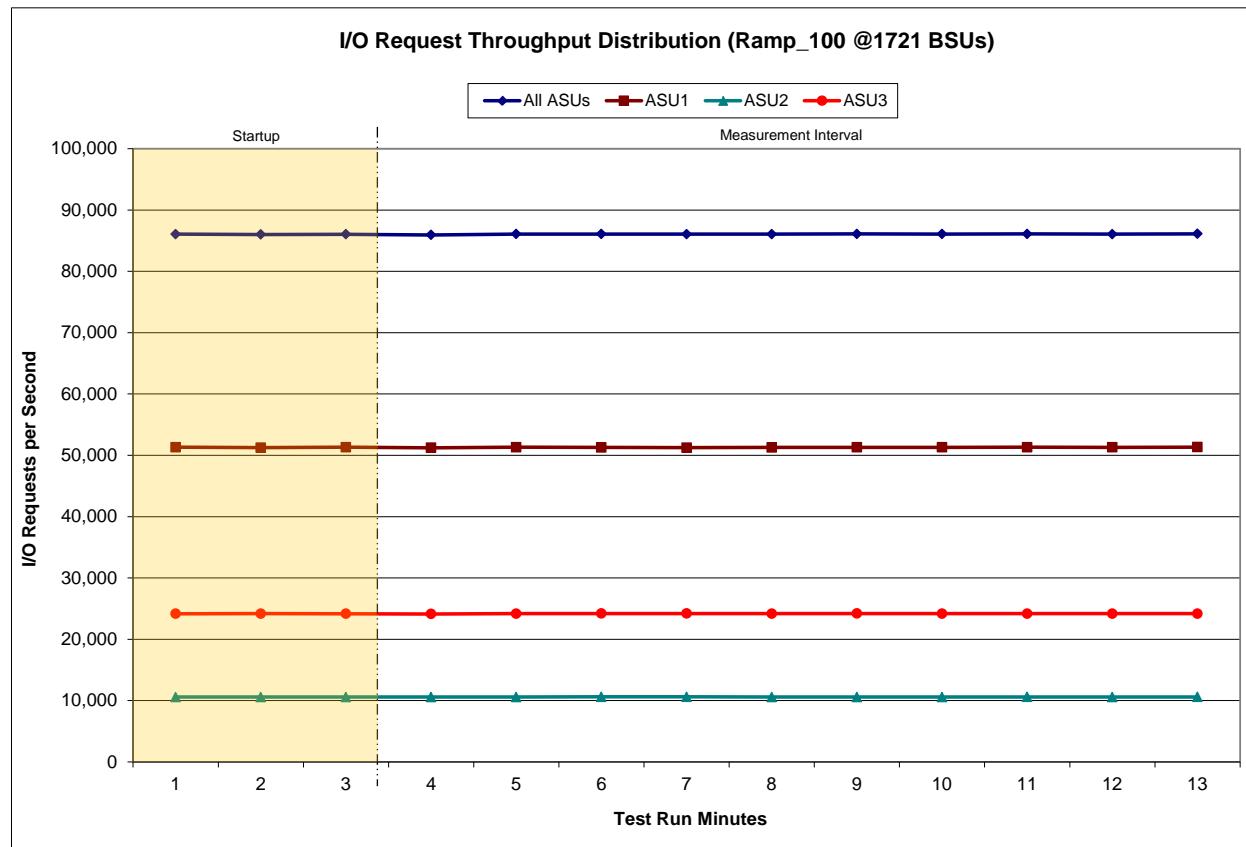
A link to the test results file generated from the IOPS Test Run is listed below.

[IOPS Test Results File](#)

IOPS Test Run – I/O Request Throughput Distribution Data

1,721 BSUs	Start	Stop	Interval	Duration
<i>Start-Up/Ramp-Up</i>	9:41:54	9:44:55	0-2	0:03:01
<i>Measurement Interval</i>	9:44:55	9:54:55	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	86,067.88	51,323.08	10,578.83	24,165.97
1	86,012.90	51,245.35	10,574.08	24,193.47
2	86,051.57	51,312.98	10,579.15	24,159.43
3	85,938.18	51,224.80	10,584.62	24,128.77
4	86,089.25	51,311.25	10,585.05	24,192.95
5	86,091.70	51,283.85	10,611.05	24,196.80
6	86,060.80	51,255.95	10,608.10	24,196.75
7	86,073.70	51,292.17	10,589.67	24,191.87
8	86,093.35	51,305.93	10,581.08	24,206.33
9	86,085.20	51,308.35	10,585.95	24,190.90
10	86,104.72	51,314.70	10,602.17	24,187.85
11	86,063.78	51,297.27	10,586.83	24,179.68
12	86,122.58	51,334.07	10,605.15	24,183.37
Average	86,072.33	51,292.83	10,593.97	24,185.53

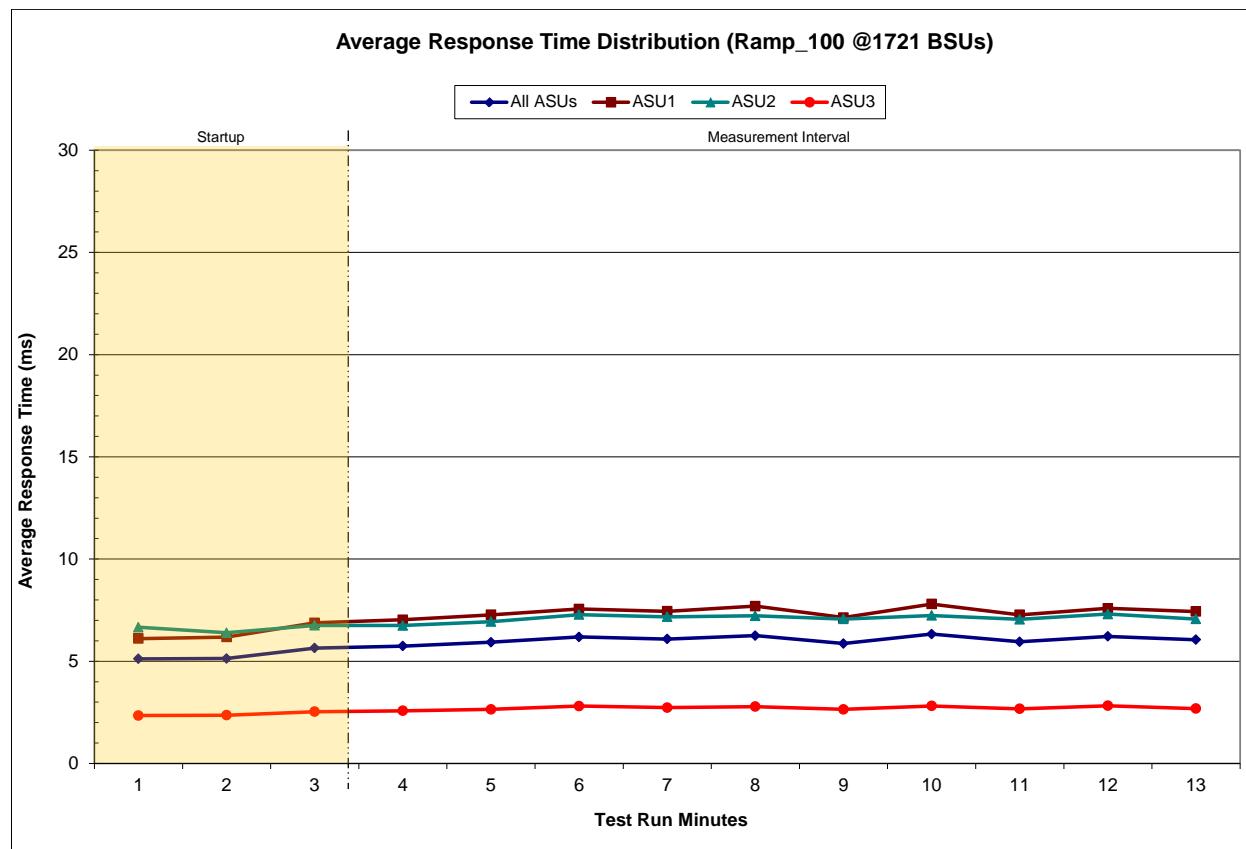
IOPS Test Run – I/O Request Throughput Distribution Graph



IOPS Test Run – Average Response Time (ms) Distribution Data

1,721 BSUs	Start	Stop	Interval	Duration
Start-Up/Ramp-Up	9:41:54	9:44:55	0-2	0:03:01
Measurement Interval	9:44:55	9:54:55	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	5.12	6.11	6.66	2.34
1	5.14	6.19	6.39	2.36
2	5.64	6.88	6.75	2.54
3	5.74	7.03	6.75	2.58
4	5.93	7.27	6.94	2.65
5	6.19	7.56	7.28	2.81
6	6.09	7.45	7.18	2.73
7	6.26	7.69	7.22	2.78
8	5.87	7.14	7.07	2.65
9	6.33	7.80	7.24	2.81
10	5.95	7.27	7.05	2.67
11	6.22	7.59	7.31	2.82
12	6.05	7.43	7.06	2.68
Average	6.06	7.42	7.11	2.72

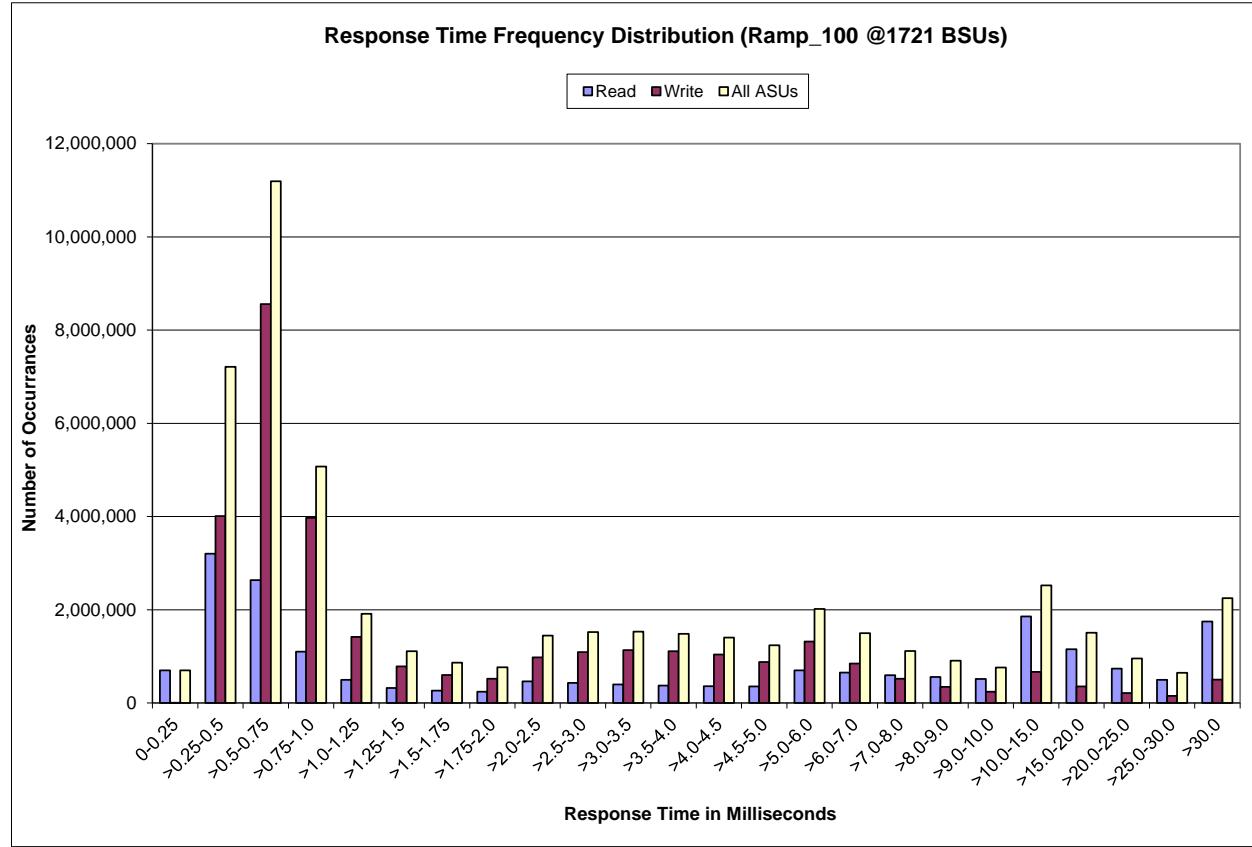
IOPS Test Run – Average Response Time (ms) Distribution Graph



IOPS Test Run –Response Time Frequency Distribution Data

Response Time (ms)	0-0.25	>0.25-0.5	>0.5-0.75	>0.75-1.0	>1.0-1.25	>1.25-1.5	>1.5-1.75	>1.75-2.0
Read	700,782	3,201,654	2,635,886	1,103,225	497,328	322,583	263,829	243,480
Write	986	4,009,106	8,556,286	3,971,845	1,417,375	786,652	603,193	521,626
All ASUs	701,768	7,210,760	11,192,172	5,075,070	1,914,703	1,109,235	867,022	765,106
ASU1	513,946	4,452,384	6,082,139	2,667,091	1,032,865	611,691	480,873	426,593
ASU2	187,491	969,973	1,035,199	520,704	223,214	129,673	100,930	90,996
ASU3	331	1,788,403	4,074,834	1,887,275	658,624	367,871	285,219	247,517
Response Time (ms)	>2.0-2.5	>2.5-3.0	>3.0-3.5	>3.5-4.0	>4.0-4.5	>4.5-5.0	>5.0-6.0	>6.0-7.0
Read	466,329	431,448	396,050	374,531	362,032	356,401	699,983	652,420
Write	980,945	1,092,067	1,133,287	1,111,285	1,042,397	880,941	1,318,373	846,289
All ASUs	1,447,274	1,523,515	1,529,337	1,485,816	1,404,429	1,237,342	2,018,356	1,498,709
ASU1	802,804	815,611	802,595	778,038	738,614	666,336	1,134,135	897,872
ASU2	177,680	183,137	179,176	169,430	154,550	134,906	225,419	183,784
ASU3	466,790	524,767	547,566	538,348	511,265	436,100	658,802	417,053
Response Time (ms)	>7.0-8.0	>8.0-9.0	>9.0-10.0	>10.0-15.0	>15.0-20.0	>20.0-25.0	>25.0-30.0	>30.0
Read	595,657	560,285	515,800	1,858,116	1,152,554	737,460	498,043	1,746,446
Write	519,086	346,701	244,056	665,567	353,492	215,705	150,582	501,700
All ASUs	1,114,743	906,986	759,856	2,523,683	1,506,046	953,165	648,625	2,248,146
ASU1	720,548	620,955	540,661	1,850,017	1,109,681	701,292	481,030	1,846,772
ASU2	150,596	132,041	116,451	413,194	259,953	169,344	114,278	334,096
ASU3	243,599	153,990	102,744	260,472	136,412	82,529	53,317	67,278

IOPS Test Run –Response Time Frequency Distribution Graph



IOPS Test Run – I/O Request Information

I/O Requests Completed in the Measurement Interval	I/O Requests Completed with Response Time = or < 30 ms	I/O Requests Completed with Response Time > 30 ms
51,641,864	49,393,718	2,248,146

IOPS Test Run – Measured Intensity Multiplier and Coefficient of Variation

Clause 3.4.3

IM – Intensity Multiplier: The ratio of I/Os for each I/O stream relative to the total I/Os for all I/O streams (ASU1-1 – ASU3-1) as required by the benchmark specification.

Clauses 5.1.10 and 5.3.15.2

MIM – Measured Intensity Multiplier: The Measured Intensity Multiplier represents the ratio of measured I/Os for each I/O stream relative to the total I/Os measured for all I/O streams (ASU1-1 – ASU3-1). This value may differ from the corresponding Expected Intensity Multiplier by no more than 5%.

Clause 5.3.15.3

COV – Coefficient of Variation: This measure of variation for the Measured Intensity Multiplier cannot exceed 0.2.

	ASU1-1	ASU1-2	ASU1-3	ASU1-4	ASU2-1	ASU2-2	ASU2-3	ASU3-1
IM	0.0350	0.2810	0.0700	0.2100	0.0180	0.0700	0.0350	0.2810
MIM	0.0350	0.2810	0.0700	0.2099	0.0180	0.0700	0.0350	0.2810
COV	0.003	0.001	0.002	0.001	0.002	0.001	0.003	0.000

Primary Metrics Test – Response Time Ramp Test Phase

Clause 5.4.4.3

The Response Time Ramp Test Phase consists of five Test Runs, one each at 95%, 90%, 80%, 50%, and 10% of the load point (100%) used to generate the SPC-1 IOPS™ primary metric. Each of the five Test Runs has a Measurement Interval of ten (10) minutes. The Response Time Ramp Test Phase immediately follows the IOPS Test Phase without any interruption or manual intervention.

The five Response Time Ramp Test Runs, in conjunction with the IOPS Test Run (100%), demonstrate the relationship between Average Response Time and I/O Request Throughput for the Tested Storage Configuration (TSC) as illustrated in the response time/throughput curve on page 16.

In addition, the Average Response Time measured during the 10% Test Run is the value for the SPC-1 LRT™ metric. That value represents the Average Response Time of a lightly loaded TSC.

Clause 9.4.3.7.4

The following content shall appear in the FDR for the Response Time Ramp Phase:

1. *A Response Time Ramp Distribution.*
2. *The human readable Test Run Results File produced by the Workload Generator for each Test Run within the Response Time Ramp Test Phase.*
3. *For the 10% Load Level Test Run (SPC-1 LRT™ metric) an Average Response Time Distribution.*
4. *A listing or screen image of all input parameters supplied to the Workload Generator.*

SPC-1 Workload Generator Input Parameters

The SPC-1 Workload Generator input parameters for the Sustainability, IOPS, Response Time Ramp, Repeatability, and Persistence Test Runs are documented in [Appendix E: SPC-1 Workload Generator Input Parameters](#) on Page [102](#).

Response Time Ramp Test Results File

A link to each test result file generated from each Response Time Ramp Test Run listed below.

[95% Load Level](#)

[90% Load Level](#)

[80% Load Level](#)

[50% Load Level](#)

[10% Load Level](#)

Response Time Ramp Distribution (IOPS) Data

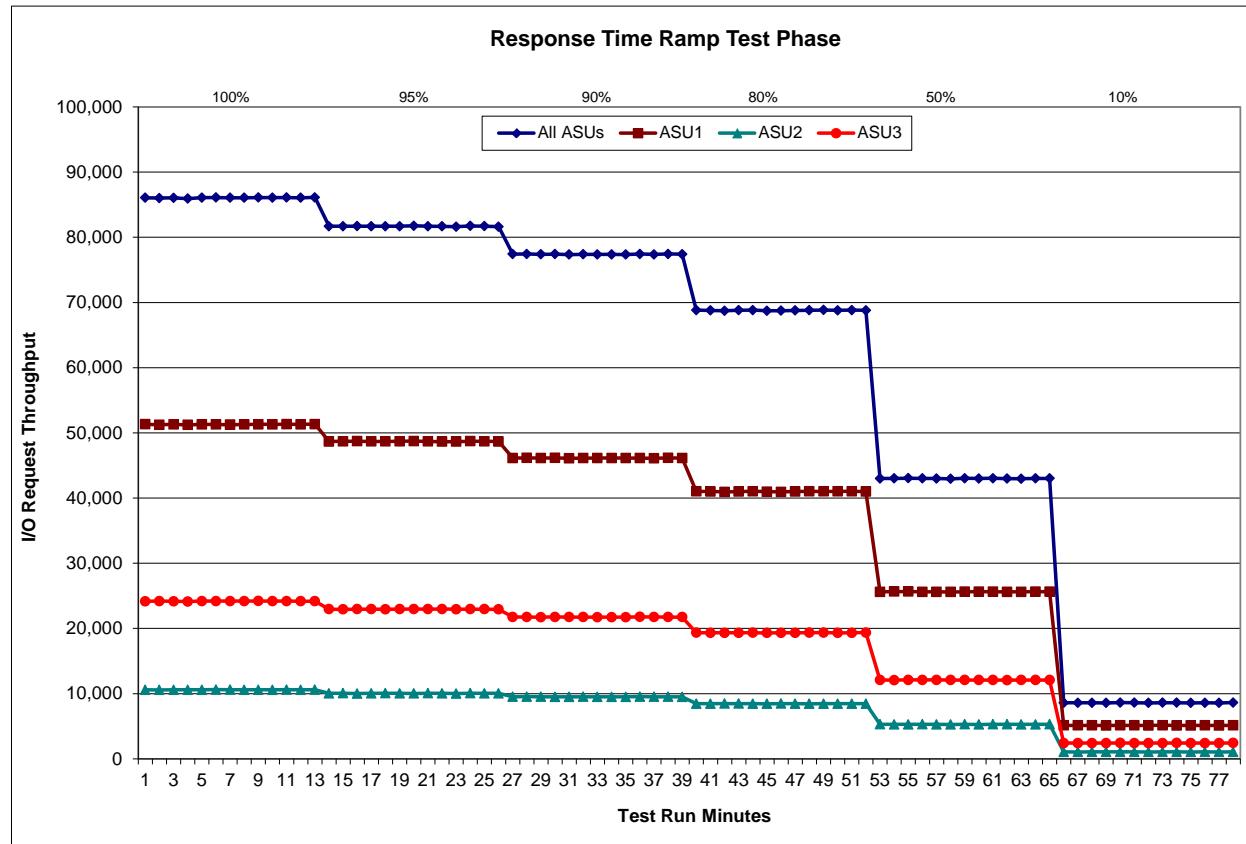
The five Test Runs that comprise the Response Time Ramp Phase are executed at 95%, 90%, 80%, 50%, and 10% of the Business Scaling Unit (BSU) load level used to produce the SPC-1 IOPSTM primary metric. The 100% BSU load level is included in the following Response Time Ramp data table and graph for completeness.

100% Load Level: 1,721 BSUs				95% Load Level: 1,643 BSUs				80% Load Level: 1,376 BSUs				
Start-Up/Ramp-Up		Measurement Interval		Start-Up/Ramp-Up		Measurement Interval		Start-Up/Ramp-Up		Measurement Interval		
9:41:54		9:44:55		9:55:28		9:58:29		10:22:33		10:25:34		
(60 second intervals)		All ASUs		ASU-1		ASU-2		ASU-1		ASU-2		
0	86,067.88	51,323.08	10,578.83	24,165.97				0	81,686.88	48,672.68	10,052.70	22,961.50
1	86,012.90	51,245.35	10,574.08	24,193.47				1	81,687.27	48,705.13	10,055.68	22,926.45
2	86,051.57	51,312.98	10,579.15	24,159.43				2	81,704.70	48,726.78	10,017.45	22,960.47
3	85,938.18	51,224.80	10,584.62	24,128.77				3	81,699.35	48,693.02	10,048.83	22,957.50
4	86,089.25	51,311.25	10,585.05	24,192.95				4	81,689.08	48,690.53	10,054.43	22,944.12
5	86,091.70	51,283.85	10,611.05	24,196.80				5	81,700.92	48,698.03	10,040.10	22,962.78
6	86,060.80	51,255.95	10,608.10	24,196.75				6	81,756.12	48,745.78	10,028.50	22,981.83
7	86,073.70	51,292.17	10,589.67	24,191.87				7	81,700.12	48,686.83	10,057.47	22,955.82
8	86,093.35	51,305.93	10,581.08	24,206.33				8	81,676.18	48,649.77	10,053.25	22,973.17
9	86,085.20	51,308.35	10,585.95	24,190.90				9	81,628.85	48,662.60	10,019.08	22,947.17
10	86,104.72	51,314.70	10,602.17	24,187.85				10	81,744.93	48,716.22	10,064.43	22,964.28
11	86,063.78	51,297.27	10,586.83	24,179.68				11	81,708.87	48,705.10	10,045.18	22,958.58
12	86,122.58	51,334.07	10,605.15	24,183.37				12	81,630.72	48,669.32	10,040.85	22,920.55
Average	86,072.33	51,292.83	10,593.97	24,185.53				Average	81,693.51	48,691.72	10,045.21	22,956.58
90% Load Level: 1,548 BSUs				80% Load Level: 1,376 BSUs				Start-Up/Ramp-Up				
Start-Up/Ramp-Up		Measurement Interval		Start-Up/Ramp-Up		Measurement Interval		Start-Up/Ramp-Up		Measurement Interval		
10:09:01		10:12:02		0-3		0:03:01		10:22:33		0-3		
10:12:02		10:22:02		3-12		0:10:00		10:25:34		3-12		
(60 second intervals)		All ASUs		ASU-1		ASU-2		ASU-1		ASU-2		
0	77,430.42	46,142.37	9,522.12	21,765.93				0	68,830.58	41,024.42	8,461.73	19,344.43
1	77,432.40	46,153.17	9,531.93	21,747.30				1	68,781.30	40,999.15	8,457.80	19,324.35
2	77,391.03	46,139.82	9,521.75	21,729.47				2	68,733.90	40,944.32	8,466.68	19,322.90
3	77,424.68	46,149.28	9,510.97	21,764.43				3	68,805.25	40,998.10	8,474.07	19,333.08
4	77,352.87	46,103.52	9,506.70	21,742.65				4	68,832.27	41,036.75	8,453.20	19,342.32
5	77,390.15	46,130.45	9,520.68	21,739.02				5	68,734.30	40,965.18	8,448.10	19,321.02
6	77,372.43	46,139.82	9,500.47	21,732.15				6	68,738.08	40,933.55	8,470.62	19,333.92
7	77,363.97	46,140.35	9,499.02	21,724.60				7	68,786.28	41,004.98	8,451.07	19,330.23
8	77,360.27	46,133.37	9,506.38	21,720.52				8	68,807.20	41,024.80	8,440.93	19,341.47
9	77,437.65	46,128.72	9,536.43	21,772.50				9	68,842.15	41,020.23	8,453.82	19,368.10
10	77,372.28	46,098.82	9,518.58	21,754.88				10	68,792.50	41,024.17	8,460.28	19,308.05
11	77,430.55	46,155.33	9,525.47	21,749.75				11	68,833.27	41,034.22	8,465.22	19,333.83
12	77,398.93	46,133.87	9,512.70	21,752.37				12	68,802.88	40,984.82	8,461.70	19,356.37
Average	77,390.38	46,131.35	9,513.74	21,745.29				Average	68,797.42	41,002.68	8,457.90	19,336.84

Response Time Ramp Distribution (IOPS) Data (*continued*)

50% Load Level: 860 BSUs	Start	Stop	Interval	Duration	10% Load Level: 172 BSUs	Start	Stop	Interval	Duration
Start-Up/Ramp-Up Measurement Interval	10:36:04	10:39:05	0-3	0:03:01	Start-Up/Ramp-Up Measurement Interval	10:49:35	10:52:36	0-3	0:03:01
(60 second intervals)	All ASUs	ASU-1	ASU-2	ASU-3	(60 second intervals)	All ASUs	ASU-1	ASU-2	ASU-3
0	42,996.82	25,612.22	5,301.63	12,082.97	0	8,599.77	5,125.17	1,056.40	2,418.20
1	43,013.92	25,657.32	5,286.75	12,069.85	1	8,584.98	5,125.32	1,047.78	2,411.88
2	43,047.45	25,668.95	5,281.80	12,096.70	2	8,590.30	5,125.45	1,056.47	2,408.38
3	43,018.20	25,606.33	5,308.42	12,103.45	3	8,580.40	5,114.60	1,057.70	2,408.10
4	43,004.60	25,608.52	5,285.77	12,110.32	4	8,621.17	5,136.30	1,061.52	2,423.35
5	42,954.57	25,587.75	5,271.18	12,095.63	5	8,601.97	5,126.07	1,058.45	2,417.45
6	43,037.50	25,633.48	5,307.88	12,096.13	6	8,573.93	5,115.72	1,051.97	2,406.25
7	42,997.32	25,642.00	5,273.97	12,081.35	7	8,617.63	5,144.13	1,062.77	2,410.73
8	43,032.12	25,635.50	5,306.13	12,090.48	8	8,605.97	5,117.42	1,058.63	2,429.92
9	42,989.22	25,618.03	5,299.70	12,071.48	9	8,578.60	5,109.03	1,050.45	2,419.12
10	42,971.95	25,604.00	5,283.85	12,084.10	10	8,597.45	5,128.42	1,055.52	2,413.52
11	43,015.93	25,646.15	5,287.22	12,082.57	11	8,581.10	5,116.63	1,052.42	2,412.05
12	43,011.77	25,632.17	5,298.97	12,080.63	12	8,634.70	5,135.42	1,063.43	2,435.85
Average	43,003.32	25,621.39	5,292.31	12,089.62	Average	8,599.29	5,124.37	1,057.29	2,417.63

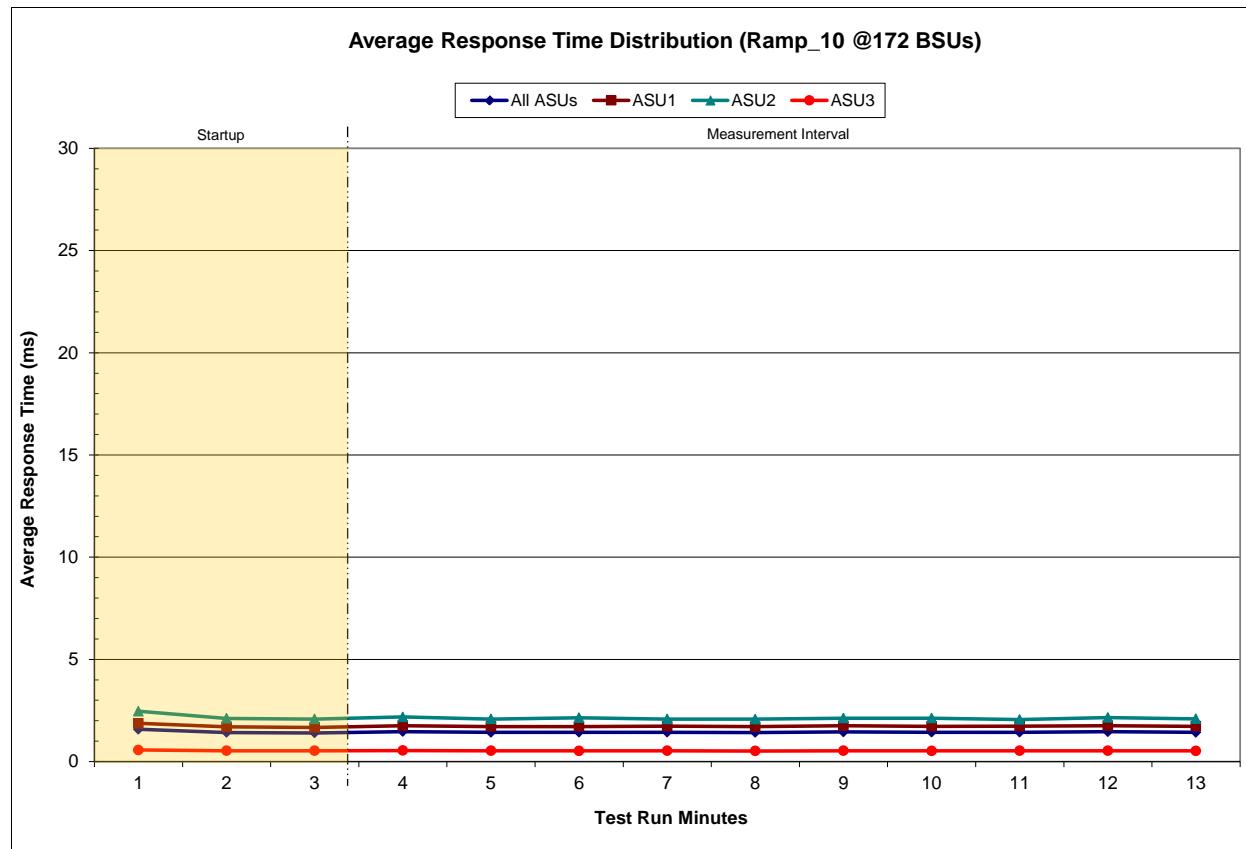
Response Time Ramp Distribution (IOPS) Graph



SPC-1 LRT™ Average Response Time (ms) Distribution Data

172 BSUs	Start	Stop	Interval	Duration
Start-Up/Ramp-Up	10:49:35	10:52:36	0-2	0:03:01
Measurement Interval	10:52:36	11:02:36	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	1.58	1.87	2.46	0.57
1	1.42	1.70	2.11	0.53
2	1.40	1.67	2.08	0.53
3	1.47	1.76	2.18	0.54
4	1.43	1.72	2.08	0.53
5	1.43	1.72	2.14	0.53
6	1.44	1.73	2.08	0.53
7	1.42	1.71	2.08	0.52
8	1.45	1.75	2.13	0.53
9	1.44	1.73	2.12	0.53
10	1.44	1.73	2.06	0.53
11	1.46	1.76	2.16	0.53
12	1.43	1.73	2.09	0.52
Average	1.44	1.73	2.11	0.53

SPC-1 LRT™ Average Response Time (ms) Distribution Graph



SPC-1 LRT™ (10%) – Measured Intensity Multiplier and Coefficient of Variation

Clause 3.4.3

IM – Intensity Multiplier: The ratio of I/Os for each I/O stream relative to the total I/Os for all I/O streams (ASU1-1 – ASU3-1) as required by the benchmark specification.

Clauses 5.1.10 and 5.3.15.2

MIM – Measured Intensity Multiplier: The Measured Intensity Multiplier represents the ratio of measured I/Os for each I/O stream relative to the total I/Os measured for all I/O streams (ASU1-1 – ASU3-1). This value may differ from the corresponding Expected Intensity Multiplier by no more than 5%.

Clause 5.3.15.3

COV – Coefficient of Variation: This measure of variation for the Measured Intensity Multiplier cannot exceed 0.2.

	ASU1-1	ASU1-2	ASU1-3	ASU1-4	ASU2-1	ASU2-2	ASU2-3	ASU3-1
IM	0.0350	0.2810	0.0700	0.2100	0.0180	0.0700	0.0350	0.2810
MIM	0.0350	0.2810	0.0699	0.2100	0.0180	0.0699	0.0350	0.2811
COV	0.007	0.003	0.005	0.002	0.009	0.003	0.006	0.003

Repeatability Test

Clause 5.4.5

The Repeatability Test demonstrates the repeatability and reproducibility of the SPC-1 IOPS™ primary metric and the SPC-1 LRT™ metric generated in earlier Test Runs.

There are two identical Repeatability Test Phases. Each Test Phase contains two Test Runs. Each of the Test Runs will have a Measurement Interval of no less than ten (10) minutes. The two Test Runs in each Test Phase will be executed without interruption or any type of manual intervention.

The first Test Run in each Test Phase is executed at the 10% load point. The Average Response Time from each of the Test Runs is compared to the SPC-1 LRT™ metric. Each Average Response Time value must be less than the SPC-1 LRT™ metric plus 5% or less than the SPC-1 LRT™ metric plus one (1) millisecond (ms).

The second Test Run in each Test Phase is executed at the 100% load point. The I/O Request Throughput from the Test Runs is compared to the SPC-1 IOPS™ primary metric. Each I/O Request Throughput value must be greater than the SPC-1 IOPS™ primary metric minus 5%. In addition, the Average Response Time for each Test Run cannot exceed 30 milliseconds.

If any of the above constraints are not met, the benchmark measurement is invalid.

Clause 9.4.3.7.5

The following content shall appear in the FDR for each Test Run in the two Repeatability Test Phases:

1. A table containing the results of the Repeatability Test.
2. An I/O Request Throughput Distribution graph and table.
3. An Average Response Time Distribution graph and table.
4. The human readable Test Run Results File produced by the Workload Generator.
5. A listing or screen image of all input parameters supplied to the Workload Generator.

SPC-1 Workload Generator Input Parameters

The SPC-1 Workload Generator input parameters for the Sustainability, IOPS, Response Time Ramp, Repeatability, and Persistence Test Runs are documented in [Appendix E: SPC-1 Workload Generator Input Parameters](#) on Page [102](#).

Repeatability Test Results File

The values for the SPC-1 IOPS™, SPC-1 LRT™, and the Repeatability Test measurements are listed in the tables below.

	SPC-1 IOPS™
Primary Metrics	86,072.33
Repeatability Test Phase 1	86,044.95
Repeatability Test Phase 2	86,058.99

The SPC-1 IOPS™ values in the above table were generated using 100% of the specified Business Scaling Unit (BSU) load level. Each of the Repeatability Test Phase values for SPC-1 IOPS™ must greater than 95% of the reported SPC-1 IOPS™ Primary Metric.

	SPC-1 LRT™
Primary Metrics	1.44 ms
Repeatability Test Phase 1	1.43 ms
Repeatability Test Phase 2	1.43 ms

The average response time values in the SPC-1 LRT™ column were generated using 10% of the specified Business Scaling Unit (BSU) load level. Each of the Repeatability Test Phase values for SPC-1 LRT™ must be less than 105% of the reported SPC-1 LRT™ Primary Metric or less than the reported SPC-1 LRT™ Primary Metric plus one (1) millisecond (ms).

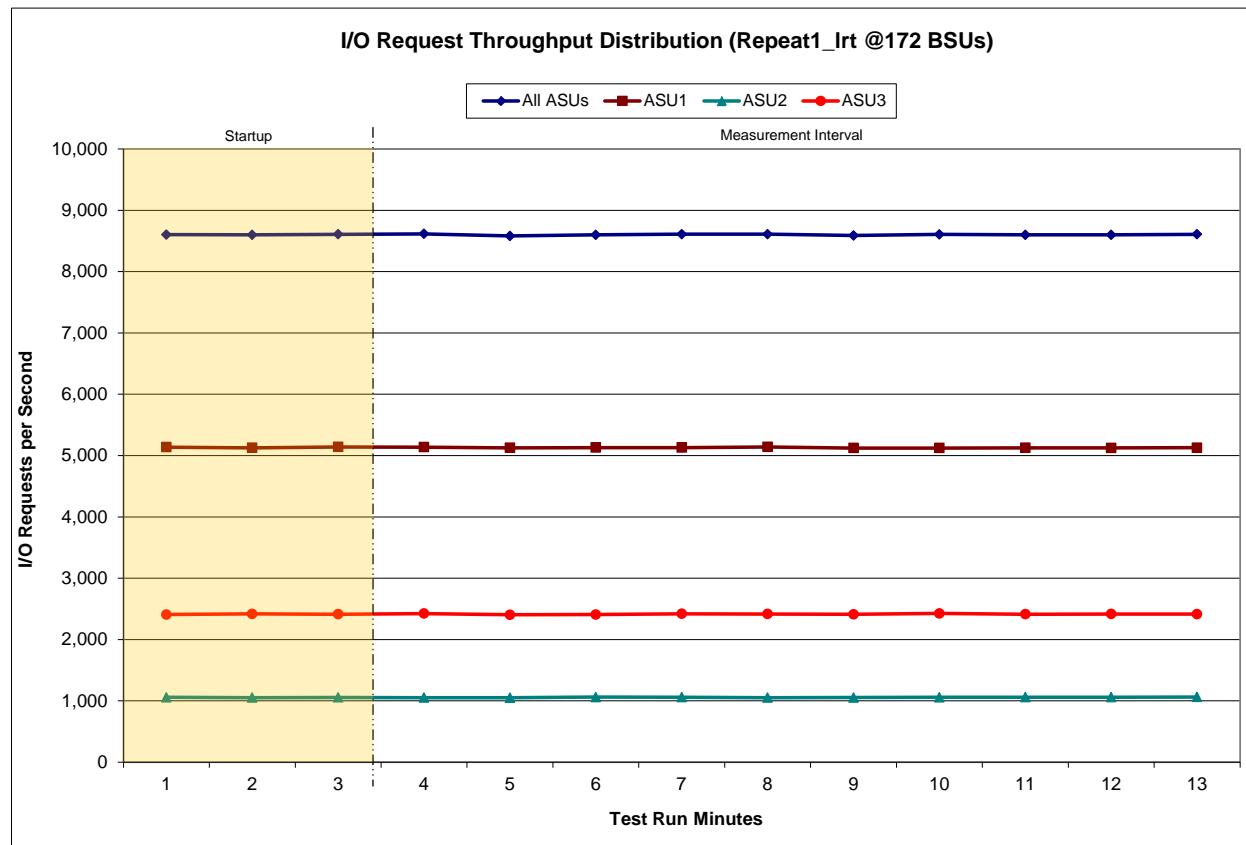
A link to the test result file generated from each Repeatability Test Run is listed below.

- [Repeatability Test Phase 1, Test Run 1 \(LRT\)](#)
- [Repeatability Test Phase 1, Test Run 2 \(IOPS\)](#)
- [Repeatability Test Phase 2, Test Run 1 \(LRT\)](#)
- [Repeatability Test Phase 2, Test Run 2 \(IOPS\)](#)

Repeatability 1 LRT – I/O Request Throughput Distribution Data

172 BSUs	Start	Stop	Interval	Duration
<i>Start-Up/Ramp-Up</i>	11:03:10	11:06:10	0-2	0:03:00
<i>Measurement Interval</i>	11:06:10	11:16:10	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	8,602.95	5,137.18	1,058.55	2,407.22
1	8,598.93	5,127.45	1,053.57	2,417.92
2	8,609.12	5,140.22	1,056.33	2,412.57
3	8,615.18	5,138.07	1,053.17	2,423.95
4	8,581.98	5,126.67	1,050.70	2,404.62
5	8,598.95	5,130.52	1,060.97	2,407.47
6	8,610.50	5,129.97	1,059.53	2,421.00
7	8,609.48	5,140.28	1,053.02	2,416.18
8	8,588.98	5,122.25	1,055.03	2,411.70
9	8,605.92	5,123.60	1,057.25	2,425.07
10	8,599.80	5,126.12	1,060.53	2,413.15
11	8,599.25	5,124.00	1,059.13	2,416.12
12	8,607.68	5,129.10	1,062.97	2,415.62
Average	8,601.77	5,129.06	1,057.23	2,415.49

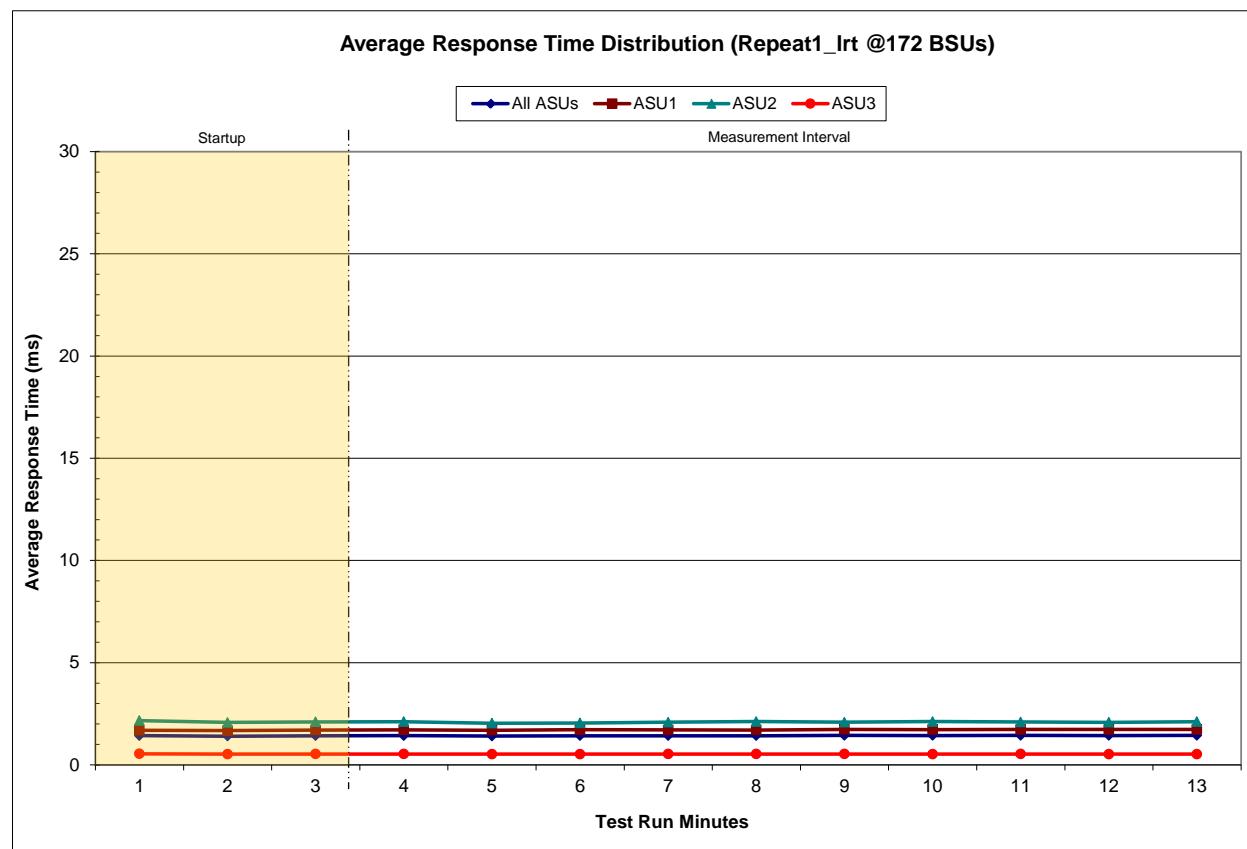
Repeatability 1 LRT – I/O Request Throughput Distribution Graph



Repeatability 1 LRT –Average Response Time (ms) Distribution Data

172 BSUs	Start	Stop	Interval	Duration
Start-Up/Ramp-Up	11:03:10	11:06:10	0-2	0:03:00
Measurement Interval	11:06:10	11:16:10	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	1.43	1.69	2.17	0.54
1	1.40	1.68	2.07	0.52
2	1.42	1.70	2.10	0.53
3	1.43	1.71	2.11	0.53
4	1.41	1.69	2.04	0.53
5	1.43	1.72	2.05	0.53
6	1.43	1.71	2.09	0.53
7	1.43	1.71	2.12	0.53
8	1.44	1.74	2.09	0.53
9	1.44	1.73	2.12	0.53
10	1.44	1.73	2.10	0.53
11	1.44	1.73	2.08	0.52
12	1.44	1.73	2.11	0.53
Average	1.43	1.72	2.09	0.53

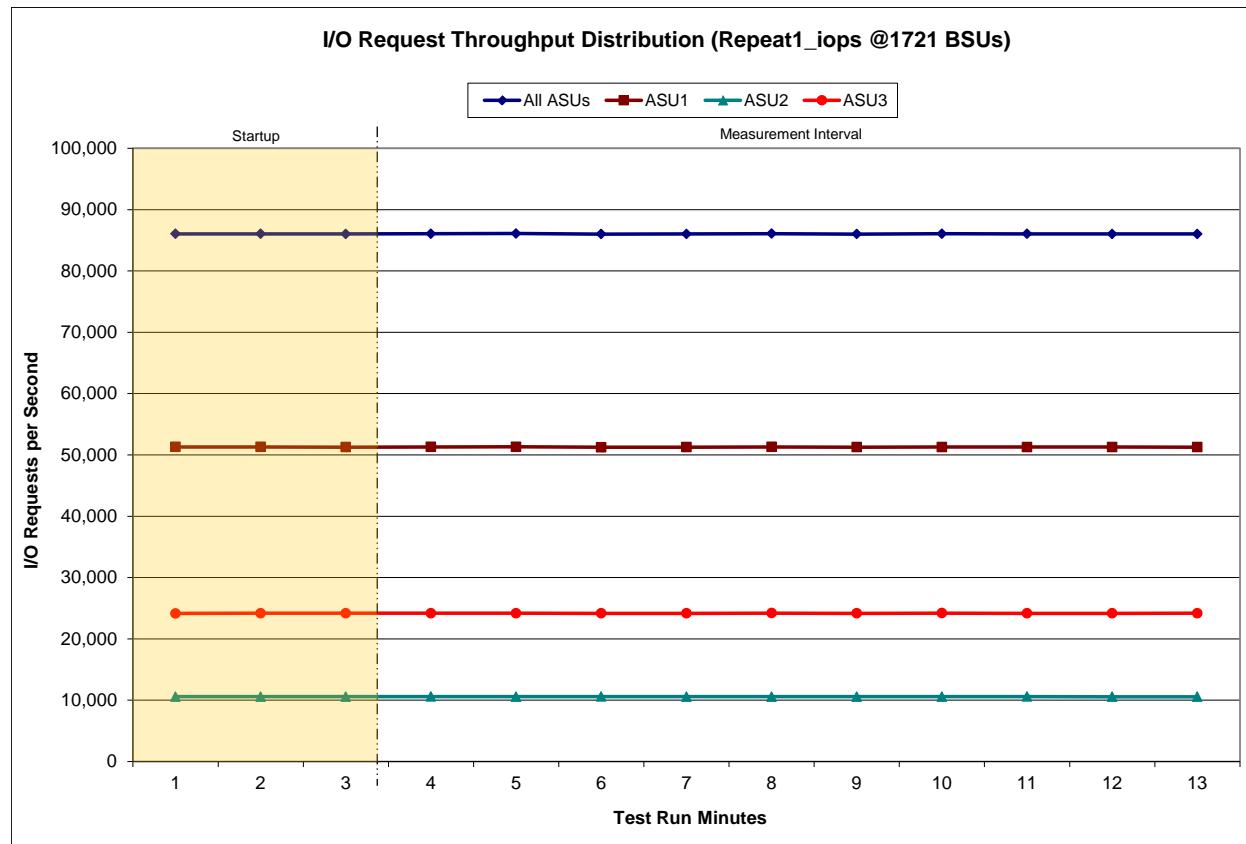
Repeatability 1 LRT –Average Response Time (ms) Distribution Graph



Repeatability 1 IOPS – I/O Request Throughput Distribution Data

1,721 BSUs	Start	Stop	Interval	Duration
Start-Up/Ramp-Up	11:16:43	11:19:44	0-2	0:03:01
Measurement Interval	11:19:44	11:29:44	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	86,050.30	51,306.77	10,588.72	24,154.82
1	86,050.67	51,297.95	10,575.95	24,176.77
2	86,029.73	51,266.68	10,578.23	24,184.82
3	86,072.90	51,301.05	10,595.45	24,176.40
4	86,095.23	51,313.98	10,589.37	24,191.88
5	86,009.37	51,247.95	10,601.35	24,160.07
6	86,020.55	51,269.08	10,584.50	24,166.97
7	86,078.88	51,297.00	10,572.50	24,209.38
8	86,016.28	51,269.10	10,578.25	24,168.93
9	86,065.52	51,283.25	10,582.93	24,199.33
10	86,048.10	51,290.33	10,590.32	24,167.45
11	86,021.05	51,285.15	10,567.33	24,168.57
12	86,021.65	51,272.27	10,569.87	24,179.52
Average	86,044.95	51,282.92	10,583.19	24,178.85

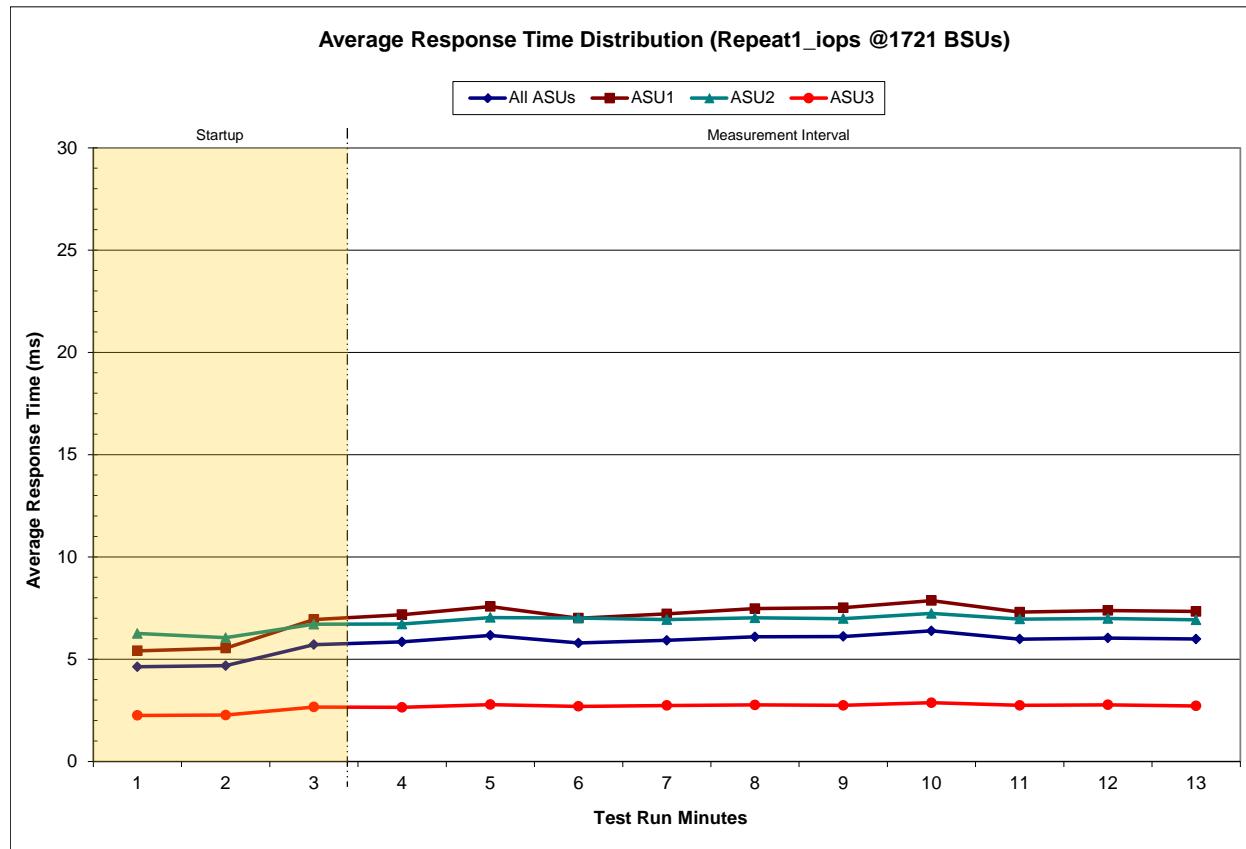
Repeatability 1 IOPS – I/O Request Throughput Distribution Graph



Repeatability 1 IOPS –Average Response Time (ms) Distribution Data

1,721 BSUs	Start	Stop	Interval	Duration
Start-Up/Ramp-Up	11:16:43	11:19:44	0-2	0:03:01
Measurement Interval	11:19:44	11:29:44	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	4.63	5.41	6.26	2.25
1	4.68	5.54	6.05	2.26
2	5.71	6.94	6.71	2.66
3	5.85	7.18	6.72	2.64
4	6.16	7.57	7.04	2.78
5	5.79	7.00	7.01	2.69
6	5.92	7.21	6.93	2.74
7	6.09	7.47	7.02	2.76
8	6.11	7.52	6.98	2.74
9	6.38	7.87	7.23	2.87
10	5.98	7.30	6.96	2.74
11	6.04	7.38	6.99	2.77
12	5.98	7.33	6.92	2.72
Average	6.03	7.38	6.98	2.75

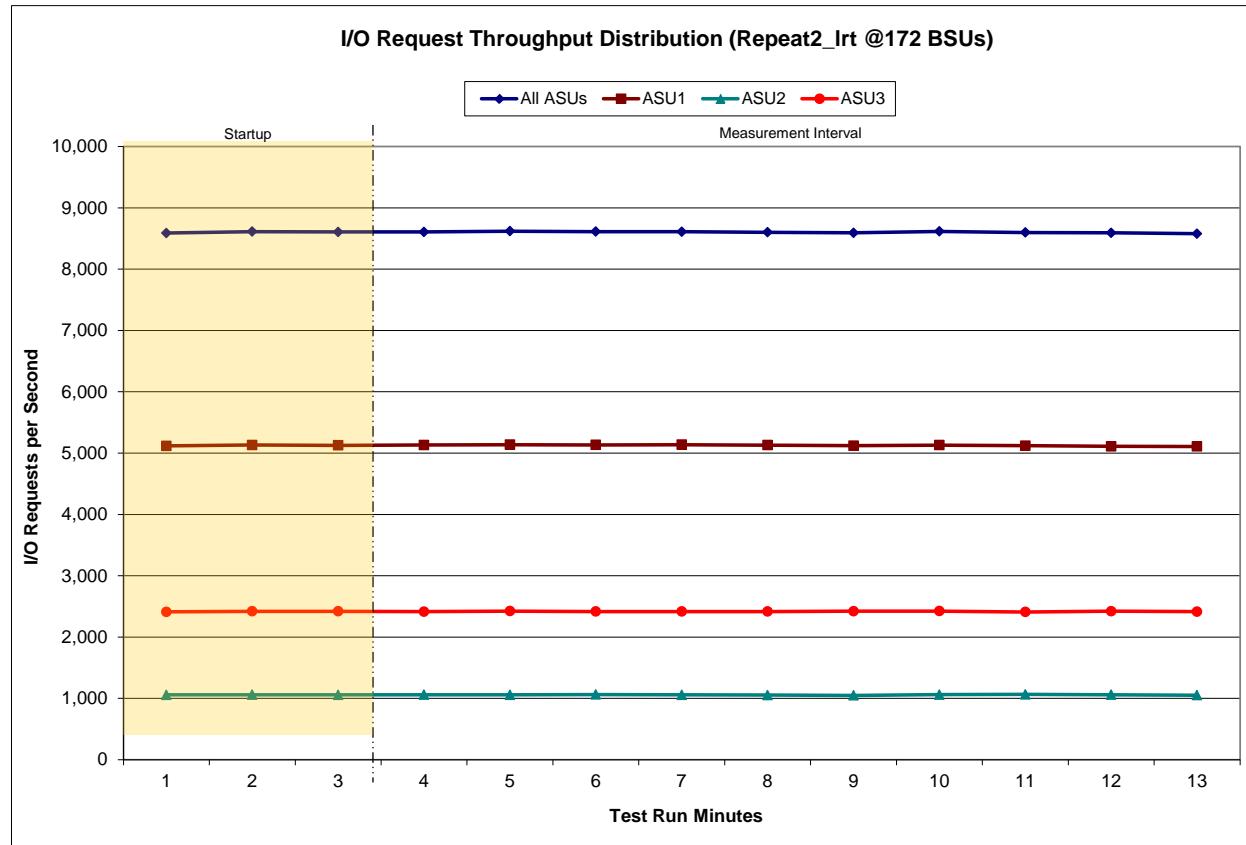
Repeatability 1 IOPS –Average Response Time (ms) Distribution Graph



Repeatability 2 LRT – I/O Request Throughput Distribution Data

172 BSUs	Start	Stop	Interval	Duration
Start-Up/Ramp-Up	11:30:18	11:33:18	0-2	0:03:00
Measurement Interval	11:33:18	11:43:18	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	8,588.32	5,118.43	1,058.77	2,411.12
1	8,612.37	5,132.12	1,059.53	2,420.72
2	8,606.82	5,127.43	1,058.65	2,420.73
3	8,606.03	5,131.65	1,059.05	2,415.33
4	8,619.45	5,137.43	1,058.92	2,423.10
5	8,612.27	5,133.50	1,062.90	2,415.87
6	8,610.82	5,136.97	1,057.57	2,416.28
7	8,601.33	5,130.53	1,054.03	2,416.77
8	8,592.83	5,121.40	1,048.85	2,422.58
9	8,615.92	5,129.82	1,062.38	2,423.72
10	8,596.82	5,121.85	1,064.83	2,410.13
11	8,592.07	5,110.82	1,059.03	2,422.22
12	8,577.13	5,109.52	1,053.28	2,414.33
Average	8,602.47	5,126.35	1,058.09	2,418.03

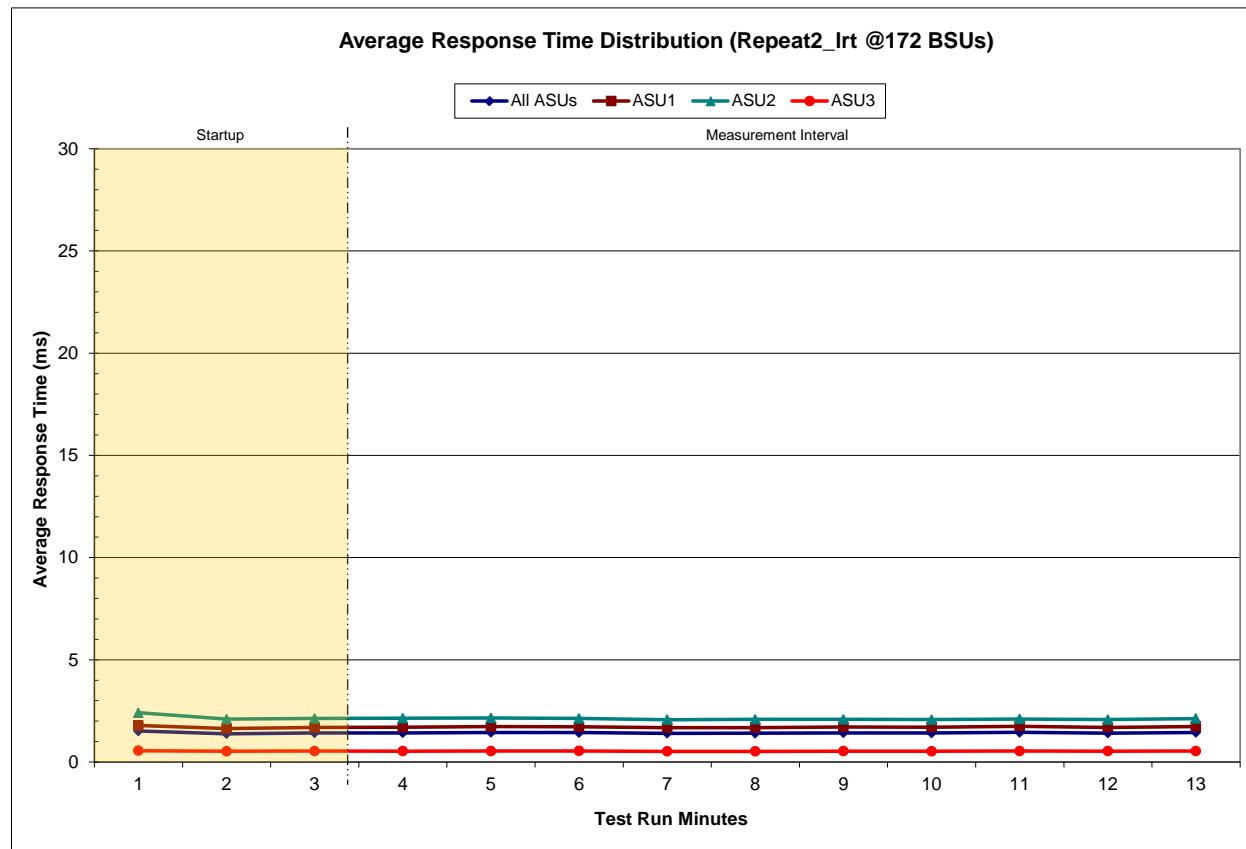
Repeatability 2 LRT – I/O Request Throughput Distribution Graph



Repeatability 2 LRT –Average Response Time (ms) Distribution Data

172 BSUs	Start	Stop	Interval	Duration
Start-Up/Ramp-Up	11:30:18	11:33:18	0-2	0:03:00
Measurement Interval	11:33:18	11:43:18	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	1.52	1.79	2.41	0.55
1	1.38	1.64	2.10	0.52
2	1.42	1.70	2.13	0.54
3	1.42	1.70	2.14	0.53
4	1.45	1.73	2.15	0.54
5	1.44	1.72	2.13	0.54
6	1.40	1.68	2.07	0.52
7	1.41	1.68	2.09	0.52
8	1.43	1.71	2.09	0.53
9	1.42	1.70	2.07	0.52
10	1.45	1.74	2.10	0.53
11	1.41	1.69	2.08	0.53
12	1.45	1.74	2.13	0.53
Average	1.43	1.71	2.10	0.53

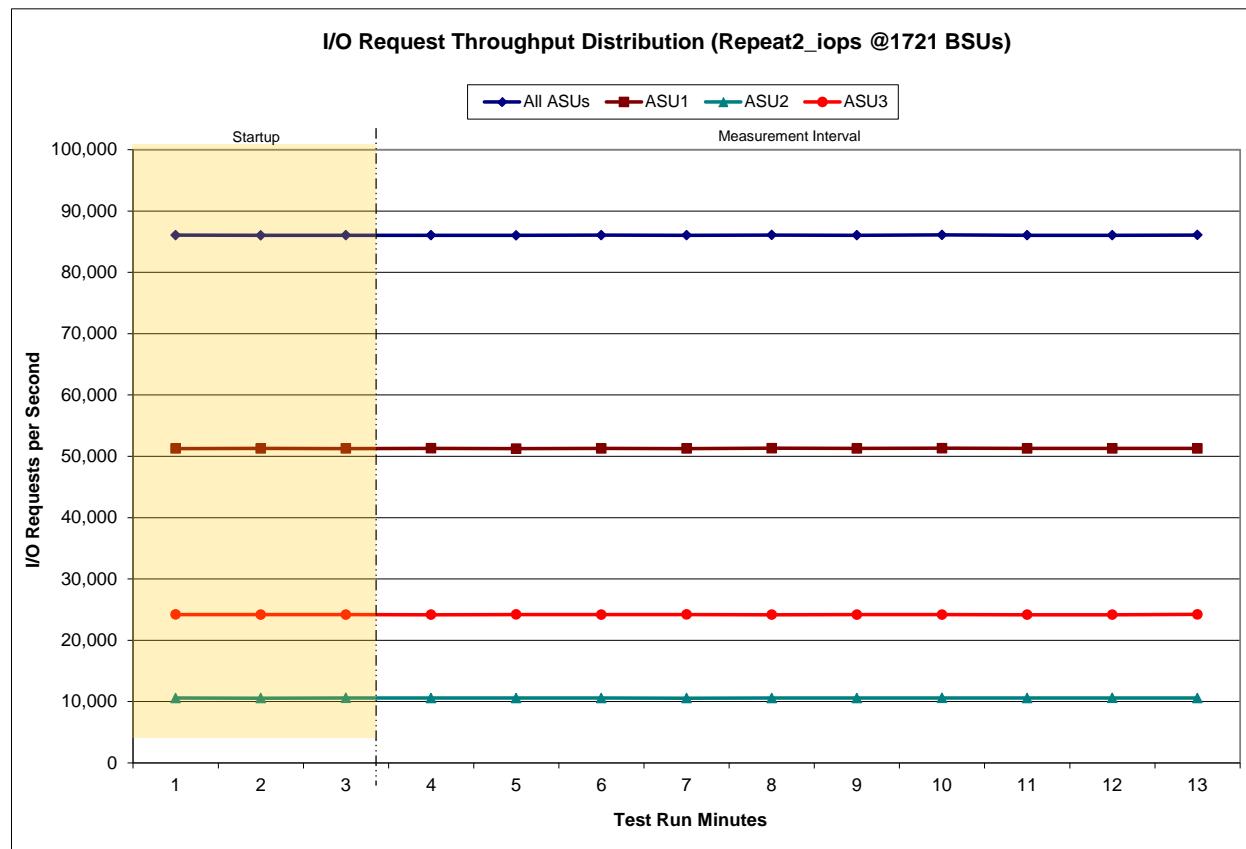
Repeatability 2 LRT –Average Response Time (ms) Distribution Graph



Repeatability 2 IOPS – I/O Request Throughput Distribution Data

1,721 BSUs	Start	Stop	Interval	Duration
Start-Up/Ramp-Up	11:43:51	11:46:52	0-2	0:03:01
Measurement Interval	11:46:52	11:56:52	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	86,056.33	51,267.73	10,587.55	24,201.05
1	86,033.63	51,289.02	10,558.57	24,186.05
2	86,042.20	51,258.27	10,593.13	24,190.80
3	86,056.12	51,308.30	10,586.77	24,161.05
4	86,028.58	51,250.02	10,579.28	24,199.28
5	86,059.27	51,291.35	10,575.05	24,192.87
6	86,045.02	51,267.67	10,567.35	24,210.00
7	86,078.47	51,321.15	10,582.48	24,174.83
8	86,047.33	51,279.15	10,588.40	24,179.78
9	86,098.58	51,312.78	10,598.75	24,187.05
10	86,050.42	51,292.42	10,584.32	24,173.68
11	86,046.07	51,288.38	10,590.22	24,167.47
12	86,080.00	51,283.58	10,580.32	24,216.10
Average	86,058.99	51,289.48	10,583.29	24,186.21

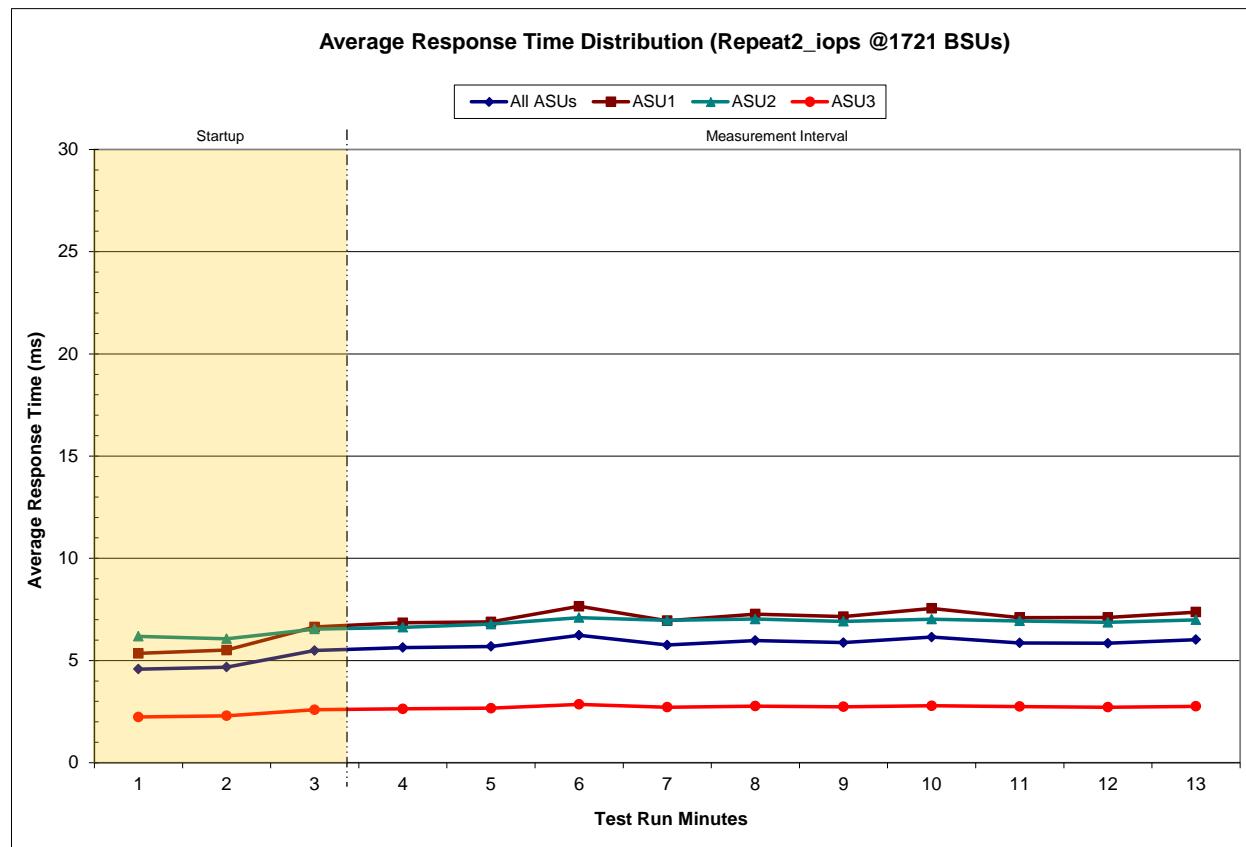
Repeatability 2 IOPS – I/O Request Throughput Distribution Graph



Repeatability 2 IOPS –Average Response Time (ms) Distribution Data

1,721 BSUs	Start	Stop	Interval	Duration
Start-Up/Ramp-Up	11:43:51	11:46:52	0-2	0:03:01
Measurement Interval	11:46:52	11:56:52	3-12	0:10:00
60 second intervals	All ASUs	ASU1	ASU2	ASU3
0	4.58	5.35	6.19	2.24
1	4.67	5.51	6.07	2.30
2	5.49	6.64	6.54	2.59
3	5.64	6.85	6.62	2.64
4	5.69	6.89	6.78	2.67
5	6.24	7.66	7.10	2.86
6	5.76	6.95	6.97	2.72
7	5.98	7.27	7.03	2.77
8	5.88	7.15	6.92	2.74
9	6.15	7.55	7.02	2.79
10	5.86	7.10	6.94	2.75
11	5.84	7.11	6.87	2.71
12	6.02	7.37	6.99	2.76
Average	5.91	7.19	6.92	2.74

Repeatability 2 IOPS –Average Response Time (ms) Distribution Graph



Repeatability 1 (LRT)

Measured Intensity Multiplier and Coefficient of Variation

Clause 3.4.3

IM – Intensity Multiplier: The ratio of I/Os for each I/O stream relative to the total I/Os for all I/O streams (ASU1-1 – ASU3-1) as required by the benchmark specification.

Clauses 5.1.10 and 5.3.15.2

MIM – Measured Intensity Multiplier: The Measured Intensity Multiplier represents the ratio of measured I/Os for each I/O stream relative to the total I/Os measured for all I/O streams (ASU1-1 – ASU3-1). This value may differ from the corresponding Expected Intensity Multiplier by no more than 5%.

Clause 5.3.15.3

COV – Coefficient of Variation: This measure of variation for the Measured Intensity Multiplier cannot exceed 0.2.

	ASU1-1	ASU1-2	ASU1-3	ASU1-4	ASU2-1	ASU2-2	ASU2-3	ASU3-1
IM	0.0350	0.2810	0.0700	0.2100	0.0180	0.0700	0.0350	0.2810
MIM	0.0350	0.2813	0.0701	0.2099	0.0180	0.0700	0.0349	0.2808
COV	0.007	0.002	0.007	0.002	0.012	0.005	0.006	0.002

Repeatability 1 (IOPS)

Measured Intensity Multiplier and Coefficient of Variation

	ASU1-1	ASU1-2	ASU1-3	ASU1-4	ASU2-1	ASU2-2	ASU2-3	ASU3-1
IM	0.0350	0.2810	0.0700	0.2100	0.0180	0.0700	0.0350	0.2810
MIM	0.0350	0.2809	0.0700	0.2101	0.0180	0.0700	0.0350	0.2810
COV	0.002	0.000	0.002	0.001	0.003	0.002	0.003	0.000

Repeatability 2 (LRT)

Measured Intensity Multiplier and Coefficient of Variation

	ASU1-1	ASU1-2	ASU1-3	ASU1-4	ASU2-1	ASU2-2	ASU2-3	ASU3-1
IM	0.0350	0.2810	0.0700	0.2100	0.0180	0.0700	0.0350	0.2810
MIM	0.0350	0.2807	0.0700	0.2102	0.0180	0.0700	0.0350	0.2811
COV	0.008	0.002	0.002	0.003	0.013	0.004	0.006	0.002

Repeatability 2 (IOPS)
Measured Intensity Multiplier and Coefficient of Variation

	ASU1-1	ASU1-2	ASU1-3	ASU1-4	ASU2-1	ASU2-2	ASU2-3	ASU3-1
<i>IM</i>	0.0350	0.2810	0.0700	0.2100	0.0180	0.0700	0.0350	0.2810
MIM	0.0350	0.2809	0.0700	0.2101	0.0180	0.0700	0.0350	0.2810
COV	0.002	0.001	0.001	0.001	0.004	0.001	0.003	0.001

Data Persistence Test

Clause 6

The Data Persistence Test demonstrates the Tested Storage Configuration (TSC):

- Is capable of maintaining data integrity across a power cycle.
- Ensures the transfer of data between Logical Volumes and host systems occurs without corruption or loss.

The SPC-1 Workload Generator will write 16 block I/O requests at random over the total Addressable Storage Capacity of the TSC for ten (10) minutes at a minimum of 25% of the load used to generate the SPC-1 IOPSTM primary metric. The bit pattern selected to be written to each block as well as the address of the block will be retained in a log file.

The Tested Storage Configuration (TSC) will be shutdown and restarted using a power off/power on cycle at the end of the above sequence of write operations. In addition, any caches employing battery backup must be flushed/emptied.

The SPC-1 Workload Generator will then use the above log file to verify each block written contains the correct bit pattern.

Clause 9.4.3.8

The following content shall appear in this section of the FDR:

1. A listing or screen image of all input parameters supplied to the Workload Generator.
2. For the successful Data Persistence Test Run, a table illustrating key results. The content, appearance, and format of this table are specified in Table 9-12. Information displayed in this table shall be obtained from the Test Run Results File referenced below in #3.
3. For the successful Data Persistence Test Run, the human readable Test Run Results file produced by the Workload Generator (may be contained in an appendix).

SPC-1 Workload Generator Input Parameters

The SPC-1 Workload Generator input parameters for the Sustainability, IOPS, Response Time Ramp, Repeatability, and Persistence Test Runs are documented in [Appendix E: SPC-1 Workload Generator Input Parameters](#) on Page [102](#).

Data Persistence Test Results File

A link to each test result file generated from each Data Persistence Test is listed below.

[Persistence 1 Test Results File](#)

[Persistence 2 Test Results File](#)

Data Persistence Test Results

Data Persistence Test Results	
Data Persistence Test Run Number: 1	
Total Number of Logical Blocks Written	206,501,008
Total Number of Logical Blocks Verified	141,130,032
Total Number of Logical Blocks that Failed Verification	0
Time Duration for Writing Test Logical Blocks	10 minutes
Size in bytes of each Logical Block	512
Number of Failed I/O Requests in the process of the Test	0

In some cases the same address was the target of multiple writes, which resulted in more Logical Blocks Written than Logical Blocks Verified. In the case of multiple writes to the same address, the pattern written and verified must be associated with the last write to that address.

PRICED STORAGE CONFIGURATION AVAILABILITY DATE

Clause 9.4.3.9

The committed delivery date for general availability (Availability Date) of all products that comprise the Priced Storage Configuration must be reported. When the Priced Storage Configuration includes products or components with different availability dates, the reported Availability Date for the Priced Storage Configuration must be the date at which all components are committed to be available.

The NetApp FAS8040 as documented in this Full Disclosure Report is currently available for customer purchase and shipment.

PRICING INFORMATION

Clause 9.4.3.3.6

The Executive Summary shall contain a pricing spreadsheet as documented in Clause 8.3.1.

Pricing information may be found in the Priced Storage Configuration Pricing section on page 17.

TESTED STORAGE CONFIGURATION (TSC) AND PRICED STORAGE CONFIGURATION DIFFERENCES

Clause 9.4.3.3.8

The Executive Summary shall contain a list of all differences between the Tested Storage Configuration (TSC) and the Priced Storage Configuration.

A list of all differences between the Tested Storage Configuration (TSC) and Priced Storage Configuration may be found in the Executive Summary portion of this document on page 17.

ANOMALIES OR IRREGULARITIES

Clause 9.4.3.10

The FDR shall include a clear and complete description of any anomalies or irregularities encountered in the course of executing the SPC-1 benchmark that may in any way call into question the accuracy, verifiability, or authenticity of information published in this FDR.

There were no anomalies or irregularities encountered during the SPC-1 Remote Audit of the NetApp FAS8040.

APPENDIX A: SPC-1 GLOSSARY

“Decimal” (*powers of ten*) Measurement Units

In the storage industry, the terms “kilo”, “mega”, “giga”, “tera”, “peta”, and “exa” are commonly used prefixes for computing performance and capacity. For the purposes of the SPC workload definitions, all of the following terms are defined in “powers of ten” measurement units.

A kilobyte (KB) is equal to 1,000 (10^3) bytes.

A megabyte (MB) is equal to 1,000,000 (10^6) bytes.

A gigabyte (GB) is equal to 1,000,000,000 (10^9) bytes.

A terabyte (TB) is equal to 1,000,000,000,000 (10^{12}) bytes.

A petabyte (PB) is equal to 1,000,000,000,000,000 (10^{15}) bytes

An exabyte (EB) is equal to 1,000,000,000,000,000,000 (10^{18}) bytes

“Binary” (*powers of two*) Measurement Units

The sizes reported by many operating system components use “powers of two” measurement units rather than “power of ten” units. The following standardized definitions and terms are also valid and may be used in this document.

A kibibyte (KiB) is equal to 1,024 (2^{10}) bytes.

A mebibyte (MiB) is equal to 1,048,576 (2^{20}) bytes.

A gigabyte (GiB) is equal to 1,073,741,824 (2^{30}) bytes.

A tebibyte (TiB) is equal to 1,099,511,627,776 (2^{40}) bytes.

A pebibyte (PiB) is equal to 1,125,899,906,842,624 (2^{50}) bytes.

An exbibyte (EiB) is equal to 1,152,921,504,606,846,967 (2^{60}) bytes.

SPC-1 Data Repository Definitions

Total ASU Capacity: The total storage capacity read and written in the course of executing the SPC-1 benchmark.

Application Storage Unit (ASU): The logical interface between the storage and SPC-1 Workload Generator. The three ASUs (Data, User, and Log) are typically implemented on one or more Logical Volume.

Logical Volume: The division of Addressable Storage Capacity into individually addressable logical units of storage used in the SPC-1 benchmark. Each Logical Volume is implemented as a single, contiguous address space.

Addressable Storage Capacity: The total storage (sum of Logical Volumes) that can be read and written by application programs such as the SPC-1 Workload Generator.

Configured Storage Capacity: This capacity includes the Addressable Storage Capacity and any other storage (parity disks, hot spares, etc.) necessary to implement the Addressable Storage Capacity.

Physical Storage Capacity: The formatted capacity of all storage devices physically present in the Tested Storage Configuration (TSC).

Data Protection Overhead: The storage capacity required to implement the selected level of data protection.

Required Storage: The amount of Configured Storage Capacity required to implement the Addressable Storage Configuration, excluding the storage required for the three ASUs.

Global Storage Overhead: The amount of Physical Storage Capacity that is required for storage subsystem use and unavailable for use by application programs.

Total Unused Storage: The amount of storage capacity available for use by application programs but not included in the Total ASU Capacity.

SPC-1 Data Protection Levels

Protected 1: The single point of failure of any *storage device* in the configuration will not result in permanent loss of access to or integrity of the SPC-1 Data Repository.

Protected 2: The single point of failure of any *component* in the configuration will not result in permanent loss of access to or integrity of the SPC-1 Data Repository.

SPC-1 Test Execution Definitions

Average Response Time: The sum of the Response Times for all Measured I/O Requests divided by the total number of Measured I/O Requests.

Completed I/O Request: An I/O Request with a Start Time and a Completion Time (see “I/O Completion Types” below).

Completion Time: The time recorded by the Workload Generator when an I/O Request is satisfied by the TSC as signaled by System Software.

Data Rate: The data transferred in all Measured I/O Requests in an SPC-1 Test Run divided by the length of the Test Run in seconds.

Expected I/O Count: For any given I/O Stream and Test Phase, the product of 50 times the BSU level, the duration of the Test Phase in seconds, and the Intensity Multiplier for that I/O Stream.

Failed I/O Request: Any I/O Request issued by the Workload Generator that could not be completed or was signaled as failed by System Software. A Failed I/O Request has no Completion Time (see “I/O Completion Types” below).

I/O Request Throughput: The total number of Measured I/O requests in an SPC-1 Test Run divided by the duration of the Measurement Interval in seconds.

In-Flight I/O Request: An I/O Request issued by the I/O Command Generator to the TSC that has a recorded Start Time, but does not complete within the Measurement Interval (see “I/O Completion Types” below).

Measured I/O Request: A Completed I/O Request with a Completion Time occurring within the Measurement Interval (see “I/O Completion Types” below).

Measured Intensity Multiplier: The percentage of all Measured I/O Requests that were issued by a given I/O Stream.

Measurement Interval: The finite and contiguous time period, after the TSC has reached Steady State, when data is collected by a Test Sponsor to generate an SPC-1 test result or support an SPC-1 test result.

Ramp-Up: The time required for the Benchmark Configuration (BC) to produce Steady State throughput after the Workload Generator begins submitting I/O Requests to the TSC for execution.

Ramp-Down: The time required for the BC to complete all I/O Requests issued by the Workload Generator. The Ramp-Down period begins when the Workload Generator ceases to issue new I/O Requests to the TSC.

Response Time: The Response Time of a Measured I/O Request is its Completion Time minus its Start Time.

Start Time: The time recorded by the Workload Generator when an I/O Request is submitted, by the Workload Generator, to the System Software for execution on the Tested Storage Configuration (TSC).

Start-Up: The period that begins after the Workload Generator starts to submit I/O requests to the TSC and ends at the beginning of the Measurement Interval.

Shut-Down: The period between the end of the Measurement Interval and the time when all I/O Requests issued by the Workload Generator have completed or failed.

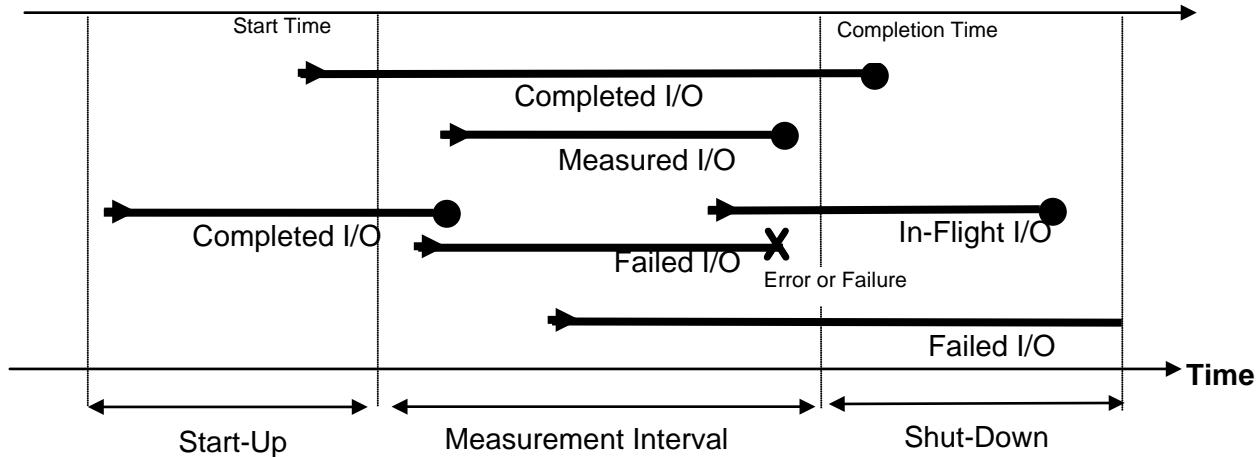
Steady State: The consistent and sustainable throughput of the TSC. During this period the load presented to the TSC by the Workload Generator is constant.

Test: A collection of Test Phases and or Test Runs sharing a common objective.

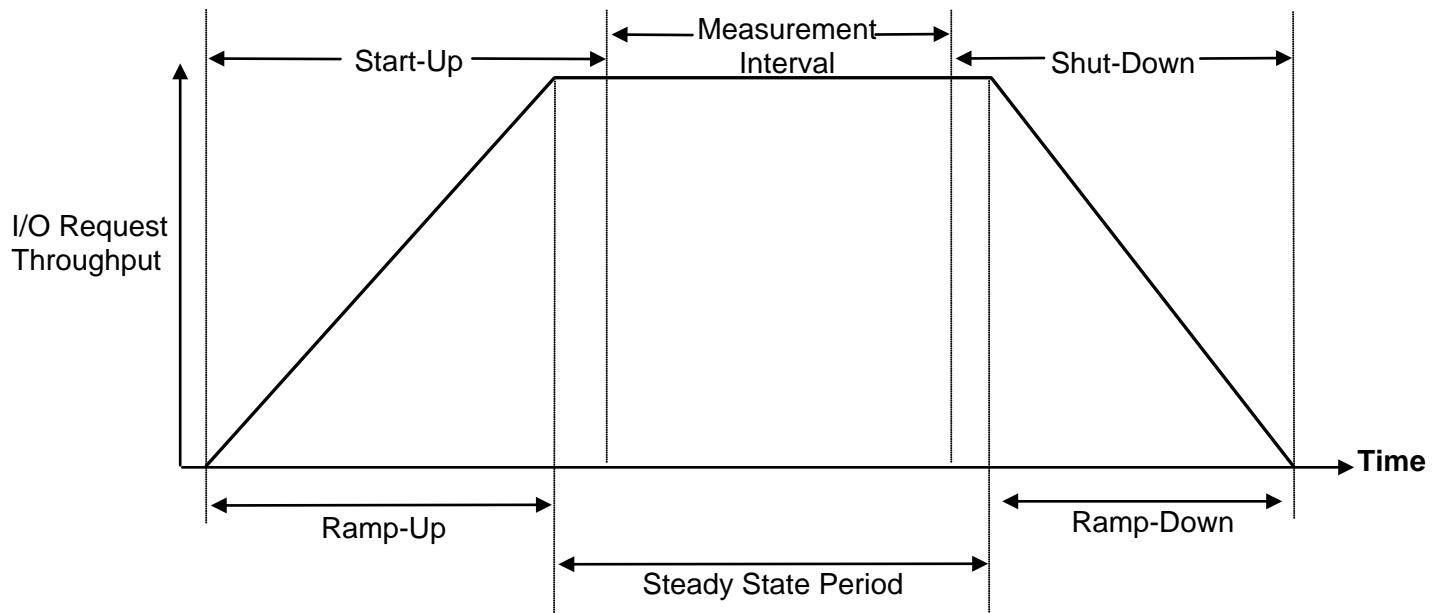
Test Run: The execution of SPC-1 for the purpose of producing or supporting an SPC-1 test result. SPC-1 Test Runs may have a finite and measured Ramp-Up period, Start-Up period, Shut-Down period, and Ramp-Down period as illustrated in the “SPC-1 Test Run Components” below. All SPC-1 Test Runs shall have a Steady State period and a Measurement Interval.

Test Phase: A collection of one or more SPC-1 Test Runs sharing a common objective and intended to be run in a specific sequence.

I/O Completion Types



SPC-1 Test Run Components



APPENDIX B: CUSTOMER TUNABLE PARAMETERS AND OPTIONS

Red Hat Enterprise Linux 6.4 (64-bit)

- Change the I/O scheduler from the default of **cfq** to **noop** on each Host System, which will result in all incoming I/O requests inserted into a simple, unordered FIFO queue. This was done by adding the following parameter in **/boot/grub.conf** file at the end of the kernel line on each Host System.

elevator= noop

- Change the QLogic driver queue depth from a default of 64 to 255 on each Host System by manually creating the file, **/etc/modprobe.d/qla2xxx.conf**, which contains the following line:

options qla2xxx ql2maxqdepth=255

Once the file is created, rebuild the RHEL 6.4 ramdisk by issuing the CLI commands listed below on each Host System, then reboot each Host System so that the QLogic drive automatically loads with a queue depth of 255. This change is persistent across reboots.

```
cd /boot
dracut --force "initramfs-$(uname -r).img" $(uname -r)
```

Storage Controllers

The following parameters/options were changed on each controller with the scripts documented in [Appendix C: Tested Storage Configuration \(TSC\) Creation](#).

Root Aggregate

- Change the aggregate snapshot reserve, **snap reserve**, to 0 bytes so that no space is reserved for snapshots.
- Change the aggregate parameter, **nosnap**, from its default value of **off** to **on**. This disables the automatic creation of snapshots.

SPC Data Aggregates

- Change the aggregate snapshot reserve, **snap reserve**, to 0 bytes so that no space is reserved for snapshots.
- Change the aggregate parameter, **nosnap**, from the default value of **off** to **on**. This disables the automatic creation of snapshots.
- Change the aggregate parameter, **max-write-alloc-blocks**, from the default value of **64** to **256**. This changes the number of blocks written to a spindle before the write allocator changes to a new spindle, providing a better on disk layout for long sequential reads.
- Create the volumes with a space reservation requirement of **none** instead of the default volume.

LUN Options

Create the LUNs with space reservations using the **space-reserve enabled** option during LUN creation.

Storage System Options

The **walf.optimize_write_once** option was changed from the default value of **on** to **off**. This option affects the initial layout of data within a newly created aggregate. The default data layout favors applications that do not overwrite data.

APPENDIX C: TESTED STORAGE CONFIGURATION (TSC) CREATION

This documentation begins with the all of benchmark configuration hardware and software installed and configured on the Host Systems and TSC, but the storage devices have yet to be configured for execution of the SPC-1 Test Runs.

Configure the Storage

Two main Python scripts, [**config_storage.py**](#) and [**config_hosts.py**](#), are used to create the RAID groups, aggregates, volumes, LUNs and SPC-1 Logical Volumes that comprise the SPC-1 data repository. In addition, the scripts will change the appropriate customer tunable parameter/options, documented on page [66 \(*Appendix B: Customer Tunable Parameters and Options*\)](#).

The two main Python scripts require three basic Python modules: [**harness.py**](#), [**storage.py**](#) and [**host.py**](#). In addition, the two main scripts require a configuration file, [**config_full**](#), which was manually created with the appropriate configuration variable values.

The two main scripts, configuration file and three subordinate scripts configuration file appear in the [Referenced File and Scripts](#) section at the end of this section, after the detailed description of each of the two main scripts.

Create and Configure RAID Groups, Aggregates, Volumes and LUNs

This first script, [**config_storage.py**](#), is invoked from any of the Host Systems where Python is installed, as follows:

```
python config_storage.py --config < config file> --log <dir to store output logs> -n
```

The script does the following on each controller:

- Set the following option: **walf.optimize_write_once option off**
- Set the timezone and create NTP server as defined in [**config_full**](#).
- Rename the root aggregates as specified in [**config_full**](#).
- Set root aggregate snapshot setting as follows:
 - Set aggregate option **nosnap=on**.
 - Set aggregate snap sched to 0 0 0
 - Delete any existing snapshot.
- Set root volume snapshot setting as follows:
 - Set volume option **nosnap=on**
 - Set volume snap sched to 0 0 0
 - Set snap reserve=0
 - Delete any existing snapshot.
- Create the SPC-1 data aggregates as specified in [**config_full**](#), which will create five RAID groups with a RAID group size of 19 disks and 1 spare.
- Set the SPC-1 data aggregate snapshot setting as follows:
 - Set aggregate option **nosnap=on**.
 - Set aggregate snap sched to 0 0 0

- Set the following field on the data aggregates: ***max-write_alloc-blocks=256***.
- Create a **Vserver**, “**vs1**”, as specified in [**config_full**](#).
(A Vserver is virtual storage server in a NetApp clustered storage configuration that manages an individual workload.)
- Set the following fields for the **Vserver**, “**vs1**”:
-nsswitch=file
-rootvolume-security-style=unix
-snapshot-policy none
-allowed-protocols=fcp
-percent-snapshot-space=0
- Create the following data volumes on each storage controller as specified in [**config_full**](#):
 - 12 volumes of 1,032,324,828 KB each in the data aggregate. Those volumes will contain the LUNs for ASU-1.
 - 12 volumes of 1,032,324,828 KB each in the data aggregate. Those volumes will contain the LUNs for ASU-2.
 - 12 volumes of 229,405,517 KB each in the data aggregate. Those volumes will contain the LUNs for ASU-3.
- Set the following fields on each of the data volumes:
-security-style=unix
-space-guarantee=none
-percent-snapshot-space=0
-snapshot-policy none
- Create the LUNs on each storage controller as specified in [**config_full**](#), which will create the following on each storage controller:
 - One LUN of 589,899,902 KB in each data volume designated for ASU-1.
 - One LUN of 589,899,902 KB in each data volume designated for ASU-2.
 - One LUN of 131,088,867 KB in each data volume designated for ASU-3.
- Set the following fields on each of the LUNs:
-ostype=linux
-space-reserve=enabled
- Create an **igroup** and map the LUNs as specified in [**config_full**](#), which will do the following:
 - Create an **igroup**, “**ig1**”, with the following settings:
-protocol=fcp
-ostype=linux
-Contains eight initiator WWPN (*two from each Host System specified in config_full*)
 - Map each LUN to **igroup**, “**ig1**” and assign the LUN to **Vserver** “**vs1**”.
- Create two data LIFs (*logical interfaces*), as specified in [**config_full**](#), using physical on board FC target ports **0e** and **0f** on each storage controller with the following settings, which will assign each LIF to the **Vserver** “**vs1**”:

```
-vserver=vs1
-role=data
-data-protocol=fcp
-home-node=<name of the cluster node physical FC target ports reside on>
-home-port=<either 0e or 0f, as appropriate>
```

- Start an FCP server (*fibre channel service*) on the **Vserver “vs1”**, with the following setting:

-state-admin=up

-vserver=vs1

Create and Configure SPC-1 Logical Volumes

The [**config hosts.py**](#) script is executed on the Master Host System after successful completion of [**config storage.py**](#) and does the following:

- Reboot the Master Host System to scan all LUNs.
- Build a mapping of the LUNs to the associated Red Hat native multipath device on the Master Host System using the output of the NetApp Linux FC Utilities 6.1 command: **sanlun lun show -p**.
- Create the RHEL 6.4 LVM2 logical volumes, which correspond to the SPC-1 Logical volumes, as specified in [**config full**](#).
 - Initialize the first two logical volumes for ASU-1 from each storage controller using **pvcreate**.

-i 2 (number of stripes)

-I 1024 (stripe size in KiB)

-I 100%VG (use 100% of space in the volume group)

-n <name of the lv>
 - Create a volume group comprised of the two multipath devices that were initialized in the previous step.
 - Create a logical volume with the following settings:

-i 2 (number of stripes)

-I 1024 (stripe size in KiB)

-I 100%VG (use 100% of space in the volume group)

-n <name of the lv>
 - Repeat the previous steps #1-#3 36 additional times, which will result in the following:
 - A total of 36 SPC-1 Logical Volumes: 12 for ASU-1, 12 for ASU-2 and 12 for ASU-3.
 - One SPC-1 Logical Volume per volume group.
 - Each SPC-1 Logical Volume is striped across two multipath devices (LUNs); one LUN from each storage controller.
- Reboot all four Host Systems so that each Host System will scan the LUNs and the SPC-1 Logical Volumes. Logical volume meta data was written on the LUNs when they were created from the Master Host System, therefore, when the remaining Host Systems scan the LUNs, the same meta data will be read by all Host Systems resulting in same SPC-1 ASU Logical Volumes visible on all Host Systems.
- The script will wait until all Host Systems are back up and then build SPC-1 configuration and JVM slave configuration files for Master and Slave Host Systems,

and place the appropriate configuration files in the SPC-1 run directory on each Host System defined in [config_full](#).

Referenced File and Scripts

config_full

```
[DEFAULT]
timeserver = 10.60.252.15
timezone = EST5EDT
sanlun = /usr/sbin/sanlun

## Benchmark configuration data
[SPC-1]
bsu_rate = 5001
rundir = /SPC1RunDir
Xms = 1536m
Xmx = 2048m
Xss = 256k
size_linux = /SPC1RunDir/scripts/spc_audit/bin/size_linux

## Host configuration data
[hosts]
master = stlx300s7-120
slaves = stlx300s7-120 stlx300s7-121 stlx300s7-122 stlx300s7-123
login = root
password = Netapp123
post_reboot_init_time = 60

[stlx300s7-120]
slave_num = 1
ip_addr = 10.63.173.44
initiators = 21000024ff49bc56 21000024ff49bc57

[stlx300s7-121]
slave_num = 2
ip_addr = 10.63.173.46
initiators = 21000024ff49d702 21000024ff49d703

[stlx300s7-122]
slave_num = 3
ip_addr = 10.63.173.48
initiators = 21000024ff49bc84 21000024ff49bc85

[stlx300s7-123]
slave_num = 4
ip_addr = 10.63.173.50
initiators = 21000024ff49d7d6 21000024ff49d7d7

[lvm]
stripes = 2
stripesize = 1024
extents = 100%%VG
lv_create_host = stlx300s7-120

## ONTAP Storage configuration data
[cluster]
name = spc-1
nodes = spc-1-01 spc-1-02
ip_addr = 10.63.173.244
login: admin
```

```
password: Netapp123
data_aggrs = aggr1
vserver = vs1
host_ostype = linux
igroup_name = ig1
fcp_targets = 0e 0f

[spc-1-01]
node_num = 1
root_aggr = aggr0_spc_1_01_0

[spc-1-02]
node_num = 2
root_aggr = aggr0_spc_1_02_0

[aggr1]
diskcount = 92
raidtype = raid_dp
maxraidsize = 19
volume-style = flex

[vsl]
rootvolume = vsl_rootvol0
aggregate = aggr1_n1

[asu1]
vol_size = 1032324828KB
vols_per_node = 12
luns_per_vol = 1
lun_size = 589899902KB

[asu2]
vol_size = 1032324828KB
vols_per_node = 12
luns_per_vol = 1
lun_size = 589899902KB

[asu3]
vol_size = 229405517KB
vols_per_node = 12
luns_per_vol = 1
lun_size = 131088867KB
```

config_storage.py

```
import getopt
import sys
import time

print sys.path
from lib.storage import StorageHarness

def usage(err=None):
    if err:
        sys.stderr.write("ERROR: %s\n" % (err))
    sys.stderr.write("Usage:\n")
    sys.stderr.write("> config_storage.py --config <config_file> --log <dir> [--"
print] [--ntp_setup] [--rename_root_aggrs]\n")
    sys.stderr.write(" --config|-c <config_file>  use configuration file in \"ini\""
format\n")
```

```

        sys.stderr.write(" --log|-l <dir>
(default=./logs\")\n")
        sys.stderr.write(" --print|-p
be executed,\n")
        sys.stderr.write(" anything\n")
        sys.stderr.write(" --ntp_setup|-n
settings\n")
        sys.stderr.write(" --rename_root_aggrs|-r
aggr0_n[0-9]+\n")
        sys.stderr.write(" --help|-h\n\n")
        sys.stderr.write("Examples:\n")
        sys.stderr.write(" config_storage.py --config ./spc_script_config\n")
        sys.stderr.write(" - performs storage configuration on NetApp cluster
specified in the\n")
        sys.stderr.write("     file ./spc_script_config\n")
        sys.stderr.write(" config_storage.py --config ./spc_script_config --print\n")
        sys.stderr.write(" - report all of the commands that will be executed in
order to configure\n")
        sys.stderr.write("     the NetApp cluster specified in the config file
./spc_script_config\n\n")
        sys.stderr.write("Notes:\n")
        sys.stderr.write(" - config file format:
U{http://effbot.org/librarybook/configparser.htm}\n")

def command_line_args(argv):
    """Processes command line arguments into user_data dict"""
    user_data = {"print":False, "logdir": "./logs", "ntp_setup":False,
                 "rename_root_aggrs":False}
    if not argv[1:]:
        usage("missing arguments")
        sys.exit(1)
    opts, args = getopt.getopt(argv[1:], "c:l:nphr", ["config=", "print", "help",
                                                       "log=", "ntp_setup",
                                                       "rename_root_aggrs"])
    for a, o in opts:
        if a in ("-h", "--help"):
            usage()
            sys.exit(0)
        elif a in ("-c", "--config"):
            user_data["config_file"] = o
        elif a in ("-p", "--print"):
            user_data["print"] = True
        elif a in ("l", "--log"):
            user_data["logdir"] = o
        elif a in ("n", "--ntp_setup"):
            user_data["ntp_setup"] = True
        elif a in ("r", "--rename_root_aggrs"):
            user_data["rename_root_aggrs"] = True
        else:
            usage("unrecognized argument: %s" % (a))
            sys.exit(1)
    if not user_data.has_key("config_file"):
        usage("missing required argument '--config_file'")
        sys.exit(1)
    return user_data

def configure_cluster_shell(storage):
    """Some shell settings to help with automation"""
    storage.send_cluster("rows 0")
    storage.send_cluster("set -confirmations off")

```

```

storage.send_cluster("set -privilege diagnostic")

def apply_wafl_options(storage):
    storage.node_run_all("options wafl.optimize_write_once off")

def apply_time_settings(storage):
    timezone = storage.config_data("timezone")
    timeserver = storage.config_data("timeserver")
    storage.send_cluster("timezone -timezone %s" % (timezone))
    for node in storage.nodes():
        ntp_create = "ntp server create -node %s -server %s" % (node,
                                                               timeserver)
        storage.send_cluster(ntp_create)
    storage.send_cluster("ntp config modify -enabled true")

def rename_root_aggrs(storage):
    for node in storage.nodes():
        old_aggr_name = storage.config_data("root_aggr", node.name)
        new_aggr_name = "aggr0_n%s" % (node.number)
        rename_cmd = ("storage aggregate rename -aggregate %s -newname %s" %
                      (old_aggr_name, new_aggr_name))
        storage.send_cluster(rename_cmd)
        # save the new name for later use
        node.root_aggr = new_aggr_name

def eliminate_root_aggr_snapshots(storage):
    for node in storage.nodes():
        storage.node_run(node, "aggr options %s nosnap on" % (node.root_aggr))
        storage.node_run(node, "snap sched -A %s 0 0 0" % (node.root_aggr))
        storage.node_run(node, "snap delete -A -a -f %s" % (node.root_aggr))

def eliminate_root_vol_snapshots(storage):
    storage.node_run_all("vol options vol0 nosnap on")
    storage.node_run_all("snap sched vol0 0 0 0")
    storage.node_run_all("snap reserve vol0 0")
    storage.node_run_all("snap delete -a -f vol0")

def create_data_aggrs(storage):
    for node in storage.nodes():
        data_aggr = storage.config_data("data_aggrs", "cluster")
        aggr_name = "%s_n%s" % (data_aggr, node.number)
        diskcount = storage.config_data("diskcount", data_aggr)
        raidtype = storage.config_data("raidtype", data_aggr)
        maxraidszie = storage.config_data("maxraidszie", data_aggr)
        vol_style = storage.config_data("volume-style", data_aggr)
        aggr_create = ("aggregate create -aggregate %s" % (aggr_name) +
                      "--diskcount %s -nodes %s -raidtype %s -maxraidszie %s" %
                      (diskcount, node, raidtype, maxraidszie) +
                      " -volume-style %s" % (vol_style))
        storage.send_cluster(aggr_create, timeout=180)
        # waits a max of 15*60 seconds for aggr create to complete
        # storage.wait_for_last_job_complete(interval=60, retries=10)
        time.sleep(15)
        storage.node_run(node, "aggr options %s nosnap on" % (aggr_name))
        storage.node_run(node, "snap sched -A %s 0 0 0" % (aggr_name))
        # save this for later use
        node.data_aggr = aggr_name

```

```

def set_max_walloc_blocks_on_data_aggrs(storage, max_walloc_blocks="256"):
    aggr_modify = ("aggr modify -aggregate aggr1_n* " +
                   "-max-write-alloc-blocks %s" % (max_walloc_blocks))
    storage.send_cluster(aggr_modify, timeout=120)

def create_vserver(storage):
    vserver = storage.config_data("vserver", "cluster")
    vs_root = storage.config_data("rootvolume", vserver)
    aggr = storage.config_data("aggregate", vserver)
    vs_create = ("vserver create -vserver %s -rootvolume %s -aggregate %s " %
                 (vserver, vs_root, aggr) +
                 "-ns-switch file -rootvolume-security-style unix " +
                 "-snapshot-policy none")
    storage.send_cluster(vs_create)
    storage.send_cluster("vserver modify -vserver %s -allowed-protocols fcp" %
                         (vserver))
    storage.send_cluster("volume modify -volume %s -vserver %s " %
                         (vs_root, vserver) +
                         "-percent-snapshot-space 0")
    # save the vserver for later use
    storage.cluster.vserver = vserver

def create_data_vols(storage):
    for node in storage.nodes():
        node.asu_vols = {}
        vol_num = 0
        vol_num += create_node_asu_vols(storage, node, "asu1", vol_num)
        vol_num += create_node_asu_vols(storage, node, "asu2", vol_num)
        vol_num += create_node_asu_vols(storage, node, "asu3", vol_num)

def create_node_asu_vols(storage, node, asu, vol_num):
    num_vols = int(storage.config_data("vols_per_node", asu))
    # save vols per asu in a list for later use
    node.asu_vols[asu] = []
    for i in range(num_vols):
        vol_name = "%s_n%s_v%02d" % (asu, node.number, vol_num)
        vol_size = storage.config_data("vol_size", asu)
        vol_create = ("volume create -volume %s -vserver %s -aggregate %s " %
                     (vol_name, storage.cluster.vserver, node.data_aggr) +
                     "-size %s -state online -security-style unix " %
                     (vol_size) + "-space-guarantee none " +
                     "-percent-snapshot-space 0 -snapshot-policy none")
        storage.send_cluster(vol_create, timeout=60)
        node.asu_vols[asu].append(vol_name)
        vol_num += 1
    return num_vols

def create_data_luns(storage):
    for node in storage.nodes():
        lun_num = 0
        lun_num += create_node_asu_luns(storage, node, "asu1", lun_num)
        lun_num += create_node_asu_luns(storage, node, "asu2", lun_num)
        lun_num += create_node_asu_luns(storage, node, "asu3", lun_num)

def create_node_asu_luns(storage, node, asu, lun_num):
    lun_count = 0

```

```

luns_per_vol = int(storage.config_data("luns_per_vol", asu))
lun_size = storage.config_data("lun_size", asu)
lun_ostype = storage.config_data("host_ostype", "cluster")
for data_vol in node.asu_vols[asu]:
    vol_path = "/vol/%s" % (data_vol)
    for i in range(luns_per_vol):
        lun_name = "%s_n%s_l%02d" % (asu, node.number, lun_num)
        lun_path = "%s/%s" % (vol_path, lun_name)
        lun_create = ("lun create %s -size %s -ostype %s " %
                      (lun_path, lun_size, lun_ostype) +
                      "-space-reserve enabled -vserver %s" %
                      (storage.cluster.vserver))
        storage.send_cluster(lun_create)
        lun_num += 1
        lun_count += 1
    return lun_count

def get_initiator_wwpns(storage):
    """Returns a list of wwpns from each host definition"""
    wwpns = set()
    for host in (storage.config_data("master", "hosts").split() +
                 storage.config_data("slaves", "hosts").split()):
        for wwpn in storage.config_data("initiators", host).split():
            wwpns.add(wwpn)
    return sorted(wwpns)

def create_igroup_and_map_luns(storage):
    igroup_name = storage.config_data("igroup_name", "cluster")
    igroup_ostype = storage.config_data("host_ostype", "cluster")
    initiator_wwpns = ",".join(get_initiator_wwpns(storage))
    igroup_create = ("igroup create -igroup %s -protocol fcp -ostype %s " %
                     (igroup_name, igroup_ostype) +
                     "-vserver %s -initiator %s" % (storage.cluster.vserver,
                                                     initiator_wwpns))
    storage.send_cluster(igroup_create)
    storage.send_cluster("lun map -path /vol/asu*/asu* -igroup %s -vserver %s" %
                         %(igroup_name, storage.cluster.vserver))

def create_data_lifs(storage):
    vserver = storage.cluster.vserver
    for node in storage.nodes():
        for fcp_port in storage.config_data("fcp_targets", "cluster").split():
            lif_name = "n%s%s" % (node.number, fcp_port)
            lif_create = ("network interface create -vserver %s -lif %s " %
                          (vserver, lif_name) + "-role data " +
                          "-data-protocol fcp -home-node %s -home-port %s" %
                          (node, fcp_port))
            storage.send_cluster(lif_create)

def create_fcp_server(storage):
    fcp_create = ("vserver fcp create -status-admin up -vserver %s" %
                  (storage.cluster.vserver))
    storage.send_cluster(fcp_create)

if __name__ == "__main__":
    # Begin main
    logger = None
    user_args = command_line_args(sys.argv)

```

```

storage = StorageHarness(user_args["config_file"],
                        logdir=user_args["logdir"],
                        diag_mode=user_args["print"])
try:
    storage.initialize()
    logger = storage.getLogger()
    storage.console_message("Starting storage configuration...")
    logger.info("Starting storage configuration...")
    configure_cluster_shell(storage)
    apply_wafl_options(storage)
    if user_args["ntp_setup"]:
        apply_time_settings(storage)
    if user_args["rename_root_aggrs"]:
        rename_root_aggrs(storage)
        eliminate_root_aggr_snapshots(storage)
        eliminate_root_vol_snapshots(storage)
    create_data_aggrs(storage)
    set_max_walloc_blocks_on_data_aggrs(storage, max_walloc_blocks="256")
    create_vserver(storage)
    create_data_vols(storage)
    create_data_luns(storage)
    create_igroup_and_map_luns(storage)
    create_data_lifs(storage)
    create_fcp_server(storage)
finally:
    storage.cleanup()
    logger.info("...storage configuration complete!")

storage.console_message("...storage configuration complete!")

```

config_hosts.py

```

import os
import re
import getopt
import sys
import math

from lib.host import SPCHarness, SanlunDevMap, LinuxVolumeGroup, LinuxVolume

def usage(err=None):
    if err:
        sys.stderr.write("ERROR: %s\n" % (err))
    sys.stderr.write("Usage:\n")
    sys.stderr.write("> config_hosts.py --config <config_file> --log <dir> [--print]
[--sanlun_out <sanlun_output_file>] [--spc1_config]\n")
    sys.stderr.write("  --config|-c <config_file>  use configuration file in \"ini\""
format"\n")
    sys.stderr.write("  --log|-l <dir>                  directory to store output logs
(default=\"./logs\")\n")
    sys.stderr.write("  --print|-p                   generate a report of commands to
be executed,\n")
    sys.stderr.write("  anything\n")                  without actually executing
                                         provide a sanlun output file to
                                         NOTE: this is useful in
                                         build SPC-1 input configuration
                                         files and update SPC-1 RunDir\n")
    sys.stderr.write("  --help|-h\n")                sys.stderr.write("Examples:\n")

```

```

        sys.stderr.write(" config_hosts.py --config ./spc_script_config\n")
        sys.stderr.write(" - performs host configuration on NetApp cluster specified
in the\n")
        sys.stderr.write("     file ./spc_script_config\n")
        sys.stderr.write(" config_hosts.py --config ./spc_script_config --print --
sanlun_out=./sanlun.out\n")
        sys.stderr.write(" - report all of the commands that will be executed in
order to configure\n")
        sys.stderr.write("     SPC hosts specified in the config file
./spc_script_config using the output\n")
        sys.stderr.write("     on sanlun in the file ./sanlun.out\n\n")
        sys.stderr.write("Notes:\n")
        sys.stderr.write(" - config file format:
U{http://effbot.org/librarybook/configparser.htm}\n")
        sys.stderr.write(" - --print without --sanlun_out will cause 'sanlun lun show -
p' to be\n")
        sys.stderr.write("     executed on the SPC master host\n")
        sys.stderr.write(" - --spc1_config builds the master, SPC1.cfg file and a
<jvm>.txt file\n")
        sys.stderr.write("     for each JVM to be created and will update the rundir with
these\n")
        sys.stderr.write("     files if 'rundir' is specified in the config file\n")

def command_line_args(argv):
    """Processes command line arguments into user_data dict"""
    user_data = {"print": False, "logdir": "./logs", "sanlun_out": None,
                 "spc1_config": False}
    if not argv[1:]:
        usage("missing arguments")
        sys.exit(1)
    opts, args = getopt.getopt(argv[1:], "c:l:ps:hx", ["config=", "print",
                                                       "help", "log=",
                                                       "sanlun_out=",
                                                       "spc1_config"])
    for a, o in opts:
        if a in ("-h", "--help"):
            usage()
            sys.exit(0)
        elif a in ("--config"):
            user_data["config_file"] = o
        elif a in ("--print"):
            user_data["print"] = True
        elif a in ("--log"):
            user_data["logdir"] = o
        elif a in ("--sanlun_out"):
            user_data["sanlun_out"] = o
        elif a in ("--spc1_config"):
            user_data["spc1_config"] = True
        else:
            usage("unrecognized argument: %s" % (a))
            sys.exit(1)
    if not user_data.has_key("config_file"):
        usage("missing required argument '--config_file'")
        sys.exit(1)
    return user_data

def reboot_lv_host(harness):
    lv_host = harness.get_lv_host()
    harness.reboot_host(lv_host)

```

```

def build_sanlun_map(harness, sanlun_outfile):
    re_no_sanlun = "No such file or directory"
    sanlun_bin = harness.config_data("sanlun")
    sanlun_output = harness.send_lv_host("%s lun show -p" % (sanlun_bin))
    # Use the sanlun output file if one was provided
    if sanlun_outfile != None:
        sanlun_output = open(sanlun_outfile, "r")
    elif re.search(re_no_sanlun, sanlun_output):
        raise RuntimeError("%s. Please make sure the 'sanlun' field is %s
                            (sanlun_output) + "set correctly in your config " +
                            "file: %s" % (harness.config_file))
    return SanlunDevMap(sanlun_output)

def build_asu_map(dev_map,asu_map):
    """Runs through devices found from sanlun and updates their ASU and LUN#"""
    for dev in dev_map:
        lun_path = dev.lun_path
        (asu_num, lun_num) = extract_info_from_lun_path(lun_path)
        dev.asu = asu_num
        dev.lun_num = lun_num
        update_asu_map(dev.asu, dev.lun_num, dev, asu_map)
    return asu_map

def extract_info_from_lun_path(lun_path):
    """Retrieves asu# and lun# from storage lun path

    @attention: assumes that Volumes and LUNs were named according to the
    config_storage.py naming conventions.

    """
    re_lun_path = "/vol/[^\n]+/(asu[1-3])_n[\d]+_(\l[\d]+)"
    match = re.search(re_lun_path, lun_path)
    key_info = None
    if match:
        asu_num = match.group(1)
        lun_num = match.group(2)
        key_info = (asu_num, lun_num)
    else:
        raise RuntimeError("Could not determine ASU and LUN num from %s " %
                           (lun_path) + "using regexp: %s" %
                           (re_lun_path))
    return key_info

def update_asu_map(asu, lun_num, cur_dev, asu_map):
    """Divide devices into lists that can be referenced via the key (asu, lun#)

    """
    asu_key = (asu, lun_num)
    if not asu_map.has_key(asu_key):
        asu_map[asu_key] = []
    asu_map[asu_key].append(cur_dev)

def define_logical_volumes(harness, asu_map, lv_devs, spcl_devs):
    cur = 0
    lv_stripes = harness.config_data("stripes", "lvm")
    lv_stripesize = harness.config_data("stripesize", "lvm")
    lv_extents = harness.config_data("extents", "lvm")
    sorted_keys = sorted(asu_map.keys())
    for (asu, lun_num) in sorted_keys:

```

```

       asu_key = (asu, lun_num)
        dev_list = asu_map[asu_key]
        vg_name = "%s_vg%02d" % (asu, cur)
        lv_name = "%s_lv%02d" % (asu, cur)
        lv_dev = LinuxVolume(lv_name, LinuxVolumeGroup(vg_name, dev_list),
                             lv_stripes, lv_stripesize, lv_extents)
        lv_devs.append(lv_dev)
        spcl_devs[asu_key] = lv_dev
        cur += 1

def create_logical_volumes(harness, lv_list):
    for lv in lv_list:
        create_volume_group(harness, lv.volume_group)
        harness.send_lv_host("lvcreate -i %s -I %s -l %s -n %s %s" %
                             (lv.stripes, lv.stripesize, lv.extents, lv.name,
                              lv.volume_group.name))

def create_volume_group(harness, volume_group):
    initialize_partition(harness, volume_group)
    vg_create = "vgcreate %s" % (volume_group.name)
    for dev in volume_group.dev_list:
        vg_create += "%s" % (dev.mpath_dev)
    harness.send_lv_host(vg_create)

def initialize_partition(harness, volume_group):
    for dev in volume_group.dev_list:
        harness.send_lv_host("pvcreate %s" % (dev.mpath_dev))

def set_spcl_dev_sizes(harness, spcl_devs, size_out=None):
    for lv_dev in spcl_devs.values():
        set_dev_size(harness, lv_dev, size_out)

def set_dev_size(harness, device, size_out=None):
    re_size_bytes = "file size is ([0-9]+)"
    re_no_size_linux = "No such file or directory"
    size_linux = harness.config_data("size_linux", "SPC-1")
    cmd = "%s %s" % (size_linux, device.path())
    size_output = harness.send_lv_host(cmd)
    if size_output is None and size_out != None:
        size_output = size_out
    match = re.search(re_size_bytes, size_output)
    if match:
        device.size = match.group(1)
    elif re.search(re_no_size_linux, size_output):
        raise RuntimeError("%s. Please check your 'size_linux' field in %s" %
                           (size_output) + "your config file: %s" %
                           (harness.config_file))
    else:
        raise Exception("unable to determine size of device %s from '%s' " %
                       (device, size_linux) + "output: '%s'\n" % (size_output)
                      + "...please check logs in %s" % (harness.logdir) +
                      "for errors.")

def jvm_per_host(harness):
    """Returns the number of JVMs per host needed to meet the BSU rate"""
    logger = harness.getLogger()
    num_hosts = len(harness.slaves)

```

```

bsu_rate = float(harness.config_data("bsu_rate", "SPC-1"))
bsu_per_host = math.ceil(bsu_rate/num_hosts)
jvms_per_host = math.ceil(bsu_per_host/100.0)
logger.debug("Building SPC-1 config for %s hosts, total BSUs=%s, " %
             (num_hosts, bsu_rate) + "BSUs per host=%s, JVMs per host=%s" %
             (bsu_per_host, jvms_per_host))
return int(jvms_per_host)

def build_spcl_config_files(harness, spcl_devs):
    slave_jvms = build_slave_config(harness, spcl_devs)
    build_master_config(harness, spcl_devs, slave_jvms)

def build_master_config(harness, spcl_devs, slave_jvm_list):
    master_config = []
    master_config.append(jvm_settings(harness))
    master_config.append("host=master")
    master_config.append("slaves=(%s)" % (",".join(slave_jvm_list)))
    master_config.extend(spcl_config_body(harness, spcl_devs))
    write_spcl_config(harness, harness.master, master_config, "spcl.cfg")

def build_slave_config(harness, spcl_devs):
    """Builds slave configurations and returns slave jvm names for master"""
    slave_jvms = []
    for host in harness.slaves:
        slave_num = harness.config_data("slave_num", host.name)
        jvm_num = 1
        for jvm in range(jvm_per_host(harness)):
            config_lines = []
            jvm_name = "h%s_slave%s" % (slave_num, jvm_num)
            slave_jvms.append(jvm_name)
            config_lines.extend(spcl_config_slave_header(harness, jvm_name))
            config_lines.extend(spcl_config_body(harness, spcl_devs))
            write_spcl_config(harness, host, config_lines,
                              ("%s.txt" % (jvm_name)))
            jvm_num += 1
    return slave_jvms

def write_spcl_config(harness, host, config_lines, filename):
    rundir = harness.config_data("rundir", "SPC-1")
    config_file = os.path.join(rundir, filename)
    harness.send_host(host, "rm -f %s" % (config_file))
    for line in config_lines:
        harness.send_host(host, "echo '%s' >> %s" % (line, config_file))

def spcl_config_slave_header(harness, jvm_name):
    config_lines = []
    config_lines.append(jvm_settings(harness))
    config_lines.append("master=%s" % (harness.master.ip_addr))
    config_lines.append("host=%s" % (jvm_name))
    return config_lines

def jvm_settings(harness):
    Xms = harness.config_data("Xms", "SPC-1")
    Xmx = harness.config_data("Xmx", "SPC-1")
    Xss = harness.config_data("Xss", "SPC-1")
    return 'javaparms="-Xms%s -Xmx%s -Xss%s"' % (Xms, Xmx, Xss)

```

```

def spc1_config_body(harness, spc1_devs):
    config_lines = []
    sorted_keys = sorted(spc1_devs.keys())
    for (asu, lun_num) in sorted_keys:
        sd_name = "%s_%s" % (asu, lun_num)
        lv_dev = spc1_devs[(asu, lun_num)]
        lun_name = lv_dev.path()
        size = lv_dev.size
        config_line = "sd=%s,lun=%s,size=%s" % (sd_name, lun_name, size)
        config_lines.append(config_line)
    return config_lines

if __name__ == "__main__":
    # Begin main
    logger = None
    dev_map = None
    asu_map = {}
    lv_devs = []
    spc1_devs = {}
    user_args = command_line_args(sys.argv)
    spc_harness = SPCHarness(user_args["config_file"],
                             logdir=user_args["logdir"],
                             diag_mode=user_args["print"],
                             name="config_hosts")
    try:
        spc_harness.initialize()
        logger = spc_harness.getLogger()
        logger.info("Starting SPC-1 host configuration...")
        spc_harness.console_message("Starting SPC-1 host configuration...")
        reboot_lv_host(spc_harness)
        dev_map = build_sanlun_map(spc_harness, user_args["sanlun_out"])
        build_asu_map(dev_map, asu_map)
        define_logical_volumes(spc_harness, asu_map, lv_devs, spc1_devs)
        create_logical_volumes(spc_harness, lv_devs)
        spc_harness.reboot_hosts()
        if user_args["spc1_config"]:
            size_out = None
            if user_args["print"]:
                # to help with diag mode
                size_out = "file size is 000 (00.00 megabytes)"
            set_spc1_dev_sizes(spc_harness, spc1_devs, size_out)
            build_spc1_config_files(spc_harness, spc1_devs)
    finally:
        try:
            spc_harness.cleanup()
            logger.info("...SPC-1 host configuration complete!")
            spc_harness.console_message("...SPC-1 host configuration complete!")
        except:
            pass

```

harness.py

```

import os
import logging
import socket
import time
import sys

import ConfigParser
import pexpect

```

```

# Some regular expressions to help with prompt matching
re_unix_prompt = '(?m)([\r\n][^%#$\r\n]*[%#$][\s])'
# The regexp needs self.name applied with format(name) operation
re_cluster_prompt = '(?m)({0}({#|%|>|\*|>|::})|::[a-zA-Z0-9 ]*[*]> )'

def get_logger(category='main', filename=None):
    if len(logging.getLogger(category).handlers) <= 0:
        if not filename:
            raise IOError("no filename provided for logger('%s')" % (category))
        return file_logger(filename, open_mode="a", category=category)
    else:
        return logging.getLogger(category)

def log_formatter(category='main', date_format = '%a %b %d %X %Z %Y'):
    hostname = socket.gethostname()
    fmt = ('%({asctime)s (' + hostname + ' %(process)d-%(threadName)s' + ') ' +
           '[%(lineno)d %(module)s.%funcName)s]: ' +
           '%(levelname)s: %(message)s')
    logging.Formatter.converter = time.gmtime
    return logging.Formatter(fmt=fmt, datefmt=date_format)

def file_logger(filename, open_mode="a", category="main", level=None):
    if level is None:
        level = logging.DEBUG
    hand = logging.FileHandler(filename, open_mode)
    log = logging.getLogger(category)
    log.setLevel(level)
    hand.setFormatter(log_formatter(category))
    log.addHandler(hand)
    return log

class DiagReport(object):

    def __init__(self):
        self.commands = []

    def __len__(self):
        return len(self.commands)

    def __getitem__(self, index):
        return self.commands[index]

    def __contains__(self, key):
        return key in self.commands

    def __str__(self):
        return self.generate()

    def append(self, item):
        self.commands.append(item)

    def generate(self):
        report = "---Diagnostic Report---\n"
        for i in range(len(self)):
            report += "%s. %s\n" % (i+1, self.commands[i])
        return report

```

```

class ScriptHarness(object):

    def __init__(self, config_file, name=None, logdir=None, diag_mode=False):
        self.config_file = config_file
        self.name = name
        self.logdir = logdir
        self.config = None
        self.logger = None
        self.diag_report = None
        self.logfile = None
        if diag_mode:
            self.console_message("toggling diagnostic report mode", "DEBUG")
            self.diag_report = DiagReport()
        if self.name is None:
            self.name = "unknown_script_harness"

    def __str__(self):
        return "script harness %s" % (self.name)

    def _time(self):
        return time.asctime(time.localtime(time.time()))

    def console_message(self, msg, level="INFO"):
        sys.stderr.write("%s: %s:(%s): %s\n" % (self._time(), level, self.name,
                                                   msg))

    def create_logdir(self):
        """Creates the directory where logging output will be contained

        @postcondition: self.logdir is not None
        @postcondition: os.isdir(self.logdir)

        """
        if not self.logdir:
            self.logdir = self.default_logdir()
        if not os.path.isdir(self.logdir):
            self.console_message("logging in directory: %s" % (self.logdir))
            os.mkdir(self.logdir)

    def default_logdir(self):
        """Returns a path to the default logging directory"""
        return os.path.join(os.getcwd(), "logs")

    def main_logfile(self):
        """Returns the full path to the main script harness logfile

        @precondition: self.logdir is not None

        """
        self.logfile = os.path.join(self.logdir, "%s.log" % (self.name))
        return self.logfile

    def setup_logging(self):
        """Sets up the main script logger

        @postcondition: self.logger is not None

        """
        log_file = self.main_logfile()
        self.create_logdir()
        self.console_message("main script log is: %s" % (log_file))
        self.logger = get_logger(category=self.name, filename=log_file)

```

```

def getLogger(self):
    """Returns the main logger for this harness

    @note: this will initialize and create the logger if it hasn't already
    been created. Client code should call harness.getLogger().
    @return: logger obj

    """
    if self.logger is None:
        self.setup_logging()
    return self.logger

def initialize(self):
    self.check_config_file()
    self.init_config()
    self.getLogger()

def check_config_file(self):
    """Checks to see that self.config_file is readable

    @raise IOError: if config_file not readable

    """
    if not os.access(self.config_file, os.R_OK):
        raise IOError("%s cannot open config file %s for reading" %
                      (self, self.config_file))

def init_config(self):
    """Initialize self.config from config_file using ConfigParser

    @see: http://docs.python.org/library/configparser.html#module-ConfigParser
    @postcondition: self.config is not None

    """
    self.config = ConfigParser.ConfigParser()
    self.config.read(self.config_file)

def config_data(self, field, section="DEFAULT"):
    """Get configuration data that was initialized from the config_file

    @return: entry for '[section] field: entry'
    @rtype: str

    """
    if self.config is None:
        raise LookupError("%s was not yet initialized" % (self))
    return self.config.get(section, field)

def cleanup(self):
    if self.diag_report is not None:
        print self.diag_report.generate()

class InteractiveSSH(object):

    def __init__(self, login, passwd, hostname, logdir, re_shell_prompt):
        self.login = login
        self.passwd = passwd
        self.hostname = hostname
        self.logdir = logdir
        self.re_shell_prompt = re_shell_prompt
        self.session = None
        self.logfile = None

```

```

        self.logfile_fh = None
        self.cmd = None

    def __str__(self):
        return "InteractiveSSH session: %s" % (self.command_string())

    def command_string(self):
        return ('ssh %s %s %s@%s' % ('-o StrictHostKeyChecking=no',
                                       '-o UserKnownHostsFile=/dev/null',
                                       self.login, self.hostname))

    def setup_logfile(self):
        if self.logfile_fh is None or self.logfile_fh.closed:
            self.logfile = os.path.join(self.logdir, "%s_ssh.log" %
                                         (self.hostname))
            self.logfile_fh = open(self.logfile, "a")

    def open(self, timeout=600):
        self.setup_logfile()
        self.session = pexpect.spawn(self.command_string(),
                                     logfile=self.logfile_fh,
                                     timeout=timeout)
        self.session.setecho(False)
        self._match_passwd()
        return self.session

    def _match_passwd(self):
        i = self.session.expect(['ssword:', self.re_shell_prompt])
        if i == 0:
            self.session.sendline(self.passwd)
            self.session.expect(self.re_shell_prompt)
        elif i == 1:
            # @todo: log message here saying we got prompt without password
            pass

    def close(self):
        if self.session:
            if not self.session.eof() and self.session.isalive():
                self.session.close()
            self.session = None
        if self.logfile_fh and not self.logfile_fh.closed:
            self.logfile_fh.close()

    class CommandSender(object):
        """Interface for sending commands/retrieving output through a remote shell

        Client code will typically do something like:

        ssh = InteractiveSSH(login, passwd, host, logdir)
        command_sender = CommandSender(ssh.open(), shell_regexp)
        msg_output = command_sender.send("cat /var/log/messages", timeout=5)

        """

        def __init__(self, pexpect_session, re_shell_prompt):
            self.session = pexpect_session
            self.re_shell_prompt = re_shell_prompt

        def send(self, cmd, timeout=60):
            cmd_output = None
            self.clear_session_buffer()
            self.session.sendline(cmd)

```

```

        self.expect_prompt(timeout=timeout)
        cmd_output = self.remove_command_echo(cmd, self.session.before)
        return cmd_output.strip()

    def clear_session_buffer(self):
        """Clear out any non-whitespace chars in buffer before next sendline"""
        self.session.timeout = 0.5
        self.session.expect(["^[\s+]$", pexpect.TIMEOUT])

    def remove_command_echo(self, command, output):
        new_output = output
        str_index = output.find(command)
        if str_index > -1:
            new_output = output[str_index + len(command):]
        return new_output

    def expect_prompt(self, timeout):
        self.session.timeout = timeout
        self.session.expect([self.re_shell_prompt])
        self.decontaminate_output()

    def decontaminate_output(self):
        """Checks session to see if output was polluted by esc chars

        esc[A shows up when the prompt + command add up to exactly the
        column width of the terminal and a second prompt is output.

        """
        if (self.session.before.find("\x1b[A") > -1 or
            self.session.after.find("\x1b[A") > -1):
            self.session.expect(["\x1b\[K\r\n"])
            self.expect_prompt(5)

class HardwareInterface(object):
    """Class to help communicate with Test Hardware objs"""

    def __init__(self, name, ip_addr, login, password, re_shell_prompt):
        self.name = name
        self.ip_addr = ip_addr
        self.login = login
        self.password = password
        self.re_shell_prompt = re_shell_prompt
        self.ssh = None
        self.sender = None

    def __str__(self):
        return "Host: %s ip addr: %s" % (self.name, self.ip_addr)

    def connect(self, logdir, diag_report):
        self.ssh = InteractiveSSH(self.login, self.password, self.ip_addr,
                                  logdir, self.re_shell_prompt)
        if diag_report is not None:
            diag_report.append("%s" % (self.ssh))
        else:
            self.sender = CommandSender(self.ssh.open(), self.re_shell_prompt)

    def send(self, command, timeout=60, diag_report=None):
        output = None
        if not self.sender and diag_report is None:
            raise RuntimeError("No connection to %s! Please call connect()" %
                               (self) + " method before attempt to send " +
                               "commands.")

```

```

        elif diag_report != None:
            diag_report.append("Sent to: %s " % (self) + command)
        else:
            cmd_output = self.sender.send(command, timeout=timeout)
            output = self._check_command_errors(command, cmd_output)
        return output

    def check_command_errors(self, command, output):
        """Sub-classes provide a custom impl that raises exc if error"""
        pass

    def _check_command_errors(self, command, output):
        """Class sub-class command error checker

        @return: output
        @raise exception: if check fails

        """
        if output:
            self.check_command_errors(command, output)
        return output

```

storage.py

```

import re
import time

from lib.harness import ScriptHarness, re_cluster_prompt, HardwareInterface
from lib.harness import InteractiveSSH, CommandSender


class ONTAPCommandError(Exception):
    pass


class StorageHarness(ScriptHarness):

    def __init__(self, config_file, logdir=None, diag_mode=False):
        ScriptHarness.__init__(self, config_file, name="config_storage",
                              logdir=logdir, diag_mode=diag_mode)
        self.cluster = None

    def initialize(self):
        """Initializes and connects to storage

        The Cluster data is initialized and we attempt to make a(n ssh)
        connection with the cluster.

        """
        ScriptHarness.initialize(self)
        self.init_cluster()
        self.init_nodes()
        self.cluster.connect(self.logdir, self.diag_report)

    def init_cluster(self):
        """Initialize the cluster from the configuration data

        @postcondition: self.cluster != None

        """
        self.cluster = Cluster(self.cluster_data("name"),
                              self.cluster_data("ip_addr"),

```

```

                self.cluster_data("login"),
                self.cluster_data("password"))
        self.logger.debug("initialized: %s" % (self.cluster))

def init_nodes(self):
    """Initialize the nodes in the cluster from configuration data

    @precondition: self.cluster != None
    @postcondition: len(self.cluster.nodes) > 0

    """
    for node_name in self.cluster_data("nodes").split():
        node_num = self.config_data("node_num", node_name)
        node = Node(node_name, node_num, self.cluster)
        self.cluster.nodes.append(node)
        self.logger.debug("initialized: %s" % (node))

def nodes(self):
    return self.cluster.nodes

def cluster_data(self, field):
    """Get cluster configuration data from the config_file

    """
    return self.config_data(field, "cluster")

def node_run(self, node, command, timeout=60):
    node_run_cmd = "node run -node %s -command %s" % (node, command)
    return self.send_cluster(node_run_cmd, timeout)

def node_run_all(self, command, timeout=60):
    node_run_cmd = "node run -node * -command %s" % (command)
    return self.send_cluster(node_run_cmd, timeout)

def send_cluster(self, command, timeout=60):
    output = None
    if self.diag_report is not None:
        self.diag_report.append(command)
    else:
        output = self.cluster.send(command, timeout)
    return output

def cleanup(self):
    self.cluster.ssh.close()
    ScriptHarness.cleanup(self)

def wait_for_last_job_complete(self, interval=60, retries=5):
    """Waits for the last job sent to the cluster shell to complete
    interval=60, retries=5

    This is useful for commands like aggregate create that run in the
    background

    @raise RuntimeError: if job doesn't complete after interval*retries
                        seconds

    """
    for cur_try in range(retries):
        cur_state = self.cluster.last_job_state()
        if cur_state in (None, "Success"):
            self.logger.info("Job %s has completed with state of %s" %
                            (self.cluster.job_id, cur_state))
            break

```

```

        else:
            self.logger.info("Attempt: %s. Job %s is in state %s..." %
                             (cur_try+1, self.cluster.job_id, cur_state) +
                             "checking state again in %s seconds." %
                             (interval))
            time.sleep(interval)
    if cur_try == retries-1 and not cur_state in (None, "Success"):
        raise RuntimeError("It appears that Job %s never completed " %
                           (self.cluster.job_id) + "after %s retries " %
                           (retries) + "of %s second intervals." %
                           (interval))

class Cluster(HardwareInterface):

    job_id_pattern = re.compile("\[Job (\d+)\]", re.M)

    def __init__(self, name, ip_addr, login, password):
        HardwareInterface.__init__(self, name, ip_addr, login, password,
                                   re_cluster_prompt)
        # Applies cluster name to regexp
        self.re_shell_prompt = self.re_shell_prompt.format(self.name)
        self.nodes = []
        self.job_id = None

    def __str__(self):
        return "cluster: %s management ip: %s" % (self.name, self.ip_addr)

    def send(self, command, timeout=60):
        cmd_output = HardwareInterface.send(self, command, timeout)
        self.set_job_id(cmd_output)
        return cmd_output

    def check_command_errors(self, cmd, cmd_output):
        """Checks command output for indication of ONTAP error

        @raise ONTAPCommandError: if errors are detected in command output
        @return: command output

        """
        re_smf_error = "(?mi)Error: (.*"
        match = re.search(re_smf_error, cmd_output, re.DOTALL)
        if match is not None:
            err_msg = match.group(1)
            raise ONTAPCommandError("ONTAP command '%s' failed with:\n%s" %
                                   (cmd, err_msg))

    def set_job_id(self, output):
        """Sets self.job_id to the last [Job $id] from cluster shell output"""
        match = Cluster.job_id_pattern.search(output)
        if match:
            self.job_id = match.group(1)
        else:
            self.job_id = None
        return output

    def last_job_state(self):
        state = None
        if self.job_id is not None:
            re_state = "(?m)%s %s (\w+)" % (self.job_id, self.name)
            re_no_entries = "There are no entries matching your query"
            job_show_output = self.send("job show -id %s -fields state" %
                                       (self.job_id))

```

```

        if re.search(re_state, job_show_output):
            state = re.search(re_state, job_show_output).group(1)
        elif re.search(re_no_entries, job_show_output):
            pass
        else:
            raise RuntimeError("Unable to determine state of Job %s from" %
                               (self.job_id) + "'job show' output:\n%s" %
                               (job_show_output))
    return state

class Node(object):

    def __init__(self, name, number, cluster):
        self.name = name
        self.number = str(number)
        self.cluster = cluster

    def __str__(self):
        return self.name

```

host.py

```

import re
import time
import pexpect
import subprocess
from StringIO import StringIO

from lib.harness import ScriptHarness, InteractiveSSH, re_unix_prompt
from lib.harness import CommandSender, HardwareInterface

class HostRebootFailure(Exception):
    pass

class HostHarness(ScriptHarness):

    def __init__(self, config_file, logdir=None, diag_mode=False,
                 name="host_harness"):
        ScriptHarness.__init__(self, config_file, name=name, logdir=logdir,
                               diag_mode=diag_mode)
        self.hosts = []

    def initialize(self):
        ScriptHarness.initialize(self)
        self.init_hosts()
        self.connect_hosts()

    def connect_hosts(self):
        for host in self.hosts:
            host.connect(self.logdir, self.diag_report)

    def init_hosts(self):
        for hostname in self.config_data("hostnames", "hosts").split():
            self.init_host(hostname)

    def init_host(self, hostname):
        new_host = LinuxHost(hostname, self.config_data("ip_addr", hostname),
                            self.config_data("login", "hosts"),
                            self.config_data("password", "hosts"))

```

```

        self.hosts.append(new_host)
        return new_host

    def reboot_hosts(self, reboot_init=None):
        for host in self.hosts:
            self.reboot_host(host, reboot_init)

    def reboot_host(self, host, reboot_init_time=None):
        host.reboot(self.diag_report)
        if reboot_init_time is None:
            reboot_init_time = float(self.config_data("post_reboot_init_time",
                                                       "hosts"))

        if self.diag_report is None:
            self.wait_for_host_to_reboot(host)
            self.console_message("%s came up after reboot successfully" %
                                 (host))
            self.logger.info("waiting %s seconds for %s to initialize" %
                             (reboot_init_time, host))
            time.sleep(reboot_init_time)
        host.connect(self.logdir, self.diag_report)

    def wait_for_host_to_reboot(self, host, ping_attempts=20, retry_sleep=30):
        """Waits for a host to come up after a reboot

        @raise HostRebootFailure: if ping_attempts have been exceeded

        """
        cur_try = 1
        start_time = time.time()
        while host.ping() != 0:
            if cur_try == ping_attempts:
                raise HostRebootFailure("%s failed to come up after reboot " %
                                         (host) + "after %d seconds." %
                                         (time.time() - start_time))

            cur_try += 1
            self.logger.info("%s not rebooted after %d seconds..." %
                            (host, time.time() - start_time) +
                            "retry again in %s seconds" % (retry_sleep))
            time.sleep(retry_sleep)
        self.logger.info("%s came up after %d seconds...reboot successful!" %
                        (host, time.time() - start_time))

    def send_host(self, host, command, timeout=120):
        output = None
        if self.diag_report is not None:
            self.diag_report.append("Sent to %s: " % (host) + command)
        else:
            output = host.send(command, timeout)
        return output

    def cleanup(self):
        for host in self.hosts:
            host.ssh.close()
        ScriptHarness.cleanup(self)

class SPCHarness(HostHarness):

    def __init__(self, config_file, logdir=None, diag_mode=False,
                 name="spc_harness"):
        HostHarness.__init__(self, config_file, name=name, logdir=logdir,
                             diag_mode=diag_mode)
        self.master = None

```

```

        self.slaves = []

    def initialize(self):
        ScriptHarness.initialize(self)
        self.init_master()
        self.init_slaves()
        self.connect_hosts()

    def init_master(self):
        master_host = self.config_data("master", "hosts")
        self.master = self.init_host(master_host)

    def init_slaves(self):
        for slave in self.config_data("slaves", "hosts").split():
            # The master can also be a slave
            if slave == self.master.name:
                self.slaves.append(self.master)
            else:
                self.slaves.append(self.init_host(slave))

    def send_lv_host(self, command, timeout=60):
        lv_host = self.get_lv_host()
        return self.send_host(lv_host, command, timeout)

    def get_lv_host(self):
        name = self.config_data("lv_create_host", "lvm")
        lv_host = None
        for host in self.hosts:
            if host.name == name:
                lv_host = host
                return lv_host
        if lv_host is None:
            raise Exception("No host obj was created with hostname %s" %
                           (name))

    def send_master(self, command, timeout=60):
        return self.send_host(self.master, command, timeout)

class LinuxHost(HardwareInterface):

    def __init__(self, name, ip_addr, login, password, sanlun=None):
        HardwareInterface.__init__(self, name, ip_addr, login, password,
                                   re_unix_prompt)
        self.dev_map = None

    def __str__(self):
        return "Linux host: %s ip addr: %s" % (self.name, self.ip_addr)

    def connect(self, logdir, diag_report):
        HardwareInterface.connect(self, logdir, diag_report)
        # start a sane shell -- should not be required
        self.send("exec sh", timeout=5, diag_report=diag_report)

    def reboot(self, diag_report):
        try:
            self.send("reboot -f -n", diag_report=diag_report)
        except (pexpect.EOF, pexpect.TIMEOUT):
            pass
        finally:
            self.ssh.close()
            self.sender = None

```

```

def ping(self):
    """Returns the result of 1 count, 1 second ping

    @return: 0 on success, <0 means ping failed

    """
    ping_command = '/bin/ping -c 1 -w 1 %s' % (self.ip_addr)
    return subprocess.call(ping_command, shell=True,
                          stdout=open('/dev/null', 'w'),
                          stderr=subprocess.STDOUT)

class SanlunDevMap(object):

    class SanDev(object):

        def __init__(self, vserver, lun_path,asu=None, lun_num=None,
                     mpath_dev=None):
            self.vserver = vserver
            self.lun_path = lun_path
            self.asu = asu
            self.lun_num = lun_num
            self.mpath_dev = None
            if mpath_dev != None:
                self.set_mpath(mpath_dev)

        def set_mpath(self, dev):
            self.mpath_dev = "/dev/mapper/%s" % (dev)

        def __str__(self):
            return "SAN Device: %s:%s Mapped to %s, %s" % (self.vserver,
                                                             self.lun_path,
                                                             self.asu,
                                                             self.mpath_dev)

        def __repr__(self):
            return str(self)

    re_ontap_lun = "ONTAP Path: ([^:]+):([\w/]+)"
    re_host_device = "Host Device: [a-z]*[\(\)]*([0-9a-f]+)[\)\]*"

    def __init__(self, sanlun_output=None):
        self.devs = []
        if sanlun_output != None:
            self.process_sanolun_output(sanolun_output)

    def __str__(self):
        str_map = 'SanlunDevMap containing SAN devices:\n'
        for dev in self:
            str_map += str(dev)
        return str_map

    def __len__(self):
        return len(self.devs)

    def __iter__(self):
        for dev in self.devs:
            yield dev

    def process_sanolun_output(self, output):
        """Processes the output of sanlun lun show -p to build self.devs"""
        output_fh = self._make_file_io(output)
        new_ontap_dev = None

```

```

        cur_dev = None
        for line in output_fh:
            if re.search(self.re_ontap_lun, line):
                match = re.search(self.re_ontap_lun, line)
                dev_path = match.group(2)
                cur_dev = SanlunDevMap.SanDev(match.group(1), dev_path)
            elif re.search(self.re_host_device, line):
                match = re.search(self.re_host_device, line)
                cur_dev.set_mpath(match.group(1))
                self.devs.append(cur_dev)
                cur_dev = None
        self.sort_devs()

    def sort_devs(self):
        """Sorts devs by lun_path so that LUNs are striped in correct order

        """
        self.devs.sort(key = lambda x: x.lun_path)

    def _make_file_io(self, output):
        ret_val = None
        if isinstance(output, str):
            ret_val = StringIO(output)
        elif isinstance(output, file):
            ret_val = output
        else:
            raise TypeError("cannot parse sanlun output of type: %s" %
                            type(output))
        return ret_val

    class LinuxVolumeGroup(object):

        def __init__(self, name, dev_list):
            self.name = name
            self.dev_list = dev_list
            self.loc = "/dev/mapper"

        def __str__(self):
            return ("VolumeGroup %s/%s consisting of devices: %s" % (self.loc,
                                                                     self.name,
                                                                     self.dev_list))

    class LinuxVolume(object):

        def __init__(self, name, volume_group, stripes, stripe_size="1024",
                     extents="100%VG"):
            self.name = name
            self.volume_group = volume_group
            self.stripes = stripes
            self.stripe_size = stripe_size
            self.extents = extents
            self.size = None

        def __str__(self):
            return ("LinuxVolume %s, with stripes=%s, stripe_size=%s, extents=%s"
                   % (self.name, self.stripes, self.stripe_size, self.extents) +
                   " and the following Volume Group:\n\t%s" % (self.volume_group))

        def path(self):
            return "/dev/mapper/%s-%s" % (self.vg_name(), self.name)

```

```
def vg_name(self):  
    return self.volume_group.name
```

APPENDIX D: SPC-1 WORKLOAD GENERATOR STORAGE COMMANDS AND PARAMETERS

ASU Pre-Fill

The content of command and parameter file, used in this benchmark to execute the required ASU pre-fill, is listed below.

```
compratio=1
hd=localhost, jvms=36
sd=default, threads=288, openflags=o_direct
sd=sd1, lun=/dev/mapper/asul_vg00-asul_lv00, size=1208236376064
sd=sd2, lun=/dev/mapper/asul_vg01-asul_lv01, size=1208236376064
sd=sd3, lun=/dev/mapper/asul_vg02-asul_lv02, size=1208236376064
sd=sd4, lun=/dev/mapper/asul_vg03-asul_lv03, size=1208236376064
sd=sd5, lun=/dev/mapper/asul_vg04-asul_lv04, size=1208236376064
sd=sd6, lun=/dev/mapper/asul_vg05-asul_lv05, size=1208236376064
sd=sd7, lun=/dev/mapper/asul_vg06-asul_lv06, size=1208236376064
sd=sd8, lun=/dev/mapper/asul_vg07-asul_lv07, size=1208236376064
sd=sd9, lun=/dev/mapper/asul_vg08-asul_lv08, size=1208236376064
sd=sd10, lun=/dev/mapper/asul_vg09-asul_lv09, size=1208236376064
sd=sd11, lun=/dev/mapper/asul_vg10-asul_lv10, size=1208236376064
sd=sd12, lun=/dev/mapper/asul_vg11-asul_lv11, size=1208236376064
sd=sd13, lun=/dev/mapper/asu2_vg12-asu2_lv12, size=1208236376064
sd=sd14, lun=/dev/mapper/asu2_vg13-asu2_lv13, size=1208236376064
sd=sd15, lun=/dev/mapper/asu2_vg14-asu2_lv14, size=1208236376064
sd=sd16, lun=/dev/mapper/asu2_vg15-asu2_lv15, size=1208236376064
sd=sd17, lun=/dev/mapper/asu2_vg16-asu2_lv16, size=1208236376064
sd=sd18, lun=/dev/mapper/asu2_vg17-asu2_lv17, size=1208236376064
sd=sd19, lun=/dev/mapper/asu2_vg18-asu2_lv18, size=1208236376064
sd=sd20, lun=/dev/mapper/asu2_vg19-asu2_lv19, size=1208236376064
sd=sd21, lun=/dev/mapper/asu2_vg20-asu2_lv20, size=1208236376064
sd=sd22, lun=/dev/mapper/asu2_vg21-asu2_lv21, size=1208236376064
sd=sd23, lun=/dev/mapper/asu2_vg22-asu2_lv22, size=1208236376064
sd=sd24, lun=/dev/mapper/asu2_vg23-asu2_lv23, size=1208236376064
sd=sd25, lun=/dev/mapper/asu3_vg24-asu3_lv24, size=268469010432
sd=sd26, lun=/dev/mapper/asu3_vg25-asu3_lv25, size=268469010432
sd=sd27, lun=/dev/mapper/asu3_vg26-asu3_lv26, size=268469010432
sd=sd28, lun=/dev/mapper/asu3_vg27-asu3_lv27, size=268469010432
sd=sd29, lun=/dev/mapper/asu3_vg28-asu3_lv28, size=268469010432
sd=sd30, lun=/dev/mapper/asu3_vg29-asu3_lv29, size=268469010432
sd=sd31, lun=/dev/mapper/asu3_vg30-asu3_lv30, size=268469010432
sd=sd32, lun=/dev/mapper/asu3_vg31-asu3_lv31, size=268469010432
sd=sd33, lun=/dev/mapper/asu3_vg32-asu3_lv32, size=268469010432
sd=sd34, lun=/dev/mapper/asu3_vg33-asu3_lv33, size=268469010432
sd=sd35, lun=/dev/mapper/asu3_vg34-asu3_lv34, size=268469010432
sd=sd36, lun=/dev/mapper/asu3_vg35-asu3_lv35, size=268469010432

wd=wd1, sd=sd1, seekpct=eof, rdpc=0, xfersize=1m
wd=wd2, sd=sd2, seekpct=eof, rdpc=0, xfersize=1m
wd=wd3, sd=sd3, seekpct=eof, rdpc=0, xfersize=1m
wd=wd4, sd=sd4, seekpct=eof, rdpc=0, xfersize=1m
wd=wd5, sd=sd5, seekpct=eof, rdpc=0, xfersize=1m
wd=wd6, sd=sd6, seekpct=eof, rdpc=0, xfersize=1m
wd=wd7, sd=sd7, seekpct=eof, rdpc=0, xfersize=1m
wd=wd8, sd=sd8, seekpct=eof, rdpc=0, xfersize=1m
wd=wd9, sd=sd9, seekpct=eof, rdpc=0, xfersize=1m
wd=wd10, sd=sd10, seekpct=eof, rdpc=0, xfersize=1m
wd=wd11, sd=sd11, seekpct=eof, rdpc=0, xfersize=1m
wd=wd12, sd=sd12, seekpct=eof, rdpc=0, xfersize=1m
```

```
wd=wd13, sd=sd13, seekpct=eof, rdpct=0, xfersize=1m
wd=wd14, sd=sd14, seekpct=eof, rdpct=0, xfersize=1m
wd=wd15, sd=sd15, seekpct=eof, rdpct=0, xfersize=1m
wd=wd16, sd=sd16, seekpct=eof, rdpct=0, xfersize=1m
wd=wd17, sd=sd17, seekpct=eof, rdpct=0, xfersize=1m
wd=wd18, sd=sd18, seekpct=eof, rdpct=0, xfersize=1m
wd=wd19, sd=sd19, seekpct=eof, rdpct=0, xfersize=1m
wd=wd20, sd=sd20, seekpct=eof, rdpct=0, xfersize=1m
wd=wd21, sd=sd21, seekpct=eof, rdpct=0, xfersize=1m
wd=wd22, sd=sd22, seekpct=eof, rdpct=0, xfersize=1m
wd=wd23, sd=sd23, seekpct=eof, rdpct=0, xfersize=1m
wd=wd24, sd=sd24, seekpct=eof, rdpct=0, xfersize=1m
wd=wd25, sd=sd25, seekpct=eof, rdpct=0, xfersize=1m
wd=wd26, sd=sd26, seekpct=eof, rdpct=0, xfersize=1m
wd=wd27, sd=sd27, seekpct=eof, rdpct=0, xfersize=1m
wd=wd28, sd=sd28, seekpct=eof, rdpct=0, xfersize=1m
wd=wd29, sd=sd29, seekpct=eof, rdpct=0, xfersize=1m
wd=wd30, sd=sd30, seekpct=eof, rdpct=0, xfersize=1m
wd=wd31, sd=sd31, seekpct=eof, rdpct=0, xfersize=1m
wd=wd32, sd=sd32, seekpct=eof, rdpct=0, xfersize=1m
wd=wd33, sd=sd33, seekpct=eof, rdpct=0, xfersize=1m
wd=wd34, sd=sd34, seekpct=eof, rdpct=0, xfersize=1m
wd=wd35, sd=sd35, seekpct=eof, rdpct=0, xfersize=1m
wd=wd36, sd=sd36, seekpct=eof, rdpct=0, xfersize=1m
rd=rd1, wd=wd1, elapsed=72h, interval=60, iorate=max
rd=rd2, wd=wd2, elapsed=72h, interval=60, iorate=max
rd=rd3, wd=wd3, elapsed=72h, interval=60, iorate=max
rd=rd4, wd=wd4, elapsed=72h, interval=60, iorate=max
rd=rd5, wd=wd5, elapsed=72h, interval=60, iorate=max
rd=rd6, wd=wd6, elapsed=72h, interval=60, iorate=max
rd=rd7, wd=wd7, elapsed=72h, interval=60, iorate=max
rd=rd8, wd=wd8, elapsed=72h, interval=60, iorate=max
rd=rd9, wd=wd9, elapsed=72h, interval=60, iorate=max
rd=rd10, wd=wd10, elapsed=72h, interval=60, iorate=max
rd=rd11, wd=wd11, elapsed=72h, interval=60, iorate=max
rd=rd12, wd=wd12, elapsed=72h, interval=60, iorate=max
rd=rd13, wd=wd13, elapsed=72h, interval=60, iorate=max
rd=rd14, wd=wd14, elapsed=72h, interval=60, iorate=max
rd=rd15, wd=wd15, elapsed=72h, interval=60, iorate=max
rd=rd16, wd=wd16, elapsed=72h, interval=60, iorate=max
rd=rd17, wd=wd17, elapsed=72h, interval=60, iorate=max
rd=rd18, wd=wd18, elapsed=72h, interval=60, iorate=max
rd=rd19, wd=wd19, elapsed=72h, interval=60, iorate=max
rd=rd20, wd=wd20, elapsed=72h, interval=60, iorate=max
rd=rd21, wd=wd21, elapsed=72h, interval=60, iorate=max
rd=rd22, wd=wd22, elapsed=72h, interval=60, iorate=max
rd=rd23, wd=wd23, elapsed=72h, interval=60, iorate=max
rd=rd24, wd=wd24, elapsed=72h, interval=60, iorate=max
rd=rd25, wd=wd25, elapsed=72h, interval=60, iorate=max
rd=rd26, wd=wd26, elapsed=72h, interval=60, iorate=max
rd=rd27, wd=wd27, elapsed=72h, interval=60, iorate=max
rd=rd28, wd=wd28, elapsed=72h, interval=60, iorate=max
rd=rd29, wd=wd29, elapsed=72h, interval=60, iorate=max
rd=rd30, wd=wd30, elapsed=72h, interval=60, iorate=max
rd=rd31, wd=wd31, elapsed=72h, interval=60, iorate=max
rd=rd32, wd=wd32, elapsed=72h, interval=60, iorate=max
rd=rd33, wd=wd33, elapsed=72h, interval=60, iorate=max
rd=rd34, wd=wd34, elapsed=72h, interval=60, iorate=max
rd=rd35, wd=wd35, elapsed=72h, interval=60, iorate=max
rd=rd36, wd=wd36, elapsed=72h, interval=60, iorate=max
```

Primary Metrics and Repeatability Tests

The content of SPC-1 Workload Generator command and parameter file used in this benchmark to execute the Primary Metrics (*Sustainability Test Phase, IOPS Test Phase, and Response Time Ramp Test Phase*) Test and Repeatability Test (*Repeatability Test Phase 1 and Repeatability Test Phase 2*), which used multiple Host Systems, is listed below.

```
javaparms="-Xms1536m -Xmx2048m -Xss256k"
host=master
slaves=(h1_slave1,h1_slave2,h1_slave3,h1_slave4,h1_slave5,h1_slave6,h1_slave7,h1_slave8,h1_slave9,h1_slave10,h1_slave11,h1_slave12,h1_slave13,h2_slave1,h2_slave2,h2_slave3,h2_slave4,h2_slave5,h2_slave6,h2_slave7,h2_slave8,h2_slave9,h2_slave10,h2_slave11,h2_slave12,h2_slave13,h3_slave1,h3_slave2,h3_slave3,h3_slave4,h3_slave5,h3_slave6,h3_slave7,h3_slave8,h3_slave9,h3_slave10,h3_slave11,h3_slave12,h3_slave13,h4_slave1,h4_slave2,h4_slave3,h4_slave4,h4_slave5,h4_slave6,h4_slave7,h4_slave8,h4_slave9,h4_slave10,h4_slave11,h4_slave12,h4_slave13)
sd=asu1_100,lun=/dev/mapper/asu1_vg00-asu1_lv00,size=1208236376064
sd=asu1_101,lun=/dev/mapper/asu1_vg01-asu1_lv01,size=1208236376064
sd=asu1_102,lun=/dev/mapper/asu1_vg02-asu1_lv02,size=1208236376064
sd=asu1_103,lun=/dev/mapper/asu1_vg03-asu1_lv03,size=1208236376064
sd=asu1_104,lun=/dev/mapper/asu1_vg04-asu1_lv04,size=1208236376064
sd=asu1_105,lun=/dev/mapper/asu1_vg05-asu1_lv05,size=1208236376064
sd=asu1_106,lun=/dev/mapper/asu1_vg06-asu1_lv06,size=1208236376064
sd=asu1_107,lun=/dev/mapper/asu1_vg07-asu1_lv07,size=1208236376064
sd=asu1_108,lun=/dev/mapper/asu1_vg08-asu1_lv08,size=1208236376064
sd=asu1_109,lun=/dev/mapper/asu1_vg09-asu1_lv09,size=1208236376064
sd=asu1_110,lun=/dev/mapper/asu1_vg10-asu1_lv10,size=1208236376064
sd=asu1_111,lun=/dev/mapper/asu1_vg11-asu1_lv11,size=1208236376064
sd=asu2_112,lun=/dev/mapper/asu2_vg12-asu2_lv12,size=1208236376064
sd=asu2_113,lun=/dev/mapper/asu2_vg13-asu2_lv13,size=1208236376064
sd=asu2_114,lun=/dev/mapper/asu2_vg14-asu2_lv14,size=1208236376064
sd=asu2_115,lun=/dev/mapper/asu2_vg15-asu2_lv15,size=1208236376064
sd=asu2_116,lun=/dev/mapper/asu2_vg16-asu2_lv16,size=1208236376064
sd=asu2_117,lun=/dev/mapper/asu2_vg17-asu2_lv17,size=1208236376064
sd=asu2_118,lun=/dev/mapper/asu2_vg18-asu2_lv18,size=1208236376064
sd=asu2_119,lun=/dev/mapper/asu2_vg19-asu2_lv19,size=1208236376064
sd=asu2_120,lun=/dev/mapper/asu2_vg20-asu2_lv20,size=1208236376064
sd=asu2_121,lun=/dev/mapper/asu2_vg21-asu2_lv21,size=1208236376064
sd=asu2_122,lun=/dev/mapper/asu2_vg22-asu2_lv22,size=1208236376064
sd=asu2_123,lun=/dev/mapper/asu2_vg23-asu2_lv23,size=1208236376064
sd=asu3_124,lun=/dev/mapper/asu3_vg24-asu3_lv24,size=268469010432
sd=asu3_125,lun=/dev/mapper/asu3_vg25-asu3_lv25,size=268469010432
sd=asu3_126,lun=/dev/mapper/asu3_vg26-asu3_lv26,size=268469010432
sd=asu3_127,lun=/dev/mapper/asu3_vg27-asu3_lv27,size=268469010432
sd=asu3_128,lun=/dev/mapper/asu3_vg28-asu3_lv28,size=268469010432
sd=asu3_129,lun=/dev/mapper/asu3_vg29-asu3_lv29,size=268469010432
sd=asu3_130,lun=/dev/mapper/asu3_vg30-asu3_lv30,size=268469010432
sd=asu3_131,lun=/dev/mapper/asu3_vg31-asu3_lv31,size=268469010432
sd=asu3_132,lun=/dev/mapper/asu3_vg32-asu3_lv32,size=268469010432
sd=asu3_133,lun=/dev/mapper/asu3_vg33-asu3_lv33,size=268469010432
sd=asu3_134,lun=/dev/mapper/asu3_vg34-asu3_lv34,size=268469010432
sd=asu3_135,lun=/dev/mapper/asu3_vg35-asu3_lv35,size=268469010432
```

SPC-1 Persistence Test

The content of SPC-1 Workload Generator command and parameter file used in this benchmark to execute the Persistence Test, which used a single Host System, is listed below.

```
javaparms="-Xms1536m -Xmx4096m -Xss256k"
sd=asul_100,lun=/dev/mapper/asul_vg00-asul_lv00,size=1208236376064
sd=asul_101,lun=/dev/mapper/asul_vg01-asul_lv01,size=1208236376064
sd=asul_102,lun=/dev/mapper/asul_vg02-asul_lv02,size=1208236376064
sd=asul_103,lun=/dev/mapper/asul_vg03-asul_lv03,size=1208236376064
sd=asul_104,lun=/dev/mapper/asul_vg04-asul_lv04,size=1208236376064
sd=asul_105,lun=/dev/mapper/asul_vg05-asul_lv05,size=1208236376064
sd=asul_106,lun=/dev/mapper/asul_vg06-asul_lv06,size=1208236376064
sd=asul_107,lun=/dev/mapper/asul_vg07-asul_lv07,size=1208236376064
sd=asul_108,lun=/dev/mapper/asul_vg08-asul_lv08,size=1208236376064
sd=asul_109,lun=/dev/mapper/asul_vg09-asul_lv09,size=1208236376064
sd=asul_110,lun=/dev/mapper/asul_vg10-asul_lv10,size=1208236376064
sd=asul_111,lun=/dev/mapper/asul_vg11-asul_lv11,size=1208236376064
sd=asu2_112,lun=/dev/mapper/asu2_vg12-asu2_lv12,size=1208236376064
sd=asu2_113,lun=/dev/mapper/asu2_vg13-asu2_lv13,size=1208236376064
sd=asu2_114,lun=/dev/mapper/asu2_vg14-asu2_lv14,size=1208236376064
sd=asu2_115,lun=/dev/mapper/asu2_vg15-asu2_lv15,size=1208236376064
sd=asu2_116,lun=/dev/mapper/asu2_vg16-asu2_lv16,size=1208236376064
sd=asu2_117,lun=/dev/mapper/asu2_vg17-asu2_lv17,size=1208236376064
sd=asu2_118,lun=/dev/mapper/asu2_vg18-asu2_lv18,size=1208236376064
sd=asu2_119,lun=/dev/mapper/asu2_vg19-asu2_lv19,size=1208236376064
sd=asu2_120,lun=/dev/mapper/asu2_vg20-asu2_lv20,size=1208236376064
sd=asu2_121,lun=/dev/mapper/asu2_vg21-asu2_lv21,size=1208236376064
sd=asu2_122,lun=/dev/mapper/asu2_vg22-asu2_lv22,size=1208236376064
sd=asu2_123,lun=/dev/mapper/asu2_vg23-asu2_lv23,size=1208236376064
sd=asu3_124,lun=/dev/mapper/asu3_vg24-asu3_lv24,size=268469010432
sd=asu3_125,lun=/dev/mapper/asu3_vg25-asu3_lv25,size=268469010432
sd=asu3_126,lun=/dev/mapper/asu3_vg26-asu3_lv26,size=268469010432
sd=asu3_127,lun=/dev/mapper/asu3_vg27-asu3_lv27,size=268469010432
sd=asu3_128,lun=/dev/mapper/asu3_vg28-asu3_lv28,size=268469010432
sd=asu3_129,lun=/dev/mapper/asu3_vg29-asu3_lv29,size=268469010432
sd=asu3_130,lun=/dev/mapper/asu3_vg30-asu3_lv30,size=268469010432
sd=asu3_131,lun=/dev/mapper/asu3_vg31-asu3_lv31,size=268469010432
sd=asu3_132,lun=/dev/mapper/asu3_vg32-asu3_lv32,size=268469010432
sd=asu3_133,lun=/dev/mapper/asu3_vg33-asu3_lv33,size=268469010432
sd=asu3_134,lun=/dev/mapper/asu3_vg34-asu3_lv34,size=268469010432
sd=asu3_135,lun=/dev/mapper/asu3_vg35-asu3_lv35,size=268469010432
```

Slave JVMs

Each Slave JVM was invoked with a command and parameter file similar to the example listed below. The only difference in each file was **host** parameter value, which was unique to each Slave JVM, e.g. **h1_slave1...h4_slave13**.

```
javaparms="-Xms1536m -Xmx2048m -Xss256k"
master=10.63.173.44
host=h1_slave1
sd=asu1_100,lun=/dev/mapper/asu1_vg00-asu1_lv00,size=1208236376064
sd=asu1_101,lun=/dev/mapper/asu1_vg01-asu1_lv01,size=1208236376064
sd=asu1_102,lun=/dev/mapper/asu1_vg02-asu1_lv02,size=1208236376064
sd=asu1_103,lun=/dev/mapper/asu1_vg03-asu1_lv03,size=1208236376064
sd=asu1_104,lun=/dev/mapper/asu1_vg04-asu1_lv04,size=1208236376064
sd=asu1_105,lun=/dev/mapper/asu1_vg05-asu1_lv05,size=1208236376064
sd=asu1_106,lun=/dev/mapper/asu1_vg06-asu1_lv06,size=1208236376064
sd=asu1_107,lun=/dev/mapper/asu1_vg07-asu1_lv07,size=1208236376064
sd=asu1_108,lun=/dev/mapper/asu1_vg08-asu1_lv08,size=1208236376064
sd=asu1_109,lun=/dev/mapper/asu1_vg09-asu1_lv09,size=1208236376064
sd=asu1_110,lun=/dev/mapper/asu1_vg10-asu1_lv10,size=1208236376064
sd=asu1_111,lun=/dev/mapper/asu1_vg11-asu1_lv11,size=1208236376064
sd=asu2_112,lun=/dev/mapper/asu2_vg12-asu2_lv12,size=1208236376064
sd=asu2_113,lun=/dev/mapper/asu2_vg13-asu2_lv13,size=1208236376064
sd=asu2_114,lun=/dev/mapper/asu2_vg14-asu2_lv14,size=1208236376064
sd=asu2_115,lun=/dev/mapper/asu2_vg15-asu2_lv15,size=1208236376064
sd=asu2_116,lun=/dev/mapper/asu2_vg16-asu2_lv16,size=1208236376064
sd=asu2_117,lun=/dev/mapper/asu2_vg17-asu2_lv17,size=1208236376064
sd=asu2_118,lun=/dev/mapper/asu2_vg18-asu2_lv18,size=1208236376064
sd=asu2_119,lun=/dev/mapper/asu2_vg19-asu2_lv19,size=1208236376064
sd=asu2_120,lun=/dev/mapper/asu2_vg20-asu2_lv20,size=1208236376064
sd=asu2_121,lun=/dev/mapper/asu2_vg21-asu2_lv21,size=1208236376064
sd=asu2_122,lun=/dev/mapper/asu2_vg22-asu2_lv22,size=1208236376064
sd=asu2_123,lun=/dev/mapper/asu2_vg23-asu2_lv23,size=1208236376064
sd=asu3_124,lun=/dev/mapper/asu3_vg24-asu3_lv24,size=268469010432
sd=asu3_125,lun=/dev/mapper/asu3_vg25-asu3_lv25,size=268469010432
sd=asu3_126,lun=/dev/mapper/asu3_vg26-asu3_lv26,size=268469010432
sd=asu3_127,lun=/dev/mapper/asu3_vg27-asu3_lv27,size=268469010432
sd=asu3_128,lun=/dev/mapper/asu3_vg28-asu3_lv28,size=268469010432
sd=asu3_129,lun=/dev/mapper/asu3_vg29-asu3_lv29,size=268469010432
sd=asu3_130,lun=/dev/mapper/asu3_vg30-asu3_lv30,size=268469010432
sd=asu3_131,lun=/dev/mapper/asu3_vg31-asu3_lv31,size=268469010432
sd=asu3_132,lun=/dev/mapper/asu3_vg32-asu3_lv32,size=268469010432
sd=asu3_133,lun=/dev/mapper/asu3_vg33-asu3_lv33,size=268469010432
sd=asu3_134,lun=/dev/mapper/asu3_vg34-asu3_lv34,size=268469010432
sd=asu3_135,lun=/dev/mapper/asu3_vg35-asu3_lv35,size=268469010432
```

APPENDIX E: SPC-1 WORKLOAD GENERATOR INPUT PARAMETERS

The ‘master’ script, **spc1.start1.sh**, was invoked to execute the required ASU pre-fill, the Primary Metrics Test (*Sustainability Test Phase, IOPS Test Phase, and Response Time Ramp Test Phase*), the Repeatability Test (*Repeatability Test Phase 1 and Repeatability Test Phase 2*), and SPC-1 Persistence Test Run 1 in an uninterrupted sequence.

The **spc1.start2.sh** script was executed after the required TSC power off/power on cycle completed to execute SPC-1 Persistence Test Run 2.

The **spc1.start1.sh** and **spc1.start2.sh** scripts also included the appropriate commands to capture the detailed TSC profile listings required for a Remote Audit.

spc1.start1.sh

```
#!/bin/bash

source /SPC1RunDir/spc_env.sh

# Collect Audit Inventory Data before SPC-1 Audit runs
/SPC1RunDir/collect.before.audit.inventory.sh

# Vdbench luns pre-fill run
cd /SPC1RunDir/vdbench
/SPC1RunDir/vdbench/vdbench -f /SPC1RunDir/vdbench/prefill_luns.txt -o
/SPC1RunDir/vdbench/luns.prefill

# SPC-1 benchmark Metrics Test
cd /SPC1RunDir
/SPC1RunDir/start_all_slaves_jvms.sh
java -Xms1536m -Xmx2048m -Xss256k metrics -b 1721 -t 28800 -r 600 -s 10800:180

# SPC-1 benchmark Repeatability1 Test
java -Xms1536m -Xmx2048m -Xss256k repeat1 -b 1721

# SPC-1 benchmark Repeatability2 Test
java -Xms1536m -Xmx2048m -Xss256k repeat2 -b 1721
/SPC1RunDir/stop_all_slaves_jvms.sh

# Prepare spc1.cfg file for Persistence1 Test
/bin/cp /SPC1RunDir/spc1.cfg /SPC1RunDir/spc1.cfg.multihost
/bin/cp /SPC1RunDir/spc1.cfg.persist /SPC1RunDir/spc1.cfg

# SPC-1 benchmark Persistence1 Test
java -Xms1536m -Xmx2048m -Xss256k persist1 -b 1721
```

start_all_slaves_jvms.sh

This script was invoked from [spc1.start1.sh](#) to start all of the Slave JVMs on each of the four Host Systems prior to execution of the Test Runs.

```
#!/bin/bash
/usr/bin/ssh stlx300s7-120 'cd /SPC1RunDir; /SPC1RunDir/launch_host1_slaves.sh' &
sleep 10
/usr/bin/ssh stlx300s7-121 'cd /SPC1RunDir; /SPC1RunDir/launch_host2_slaves.sh' &
sleep 10
/usr/bin/ssh stlx300s7-122 'cd /SPC1RunDir; /SPC1RunDir/launch_host3_slaves.sh' &
sleep 10
/usr/bin/ssh stlx300s7-123 'cd /SPC1RunDir; /SPC1RunDir/launch_host4_slaves.sh' &
sleep 10
```

launch_host1_slaves.sh

```
#!/bin/bash
export CLASSPATH=/SPC1RunDir
export LD_LIBRARY_PATH=/SPC1RunDir
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave1.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave2.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave3.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave4.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave5.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave6.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave7.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave8.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave9.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave10.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave11.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave12.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h1_slave13.txt 2>&1 &
```

launch_host2_slaves.sh

```
#!/bin/bash
export CLASSPATH=/SPC1RunDir
export LD_LIBRARY_PATH=/SPC1RunDir
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave1.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave2.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave3.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave4.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave5.txt 2>&1 &
sleep 1
```

```
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave6.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave7.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave8.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave9.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave10.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave11.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave12.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h2_slave13.txt 2>&1 &
```

launch_host3_slaves.sh

```
#!/bin/bash
export CLASSPATH=/SPC1RunDir
export LD_LIBRARY_PATH=/SPC1RunDir
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave1.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave2.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave3.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave4.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave5.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave6.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave7.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave8.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave9.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave10.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave11.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave12.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h3_slave13.txt 2>&1 &
```

launch_host4_slaves.sh

```
#!/bin/bash
export CLASSPATH=/SPC1RunDir
export LD_LIBRARY_PATH=/SPC1RunDir
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h4_slave1.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h4_slave2.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h4_slave3.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h4_slave4.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spcl -f /SPC1RunDir/h4_slave5.txt 2>&1 &
sleep 1
```

```
java -Xms1536m -Xmx2048m -Xss256k spc1 -f /SPC1RunDir/h4_slave6.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spc1 -f /SPC1RunDir/h4_slave7.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spc1 -f /SPC1RunDir/h4_slave8.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spc1 -f /SPC1RunDir/h4_slave9.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spc1 -f /SPC1RunDir/h4_slave10.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spc1 -f /SPC1RunDir/h4_slave11.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spc1 -f /SPC1RunDir/h4_slave12.txt 2>&1 &
sleep 1
java -Xms1536m -Xmx2048m -Xss256k spc1 -f /SPC1RunDir/h4_slave13.txt 2>&1 &
```

spc1.start2.sh

```
#!/bin/bash

source /SPC1RunDir/spc_env.sh
#export CLASSPATH=/SPC1RunDir
#export LD_LIBRARY_PATH=/SPC1RunDir

# Reboot and power cycle here and then continue as follows

# SPC-1 benchmark Persistence2 Test
cd /SPC1RunDir
java -Xms1536m -Xmx2048m -Xss256k persist2
sleep 30
# Collect Audit Inventory Data before SPC-1 Audit runs
/SPC1RunDir/collect.after.audit.inventory.sh
/bin/cp /SPC1RunDir/spc1.cfg.multihost /SPC1RunDir/spc1.cfg
```