HPC Architectures

Types of resource currently in use

Reusing this material

Outline

- Shared memory architectures
- Distributed memory architectures
- Distributed memory with shared-memory nodes
- Accelerators
- Filesystems

- What is the difference between different tiers of machine?
  - Interconnect
  - Software
  - Job-size bias (capability)

Shared memory architectures

Simplest to use, hardest to build

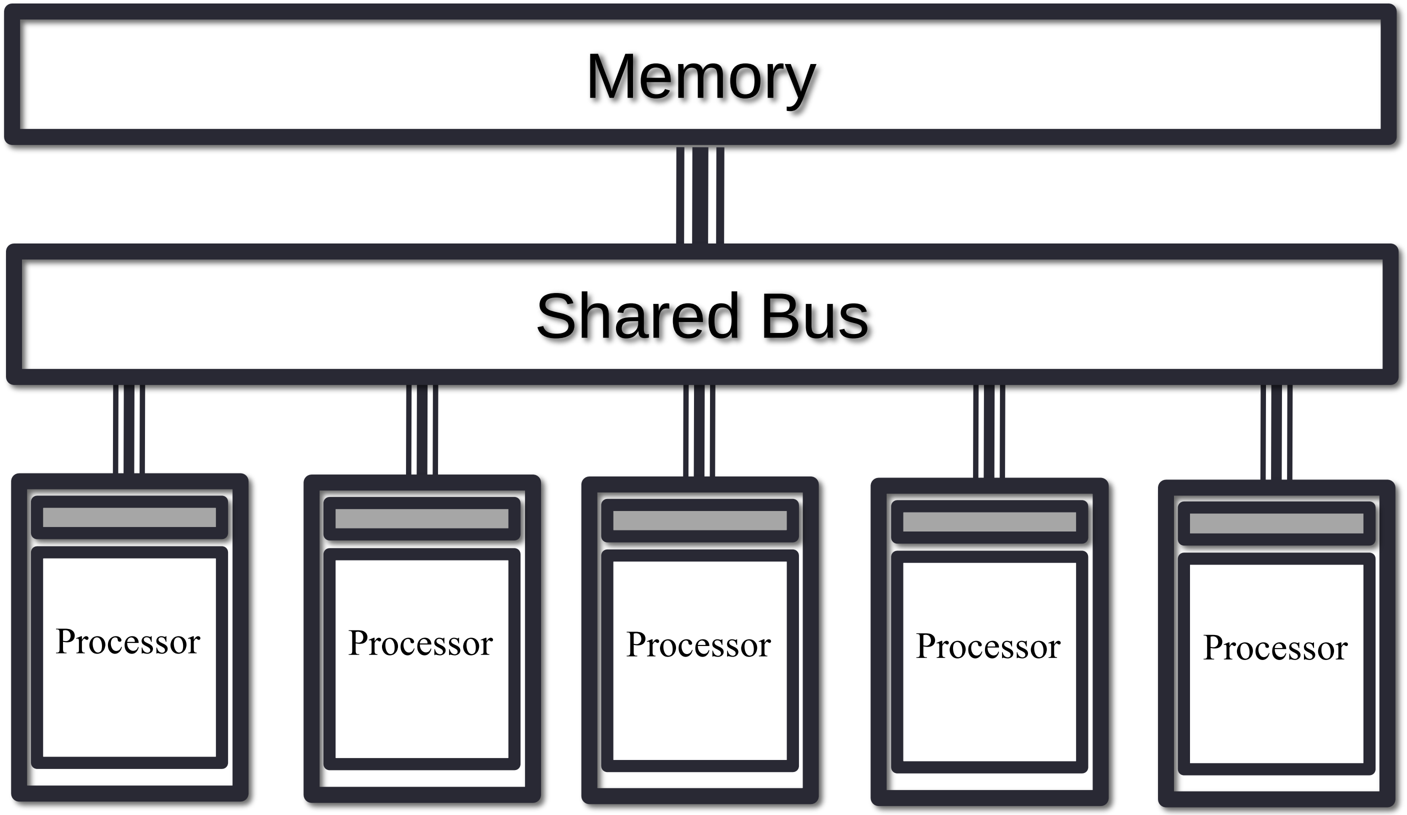
Shared-Memory Architectures

- Multi-processor shared-memory systems have been common since the early 90's
  - originally built from many single-core processors
  - multiple sockets sharing a common memory system

- A single OS controls the entire shared-memory system

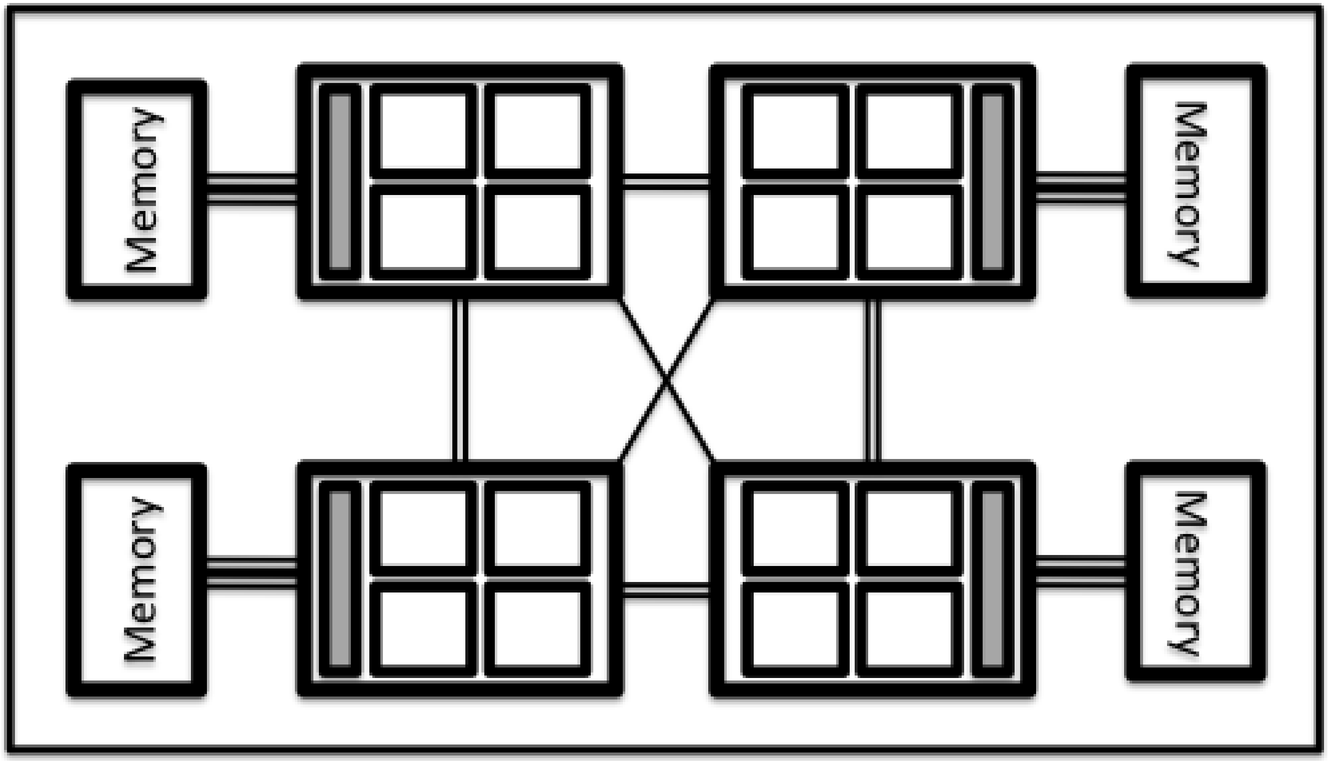
- Modern multicore processors are just shared-memory systems on a single chip
  - can't buy a single core processor even if you wanted one!

Symmetric Multi-Processing Architectures



- All cores have the same access to memory, e.g. a multicore laptop

Non-Uniform Memory Access Architectures



- Cores have faster access to their own local memory

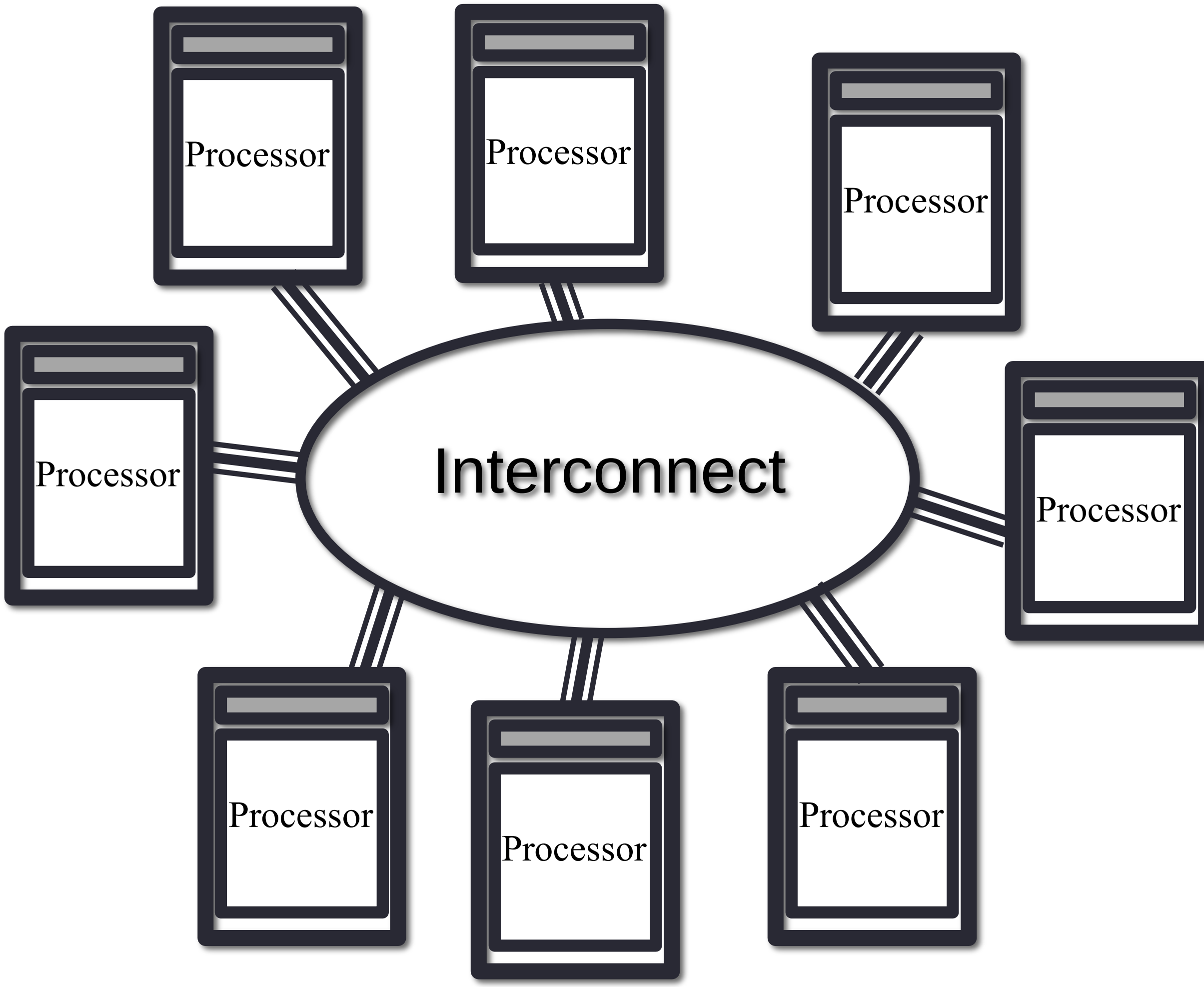
Shared-memory architectures

- Most computers are now shared memory machines due to multicore
- Some true SMP architectures…
  - *e.g.* BlueGene/Q nodes
- …but most are NUMA
  - Program NUMA as if they are SMP – details hidden from the user
  - all cores controlled by a single OS
- Difficult to build shared-memory systems with large core numbers (> 1024 cores)
  - Expensive and power hungry
  - Difficult to scale the OS to this level

Distributed memory architectures

Clusters and interconnects

Multiple Computers



- Each self-contained part is called a *node*.
  - each node runs its own copy of the OS

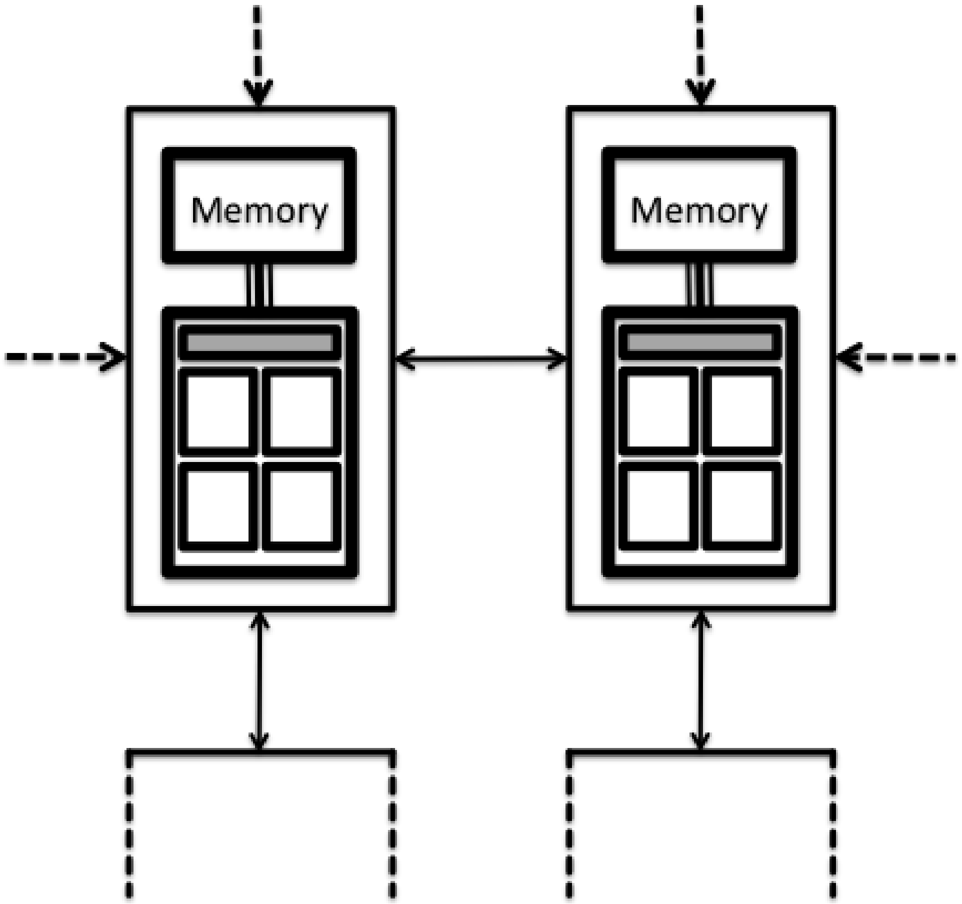
Distributed-memory architectures

- Almost all HPC machines are distributed memory
- The performance of parallel programs often depends on the *interconnect* performance
  - Although once it is of a certain (high) quality, applications usually reveal themselves to be CPU, memory or IO bound
  - Low quality interconnects (e.g. 10Mb/s – 1Gb/s Ethernet) do not usually provide the performance required
  - Specialist interconnects are required to produce the largest supercomputers. *e.g.* Cray Aries, IBM BlueGene/Q
  - Infiniband is dominant on smaller systems.
- High bandwidth relatively easy to achieve
  - low latency is usually more important and harder to achieve

Distributed/shared memory hybrids

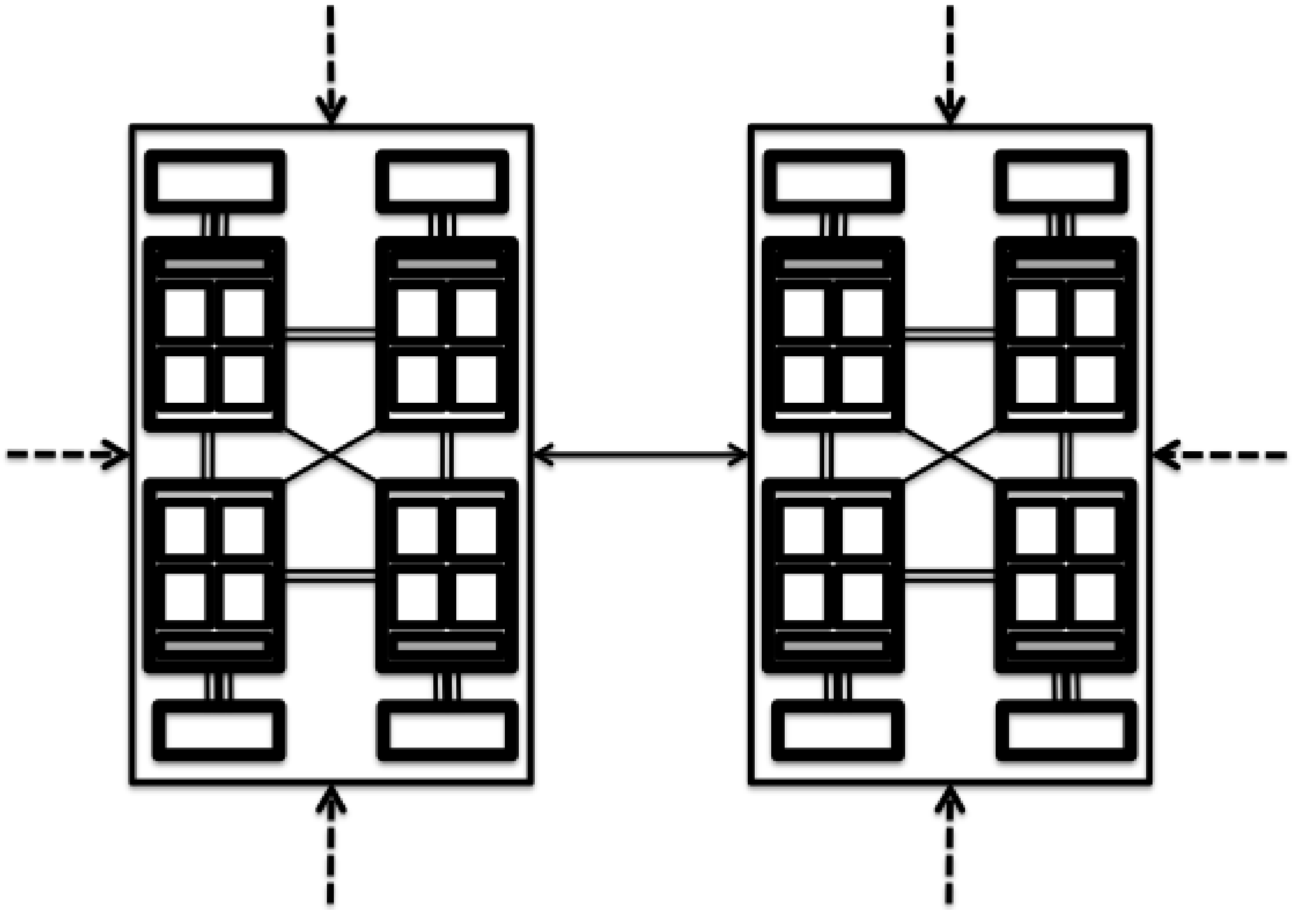
Almost everything now falls into this class

Multicore nodes



- In a real system:
  - each node will be a shared-memory system
    - e.g. a multicore processor
  - the network will have some specific topology
    - e.g. a regular grid

Hybrid architectures



- Now normal to have NUMA nodes
  - e.g. multi-socket systems with multicore processors
- Each node still runs a single copy of the OS

Hybrid architectures

- Almost all HPC machines fall in this class
- Most applications use a message-passing (MPI) model for programming
  - Usually use a single process per core
- Increased use of hybrid message-passing + shared memory (MPI+OpenMP) programming
  - Usually use 1 or more processes per NUMA region and then the appropriate number of shared-memory threads to occupy all the cores
- Placement of processes and threads can become complicated on these machines

Examples

- ARCHER has two 12-way multicore processors per node
  - 2 x 2.7 GHz Intel E5-2697 v2 (Ivy Bridge) processors
  - each node is a 24-core, shared-memory, NUMA machine
  - each node controlled by a single copy of Linux
  - 4920 nodes connected by the high-speed ARIES Cray network

- Cirrus has two 18-way multicore processors per node
  - 2 x 2.1 GHz Intel E5-2695 v4 (Broadwell) processors
  - each node is a 36-core, shared-memory, NUMA machine
  - each node controlled by a single copy of Linux
  - 280 nodes connected by the high-speed Infiniband (IB) fabric

cirrus
Powered by epcc

epcc

Accelerators

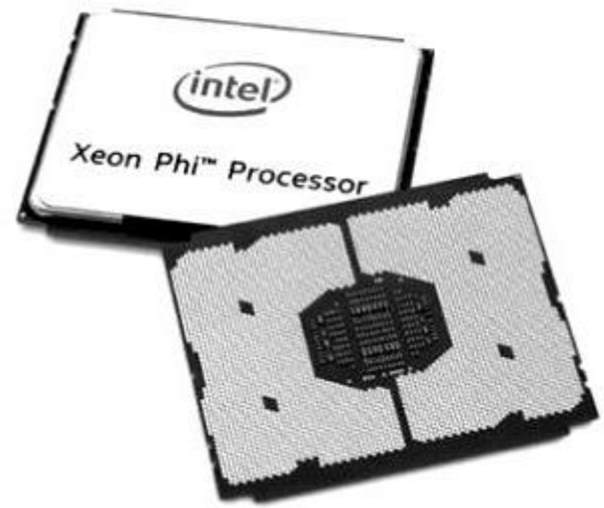
How are they incorporated?

Including accelerators

- Accelerators are usually incorporated into HPC machines using the hybrid architecture model
  - A number of accelerators per node
  - Nodes connected using interconnects
- Communication from accelerator to accelerator depends on the hardware:
  - NVIDIA GPU support direct communication
  - AMD GPU have to communicate via CPU memory
  - Intel Xeon Phi communication via CPU memory
  - Communicating via CPU memory involves lots of extra copy operations and is usually very slow

Example: ARCHER KNL

- 12 nodes with Knights Landing (Xeon Phi) recently added
- Each node has a 64-core KNL
  - 4 concurrent hyper-threads per core
  - Each node has 96GB RAM and each KNL has 16GB on chip memory



- The KNL is self hosted, i.e. in place of the CPU
  - Parallelism via shared memory (OpenMP) or message passing (MPI)
  - Can do internode parallelism via message passing
- Specific considerations needed for good performance

Example: Cirrus GPUs

- 2 nodes with NVIDIA Tesla V100 (Volta) GPUs
  - 4 GPUs per node

- These GPUs are not self-hosting
  - each GPU requires a host CPU
  - each has 2 x 2.4 GHz, 20-core Intel Xeon Gold (Skylake) CPUs
  - connected to CPU via PCIe bus

- Accessed via a special GPU queue: qsub –q gpu ...

epcc

THE UNIVERSITY OF EDINBURGH

Filesystems

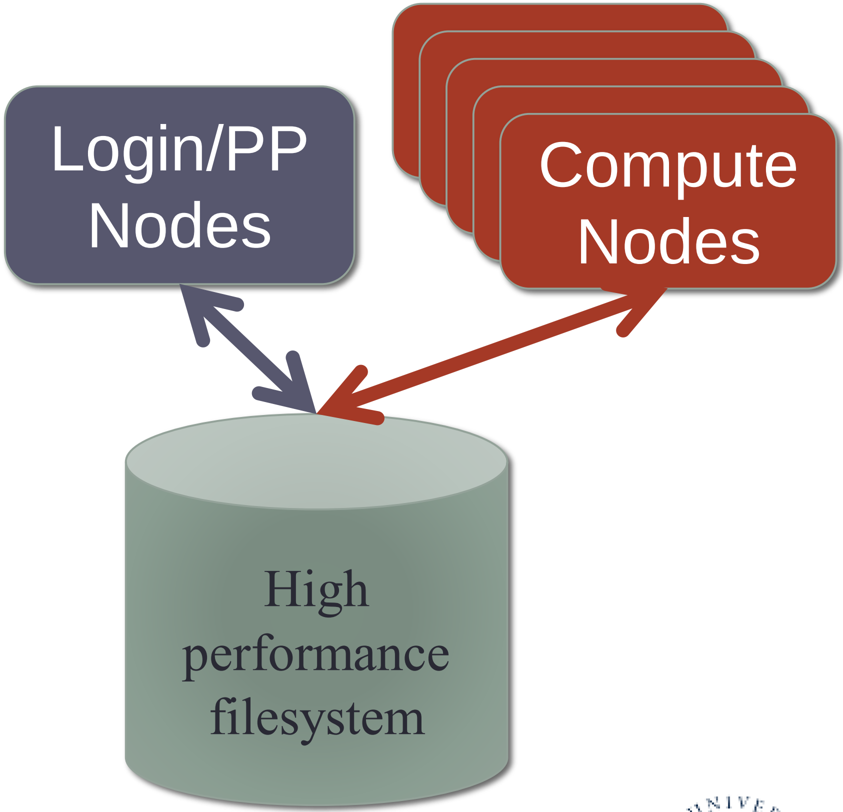
How is data stored?

High performance IO

- We have focused on the significant computation power of HPC machines so far
  - It is important that writing to and reading from the filesystem does not negate this
- High performance filesystems
  - Such as Lustre
  - Computational nodes are typically diskless and connect via the network to the filesystem
  - Due to its size this high performance filesystem is often NOT backed up
  - Connected to the nodes in two common ways
  - Not a silver bullet! There are lots of configuration and IO techniques which need to be leveraged for good performance and these are beyond the scope of this course.

Single unified filesystem

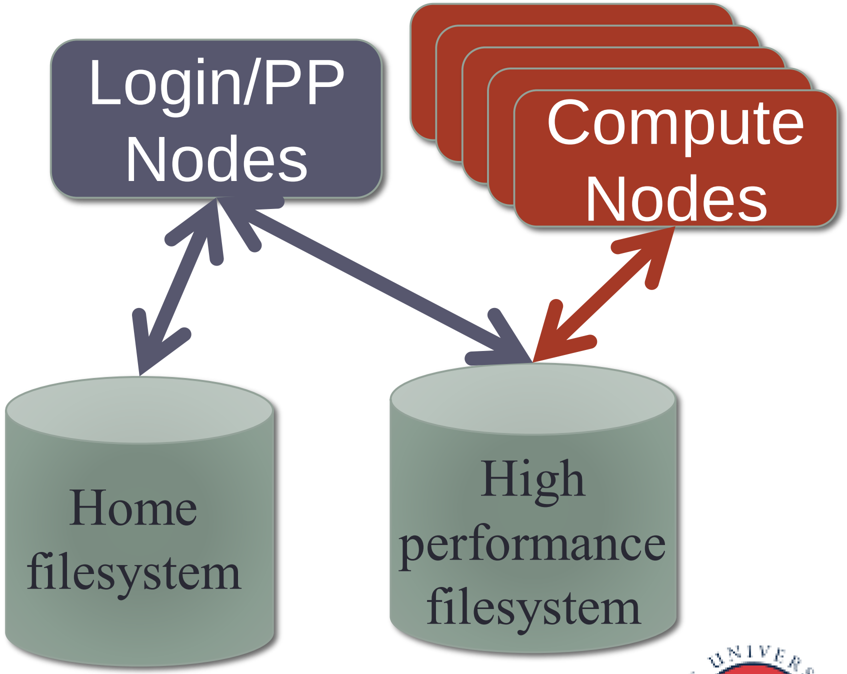
- One filesystem for the entire machine,
  - All parts of the machine can see this; all files are stored here
  - E.g. Cirrus (406 TiB Lustre FS)
- Advantages
  - Conceptually simple as all files are stored on the same filesystem
  - Preparing runs (e.g. compiling code) exhibits good IO performance
- Disadvantages
  - Lack of backup on the machine
  - This high performance filesystem can get clogged with significant unnecessary data (such as results from post processing/source code.)



Multiple disparate filesystems

- High performance filesystem focused on execution
  - Other filesystems for preparing & compiling code, as well as long term data storage
  - E.g. ARCHER which has an additional (low performance, huge capacity) long term data storage filesystem too

- Advantages
  - Home filesystem typically backed up
- Disadvantages
  - More complex as high performance FS is only one visible from compute nodes
- High performance FS may be called work or scratch

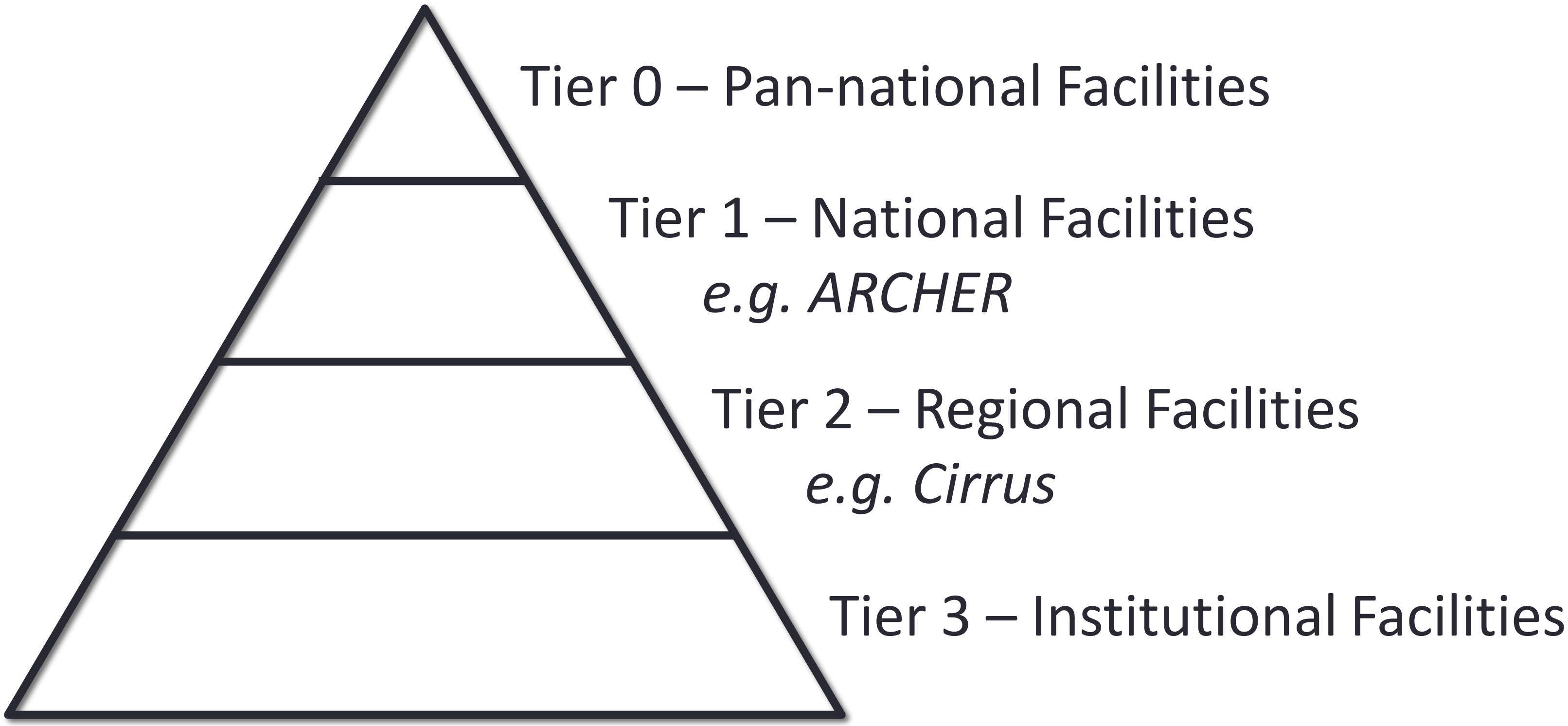


Comparison of machine types

What is the difference between different tiers of machine?

HPC Facility Tiers

- HPC facilities are often spoken about as belonging to *Tiers*



List of tier 2 facilities at
https://www.epsrc.ac.uk/research/facilities/hpc/tier2/

Summary

Summary

- Vast majority of HPC machines are shared-memory nodes linked by an interconnect.
  - Hybrid HPC architectures – combination of shared and distributed memory
  - Most are programmed using a pure MPI model (more later on MPI) - does not really reflect the hardware layout
- Accelerators are incorporated at the node level
  - Very few applications can use multiple accelerators in a distributed memory model
- Shared HPC machines span a wide range of sizes:
  - From Tier 0 – Multi-petaflops (1 million cores)
  - To workstations with multiple CPUs (+ Accelerators)