Tier 0 Deployment Guide

HAMMERSPACE

Table of Contents

About This Documentation

Hammerspace unlocks a new tier of storage by transforming existing local NVMe storage on GPU servers into a Tier 0 of ultra-fast, persistent shared storage. By activating this previously 'stranded' local NVMe storage seamlessly into the Hammerspace Global Data Platform, Tier 0 delivers data directly to GPUs at local NVMe speeds, unleashing untapped potential and redefining both GPU computing performance and storage efficiency.

This innovation works with existing on-premises and/or cloud-based GPU (or CPU/ARM) servers from any vendor, by activating internal local NVMe capacity that has already been paid for.

This ultra-fast local NVMe capacity - which is largely underutilized - is a sunk cost, since it is already included within the price of GPU servers in both on-premises and cloud-based deployments. The fact that this also doesn’t add any additional power requirements is also a net benefit for data centers who are near the limit of their available power.

Activating this local NVMe as Tier 0 shared storage is done leveraging intelligence built into the Linux kernel, which is included in all standard distributions. This means that servers do not require installation of proprietary client software, or any other alterations as they are delivered from manufacturers.

Moreover, utilizing this local Tier 0 capacity reduces or can even eliminate the costs of adding extreme-performance networking infrastructure and/or additional external high-performance storage systems, all of which are very expensive ways of trying to lessen the impact of the network bottleneck when trying to keep the GPUs fully utilized.

Tier 0 delivers these benefits while also providing global multi-protocol file/object access, non-disruptive data orchestration, data protection, and other enterprise-class data services across a global namespace that unifies storage silos from any vendor in one or more locations, including hybrid cloud environments, accelerating time to value for AI investments.

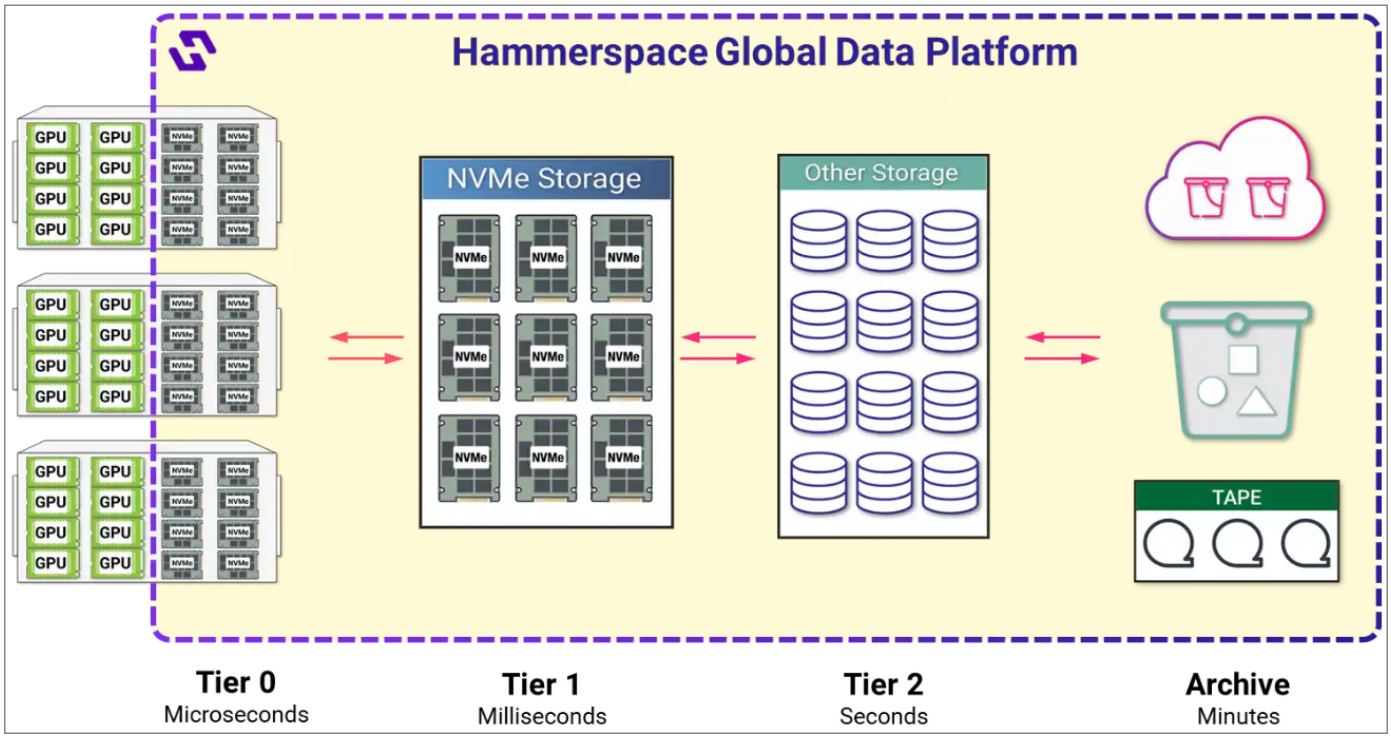

Figure 1. About this documentation

With Tier 0, organizations can fully leverage existing local NVMe on GPU servers as a shared resource, activating this higher-performing tier with the redundancy and protection of conventional external storage tiers. Moreover, utilizing this local Tier 0 capacity reduces or can even eliminate the costs of adding extreme-performance networking infrastructure and/or additional external high-performance storage systems, all of which are very expensive ways of trying to lessen the impact of the network bottleneck when trying to keep the GPUs fully utilized.

For support resources, see the Hammerspace Support Hub and the Hammerspace Licensing and Delivery Portal.

> Do not install additional or third-party software, agents, or packages directly on Hammerspace Anvil (metadata) or DSX (data) nodes. These nodes run a managed appliance software stack; anything installed outside that stack is unsupported and is removed during a software upgrade. Such software can also interfere with node operation. If you need monitoring or integration with an external tool, contact Hammerspace Support for a supported approach.

Chapter 1. What Is Tier 0?

Tier 0 means making use of otherwise stranded or inefficiently used storage devices in compute nodes (including GPU and CPU based compute nodes). The name is taken from the idea that storage systems outside of the GPU nodes make up traditional tiers, starting with tier 1. Tier 0 is simply the tier closest to the compute node - right inside it.

There are multiple ways these devices can be put to work, increasing efficiency in multiple dimensions which may include simply using the space, through making the whole AI job more efficient by putting data used by the GPU nodes running the job on the nodes actually doing the work.

It's important to understand the distinction between Tier 0 nodes and Linux Storage Server (LSS) nodes. Tier 0 nodes include the compute function such as GPUs. The only storage service they provide is exporting (NFS term for sharing) a number of internal storage devices as file systems managed by Hammerspace, but consumed by Tier 0 compute nodes. GPU nodes are usually sold in a specific server configuration including a set of internal storage devices, usually NVME, and run an operating system dictated by the GPU vendor or systems integrator. For example, Nvidia based platforms typically require Ubuntu as the operating system on the GPU nodes, which is fully supported by Hammerspace.

LSS nodes are more flexible in terms of both configuration and operating system choice. Customers can select the server vendor, model, and configuration of their choice, provided it meets the requirements. Hammerspace offers some predefined configurations referred to as "ready nodes" as one option for LSS. While several supported options exist for operating systems, Hammerspace recommends RHEL-based distributions based on 9.5 including Rocky 9.5. However, since NFS is an open standard and built into all Linux kernels, we do not limit to 9.5. Customers should consult with their Hammerspace technical team to verify any compatibility, feature support, or other issues.

LSS nodes can provide the following services:

- Storage volumes
- Data Mover
  - NFS mover, also known as Data Instantiator (DI)
  - Cloud Mover (CM), which supports any S3-compatible cloud or object storage
- Hammerspace S3 services
- Quorum services for Anvil nodes

Some customers have Tier 0 as the first tier, and LSS as the second, which they may call Tier 1. Others have third-party enterprise storage as Tier 1. You might have LSS as the first tier of storage, and not use any of the storage in GPU nodes as Hammerspace managed volumes. Although that technically isn't Tier 0, it is an architecture used by some large Hammerspace customers, and the storage configuration of the LSS is essentially similar to how it would be done on Tier 0 nodes.

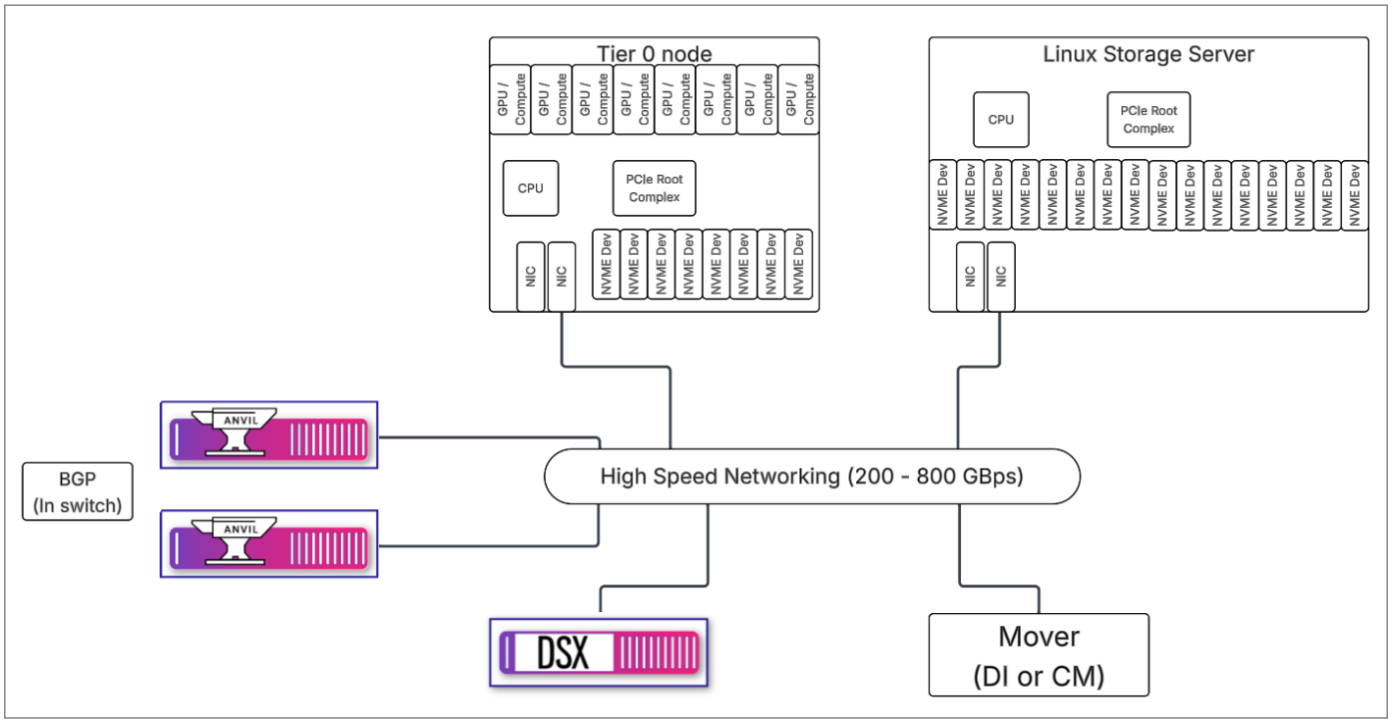

Figure 2. What is Tier 0?

1.1. Goal of This Document

The purpose of this document is to outline the setup, configuration, and management of Hammerspace Tier 0.

This document will:

- Provide an overview of solution architecture, including components, their relationships, and connections
- Provide an overview of how to configure Tier 0 and Linux Storage Server nodes, add them to a Hammerspace cluster both as servers and as clients
- Provide an overview of how to install, configure, and connect an NFS mover
- Provide some example objectives for data placement

Chapter 2. Tier 0 Superpowers

These superpowers are listed from the most basic, to the most sophisticated. Each builds on the previous capabilities.

2.1. Make Use of Stranded Capacity

The most basic level of utility for Tier 0 is putting data on any drive on any node in the cluster without regards to affinity to nodes, whether data is active or idle, or where else the data might be needed. Any orchestration is limited to having multiple instances of each file on different nodes.

2.2. Data Orchestration Across Multiple Tiers

The next level of use case adds orchestration to keep active data instances on Tier 0 and move idle data to less expensive tiers that are separate from the GPU nodes. The orchestration is managed by a storage service, with no regard for integration with the GPU/AI job management.

2.3. Job scheduler-Integrated Orchestration

The AI job scheduler knows which nodes will be assigned to a job. Integrating storage and data management APIs into the job setup workflow allows the scheduler to control data placement into the nodes that will run the job.

Advantages:

- Higher probability data needed by the job is on a node running the job.
- Remove vulnerability of instance or data loss due to reimage of nodes used for other jobs.

2.4. Data to Node Affinity

This level of Tier 0 consists of three elements:

1. Orchestration to place data on job-specific nodes.
2. When data is placed on multiple nodes, Anvil provides layout with the requesting client as the first entry in the layout.
3. The ability for I/O to bypass the NFS and network stack when accessing a local instance of a file. This is referred to as localio in the Linux NFS client.

2.5. HCI and CPU-based Servers

While this document mostly discusses GPU-based workloads that are most urgently needed for AI use cases, the fact is that CPU-based servers often ship with the same ultra-fast local NVMe drives that can be activated as Tier 0. This is also true for low-power ARM CPUs.

These server types are often confused with Hyperconverged Infrastructure (HCI), which typically combine compute and storage in a single unit. HCI systems are predominantly used for structured data, VMs, and other block-oriented I/O patterns that are not suitable for CPU-intensive unstructured data workloads. And while HCI systems can use distributed storage architectures and other tiering solutions, their focus on block-oriented use cases make them unsuitable for processing unstructured data.

There are numerous CPU-intensive use cases that need large volumes of unstructured data, and that can take advantage of the same performance and cost improvements of Tier 0 as the GPU-computing workloads outlined above. Among them are other AI use cases, which include CPU-based AI inferencing, predictive analytics, or machine-learning pipelines dealing directly with unstructured datasets.

However, beyond AI use cases there are numerous other CPU-based application workflows that demand high performance and that can directly benefit from the ultra-low latency and enhanced throughput provided by activating Tier 0 on local NVMe drives within CPU servers.

Some of these use cases include:

- **Real-time analytics and data streaming:** Rapid processing of log data, sensor inputs, or clickstream data to generate insights instantly.
- **High-performance search and indexing:** Fast indexing and search across large-scale document repositories or datasets (e.g., Elasticsearch, Splunk).
- **Video and media processing:** Low-latency editing, transcoding, rendering, and content delivery requiring quick, frequent data access.
- **Financial trading and modeling:** Real-time risk analysis, algorithmic trading, and financial modeling needing immediate access to large datasets.
- **Genomics and bioinformatics:** Rapid processing of genome sequencing data, imaging data, or bioinformatics workloads demanding near-instantaneous data retrieval.

The benefits for CPU-based Tier 0 include the same advantages highlighted in the GPU-based examples above:

- The lower cost of activating ultra-high performance local NVMe drives within the servers vs. going over expensive networks to external arrays.
- The significant boost in performance and lower latency that Tier 0 makes possible compared with external storage, which directly benefits these applications.

Chapter 3. Architecture

3.1. Servers

3.1.1. Tier 0 Node

A Tier 0 node consists of a server platform with CPU, memory, disk, and networking as the baseline, with some specialized or additional compute which may be in the form of GPUs or extra CPU, along with additional storage beyond the boot device. It also often includes additional networking, some of which may be dedicated or intended for node to node compute related communication.

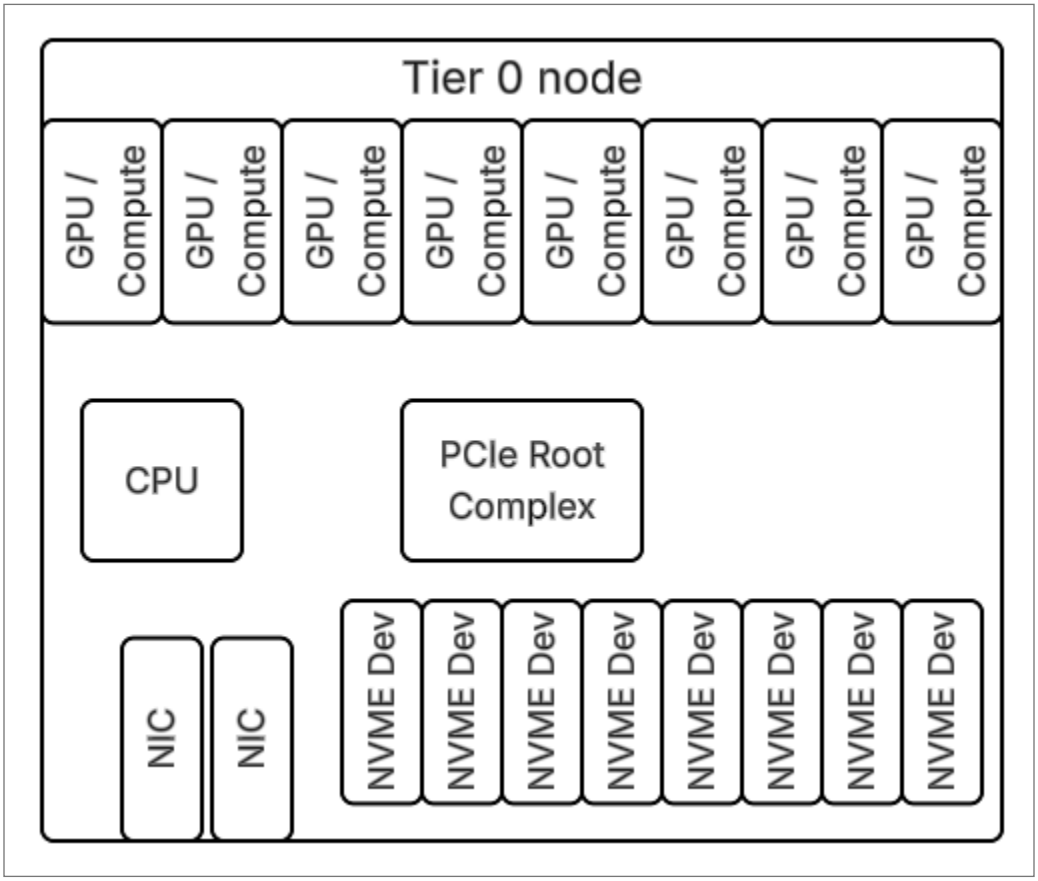

Figure 3. Tier 0 node

3.1.2. Linux Storage Server

LSS typically has more storage devices than a Tier 0 node (24, 32, or more depending on the system selected). An LSS can also provide additional services such as data movers and quorum. An LSS is typically not configured as a client, except as necessary for services it provides.

3.2. Networking

Tier 0 deployments should use recent high speed networking technology supporting 200 - 800Gbps speeds.

3.2.1. Flat Networks

Hammerspace recommends flat networks as much as practicable in Tier 0 environments. The goal is to minimize network hops between nodes. With multiple availability zones, each zone could be a flat network (subnet).

3.2.2. Jumbo Frames (MTU)

- It is recommended to use the customer's typical MTU (Maximum Transmission Unit) settings. MTU values should generally be in the range of 1500 to 9000; anything outside of this range should be discussed.
- Switches should typically be set to an MTU of 9216.
- Modern switches often ship with Jumbo frames enabled by default. While Jumbo frames may not significantly increase wire speed, they can reduce CPU consumption and interrupt handling due to fewer packets.
- Test network connectivity with ping using a specific packet size (e.g., 8972 for a 9000 MTU) and the "do not fragment" flag is advised to confirm end-to-end connectivity.

3.2.3. Bonding (Link Aggregation)

- **General Recommendation (Tier 0):** If used on Tier 0 nodes, bonding must be configured carefully due to the NUMA architecture of some server platforms. On some dual socket platforms, it is critical that bonds consist of interfaces all in the same NUMA domain to avoid a performance penalty due to I/O traversing domains. If you plan to use RDMA, bonds must consist of ports on the same physical NIC. In many cases, using an IP per NIC may be simpler and more reliable. Bonding is supported on Hammerspace Anvil and DSX nodes (bonding mode 4, also known as 802.3ad). This requires configuration of port channels with LACP of the ports on the connected switches.
- **Packet Distribution**: The default packet distribution algorithm in Linux (referred to as xmit_hash_policy) is typically layer2 or layer2+3. Depending on the network topology, you may see better packet outbound packet distribution using layer3+4. These parameters are described in detail in RedHat's documentation.

You can change the xmit_hash_policy by editing the BONDING_OPTS= parameter in /etc/sysconfig/network-scripts/ifcfg-bond0 (or whatever the bond number is).

- **Caveat:** Some customer environments (e.g., cloud, pre-existing infrastructure) may require bonding for redundancy or throughput. In such cases, professional services engagement is necessary to ensure proper configuration.

3.2.4. IP per NIC

- **Recommendation:** For systems with multiple NICs, assign a unique IP address to each NIC.
- **Rationale:** Provides multiple independent network paths, improving parallelism and resilience.
- **Configuration:** Requires specific sysctl and network configuration to ensure proper routing and load balancing without introducing routing conflicts.

3.3. Availability Zones

An availability zone is a part of an IT infrastructure system that is designed to be completely independent of other availability zones. This means it doesn't share any critical components, like power, cooling, or network, with other zones.

Components to consider may include:

- Power, whether circuit, rail, panel or similar
- Cooling
- Fire suppression zones
- Separate rooms or floors within a datacenter
- Racks or rows
- Network equipment and architecture
- Security access zones

While many discussions focus on cloud provider AZs, in this context, we focus on AZs within a datacenter. Hammerspace supports defining AZs for storage resources using a naming convention with a prefix indicating the AZ.

Hammerspace recommends that availability zones be designed approximately symmetrical in that they each have the same number of storage nodes (Tier 0 or LSS) and volumes. You should also place redundant Hammerspace nodes, including Anvil, DSX (if used), and mover nodes in different AZs. If Anvil nodes are in different AZs, and each AZ is a different subnet (or set of subnets), you will need to implement the ExaBGP (Border Gateway Protocol service) on the Anvils and configure the switches to allow Anvils to trigger the movement of the subnet containing the cluster management IP to the switch port to which the primary Anvil is connected.

When data is stored only on Tier 0, a minimum of 4 AZs is required, and 6 AZs is recommended. With 4 AZs, one can be taken offline for maintenance, leaving 3 providing full 3-way redundancy. But that doesn't allow for multiple failures during maintenance. Having additional AZs (5 or 6 total) allows for failures with automatic resilvering across three surviving AZs when an AZ is offline for maintenance.

Chapter 4. Prerequisites

4.1. Configuration Management with Teams & Admins

Ensure other teams/admins with control over storages nodes, especially Tier 0/GPU nodes, do not take control of drives used for Tier 0 and reformat or remount. One option is to let them dictate mount point, and agree to how much space will be used for different purposes. Hammerspace can configure thresholds to comply with usage guidelines.

4.2. Repository Access

Since several packages need to be installed, access to a repository containing the packages is essential. Sites without internet access will have to establish their own repos that include packages used through Tier 0 deployment procedures. Note package requirements for basic LSS /Tier 0 node deployments, as well as for additional components such as data movers. Requirements include RPM style packages as well as modules for Python scripts, and, of course, software provided by Hammerspace.

4.3. Packages for Tier 0 Nodes

- mdadm
- mkfs.xfs (Not installed by default on some distributions)
- nfs-server
- If using RDMA and Intel or Broadcom NICs, the OEM driver is required. Do not use the Linux distribution inbox driver.

4.4. Kernel Management

To ensure stability and compatibility with Hammerspace's upstream NFSv4.2 enhancements, follow these kernel guidelines:

- **Use Validated Kernels:** Recommended versions include RHEL 8, 9 or 10. If possible, do not use the Ubuntu 6.8 kernel for Tier 0 nodes. It lacks upstream LTS support and back-ports from standard kernel processes.
- **Preferred LTS Releases:** If not using RHEL, stick to Kernel.org Long Term Support (LTS) releases such as 5.15, 6.1, 6.6, or 6.12.

Chapter 5. Preparation

Gather nodes, server, and client information. Having a master list will help manage things like export rules.

- **Hammerspace 5.1 or later**
- Administrative (root or equivalent) credentials
- Hostname, IP address(es), network connectivity
- Roles: Client only, Server only (LSS), Client and server (GPU / Tier 0 nodes), Mover
- Cluster membership
- Availability zone
- Unusual configuration, drivers, etc. that need to be set or installed

Chapter 6. Setup

6.1. Automation

As you read through these steps and consider the number of nodes involved, it will become obvious that these procedures are repetitive and potentially error-prone. Automation should be utilized to set up, configure, and manage.

Reference the **Hammerspace Ansible Tier 0 project** to fully automate the setup of RAID arrays, filesystems, NFS exports, and firewall configuration for Hammerspace Tier 0 and Linux Storage Server (LSS) deployments. Refer to the project's README.md file for full details.

6.2. System BIOS, Firmware, NIC Firmware

The following items need to be considered and prepared on the server and peripheral hardware:

- System BIOS
  - Nodes Per Socket (NPS) - Hammerspace recommends that Nodes Per Socket be set to 1 on AMD platforms. On Intel, this is called Sub-NUMA Clustering (SNC).
  - Hyperthreading - Simultaneous Multithreading (SMT) can depend on server function, but the general recommendation is to set it to off in the system BIOS since it can introduce latency variance and reduce sustained memory copy performance, impacting applications sensitive to predictable I/O.
- Memory architecture and optimization
  - DIMM types and speeds should match the memory controller of the CPU.
  - Populate all banks appropriately to maintain maximum memory speed. Review CPU specifications for number of memory channels and impact of number of DIMMs per channel (DPC).
- Drive firmware
- NIC firmware
- NIC Driver
- NVME configuration
  - Optimal slot / bay usage considering PCIe lanes, NUMA node. Some of this is dictated by the server architecture.
  - A single namespace per drive is sufficient. With RAID, the recommendation is a single namespace for each NVME device.
  - Select an NVME format that does not use LBA Metadata space (PI)
  - Sector size (512 vs 4096 byte). 4096 byte sector size is strongly recommended for Tier 0 and LSS nodes. Hammerspace node boot devices currently require 512 bytes per sector.
- Device drivers and settings especially for NICs
  - There may be different device drivers for advanced networking such as RDMA.

6.3. Drives, RAID, Filesystems

Check how NVME drives enumerate (what device name they appear as). Are they consistent between reboots or other events? On some systems, NVME devices may not enumerate as the same /dev/nvmexxx device on reboot. To prevent devices being mounted on the wrong mount point, UUID-based mounts should be used.

NVME drives often support different formats that may include metadata space for each sector (NVME metadata space, not Hammerspace). On some systems, this might be used for additional checksums. With Tier 0 deployments, NVME drives should be configured and formatted with 0 bytes of metadata space.

Determine whether RAID or individual drives will be used. One consideration is how many volumes will need to be managed, which is a function of how many servers times how many drives they have. If the number of volumes is projected to be over 1000, RAID configuration should be favored.

Tips to Improve RAID Performance

- Use RAID set sizes that are a power of 2 where possible
- Attempt to build RAID sets on NUMA local NVMe
- The default mdadm chunk size of 512KiB will suffice for many deployments. For advanced tuning, consult with your Hammerspace tech team to calculate a chunk size matched to I/O profile.

Determine NUMA placement of NVME devices as follows:

```
# grep "" /sys/block/nvme*n1/device/numa_node

/sys/block/nvme0n1/device/numa_node:0

/sys/block/nvme10n1/device/numa_node:1

/sys/block/nvme11n1/device/numa_node:1

/sys/block/nvme1n1/device/numa_node:0

/sys/block/nvme2n1/device/numa_node:0

/sys/block/nvme3n1/device/numa_node:0

/sys/block/nvme4n1/device/numa_node:0

/sys/block/nvme5n1/device/numa_node:0

/sys/block/nvme6n1/device/numa_node:1

/sys/block/nvme7n1/device/numa_node:1

/sys/block/nvme8n1/device/numa_node:1

/sys/block/nvme9n1/device/numa_node:1
```

6.4. Pre-Setup Checks

Once the base server hardware is set up, operating system installed, and basic networking is configured, you can check the recommendations listed above.

If jumbo frames are in use, check the configuration with ping using a large frame size and the Do Not Fragment flag set. Note that the frame size of the ping is smaller than the MTU because of packet overhead. The most common MTU, 9000, should be checked with a ping size of 8972. You should check node (Tier 0 and LSS) to node, node to Hammerspace nodes, and other combinations including DI, etc. Every node should be checked.

```
# ping -M do -s 8972 192.168.0.101

PING 192.168.0.101 (192.168.0.101) 8972(9000) bytes of data.

8980 bytes from 192.168.0.101: icmp_seq=1 ttl=64 time=0.013 ms

8980 bytes from 192.168.0.101: icmp_seq=2 ttl=64 time=0.004 ms

8980 bytes from 192.168.0.101: icmp_seq=3 ttl=64 time=0.004 ms

^C

--- 192.168.0.101 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2038ms

rtt min/avg/max/mdev = 0.004/0.007/0.013/0.004 ms
```

Listing Storage Devices and Current Usage:

```
# lsblk -n

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS

nvme11n1 259:0 0 894.3G 0 disk

nvme10n1 259:1 0 745.1G 0 disk

 ├──nvme10n1p1 259:2 0 600M 0 part /boot/efi

 ├──nvme10n1p2 259:3 0 1G 0 part /boot

 ├──nvme10n1p3 259:4 0 32G 0 part [SWAP]

 └──nvme10n1p4 259:5 0 711.5G 0 part /

nvme0n1 259:6 0 7T 0 disk
```

```
nvme1n1 259:7 0 7T 0 disk

nvme2n1 259:8 0 7T 0 disk

nvme3n1 259:9 0 7T 0 disk

nvme4n1 259:10 0 7T 0 disk

nvme5n1 259:11 0 7T 0 disk

nvme6n1 259:12 0 7T 0 disk

nvme7n1 259:13 0 7T 0 disk

nvme8n1 259:14 0 7T 0 disk

nvme9n1 259:15 0 7T 0 disk

nvme12n1 259:16 0 7T 0 disk

nvme13n1 259:17 0 7T 0 disk

nvme13n1 259:18 0 7T 0 disk

nvme15n1 259:19 0 7T 0 disk

nvme16n1 259:20 0 7T 0 disk
```

6.4.1. Configuring the Server: RAID and Filesystems

This shows an example RAID configuration of 3 RAID arrays of 8 drives each. The last two commands ensure that the RAID configuration is saved in the mdadm configuration file.

```
# mdadm --create --verbose /dev/md0 --level=0 --raid-devices=8 /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1
/dev/nvme3n1 /dev/nvme4n1 /dev/nvme5n1 /dev/nvme6n1 /dev/nvme7n1
# mdadm --create --verbose /dev/md1 --level=0 --raid-devices=8 /dev/nvme9n1 /dev/nvme10n1 /dev/nvme11n1
/dev/nvme12n1 /dev/nvme13n1 /dev/nvme14n1 /dev/nvme15n1 /dev/nvme16n1
# mdadm --create --verbose /dev/md2 --level=0 --raid-devices=8 /dev/nvme17n1 /dev/nvme18n1 /dev/nvme19n1
/dev/nvme20n1 /dev/nvme21n1 /dev/nvme22n1 /dev/nvme23n1 /dev/nvme24n1
# mkdir /etc/mdadm
# mdadm --detail --scan | tee -a /etc/mdadm/mdadm.conf
```

Set up mount points, partition arrays, make a file system, mount, and add to fstab:

```
# mkdir /hsvol0
# parted /dev/md0 mklabel gpt
# parted -a opt /dev/md0 mkpart primary xfs 0% 100%
# mkfs.xfs -d agcount=512 -L hsvol0 /dev/md0p1
```

```
# mount /dev/md0p1 /hsvol0
# echo '/dev/md0p1 /hsvol0 xfs 0 0' >> /etc/fstab
```

After editing /etc/fstab, you should run 'systemctl daemon-reload' to update systemd units generated from the file.

6.4.2. NFS, Exports, /etc/exports

If the NFS server (nfs-kernel-server) is already installed, check /etc/nfs.conf and /etc/nfs.conf.d/local.conf for non-default TCP/UDP port configuration. Hammerspace recommends the following /etc/nfs.conf settings for use in Tier 0 environments:

Example /etc/nfs.conf

```
[nfsd]
threads=128
# udp=n
# tcp=y
vers3=y
# vers4=y
vers4.0=n
vers4.1=n
vers4.2=y
rdma=y
rdma-port=20049
```

NFS v4.2 is required for Parallel NFS, client side mirroring and other advanced features used with Tier 0 implementations. Although clients mount the Hammerspace share(s) via NFS v4.2, the I/O from clients to storage volumes uses NFS v3, so NFS servers including Tier 0 nodes must enable NFS v3.

Run the following command to start NFS and configure it to start on system startup:

```
# systemctl enable --now nfs-server.service
```

On some Linux distributions, the NFS service might be nfs-kernel-server, although that is usually linked to nfs-server. Ensure showmount is *not* disabled (uncommon).

For /etc/exports:

- Add Hammerspace nodes and Anvil cluster mgmt floating IP RW, no_root_squash,sync,secure,mp,no_subtree_check.
- Add clients RW,root_squash,sync,secure,mp,no_subtree_check. This can be done by subnet or host

6.4.3. Export Options

The following export options are recommended for nodes that serve NFS, including Tier 0 nodes and LSS:

- rw - Read/write access
- no_root_squash - Allows appropriate nodes to have root access on the export. This setting is required for Hammerspace nodes (Anvil and DSX), and nodes running mover services (DI and CM).
- root_squash - Prohibits nodes from root access on the export. This should be used for Tier 0 and LSS nodes that are not running mover services.
- sync - Causes the NFS server to ensure writes are committed to storage before replying to the client. This is default behavior in modern NFS server implementations, but specifying the option may suppress warning messages.
- secure - Requires that clients use a secure, well-known TCP port (less than 1024).
- mp - Mountpoint indicates that the export should have a filesystem mounted for export. Using this option prevents accidentally exporting the directory under the mounted filesystem, which would result in data being written to the root filesystem.
- no_subtree_check - This is a default in modern NFS server implementations. The main reason to specify this parameter is to suppress a warning that the default behavior has changed.

The following snippet of /etc/exports shows 3 exported volumes with 3 exports for a Hammerspace Anvil node and 3 exports for a subnet of Tier 0.

Example /etc/exports

```
/hsvol0 10.1.2.3(rw,no_root_squash,sync,secure,mp,no_subtree_check)
/hsvol1 10.1.2.3(rw,no_root_squash,sync,secure,mp,no_subtree_check)
/hsvol2 10.1.2.3(rw,no_root_squash,sync,secure,mp,no_subtree_check)
/hsvol0 10.2.1.0/24(rw,root_squash,sync,secure,mp,no_subtree_check)
/hsvol1 10.2.1.0/24(rw,root_squash,sync,secure,mp,no_subtree_check)
/hsvol2 10.2.1.0/24(rw,root_squash,sync,secure,mp,no_subtree_check)
```

The challenge with the above syntax is that although you can put multiple hosts on each line, you have to specify export options in parentheses for each host which gets messy and error prone.

Many Linux distributions allow multiple IPs on a single line with their common export options first:

Example /etc/exports

```
/mnt/hsvol1 -rw,no_root_squash,sync,secure,mp,no_subtree_check 10.200.104.38 10.200.105.195
/mnt/hsvol1 -rw,root_squash,sync,secure,mp,no_subtree_check 10.200.102.33 10.200.103.223 10.200.101.217
10.200.102.92 10.200.102.34 10.200.103.170 10.200.102.172 10.200.103.173
```

After updating /etc/exports, run the following command to re-read the exports file:

```
# exportfs -r
```

6.4.4. Firewall

You may need to allow ports used by NFS and related protocols and services. Check /etc/nfs.conf and the files in /etc/nfs.conf.d for port specifications, and also rpcinfo -p to confirm. We have seen sites that had other applications on GPU nodes configured to use ports normally used by NFS.

For Rocky, Centos, RHEL using standard ports:

```
# firewall-cmd --permanent --add-service=nfs --add-service=rpc-bind --add-service=mountd ; firewall-cmd
--reload
```

For Ubuntu and Debian using standard ports:

```
# ufw allow port 111 # portmapper
# ufw allow port 2049 # NFS
# ufw allow port 20048 # mountd
```

Confirm it works by checking exports from another client (can be another LSS/Tier 0 server):

```
# showmount -e 10.200.103.167

Export list for 10.200.103.167:

/mnt/hsvol1
10.200.112.0/21,10.200.10.0/24,10.200.103.166,10.200.103.167,10.200.102.220,10.200.101.3,10.200.106.197

/mnt/hsvol0
10.200.112.0/21,10.200.10.0/24,10.200.103.166,10.200.103.167,10.200.102.220,10.200.101.3,10.200.106.197
```

Chapter 7. Hammerspace Deployment and Integration

7.1. Hammerspace Installation

General installation of Hammerspace nodes is well-documented in the Hammerspace Installation and Licensing Guide. What follows here are specifics that may apply to Tier 0 in general, or your environment.

- **Anvils:** If Anvils are to be installed in different subnets, BGP (Border Gateway Protocol) is required, using the ExaBGP tools in the Anvil nodes to communicate with connected switches in order to control which port routes the subnet for the Anvil floating management IP. Contact your Hammerspace tech team for architecture and deployment details.
- **DSXs:** DSX may not be required for some Tier 0 installations. Mover (DI) may be deployed as RPM or container. See Appendix 1 for instructions on deploying the DI RPM.

7.2. Cluster Integration

LSS and Tier 0 storage services can be added using the Hammerspace GUI, CLI, or scripted with the API. For a small installation, it may be simple enough to add storage systems via GUI or CLI. For larger installations, setup should be automated using a script or configuration management or automation tools such as Ansible or Terraform.

7.2.1. Adding a Tier 0 Node or LSS as a Storage Node

Add a storage node using the node-add command:

```
anvil1> node-add --type OTHER --name lss001 --ip 10.1.2.2
```

If you plan to use per-node objectives (for example to achieve local write affinity), you can have the node-add command create them as part of adding the node:

```
anvil1> node-add --type OTHER --name lss001 --ip 10.1.2.2 --create-placement-objectives
```

If successful, the node-add command responds with a description of the node, with the exports listed as logical volumes.

7.2.2. Add Node Exports as Volumes

Add each export as a volume using the volume-add command:

```
anvil1> volume-add --name lss001::/hsvol1 --node-name lss001 --access-type read_write --logical-volume-id
123 --low-threshold 90 --high-threshold 95 --skip-performance-test
```

You can use either --logical-volume-name or --logical-volume-id to specify the volume. With scripting or other automation, it may be easier to use the logical volume name, since for simple NFS servers, the logical volume name is the same as the export. Alternatively, you can get the list of logical volume IDs from the output of the node-add command. When adding many volumes in rapid succession, you should skip the performance test.

If you plan to use availability zones, which are a best practice, the volume name must be prefixed with "AZ", a whole number, colon, then a unique name.

The example below shows adding a volume with the "AZ1:" prefix.

```
anvil1> volume-add --name AZ1:lss001::/hsvol1 --node-name lss001 --access-type read_write --logical
-volume-id 123 --low-threshold 90 --high-threshold 95 --skip-performance-test
```

You can also name the node with the AZx: prefix, then by not specifying the --node-name parameter, with the volume-add command, the volume name will default to node::/path. With this trick, the volume name gets the AZx: prefix from the node name, since all volumes on a node would be in the same AZ.

7.2.3. Create Share(s)

You can create many shares for various purposes, such as checkpointing, to contain job-specific data, or results/output.

Run the following to create a share:

```
anvil1> share-create --name checkpoints --path /checkpoints --export-option 10.1.2.0/24,rw,root-squash
```

If you don’t specify --path, it will default to the share name. You can specify a size, but must plan this carefully so jobs and applications don’t run out of space unexpectedly due to a share quota. If size is not specified, the share size and free space will show as the total size and free space of all local (NFS, not object) volumes.

You must provide the necessary set of export options to include all clients needing access to the share, including all Tier 0 nodes. If LSS nodes are not used as clients, they do not need to be listed in the Hammerspace share exports. You may need to specify multiple --export-option parameters. If the command line gets too long, you can add export options using the share-update command. Clients can be specified by IP address, subnet in CIDR format, hostname, FQDN, or netgroup.

Hammerspace exports support ro or rw, root_squash or no_root_squash, and secure (default) or insecure. Generally, "rw,root_squash,secure" is the recommended set for Hammerspace exports to Tier 0 nodes.

You can specify objectives during share creation or add them later. See Data Placement Using Objectives.

7.3. Mounting Shares on Clients

Once shares are created, clients can mount them using NFS v4.2. The exact mount options may depend on a variety of factors including use case, model of NICs, and especially client kernel version, since the kernel version dictates which advanced NFS features are available.

```
# mkdir /mnt/checkpoints

# mount -o vers=4.2,nconnect=8 <anvil_IP>:/checkpoints /mnt/checkpoints
```

7.3.1. Mount Parameter and Option Notes

For NFSv4.2, the IP or other host specification used is for the floating data IP of the Anvil, even though the data resides on Tier 0 and/or LSS nodes.

If mount fails with "mount.nfs: Protocol not supported", first check firewall settings. If firewall is not the issue, your Linux kernel may be newer than the Hammerspace supported kernel whitelist allows. You can bypass this using the mount option port=20492, but *please report the kernel version to your Hammerspace technical team*.

7.3.2. Guidance for Setting NCONNECT with Respect to NFS Threads

When using NFS with nconnect in large-scale environments, it's important to manage oversubscription of the NFS server's knfsd threads. Excessive oversubscription can lead to contention, latency, or degraded performance.

Goal

Maintain a maximum 8:1 oversubscription ratio of client connections (nconnect) to server knfsd threads, while also enforcing a minimum of 2 and a maximum of 16 to ensure basic parallelism.

Formula

Given:

T = number of knfsd threads (from nfs.conf, e.g., threads=128)

N = number of client nodes expected to connect to the server

R = desired oversubscription ratio (e.g., R = 8)

C = number of nconnect connections each client should use

Then:

C = max(2, floorT * R) / N

Example

If:

T = 128 knfsd threads

N = 1023 clients

R = 8 (oversubscription target)

Then:

C = max(2, floor128 * 8) / 1023

= max(2, floor(1024 / 1023))

= max(2, 1)

= 2

You should configure nconnect=2. This results in a 16:1 oversubscription in this specific scenario, which exceeds the 8:1 target but honors the enforced minimum of 2.

Takeaway

To optimize performance, adjust nconnect downward as the cluster grows, using the formula above. Monitor actual server thread utilization to verify assumptions, especially under real workloads. Consider increasing threads in nfs.conf on high-demand servers if consistent oversubscription is unavoidable.

7.3.3. Noatime

With most AI and similar workloads, access time is not a critical file attribute to maintain. Hammerspace recommends setting noatime.

Chapter 8. Validation

8.1. Configuration Validation

- Using the GUI or CLI, check the storage system list and the volume list to see that the expected systems and volumes are added
- Check that newly added volumes have 0 capacity used. If a volume has an unexpected used size, it is possible or even likely that the device is not mounted as intended and what is exported is the empty directory in the root volume of the server. Using the mp export option on the NFS exports should prevent this issue.
- Check the root file handle on volumes added from the same server. If the root file handles are the same on different volumes, it is likely that the devices are not mounted as intended and what is exported are empty directories in the root volume of the server. Again, the mp export option should prevent this issue.

Command:

```
volume-list --name AZ2:node2::/mnt/hsvol1
```

Expected output:

```
ID: d0f22c08-4089-4a5c-b3ee-4bdc52523b21
Name: AZ2:node2::/mnt/hsvol1
Internal ID: 15
Discovered addresses:
[IP: 10.200.103.53/32, Port: 2049, NetId: tcp, NodeNum: 0]
Effective IPs:
[IP: 10.200.103.53/32, Port: 2049, NetId: tcp, NodeNum: 0]
Path: /mnt/hsvol1
State: OK
Access type: Read Write
Node: AZ2:node2
Oper state: Up
Admin state: Up
Capacity: [Total: 7TB, Used: 158.6MB (<1%), Free: 7TB]
Capabilities: Max read IOPs: 0
Max write IOPs: 0
Min read latency: 0.001 milliseconds
Min write latency: 0.001 milliseconds
Max read bandwidth: 0 KB/s
Max write bandwidth: 0 KB/s
NFS read request size: 1.0 MiB
NFS write request size: 1.0 MiB
Tolerable read latency: 100000 milliseconds
Tolerable write latency: 100000 milliseconds
High threshold: 95%
Low threshold: 91%
```

```
Availability: 99%
Availability (effective): 99%
Availability drop: Disabled
Durability: 99.9%
Durability (effective): 99.9%
Encryption: false
Clone status: Not supported
Online delay: Online
Created: 2025-07-10 18:13:02 UTC
Modified: 2025-07-10 18:13:02 UTC
Root file handle: 010006002ec5b990aceb436693c0d035b7325b53
Locations:
[Type: Node, ID: 74a111ce-6069-4427-8ebd-d75ddae8a63a, Name: AZ2:node2]
[Type: Storage Volume, ID: d0f22c08-4089-4a5c-b3ee-4bdc52523b21, Name: AZ2:node2::/mnt/hsvol1]
[Type: Volume Group, ID: 22c4be93-2717-4dd5-badd-345d1c912e62, Name: all]
Random writes: Enabled
```

8.2. Data Placement

Using one of the clients, create a number of files in a Hammerspace share, ideally more files than the number of volumes or a multiple of the number of volumes as the number of files. Check the locations of these files. You can use tools like HSTK (Hammerspace ToolKit - a BASH plugin) and the command hs usage volume <share path> to see the distribution of files across the volumes. You can look at the placement of individual files using hs eval -e instances.volume <filename>.

8.3. Data Placement Using Objectives

Objectives are Hammerspace rules that control the behavior of files. They control creation and placement of instances on storage volumes and other advanced functionality. There is extensive documentation on objectives in the Hammerspace Administration Guide, and in the {kb-objectives-url}[{kb-objectives-title}] {kb-login-note}.

Objectives can be set on the whole share, or directories and files within a share, giving the flexibility and granularity to control files for many use cases.

8.3.1. Default Behavior

Simply by creating a share in Hammerspace, you get a set of 8 default objectives that control sane behavior of files. For example, they prevent instances from filling up one volume when there are other appropriate volumes for the data in question. Effectively, they balance the usage of volumes.

If you create a set of files on a share with default objectives, you should see the files spread across all volumes with available space. These objectives will respect capacity thresholds for volumes. If a volume is at or above the low threshold, data will be placed in other volumes. If a volume is at or above the high threshold, data will be evacuated to other volumes.

Assuming a share called share1, we can test this behavior as follows:

```
# mkdir /mnt/share1

# mount -o vers=4.2,nconnect=8 <anvil_IP>:/share1 /mnt/share1

# mkdir /mnt/share1/dir1

# cd /mnt/share1/dir1

# for f in `seq 1 100`; do dd if=/dev/urandom of=f$f count=10 bs=1024k; done

10+0 records in

10+0 records out

10485760 bytes (10 MB, 10 MiB) copied, 0.0790123 s, 133 MB/s

<snip>

# hs usage volume

SUMS_TABLE{

        |KEY = STORAGE_VOLUME('AZ1:node1::/mnt/hsvol0'),

        |VALUE = 9;

        |KEY = STORAGE_VOLUME('AZ1:node1::/mnt/hsvol1'),

        |VALUE = 9;

        |KEY = STORAGE_VOLUME('AZ2:node2::/mnt/hsvol1'),

        |VALUE = 9;

        |KEY = STORAGE_VOLUME('AZ2:node2::/mnt/hsvol0'),

        |VALUE = 9;

        |KEY = STORAGE_VOLUME('AZ3:node3::/mnt/hsvol0'),

        |VALUE = 8;

        |KEY = STORAGE_VOLUME('AZ3:node3::/mnt/hsvol1'),

        |VALUE = 8;

        |KEY = STORAGE_VOLUME('AZ4:node4::/mnt/hsvol0'),

        |VALUE = 8;
```

```
        |KEY = STORAGE_VOLUME('AZ4:node4::/mnt/hsvol1'),

        |VALUE = 8;

        |KEY = STORAGE_VOLUME('AZ5:node5::/mnt/hsvol0'),

        |VALUE = 8;

        |KEY = STORAGE_VOLUME('AZ5:node5::/mnt/hsvol1'),

        |VALUE = 8;

        |KEY = STORAGE_VOLUME('AZ6:node6::/mnt/hsvol1'),

        |VALUE = 8;

        |KEY = STORAGE_VOLUME('AZ6:node6::/mnt/hsvol0'),

        |VALUE = 8}
```

The hs command above is from the HSTK. You can get this from GitHub, and it is extensively described in {kb-hstk-url}[{kb-hstk-title}] {kb-login-note}.

8.3.2. Simple Objective to Make Multiple Instances

There are a few different ways to create and apply objectives to create multiple instances of files on different storage volumes. You could create "place-on" objectives for each volume, but that quickly becomes impractical since you would have to apply different objectives to each file. A simpler way to use an objective to make multiple instances and have them balanced across volumes, nodes or even AZs is to use a durability or availability objective, specified as a number of "9s" such as 99.999%, also referred to as "five nines". Some of these objectives are created by default, but you can create more with different durability or availability requirements. The main difference between durability and availability is that with durability, the system or volume might go offline, but the data is intact and will be there when the volume comes back online. With availability, the volume is expected to stay online for the specified percentage of time.

The way this works is when a durability or availability objective is applied, the system looks for a storage volume that by itself meets or exceeds the requirement. If there isn’t one, the system determines a set of volumes that together add up to the requirement and places an instance of the file on each volume in the set. It turns out that the probability math works out that you can add up the number of nines from each volume and get the same effective reliability of a single volume with that number of nines. When there are many volumes and multiple files are created, the system determines a different set of volumes for each file.

It goes even further. If you configure availability zones using the "AZx:" prefix for volume names as discussed earlier, the system will place instances such that no two instances of the same file are placed in volumes in the same availability zone.

The default volume durability is 99.9% and availability is 99%. These values can be changed, and for most storage platforms, should be set to values determined based on the storage system and the vendor documentation and guidance. For Tier 0 and LSS, the defaults are reasonable.

The following table lists default durability or availability objectives as included with a Hammerspace installation, and the number of instances you should expect with the default volume durability and availability:

Objective	Number of Instances
availability-1-nine	1
availability-3-nines	2
availability-5-nines	3
durability-1-nine	1
durability-3-nines	1
durability-5-nines	2
durability-10-nines	4

To get 3 instances using durability, we would need to create an objective with 2 times the volume durability plus 1, which would be 7 nines.

```
anvil1> objective-create --name durability-7-nines --durability 7 --description 'Make 3 instances when
vol dur = 99.9'

ID: f27e26e4-6c3d-40a4-907d-9fb154f35d2b

Name: durability-7-nines

Internal ID: 8

Priority: MEDIUM

Durability: 99.99999%

Description: Make 3 instances when vol dur = 99.9
```

Assuming we create a directory called /csm3, we can apply the durability-7-nines to that directory:

```
anvil1> share-objective-add --name test4 --objective durability-7-nines --path /csm3

Share name: test4

Share internal ID: 2

Path: /csm3

Locally applied objectives:

[Name: durability-7-nines, Applicability: TRUE]
```

```
All applied objectives:

[Name: delegate-on-open, Applicability: true]

[Name: durability-3-nines, Applicability: DATA_ORIGIN_LOCAL AND IS_DURABLE]

[Name: durability-7-nines, Applicability: TRUE]

[Name: keep-online, Applicability: IS_BEING_CREATED OR HAS_KEEP_ON_THIS_SITE]

[Name: layout-get-on-open, Applicability: IS_BEING_CREATED OR HAS_ONLINE_INSTANCE]

[Name: optimize-for-capacity, Applicability: true]

[Name: availability-1-nine, Applicability: DATA_ORIGIN_LOCAL?ALWAYS]

[Name: durability-1-nine, Applicability: DATA_ORIGIN_LOCAL?ALWAYS]

[Name: keep-online, Applicability: IS_BEING_CREATED OR HAS_ONLINE_INSTANCE AND IS_RECENTLY_USED?ALWAYS]

Active objectives:

[Name: durability-7-nines]

[Name: optimize-for-capacity]

[Name: delegate-on-open]
```

You can also set objectives on files and directories using HSTK from a client:

```
# cd csm3

# hs objective add durability-7-nines .
```

Then you can run the same loop to create files in the /csm3 directory and look at the file locations and distributions.

```
# cd csm3

# for f in `seq 1 100`; do dd if=/dev/urandom of=f$f count=10 bs=1024k; done

10485760 bytes (10 MB, 10 MiB) copied, 5.95156 s, 1.8 MB/s

10+0 records in

10+0 records out
```

```
<snip>

# hs eval -e instances.volume f3

INSTANCES_TABLE{

        |VOLUME = STORAGE_VOLUME('AZ4:node4::/mnt/hsvol1');

        |VOLUME = STORAGE_VOLUME('AZ5:node5::/mnt/hsvol1');

        |VOLUME = STORAGE_VOLUME('AZ6:node6::/mnt/hsvol0')}

[root@peterlearmonth-23360-07102025-client6 csm3]# hs eval -e instances.volume f55

INSTANCES_TABLE{

        |VOLUME = STORAGE_VOLUME('AZ1:node1::/mnt/hsvol1');

        |VOLUME = STORAGE_VOLUME('AZ2:node2::/mnt/hsvol1');

        |VOLUME = STORAGE_VOLUME('AZ6:node6::/mnt/hsvol0')}

[root@peterlearmonth-23360-07102025-client6 csm3]# hs usage volume

SUMS_TABLE{

        |KEY = STORAGE_VOLUME('AZ1:node1::/mnt/hsvol0'),

        |VALUE = 40;

        |KEY = STORAGE_VOLUME('AZ1:node1::/mnt/hsvol1'),

        |VALUE = 40;

        |KEY = STORAGE_VOLUME('AZ2:node2::/mnt/hsvol1'),

        |VALUE = 33;

        |KEY = STORAGE_VOLUME('AZ2:node2::/mnt/hsvol0'),

        |VALUE = 16;

        |KEY = STORAGE_VOLUME('AZ3:node3::/mnt/hsvol0'),

        |VALUE = 22;

        |KEY = STORAGE_VOLUME('AZ3:node3::/mnt/hsvol1'),

        |VALUE = 22;
```

```
        |KEY = STORAGE_VOLUME('AZ4:node4::/mnt/hsvol0'),

        |VALUE = 22;

        |KEY = STORAGE_VOLUME('AZ4:node4::/mnt/hsvol1'),

        |VALUE = 22;

        |KEY = STORAGE_VOLUME('AZ5:node5::/mnt/hsvol0'),

        |VALUE = 21;

        |KEY = STORAGE_VOLUME('AZ5:node5::/mnt/hsvol1'),

        |VALUE = 21;

        |KEY = STORAGE_VOLUME('AZ6:node6::/mnt/hsvol1'),

        |VALUE = 20;

        |KEY = STORAGE_VOLUME('AZ6:node6::/mnt/hsvol0'),

        |VALUE = 21}
```

In the current version of Hammerspace, distribution may not be exactly even, as shown above. This is fixed in the next major release. However, each file has instances in 3 different availability zones.

When you set an objective that requires more than one instance of files, and your Linux client has a recent enough kernel, up to 3 instances (with the current Hammerspace product) will be written by the client. This is referred to as Client Side Mirroring, as defined in RFC 8435. If you set an objective that requires more than 3 instances, the first 3 will be written by the client using CSM, and the rest will be created by a mover (DI).

8.3.3. Keep Instances Where You Want Them

When you have another tier of storage that has higher availability and durability than your Tier 0 or LSS nodes, the system will attempt to satisfy availability and durability objectives by placing fewer instances on a volume or volumes with higher availability and durability. With many Tier 0 use cases, the data should remain on Tier 0 volumes for some period of time. To keep data on specific volumes or a set of volumes, we can use a confine-to objective with a volume group.

First, create the desired volume group. Volume groups can consist of any combination of storage systems, volumes, or even other volume groups. It may make sense to have a volume group for each AZ, and a volume group that includes all AZ volume groups.

```
anvil1> volume-group-create --name AZ1 --expressions
'node:AZ1:node101,node:AZ1:node102,node:AZ1:node103,node:AZ1:node104,node:AZ1:node105,node:AZ1:node106'

anvil1> volume-group-create --name AZ2 --expressions
```

```
'node:AZ2:node201,node:AZ2:node202,node:AZ2:node203,node:AZ2:node204,node:AZ2:node205,node:AZ2:node206'

anvil1> volume-group-create --name AZ3 --expressions
'node:AZ3:node301,node:AZ3:node302,node:AZ3:node303,node:AZ3:node304,node:AZ3:node305,node:AZ3:node306'

anvil1> volume-group-create --name AZ4 --expressions
'node:AZ4:node401,node:AZ4:node402,node:AZ4:node403,node:AZ4:node404,node:AZ4:node405,node:AZ4:node406'

anvil1> volume-group-create --name AZ5 --expressions
'node:AZ5:node501,node:AZ5:node502,node:AZ5:node503,node:AZ5:node504,node:AZ5:node505,node:AZ5:node506'

anvil1> volume-group-create --name AZ6 --expressions
'node:AZ6:node601,node:AZ6:node602,node:AZ6:node603,node:AZ6:node604,node:AZ6:node605,node:AZ6:node606'

anvil1> volume-group-create --name AZ_all --expressions 'volume-group:AZ1,volume-group:AZ2,volume-
group:AZ3,volume-group:AZ4,volume-group:AZ5,volume-group:AZ6'
```

When you create a volume group, the system automatically creates a set of objectives for that volume group including place-on, exclude-from, and confine-to. Now you can apply the confine-to for AZ_all to the data that needs to stay on the Tier 0 volumes. You can even include an expression to define an "applicability" such as how long the data should stay on Tier 0.

```
anvil1> share-objective-add --name test4 --path /csm3 --objective confine-to-AZ_all --applicability
'LAST_USE_AGE<1*HOURS?TRUE'
```

8.4. Configure Tier 0 Nodes to Write Local

Future versions of Hammerspace are planned to have additional functionality to simplify node affinity for use cases like checkpointing. Currently, it is possible to use node objectives to direct I/O for a given directory to storage volumes on that node.

Verify that the job manager can instruct Tier 0 nodes to write checkpoints to a specific directory within the share.

Create per Tier 0 node directories. From a client with the necessary permissions, use the mkdir command:

```
# mkdir /mnt/checkpoints/node001
```

Add a place-on-node objective to the directory:

```
anvil1> share-objective-add --name checkpoints --objective place-on-node001 --path /node001
```

To protect checkpoints after the initial fast write to local storage, you need an objective to place instances of the checkpoint on other storage, either other Tier 0 nodes, on one or more Linux Storage Servers, or other robust storage such as a NAS or cloud. You could require multiple instances by specifying a higher availability

or durability than one node provides, which results in multiple instances of files. Basically, take the number of nines specified in the objective and divide by the number of nines Tier 0 volumes provide and that gives the number of instances needed. If there is a more robust storage system (another tier), a place-on for that system would create an instance of each file in that system. With either of these, a brief time condition is needed to ensure the system doesn't try to immediately write first instances to non-local storage.

```
anvil1> hscli objective-create --name place-on-BigStore1 --place-on-list node:BigStore1

ID: d170c879-5c24-45d5-8de4-ba94a829470e

Name: place-on-BigStore1

Internal ID: 5

Priority: MEDIUM

Place on:

[node: BigStore1]

anvil1> hscli objective-create --name tier0_to_tier1 --applied-objective "d170c879-5c24-45d5-8de4-
ba94a829470e,MODIFY_AGE>5*MINUTES"

ID: 50b0db52-85f6-42d3-ab8e-ef08bd49ea26

Name: tier0_to_tier1

Internal ID: 6

Applied objectives:

[ID: 1, Objective ID: d170c879-5c24-45d5-8de4-ba94a829470e, Objective name: place-on-BigStore1,
Applicability: MODIFY_AGE>5*MINUTES]
```

Chapter 9. After-Setup Tasks

Send a support bundle to Hammerspace to establish a baseline for support.

```
anvil1> support-bundle --push
```

Note that support bundles post to Hammerspace using HTTPS over port 443. If firewall rules block this traffic, and there is a proxy available, you can configure the proxy using the cluster-config command.

```
anvil1> cluster-config --proxy-url http://10.99.42.123:12345
```

The proxy-url parameter accepts http or https as the protocol.

If port 443 is blocked and there is no proxy, you can create local support bundles, download from a URL on the Anvil, then transfer them to Hammerspace using any mechanism supported in your environment. Hammerspace support can provide a secure manual upload URL.

```
anvil1> support-bundle --local --all
ID: 62c491e5-446b-454f-bc32-4901e9274413
Name: support-bundle
Status: COMPLETED
Status message: You may download the support bundle from https://10.99.42.40:8443/support/support.peter-
anvil.20250611.183518281.tar
Progress: 100%
Created: 2025-06-11 18:35:17 UTC
Started: 2025-06-11 18:35:17 UTC
Ended: 2025-06-11 18:38:18 UTC
Params: node: ; created-by-name: admin; should-push: false; node-all: true; created-by: Uoid
[uuid=de46fcae-c3d5-4151-85aa-b00dfff1ae65, objectType=USER]
Context: output-file-path: /support/support.peter-anvil.20250611.183518281.tar; collect-local-success:
true
```

Chapter 10. Ongoing Usage

10.1. General

- Ensure that jobs are configured to use storage on Hammerspace shares, both in terms of mounting the shares on the GPU nodes, and placing data on them.
- Ensure that node maintenance workflows do not take ownership of NVME drives and reformat or remount them. The Hammerspace Tier 0 Ansible playbooks include some scripts and a remount service to try to prevent GPU farm administrators and automation from unmounting volumes used by Hammerspace Tier 0.
- Ensure that maintenance operations affect nodes within a single availability zone at a time.
- For fleet updates (e.g., kernel patches), perform rolling updates at rack scale.
- Disable "Availability Drop" during maintenance to prevent the system from prematurely evicting nodes that are merely rebooting.
- Do not re-provision instances if a simple reboot suffices; re-provisioning can change IPs and wipe local storage volumes.

10.2. Ingesting Data

There are essentially three ways to get data into Hammerspace:

- **Copy in** - Any data copy tool that can write to an NFS mount could be used to bring existing data into Hammerspace.
- **Create new** - Workloads produce some kind of output or result. The Tier 0 architecture is designed to store that output.
- **{kb-assimilation-url}[{kb-assimilation-title}] {kb-login-note}** - For existing data currently on other storage platforms, Hammerspace provides a way to "import" the metadata and hook a filesystem into a Hammerspace share, initially without moving the actual data. Objectives can be applied to the share into which the data is assimilated to move instances of the data onto Tier 0 or LSS nodes based on any metadata criteria such as time stamps, filename or extension, or path.

10.3. Snapshots

Hammerspace has a snapshot technology as one level of data protection. With supported storage systems, Hammerspace snapshots can use offloaded file cloning. This currently supports DSX nodes and some third-party storage systems - it does not currently support XFS file cloning on Linux servers including Tier 0 and LSS. When offloaded cloning is not available, the snapshots revert to full file copy on change. For this reason, snapshots should be avoided during checkpoint creation.

Chapter 11. Monitoring

Hammerspace recommends Grafana as a monitoring tool, with Prometheus as the time series database service to collect metrics from Hammerspace and LSS / Tier 0 nodes. The Prometheus exporters are enabled on a Hammerspace cluster with the cluster-config command as follows:

```
anvil1> cluster-config --prometheus-exporters-enable
ID: 07106af1-4ae9-4509-8d3c-497614109567
Name: AE9Y4XC9E5X9J5
State: Standalone
Management IPs: [10.200.106.160/20]
Data IPs: [10.200.106.160/20]
Cluster floating IPs: [10.200.106.160/20]
Since: 2025-06-18 19:20:10 UTC
Timezone: UTC
GFS participant max suspected time: 30 minutes
Prometheus exporters: Enabled
Evaluation expiration: 2025-07-18 19:18:47 UTC
Online license activation support: true
NAS volume capacity: [Total: 136.6GB, Used: 1.2GB, Free: 135.3GB]
Share space (quota): [Total: 0B, Used: 0B, Free: 0B]
Metadata servers:
[Object type: Anvil, Node name: anvil1.hammer.space, Role: Primary, Admin state: Up, Oper state: Up]
```

For more information, reference the following:

- For the Hammerspace Grafana dashboards — downloading, installing, and configuring them — see *Monitoring Cluster Health* in the Hammerspace Administration Guide.
- Refer to Monitoring Linux host metrics with the Node Exporter | Prometheus for installation instructions for Prometheus exporters for Linux nodes.
- Refer to DCGM Exporter — NVIDIA GPU Telemetry 1.0.0 documentation for using Prometheus and Grafana with Nvidia GPU nodes.

Chapter 12. Maintenance

12.1. Volume Concepts

A volume used by Hammerspace can be in one of several states as defined in this table.

Volume State	Description
Healthy	Fully operational volume.
Suspected	Volume is detected as potentially unhealthy (e.g., RPC communication failure). Transition to Suspected is typically fast (seconds to a couple of minutes).
Unavailable	Volume is confirmed unhealthy and automatically transitions from Suspected after a configurable Max Suspected Time (default is 30 minutes).
Failed	Volume is permanently offline or explicitly marked as failed.

12.2. Adding Tier 0 or LSS Nodes

When adding nodes, one key consideration is to update the exports on all other nodes that provide services so the new nodes can access shares and exports with the same options as existing nodes. To facilitate this, it is useful to maintain a master list of IPs and export options.

12.3. Node Failures

Managing how the cluster responds to a "suspected" node failure is critical to preventing unnecessary data migration.

Calculate "Max Suspected Time": Max Suspected Time is the amount of time a volume may remain in a SUSPECTED operational state before its storage volume state is automatically transitioned to UNAVAILABLE.

Max Suspected Time must exceed the sum of a node’s shutdown and startup times. For example, if it takes 5 minutes to shut down and 10 minutes to start up, a 15-minute Max Suspected Time is too short and will trigger an unnecessary data resilver process. You should test the actual shutdown/startup/reboot times of some of your nodes, and add a buffer, such as 5 minutes, to the reboot time and use that value as the Max Suspected Time.

Tier 0 Node Hardening: Apply safety breaks via automation (like Ansible) to prevent unintentional unmounts. Ensure a service is running that automatically remounts volumes within seconds if they are detached. The Hammerspace Tier 0 Ansible adds scripts and services to ensure devices and volumes used for Tier 0 remain properly mounted. Reimaging a node may remove these features, but re-running the Ansible playbooks will reinstall them.

12.3.1. Failure Recovery Workflow

1. Identify failure type: First, determine if it's a "soft" (recoverable via reboot) or "hard" failure. Soft failures shouldn't require instance deletion. Check if the volume is listed as "Unavailable" in the Hammerspace Admin UI. If the volume is offline but the node is still pingable, it may be a soft failure of the storage service. If the node is entirely unresponsive, it is likely a hardware or kernel-level hang.
2. Check the time: **If a node has been unavailable for more than the Max Suspected Time and "availability drop" is enabled, the system has likely begun migrating or re-silvering data to other nodes.**
3. Safe Removal: Verify if failed nodes span multiple AZs before proceeding to remove. Deleting instances across 3 AZs can result in permanent data loss.
4. Cleanup: Manually "fail" volumes and "remove" nodes in the Hammerspace admin to prevent orphaned or trash entries from remaining in the database.

12.3.2. Failure Cleanup

Regardless of the failure type, once a node is determined to be non-recoverable (hard) or has exceeded the Max Suspected Time (soft-turned-hard failure), you must:

- **Fail the Volume** by marking it as failed and remove it in the Hammerspace GUI or CLI, or use an Ansible playbook.
- **Remove the Node** by ejecting the failed node from the cluster to prevent "orphaned" metadata from lingering in the database.
- **Redeploy** by adding a new instance back into the cluster to maintain capacity and protection levels.

12.4. Taking Nodes Offline for Maintenance

With availability zones, you can take one Tier 0 node or many offline at the same time, **provided they are all within the same AZ**. Certain steps should be taken to minimize impact, depending on how long a node will be offline, meaning short term, long term or even permanently.

IMPORTANT: Before removing nodes for maintenance, query the fault domains to ensure you aren't removing nodes from more than one AZ simultaneously. See Availability Zones for more information.

12.4.1. Brief Planned Outage

When taking Tier 0 nodes offline briefly for maintenance, whether for a simple reboot, replacing a failed component, or other brief task, it may not be desirable for files with instances on the down node to be resilvered (copied from other nodes that have instances to one or more nodes that don't, in order to keep the instance count compliant with the policy).

Hammerspace refers to the state where a file has all the instances it is supposed to as "aligned to its objectives". To prevent data mobility from making more instances during a brief, planned outage, Hammerspace offers the ability to disable the function that determines a volume is not available, and the data requires resilvering. The per-volume parameter is called availability-drop.

Another important consideration before taking nodes offline is whether the data on them is currently aligned to

its objectives. In other words, do the files with instances on the volumes of these nodes have all the instances they should have somewhere else? Here are two ways to see that.

In the Hammerspace GUI, go to **Infrastructure > Volumes**, then click the **volume name**, then click **Alignment** at the top.

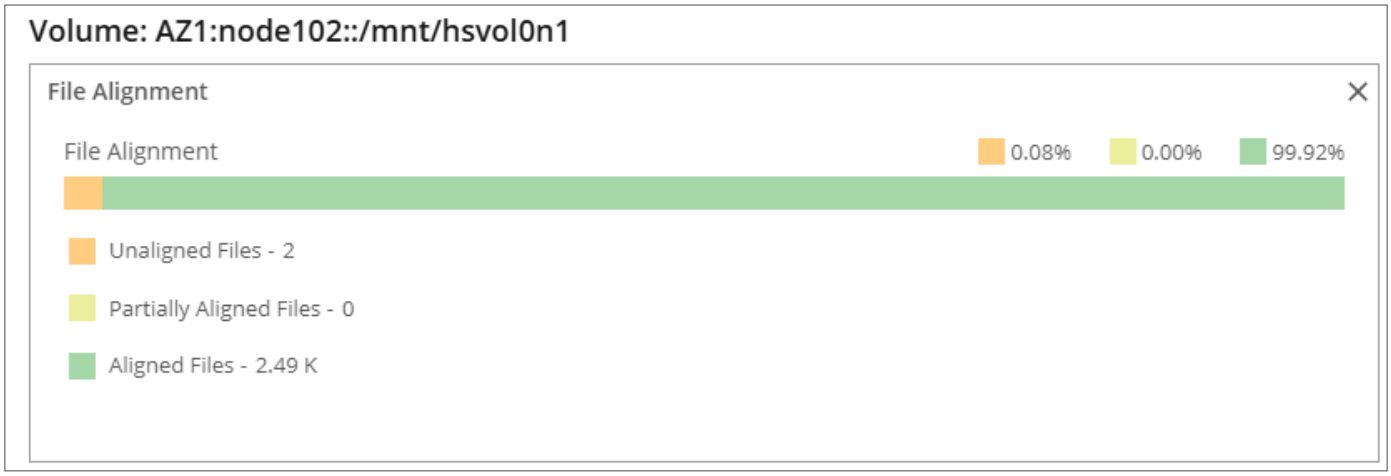

Figure 4. Brief planned outage

In this example, 2 files are unaligned, which should be resolved before this node is taken offline. The mobility screen in the GUI may provide clues such as failed mobilities from the volume. The List view can show specific files that are failing mobility, along with their sources and destination volumes. The following HSTK command shows misaligned files for a specific share:

```
# hs eval -r -e
'IS_FILE&&OVERALL_ALIGNMENT!=ALIGNMENT("ALIGNED")&&!ISNA(instances[|volume=storage_volume("gpu67.compute.
company.com::/hammerspace/hsvol0")])?{PATH,SIZE/BYTES}' /mnt/hs_share01

{"./urnumber", 8675309}

{"./something", 3502}
```

Note that you may need to run the above HSTK command for each Hammerspace share.

You can also query alignment of files with instances on a given volume using the API. This curl command queries the same volume in the GUI example above, referenced by its UUID.

```
# curl -s -H "Content-Type: application/json" -u "admin:$ADMIN_PASSWORD" -k
"https://10.200.109.239:8443/mgmt/v1.2/rest/reports/moe/compliance?storageContainerUuid=e628ce6d-d948-
405f-9c4d-6d81fc990f06&storageContainerType=STORAGE_VOLUME&breakdown=NONE"|jq

{

"breakdown": [

{
```

```
"storageContainer": {

"uoid": {

"uuid": "e628ce6d-d948-405f-9c4d-6d81fc990f06",

"objectType": "STORAGE_VOLUME"

},

"name": "AZ1:node102::/mnt/hsvol0n1"

},

"data": {

"compliantFilesUsed": 2499,

"nonCompliantFilesUsed": 0,

"severelyNonCompliantFilesUsed": 2,

"compliantSpaceUsed": 10235904,

"nonCompliantSpaceUsed": 0,

"severelyNonCompliantSpaceUsed": 8192,

"timestamp": 1758759428575

}

}

]

}
```


1. Check the alignment of files on volumes on the node(s) requiring a maintenance outage, as explained above.
2. Disable availability drop for the affected volumes. This example uses UUID, but you can use name or internal ID.

```
anvil1> volume-update --id e628ce6d-d948-405f-9c4d-6d81fc990f06 --availability-drop-disabled

ID: e628ce6d-d948-405f-9c4d-6d81fc990f06

Name: AZ1:node102::/mnt/hsvol0n1
```

```
Internal ID: 58

Discovered addresses:

[IP: 10.200.100.17/32, Port: 2049, NetId: tcp, NodeNum: 0]

Effective IPs:

[IP: 10.200.100.17/32, Port: 2049, NetId: tcp, NodeNum: 0]

Path: /mnt/hsvol0n1

State: OK

Access type: Read Write

Node: AZ1:node102

Oper state: Up

Admin state: Up

Capacity: [Total: 2GB, Used: 89.9MB (4%), Free: 1.9GB]

Capabilities:

Availability: 99%

Availability (effective): 99%

Availability drop: Disabled
```

3. Take the nodes down, perform maintenance, and bring them back up.
4. Enable availability drop for the affected volumes.

```
anvil1> volume-update --id e628ce6d-d948-405f-9c4d-6d81fc990f06 --availability-drop-enabled

ID: e628ce6d-d948-405f-9c4d-6d81fc990f06

Name: AZ1:node102::/mnt/hsvol0n1

Internal ID: 58

Discovered addresses:

[IP: 10.200.100.17/32, Port: 2049, NetId: tcp, NodeNum: 0]

Effective IPs:
```

```
[IP: 10.200.100.17/32, Port: 2049, NetId: tcp, NodeNum: 0]

Path: /mnt/hsvol0n1

State: OK

Access type: Read Write

Node: AZ1:node102

Oper state: Up

Admin state: Up

Capacity: [Total: 2GB, Used: 89.9MB (4%), Free: 1.9GB]

Capabilities:

Availability: 99%

Availability (effective): 99%

Availability drop: Enabled
```

12.4.2. Long Term or Permanent Outage

Data must be removed and placed on other storage (evacuated) before a node can be taken offline long term or permanently. While Hammerspace will self-heal when there are objectives requiring multiple instances and "the plug is pulled" on a node, it is safer to decommission the volume(s) using the built-in functionality, either via GUI, CLI, or API, which can also remove the volume from Hammerspace after evacuation. The volume-decommission command evacuates the volume without removing it from Hammerspace. You can then use volume-remove to remove it once you're satisfied that the data is evacuated. You can combine both of these by just running volume-remove which will do the decommission if it wasn't already done. There is a --replace-with parameter for these two commands, but it is usually better to let Anvil figure out for itself where to place the evacuated instances.

It is also possible to use an exclude-from-node objective to evacuate data from all the volumes on the node. You still need to remove the volumes from Hammerspace.

1. Check the alignment of files on volumes on the node(s) requiring a maintenance outage, as explained above. While the decommission procedure will create an instance elsewhere, it is better to start from a fully protected state.
2. (Optional) Run volume-decommission to evacuate the volume.

```
anvil1> volume-decommission --name "AZ1:node102::/mnt/hsvol0n1"

ID: e628ce6d-d948-405f-9c4d-6d81fc990f06
```

```
Name: AZ1:node102::/mnt/hsvol0n1

Internal ID: 58

Discovered addresses:

[IP: 10.200.100.17/32, Port: 2049, NetId: tcp, NodeNum: 0]

Effective IPs:

[IP: 10.200.100.17/32, Port: 2049, NetId: tcp, NodeNum: 0]

Path: /mnt/hsvol0n1

State: Decommissioning (quiescing)

Access type: Read Write

Node: AZ1:node102

Oper state: Up

Admin state: Up

Capacity: [Total: 2GB, Used: 89.9MB (4%), Free: 1.9GB]

Capabilities:

Availability: 99%

Availability (effective): Disposable

Availability drop: Enabled

Durability: 99.9%

Durability (effective): Disposable
```

3. Run volume-remove to remove the volume from Hammerspace. The --async option performs the task in the background allowing you to run several in parallel without waiting.

```
anvil1> volume-remove --name "AZ1:node102::/mnt/hsvol0n1" --async

The process started. To monitor progress, use: task-list --id 1ee658a1-ea85-4e37-b3d1-ff8d5384981e
```

4. Once all volumes of this node have been removed from Hammerspace, remove the node.

```
anvil1> node-remove --name AZ1:node102
```

5. Power off and/or repurpose the node.

Chapter 13. Troubleshooting

13.1. Cannot Add Volume

Anvil may refuse to add volume possibly with error message "Cannot add a volume which has 0 bytes of storage capacity." This usually indicates a problem with exports on the Tier 0 node or LSS.

What to check:

- Exports are correct, particularly that Anvil fixed and floating IPs are listed for each volume with no_root_squash

You may need to rescan the node (GUI node rescan button or CLI node-refresh --name <nodename>) after changing exports, or in extreme cases, remove and re-add the node.

13.2. Volume Issues - Volumes Go "suspected"

Causes: Node gets rebuilt, NVME reformatted or remounted in a different path. The mount directory might still be exported, although the mp (mountpoint) export option should prevent the empty directory from being exported.

What to check:

- Storage node (LSS or Tier 0) is up
- NFS service is running
- Exports are correct
- Volumes or RAID arrays have not been reformatted or remounted
- Hammerspace comb structure (/PrimaryData) is intact

If you see multiple volumes from a single storage server go suspected, and the FSID has changed, if the new FSIDs are the same on multiple volumes on the same server, this most likely indicates that it is the root filesystem being exported due to empty directory mount points.

In this example, it is fairly obvious that the volume is not properly exported.

```
anvil1> volume-list

ID: 190ad494-c79a-42e8-bac5-38587a198b19

Name: AZ3:node3::/mnt/hsvol1

Internal ID: 20

Discovered addresses:

[IP: 10.200.100.42/32, Port: 2049, NetId: tcp, NodeNum: 0]
```

```
Effective IPs:

[IP: 10.200.100.42/32, Port: 2049, NetId: tcp, NodeNum: 0]

Path: /mnt/hsvol1

State: OK

Access type: Read Write

Node: AZ3:node3

Oper state: Suspected

Oper state reason: path /mnt/hsvol1 is not exported at 10.200.100.42
```

13.2.1. Mobility Failures (Unsuccessful)

File instances aren't where they should be (not enough instances, or not on the right volumes)

GUI Mobility screen shows unsuccessful mobilities that don't resolve in short order.

Common causes include:

- No DI (Data Instantiator or NFS Mover), or DI services are not running properly. The Hammerspace GUI and CLI alerts will indicate if there are no DIs. Check the status of the DIs.
- DIs don't have RW, no_root_squash on exports of the storage servers

13.2.2. Mobility Troubleshooting

If files don't appear to have the right number of instances in the right places, especially after changing objectives or creating files with more than 3 instances, check the mobility screen in the Hammerspace GUI.

If you see any "Unsuccessful" mobilities, the most likely cause is one or more movers not having access to the exports on the source or destination storage node or volume. Check that all movers are listed in the exports of all volumes, and that they have no_root_squash.

Chapter 14. Conclusion

Hammerspace Tier 0 turns GPU server local NVMe into a new tier of high-performance shared storage, managed and protected by Hammerspace. Tier 0 storage is up to 10x faster than networked storage - so you can reduce checkpointing time for AI training and HPC, and improve response times for inferencing and agentic AI. And it enables optimal use of assets you already own so you can reduce the need for external flash storage - to reduce costs, and gain back the power and rack space those systems would otherwise consume.

Tier 0 is a game-changing solution that immediately provides value for environments of all sizes, from a small cluster with only a few GPU servers, all the way to hyperscale environments with many thousands of GPUs.

Appendices

Appendix A: Additional Resources

- Hammerspace Ansible Tier 0 Project
- {kb-assimilation-url}[{kb-assimilation-title}] {kb-login-note}
- {kb-hstk-url}[{kb-hstk-title}] {kb-login-note}
- {kb-objectives-url}[{kb-objectives-title}] {kb-login-note}
- Grafana Dashboards for Hammerspace

Appendix B: Appendix 1 - Installing Data Movers on Linux Nodes

There are two components that are collectively referred to as data movers:

- Mover, sometimes referred to as NFS mover, and internally referred to as Data Instantiator or DI. This mover moves or copies instances of files between local storage volumes via NFS. This could be between DSX nodes, to/from a Linux storage server or other third part storage system.
- Cloud Mover or CM. CM moves or copies files to and from object storage (local or remote) and cloud storage using S3, HTTPS and proprietary cloud protocols. CM also manages chunking, hash values, deduplication, and encryption for data placed in object storage.

There are three ways to implement the Hammerspace data mover technology:

- Deploy DSX nodes, which include data movers as part of their included functionality.
- Deploy the data mover RPMs on a supported Linux distribution.
- Deploy the data movers as containers

While it is supported to install DI and CM on GPU nodes, customers may choose to install on separate servers to avoid additional resource consumption on the CPUs of GPU nodes.

B.1. Supported Distributions / Platforms

DI as an RPM is available for el8 and el9 platforms.

Other x86 based Linux distributions must use container based deployments.

B.2. Dependencies

- Firewall ports 9095 and 9096 open for inbound from all Anvil IPs.
- epel-release
- platform-python
- lttng-tools
- lttng-ust
- jemalloc
- Babeltrace

Download and installation instructions for the above packages are below.

B.3. Installation, Setup, and Configuration of the DI Using RPM

Install Extra Packages for Enterprise Linux and other dependencies:

```
# dnf install -y epel-release

# dnf install -y platform-python lttng-tools lttng-ust jemalloc
```

Download and install babeltrace:

```
# wget https://dl.rockylinux.org/pub/rocky/9/devel/x86_64/os/Packages/b/babeltrace-1.5.8-
10.el9.x86_64.rpm

# dnf install -y ./babeltrace-1.5.8-10.el9.x86_64.rpm
```

B.3.1. Get the Hammerspace Data Services Components Kit

Download the Hammerspace Data Services Components appropriate to your version of Hammerspace, and the CPU architecture and Linux distribution of the servers that will run the DI. If the version you need is not listed, contact your Hammerspace tech team.

For some Hammerspace releases, all of the components are included in a single tarball. Download and extract tarball, then extract and install the DI. File names will match the version of the components.

```
# wget trans.doit.hammerspace.com/download/tier0/components/el9-components-5.1.41-452.tar.gz

# gunzip el9-components-5.1.41-452.tar.gz

# tar xvf el9-components-5.1.41-452.tar */pd-di-*

el9-components/pd-di-5.1.41-452.el9.x86_64.rpm

# dnf install -y el9-components/pd-di-5.1.41-452.el9.x86_64.rpm
```

Add the management IP address of the Hammerspace cluster to /etc/hosts. Be sure to use the correct management floating IP for your Hammerspace cluster.

```
# echo "10.1.2.3 data-cluster" >>/etc/hosts
```

Open up the firewall ports Anvil uses to talk to the DI:

```
# firewall-cmd --zone=public --add-port=9095/tcp --permanent; firewall-cmd --zone=public --add
-port=9096/tcp --permanent
```

```
# firewall-cmd --reload
```

Check firewall settings with firewall-cmd --list-all. You should see "ports: 9095/tcp 9096/tcp".

```
# firewall-cmd --list-all

public (active)

target: default

icmp-block-inversion: no

interfaces: ens192

sources:

services: cockpit dhcpv6-client mountd nfs rpc-bind ssh

ports: 9095/tcp 9096/tcp

protocols:

forward: yes

masquerade: no

forward-ports:

source-ports:

icmp-blocks:

rich rules:
```

Disable SELinux.

```
# vi /etc/selinux/config

# Set SELINUX to disabled and reboot.

SELINUX=disabled
```

Enable Linux Trace Toolkit: next generation (LTTNG) services:

```
# systemctl enable --now lttng-sessiond.service
```

```
# systemctl enable --now pd-di-lttng-recorder.service
```

Install Python PIP and other dependencies needed by the script to add the DI to the Hammerspace cluster:

```
# dnf install -y pip

# pip3 install urllib3

# pip3 install requests
```

Download and run the add_node.py script to add the DI to the cluster:

```
# ./add_node.py -i 10.4.5.6 -l 20 -d -u admin -n di-6
```

In the above command, the following options are used:

-i IP address of the DI node being added

-l Netmask prefix length of the DI node being added

-d A DI is being added (as opposed to a cloud mover)

-u Administrative user name on the Hammerspace cluster

-n Node name of the DI as you wish it to appear in Hammerspace

You can also specify admin user password with the -p option, but if not, the script will prompt.

Enable and start the DI service:

```
systemctl enable --now pd-di.service
```

You can verify the DI is properly configured and added to Hammerspace by logging in to the Hammerspace CLI as admin (or equivalent user) and running the node-list command specifying the DI node name.

```
admin@anvil> node-list --name my_di1

Name: my_di1

Type: External Mover

Internal ID: 1073744505

ID: bae33620-0281-11ef-8cab-5254009a4b8c
```

```
HW state: OK

Node state: MANAGED

Management IP: 10.4.5.6/20

S3 auth signing type: Default

Created: 2024-06-05 15:45:23 UTC

Modified: 2024-06-05 15:45:33 UTC

System services:

Object type: Data Mover

ID: 254e163c-1767-4b02-87b4-2f9c9c688b26

Internal ID: 536873594

Admin state: Up

Oper state: Up

Port: 9095
```

B.4. Troubleshooting

The following node-list command shows the status of the DI from the Anvil perspective. In this example, there is a problem with the firewall on the node running the DI.

```
admin@anvil> node-list --name di-173

Name: di-173

Type: External Mover

Internal ID: 1073741834

ID: 594662a3-e045-507b-a609-0d2cb6f16beb

HW state: OK

Node state: MANAGED

Management IP: 10.200.100.173/64

S3 auth signing type: Default
```

```
Created: 2025-06-03 23:59:31 UTC

Modified: 2025-06-05 05:29:37 UTC

System services:

Object type: Data Mover

ID: 553b23ef-0709-567b-80f0-2be3fe68a852

Internal ID: 536870923

Admin state: Up

Oper state: Down

Oper state reason: Failed port tests: [10.200.100.173:9095(Connection refused),
10.200.100.173:9096(Connection refused)]

Port: 9095
```

The above error can also be caused by the pd-di service not installed, correctly configured, or started. You might also see:

```
Oper state reason: Failed port tests: [10.200.101.3:9095(No route to host), 10.200.101.3:9096(No route to
host)]
```

When you fix firewall and pd-di service issues, it may take a few seconds for Anvil to recognize the change.

Appendix C: Appendix 2 - Use Tier 0 for Checkpointing

C.1. Checkpointing Workflow Example

1. **Checkpoint Initiation:** The application triggers a checkpoint, at which point each node creates a file on the Hammerspace global shared storage system. Based on Service Level Objectives set up in Hammerspace, the storage node selected to back each file is the NVMe that is local to each node, thus making it possible to bypass the NFS protocol and networking stack as described above.
2. **Local Write Completion:** Once the write is complete, the GPU resumes computation without delay.
3. **Asynchronous Replication:** Hammerspace detects the new checkpoint file and begins replicating it to designated storage tiers based on policies.
4. **Data Availability:** The checkpoint is now safely stored in multiple locations, ensuring it can be used for recovery if needed.

C.2. Tier 0 Checkpoint Analysis

This analysis quantifies the benefits of using Tier 0 storage. Specifically, leveraging local NVMe storage within compute nodes—for checkpointing, compared to traditional methods that rely on external networked storage systems, even those connected via high-speed 800Gb Ethernet (800GbE).

We will use the NVIDIA A100 GPU for our calculations, consider a 1,000-node cluster with 8,000 GPUs, 100 Petabytes of storage capacity, and 1 TB/sec of aggregate throughput.

C.2.1. Estimating Checkpoint Data Size

System Configuration:

- **Compute Node:** NVIDIA DGX A100 or HGX system
- **GPUs per Node:** 8 NVIDIA A100 GPUs
- **GPU Memory per GPU:** 80 GB (also available in 40 GB variants)
- **Total GPU Memory per Node:** 8 GPUs × 80 GB = 640 GB
- **CPU Memory:** Assume 256 GB per node (can vary)
- **Total Memory to Checkpoint:** GPU Memory + Relevant CPU Memory

Example Checkpoint Data Size Estimate

```
600 GB
```

C.2.2. Estimating Checkpoint Time When Writing to External Storage

Considerations when Writing Checkpoints to External Storage:

- **Network Overheads:** Protocol overheads, congestion, and latency reduce effective bandwidth.
- **Shared Infrastructure:** Multiple nodes checkpointing simultaneously can saturate the network and storage array.
- **Effective Bandwidth per Node:** Often significantly less than theoretical maximum. Let's conservatively estimate 1 GB/s per node.

Example Time to write a 600 GB checkpoint to external storage

```
600 GB/1 GB per second = 600 seconds
```

C.2.3. Estimating Checkpoint Time When Writing to Tier 0 Local NVMe Storage

Local NVMe Storage Specifications:

- **NVMe Devices per Node:** 8 NVMe drives
- **NVMe Interface:** PCIe Gen5
- **Bandwidth per NVMe Device:** Approximately 14 GB/s (real-world write performance)
- **Total Aggregate Bandwidth:** 112 GB/s per node
- **Effective Bandwidth Assuming 90% Efficiency:** 100.8 GB/s per node

Example Time to write a 600 GB checkpoint to Tier 0 storage

```
600 GB/100.8 GB per second = 5.95 seconds (6)
```

Appendix D: Appendix 3 - Complex NUMA Systems

Some server systems have NUMA architectures that require careful configuration and device usage. Dual-socket AMD-based systems are the main example.

This AMD document describes the processor and supporting component architecture, along with guidance on BIOS settings for various workloads. One of the key parameters is NUMA Nodes per Socket (NPS), which divides the resources associated with a processor into NUMA domains. In practice, for GPU nodes this is often dictated by Nvidia or other GPU vendors, commonly setting NPS = 4. On a dual socket system, this sets up 8 NUMA domains.

The following simplified diagram illustrates the two processor sockets, the External Global Memory Interconnects (xGMI) that connect the NUMA domains, and a set of I/O devices hanging from them.

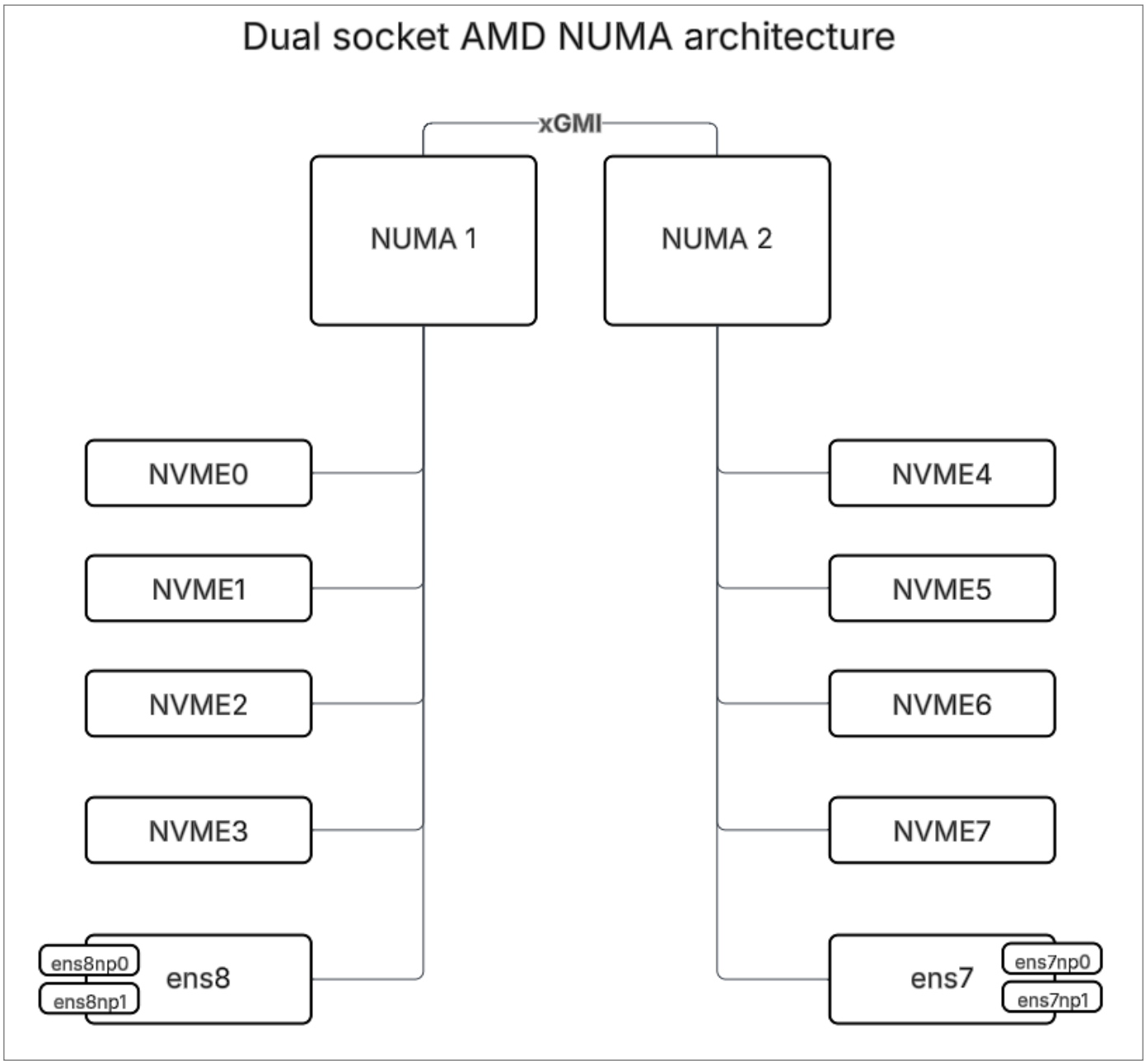

Figure 5. Appendix 3 - complex NUMA systems

The result is that I/O between devices on different NUMA domains, especially between sockets, can incur a significant latency penalty. This has been measured at customer sites, with up to 50% impact on throughput between devices on different NUMA domains.

There are several commands that can be used to determine the NUMA architecture of a system and attached or installed devices. Some commands yield different levels of detail or even what devices are included, depending on the platform. Commands like dmidecode, lspci (with -vvv or -tv options), and lstopo (if installed) can yield system architecture. The files under /sys/class are dynamically generated and are a consistent way to determine the NUMA configuration we need. The following grep commands show all the devices and their NUMA node in one concise output.

```
# grep . /sys/class/net/*/device/numa_node
```

```
/sys/class/net/ens7f0np0/device/numa_node:7

/sys/class/net/ens7f1np1/device/numa_node:7

/sys/class/net/ens8f0np0/device/numa_node:2

/sys/class/net/ens8f1np1/device/numa_node:2

# grep . /sys/class/nvme/nvme*/device/numa_node

/sys/class/nvme/nvme0/device/numa_node:0

/sys/class/nvme/nvme1/device/numa_node:1

/sys/class/nvme/nvme2/device/numa_node:2

/sys/class/nvme/nvme3/device/numa_node:3

/sys/class/nvme/nvme4/device/numa_node:4

/sys/class/nvme/nvme5/device/numa_node:5

/sys/class/nvme/nvme6/device/numa_node:6

/sys/class/nvme/nvme7/device/numa_node:7
```

Note that depending on how NVME devices enumerate, the NUMA nodes may not match the above output. The lower numbered half of the NUMA nodes are associated with the first CPU socket, NUMA 1 in the diagram above, and in AMD's documentation, while the higher numbered half of the NUMA nodes are associated with the second CPU socket and NUMA 2.

D.1. NVME Devices and RAID

The NVME devices above on NUMA nodes 0-3 are associated with the same processor, so if RAID is used, one RAID array should consist of NVME drives on these NUMA nodes, and another array with NUMA nodes 4-7.

D.2. NICs

In the example above, the dual port NIC with ens7f0np0 and ens7f1np1 is on NUMA node 7 and ens8f0np0 and ens8f1np1 are on NUMA node 2. So, devices nvme0-3 should be accessed via ens8f0np0 and ens8f1np1, and devices nvme4-7 should be accessed via ens7f0np0 and ens7f1np1.

D.2.1. Bonding

You can bond interfaces to improve performance and redundancy, but you have to be careful. A bond should include only interfaces on the same NUMA (socket). In our example, we have 2 dual-port NICs, so bonds should be created using both ports of one NIC in a bond. Keep in mind that for RDMA, bonds must use ports on the

same dual port NIC, rather than ports on two different NICs even if they're on the same NUMA node.

D.2.2. IP Assignment

Each NIC (if not bonding) or each bond needs an IP address. These IPs can be on the same subnet or different subnets. For Tier 0 use cases, different subnets work well when clients have interfaces on each subnet.

D.3. Hammerspace Volume Configuration

The last element of getting workloads to avoid crossing NUMA boundaries is on the Hammerspace side. We need to tell Hammerspace that certain volumes perform better using specific IP addresses to direct traffic over the right interfaces. In Hammerspace, each volume can have sets of Additional IPs and Excluded IPs. So it's a simple matter to specify those per volume using the GUI, CLI, or API.

You can use the volume-add command with the --additional-ip and --excluded-ip parameters to add a volume and specify the correct IPs in one step, or you can specify the IPs later using the volume-update command with --additional-ip-add and --excluded-ip-add. This part of the configuration applies the same, whether the Hammerspace volume corresponds to a RAID array or individual drive on the Tier 0 (or LSS) node.

```
anvil1> volume-update --name AZ1:node101::/mnt/hsvol0n1 --additional-ip-add 10.200.200.170/32

anvil1> volume-update --name AZ1:node101::/mnt/hsvol0n1 --excluded-ip-add 10.200.100.170

ID: 167a2a56-e36e-4806-8034-c4b885ef42fb

Name: AZ1:node101::/mnt/hsvol0n1

Internal ID: 34

Discovered addresses:

[IP: 10.200.100.170/32, Port: 2049, NetId: tcp, NodeNum: 0]

Excluded IPs: [10.200.100.170]

Additional IPs:

[IP: 10.200.200.170/32, Port: 2049, NetId: tcp, NodeNum: 0]

Effective IPs:

[IP: 10.200.200.170/32, Port: 2049, NetId: tcp, NodeNum: 0]

Path: /mnt/hsvol0n1

State: OK

Access type: Read Write
```

```
Node: AZ1:node101

Oper state: Up

Admin state: Up
```

Appendix E: Glossary

Term/Component	Description
Anvil Nodes	A Hammerspace metadata server. Responsible for managing all the metadata for the Global File System, and hosts the API and Management GUI services. Manages data policies, objectives, and failover coordination.
Cloud Mover (CM)	A Hammerspace data service responsible for uploading and downloading data from cloud and object storage. This service can run on a Hammerspace DSX node or a supported Linux server.
DI (Data Instantiator)	Responsible for moving data from one NFS (Network File System) location to another NFS location, also known as an "NFS to NFS data mover". This service can run on a Hammerspace DSX node or a supported Linux server.
Data Services Nodes (DSX)	The Hammerspace data services node. Includes functional capabilities like Portal, Mover, Cloud Mover, and Store.
HSTK	Hammerspace Toolkit is a BASH command line tool that provides access to Hammerspace metadata and other functionality for clients.
Instance	A managed copy of a file placed on a Hammerspace volume. A file can have one or many instances for performance or protection reasons, but this is typically invisible to applications and users.
Linux Storage Server (LSS)	Servers running Linux that are specifically configured and optimized to serve as storage nodes within a larger system.
Mover	A stateless DSX service that moves files non-disruptively between file storage volumes.
NFS (Network File System)	A distributed file system protocol that allows users on a client computer to access files over a network as if they were stored locally.
RDMA (Remote Direct Memory Access)	Architecture where data can be transferred directly between computer memory and storage devices (or between two storage devices) without involving the CPU or operating system of the sending or receiving systems.

Term/Component	Description
SMB (Server Message Block)	A network communication protocol for providing shared access to files, printers, and serial ports between nodes on a network.
Tier 0 Nodes	GPU compute nodes (e.g., NVIDIA DGX) with local NVMe used as high-performance storage.