HAMMERSPACE

MLPerf Storage v1.0

Configurating for testing using AWS

Table of Contents

Executive Summary

This document explains how to create a Hammerspace testing environment using the MLPerf Storage Benchmark Suite v1.0.

MLPerf is a collection of performance benchmarks published by the MLCommons, an open, AI-focused engineering consortium. The MLPerf Storage benchmark suite is designed to evaluate the ability of storage systems to deliver data to GPUs (aka accelerators). This version, v1.0, uses three simulated machine learning (ML) workloads.

Benchmark Details

The MLPerf Storage Benchmark Suite contains three workloads:

- **3D-Unet:** Simulates a visual ML workload segmenting medical images
- **RESNET50:** A convolutional neural network (CNN) that excels at image classification. It's like a highly trained image analyst who can dissect a picture, identify objects and scenes within it, and categorize them accordingly
- **Cosmoflow:** A deep learning-based approach for analyzing and predicting the large-scale structure of the universe from cosmological simulations

Testers may choose to run one, two, or all three of the workloads.

There are two “divisions” defined by MLCommons for testing, Open and Closed. Open division submissions have more flexibility for tuning both the benchmark and the storage system configuration, and results are not directly comparable. Closed division rules establish a level playing field by more strictly dictating configuration parameters, which allows submissions from different vendors to be compared.

The configurations outlined in this document were used to perform all tests under Closed division rules.

Four tests were run with each workload:

- Single client simulating multiple NVIDIA A100 GPU’s
- Multiple clients simulating multiple NVIDIA A100 GPU’s
- Single client simulating multiple NVIDIA H100 GPU’s
- Multiple clients simulating multiple NVIDIA H100 GPU’s

For each test, the goal is to determine the maximum number of simulated NVIDIA GPUs the tested storage system can support. Each GPU must be supplied with sufficient data to keep its utilization above 90%. This highlights storage throughput and latency.

Test System Configuration

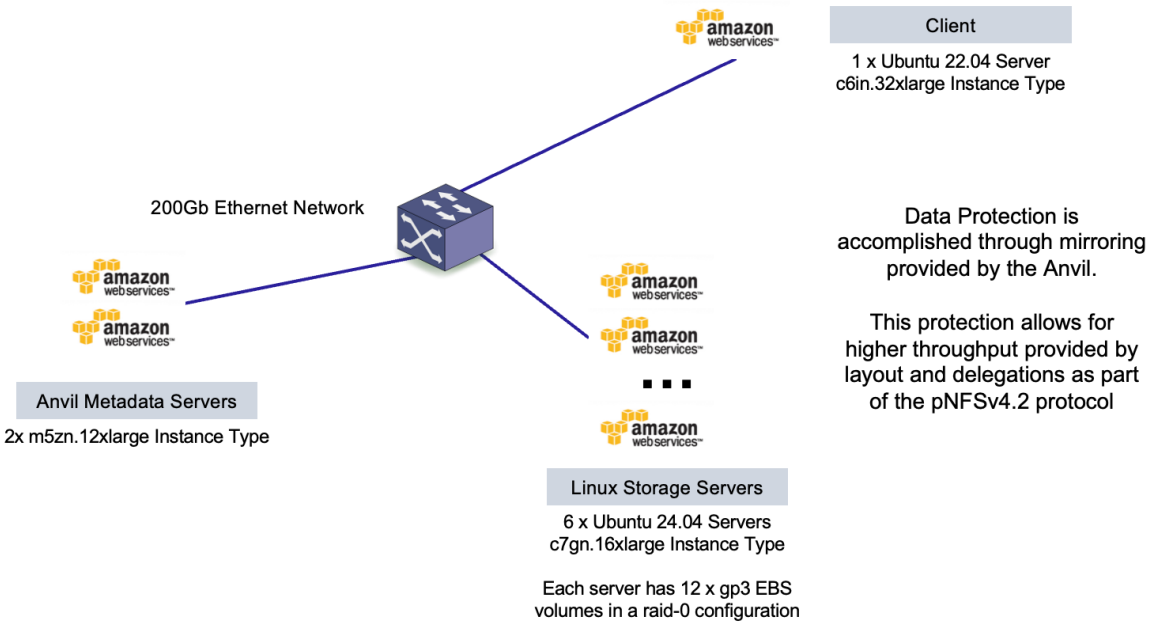

Figure 1 - Hammerspace Test System Architecture – Single Client

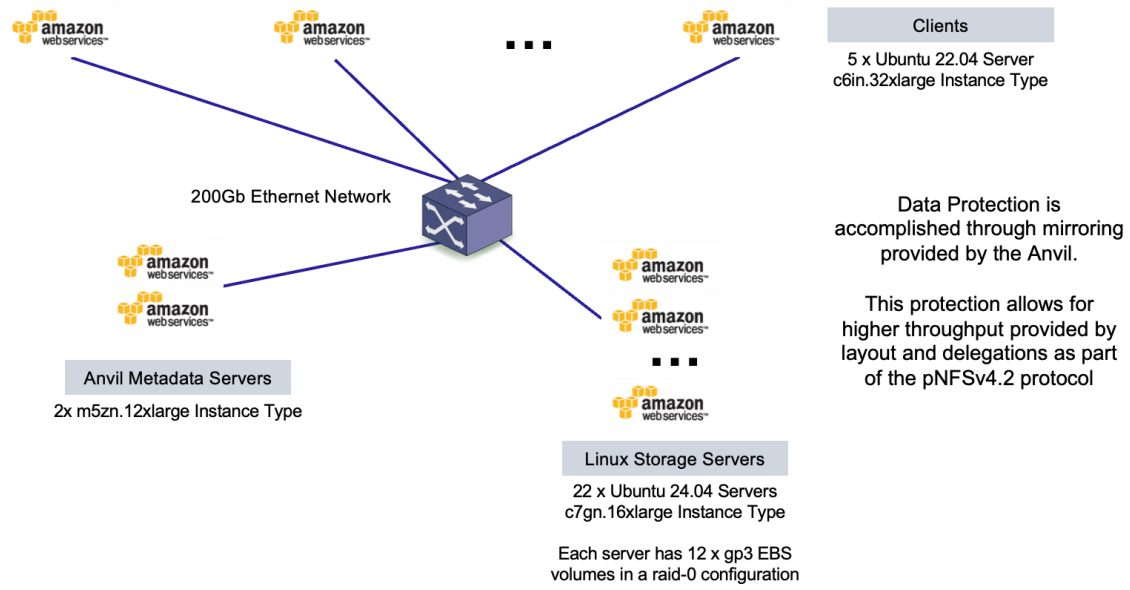

Figure 2 - Hammerspace Test System Architecture – Multiple Client

All the testing occurred within Amazon Web Services (AWS).

The Hammerspace system consists of two redundant Anvil metadata nodes in an active/passive configuration and up to twenty-two Storage Nodes. The Anvil is responsible for metadata operations and cluster coordination tasks, while the Storage System nodes serve test data using GP3 EBS (SSD) storage devices.

As mentioned above, two types of tests were run with two different GPU types.

The single client tests utilized an architecture as outlined in Figure 1 (above). A single client was used to run the tests, mounting a Hammerspace share via parallel NFSv4.2. All single client tests utilized six Storage System Nodes.

The multiple client tests utilized an architecture as outlined in Figure 2 (above). Five clients were used to run the tests, mounting a Hammerspace share via parallel NFSv4.2. All multiple client tests utilized twenty-two Storage System Nodes.

All clients and storage servers were connected via a 200GbE network. The Anvil nodes (metadata servers) were connected to both the clients and storage servers via a 100GbE network. Since the Anvils only supply file layouts and delegations, it was determined that a 200GbE network connection was not needed.

The hardware configuration is detailed in the tables below.

Resource	Specification – Anvil
AWS Instance Type	m5zn.12xlarge
CPU Type	Intel Xeon Platinum 8252
CPU Cores	48 x Virtual CPUs
CPU Frequency	3.8 GHz
Memory	192 GB
Boot disk	200GB GP3 EBS Volume (SSD)
Drives for metadata storage (Anvil)	1TB IO2 EBS Volume (SSD)
Network adapters	100 Gb/s virtual networking

Table 1 – Anvil (metadata server) Hardware Specification

Resource	Specification – Storage Server
AWS Instance Type	c7gn.16xlarge
CPU Type	AWS Graviton3 Processor
CPU Cores	64 x Virtual CPU's
CPU Frequency	2.6 GHz
Memory	128 GB
Boot disk	125GB GP3 EBS Volume (SSD)
Drives for data storage	12 x 500GB GP3 EBS Volume (SSD)
Network adapters	200 Gb/s virtual networking
Operating System	Ubuntu 24.04 LTS

Table 2 – Storage Server Specification

Resource	Specification – Client
AWS Instance Type	C6in.32xlarge
CPU Type	Intel Xeon 8375C (Ice Lake)
CPU Cores	128 x Virtual CPU's
CPU Frequency	2.9 GHz
Memory	256 GB
Boot disk	200GB GP3 EBS Volume (SSD)
Network adapters	200 Gb/s virtual networking
Operating System	Ubuntu 22.04 LTS

Table 3 - Client Specification

Software

Anvil

The Anvil nodes ran Hammerspace version 5.1.6-166. As previously mentioned, the Anvil controls the layout and delegation of files. A client, through the parallel NFSv4.2 mount, will request that a file be created or opened. The Anvil will determine the correct volume on a specific storage server in which to layout that data. Once that is completed, the Anvil will delegate the remainder of all file operations to the client. At this point, the Anvil is "divorced" from the file operations. The client communicates directly with the volume on the specific storage server.

To provide data protection and higher throughput, only one parameter was added to the Hammerspace share. Parameters added to any share are referred to as ***Objectives***.

The objective added to the mlperf share was availability-3-nines. This objective directs the Anvil to layout multiple copies of any file stored within the mlperf share. Each copy of the data is stored on separate volumes on different storage servers. This provides for higher performance as delegations to each of the duplicate files can be handed out to the clients. Because of the multiple copies of the data on completely different storage servers, any failure in a storage volume or a complete storage server would not affect data availability.

Clients

Each client was loaded with Ubuntu Server version 22.04.4 LTS. Each client communicated with the Anvil and the Storage Servers through NFS. The mount command used was:

mount -t nfs -o vers=4.2,port=20492,nconnect=16 IP_OF_ANVIL:/mlperf /mnt/mlperf

Storage Servers

The storage server was loaded with Ubuntu Server 24.04 LTS. Each storage server included twelve 500GB SSD drives that were used for data storage. To optimize performance, the twelve SSD drives were combined into a single volume running raid-0 (striping).

Additionally, the NFS server was tuned to allow for more nfs daemons.

The steps used for each storage server were:

```
#!/bin/bash

sudo apt update
sudo apt install -y net-tools nfs-common nfs-kernel-server sysstat

# Build Raid-0 volumes

sudo mdadm --create --verbose /dev/md0 --level=0 --raid-device=12 /dev/nvme1n1 /dev/nvme2n1
/dev/nvme3n1 /dev/nvme4n1 /dev/nvme5n1 /dev/nvme6n1 /dev/nvme7n1 /dev/nvme8n1 /dev/nvme9n1
/dev/nvme10n1 /dev/nvme11n1 /dev/nvme12n1

# Create filesystem

sudo mkfs -t xfs /dev/md0
```

```
# Create mountpoints

sudo mkdir /hsvol1

# Put mountpoints into fstab

echo "/dev/md0 /hsvol1 xfs defaults,nofail,discard 0 0" | sudo tee -a /etc/fstab

# Mount filesystem

sudo mount /hsvol1

# Change permissions on mountpoint so everyone can write to it

sudo chmod 777 /hsvol1

# Put mountpoints into exports for NFS

echo "/hsvol1       *(rw,sync,no_root_squash,no_subtree_check)" | sudo tee -a /etc/exports

sudo cat << EOF | sudo tee /etc/nfs.conf.d/local.conf
[nfsd]
threads = 64

[mountd]
manage-gids = 1
EOF

sudo cat <<EOF | sudo tee /etc/default/nfs-kernel-server
# Number of servers to start up

RPCNFSDCOUNT=64

# Runtime priority of server (see nice(1))

RPCNFSDPRIORITY=0

# Options for rpc.mountd.
# If you have a port-based firewall, you might want to set up
# a fixed port here using the --port option. For more information,
# see rpc.mountd(8) or http://wiki.debian.org/SecuringNFS
# To disable NFSv4 on the server, specify '--no-nfs-version 4' here

RPCMOUNTDOPTS="--manage-gids"

# Do you want to start the svcgssd daemon? It is only required for Kerberos
# exports. Valid alternatives are "yes" and "no"; the default is "no".

NEED_SVCGSSD=""

# Options for rpc.svcgssd.

RPCSVCGSSDOPTS=""

EOF
```

```
# Upgrade and reboot

sudo apt upgrade -y
sudo reboot
```