# openFlightHPC-clusters Documentation

openFlightHPC

Mar 04, 2020

# Architecture

This site contains documentation on general cluster considerations and best practices. It contains concept descriptions, implementation considerations and guidelines for developing a HPC stack.

CHAPTER 1

# Documentation Goal

The purpose of this documentation is to provide a list of considerations and guidelines for the development of a High Performance Computing (HPC) environment. This documentation should be followed through in order to properly understand the structure of the environment and that certain considerations are not missed out along the way.

To generalise the entire process, it goes as follows:

```
Cluster Architecture Design -> Platform Deployment/Management -> Environment Delivery
```

*Cluster Architecture Design* consists of many considerations regarding the platform, configuration and purpose of the cluster. Creating a suitable architecture design will maximise the ease of cluster management and workflow efficiency.

*Platform Deployment/Management* involves the creation of the cluster resources and configuring the base systems of the node in preparation for environment customisation.

*Environment Delivery* is the installation of software for the user experience on the cluster. This usually involves some sort of resource management/queuing system and application installation.

**Note:** It is recommended to read through all of the documentation before starting to design the HPC platform to understand the scope and considerations.

# CHAPTER 2

# Acknowledgements

We recognise the respect the trademarks of all third-party providers referenced in this documentation. Please see the respective EULAs for software packages used in configuring your own environment based on this knowledgebase.

## 2.1 License

# CHAPTER 3

# Table of Contents

## 3.1 Architecture Considerations: General

The scope of cluster architecture is broad, designing a cluster involves:

- Identifying the cluster platform
- Considering network configuration
- Deciding on operating system configuration
- Choosing node structure for workflow
- Identifying levels of resistance and redundancy

### 3.1.1 Platforms

With the rise of cloud computing and the constant evolution of metal and container solutions there are many platforms on which a cluster architecture can be built. To name a few:

- Cloud
  - AWS
  - Azure
  - Google Cloud
- Metal
  - Kickstart (or other automated OS build systems)
- Virtual
  - Libvirt
  - VirtualBox

Cluster architectures no longer have to be restricted to just one platform with the rise of *Hybrid Clusters*, creating overflow nodes in the cloud for on-site clusters to ensure that workloads can be completed within expected timescales.

#### Cloud

Using a cloud platform for a cluster has many advantages. It allows for flexible resource management, such as, scaling resources to match workload such that resources are less likely to be left sat around idle than in metal clusters. Cloud is also an incredibly quick platform to deploy architectures on as there is no need to order, install and test on-site hardware (like with traditional metal solutions). Furthermore, cloud platforms offer a higher level of redundancy and stability than traditional metal solutions for a fraction of the cost.

Cloud can be a worryingly external service as opposed to on-site solutions depending on the workload and data privacy. Additionally, permanent, non-scaled cloud solutions are likely to be more expensive than dedicated on-site hardware.

#### Metal

This is the most traditional platform for cluster architecture. Metal solutions provide complete control over the hardware choices and physical configuration of the servers. Additionally, a well-configured metal cluster will have security advantages over cloud with most, or all, of the cluster existing within the network of a site.

Maintaining the hardware and physical configuration of a metal cluster can be time-consuming, create server downtime and can also be costly when replacing or repairing the infrastructure.

#### Virtual

Depending on the workflow, virtual solutions can provide a low-cost, lightweight, transferrable platform for cluster computing. Virtual machines or containers do require a system to run on which would most likely be a metal machine. The use of virtualisation allows for services and nodes to be separated on shared resources, optimising the usage of the hardware it's running on.

#### Hybrid

Using a hybrid platform solution can provide solutions to some of the constraints of the various platforms. For example, a metal compute cluster can be configured to overflow to cloud resources when the load of the cluster is peaking. Additionally, the use of virtualisation for system services can reduce the quantity of multiple hardware or cloud instances (and therefore, the costs).

### 3.1.2 Networks

The network in the cluster may be broken up (physically or virtually with VLANs) into separate networks to serve different usages and isolate traffic. Potential networks that may be in the cluster architecture are:

- **Primary Network** - The main network that all systems are connected to.
- **Out-of-Band Network** - A separate network for management traffic. This could contain on-board BMCs, switch management ports and disk array management ports. Typically this network would only be accessible by system administrators from within the HPC network.
- **High Performance Network** - Usually built on an Infiniband fabric, the high performance network would be used by the compute nodes for running large parallel jobs over MPI. This network can also be used for storage servers to provide performance improvements to data access.

- **External Networks** - The network outside of the HPC environment that nodes may need to access. For example, the *Master Node* could be connected to an *Active Directory* server on the external network and behave as a slave to relay user information to the rest of the HPC environment.
- **Build Network** - This network can host a DHCP server for deploying operating systems via PXE boot kickstart installations. It allows for systems that require a new build or rebuild to be flipped over and provisioned without disturbing the rest of the network.
- **DMZ** - A demilitarised zone would contain any externally-facing services, this could be setup in conjunction with the external networks access depending on the services and traffic passing through.

The above networks could be physically or virtually separated from one another. In a physical separation scenario there will be a separate network switch for each one, preventing any sort of cross-communication. In a virtually separated network there will be multiple bridged switches that separate traffic by dedicating ports (or tagging traffic) to different VLANs. The benefit of the VLAN solution is that the bridged switches (along with bonded network interfaces) provides additional network redundancy.

**Note:** If a cloud environment is being used then it is most likely that all systems will reside on the primary network and no others. This is due to the network configuration from the cloud providers.

### 3.1.3 Operating System

There are many different operating systems which can provide the base software stability and functionality for a given workflow. With OS selection it is worth considering:

- Package versions and support for the intended software usage
- Documentation and support availability
- Licensing costs

This community project uses CentOS 7 as it is stable and free with a large amount of documentation and community support available.

### 3.1.4 Node Structure

A cluster will most likely be comprised of systems that serve different purposes within the network. Ideas of node types along with the services and purpose of those nodes can be seen below.

- **Login Node** - A login node will usually provide access to the cluster and will be the central system that users access to run applications. How users will access the system should be considered, usually this will be SSH and some graphical login service, such as, VNC.
- **Master Node** - A master node will usually run services for the cluster. Such as, the master process for a job scheduler, monitoring software and user management services.
- **Compute Node** - Compute nodes are usually used for running HPC applications that are queued through a job scheduler. Additionally, these can be used for VM deployments (via software like OpenStack) or other computational uses. Compute nodes usually have large amounts of cores and memory as well as high bandwidth interconnect (like Infiniband).
- **Special-purpose Node** - Some compute nodes may feature a particular specification to be used for a particular job, or stage in your workflow. Examples may include nodes with more memory, larger amounts of local scratch storage, or GPU/FPGA devices installed.
- **Storage Node** - The storage node will serve network storage solutions to systems on the network. It would have some sort of storage array connected to it which would provide large and resilient storage.

The above types are not strict. Services can be mixed, matched and moved around to create the desired balance and distribution of services and functions for the architecture and workflow.

### 3.1.5 Resilience

How well a system can cope with failures is crucial when designing cluster architecture. Adequate resilience can allow for maximum system availability with a minimal chance of failures disrupting the user. System resilience can be improved with many hardware and software solutions, such as:

- **RAID Arrays** - A RAID array is a collection of disks configured in such a way that they become a single storage device. There are different RAID levels which improve data redundancy or storage performance (and maybe even both). Depending on the RAID level used, a disk in the array can fail without disrupting the access to data and can be hot swapped to rebuild the array back to full functionality.[1]
- **Service Redundancy** - Many software services have the option to configure a slave/failover server that can take over the service management should the master process be unreachable. Having a secondary server that mirrors critical network services would provide suitable resilience to master node failure.
- **Failover Hardware** - For many types of hardware there is the possibility of setting up failover devices. For example, in the event of a power failure (either on the circuit or in a power supply itself) a redundant power supply will continue to provide power to the server without any downtime occurring.

There are many more options than the examples above for improving the resilience of the cluster, it is worth exploring and considering available solutions during design.

**Note:** Cloud providers are most likely to implement all of the above resilience procedures and more to ensure that their service is available at least 99.99% of the time.

### 3.1.6 Hostname and Domain Names

Using proper domain naming conventions during design of the cluster architecture is best practice for ensuring a clear, logical and manageable network. Take the below fully qualified domain name:

```
node01.pri.cluster1.compute.estate
```

Which can be broken down as follows:

- `node01` - The hostname of the system
- `pri` - The network that the interface of the system is sat on (in this case, pri = primary)
- `cluster1` - The cluster that `node01` is a part of
- `compute` - The subdomain of the greater network that `cluster1` is a part of
- `estate` - The top level domain

### 3.1.7 Security

Network security is key for both the internal and external connections of the cluster. Without proper security control the system configuration and data is at risk to attack or destruction from user error. Some tips for improving network security are below:

[1] For more information on RAID arrays see https://en.wikipedia.org/wiki/RAID

- Restrict external access points where possible. This will reduce the quantity of points of entry, minimising the attack surface from external sources.
- Limit areas that users have access to. In general, there are certain systems that users would never (and should never) have access to so preventing them from reaching these places will circumvent any potential user error risks.
- Implement firewalls to limit the types of traffic allowed in/out of systems.

It is also worth considering the performance and usability impacts of security measures.

Much like with resilience, a Cloud provider will most likely implement the above security features - it is worth knowing what security features and limitations are in place when selecting a cloud environment.

**Note:** Non-Ethernet networks usually cannot usually be secured to the same level as Ethernet so be aware of what the security drawbacks are for the chosen network technology.

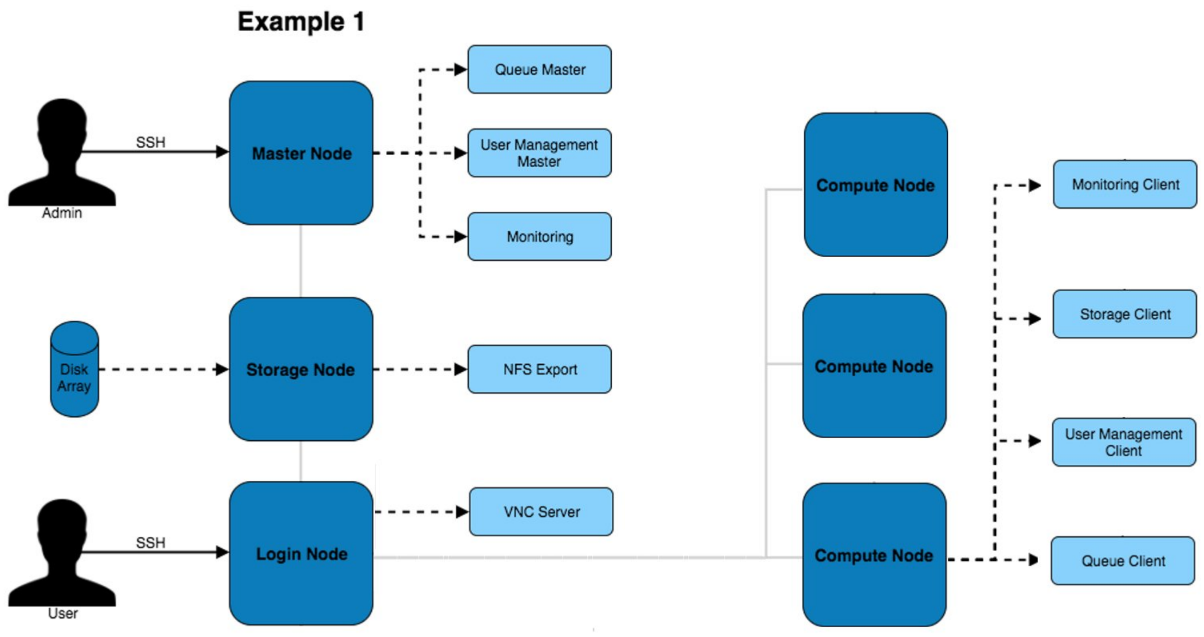
## 3.2 Architecture Considerations: Network and Hardware Design

At Alces software, the recommended network design differs slightly depending on the number of users and quantity of systems within the HPC platform.

### 3.2.1 Cluster Architectures

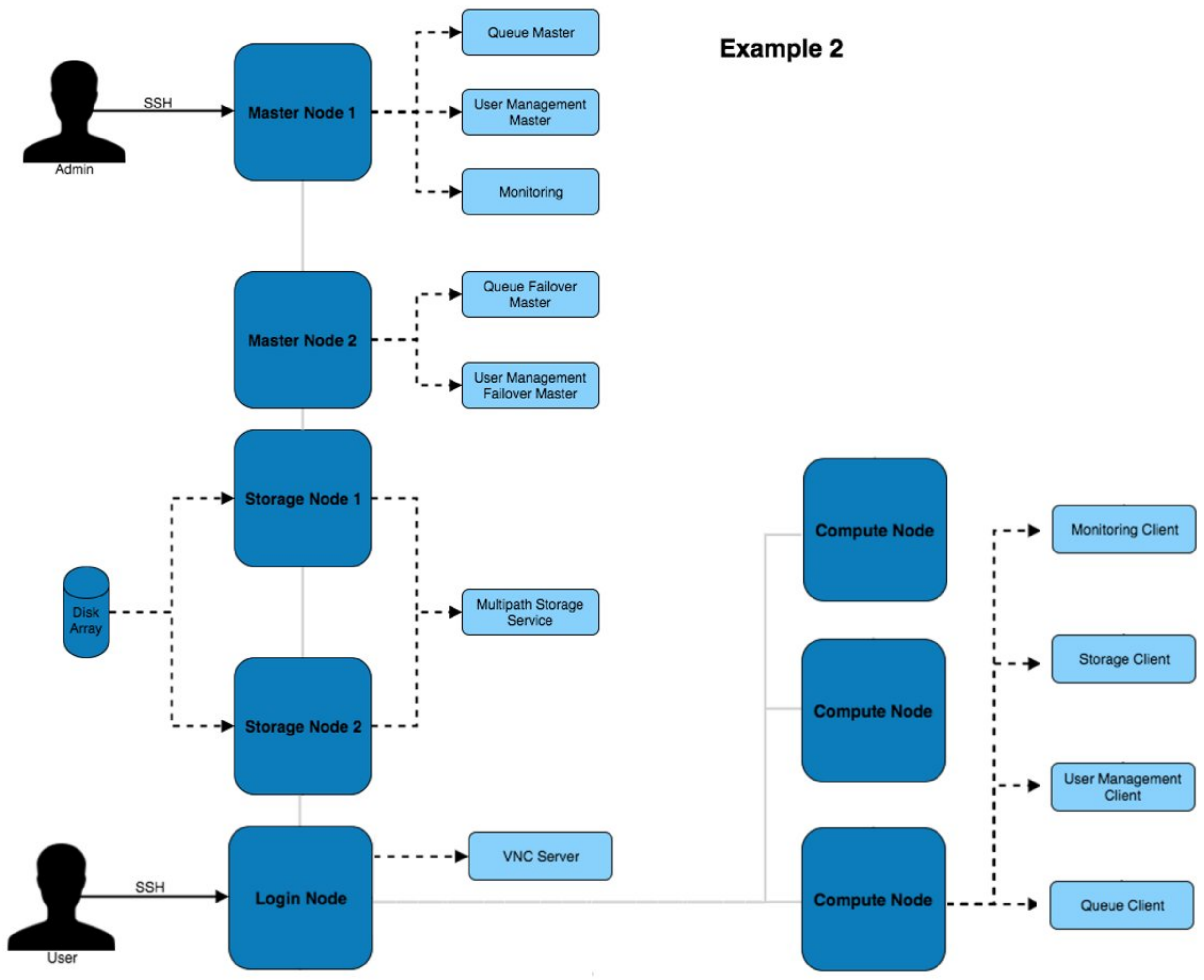
With the *General Architecture Considerations* in mind, diagrams of different architectures are below. They increase in complexity and redundancy as the list goes on.

#### Example 1 - stand-alone



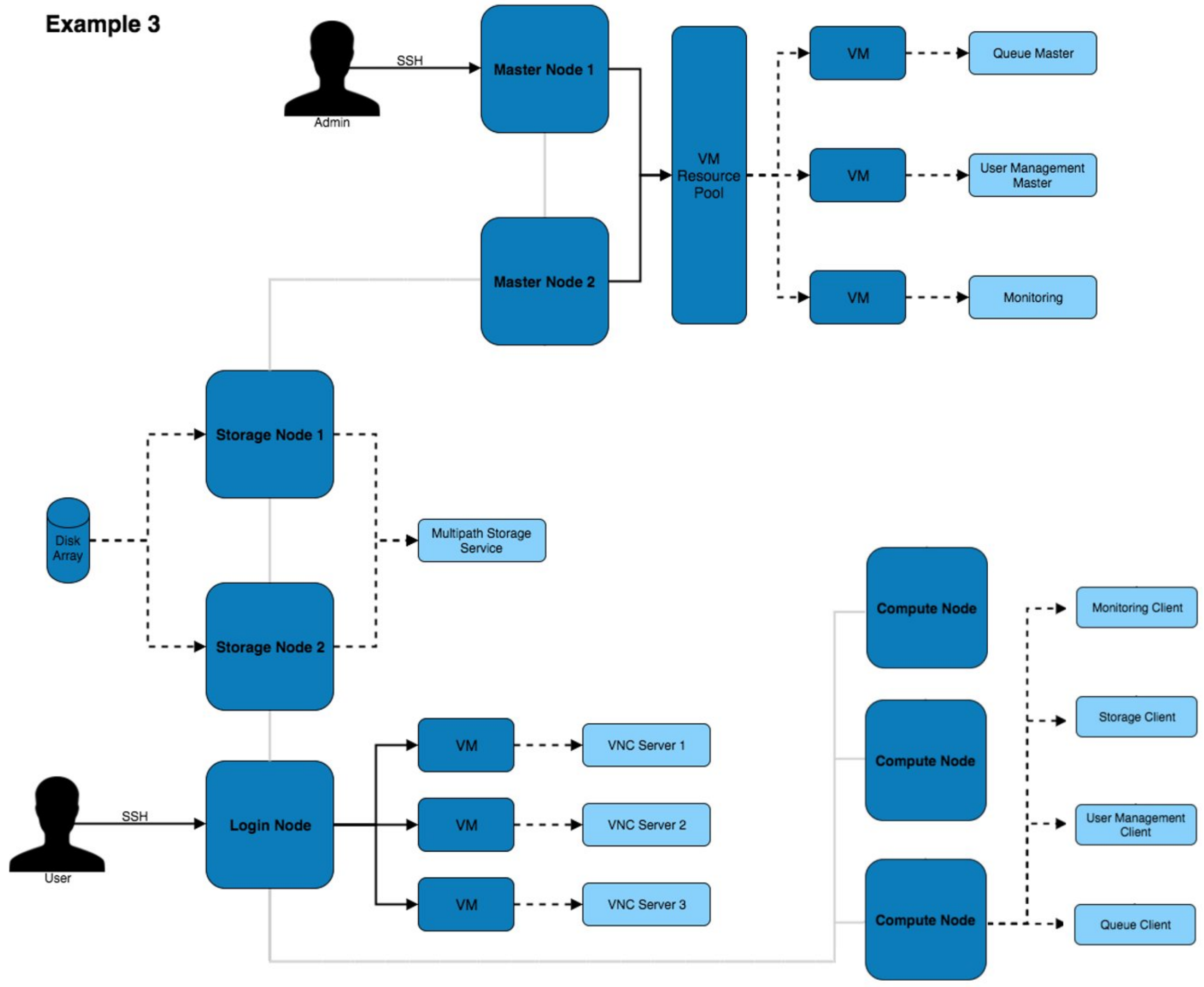
The above architecture consists of master, login and compute nodes. The services provided by the master & login nodes can be seen to the right of each node type. This architecture only separates the services for users and admins.

### Example 2 - high-availability



This architecture provides additional redundancy to the services running on the master node. For example, the disk array is connected to both master nodes which use multipath to ensure the higher availability of the storage device.

### Example 3 - HA VMs

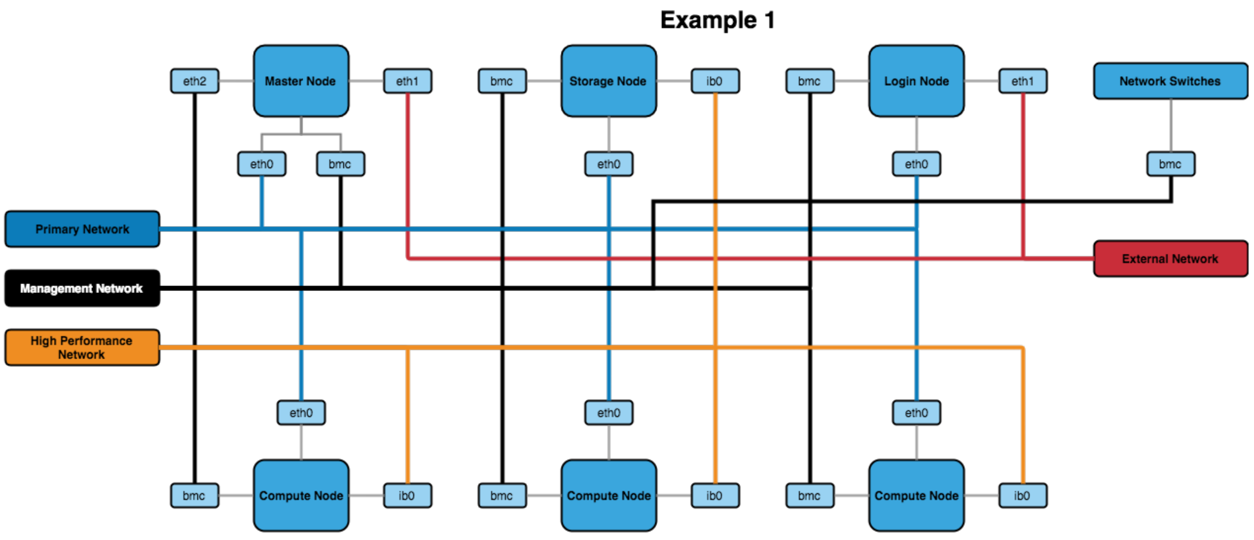


This architecture puts services inside of VMs to improve the ability to migrate and modify services with little impact to the other services and systems on the architecture. Virtual machines can be moved between VM hosts live without service disruption allowing for hardware replacements to take place on servers.

## 3.2.2 Network Designs

The above architectures can be implemented with any of the below network designs.

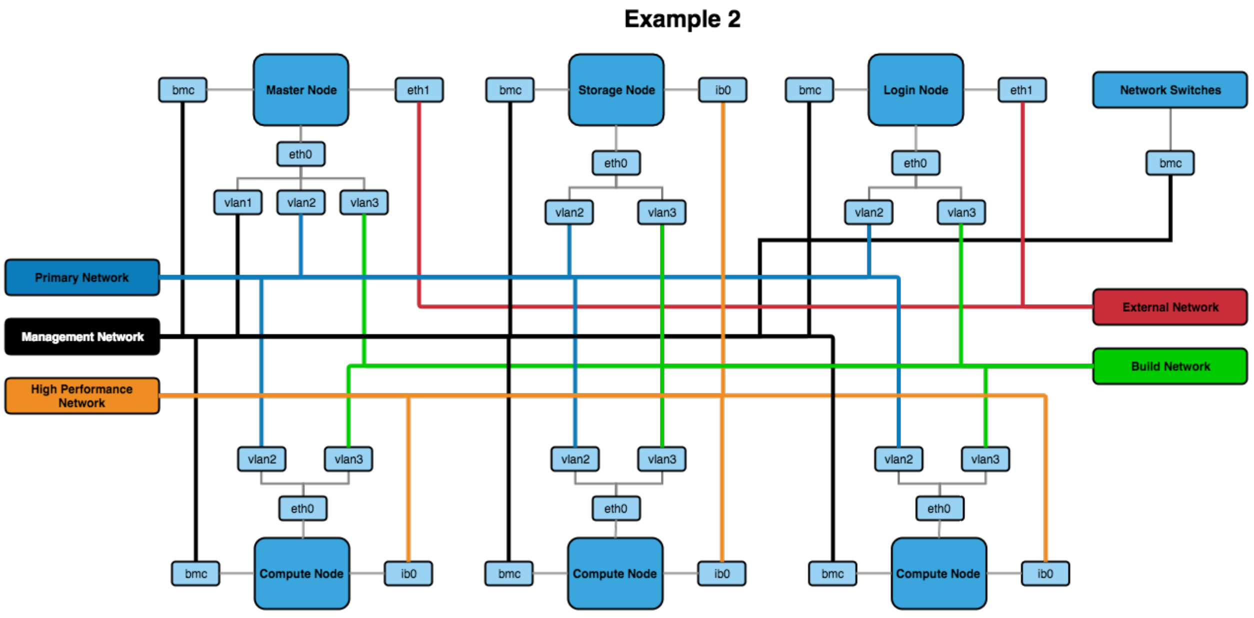
### Example 1- simple



The above design contains the minimum recommended internal networks. A primary network (for general logins and navigating the system), a management network (for BMC management of nodes and switches) and a high performance Infiniband network (connected to the nodes). The master and login nodes have access to the external network for user and admin access to the HPC network.

**Note:** The master node could additionally be connected to the high performance network so that compute nodes have a faster network connection to storage.

### Example 2 - VLANs



The above network design has a few additions to the first example. The main change is the inclusion of VLANs for the primary, management and build networks (with the build network being a new addition to this design). The build network allows for systems to be toggled over to a DHCP system that uses PXE booting to kickstart an OS installation.

### 3.2.3 Other Recommendations

#### BIOS Settings

It's recommended to ensure that the BIOS settings are reset to default and the latest BIOS version is installed before optimising the settings. This can ensure that any issues that may be present in the configuration before proceeding have been removed.

When it comes to optimising the BIOS settings on a system in the network, the following changes are recommended:

- Setting the power management to maximum performance
- Disabling CPU CStates
- Disabling Hyperthreading
- Enabling turbo mode
- Disabling quiet boot
- Setting BMC to use the dedicated port for BMC traffic
- Setting the node to stay off when power is restored after AC power loss

**Note:** The wordings for settings above may differ depending on the hardware that is being used. Look for similar settings that can be configured to achieve the same result.

## 3.3 Architecture Considerations: Infrastructure Design

Infrastructure design largely relates to the considerations made for the *Cluster Architectures*. Depending on the design being us ed, some of the infrastructure decisions may have already been made.

### 3.3.1 Infrastructure Service Availability

There are typically 3 possible service availability options to choose from, these are:

- All-in-one
- VM Platform
- High Availability VM Platform

**Note:** If using a Cloud Platform then the service availability will be handled by the cloud provider. The only additional considerations are how services will be provided in terms of native running services or containerised.

These are covered in more detail below.

#### All-in-one

This is the most common solution, an all-in-one approach loads services onto a single machine which serves the network. It is the simpl est solution as a single OS install is required and no additional configuration of virtual machine services is needed.

This solution, while quick and relatively easy to implement, is not a recommended approach. Due to the lack of redundancy options and t he lack of service isolation there is a higher risk of an issue effecting one service (or the machine) to have an effect on other servi ces.

#### VM Platform

A VM platform provides an additional layer of isolation between services. This can allow for services to be configured, migrated and modified without potentially effecting other services.

There are a number of solutions for hosting virtual machines, including:

- Open-source solutions (e.g. VirtualBox, KVM, Xen)
- Commercial solutions (e.g. VMware)

The above software solutions provide similar functionality and can all be used as a valid virtualisation platform. Further investigation into the ease of use, flexibility and features of the software is recommended to identify the ideal solution for your HPC platform.

#### High Availability VM Platform

For further redundancy, the virtualisation platform can utilise a resource pool. The service will be spread across multiple machines which allows for VMs to migrate between the hosts whilst still active. This live migration can allow for one of the hosts to be taken off of the network for maintenance without impacting the availability of the service VMs.

### 3.3.2 Node Network Configuration

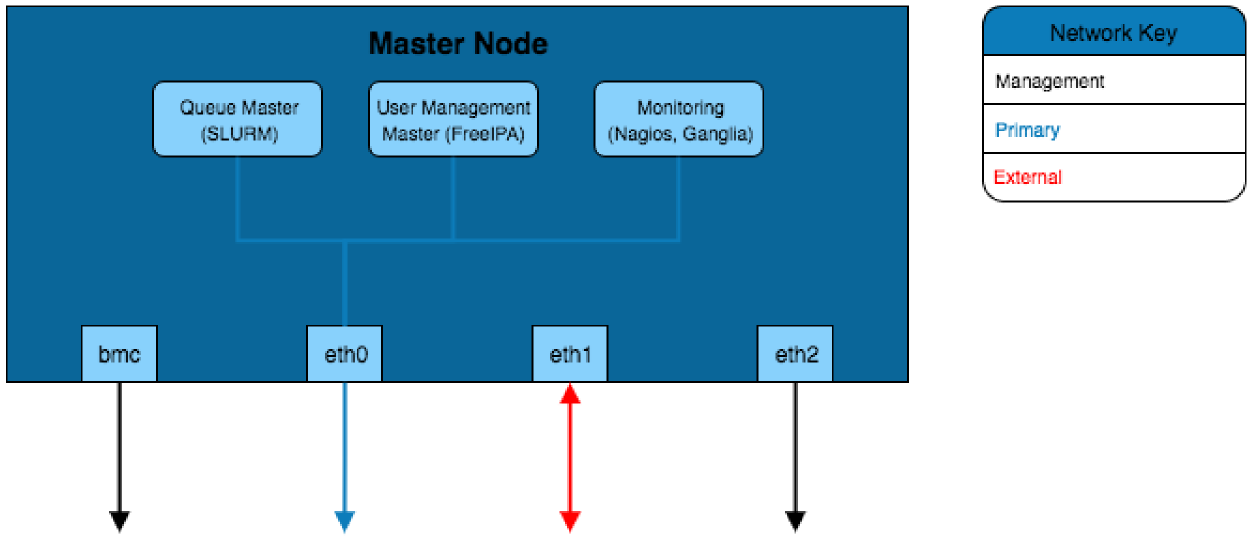
In addition to the availability of services, the network configuration on the node can provide better performance and redundancy. Some of the network configuration options that can improve the infrastructure are:

- **Channel Bonding** - Bonding interfaces allows for traffic to be shared between 2 network interfaces. If the bonded interfaces are connected to separate network switches then this solution
- **Interface Bridging** - Network bridges are used by interfaces on virtual machines to connect to the rest of the network. A bridge can sit on top of a channel bond such that the VM service network connection is constantly available.
- **VLAN Interface Tagging** - VLAN management can be performed both on a managed switch and on the node. The node is able to create subdivisions of network interfaces to add VLAN tags to packets. This will create separate interfaces that can be seen by the operating system (e.g. eth0.1 and eth0.2) which can individually have IP addresses set.

### 3.3.3 Configuration Examples

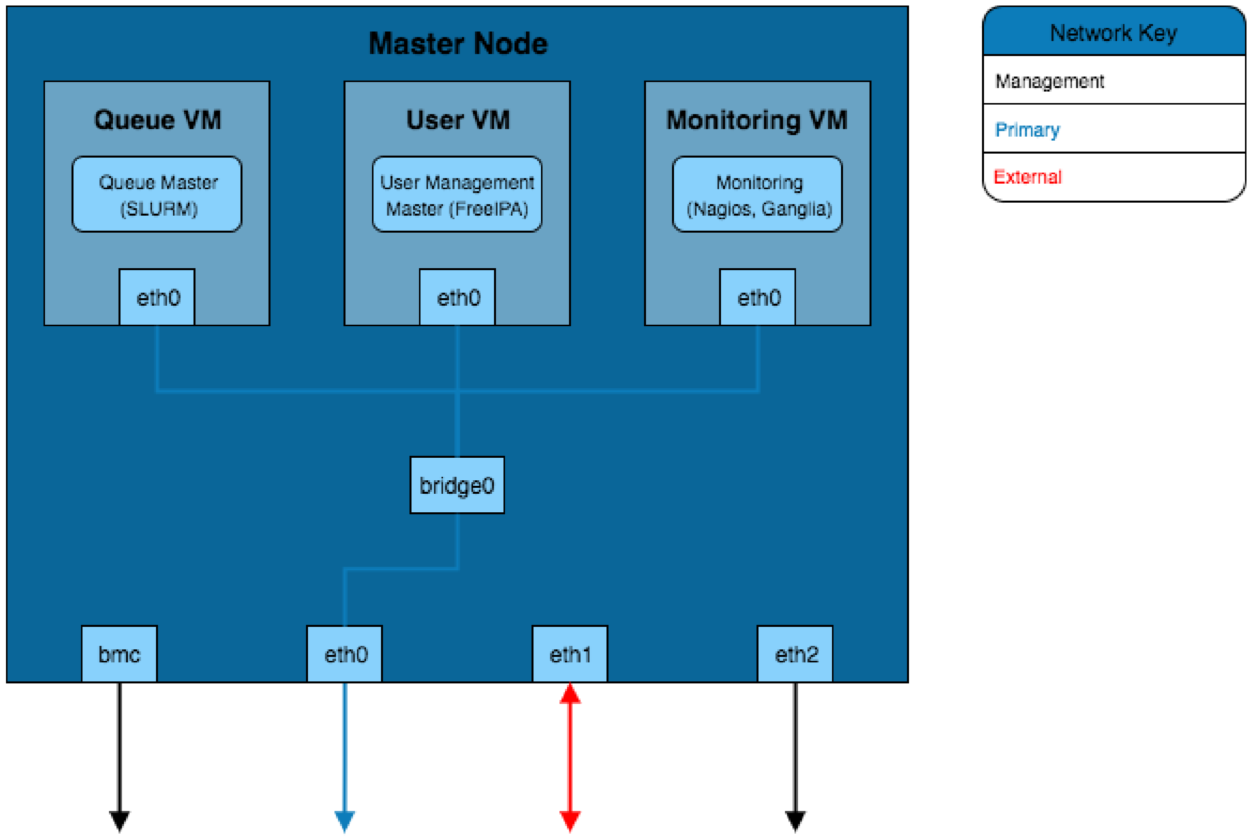
The example configurations here combine elements of the *Architecture Considerations: Network and Hardware Design*. These focus on the internal configuration of the master node but these examples can be extrapolated for configuring login, storage, compute or any other nodes that are part of the HPC environment.

### Simple Infrastructure



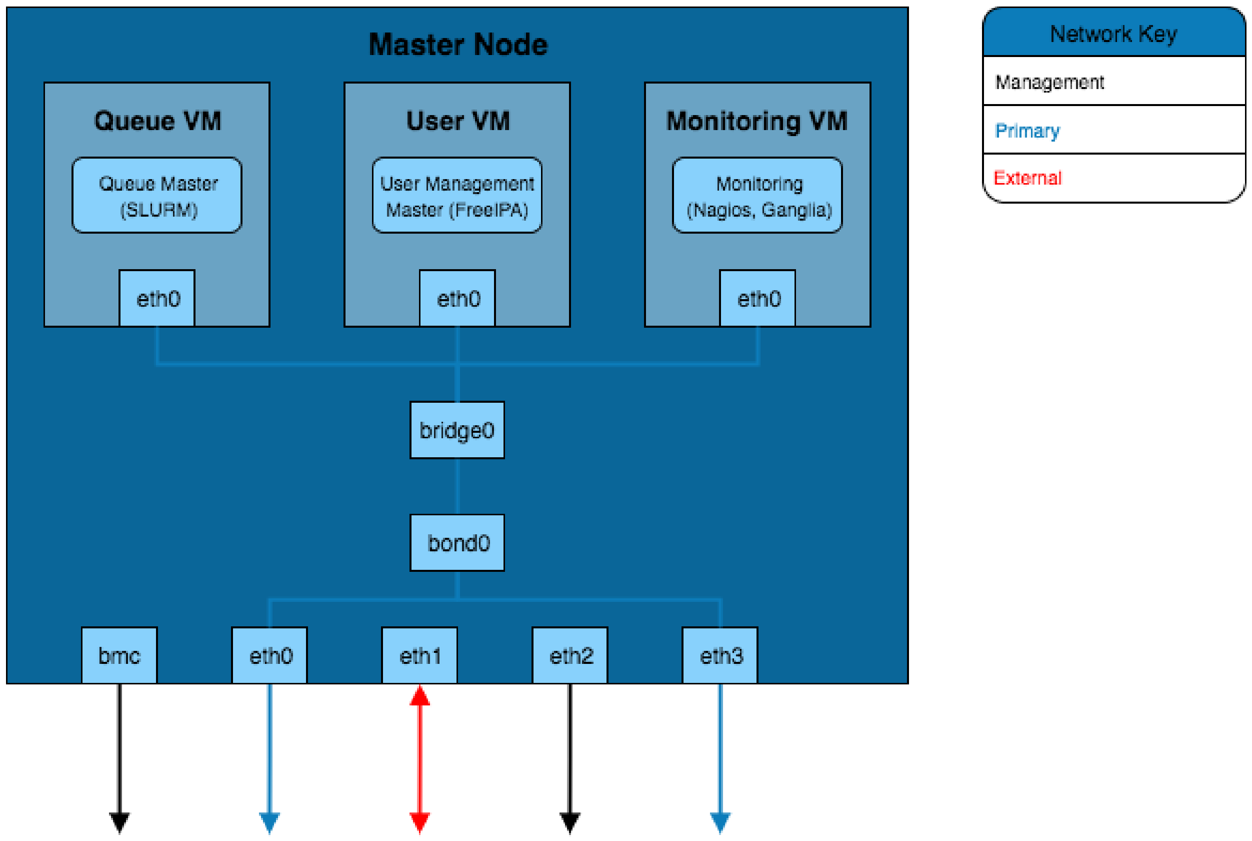
The simplest infrastructure configuration uses the all-in-one approach where services are configured on the master node's operating system.

### Virtual Machine Infrastructure



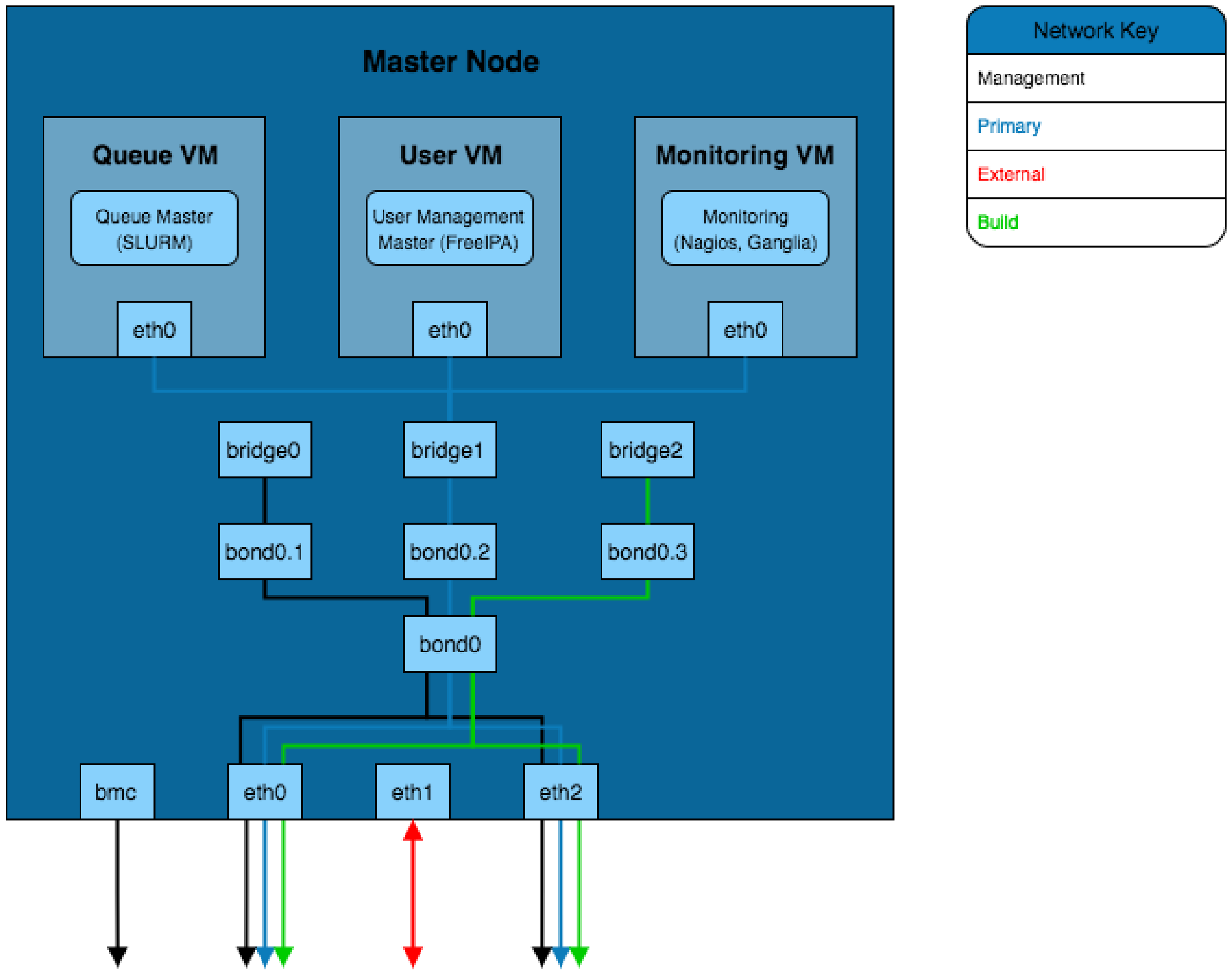
This solution separates the services into VMs running on the master node. In order for these VMs to be able to connect to the primary network a network bridge is created that allows the VM interfaces to send traffic over the eth0 interface.

### Channel Bonded Infrastructure



This example adds a layer of redundancy over the *VM Infrastructure* design by bonding the eth0 and eth3 interfaces. These interfaces are connected to separate network switches (the switches will be bridged together as well) which provides redundancy should a switch or network interface fail. Bonding of the two interfaces creates a new *bond* interface that the bridge for the virtual machines connects to.

### VLAN Infrastructure



The above solution implements the channel bonded infrastructure in a network with VLANs. The VLANs have bond and bridge interfaces created for them. This allows some additional flexibility for VM bridging as virtual interfaces can be bridged onto specific VLANs whilst maintaining the redundancy provided by the bond. This adds additional security to the network as the master node can be left without an IP on certain VLAN bond interfaces which prevents that network from accessing the master node whilst VMs on the master node are able to reach that VLAN.

## 3.4 Architecture Considerations: Storage Design

### 3.4.1 Storage Hardware

When selecting the storage solution it is worth considering the size, performance and resilience of the desired storage solution. Usually some sort of storage array will be used; e.g. a collection of disks (otherwise known as JBOD - Just a Bunch Of Disks) in the form of an internal or external RAID array.

### 3.4.2 Type of filesystem

For many requirements, a simple NFS solution can provide sufficient performance and resiliency for data. Application, library and source-code files are often small enough to be stored on an appropriately sized NFS solution; such storage systems can even be grown over time using technologies like LVM and XFS as requirements increase.

For data-sets that require high capacity (>100TB) or high-performance (>1GB/sec) access, a parallel filesystem may be suitable to store data. Particularly well suited to larger files (e.g. at least 4MB per storage server), a parallel filesystem can provide additional features such as byte-range locking (the ability for multiple nodes to update sections of a large file simultaneously), and MPI-IO (the ability to control data read/written using MPI calls). Parallel filesystems work by aggregating performance and capacity from multiple servers and allowing clients (your cluster compute and login nodes) to mount the filesystem as a single mount-point. Common examples include:

- Lustre; an open-source kernel-based parallel filesystem for Linux
- GPFS (general purpose file system; a proprietary kernel-based parallel filesystem for Linux
- BeeGFS; an open-source user-space parallel filesystem for Linux

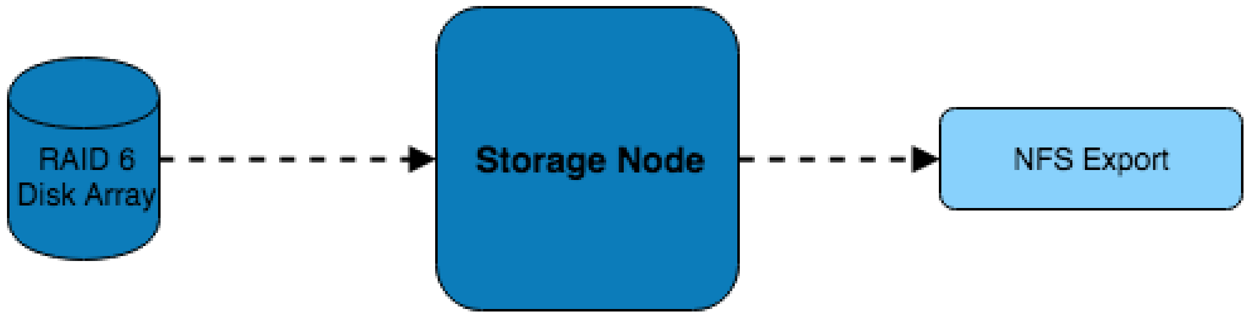
Your choice of filesystem will depend on the features you require, and your general familiarity with the technologies involved. As parallel filesystems do not perform well for smaller files, it is very common to deploy a parallel filesystem alongside a simple NFS-based storage solution.

If your data-set contains a large number of files which need to be kept and searchable for a long time (> 12-months) then an object storage system can also be considered. Accessed using client-agnostic protocols such as HTTPS, an object storage system is ideal for creating data archives which can include extended metadata to assist users to locate and organise their data. Most object storage systems include data redundancy options (e.g. multiple copies, object versioning and tiering), making them an excellent choice for long-term data storage. Examples of object-storage systems include:

- AWS Simple Storage Service (S3); a cloud-hosted service with a range of data persistence options
- Swift-stack; available as both on-premise and cloud-hosted services, the SWIFT protocol is compatible with a wide range of software and services
- Ceph; an on-premise object storage system with a range of interfaces (block, object, S3, POSIX file)

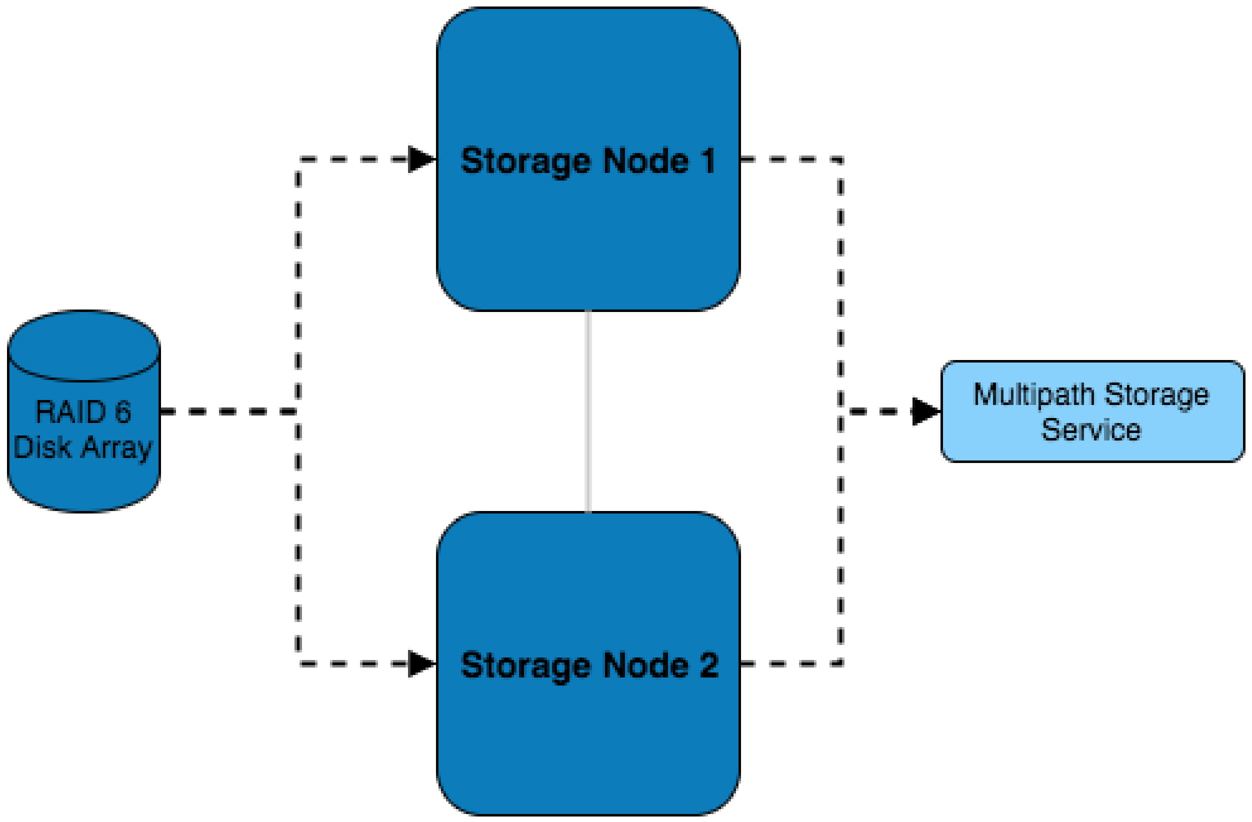
### 3.4.3 Network Storage Solutions

#### Single server with NFS



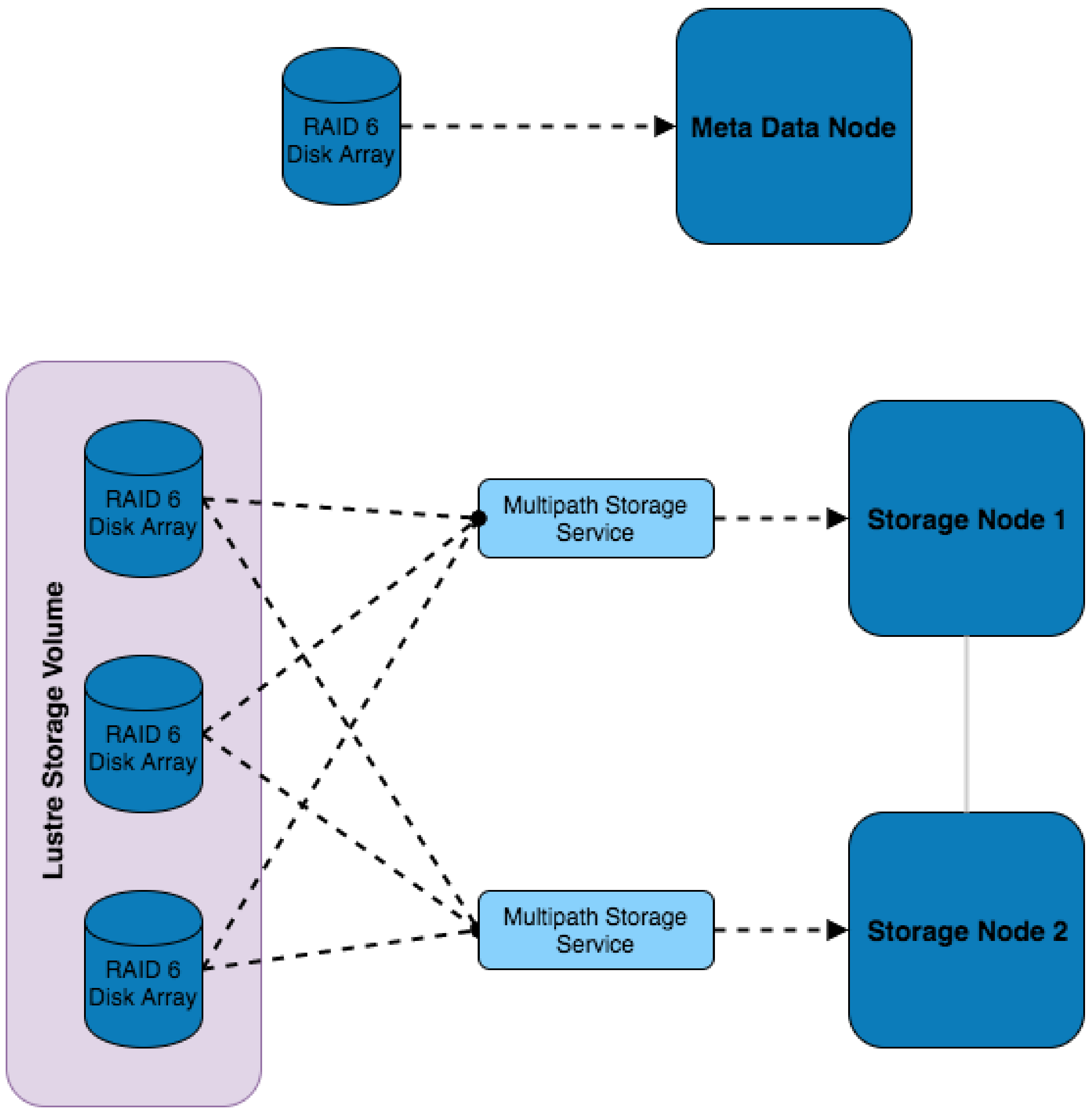
In this example, a single server is connected to a RAID 6 storage array which it is serving over NFS to the systems on the network. While simple in design and implementation, this design only provides redundancy at the RAID level.

### Multiple Servers with NFS



In addition to the previous example, this setup features multiple storage servers which balance the load of serving the disk over NFS.

#### Multiple Servers with Parallel filesystem



This setup features multiple RAID sets which are installed externally to the storage servers and are connected to both of them using multipath - this allows for multiple paths to the storage devices to be utilised. Using this storage, a Lustre volume has been configured which consists of a combination of all the external disks. Authorisation of access to the storage volume is managed by the metadata node, which also has dedicated storage.

### 3.4.4 Additional Considerations and Questions

- What data will need to be centrally stored?
- Where will data be coming from?

  - Are source files created within the HPC network or do they exist in the external network?
  - Will compute nodes be writing out logs/results from running jobs?
  - Where else might data be coming from?
- Is scratch space needed?
- What level of redundancy/stability is required for the data?
- How will the data be backed up?
  - Will there be off-site backups?
  - Should a separate storage medium be used?
  - Does all the data need backing up or only certain files?
  - For how long will we keep the data, and any backups created?
- What are my disaster recovery and business continuity plans?
  - If the storage service fails, how will my users continue working?
  - How can I recreate the storage service if it fails?
  - How long will it take to restore any data from backups?

## 3.5 Platform Considerations: General

Platform considerations broadly cover how the chosen platform will be interacted with, namely:

- Where to host the platform
- How to deploy the resources
- How to manage nodes

### 3.5.1 Deployment Location

With regards mainly to cloud platforms, the location of the resources is an important consideration. Cloud platforms offer their services in a variety of locations which allows for localisation of services and potential distribution of services across the globe.

So when it comes to deciding on where to deploy resources it's worth considering where the user base is located.

### 3.5.2 Deployment Methods

No matter what type of platform is being used for resources, there are multiple methods to deploy node configurations.

Deployment methods can include:

- **Manual** - Creating all deployment templates by hand for resources
- **Assisted** - Using helper tools to generate templates and files for creating resources

In most cases there is also a GUI and CLI tool that can be used to deploy to the platform. The ideal tool will suit the needs of the deployment team in ease-of-use and flexibility.

### 3.5.3 Platform Management

Management involves mainly the power management on the system. How the resources can be controlled remotely to check or change the power state.

Remote power management tools can provide group-control helpers and other bulk tools to allow for easier maintenance preparation.

## 3.6 Environment Considerations