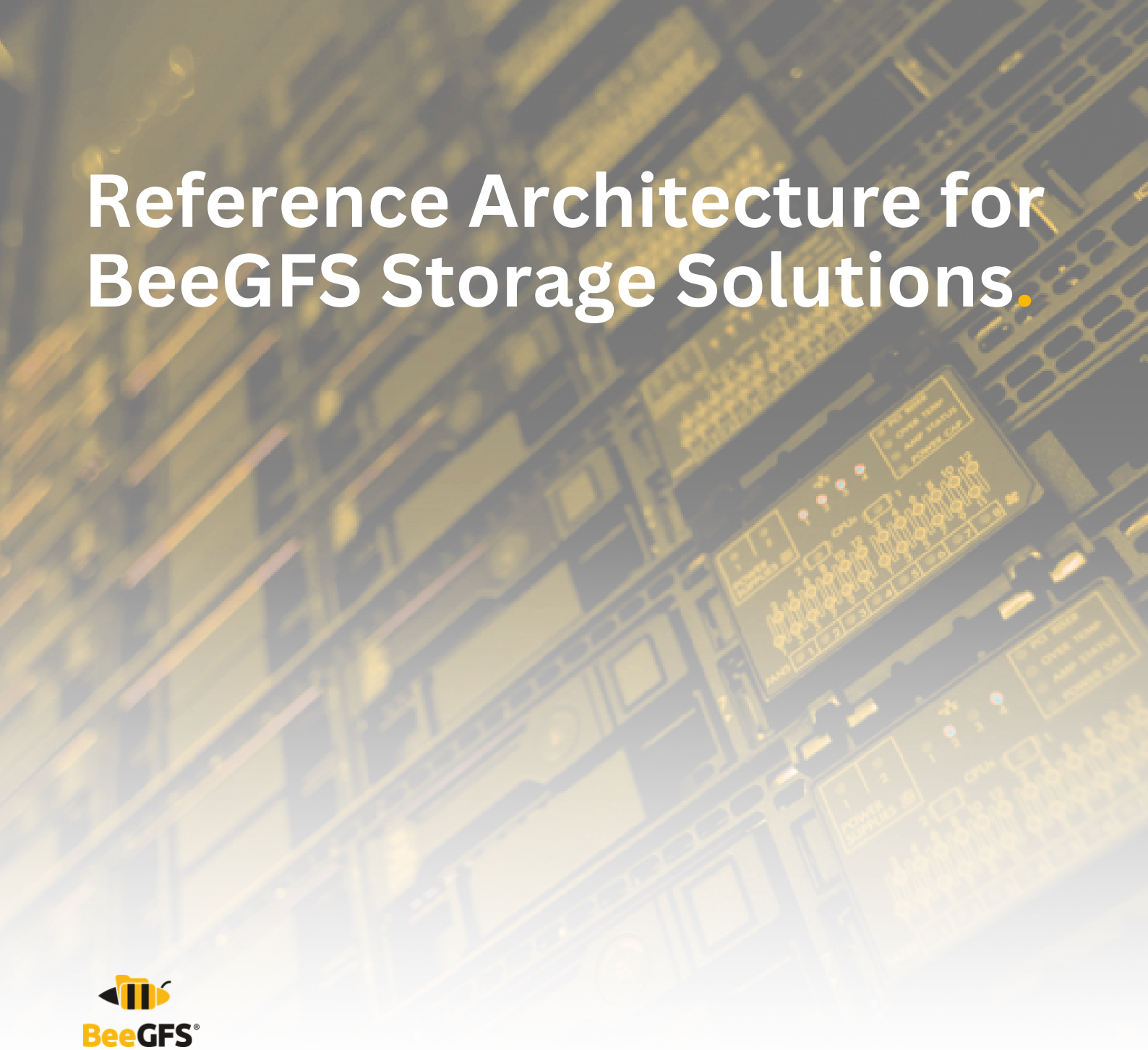
Reference Architecture for BeeGFS Storage Solutions.



BeeGFS®

Introduction

BeeGFS is a very flexible, scalable, and easy-to-deploy parallel file system. It is hardware agnostic and can be used for different storage solutions that address different needs between capacity, performance, resilience, and budget restraints. This document explains some typical architectures based on various hardware to address different requirements. Please note, It does not claim to be exhaustive.

This document focuses on current, available hardware solutions and may be adjusted later to new products. Depending on your requirements, you can select one of these solutions or combine them for several storage tiers, or to reuse existing hardware and combine them with newer hardware to support newer requirements.

Index

General configuration guidelines

Different use cases lead to different requirements and with BeeGFS you can build storage solutions that fit most of these needs.

Besides capacity, performance is one of the more obvious requirements. In general, we can divide this into two categories. File system performance (metadata operations) and bandwidth. With BeeGFS, the metadata performance can be scaled by the number of meta services. Each directory on the file system is assigned to one of the available metadata services at random, which manages the directory and its direct children. The metadata services can run in parallel on a single node (multimode) but can also be scaled out over multiple servers. This flexible design allows dedicated server nodes only for metadata services. but also for hyperconverged setups where meta and storage services run on the same server node. The bandwidth depends on the number of storage services and their targets. Other than the meta service, where each meta service has its own target, storage services can handle multiple storage targets. So, the system can have one or more storage services run on the same server node and multiple server nodes with storage services for scale out.
If the server nodes have multiple NICs, the recommendation is to also have multiple meta or storage services and bind them to the NUMA zone alongside the NIC it primarily uses and the drives that host its targets. The service can have a preferred NIC and can fail over to a different NIC in case the preferred one is not available.

The BeeGFS management service is a lightweight service that manages the inventory of the file system and acts as a single point of information for all services. This BeeGFS management service can run on one of the meta- or storage nodes or on a dedicated node (also virtualised). To avoid a single point of failure, this service should be configured in a high available mode. This can be achieved with Pacemaker and Corosync. ThinkParQ provides resource scripts for this kind of setup. Also, the meta and storage services can be configured in an HA mode. If a shared storage is used, the failover can be managed via Pacemaker/Corosync cluster to allow services to fail over to another server node using virtual IP addresses for reachability.

Another feature to build a system with higher resilience is Buddy Mirroring. With Buddy Mirroring you can build a system that does synchronous replication between pairs of storage or meta targets. For this, every target can be part of a buddy group. Inside this group one is the primary target, and another is the secondary target. All write commands will go first to the primary, and will be synchronously replicated to the secondary target. Only after the data is written to both targets, the operation is acknowledged to the client. As this is a sequential process, it will add latency to the write operations. So it is important to consider this also in terms of performance sizing. As data is written twice, twice the raw storage capacity is required to get to the same usable capacity compared to a system that doesn't use Buddy Mirroring.

If additional features like BeeGFS Copy or Remote Storage Targets are planned, dedicated clients acting as sync nodes might be required. Alternatively, regular clients can be used to move the data.

General hardware guidelines

BeeGFS is hardware agnostic and can deliver higher performance close to the HW limits.
It is important to design the system in a way that bottlenecks can be eliminated as far as possible. On the other side, the storage solution must also fit into the budget. So all the recommendations mentioned here are to be the best for performance and to avoid bottlenecks. The network topology often depends on the customer's existing network. It is necessary to do the bandwidth calculation also on the network site to avoid bottlenecks.

For calculating the bandwidth, the hardware vendor should be the first point of contact for this information. Specifically for NVMe based solutions, the available number of PCIe lanes and how they are distributed is important information where HW vendors typically can provide the necessary details. Also, the selection of the CPU type and vendor is based on the available HW platform. To get optimal performance, the system should have enough PCI lanes to support the NVMe + the network cards and possible storage HBA's.

The number of CPU cores defines how many processes (BeeGFS workers) can run in parallel, and is important when it comes to highly parallel processing. The clock speed is important for the per worker performance. For meta operations, it is highly recommended to use CPU's with a high clock count > 3GHz. If software solutions are used for RAID protection of user data, these workloads must be added and not underestimated. Here, the core count is the most important parameter.

For server nodes acting as meta servers we recommend

- min 32 cores > 3GHz
- min 256GB RAM (all memory channels should be used)
- disable all power saving modes in BIOS and OS
- numa binding meta services + meta targets NVMe + NIC to same numa zone

Every meta service can be bound to a specific network device and has a fallback option in case something happens to the preferred NIC. So the meta server nodes can utilize multiple NIC's for distributed load and lowest possible latency if the system runs in multimode (multiple meta services on one physical node)

For server nodes acting as storage server - NVMe

- min 32 cores
- 384GB RAM (all dimm slot occupied)
- disable all power saving modes for CPU in BIOS and OS
- disable all power saving modes for NVMe in BIOS and OS
- numa binding storage services + storage targets + NIC to same numa zone

Storage services are also bound to a specific network device and have their own fallback options. To make use of multiple NICs in a single machine, multiple services need to be started, each using one of the NICs as their preferred interface and the others as fallback.

Clients can be configured to support multirail usage.

Reference Architecture

1) High-performance, non-HA single-server NVMe appliance

This is the simplest solution and should be positioned as the **maximum-performance / minimum-complexity** option. The expected performance for such a server is between **100 -150GB/s** sequential write/read per server and can be scaled out.

Important architectural points from the docs:

- 1x single storage server based on PCIe5
- dual-socket CPU
- 24x local PCIe5 NVMe SSDs
- 4x 400Gbps NICs or equivalent bandwidth
- BeeGFS storage + metadata on the same node split by drive partitions
- no node failover, no shared storage, scratch only

Best-fit use cases:

- scratch tier
- AI training data staging
- customers prioritizing performance and low cost over resilience
- Buffer space for RST (Fast Object Store Cache)

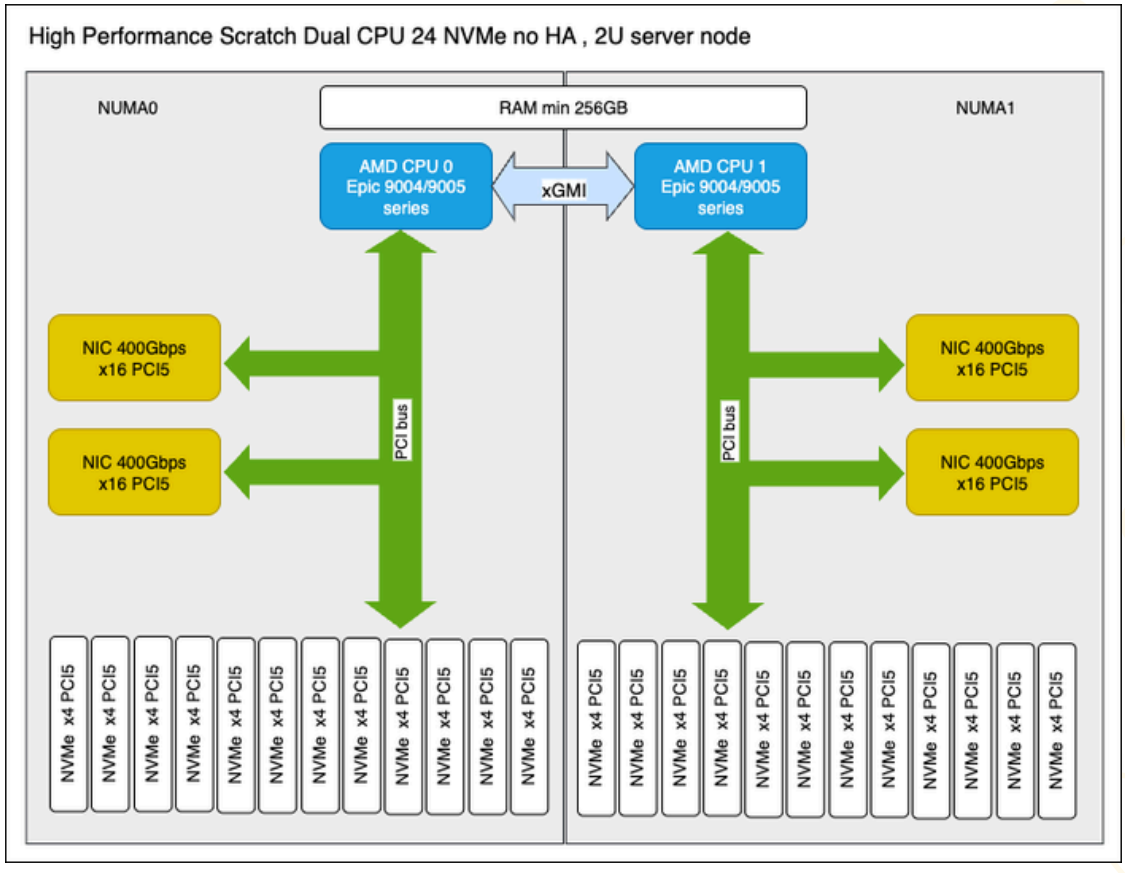
Suggested design notes:

- reserve a small NVMe partition on each NVMe for fast RAID10 for metadata
- use remaining drives for storage targets
- emphasize simplest deployment and best price/performance

explicitly note: **server failure = service outage, NVMe failure = data lost.** The risk of data loss can be eliminated by implementing parity RAID on the array.

Suggested positioning

- max possible performance.



Reference Architecture

2) High-performance dual-node shared-NVMe appliance

This is the flagship **dense 2U all-flash HA performance solution.** The expected performance for such a server is between

100 -150GB/s sequential write/read per server and can be scaled out.

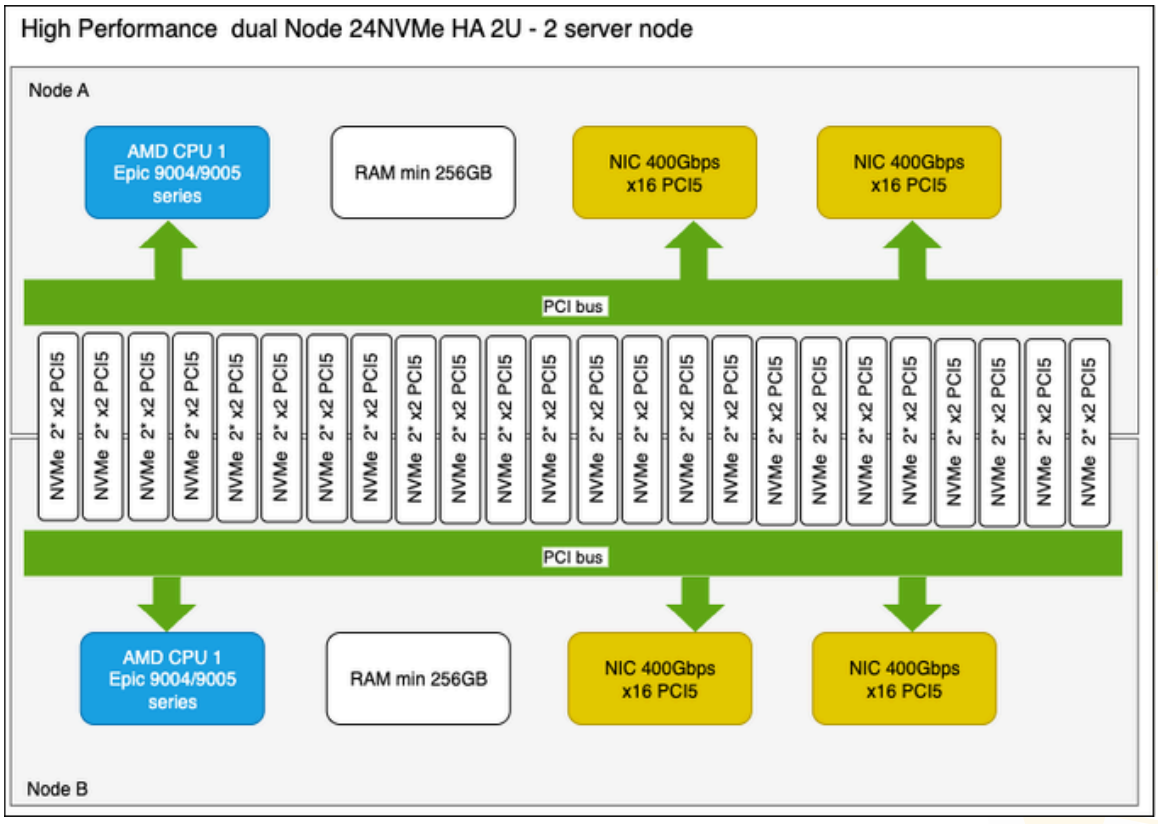
- dual independent nodes in one chassis
- shared dual-ported NVMe backplane
- 24 hot-swap U.2 NVMe drives
- PCIe Gen5
- tested on 400G fabric
- 2 NIC cards per node for redundancy / preserving CPU resources

Important architectural points:

- both nodes can access all 24 dual-port NVMe SSDs
- metadata is built on RAID10 using 4 drives total, and data targets use the remaining NVMe devices
- best approach splits each NVMe drive into two namespaces so both nodes actively utilize drive PCIe bandwidth
- Third party RAID management like Xinnor or ZFS or HW based GRAID
- HA is managed with Corosync + Pacemaker, with services and device ownership failing over to the surviving node

Suggested positioning:

- highest density
- very high throughput
- in-chassis HA
- excellent for AI / HPC active data tiers



Reference Architecture

3) HA shared-storage solution SAS - high capacity

This is a **shared-drive or shared-enclosure HA**, where two nodes can access the same media and fail over services. The expected performance for such a pair of servers depends on the number of disks and the selected RAID technology, but is expected in a range of 5-10GB/s sequential write/read for high density HDD chassis (60-104 disk)

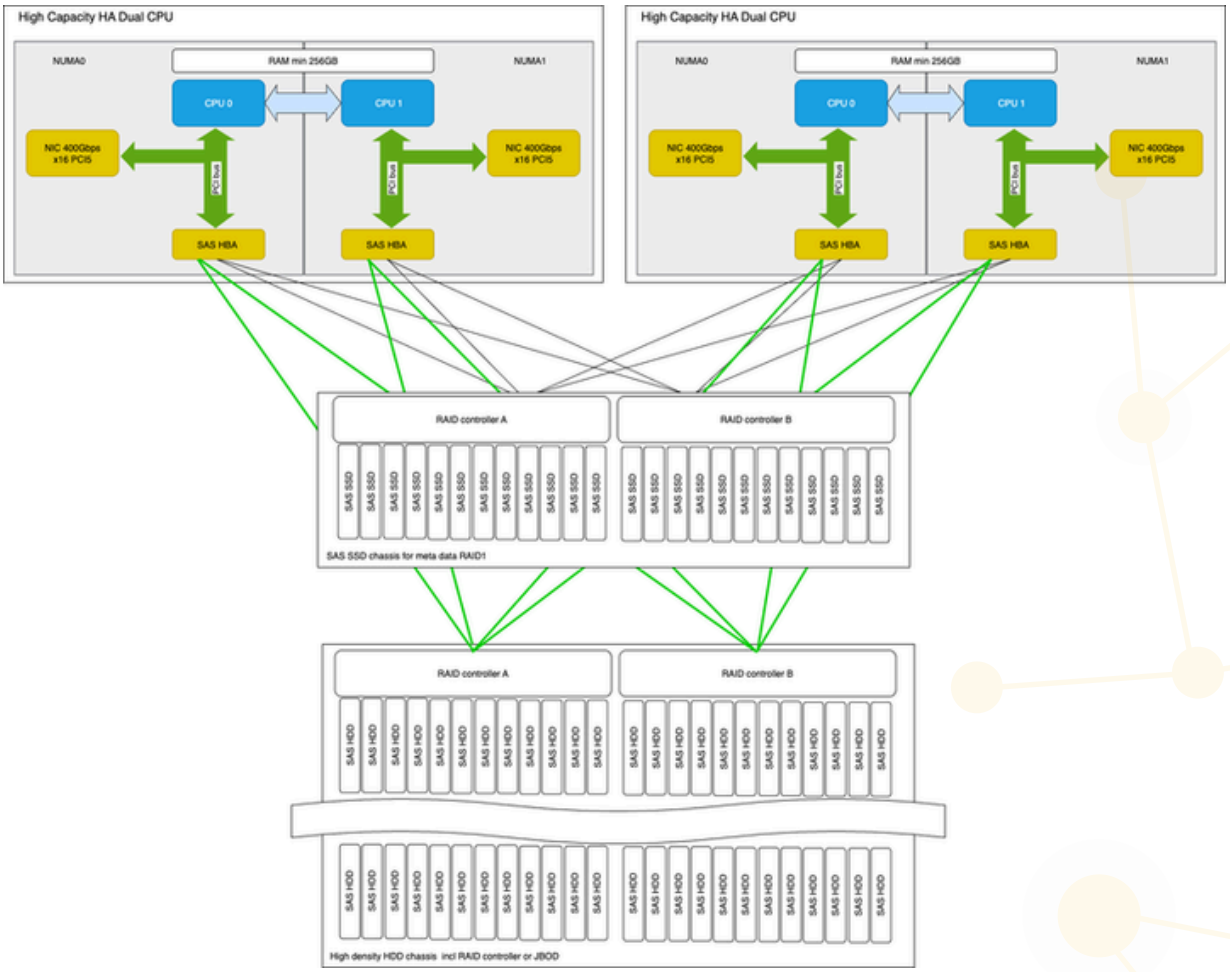
- 2 nodes
- shared SAS storage
- BeeGFS metadata and storage services active on both nodes or active/passive per resource group
- Pacemaker/Corosync for failover, virtual IPs, and fencing

Important architectural points:

- both nodes can access all drives in full redundant paths
- metadata is built on RAID10 / 1 using 4/2 drives total, and data targets are protected by HW RAID controller or ZFS if JBODS used
- HA is managed with Corosync + Pacemaker, with services and device ownership failing over to the surviving node

Suggested positioning:

- highest capacity
- full HA
- excellent for high capacity data repositories



Reference Architecture

3.1) HA shared-storage solution NVMe oF

This is a **shared-drive or shared-enclosure HA**, where two nodes can access the same media and fail over services. The expected performance for such a pair of servers is between **100 -150GB/s** sequential write/read per server and can be scaled out.

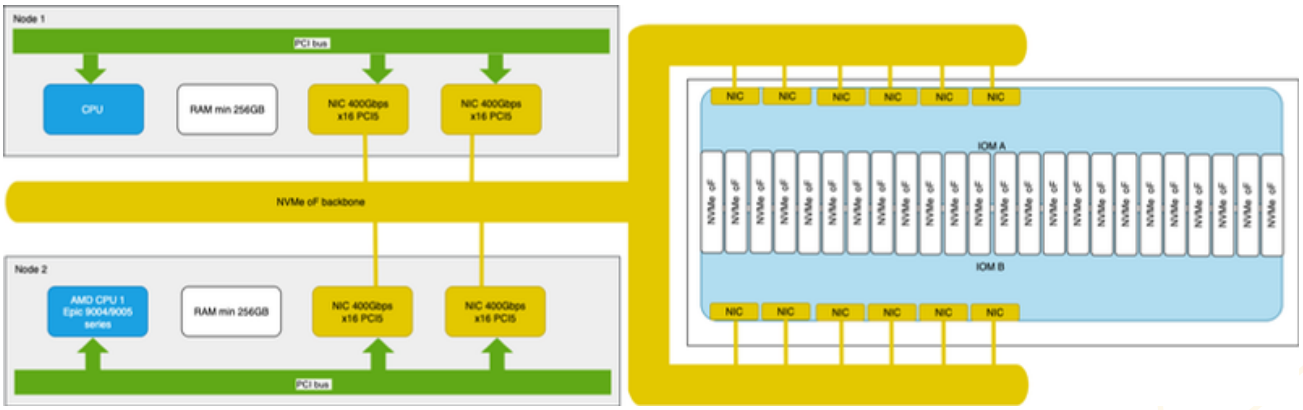
- 2 nodes
- shared NVMe oF storage
- BeeGFS metadata and storage services active on both nodes or active/passive per resource group
- Pacemaker/Corosync for failover, virtual IPs and fencing

Important architectural points:

- both nodes can access all drives in fully redundant paths
- metadata is built on RAID10 / 1 using 4/2 drives total, and data targets are protected by third party RAID solutions like Xinnor , ZFS, GRAID
- HA is managed with Corosync + Pacemaker, with services and device ownership failing over to the surviving node

Suggested positioning:

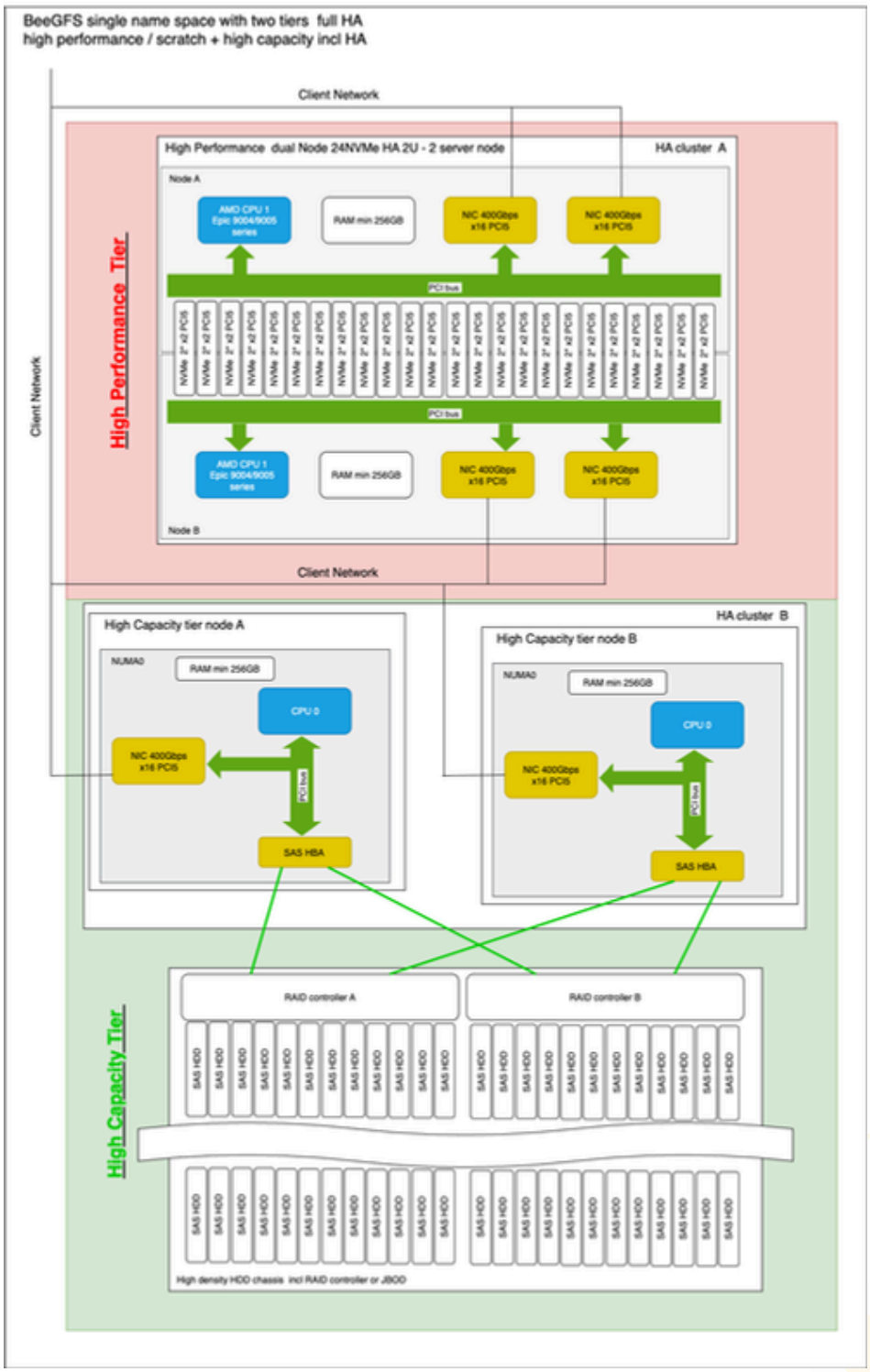
- high-performance disaggregated storage
- full HA
- excellent for high performance in distributed environments



Reference Architecture

4. Multitier - Full HA - Solution

This is an example of a solution where high-performance tiers and high capacity tiers can be combined in a single namespace, and BeeGFS tiering can be used to move data between the storage tiers. In this example all components are configured in a high availability setup, no single point of failure. This is also an example of combining existing storage clusters with new, maybe NVMe based flash storage tiers and providing these as a single namespace to the users. Files or directories can then be assigned to a specific storage pool, and a data placement policy can be automated via a job scheduler or via the administrator/user.



BeeGFS extension with S3 based object storage

In combination with the Remote Storage Target option, a third tier on object storage can be implemented. Therefore, sync nodes - connected as BeeGFS-client and on another NIC interface to S3 storage - can create copies of the files in certain directories into an S3 bucket, offload data to an S3 bucket and leave only a stub file, and import data from an S3 bucket into BeeGFS. With this functionality, a full data management solution is available with the tools built into BeeGFS Enterprise version.

Possible workflows are:

- Sharing data between different departments or research groups using a S3 based object storage in the cloud or on premise
- Backup check points and other important files on an object storage using versioning and object look for safety / security reason
- Hybrid cloud deployments where an cloud storage can be used to upload project data to get accessed by a compute cluster started in the cloud and transfer the results later back into the on premise data center
- Data lifecycle management - to offload data from the high performance tier for long term preservation, with the S3 glacier support even to tape
- Using an object storage with all his data protection capabilities as central repository for all raw data and BeeGFS on the front end as a scratch, high performance file system with less overhead for data protection and safety

Remarks

As shown in the examples above, BeeGFS is extremely flexible and can utilize a wide variety of hardware to build heterogeneous storage systems. The configurations can be used as building blocks to scale out the solution depending on the requirements. The tools available in BeeGFS Enterprise enable data tiering within the namespace, as well as offloading data to object storage and even to cold storage such as AWS S3 Glacier or on-premises solutions that offload data to tape via S3 Glacier.

For more information about BeeGFS

www.beegfs.io

https://doc.beegfs.io/latest/index.html

To get in contact with ThinkParQ consulting team:
consulting@thinkparq.com

www.beegfs.io