intel + VAST

BREAKING SCALABILITY AND CONCURRENCY BARRIERS WITH VAST DATA

Benchmarking Guide to Maximize Client Performance (with NFS & S3)

JANUARY 2022

REVISION HISTORY

Revision Number	Description	Date
1.0	Initial Release	January 2022

KEY CONTACTS

Name	Title	Email
Chunhong Mao	Solutions Architect	chunhong.mao@intel.com
Frank Ober	Solutions Architect, Principal	frank.ober@intel.com
Allison Goodman	Director of IOG Solutions Architecture	allison.goodman@intel.com
Andy Pernsteiner	Field CTO, Manager	andy@vastdata.com
Sven Breuner	Field CTO	sven.breuner@vastdata.com
Rob Mallory	Field CTO	rob@vastdata.com

Executive Summary

VAST Data's Universal Storage is a next-generation, scale-out file and object storage solution that breaks decades of storage tradeoffs between performance, capacity, and cost. Through game-changing storage innovations that lower the cost of flash, VAST Data makes - for the very first time - flash affordable for all applications, from the highest performance datasets to the largest data archives.

To accomplish this monumental achievement, VAST Data pioneered a new scale-out storage architecture called DASE (Disaggregated Shared Everything). DASE is powered by three new technologies including low-cost hyperscale (QLC) flash, persistent storage class memory (such as high-performance Intel® Optane™ Technology),, and low-latency NVMe-over-Fabrics networking. Scalable to exabytes, this new architecture delivers linear performance scale by eliminating east-west communication across storage CPUs, while enabling high-speed random I/O access to billions of small files.

To demonstrate the scaling benefits of the DASE architecture, along with random I/O access benefits of an all-flash storage system, Intel and VAST Data jointly conducted several key benchmarking tests to simulate various degrees of concurrency and parallelism across 10's and 100's of simultaneous clients.

Highlights and key findings from the testing include:

- Near-perfect linear client performance scaling with 100GbE clients (Figure 15) and 10GbE clients (Figure 16).
- Achieved 117GB/s of read bandwidth (97% of theoretical max) with only 14 (100GbE) clients (Table 14).
- Sequential and random I/O access exhibit performance parity (+/- 2%) using same block size (4K) (Table 12).
- S3 read performance delivered 110GB/s (94% of NFS performance using same file/object size) (Table 13).

The remainder of this paper provides detailed system and network configurations used, multiple testing methodologies and tools, and step-by-step instructions to obtain the results outlined.

VAST Data is powering some of the world's largest government labs, animation studios, seismic processing centers, bioinformatics pipelines, hedge fund research grids, AI efforts and more. Customers are reporting more performance from VAST than what they've seen from their legacy parallel file systems and any other NAS options. VAST is simple, affordable, and applications thrive once they've made the move to flash.

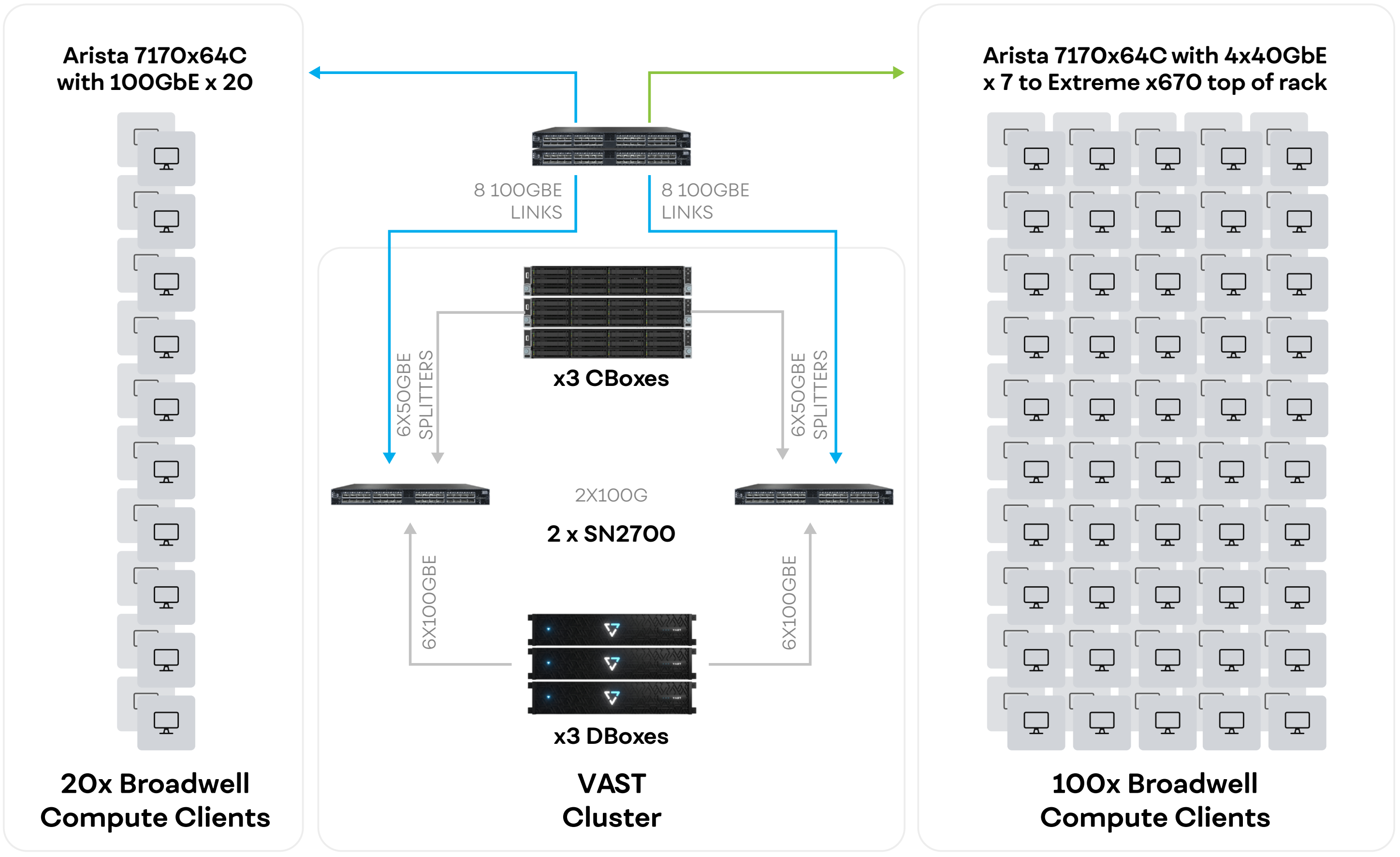
To learn more about how **Universal Storage** can help simplify your exascale initiatives, reach out to us at hello@vastdata.com

Table of Contents

1) Hardware Environment

1.1 Block Diagram

The following diagram depicts the top level of logical connectivity between the 100 client cluster and the VAST Data 3x3 cluster.



Arista 7170x64C with 100GbE x 20
Arista 7170x64C with 4x40GbE x 7 to Extreme x670 top of rack
8 100GBE LINKS
8 100GBE LINKS
x3 CBoxes
6X50GBE SPLITTERS
6X50GBE SPLITTERS
2X100G
2 x SN2700
6X100GBE
6X100GBE
x3 DBoxes
VAST Cluster
20x Broadwell Compute Clients
100x Broadwell Compute Clients

Figure 1: Block Diagram Logical View for Scale Testing

All 100 physical clients are outfitted with 10GbE network cards, while 20 of them have also been equipped with a second 100GbE NIC card.

A shell script-based test harness has been developed to build different test cases easily and execute the automated tests at scale. The test harness uses synthetic benchmarking tools FIO and Elbencho.

By using different subnets for the 10GbE and 100GbE networks, the client cluster can be configured to blast traffic to the VAST Data cluster in two different topologies via the test harness:

a) 100 clients with 10GbE networking

b) 20 clients with 100GbE networking

As both FIO and Elbencho will be running in Client/Server mode, one additional system is needed on the 10GbE network to execute the test harness.

Note that VIP pools created on the VAST Data cluster for 10Gb and 100Gb testing match the client 10Gb and 100Gb subnets respectively, but both need to be different to VAST Data's VMS. The table below shows an example of the IP assignments.

Table 1: 100GbE and 10GbE Subnets and VIP Assignments

Systems	Subnet	IP Examples
Client cluster 10GbE clients + head node	10.A.B.x	10.A.B.1~10.A.B.101
VAST Data VIP pool for 10G Testing	10.A.B.x	10.A.B.110~10.A.B.157
VAST Data VMS	10.A.C.y	10.A.C.D
Client cluster 100GbE clients	192.168.B.z	192.168.B.1~192.168.B.20
VAST Data VIP pool for 100GeE Testing	192.168.B.z	192.168.B.110~192.168.B.157

1.2 VAST Data 3x3 Cluster Hardware Configuration

The 3x3 VAST Data cluster is composed of:

1) 3 CBox enclosures, each enclosure contains:

   A. a quad node chassis

   B. 1 dual-port Mellanox MT27800 QSFP28 100GbE primary NIC per node for fast data path

   C. 1 1GbE/10GbE NIC per node for management

2) 3 DBox enclosures, each enclosure contains:

   A. a dual node chassis

   B. 2 dual-port Mellanox MT27800 QSFP28 100GbE primary NIC per node for fast data path

   C. 2 GbE NIC per node for management

3) 2 Mellanox SN2700 VAST Data switches, connecting the CBox enclosures and DBox enclosures

Each DBox provides 592TB of usable storage, for a total capacity of 1.775PB (before data reduction).

1.2.1 CBOX AND DBOX CONFIGURATIONS

Table 2: CBox and DBox Configurations

Component	**CBOX Enclosure Configurations**	**DBOX Enclosure Configuration**
Platform	Intel® Server Board S2600BPB (Cascade Lake)	Viking Enterprise Solutions (VES) NSS2560
# of Nodes	3x4	3x2
# of Sockets	2	2
CPU	Intel® Xeon® Silver 4216 CPU @ 2.10GHz	Intel® Xeon® CPU E5-2630 v4 @ 2.20GHz
Cores per socket / threads per socket	16/32	10/20
Microcode	0x5002f01	0xb000038
Intel® Hyper-Threading Technology	ON	ON
Intel® Turbo Boost Technology	OFF	ON
BIOS version	SE5C620.8 6B.02.01.0008.031920191559	11.03
System DDR memory configuration: slots/ capacity/speed	8 slots/32 GB/2400 MT/s (Micron) 8 slots/ No DRAM modules	8 slots/8 GB/2933 MT/s (Micron) 16 slots/ No DRAM modules
Total memory per node (All DRAM)	256G	64G
Storage	Boot: Intel SSD D3-S4510 960G	6 x Intel® Optane™ SSD P5800 1.6T 22 x Intel SSD D5-P4326 15.36T
Network Interface Card	1 x Mellanox® MT27800 ConnectX®-5 VPI Dual-Port 100 Gigabit Ethernet Adapter / Node 1 x Ethernet Controller 10G X550T / Node	2 x Mellanox® MT27800 ConnectX®-5 VPI Dual-Port 100 Gigabit Ethernet Adapter / Node 2 x Intel® 82574L Gigabit Ethernet / Node
NIC firmware	MT27800: 16.26.4012 (MT_0000000008) X550T: 0x80000a42, 1.1767.0	MT27800: 16.27.6008 (MT_0000000207-1) 82574L: 2.1-3

1.2.2 MELLANOX® SN2700 SWITCH

22 out of the 32-port 100GbE Mellanox® SN2700 switch are being allocated for data path connectivity.

Table 3: Mellanox® SN2700 Port Allocation

Number of Ports	Speed / Port	Connectivity
8	100GbE	Arista
6	100GbE	CBox (12 x 50GbE)
6	100GbE	DBox
2	100GbE	ISL (inter-switch link)

1.3 100 Clients With 10GbE Networking

This section outlines the hardware details used to set up the benchmarking environment of 100 servers with 10GbE networking in-between the client cluster and the VAST Data cluster.

1.3.1 NETWORK CONNECTIVITY

The diagram below depicts the network connectivity between client cluster and VAST Data cluster.

A. 100 clients are connected to 7 Extreme x760v switches, with up to 16 clients / per switch to ensure each client's 10GbE networking interface speed can be fully utilized.

B. Each of the 7 Extreme switches is connected to the Arista 7170-64C switch through a 4 x 40Gbit uplink.

C. The Arista 7170-64C is also connected to Mellanox SN2700 within the VAST Data cluster through 16 x 100Gbit links.

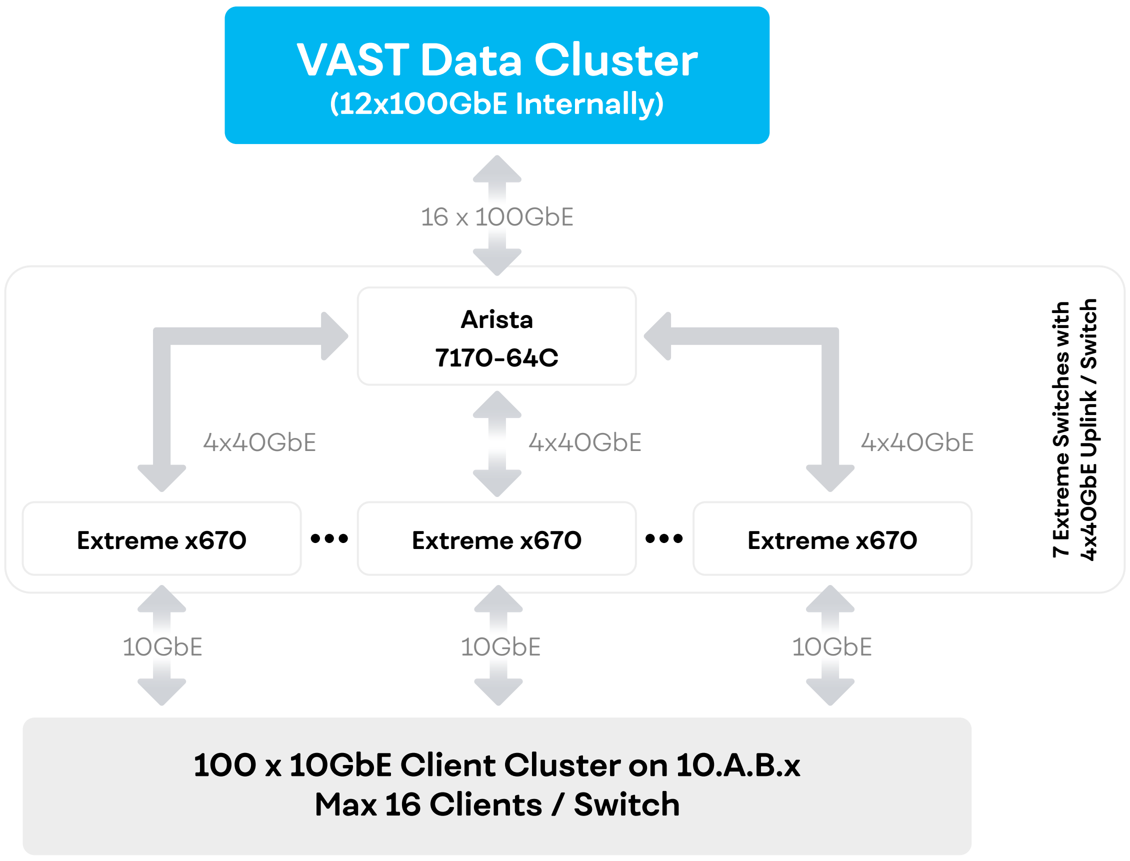


Figure 2: 100 x 10GbE Client Cluster and VAST Data Cluster Connectivity

Note that the head node is not shown in the diagram. Although the head node doesn't blast IO to VAST Data cluster itself, it does need to mount the VIPs as well for test harness utilities. So one additional 10GbE node is used for this purpose.

1.3.2 ARISTA 7170-64C SWITCH

One Arista 7170-64C switch is used as the hub in-between the VAST Data cluster and the client cluster. Please refer to https://www.arista.com/assets/data/pdf/Datasheets/7170-Datasheet.pdf for the datasheet specifications of the Arista 7170-64C switch.

In summary, it has:

A. 64 x 40/100GbE QSFP100 ports

B. 2 x 1GbE SFP+ ports for additional high speed management

Port allocations of the Arista 7170-64C is indicated in the table below:

Table 4: Arista® 7170-64C Port Allocation

Number of Ports	Speed / Port	Connectivity	Total BW Allowed at Arista 7170-64C
28	40GbE (5GB)	7 Extreme Switches	28 x 5 GB = 140 GB
20	100GbE (12.5GB)	20 clients with 100GbE	20 x 12.5 GB = 250 GB
16	100GbE (12.5GB)	3 x 3 VAST Data cluster	16 x 12.5 GB = 200 GB

1.3.3 EXTREME X670V SWITCH

Extreme x670v switch has:

A. 48 ports, 10Gbit / port

B. 4 QSFP+ ports, 40Gbit / port for uplinks

Here we use the 10Gbit ports to connect to client systems, and the QSFP+ ports to connect to the Arista switch.

- Max Bandwidth of the 100 10Gbit client systems: 100 x 10Gbit = 1000 Gbit = 125GB
- Max Bandwidth of a single Extreme Switch: 4 x 40Gbit = 160Gbit = 20GB

Important: although each Extreme x670v switch has 48 ports, the maximum ports that can be used to connect to client systems shall be no more than 16 because each Extreme x670 switch can only supply 160Gbit of uplink bandwidth using the 4 QSFP+ ports. As a result, 7 Extreme switches were required for this exercise.

1.4 20 Clients With 100GbE Networking

This section outlines the hardware details used to set up the benchmarking environment of 20 client systems with 100GbE networking in-between the clients and the VAST Data cluster.

1.4.1 NETWORK CONNECTIVITY

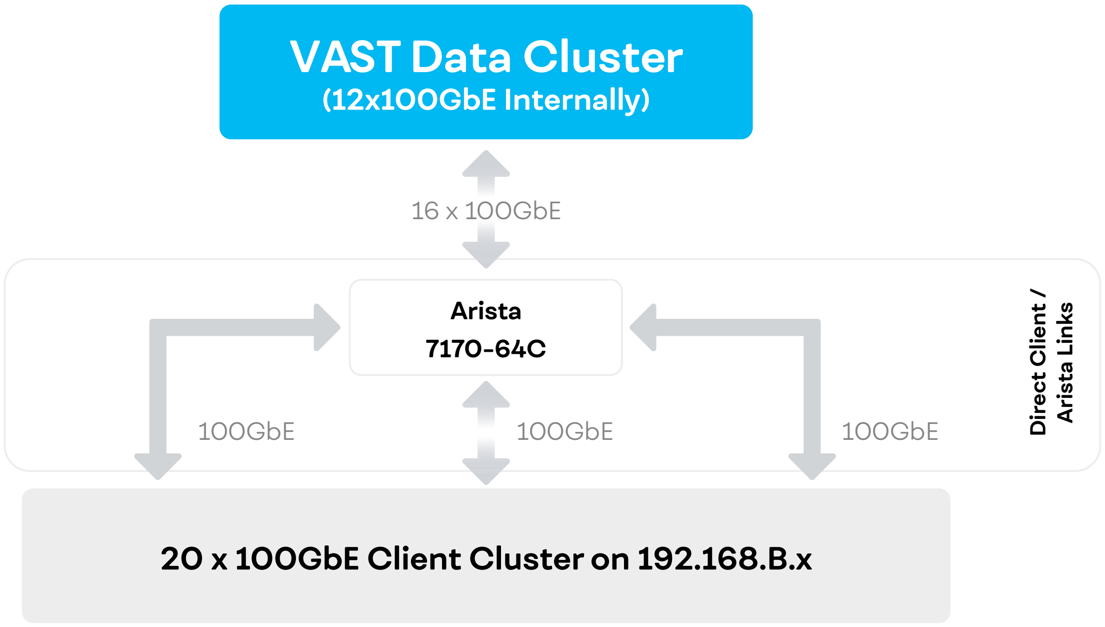


Figure 3: 20 x 100GbE Client Cluster and VAST DATA Cluster Connectivity

1.4.2 ARISTA® 7170-64C SWITCH

The 20 100GbE client systems are connected to Arista 7170-64C 20 100GbE ports directly.

1.4.3 EXTREME® X670V SWITCH

Unlike the 10GbE networking configuration, there is no Extreme x670v in-between the 20 100GbE clients and the Arista 7170-64C.

1.5 Client System Configurations

The table below shows the 10GbE client and 100GbE capable client system configurations. Note that the major difference between the two is simply that the 100GbE capable ones have dual NICs to support both 100GbE and 10GbE.

Table 5: Client System Configurations

Component	**Hardware Configuration 100GbE Clients**	**Hardware Configuration 10GbE Clients**
Platform	Intel® Server Board S2600WT2	Intel® Server Board S2600WT2
# of Nodes	20	101 (1 head node + 100 client nodes)
# of Sockets	2	2
CPU	Intel® Xeon® CPU E5-2699 v3 @ 2.30GHz	Intel® Xeon® CPU E5-2699 v3 @ 2.30GHz
Cores per socket / threads per socket	18/36 (72 per system)	18/36 (72 per system)
Microcode	0x46	0x44 / 0x46 (may have different versions)
Intel® Hyper-Threading Technology	ON in BIOS, configurable by test harness, default is OFF	ON in BIOS, configurable by test harness, default is OFF
Intel® Turbo Boost Technology	ON in BIOS, configurable by test harness, default is OFF	ON in BIOS, configurable by test harness, default is OFF
BIOS version	SE5C610.8 6B.01.01.0008.021120151325	SE5C610.8 6B.01.01.0008.021120151325
Local Storage used	None	None
System DDR memory configuration: slots/capacity/speed	16 slots/8 GB/2133 MT/s (Micron) 8 slots/ No DRAM modules	16 slots/8 GB/2133 MT/s (Micron) 8 slots/ No DRAM modules
Total memory per node (All DRAM)	128 GB	128 GB
Storage	1x Intel® SSD DC S3500 800GB (boot device)	1x Intel® SSD DC S3500 800GB (boot device)
Network Interface Card	1x Intel® Ethernet Adapter E810-C 1 x Ethernet Controller 10-Gigabit X540-AT2	1 x Ethernet Controller 10-Gigabit X540-AT2
NIC firmware	E810-C: 2.15 0x800049c3 1.2789.0 X540-AT2: 0x800003f1	0x800003f1

1.6 Software Configurations

The table below lists the software configurations for VAST Data CNodes, DNodes and client systems.

Table 6: SW Configurations

Software / Benchmarking Tool	**Version**	**Link**
Client OS Distribution	CentOS Linux 8.4.2105	
Client Kernel	4.18.0-240.22.1.el8_3.x86_64	
VAST Data Appliance	release-3.6.0-sp7 build 465786	Proprietary, contact support@vastdata.com
VAST Data Cnode OS Distribution	CentOS Linux release 7.7.19088.2003 (Core)	
VAST Data CNode Kernel	3.10.0-1127.18.2.el7.vastos.9.x86_64	
VAST Data DNode OS Distribution	CentOS Linux release 7.8.2003 (Core)	
VAST Data DNode Kernel	3.10.0-1127.18.2.el7.VASTos.9.x86_64	
Multipath and NConnect Bundled Driver for CentOS 4.18.0.240	mlnx-nfsrdma-VAST_3.9.3.for.4.18.0.240.x.centos-kernel_4.18.0_240.22.1.el8_3.x86_64.rpm (built by VAST Data)	
irqbalance	1.3.0	
FIO	3.19	fio-3.19-3.el8.x86_64.rpm
Elbencho	Docker image id: 6ac5b651eaa9	Source Repo: https://github.com/breuner/elbencho Docker Registry: https://hub.docker.com/r/breuner/elbencho
Test Harness	a3bf9ac996062de	https://github.com/intel/scale-testing-for-vastdata.git

2) Testing Prerequisites and Setup

Prior to executing the test harness, the following steps must be performed on both the VAST Data cluster and the client systems:

1. Prepare the VAST Data cluster by creating the VIP Pools and S3 User and Access/Secret Keys for testing. The VIP ranges and S3 User and Access/Secret Keys created shall be added to env.conf.override.
2. Prepare the client cluster
   A. Clone the test harness repo onto the head node
   B. Create env.conf.override with the appropriate VIP pool IPs, client cluster host ranges, S3 User and Access/Secret Keys which are mandatory. It can also optionally contain any tunable configurations that are different to the default ones as in env.conf.
   C. Install dependencies
   D. Update MTU for the client cluster (VAST Data default is > 9000 already)
   E. Sanity check of the network connectivity
3. Run test with the test harness

This remainder of this chapter covers the preparation steps in further detail. Running tests with the test harness using FIO and Elbencho are covered in subsequent chapters.

Note that the focus of this guide is on the essential steps to run benchmarks against VAST Data storage at scale, general instructions of how to install VAST Data cluster and client cluster itself are out of scope for this guide.

2.1 Preparing the VAST Data Cluster

All benchmark tests rely on the interactions with the VIP pools on the VAST Data cluster. The easiest way to create a VIP pool is to use the VAST UI.

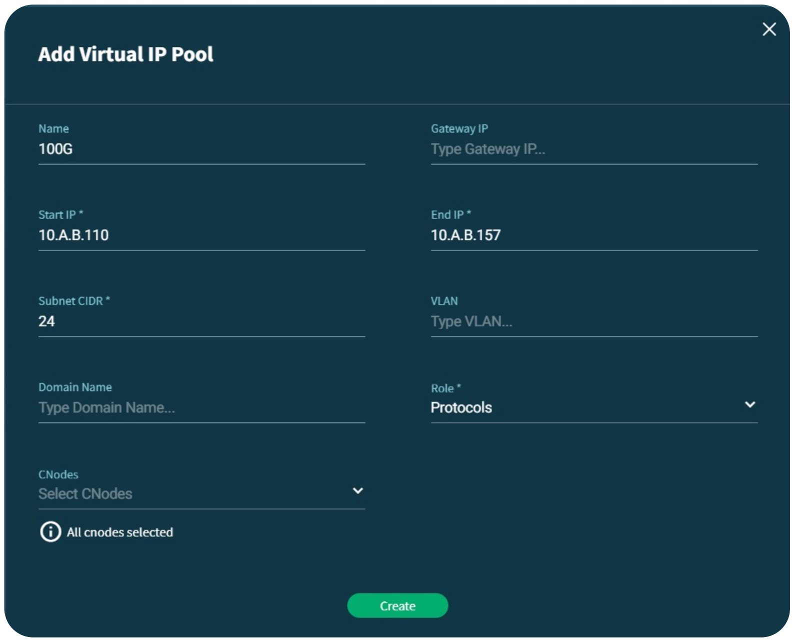
In our benchmark tests, 48 VIPs are used for both 10GbE and 100GbE.

2.1.1 CREATE VIP POOL 10.A.B.110-10.A.B.157 FOR 10GBE

Open VAST UI in a browser with VMS IP, e.g. 10.A.D.E, move the cursor to the left side bar, and click "Network Access" -> "Virtual IP Pool" -> "Create VIP Pool" on the upright corner to open the "Add Virtual IP Pool" window. Fill in Name / Start IP / End IP and Subnet CIDR, then click the "Create" button.

2.1.2 CREATE VIP POOL 192.168.B.110-192.168.B.157 FOR 100GBE

Follow the same procedure to create the VIP pool for 100GbE.

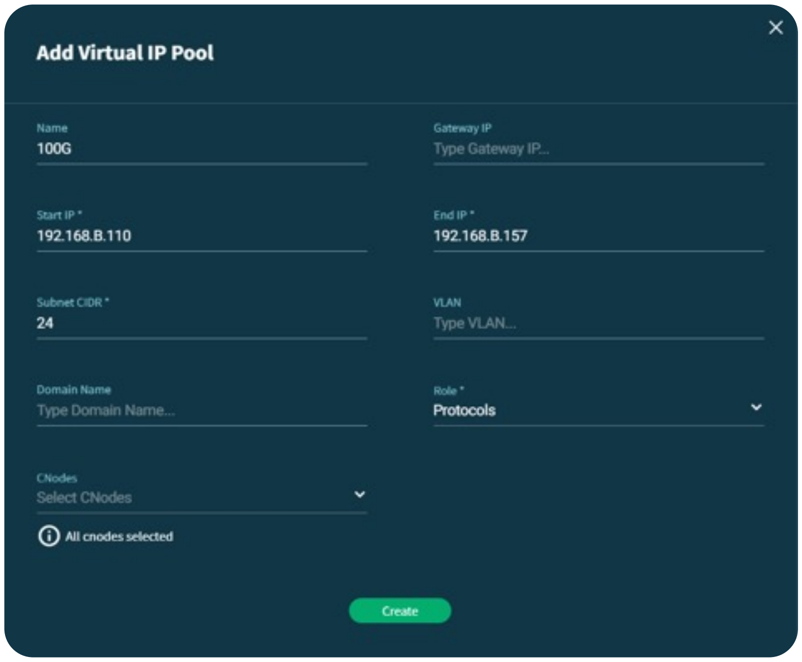


Add Virtual IP Pool

Name: 100G	Gateway IP: Type Gateway IP...
Start IP *: 10.A.B.110	End IP *: 10.A.B.157
Subnet CIDR *: 24	VLAN: Type VLAN...
Domain Name: Type Domain Name...	Role *: Protocols
CNodes: Select CNodes	

All cnodes selected

Figure 4: Create VIP Pool for 10GbE



Add Virtual IP Pool

Name: 100G	Gateway IP: Type Gateway IP...
Start IP *: 192.168.B.110	End IP *: 192.168.B.157
Subnet CIDR *: 24	VLAN: Type VLAN...
Domain Name: Type Domain Name...	Role *: Protocols
CNodes: Select CNodes	

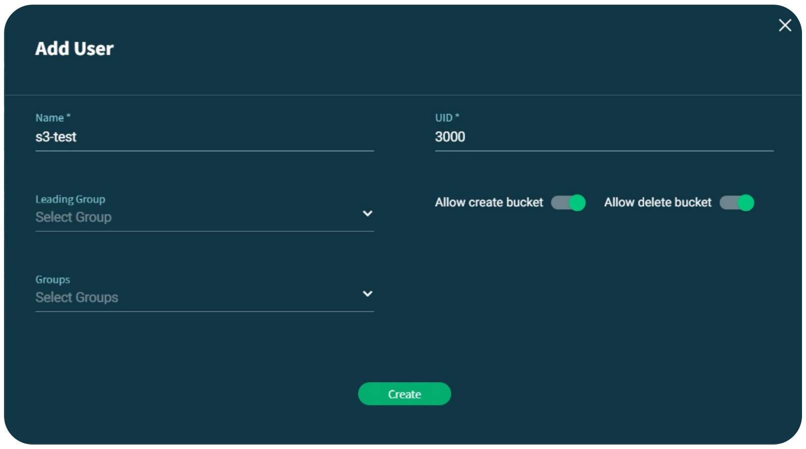
All cnodes selected

Figure 5: Create VIP Pool for 100GbE

2.1.3 CREATE USER FOR ELBENCHO S3 TEST

S3 test requires Access Key and Access Secret Key to be created for a user.

A. Open VAST UI in the browser with VMS IP, move the cursor to the left side bar, and click "User Management" -> "Create User" on the upright corner to open the "Add User" window. Fill in Name / UID, enable "Allow create bucket" and "Allow delete bucket", then click the "Create" button.

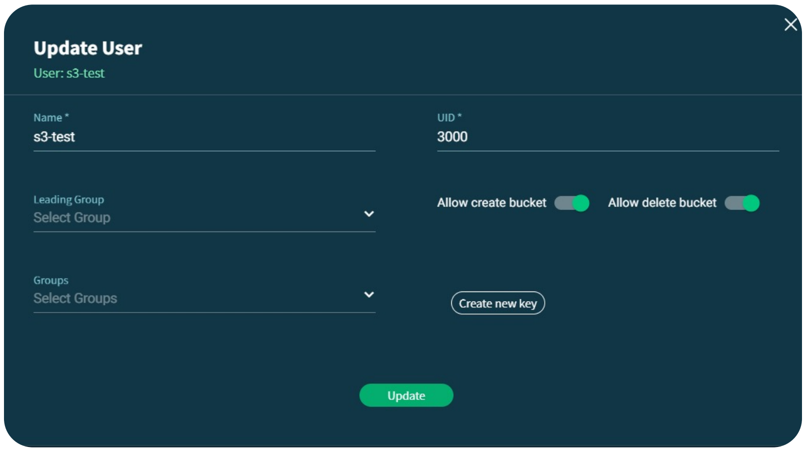


Add User

Name *: s3-test
UID *: 3000
Leading Group: Select Group
Allow create bucket: ✓
Allow delete bucket: ✓
Groups: Select Groups

Figure 6: Create S3 User

B. Click the 3 dots below "Actions" in the row of the newly created user in the "User Management" window -> click "Edit" to open the "Update User" window.

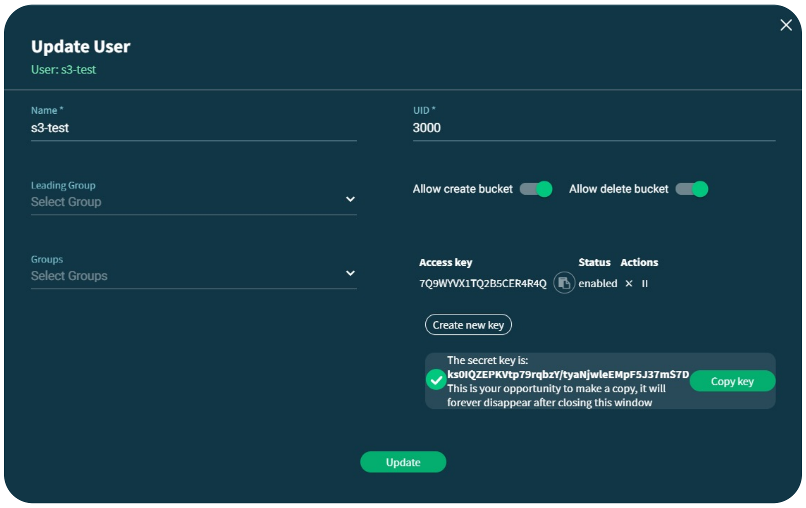


Update User
User: s3-test

Name *: s3-test
UID *: 3000
Leading Group: Select Group
Allow create bucket: ✓
Allow delete bucket: ✓
Groups: Select Groups

Figure 7: Update S3 User

C. Click "Create new key" button. Access Key and Secret Key will show on the screen. Click the icon in-between the Access Key and Status to copy the Secret Key. And click "Copy key" to copy the Secret Key. Click "Update" after both keys have been copied. Note that once closing the "Update User" window, the Secret Key is not visible anymore when clicking "Edit" again. However, "Create new key" can be used to add new key pairs.



Update User
User: s3-test

Name *: s3-test
UID *: 3000
Leading Group: Select Group
Allow create bucket: ✓
Allow delete bucket: ✓
Groups: Select Groups

Access key	Status	Actions
7Q9WYVX1TQ2B5CER4R4Q	enabled	× ⏸

The secret key is:
ks0IQZEPKVtp79rqbzY/tyaNjwleEMpF5J37mS7D
This is your opportunity to make a copy, it will forever disappear after closing this window

Figure 8: Create S3 Access and Secret Keys

2.2 Preparing the Client Cluster

2.2.1 CLONE THE TEST HARNESS REPO

Both FIO and Elbencho benchmarks are automated, test harness can be cloned from GitHub. Usually this is cloned to a head node which facilitates the test; however no workloads will be running in-between the head node itself and VAST Data cluster.

```
$ git clone https://github.com/intel/scale-testing-for-vastdata.git
```

Figure 9 below illustrates how the test harness interacts with Elbencho/FIO and ClusterShell to run tests against the cluster:

- VAST Data FS is exposed by VIPs mounted to the head (for utilities) and client nodes (for IO) via NFSv3 for file testing.
- VAST Data Objects are also exposed by VIPs for S3 object testing.
- Test Harness uses ClusterShell to configure clients, execute tests, and gather telemetry and results as needed.
- Both FIO and Elbencho run in Client/Server mode, with Client on the head node, and Server on client nodes.
- A separate remote server for result store can also be NFS mounted on the head node for result upload.
- $HOME/vast-scale-testing folder will be created on both the head and client nodes to facilitate the tests.
- A result folder will be created in repo top folder on the head node while the test is running.

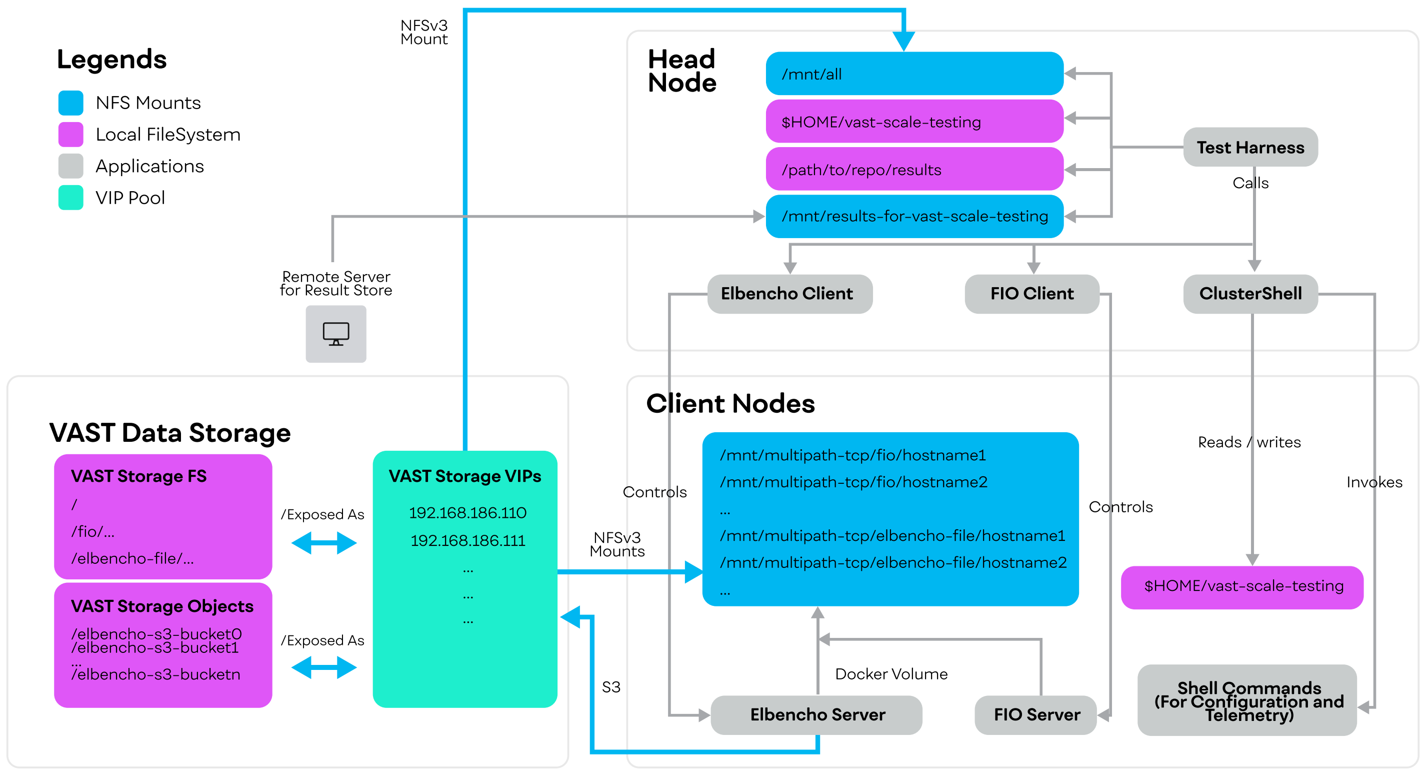
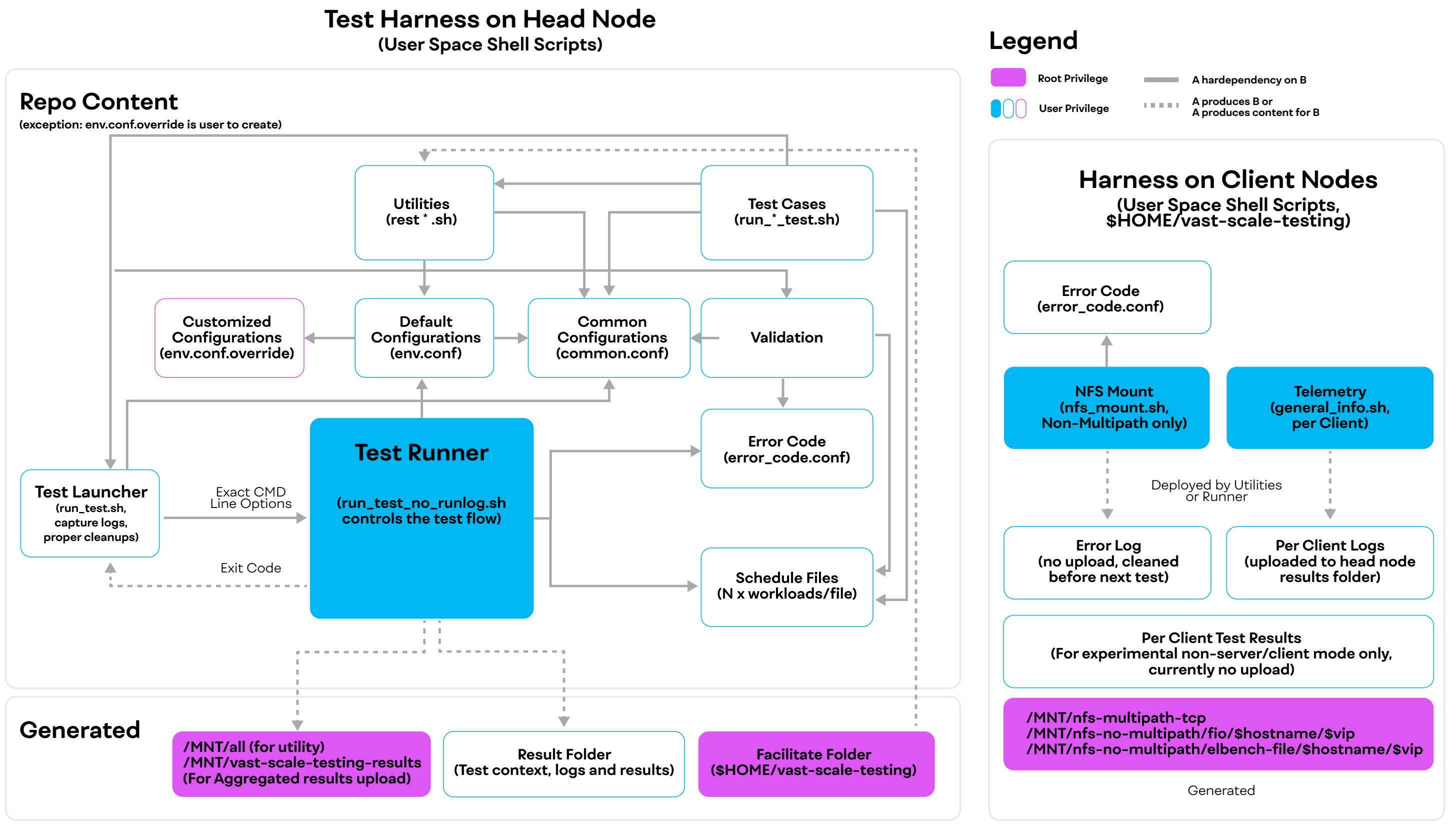


Figure 9: Block Diagram of How Test Harness Works

Figure 10 below depicts the shell script-based Test Harness composition on both the head node and client nodes.

- Repo content on the head node is via git clone.
- Some scripts will be deployed to client nodes via install_dep.sh for faster execution of test harness.
- During test, both the head node and client nodes will generate some folders, i.e. for mounts, results and test facilitation.
- Test Runner run_test_no_runlog.sh is the heart of the test harness
- The test launcher run_test.sh is a wrapper of the runner, and it invokes the runner by passing all command line options transparently.
- The other *.sh files are utilities
- env.conf / common.conf define the global variables needed by the launcher, runner and utility scripts
- common.conf is only directory aware; env.conf is topology aware
- There is also a validation component in the test harness



Test Harness on Head Node
(User Space Shell Scripts)
Repo Content
(exception: env.conf.override is user to create)
Utilities (rest *.sh)
Test Cases (run_*_test.sh)
Customized Configurations (env.conf.override)
Default Configurations (env.conf)
Common Configurations (common.conf)
Validation
Error Code (error_code.conf)
Test Runner
(run_test_no_runlog.sh controls the test flow)
Test Launcher (run_test.sh, capture logs, proper cleanups)
Exact CMD Line Options
Exit Code
Schedule Files (N x workloads/file)
Generated
/MNT/all (for utility) /MNT/vast-scale-testing-results (For Aggregated results upload)
Result Folder (Test context, logs and results)
Facilitate Folder ($HOME/vast-scale-testing)
Legend
Root Privilege
User Privilege
A hardependency on B
A produces B or A produces content for B
Harness on Client Nodes
(User Space Shell Scripts, $HOME/vast-scale-testing)
Error Code (error_code.conf)
NFS Mount (nfs_mount.sh, Non-Multipath only)
Telemetry (general_info.sh, per Client)
Deployed by Utilities or Runner
Error Log (no upload, cleaned before next test)
Per Client Logs (uploaded to head node results folder)
Per Client Test Results (For experimental non-server/client mode only, currently no upload)
/MNT/nfs-multipath-tcp /MNT/nfs-no-multipath/fio/$hostname/$vip /MNT/nfs-no-multipath/elbench-file/$hostname/$vip
Generated

Figure 10: Block Diagram of Test Harness Composition

Figure 11 below shows the flow diagram of the runner (run_test_no_runlog.sh).

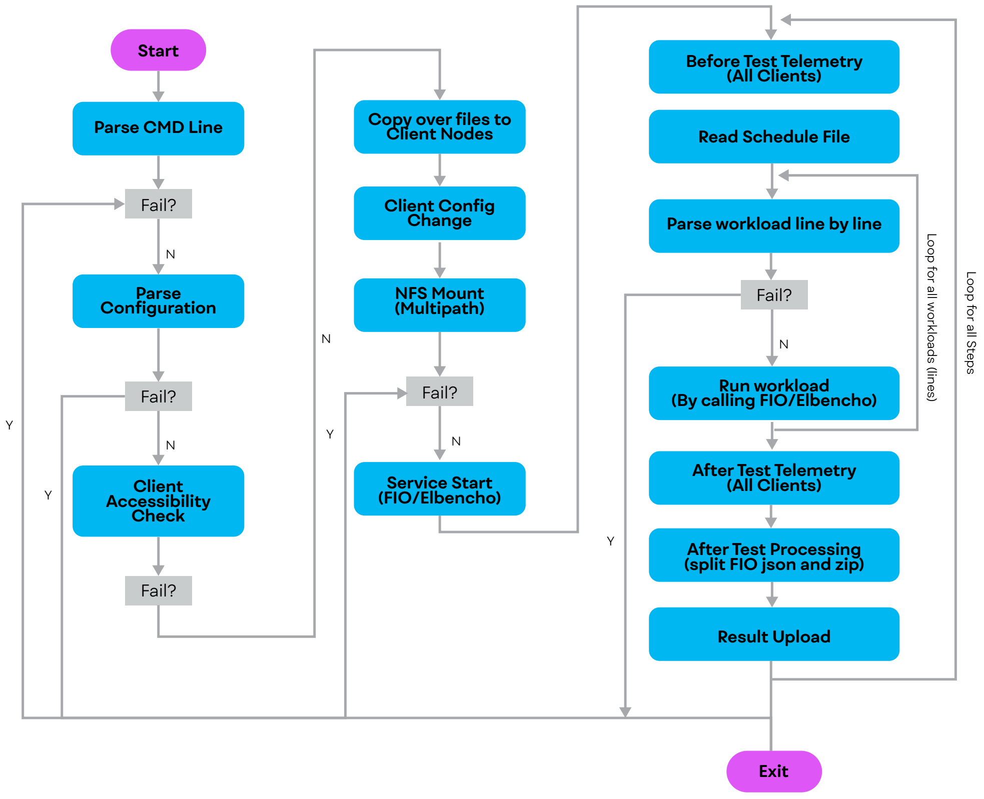

Start
Parse CMD Line
Fail?
N
Parse Configuration
Fail?
N
Client Accessibility Check
Fail?
Y
Y
N
Copy over files to Client Nodes
Client Config Change
NFS Mount (Multipath)
Fail?
N
Y
Service Start (FIO/Elbencho)
Before Test Telemetry (All Clients)
Read Schedule File
Parse workload line by line
Fail?
N
Run workload (By calling FIO/Elbencho)
After Test Telemetry (All Clients)
After Test Processing (split FIO json and zip)
Result Upload
Y
Loop for all workloads (lines)
Loop for all Steps
Exit

Figure 11: Runner Flow Diagram

There are 2 top level sub folders, namely scripts and workloads in the test harness after a fresh clone. A results folder also will be generated after running the test. A drivers folder can be created by the user in the top-level folder for storing Multipath & NConnect bundled driver required by the dependency installation script. The table below contains the high-level overview of all files.

Table 7: Overview of Test Harness Files

Folder Name	File	Description
Top Level	LICENSE	BSD 3-Clause License
	README.md	Top Level of Readme
	.gitignore	To ignore results folder and env.conf.override for source control
Workloads	fio/template.ini	It contains common fio parameters on which all FIO workloads will be based. Some options e.g. directory, numjobs, bs, iodepth, rw, create_only etc. are place holders only and might be updated by the harness after parsing command line or schedule files. The head node generates the per-client (directory option is different for each client) FIO job file, and supplies them to FIO as the –client parameter.

	fio/*.sch elbencho/file/*.sch elbencho/s3/*.sch fio/README.md elbencho/file/README.md elbencho/s3/README.md	A rich set of predefined schedule files for FIO/Elbencho File and Elbencho S3 testing respectively. Each schedule file defines 1 or more workloads, with 1 line of FIO/Elbencho File or Elbencho S3 parameters per workload. Refer to README.md in each folder to know which parameters are mandatory and which are optional. Most parameters are just native FIO/Elbencho File or Elbencho S3 parameters, with a few additions by the test harness itself. Refer to README.md for details. Suffix "10g" and "100g" indicates which interface each file is for. Parameter –size (FIO) and –file (Elbencho File and S3) for 100GbE are designed to be 5 times of 10GbE's. This is to keep the total data size across all clients of 10GbE and 100GbE testing the same, since there are 100 10GbE clients but only 20 100GbE clients. If the number of clients are different, then adjust the –size and –file accordingly for your own environment. A typical way of running a test would be from scripts folder: $ ./run_test.sh -ns 100 -sf path-to-schedule-file $ ./run_test.sh -ns 10 -sf path-to-schedule-file e.g.: $ ./run_test.sh -ns 100 -sf ../workloads/fio/bw_test_100g.sch $ ./run_test.sh -ns 10 -sf ../workloads/fio/bw_test_10g.sch Note that: 1. Schedule file suffix shall match the -ns parameter when running the test. If the suffix doesn't match, either not sufficient data will be generated, or data will be generated 5 times as designed for the scenario on write, thus takes much longer to run and also may not produce the result as designed by the test case. 2. Schedule files must be in these 3 folders for the test harness to detect the workload type, i.e. fio vs Elbencho file vs Elbencho S3 properly. Putting a schedule file outside of these 3 folders or putting them into the folder not matching its type are not supported by the test harness.
scripts	env.conf	Global configuration file. Used by most scripts that need to be topology (Clients and VIP) aware. **Usage:** $ source env.conf [network_speed step_size] **Where:** network_speed: in [10, 100]. Default: 10 step_size: in [0, CLIENT_COUNT]. Default: 0 (no incremental stepping, run on all clients in 1 step) This file is common for all users. For any user specific variables, please define them in env.conf.override instead. Dummy variables are provided in env.conf more as the references, and have to be redefined in env.conf.override. Define tunable ones in env.conf.override only if you want to configure them differently.
	env.conf.override	This is the user specific configuration file, which doesn't exist in fresh clones, but shall be created by the user. This file is ignored by .gitignore and won't be under source control. Refer to Create env.conf.override to know how to configure the variables in the file to suit the needs of a test.
	common.conf	Global configuration file. Used by scripts that only need to be folder structure aware, but not topology aware. **Usage:** $ source common.conf
	error_code.conf	Defines test harness generated error code.

install_dep.sh	Run this to install the dependencies required by the test harness. **Usage:** $ ./install_dep.sh [network_speed] **Where:** network_speed: in [10, 100]. Default: 10 Section Install the dependencies explains all dependencies to be installed.
run_test_with_no_runlog.sh	Script called by run_test.sh to launch the test. Note that the script will do FIO server cleanups and NFS umounts before the test to ensure a clean run, however it will leave the FIO servers and NFS mounts in place after the test. **Usage:** $ ./run_test_with_no_runlog.sh [options] Refer to run_test.sh/run_test_no_runlog.sh command line options for all command line options.
run_test.sh	Script to launch the test by calling run_test_with_no_runlog.sh and save the console output as run.log in the results folder at the end of the test. For better debuggability, also use this in the command line rather than run_test_with_no_runlog.sh directly. **Usage:** $ ./run_test.sh [options] Options are the same as run_test_with_no_runlog.sh as it just passes all arguments to run_test_with_no_runlog.sh transparently. Refer to run_test.sh/run_test_no_runlog.sh command line options for all command line options.
split_json.sh	Script used by run_test_no_runlog.sh to split FIO json result per client. **Usage:** $ ./split_json.sh filename Filename must have ending _summary.log, and it must be the FIO json+ output format from FIO Client/Server mode.

run_iperf_test.sh	Basic networking performance test with iperf3.
	Usage:
	Precondition: iperf3 installed on both client and VAST Data side (install_dep.sh installs that for client side only. If VAST Data side has a different version, then log on to VMS and do "clush -g cnodes sudo yum install iperf3" to install it first. Mismatching iperf, ie. iperf3 on client and iperf2 on VAST Data side won't work).
	Steps to test 10G performance:
	Step 1: log onto VAST VMS node, note down one of the VIP on the VMS node, e.g. 10.A.B.E
	Step 2: start iperf as the service on the VMS node
	$ iperf3 -s -p5900
	Step 3: run iperf test script on the client:
	$ ./run_iperf_test.sh 10 10.A.B.E
	Steps to test 100G performance:
	Step 1: log onto VAST VMS node, note down one of the VIP on the VMS node, e.g. 192.168.B.F
	Step 2: start iperf as the service on the VMS node, but bind iperf3 to the 100GbE interface IP
	$ iperf3 -s -p5900 -B 192.168.B.F
	Step 3: on client 101, run iperf test script:
	$ ./run_iperf_test.sh 100 192.168.B.F
	Refer to Sanity Check of the Network Connectivity with iperf3 for examples.

	mtu_update.sh	Script to update eth0 or specified network interface to 9000 MTU for better performance. The script will be deployed to all client nodes with install_dep.sh. **Usage:** $ ./mtu_update.sh [network_interface] **Where:** network_interface: the interface for which MTU to be changed. Default: eth0 Refer to MTU Update for more details.
	nic_info.sh	Script to query the NIC information. **Usage:** $ ./nic_info.sh [network_speed] **Where:** network_speed: in [10, 100]. Default: 10
	cleanup_test.sh	Cleanup all processes launched by the test harness. This is for manual FIO server cleanups on all clients, if desired. **Usage:** $ ./cleanup_test.sh [network_speed] **Where:** network_speed: in [10, 100]. Default: 10
	cleanup_mounts.sh	Cleanup all NFS mounts created by the test harness. This is for manual NFS mount cleanups on all clients, if desired. **Usage:** $ ./cleanup_mounts.sh [network_speed] **Where:** network_speed: in [10, 100]. Default: 10
	purge_cluster.sh	Mount a VAST Data cluster VIP root to the head node at /mnt/all, and cleanup all test data using .VAST_trash. This assumes all data generated by the test harness are stored in following folders: 1. /mnt/all/$REMOTE_PATH_FIO 2. /mnt/all/$REMOTE_PATH_ELBENCHO_FILE 3. /mnt/all/$REMOTE_PATH_ELBENCHO_S3 4. /mnt/all/elbencho-s3-bucket* It doesn't clean up other data stored outside of these folders. Refer to Create env.conf.override for how the remote paths are defined for FIO / Elbencho File and Elbencho S3.
	nfs_mount.sh	Copied by install_dep.sh to all client nodes to mount VIPs individually, used by run_test_no_runlog.sh only when MULTIPATH_ENABLED is set to 0 in env.conf. **Usage:** $ ./nfs_mount.sh MOUNT REMOTEPATH HOST_SUBNET NCONNECT_NUM VIPs **Where:** MOUNT: Mountpoint parent folder. The actual mountpoint can be a subfolder below it. REMOTEPATH: remote path depending on whether it's FIO, Elbencho file or S3 testing. HOST_SUBNET: host subnet, eg 192.168.B or 10.A.B NCONNECT_NUM: nconnect number to be used VIPs: all VIPs to be mounted **Note:** if this script gets updated, then it must be re-deployed to all clients for run_test_no_runlog.sh to execute properly. Currently install_dep.sh does the deployment; run_test_no_runlog.sh also does the deployment before every test which can be opt out in the future.

general_info.sh	Script used by run_test_no_runlog.sh to collect general system and hardware information for telemetry purpose before the test, logs will be saved to results/ rundir/context folder, where rundir is auto generated based on date, schedule filename, the head node name and prefix etc. **Usage:** $ ./docker_start.sh [network_speed] **Where:** network_speed: in [10, 100]. Default: 10

elbencho_deploy.sh	Anonymous and authenticated docker users may run into docker pull limits quickly.
	This script provides a workaround to manually pull once from the head node and deploy the elbencho docker image to all systems.
	Usage:
	Step 1: pull once on the head node
	$ docker pull breuner/elbencho
	Step 2: note down "IMAGE ID" of the pulled image
	$ docker image ls
	Example response:
	REPOSITORY TAG IMAGE ID CREATED SIZE
	$ docker.io/breuner/elbencho latest 6ac5b651eaa9 4 days ago 136 MB
	Step 3: update ELBENCHO_ID and ELBENCHO_TAG accordingly in env.conf. override if different to default. Tag is user defined string to help identify the version in a more convenient way.
	Step 4: run this script to deploy accordingly
	$ ./elbencho_deploy.sh

container_logs.sh	To query Elbencho container logs on client nodes and print to console. Useful for checking error conditions for abnormal execution. **Usage:** $ ./ container_logs.sh [network_speed] **Where:** network_speed: in [10, 100]. Default: 10
docker_start.sh	To start docker on all client systems. This can be useful if the system has not been configured to automatically start docker services after reboot. **Usage:** $ ./docker_start.sh [network_speed] **Where:** network_speed: in [10, 100]. Default: 10
run_bw_test.sh	Three iterations of BW test with FIO, Elbencho file and Elbencho S3 on both 10GbE and 100GbE clients. **Usage:** $ ./run_bw_test.sh This is a wrapper script for running workloads.
run_iops_test.sh	Three iterations of iops test with FIO, Elbencho file and Elbencho S3 on both 10GbE and 100GbE clients. **Usage:** $ ./run_iops_test.sh This is a wrapper script for running workloads.

	run_fio_bw_test_short.sh	Three iterations of short BW test with FIO, Elbencho file and Elbencho S3 on 100GbE clients. **Usage:** $ ./run_fio_bw_test_short.sh This is a wrapper script for running workloads.
	run_elbencho_s3_sweep_iosizes.sh	Three iterations of sweeping S3 iosizes test with Elbencho on 100GbE clients. **Usage:** $ ./run_elbencho_s3_sweep_iosizes.sh This is a wrapper script for running workloads.
	run_fio_sweep.sh	Worker, iodepth and iosize sweep on both 100GbE and 10GbE with FIO. **Usage:** $ ./run_fio_sweep.sh This is a wrapper script for running workloads.
	run_elbencho_file_sweep.sh	Threads, iodepth and iosize sweep on both 100GbE and 10GbE with Elbencho File. **Usage:** $ ./run_elbencho_file_sweep.sh This is a wrapper script for running workloads.
	run_elbencho_s3_sweep.sh	Threads and iosize sweep on both 100GbE and 10GbE with Elbencho S3. **Usage:** $ ./run_elbencho_s3_sweep.sh This is a wrapper script for running workloads.
results	Generated after running the test. Refer to Test Results Explained for how to understand and parse the results.	
drivers	Created by user to store kernel dependent Multipath driver (contact support@VASTdata.com for it)	

2.2.2 CREATE ENV.CONF.OVERRIDE

There are 3 types of variable definitions in this env.conf.

A. User defined dummy variables

These are environment dependent variables and updates are usually mandatory except for SHAREPOINT_HOST/SHAREPOINT_FOLDER, which depends on if RESULTS_CIFS_UPLOAD_ENABLE is 1 or not.

B. User defined tunable variables

These variables allow the test to be running in different ways, which may or may not require changes.

C. Derived variables

These are variables populated automatically, mostly by parsing the user defined variables and command line arguments. No changes required for these variables.

Usually users don't need to change env.conf, but do need to create a user specific env.conf.override by providing the actual values for dummy variables + any tunable variables that are different to the default values in env.conf. Here is an example of it:

```
#!/bin/bash


# Intel Copyright © 2022, Intel Corporation.

# SPDX-License-Identifier: BSD-3-Clause


VIP_POOL_10G="Your10GVIPRange"

VIP_POOL_100G="Your100GVIPRange"

CLIENT_NODES_10G="Your10GClientClusterRange"

CLIENT_NODES_100G="Your100GClientClusterRange"


ELBENCHO_S3KEY="ReplaceWithYouOwnKey"

ELBENCHO_S3SECRET="ReplaceWithYouOwnSecret"


FIO_PATH=/usr/local/bin
```

All user defined variables are listed below.

Table 8: User Defined Variables

Variable Types	Variables	Default	Description
Dummy	_VIP_POOL_10G _VIP_POOL_100G	10.A.B.110-10.A.B.157, 192.168.B.110-192.168.B.157	The test harness assumes the VIPs are always contiguous. Follow Create VIP Pool 10.A.B.110-10.A.B.157 for 10GbE and Create VIP Pool 192.168.B.110-192.168.B.157 for 100GbE to create them and define the actual values in env.conf.override.
	_CLIENT_NODES_10G _CLIENT_NODES_100G	client-clus-ter[01-100], cli-ent-cluster[01-20]	Define the actual client cluster hostnames and range in env. conf.override. Client node IPs don't have to be contiguous, i.e. they can be client-cluster[05,10-20,50-55, ...] The number of client nodes can be any number >= 1. The lead node (FIO client) shall not be in the list. However, do keep in mind that if the number of clients are not 20 for 100GbE and 100 for 10GbE, then the size of data generated by using the predefined schedule files for write workloads would be different, hence may have implications to read performance later depending on where the data is. To deal with different number of hosts: • If total clients are indeed different: adjust the data size in schedule file accordingly, e.g. if there are only 20 10G, then possibly you can just use the 100g version of schedule files for all of the tests, or • If total clients are 20 / 100, but for read test you would like to test a subset of clients: write with the exact 20 / 100 clients, but afterwards reduce the number of read clients as needed for read workloads.
	ELBENCHO_S3KEY ELBENCHO_S3SECRET	ELBENCHO_ S3KEY="YU5... LQ1", ELBENCHO_ S3SECRET="aqb... EtnQ"	Needed for S3 testing. Follow Create User for Elbencho S3 Test to create them and define them in env.conf.override.
	SHAREPOINT_HOST SHAREPOINT_FOLDER	YourSharepointHo-stName, //${SHARE-POINT_HOST}/ YourRemotePathOn-SharepointHost	For result upload to NFS share on remote server. Add it to env.conf.override only if RESULTS_UPLOAD_ENABLE also is defined to 1 in env.conf.override. Refer to the description of RESULTS_UPDATE_ENABLE variable for how to set it up
Tunable	TURBO_BOOST_ENABLE	0	Intel Turbo Boost control. 0: to disable it 1: to enable it 1 only works only if BIOS also has it enabled.
	HYPER_THREADING_ ENABLE	0	Hyper Threading control 0: to disable it 1: to enable it 1 only works only if BIOS also has it enabled.
	IRQBALANCE_ENABLE	0	irqbalance control. 0: to disable it 1: to enable it This requires irqbalance to be installed (covered by in-stall-dep.sh).

Tunable	ORDERED_NODES	1	Nodes can be defined as ordered or unordered. Normally ordered types can be used. However, sometimes it is useful to run in the specified order other than the sorted order, e.g. for incremental stepping test, if you would like to iterate through the faster clients first, then the slower clients. 1. Ordered nodes **CAN** be defined in 2 different formats of: a) node[01-02,30-32,10-12], or b) node{01..02} node{30..32} node{10..12} Nodes will be expanded to "node01 node02 node10 node11 node12 node30 node31 node32" with nodeset --expand 2. Unordered nodes **SHALL** be defined in format of: node{01..02} node{30..32} node{10..12} Nodes will be expanded to "node01 node02 node30 node31 node32 node10 node11 node12" with eval only.
	FIO_SERVER_CLIENT_MODE	1	To enable/disable FIO Server/Client mode. **1: to enable FIO Server/Client mode.** Aggregation: results are aggregated in fio json+ output as "All clients" Where: head node "results" folder **0: to disable FIO Server/Client mode.** Aggregation: no result aggregation of all clients, json+ output is per client. Where: in ~/$VAST_SCALE_TESTING_PATH folder For better total BW/IOPS/Latency tracking, use Server/Client mode.
	MULTIPATH_ENABLE	1	Kernel dependent Multipath driver is needed in order to enable this. Contact support@vastdata.com for it if needed. Enabling Multipath allows simpler VIP mount on client nodes and more balanced load on VAST Data CNodes and DNodes, especially when worker/thread count is low. When the variable is set to 0, VIPs are going to be mounted individually. Note that workload files could be created differently on remote nodes for MULTIPATH_ENABLE=0 and 1, hence for the same read workload, by just changing this variable, it may cause the file to be regenerated.
	MOUNT_ALL	1	Applicable to FIO and MULTIPATH_ENABLE=0 only. 1: One mount per VIP per client node. Overall mounts per client = VIP number. Mounts are created before any workloads start. This is a preferred way of mounting, as it's faster and has no noticeable performance penalty due to over mount, i.e. if mount number > job/worker number. 0: One mount per job/thread per client node. Overall mounts per client = job/thread number. Mounts are created on the fly when running workload. This could be slower.

	MOUNT_STAGGER	5	This is only applicable to MULTIPATH_ENABLE=0 and MOUNT_ALL=0 . This is to avoid creating too many clush sessions when doing individual mounts on the fly at one time.
	MOUNT_MULTIPATH_ENABLED	/mnt/nfs-multipath-tcp	The mountpoint on the client system when Multipath is enabled.
	MOUNT_MULTIPATH_DISABLED	/mnt/nfs-no-multipath	The mountpoint on the client system when Multipath is disabled.
	NCONNECT_NUM	48	NConnect allows multiple TCP connections to be created for each mount, hence improves the throughput effectively. This is independent of Multipath and can be used together with Multipath. NConnect is supported from Linux kernel 5.3. For kernel 4.18, CentOS also backported it. The Multipath driver built by VAST Data can be a bundle of both. NCONNECT_ENABLE must be set to 1 for this to take effect.

	NCONNECT_ENABLE	1	When MULTIPATH_ENABLE=0, and NCONNECT_ENABLE=1, big NCONNECT_NUM means a large number of TCPs will be created per client as the table below shows. This could (1) take a long time to mount before a test can start (2) cause connection aborts if too many TCP contentions are created.
			Best practice is to use:
			MULTIPATH_ENABLE=1
			NCONNECT_ENABLE=1
			Or:
			MULTIPATH_ENABLE=0
			NCONNECT_ENABLE=0

MULTIPATH_ENABLED	NCONNECT_ENABLE	Per Client TCPs
1	1	NCONNECT_NUM
1	0	1
0	0	VIP Count
0	1	VIP Count * NCONNECT_NUM

Table 9: Per Client TCP Connections with MOUNT_ALL=1

	NCONNECT_NUM	48	NConnect allows multiple TCP connections to be created for each mount, hence improves the throughput effectively. This is independent of Multipath and can be used together with Multipath. NConnect is supported from Linux kernel 5.3. For kernel 4.18, CentOS also backported it. The Multipath driver built by VAST Data can be a bundle of both. NCONNECT_ENABLE must be set to 1 for this to take effect.

REMOTE_PATH_FIO	fio	Folder to be created under VIP root for FIO testing
REMOTE_PATH_ ELBENCHO_FILE	elbencho-file	Folder to be created under VIP root for Elbencho File testing
REMOTE_PATH_ ELBENCHO_S3	elbencho-s3	Folder to be created under VIP root for Elbencho S3 Object testing
FIO_PATH	/usr/local/bin	If FIO is installed in /usr/bin or other path, change FIO_PATH accordingly
VAST_SCALE_TESTING_ PATH	~/VAST-scale-testing	VAST-scale-testing will be created under home to facilitate the installations and tests. For FIO non-Server/Client mode, per client result folders will also be created there.
ELBENCHO_ID	6ac5b651eaa9	Use "docker image ls" to find it on the lead node after the docker pull.
ELBENCHO_TAG	scaletest01	Change the tag after a new deployment to help identify the image.
ELBENCHO_IMG	elbencho.$ELBEN-CHO_TAG	Full image identifier by tag.
ELBENCHO_IMAGE_ FOLDER	$VAST_SCALE_TEST-ING_PATH/elbencho	This defines the folder to store the elbencho image on head and client nodes.
RESULTS_UPDATE_ENABLE	0	Disabled by default. To use to: 1. Ensure CIFSNFS share for result upload is configuredsetup on a remote server and working from the head node 2. UpdateDefine SHAREPOINT_HOST and SHAREPOINT_ FOLDER accordingly in env.conf.override 3. Create /root/.cifs with the username and password for CIFS username=your_cifs_username password=your_cifs_password 4. ChangeDefine RESULTS_CIFS_UPDATE_ENABLE assigned valueto 1 in env.conf.override
RESULTS_MNTPOINT	/mnt/VAST-scale-testing-results	Mountpoint for results upload on head node. The test harness automatically mount the sharepoint, upload the results to sharepoint, and then umount the sharepoint before ending the test.
INTER_WORKLOAD_WAIT_ IN_SECONDS	60	Sleep time in-between workloads in seconds.
INTER_STEP_WAIT_IN_ SECONDS	120	Sleep time in-between tests in seconds.
CLUSH_MAX_FAN_OUT	120	Fan out for ClusterShell comands CentOS default 64 is not optimal for scale testing with 100 client nodes, as nodes beyond the threshold will be stag-gered until the previous batch is done, causing undesired synchronization issues.

2.2.3 INSTALL THE DEPENDENCIES

S Prior to installing the dependencies, please check your kernel version with "uname -r" and acquire the Multipath + NConnect buddled driver matching your kernel version from support@vastdata.com and add MULTIPATH_DRIVER to env.conf.override. After that, the driver and all other dependencies can be installed by calling install-dep.sh.

ince 100GbE hosts are just a subset of 10GbE hosts, installing dependencies on 10GbE shall be good enough.

```
$ ./install-dep.sh
```

However, for an environment using different 100GbE and 10GbE clients, make sure both are done.

```
$ ./install-dep.sh 10

$ ./install-dep.sh 100
```

A manual reboot of all clients are needed to activate the Multipath + NConnect bundled driver before running the test. The reboot is commented out in case another convenient time is preferred for reboot.

```
$ source env.conf 10

$ clush -w $CLIENT_NODES_10G, $CLIENT_NODES_100G "sudo reboot"

$ sudo reboot  # To reboot head node too
```

After reboot, a mount with Nconnect and Multipath of the VIPs should be successful. For example, the following command if executed on 10.A.B.5 10GbE client will mount all 48 VIPs as remote ports to local port 10.A.B.5, and it's using 10.A.B.123:/ as the remote address to mount.

```
$ sudo mkdir -p /mnt/nfs-multipath-tcp

$ sudo mount -v -o vers=3,proto=tcp,nconnect=48,port=20048,localports=10.A.B.5,
emoteports=10.A.B.110-10.A.B.157 10.A.B.123:/ /mnt/nfs-multipath-tcp     ===>
here both nconnect and multipath used

$ mount

…

10.A.B.123:/ on /mnt/nfs-multipath-tcp type nfs (rw,relatime,vers=3,rsize=1048576,
wsize=1048576,namlen=255,hard,proto=tcp,nconnect=48,port=20048,timeo=600,retrans=2,
sec=sys,mountaddr=10.A.B.123,mountvers=3,mountport=20048,mountproto=tcp,
local_lock=none,addr=10.A.B.C)    ===>
where C can be any number within 110~157

$ sudo umount /mnt/nfs-multipath-tcp  ===> this will umount the VIPs
```

When Multipath is enabled, the test harness always uses the full range of VIPs defined in env.conf.override for remote ports. The remote address however will be round robinned among all clients, e.g. the 1st client uses 10.A.B.110, 2nd client uses 10.A.B.111 etc.

All these however will be managed by the test harness and is transparent to the user.

If the client systems don't restart docker automatically after reboot, also do the following to start docker.

```
$ source env.conf 10

$ ./docker_start.sh 10
```

For environments that use different 100GbE and 10GbE clients, make sure the following is also done.

```
$ source env.conf 100

$ ./docker_start.sh 100
```

If docker is not started on a client, then the entire test with Elbencho will be aborted, and there might be error like below in console or run.log.

```
$ ....Communication error in preparation phase: Connection refused. Service:
hostname:1611
```

By default, install-dep.sh only pulls Elbencho to the head node, the deployment of Elbencho to all other nodes needs to be done either by using elbencho_deploy.sh after manual steps 2 and 3 as mentioned in its comments (suitable for users with docker pull limt):

```
$./elbencho_deploy.sh
```

Or by a direct pull manually (suitable for users without docker pull limit), eg:

```
$ source env.conf 10

$ clush -w "$CLIENT_NODES" "docker pull breuner/elbencho"
```

In this case, refer to steps 2 and 3 for how to get ELBENCHO_ID and ELBENCHO_TAG added in env.conf. override. then:

```
$ source env.conf 10

$ clush -w "$CLIENT_NODES" "docker image tag $ELBENCHO_ID breuner/elbencho:$ELBENCHO_TAG"
```

Elbencho and FIO services will be started by the test harness automatically on all client nodes, no need to manually do it before the test.

```
$ ps aux | grep fio

bduser     67771  0.0  0.0 440076  5316 ?        Ss   Nov18   0:00 /usr/bin/fio
--server --daemonize=fiopid

$ docker ps

CONTAINER ID   IMAGE                          COMMAND                   CREATED
STATUS           PORTS     NAMES

d099ebdffb8f   breuner/elbencho:scaletest01   "/usr/bin/elbencho -…"   50 seconds ago
Up 50 seconds              elbencho-server
```

The services won't be cleaned up automatically after the test though, but manual cleanup is possible if desired.

```
$ ./cleanup_test.sh 10

$ ./cleanup_test.sh 100  # if 100GbE is not a subset of 10GbE hosts
```

Table 9 lists all the dependencies to be installed and why each dependency is needed.

Table 9: Dependencies to be Installed

Name	Description
ClusterShell	The harness uses clush command extensively, so that all operations can be issued from the shell of the head node without logging on to other systems.
Multipath & NConnect Driver	Refer to descriptions of MULTIPATH_ENABLE and NCONNECT_NUM in Table 8: User Defined Variables for why Multipath and NConnect are needed. This shall be installed on all client systems running FIO server or Elbencho as the service.
FIO	Server/Client mode is used to run FIO, for easier result aggregation. To understand FIO Server/Client mode more, please refer to https://linux.die.net/man/1/fio section CLIENT/SERVER. In summary: FIO Client: this is the lead node which will facilitate the FIO execution on FIO Servers, it doesn't generate the IOs to the VAST Data cluster storage. FIO Server: the 100 10GbE clients/20 100GbE clients that are performing read/write from/to the VAST Data cluster storage through NFSv3 mounts. As such, on top of the 100 client systems, another system is needed to be used as the FIO client lead node. The test harness repo is only required to be cloned on the lead node, however FIO shall be installed on all nodes including the lead node and clients.
jq	A json parser used by the harness to split FIO json results to per client files. This could make it easier to further process the json results by some tools i.e. ElasticSearch, if the smaller size of per client json result is desired.
iperf3	Utility to help with basic networking performance tests.

elbencho	Elbencho is a distributed storage benchmark for file systems, object stores & block devices with support for GPUs, developed by VAST Data.
	Source repo: https://github.com/breuner/elbencho
	Docker container registry: https://hub.docker.com/r/breuner/elbencho
	Script install-dep.sh pulls the docker container with following command to the head node only (assume docker has been installed on the system).
	`$ docker pull breuner/elbencho`
	The deployment of Elbencho to client nodes is through elbencho_deploy.sh. For users that don't have a pulling limit, docker pull can be used directly on all client nodes.
	To know how to use Elbencho, refer to help, eg:
	`$ docker run --net=host -it breuner/elbencho –help`
	`$ docker run --net=host -it breuner/elbencho –help-s3`

nfs_mount.sh	A NFS mount helper script from the harness to be deployed to all client nodes, not a true external dependency.
mtu_update.sh	A script from the test harness to be deployed to all client nodes for more convenient MTU update, not a true external dependency. Refer to MTU Update for more details.
general_info.sh	A script from the test harness to be deployed to all client nodes to collect system telemetry as the running context, not a true external dependency.

2.2.4 MTU UPDATE

For best performance, MTU 9000 is recommended for both the 10GbE and 10GbE interfaces. This can be done by following steps, supposing the network interface is eth0 for 10GbE and eth1 for 100GbE.

Important:

1. Please adapt to the proper interfaces if they are different
2. Ensure switch side connected to this interface supports 9000 too, otherwise the system may not be reachable after this update

```
$ source env.conf  # Needed by variable $VAST_SCALE_TESTING_PATH

$ clush -w $ CLIENT_NODES_10G "~/ $VAST_SCALE_TESTING_PATH/mtu_update.sh”

$ clush -w $ CLIENT_NODES_10G “cat /etc/sysconfig/network-scripts/ifcfg-eth0”

$ clush -w $ CLIENT_NODES_100G "~/ $VAST_SCALE_TESTING_PATH/mtu_update.sh eth1”

$ clush -w $ CLIENT_NODES_100G “cat /etc/sysconfig/network-scripts/ifcfg-eth1”
```

2.3 Sanity Check of the Network Connectivity with iperf3

Before running any workloads, it is recommended to verify the proper network connectivity and speed using iperf3.

2.3.1 CHECK THE 10GBE CONNECTIVITY

Step 1: log onto VAST VMS node, note down one of the VIPs on the VMS node, eg 10.A.B.E

Step 2: start iperf3 as the service on the VMS node

```
$ iperf3 -s -p5900
```

Step 3: run iperf3 test script on the head node:

```
$ ./run_iperf_test.sh 10 10.A.B.E
```

This will iterate through all the 10GbE clients as defined in env.conf.override and run iperf3 test one by one using clush.

```
client-cluster-02: Connecting to host 10.A.B.E, port 5900

client-cluster-02: [  5] local 10.A.B.2 port 43376 connected to 10.A.B.E port 5900
```

```
client-cluster-02: [ ID] Interval           Transfer     Bitrate         Retr  Cwnd
client-cluster-02: [  5]   0.00-1.00   sec  1.15 GBytes  9.89 Gbits/sec    0   1.94 MBytes
client-cluster-02: [  5]   1.00-2.00   sec  1.15 GBytes  9.89 Gbits/sec    0   1.94 MBytes
client-cluster-02: [  5]   2.00-3.00   sec  1.15 GBytes  9.91 Gbits/sec    0   2.07 MBytes
client-cluster-02: [  5]   3.00-4.00   sec  1.15 GBytes  9.90 Gbits/sec    0   2.07 MBytes
client-cluster-02: [  5]   4.00-5.00   sec  1.15 GBytes  9.90 Gbits/sec    0   2.07 MBytes
client-cluster-02: - - - - - - - - - - - - - - - - - - - - - - - - -
client-cluster-02: [ ID] Interval           Transfer     Bitrate         Retr
client-cluster-02: [  5]   0.00-5.00   sec  5.76 GBytes  9.90 Gbits/sec    0
sender
client-cluster-02: [  5]   0.00-5.00   sec  5.76 GBytes  9.89 Gbits/sec
receiver
client-cluster-02:
client-cluster-02: iperf Done.
client-cluster-03: Connecting to host 10.A.B.E, port 5900
client-cluster-03: [  5] local 10.A.B.3 port 45376 connected to 10.A.B.E port 5900
client-cluster-03: [ ID] Interval           Transfer     Bitrate         Retr  Cwnd
client-cluster-03: [  5]   0.00-1.00   sec  1.15 GBytes  9.87 Gbits/sec    0   2.15 MBytes
client-cluster-03: [  5]   1.00-2.00   sec  1.14 GBytes  9.76 Gbits/sec    0   2.15 MBytes
client-cluster-03: [  5]   2.00-3.00   sec   975 MBytes  8.18 Gbits/sec    0   2.81 MBytes
client-cluster-03: [  5]   3.00-4.00   sec  1.03 GBytes  8.84 Gbits/sec    0   2.81 MBytes
client-cluster-03: [  5]   4.00-5.00   sec  1.15 GBytes  9.90 Gbits/sec    0   2.81 MBytes
client-cluster-03: - - - - - - - - - - - - - - - - - - - - - - - - -
```

2.3.2 CHECK THE 100GBE CONNECTIVITY

Step 1: log onto VAST VMS node, note down one of the VIP on the VMS node, eg 192.168.B.F

Step 2: start iperf3 as the service on the VMS node, but bind iperf3 to the 100GbE interface IP

```
  $ iperf3 -s -p5900 -B 192.168.B.F
```

Step 3: run iperf3 test script on the head node:

```
  $ ./run_iperf_test.sh 100 192.168.B.F
```

This will iterate through all the 100GbE clients as defined in env.conf.override and run iperf test 1 by 1 using clush.

```
client-cluster-05: Connecting to host 192.168.B.F, port 5900
client-cluster-05: [  5] local 192.168.D.5 port 55085 connected to 192.168.B.F port 5900
client-cluster-05: [ ID] Interval           Transfer     Bitrate         Retr  Cwnd
client-cluster-05: [  5]   0.00-1.00   sec  1.88 GBytes  16.2 Gbits/sec    0    673 KBytes
client-cluster-05: [  5]   1.00-2.00   sec  1.98 GBytes  17.0 Gbits/sec    0    778 KBytes
client-cluster-05: [  5]   2.00-3.00   sec  2.06 GBytes  17.7 Gbits/sec    0    778 KBytes
client-cluster-05: [  5]   3.00-4.00   sec  2.03 GBytes  17.4 Gbits/sec    0    778 KBytes
client-cluster-05: [  5]   4.00-5.00   sec  2.04 GBytes  17.5 Gbits/sec    0    778 KBytes
client-cluster-05: - - - - - - - - - - - - - - - - - - - - - - - - -
client-cluster-05: [ ID] Interval           Transfer     Bitrate         Retr
client-cluster-05: [  5]   0.00-5.00   sec  10.0 GBytes  17.2 Gbits/sec    0
sender
client-cluster-05: [  5]   0.00-5.00   sec  9.99 GBytes  17.2 Gbits/sec
receiver
client-cluster-05:
client-cluster-05: iperf Done.
client-cluster-06: Connecting to host 192.168.B.F, port 5900
client-cluster-06: [  5] local 192.168.D.6 port 43215 connected to 192.168.B.F port 5900
client-cluster-06: [ ID] Interval           Transfer     Bitrate         Retr  Cwnd
client-cluster-06: [  5]   0.00-1.00   sec  1.78 GBytes  15.3 Gbits/sec    0    900 KBytes
client-cluster-06: [  5]   1.00-2.00   sec  1.93 GBytes  16.6 Gbits/sec    0    900 KBytes
client-cluster-06: [  5]   2.00-3.00   sec  1.94 GBytes  16.7 Gbits/sec    0    935 KBytes
client-cluster-06: [  5]   3.00-4.00   sec  2.03 GBytes  17.5 Gbits/sec    0    987 KBytes
client-cluster-06: [  5]   4.00-5.00   sec  2.02 GBytes  17.3 Gbits/sec    0   1.01 MBytes
client-cluster-06: - - - - - - - - - - - - - - - - - - - - - - - - -
client-cluster-06: [ ID] Interval           Transfer     Bitrate         Retr
client-cluster-06: [  5]   0.00-5.00   sec  9.70 GBytes  16.7 Gbits/sec    0
sender
client-cluster-06: [  5]   0.00-5.00   sec  9.69 GBytes  16.7 Gbits/sec
receiver
client-cluster-06:
```

For 100GbE, multiple streams need to be used in order to see the full bandwidth. This is not supported by the script yet.

2.4 Checking Client Performance With Short Bandwidth Test

It might be a good idea to run short BW constant size stepping test on 100GbE client nodes one by one, to ensure each is indeed getting close to 12.5GB of throughput.

```
$ ./run_test.sh -ns 100 -sf ../workload/fio/bw_test_short_100g.sch -co
$ ./run_test.sh -ns 100 -sf ../workload/fio/bw_test_short_100g.sch -ss 1 -st 1
```

The 1st command will write to create the files but no actual read. If the file has been created already before, it will not rewrite. This is equivalent to using "–create_only" parameter in iops_test_100g.sch for some of the workloads. fio_template.ini has "#create_only" and" #" will be removed if either "-co" is defined in the command line, or "–create_only" is defined in the schedule file.

The 2nd command is the actual read BW test.

Read throughput results like below indicate some 100GbE are not getting the expected throughput. Check the client, NIC card and switch configurations as well as cables to resolve the problem first. For example, a few items might impact the BW are:

- Loose installation of the NIC card.
- Client BIOS configuration "CPU Power and Performance Policy": "Performance" is preferred over "Performance Balanced" or other options.
- Client BIOS configuration "Fan Profile": "Performance" is preferred over "Acoustic".
- DRAM size and speed. For 100GbE, higher DRAM speed works better than bigger DRAM size, eg for some CPUs, 192G of DRAM may run at lower frequency than 128G DRAM, hence produce lower performance.

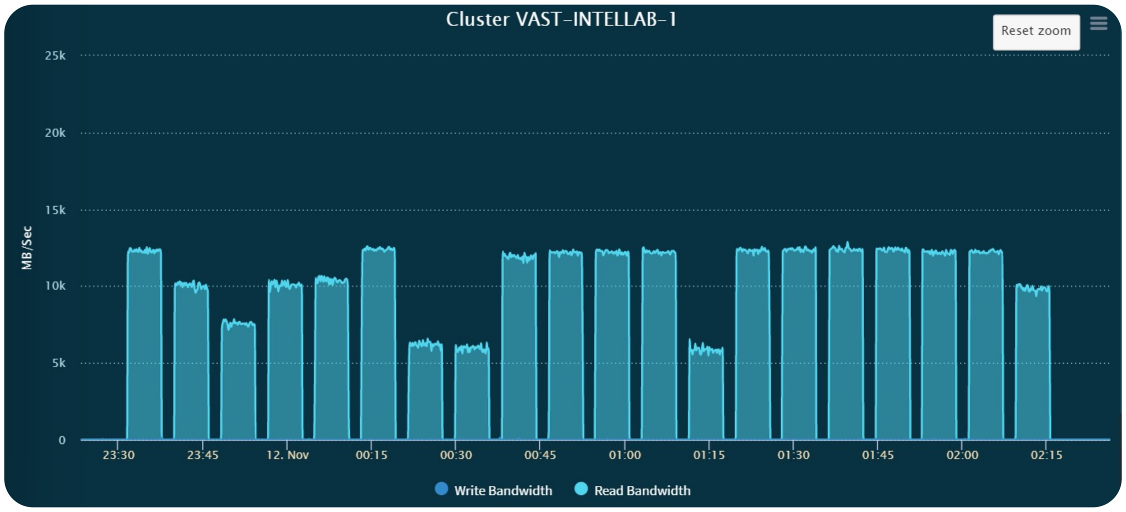


Figure 12: Constant Size Stepping Example

Of course, the same one by one stepping test can also be done on 10GbE to ensure each is getting close to 1.25GB throughput, but with 100 nodes, that will take much longer:

```
$ ./run_test.sh -ns 10 -sf ../workload/fio/bw_test_short_10.sch -co
$ ./run_test.sh -ns 10 -sf ../workload/fio/bw_test_short_10.sch -ss 1 -st 1
```

Doing incremental stepping tests may allow us to generate a quicker view of how the performance scales with the number of nodes. The command below will run a short BW test on 100GbE in steps of 2, 4, 6...20 clients at a time.

```
$ ./run_test.sh -ns 100 -sf ../workload/fio/bw_test_short_100.sch -ss 2
```

This could be helpful to narrow down the slow nodes if the performance doesn't scale as expected. For example, in the screenshot below, we observe step 4 throughput growth is extremely low, and that turns out to be at the step of adding the 6th and 7th slowest client nodes.

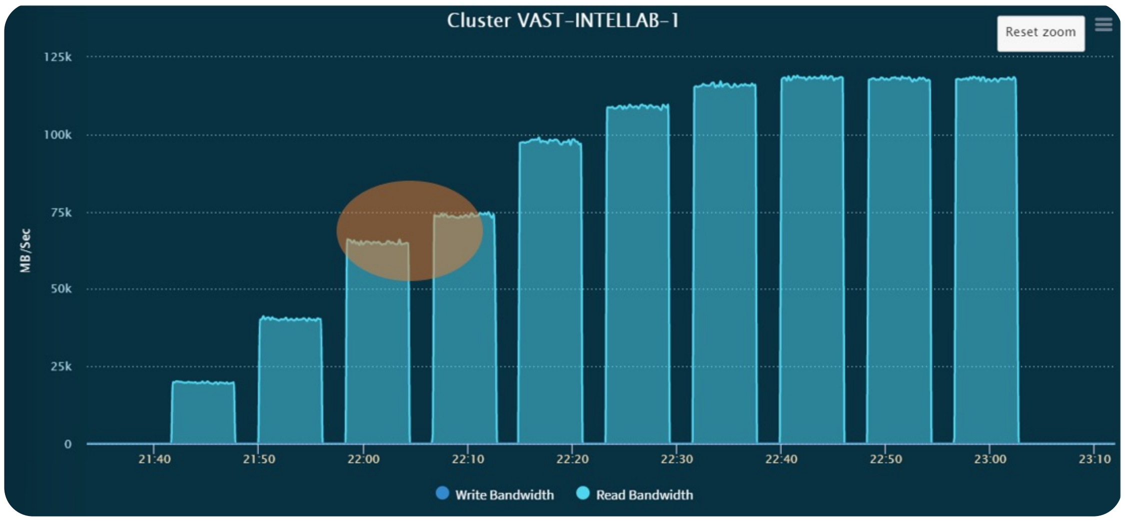


Figure 13: Incremental Stepping Example

However, keep in mind that the performance may not scale linearly when approaching the full BW anyways, even if the nodes are all getting close to full BW throughput individually. Refer to Minimum 100GbE Clients to Saturate for the actual performance scaling trend as the reference on 100GbE.

3) Performance Expectations and Test Results

3.1 Understanding Data Path Bandwidth

To eliminate misleading test results due to architectural bottlenecks, it is important to understand the bandwidth of all components within a given data path. Here are some major factors to be aware of for the hardware configurations used:

Table 10: Data Path Bandwidth

Configuration	**Clients**		**Extreme Switch <-> Arista Switch**		**Arista Cluster <-> VAST**		**VAST Cluster**		**Overall BW**	
	GBit	GB	GBit	GB	GBit	GB	GBit	GB	GBit	GB
100 Clients With 10GbE	100 x 10 = 1000	100 x1.25 = 125	1120 (7s x 4p x 40 Gbit/p)	140 (7s x 4p x 5 GB/s)	1600 (16p x 100 Gbit/p)	1600 (16p x 100 Gbit/p)	960	120	960	120
20 Clients With 100GbE	20 x100 = 2000	20 x 12.5 = 250	N/A	N/A	1600 (16p x 100 Gbit/p)	1600 (16p x 100 Gbit/p)	960	120	960	120

Notes: s – Switch, p – Switch Port

The maximum bandwidth will be limited by the minimum value of the entire data path.

3.2 Performance Test Results

In this section, we'll showcase some BW/IOPS and Latency test results using the predefined workloads from the test harness. All results in this section are averaged across 3 separate runs. The exact command lines to produce the results are listed. To parse the result, please refer to Test Results Explained for more details.

3.3 Bandwidth (1024K) by Data Size

For benchmarking, bandwidth can be data size dependent as it determines where the data will be located, i.e. in Intel® Optane™ write buffer or backend capacity QLC NAND drives. To understand VAST Data's architecture, please refer to UNIVERSAL STORAGE EXPLAINED.

1. Three different data sizes (total data written) were used for the tests:
2. 1.6TB
3. 9.6TB or 10TB
4. 24TB

Command line:

```
$ ./run_bw_test.sh
```

Table 11: BW Summary (1.6TB)

Benchmarking Tool	Workload Type	100 Clients with 10GbE				20 Clients with 100GbE			
		Write		Read		Write		Read	
		GB/s	GiB/s	GB/s	GiB/s	GB/s	GiB/s	GB/s	GiB/s
FIO	Sequential	21.1	19.7	57	53.1	20.1	18.7	53.7	50
	Random	22	20.5	60.5	56.3	22.4	20.9	60.2	56.1
Elbencho File	Sequential	22.3	20.8	57.2	53.3	20.7	19.3	53.3	49.6
	Random	22	20.5	61	56.8	20.9	19.5	58.4	54.4
Elbencho S3	Sequential	21.7	20.2	55.5	51.7	19.1	17.8	53.6	49.9
	Random	N/A	N/A	55	51.2	N/A	N/A	52.7	49.1

Table 12: BW Summary (9.6TB/10TB)

Benchmarking Tool	Workload Type	100 Clients with 10GbE				20 Clients with 100GbE			
		Write		Read		Write		Read	
		GB/s	GiB/s	GB/s	GiB/s	GB/s	GiB/s	GB/s	GiB/s
FIO (9.6T)	Sequential	18.8	17.5	110.9	103.2	18.9	17.6	114.4	106.6
	Random	21.3	19.8	110.9	103.3	21.2	19.7	117.7	109.6
Elbencho File (9.6T)	Sequential	20.4	19	108.7	101.3	19.8	18.4	114.1	106.2
	Random	21.4	19.9	108.7	101.4	21.7	20.2	115	107.1
Elbencho S3 (9.6T)	Sequential	19.8	18.4	106.3	99	18.2	17	110	102.5
	Random	N/A	N/A	106.6	99.3	N/A	N/A	111	103.4

Table 13: BW Summary (24TB)

Benchmarking Tool	Workload Type	100 Clients with 10GbE				20 Clients with 100GbE			
		Write		Read		Write		Read	
		GB/s	GiB/s	GB/s	GiB/s	GB/s	GiB/s	GB/s	GiB/s
FIO	Sequential	18.5	17.3	108.7	101.2	18.4	17.2	115.5	107.5
	Random	20.7	19.3	109.4	101.9	21.1	19.7	116.5	108.5
Elbencho File	Sequential	19.5	18.2	108.5	101	19.8	18.5	109.7	104.5
	Random	21	19.6	108.9	101.5	21.2	19.7	110.1	104.9
Elbencho S3	Sequential	18.2	17	105.8	98.5	17.3	16.1	110.7	103.1
	Random	N/A	N/A	104	96.9	N/A	N/A	110.2	102.6

Notes:

The bar graph below shows the comparison of random read BW by data size of 100GbE and 10GbE, with 3 different benchmarking methods: FIO, Elbencho File store over NFSv3 protocol, and Elbencho S3 Object store over S3 protocol.

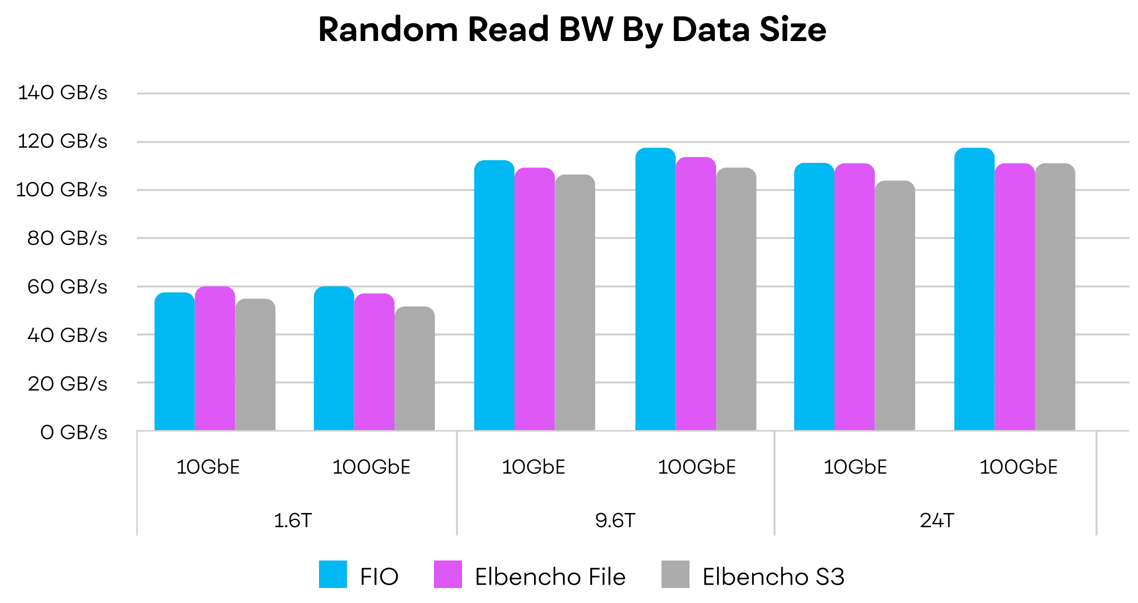


Figure 14: Random Read BW by Data size

Highlights:

- File Store:
  - 10G: FIO and Elbencho results are comparable.
  - 100G: Elbencho results are a little lower.
- S3 access: only slightly lower (~2-3%) than NFS-based access.
- Efficiency using 9.6TB FIO result:
  - 20 x 100GbE clients: 117.7GB/s (max theoretical of 120GB/s) = 98.0%.
  - 100 x 10GbE clients: 110.9GB/s (max theoretical of 120GB/s) = 92.4%.
- Read bandwidth is largely determined by where the data is persisted:
  - 1.6T: data is still in Intel® Optane™, bandwidth is lower due to the small Intel® Optane™: QLC NAND ratio (2.9%).
  - 9.6T/24T: data are mostly in capacity QLC NAND. Bandwidth is much higher with high concurrency.

Important: in real-world scenarios, the lower bandwidth of small data set sizes is rarely a concern as subsequent writes keep pushing previously written data to QLC NAND.

3.4 Bandwidth Saturation Test: 100GbE Clients

Command line:

```
$ ./run_fio_bw_test_short.sh
```

The theoretical maximum bandwidth of a single 100GbE is 12.5GB/s, which means that less than 20 clients are needed to saturate the maximum theoretical bandwidth (120GB/s) of this network configuration.

Table 14: 1024KB Random Read BW by Number of 100GbE Clients

Client	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
Actual BW	24.5	48.5	71.6	92.2	107.7	113.7	116.5	117.4	117.6	117.3
Theoretical BW	25	50	75	100	120	120	120	120	120	120

Figure 15 shows the plot of scaling trend from data in Table 14.

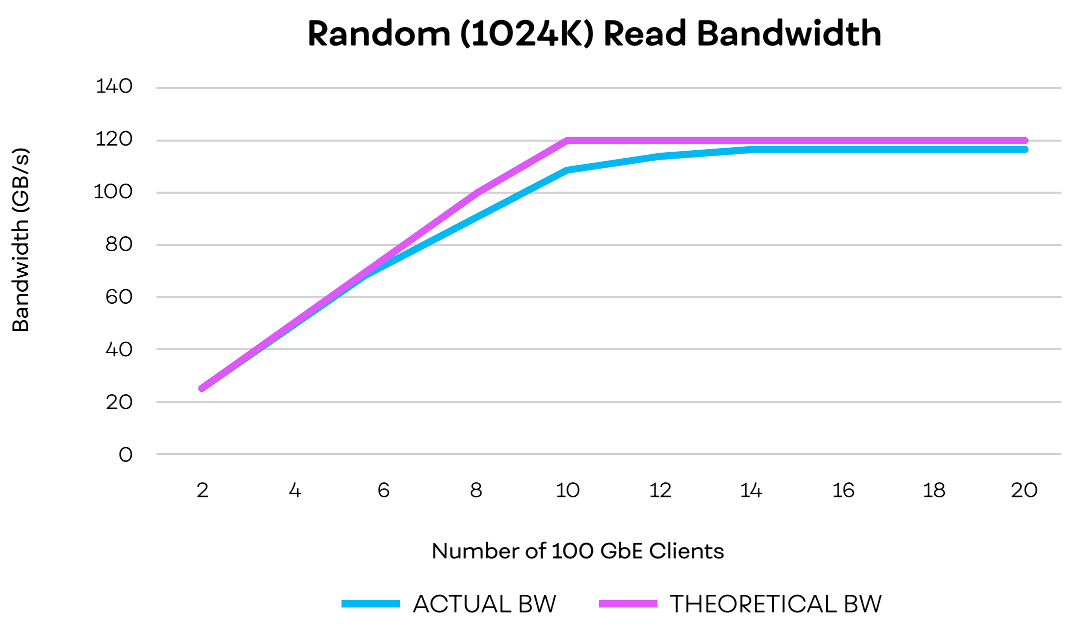


Figure 15: Random (1024K) Read BW by Number of 100GbE Clients

Highlights:

- Bandwidth scales linearly up to 8 clients, as it approaches max theoretical bandwidth of the cluster.
- It takes ~14-16 clients to saturate this hardware configuration.

3.5 Bandwidth Saturation Test: 10GbE Clients

Command line:

```
$ ./run_fio_bw_test_short.sh
```

Likewise, the theoretical maximum bandwidth of a single 10GbE is 1.25GB/s.

Table 15: Random (1024K) Read BW by Number of 10GbE Clients

Client	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
Actual BW (GB/s)	12.3	24.7	37	48.9	61.3	73.6	80.9	92.6	102.9	110.5
Theoretical BW (GB/s)	12.5	25	37.5	50	62.5	75	87.5	100	112.5	120

Figure 16 shows the plot of scaling trend from data in Table 15.

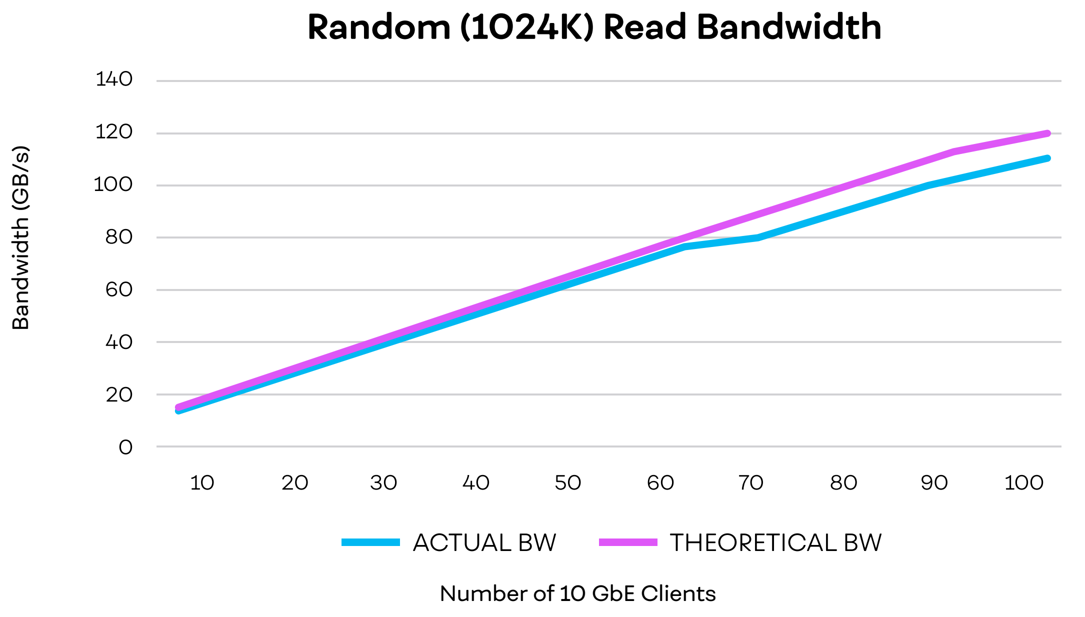


Figure 16: Random (1024K) Read BW by Number of 10GbE Clients

Highlights:

- Bandwidth scales linearly from 10 to 100 clients (in steps of 10).
- With 100 10GbE clients, total throughput reaches 92% utilization of max theoretical bandwidth.

3.6 S3 Random Read Bandwidth by Object Size

In addition to NFS-based performance testing, a S3 bandwidth test across various object sizes was also conducted. The results are detailed in Figure 17.

Command line:

```
$ ./run_elbencho_s3_sweep_iosizes.sh
```

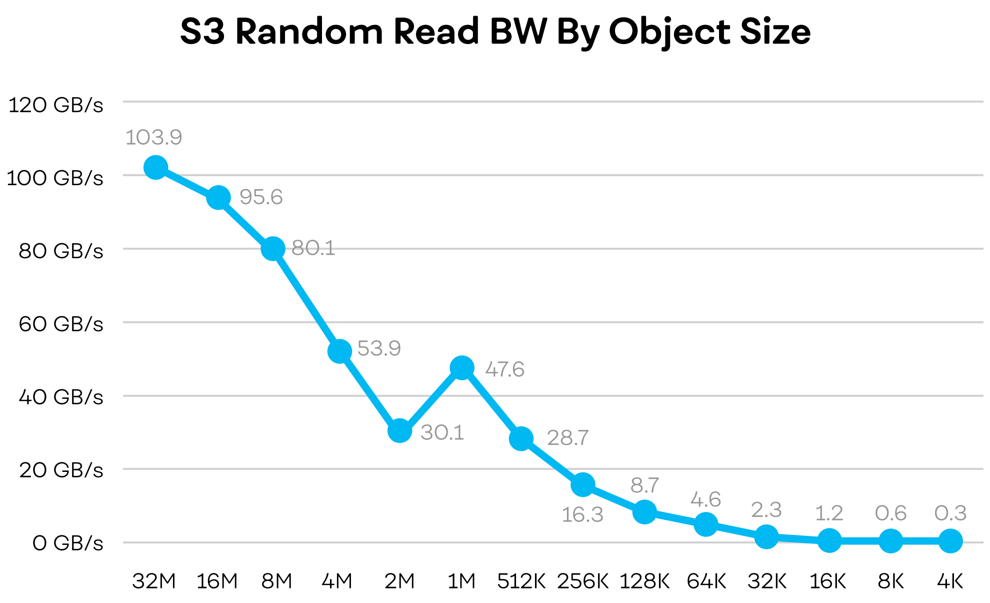


Figure 17: S3 Random Read BW by Object size

Highlights:

- Bandwidth goes up as object size increases in general, except for a dip at 2MB object size.

Notes

- Special tuning by VAST Data may resolve this dip using 2MB objects. Please contact support@vastdata.com if better performance for 2MB is needed.

3.7 Random (4K) IOPS Test

Command line:

```
$ ./run_iops_test.sh
```

Table 16: Random IOPS Summary

Benchmarking Tool	BS	Workers	iodepth	100 Clients with 10GbE			100 Clients with 10GbE		
				Write IOPS (K)	Read IOPS (K)		Write IOPS (K)	Read IOPS (K)	
					Before Flush	After Flush		Before Flush	After Flush
FIO	4K	4	64	309	895	493	347	944	477
		4	16	301	903	532	301	830	473
Elbencho File	4K	4	64	300	866	613	299	780	558
		4	16	283	875	614	263	773	529
Elbencho S3	4K: read 5M: write	4	N/A	4	120	95	2	31	24

Figure 18 shows the plot of the Random Read IOPS from Table 16.

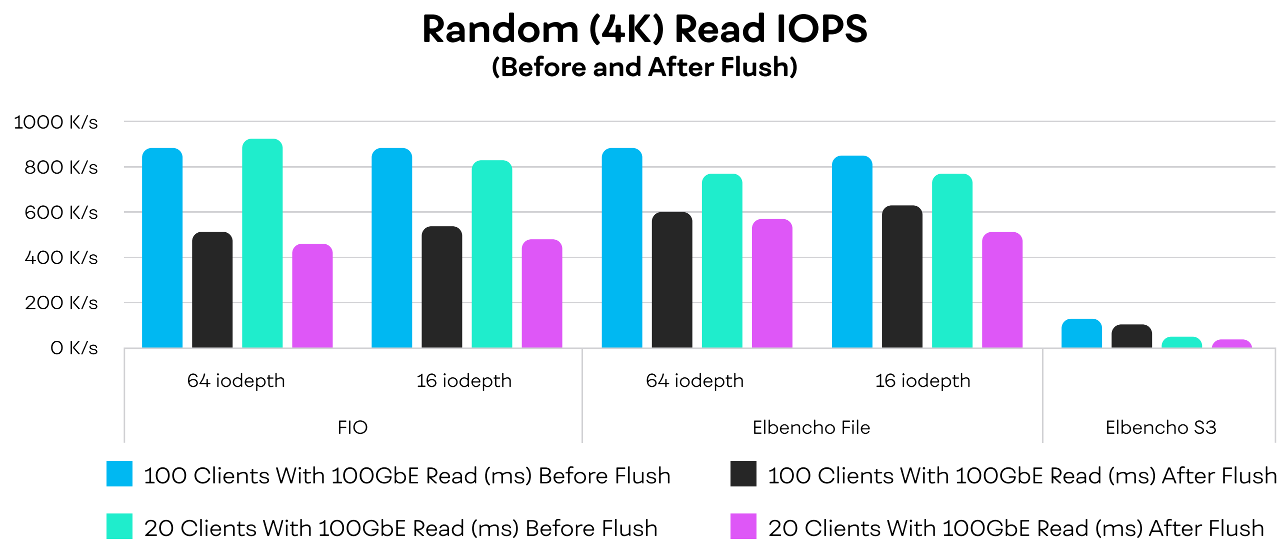


Figure 18: Random (4K) Read IOPS Before and After Flush

Notes:

- Before flush: data in Intel® Optane™ (data size < write buffer threshold)
- After flush: data in QLC NAND (data size > write buffer threshold)

Highlights:

- Random Read IOPS is 69%~97% higher before flush (FIO), due to Intel® Optane™ SSD's high IOPS.

3.8 Random (4K) Read Latency Test

Latency results are collected with the same IOPS test command in the previous section.

Table 17: Random Average Latency Summary

Benchmarking Tool	**BS**	**Workers**	**iodepth**	**100 Clients with 10GbE**			**100 Clients with 10GbE**		
				Write IOPS (K)	Read IOPS (K)		Write IOPS (K)	Read IOPS (K)	
					Before Flush	After Flush		Before Flush	After Flush
FIO	4K	4	64	82.5	28.5	51.8	15	5.4	10.7
		4	16	21.1	7	12	4.2	1.5	2.6
Elbencho File	4K	4	64	84.1	29.6	41.5	17.3	6.5	9.1
		4	16	22.5	7.2	10.4	4.8	1.6	2.4
Elbencho S3	4K: read 5M: write	4	N/A	97.3	3.3	4.1	42.9	2.5	3.2

Figure 19 shows the plot of the latencies from Table 17.

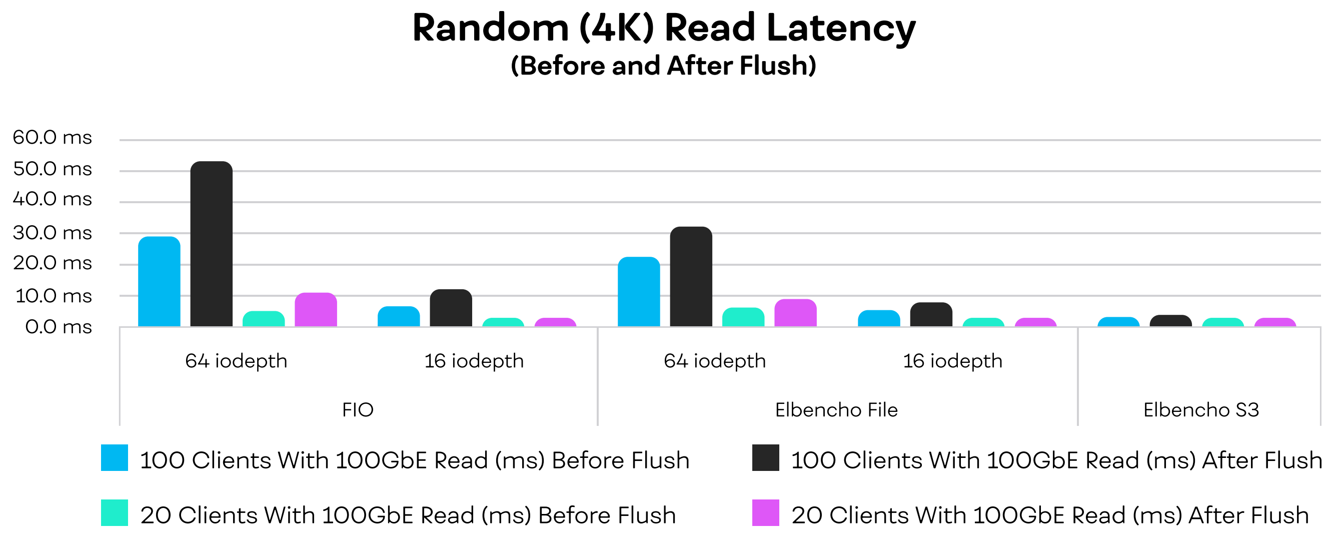


Figure 19: Random (4K) Read Latency Before and After Flush

Notes:

- Before flush: data in Intel® Optane™ (data size < write buffer threshold)
- After flush: data in QLC NAND (data size > write buffer threshold)

Highlights:

- Latency is ~41.6% ~49.5% lower before flush (FIO), due to the low latency characteristics of Intel® Optane™ SSD's

3.9 Performance Impact of Intel® Optane™

Intel® Optane™ SSD is used as a write buffer in-between CNode and QLC NAND on DNodes in VAST Data's solution, and is a key component of the architecture due to its high BW/IOPS, low latency, high endurance and persistence:

- It stores all metadata and speeds up user data access to QLC NAND, as metadata indicates where the user data is located. Slower metadata access would slow user data access to QLC NAND, hindering the backend access performance.
- It enables VAST Data's Similarity-based data reduction: algorithm requires many fast lookups in hash tables that are too big for DRAM and needs to be persistent.
- It enables data center use-cases for QLC NAND by extending flash lifetime from 5 years to 10 years, providing customers with a dramatically improved TCO.

4) Schedule Files and Running Customized Tests

Schedule files usually fall into 4 different categories:

- BW tests with 1024KB
- IOPS tests with 4KB
- Workers/threads & iodepth sweep tests with 1024KB
    - These are intended to find the optimal configuration for the BW test.
    - Our test result shows 16 workers / 8 iodepth is the optimal configuration
- iosize sweep tests
    - These are intended to do BW tests with the optimal workers/threads * iodepth configuration for iosizes in power of 2.
    - Elbencho File and S3 focus more on 100% read test
    - FIO also has iosize sweep test for:
        - 70% read + 30% write per host test, e.g. to run 70/30 mixed workload on each 100GbE client, use the command below:

```
$ ./run_test.sh -ns 100 -sf ../workloads/fio/sweep_iosizes_70rd_random_100g.sch -co
$ ./run_test.sh -ns 100 -sf ../workloads/fio/sweep_iosizes_70rd_random_100g.sch
```

        - 70% host read + 30% host write test, e.g. to run read workload on 70% 100GbE clients, write workload on 30% of the 100GbE clients, use the command below:

```
$ ./run_test.sh -ns 100 -sf ../workloads/fio/sweep_iosizes_70hostrd_random_100g.sch -co
$ ./run_test.sh -ns 100 -sf ../workloads/fio/sweep_iosizes_70hostrd_random_100g.sch
```

    - Running FIO tests on 100GbE and 10GbE simultaneously is also possible by opening 2 consoles and starting 1 test in each console, providing CLIENT_NODES_10G and CLIENT_NODES_100G defined do not overlap.

Example wrapper scripts of how to use the schedule files are provided in the scripts folder. Check.

5) Appendix

All test results in this Guide are tested with changeset listed in Software Configurations. For quick referencing, the actual schedule file contents of that particular changeset are also listed here.

5.1 References

UNIVERSAL STORAGE EXPLAINED, by VAST Data

VAST_Data-Overview.pdf, by VAST Data, Feb 2021

VAST Data Documentation, by VAST Data

VAST Data Knowledge Base, by VAST Data

Source Repo: https://github.com/breuner/elbencho, by Sven Breuner, VAST Data

Docker Registry: https://hub.docker.com/r/breuner/elbencho, by Sven Breuner, VAST Data

https://www.arista.com/assets/data/pdf/Datasheets/7170-Datasheet.pdf, by Arista Networks, Inc, 2021

https://www.ibm.com/cloud/blog/object-vs-file-vs-block-storage, By IBM Cloud Education, October 2021

5.2 Acronyms and Glossaries

Name	Description
NFS	Network File System
S3	Simple Storage Service
VIP	Virtual IP
VMS	VAST Management System
SSD	Solid State Drive
Intel® Optane™	Intel's revolutionary memory and storage innovation, built with a ground-breaking combination of endurance, consistent high performance, and low latency is designed to bring new computing possibilities to a variety of markets.
CIFS	Common Internet File System (CIFS) is a network file system protocol used for providing shared access to files and printers between machines on the network.
File Store	File storage is when all the data is saved together in a single file with a file extension type that's determined by the application used to create the file or file type, such as .jpg, .docx or .txt. For example, when you save a document on a corporate network or your computer's hard drive, you are using file storage. Files may also be stored on a network-attached storage (NAS) device. These devices are specific to file storage, making it a faster option than general network servers. Other examples of file storage devices include cloud-based file storage systems, network drives, computer hard drives and flash drives.
Object Store	Object storage is a system that divides data into separate, self-contained units that are re-stored in a flat environment, with all objects at the same level. There are no folders or sub-directories like those used with file storage. Additionally, object storage does not store all data together in a single file. Objects also contain metadata, which is information about the file that helps with processing and usability. Users can set the value for fixed-key metadata with object storage, or they can create both the key and value for custom metadata associated with an object.
Block Store	Block storage is when the data is split into fixed blocks of data and then stored separately with unique identifiers. The blocks can be stored in different environments, such as one block in Windows and the rest in Linux. When a user retrieves a block, the storage system reassembles the blocks into a single unit. Block storage is the default storage for both hard disk drive and frequently updated data. You can store blocks on Storage Area Networks (SANs) or in cloud storage environments.

5.3 run_test.sh/run_test_no_runlog.sh command line options

Command Line Option	Description
-co|--create-only	For creating the FIO files only, no actual read access. Default: 0.
-dr|--dryrun	Do not run workload. Everything else will be executed the same as a normal run including nfs mounts and starting fio and Elbencho services on clients.
-ns|--network-speed	GbE speed, valid values ["10", "100"]. Default: 10
-pf|--prefix	Extra prefix to be added to the test name
-sf|--schedule-file	Specify the schedule file to run.
-sm|--skip-mount	Specify this to skip the mount step. This can speed up the test if the mount is the same as the previous test.
-sq|--skip-all-client-config-query	Do not do query of all client configurations before the test. As query takes a few minutes, this allows the test to be started sooner for quicker benchmarking.
--ss|--step-size	Specify the number of clients to increase to run the test. Default: max clients. Examples: when st==0: if max clients = 100, then "-ss 10" will test clients in steps of 10, 20, 30…90, 100 clients if max clients = 100, then "-ss 15" will test clients in steps of 15, 30, 45…75, 90, 100 clients if max clients = 100, then "-ss 0" or "-ss 100" or omitting -ss option will test all 100 clients in 1 step when st==1: if max clients = 10, then "-ss 1 -st 1" will test the 10 clients 1 at a time, in 10 steps if max clients = 10, then "-ss 2 -st 1" will test the 10 clients 2 at a time, in 5 steps
-st|--step-type	0 - incremental (default), 1 - constant. Refer to -ss|--step-size examples to see the differences between the two types
--runtime	FIO runtime. Default: as specified in template FIO job file
--ramp_time	FIO ramptime. Default: as specified in template FIO job file
-h|--help	To print this help

5.4 Test Results Explained

5.4.1 EXAMPLE FIO RESULT

```
results_2021-11-13-21.41.22_20Clients_bw_test_short_100g_headnode
├── bw_test_short_100g.zip          => folder containing actual fio job files per client,
 |                                      zipped to reduce size
├── bw_test_short_100g.sch          => schedule file for this test
├── context.zip                     => running contexts, zipped to reduce size
├── env.conf                        => env.conf for this test
├── env.conf.override               => env.conf.override for this test
├── results_2021-11-13-21.41.22_20Clients_bw_test_short_100g_headnode_Logs.zip
 |                   => Per client FIO json+ results.
 |                   For aggregated results, check the "All Clients" sub folder.
 |                   The json+ file names match their FIO job file names,
 |                   except for extension (.ini for job file and .json for json+ result.)
 |                   Identify each workload by the matching job filename.
└── run.log          => console output for this test
```

FIO job filename is auto generated based on mandatory parameters defined by the workload, eg:

a) --rw=randread --numjobs=16 --iodepth=8 --bs=1024k --size=30g --runtime=300 --ramp_time=60

This will generate: rd_rnd_qd_128_1024k_16w.ini, where:

128: equals to numjobs 16 x iodepth 8.

1024k: the block size.

16: equals to numjobs.

rd_rnd: random read.

b) --rw=read --numjobs=16 --iodepth=8 --bs=1024k --size=30g --runtime=300 --ramp_time=60

This will generate: rd_qd_128_1024k_16w.ini, where:

rd: sequential read (as no rnd before qd).

c) --rw=read --numjobs=16 --iodepth=8 --bs=1024k --size=30g --runtime=300 --ramp_time=60 –tag=1

This may generate: rd_qd_128_1024k_16w_5tag.ini, where:

5: the workload number in the .sch, starting from 1.

5 means this is the 5th workload in the schedule file.

Note that –name parameter only affects the FIO generated data filename, not FIO job filename.

5.4.2 EXAMPLE ELBENCHO FILE RESULT

Here is an example result folder for Elbencho file testing:

```
results_2021-11-10-07.11.45_20Clients_bw_test_100g_headnodename
├── bw_test_100g.sch           => schedule file for this test
├── context.zip                => running context for this test, zipped to reduce size
├── elbencho
│   └── file
│       ├── rd-1024kbs-16t-8iodepth_rnd.csv         => result csv files
│       ├── wr-1024kbs-16t-8iodepth.csv
│       └── wr-1024kbs-16t-8iodepth_rnd.csv
├── env.conf                         => config file for this test
├── env.conf.override                => env.conf.override for this test
└── run.log                          => console output
```

Note that the results of multiple workloads in a schedule file might collapse to the same .csv file. Identify the workload by the attributes of the workloads, eg threads, size, bs size, random or not etc as defined in the schedule file for each workload. If 2 workloads have the same attributes, then identify by the order of the workload and the order or runtime in the .csv.

5.4.3 EXAMPLE ELBENCHO S3 RESULT

```
results_2021-11-11-01.04.48_100Clients_iops_test_10g_redwood-bdw106_run3
├── context.zip     => running context for this test, zipped to reduce size
├── elbencho
│   └── s3
│       ├── rd-4kbs-4t-iodepth.csv                => result .csv files
│       ├── wr-32mbs-16t-iodepth.csv
│       └── wr-5mbs-4t-iodepth.csv
├── env.conf                  => configuration file for this test
├── env.conf.override         => env.conf.override for this test
├── iops_test_10g.sch         => schedule file for this test
└── run.log                   => console output for this test
```

The same as the Elbencho file result, the results of multiple S3 workloads in a schedule file might collapse to the same .csv file as well. Identify the workload by the attributes of the workloads, e.g. threads, size, bs size, random or not etc as defined in the schedule file for each workload. If 2 workloads have the same attributes, then identify by the order of the workload and the order or runtime in the .csv.