

# Introduction to ARC Systems

Presenter Name  
Advanced Research Computing  
Division of IT

*Feb 20, 2018*

# Before We Start

- Sign in
- Request account if necessary

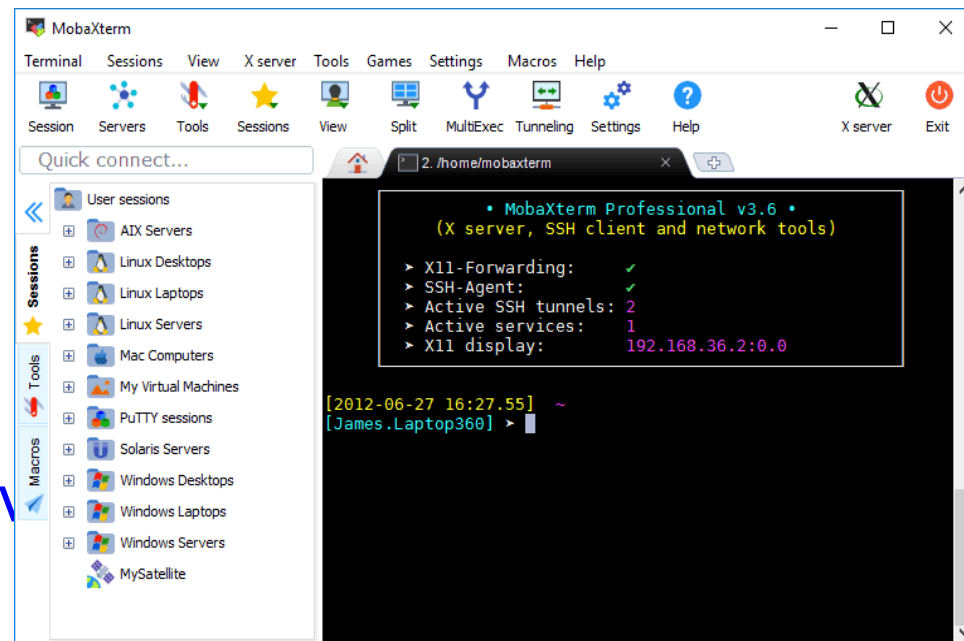
- Windows Users:

MobaXterm

PuTTY

Web Interface:

ETX [newriver.arc.vt.edu](http://newriver.arc.vt.edu)



# Today's goals

- Introduce ARC
- Give overview of HPC today
- Give overview of VT-HPC resources
- Familiarize audience with interacting with VT-ARC systems

# Should I Pursue HPC?

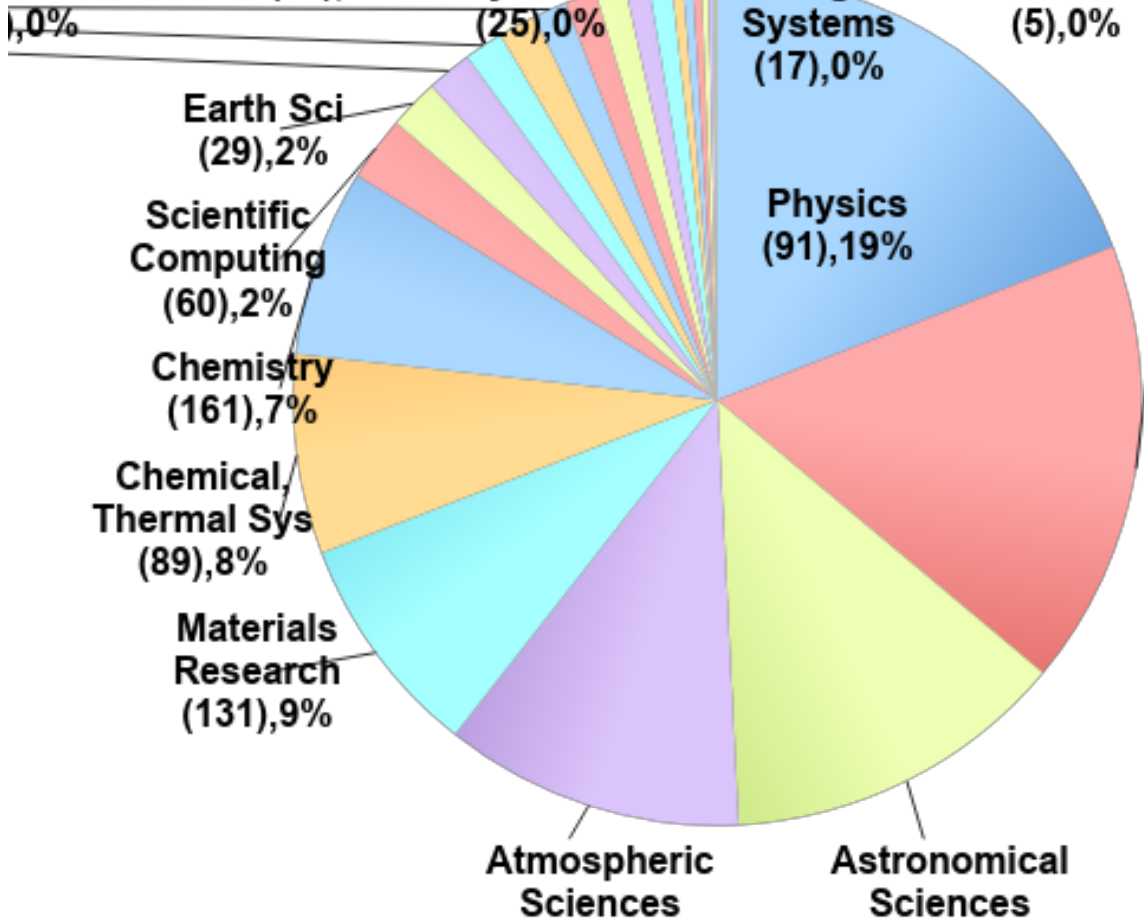
- Necessity: Are local resources insufficient to meet your needs?
  - Very large jobs
  - Very many jobs
  - Large data
- Convenience: Do you have collaborators?
  - Share projects between different entities
  - Convenient mechanisms for data sharing

# Research in HPC is Broad

- Earthquake Science and Civil Engineering
- Molecular Dynamics
- Nanotechnology
- Plant Science
- Storm modeling
- Epidemiology
- Particle Physics
- Economic analysis of phone network patterns
- Brain science
- Analysis of large cosmological simulations
- DNA sequencing
- Computational Molecular Sciences
- Neutron Science
- International Collaboration in Cosmology and Plasma Physics

# Who Uses HPC?

Behavioral and Neural Sciences (7), 0%  
Science and Engineering Education (5), 0%  
Information, Robotics, and Intelligent Systems (17), 0%  
Mechanical and Structural Systems (25), 0%  
Social and Economic Science (18), 0%



- **>2 billion core-hours allocated**
- **1400 allocations**
- **350 institutions**
- **32 research domains**

# Popular Software Packages

- Molecular Dynamics: Gromacs, LAMMPS, NAMD, Amber
- CFD: OpenFOAM, Ansys, Star-CCM+
- Finite Elements: Deal II, Abaqus
- Chemistry: VASP, Gaussian, PSI4, QCHEM
- Climate: CESM
- Bioinformatics: Mothur, QIIME, MPIBLAST
- Numerical Computing/Statistics: R, Matlab, Julia
- Visualization: ParaView, VisIt, EnSight
- Deep Learning: Caffe, TensorFlow, Torch, Theano

# Learning Curve

- Linux: Command-line interface
- Scheduler: Shares resources among multiple users
- Parallel Computing:
  - Need to parallelize code to take advantage of supercomputer's resources
  - Third party programs or libraries make this easier



# Advanced Research Computing

- Unit within the Office of the Vice President of Information Technology
- Provide centralized resources for:
  - Research computing
  - Visualization
- Staff to assist users
- Website: <http://www.arc.vt.edu>

# ARC Goals

- Advance the use of computing and visualization in VT research
- Centralize resource acquisition, maintenance, and support for research community
- Provide support to facilitate usage of resources and minimize barriers to entry
- Enable and participate in research collaborations between departments

# Personnel

- Associate VP for Research Computing: Terry Herdman
- Director, HPC: Terry Herdman
- Director, Visualization: Nicholas Polys
- Computational Scientists
  - John Burkhardt
  - Justin Krometis
  - James McClure
  - Srijith Rajamohan
  - Bob Settlage
  - Ahmed Ibrahim

# Personnel (Continued)

- System Administrators (UAS)
  - Matt Strickler
  - Josh Akers
- Solutions Architects
  - Brandon Sawyers
  - Chris Snapp
- Business Manager: Alana Romanella
- User Support GRAs: Mai Dahshan, Negin Forouzesh, Xiaolong Li.

# Personnel (Continued)

- System & Software Engineers
  - Nathan Liles
  - Open

# Computational Resources

- NewRiver
  - Cluster targeting data-Intensive problems
- DragonsTooth
  - jobs that don't require low latency interconnect i.e. pleasingly parallel
  - long running, e.g. 30 day wall time
- Blue Ridge
  - Large scale cluster equipped with Intel Xeon Phi Co-Processors
- Cascades - large general compute with GPU nodes
- Huckleberry - Machine Learning cluster
- Cloud based access (bring your user story)

# Compute Resources

System	Usage	Nodes	Node Description	Special Features
<a href="#"><u>BlueRidge</u></a>	Large-scale CPU, MIC	408	16 cores, 64 GB (2× Intel Sandy Bridge)	260 Intel Xeon Phi 4 K40 GPU 18 128GB nodes
<a href="#"><u>NewRiver</u></a>	Large-scale, Data Intensive	134	24 cores, 128 GB (2× Intel Haswell)	8 K80 GPGPU 16 “big data” nodes 24 512GB nodes 2 3TB nodes 40 P100 nodes
<a href="#"><u>Dragons Tooth</u></a>	Pleasingly parallel, long jobs	48	24 cores, 256 GB (2× Intel Haswell)	2TB SSD local storage 30 day walltime
<a href="#"><u>Huckleberry</u></a>	DeepLearning	14	Power 8, 256 GB (2x IBM “Minsky S822LC)	4xP100 / Node NVLink

# Computational Resources

Name	<u>NewRiver</u>	<u>DragonsTooth</u>	<u>BlueRidge</u>	<u>Huckleberry</u>
Key Features, Uses	Scalable CPU, Data Intensive	Pleasingly parallel, long running jobs	Scalable CPU or MIC	DeepLearning
Available	August 2015	August 2016	March 2013	Supt 2017
Theoretical Peak (TFlops/s)	152.6	.806	398.7	
Nodes	134	48	408	14
Cores	3,288	576	6,528	
Cores/Node	24	24	16	
Accelerators/Coproductors	8 Nvidia K80 GPU	N/A	260 Intel Xeon Phi 8 Nvidia K40 GPU	56 Nvidia P100 GPUs
Memory Size	34.4 TB	12.3 TB	27.3 TB	
Memory/Core	5.3 GB*	10.6 GB	4 GB*	
Memory/Node	128 GB*	256 GB	64 GB*	512 GB



# NewRiver

- 134 Nodes
- 3,288 Cores
- 34.4 TB Memory
- EDR InfiniBand (IB) Interconnect
- Data-intensive, large-memory, visualization, and GPGPU special-purpose hardware
  - 8 K80 GPGPU
  - 16 “big data” nodes
  - 24 512GB nodes
  - 2 3TB nodes
  - 40 dual P100 nodes

# NewRiver Nodes

Description	#	Hosts	CPU	Cores	Memory	Local Disk	Other Features
General	100	nr027-nr126	2 x E5-2680 v3 (Haswell)	24	128 GB, 2133 MHz	1.8 TB	
GPU	8	nr019-nr026	2 x E5-2680 v3 (Haswell)	24	512 GB	3.6 TB (2 x 1.8 TB)	NVIDIA K80 GPU
I/O	16	nr003-nr018	2 x E5-2680 v3 (Haswell)	24	512 GB	43.2 TB (24 x 1.8 TB)	2x 200 GB SSD
Large Memory	2	nr001-nr002	4 x E7-4890 v2 (Ivy Bridge)	60	3 TB	10.8 TB (6 x 1.8 TB)	
Interactive	8	newriver1-newriver8	2 x E5-2680 v3 (Haswell)	24	256 GB		NVIDIA K1200 GPU

# Storage Resources

Name	Intent	File System	Environment Variable	Per User Maximum	Data Lifespan	Available On
<u>Home</u>	Long-term storage of files	GPFS (NewRiver) NFS (Other)	\$HOME	500 GB (NewRiver) 100 GB (Other)	Unlimited	Login and Compute Nodes
Group	Shared Data Storage for Research Groups	GPFS (NewRiver)	\$GROUP	10 TB Free per faculty researcher	Unlimited	Login and Compute Nodes
<u>Work</u>	Fast I/O, Temporary storage	GPFS (NewRiver) Lustre (BlueRidge) GPFS (Other)	\$WORK	20 TB (NewRiver) 14 TB (Other)  3 million files	120 days	Login and Compute Nodes

# Storage Resources (continued)

Name	Intent	File System	Environment Variable	Per User Maximum	Data Lifespan	Available On
<a href="#"><u>Archive</u></a>	Long-term storage for infrequently -accessed files	CXFS	\$ARCHIVE	-	Unlimited	Login Nodes
<a href="#"><u>Local Scratch</u></a>	Local disk (hard drives)		\$TMPDIR	Size of node hard drive	Length of Job	Compute Nodes
<a href="#"><u>Memory (tmpfs)</u></a>	Very fast I/O	Memory (RAM)	\$TMPFS	Size of node memory	Length of Job	Compute Nodes

# Visualization Resources

- VisCube: 3D immersive environment with three 10' by 10' walls and a floor of 1920×1920 stereo projection screens
- DeepSix: Six tiled monitors with combined resolution of 7680×3200
- ROVR Stereo Wall
- AISB Stereo Wall

# Getting Started with ARC

- Review ARC's system specifications and choose the right system(s) for you
  - Specialty software
- Apply for an account online the Advanced Research Computing website
- When your account is ready, you will receive confirmation from ARC's system administrators

# Resources

- ARC Website: <http://www.arc.vt.edu>
- ARC Compute Resources & Documentation:  
<http://www.arc.vt.edu/hpc>
- New Users Guide:  
<http://www.arc.vt.edu/newusers>
- Frequently Asked Questions:  
<http://www.arc.vt.edu/faq>
- Linux Introduction:  
<http://www.arc.vt.edu/unix>

# Thank you

## Questions?



# Log In

- Log in via SSH
  - Mac/Linux have built-in client
  - Windows need to download client (e.g. PuTTY)

System	Login Address (xxx.arc.vt.edu)
NewRiver	newriver1 to newriver8
Dragonstooth	dragonstooth1
BlueRidge	blueridge1 or blueridge2
HokieSpeed	hokiespeed1 or hokiespeed2
HokieOne	hokieone

# Browser-based Access

- Browse to <http://newriver.arc.vt.edu>
- Xterm: Opens an SSH session with X11 forwarding (but faster)
- Other profiles: VisIt, ParaView, Matlab, Alinea
- Create your own!

# ALLOCATION SYSTEM

# Allocations

- “System unit” (roughly, core-hour) account that tracks system usage
- Applies only to NewRiver and BlueRidge

<http://www.arc.vt.edu/allocations>

# Allocation System: Goals

- Track projects that use ARC systems and document how resources are being used
- Ensure that computational resources are allocated appropriately based on needs
  - Research: Provide computational resources for your research lab
  - Instructional: System access for courses or other training events

# Allocation Eligibility

To qualify for an allocation, you must meet at least one of the following:

- Be a Ph.D. level researcher (post-docs qualify)
- Be an employee of Virginia Tech and the PI for research computing
- Be an employee of Virginia Tech and the co-PI for a research project led by non-VT PI

# Allocation Application Process

- Create a research project in ARC database
- Add grants and publications associated with project
- Create an allocation request using the web-based interface
- Allocation review may take several days
- Users may be added to run jobs

# Allocation Tiers

- Research allocations fall into three tiers:
- Less than 200,000 system units (SUs)
  - 200 word abstract
- 200,000 to 1 million SUs
  - 1-2 page justification
- More than 1 million SUs
  - 3-5 page justification



# Allocation Management

- Web-based:
  - User Dashboard -> Projects -> Allocations
  - Systems units allocated/remaining
  - Add/remove users
- Command line:
  - Allocation name and membership: `glsaccount`
  - Allocation size and amount remaining:  
`gbalance -h -a <name>`
  - Usage (by job): `gstatement -h -a <name>`

# USER ENVIRONMENT

# Consistent Environment

- Operating system (CentOS)
- Storage locations
- Scheduler
- Hierarchical module tree for system tools and applications

# Modules

- Modules are used to set the PATH and other environment variables
- Modules provide the environment for building and running applications
  - Multiple compiler vendors (Intel vs GCC) and versions
  - Multiple software stacks: MPI implementations and versions
  - Multiple applications and their versions
- An application is built with a certain compiler and a certain software stack (MPI, CUDA)
  - Modules for software stack, compiler, applications
- User loads the modules associated with an application, compiler, or software stack
  - modules can be loaded in job scripts

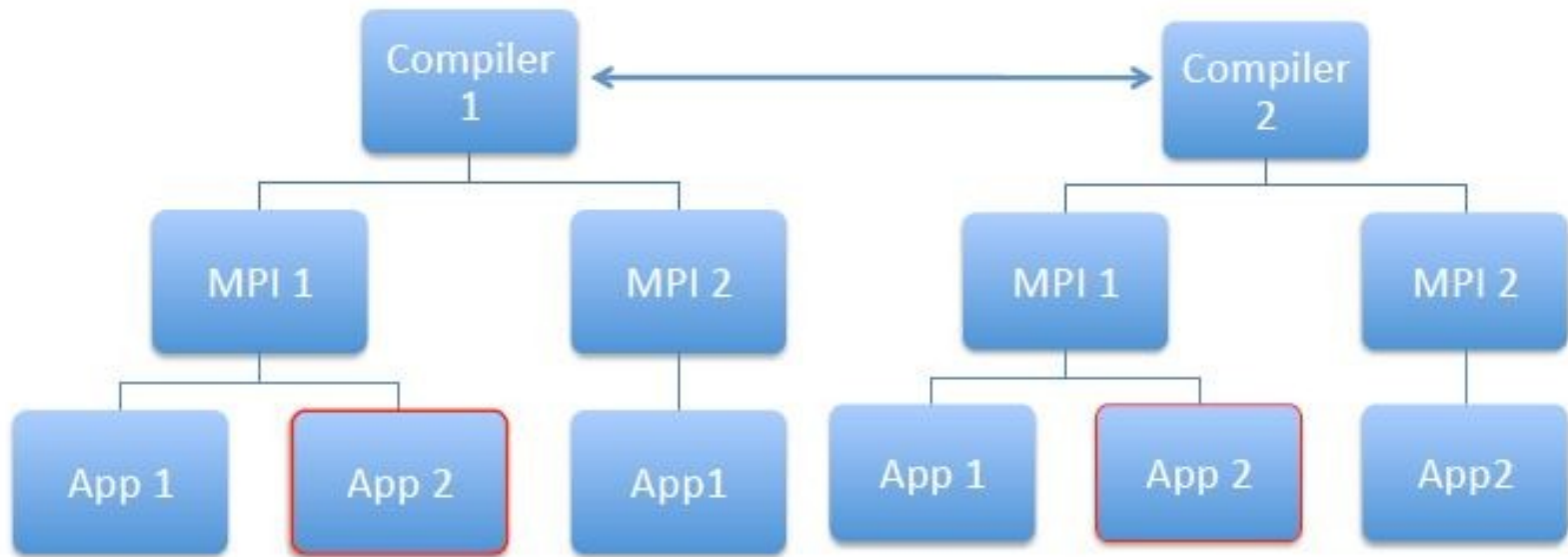
# Module commands

Command	Result
<code>module</code>	List options
<code>module list</code>	List loaded modules
<code>module avail</code>	List available modules
<code>module load &lt;module&gt;</code>	Add a module
<code>module unload &lt;module&gt;</code>	Remove a module
<code>module swap &lt;mod1&gt; &lt;mod2&gt;</code>	Swap two modules
<code>module help &lt;module&gt;</code>	Module environment
<code>module show &lt;module&gt;</code>	Module description
<code>module spider &lt;module&gt;</code>	Search modules
<code>module reset</code>	Reset to default
<code>module purge</code>	Unload all modules

# Modules

- Available modules depend on:
  - The compiler (eg. Intel, gcc) and
  - The MPI stack selected
- Defaults:
  - BlueRidge: Intel + mvapich2
  - NewRiver, DragonsTooth: none

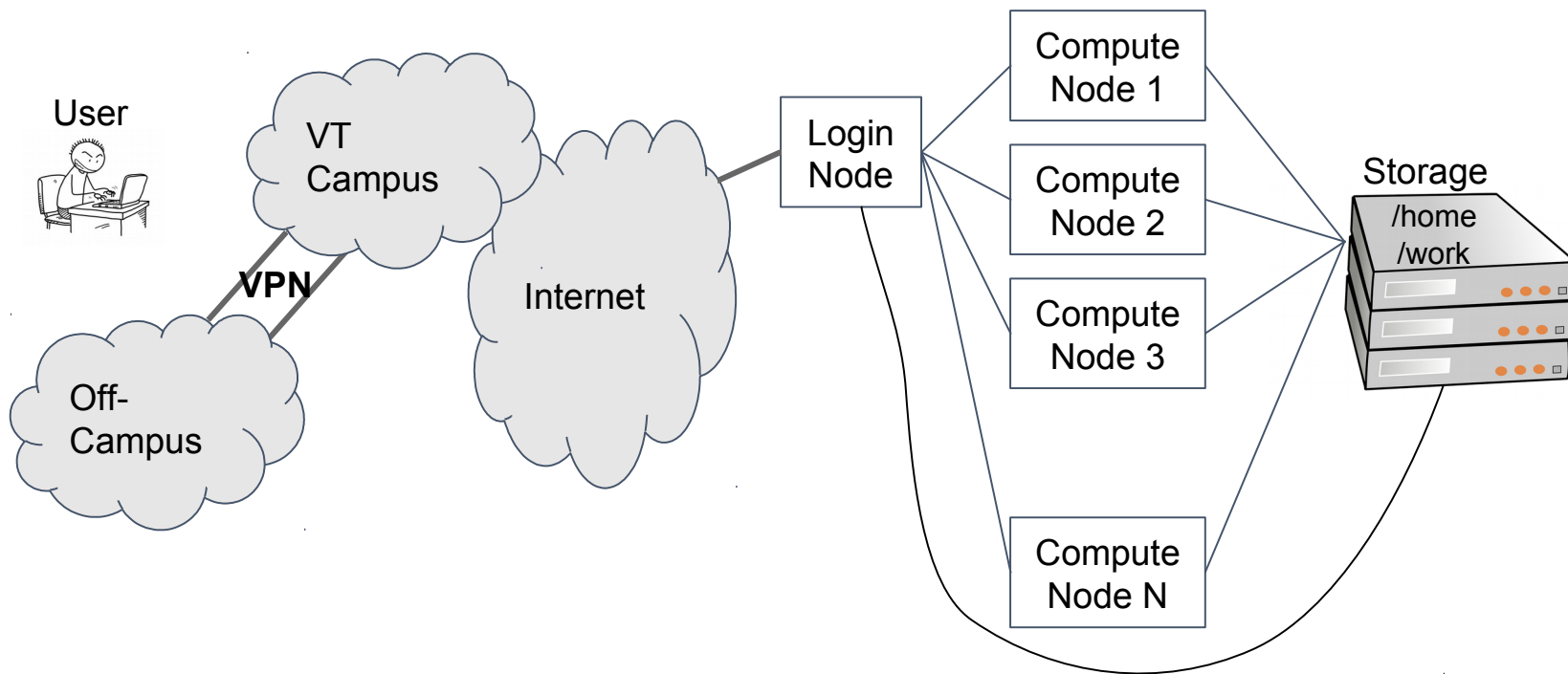
# Hierarchical Module Structure



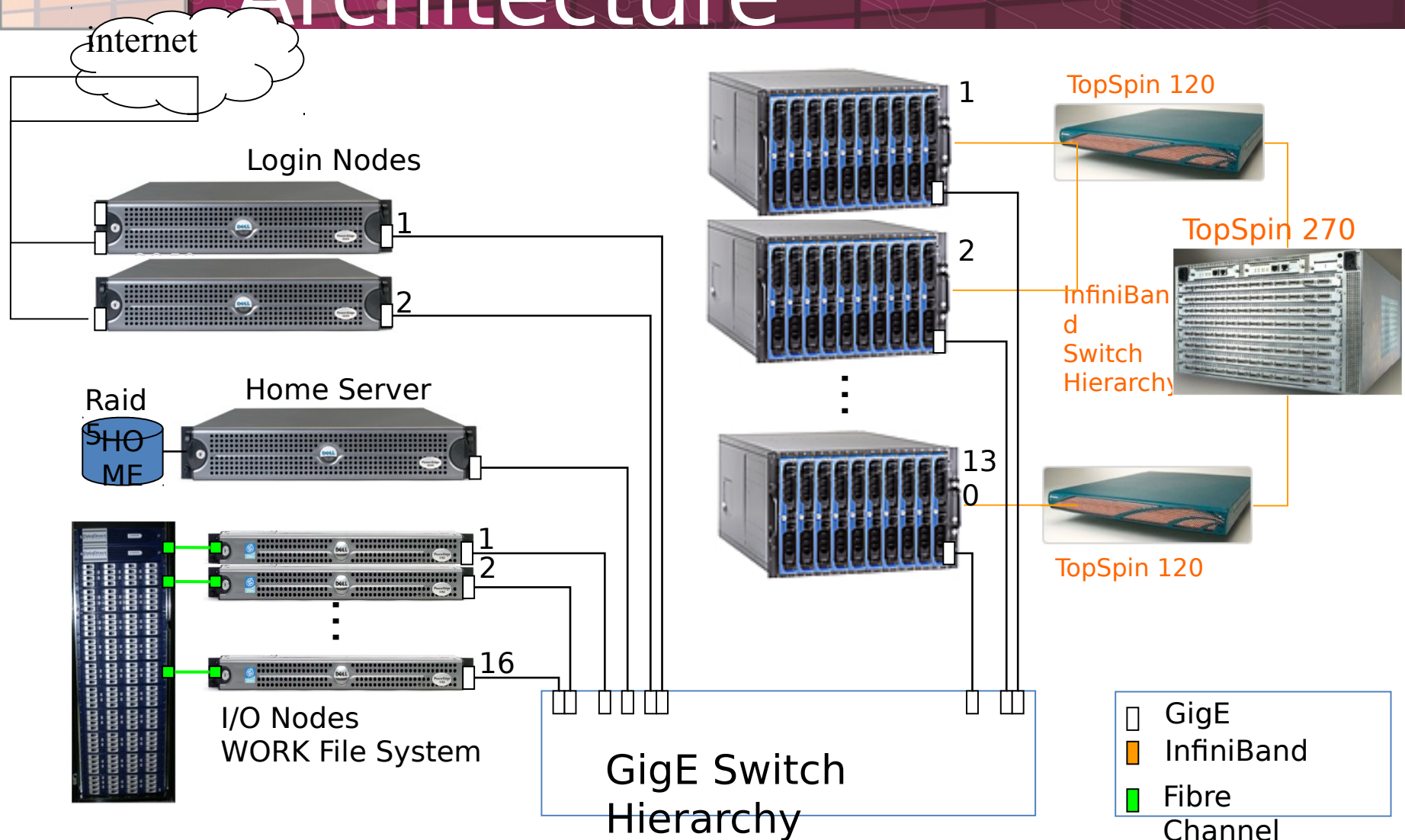
# JOB SUBMISSION & MONITORING



# Cluster System overview

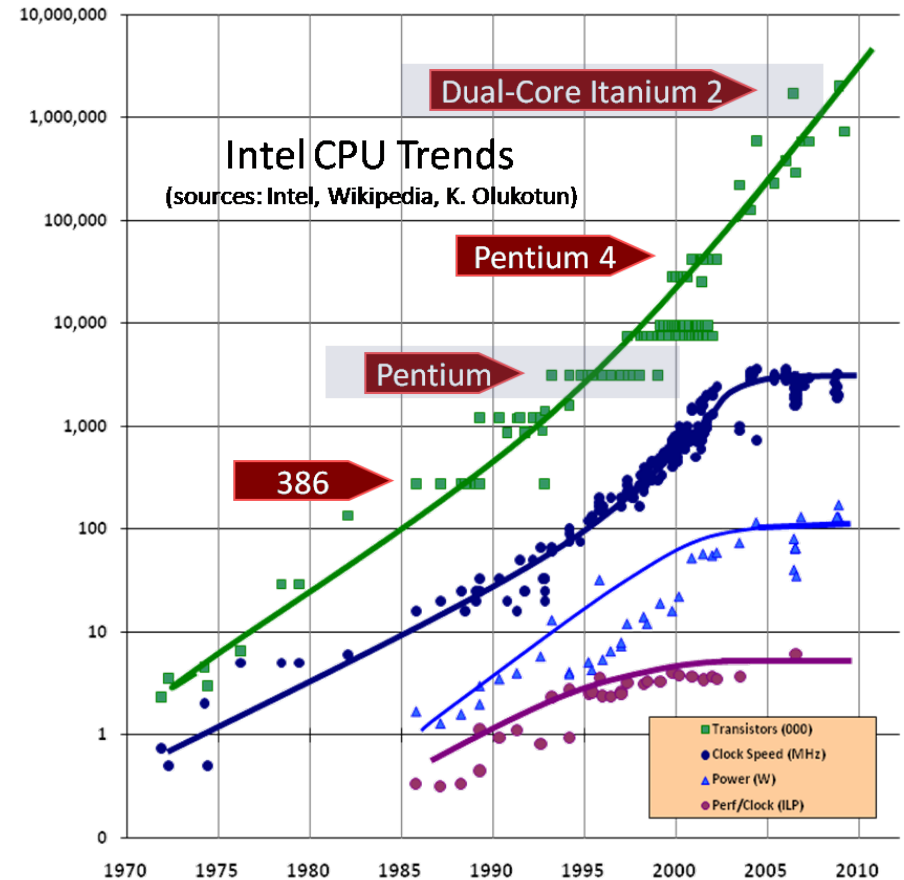


# Cluster System Architecture



# Parallelism is the New Moore's Law

- Power and energy efficiency impose a key constraint on design of micro-architectures
- Clock speeds have plateaued
- Hardware parallelism is increasing rapidly to make up the



# Essential Components of HPC

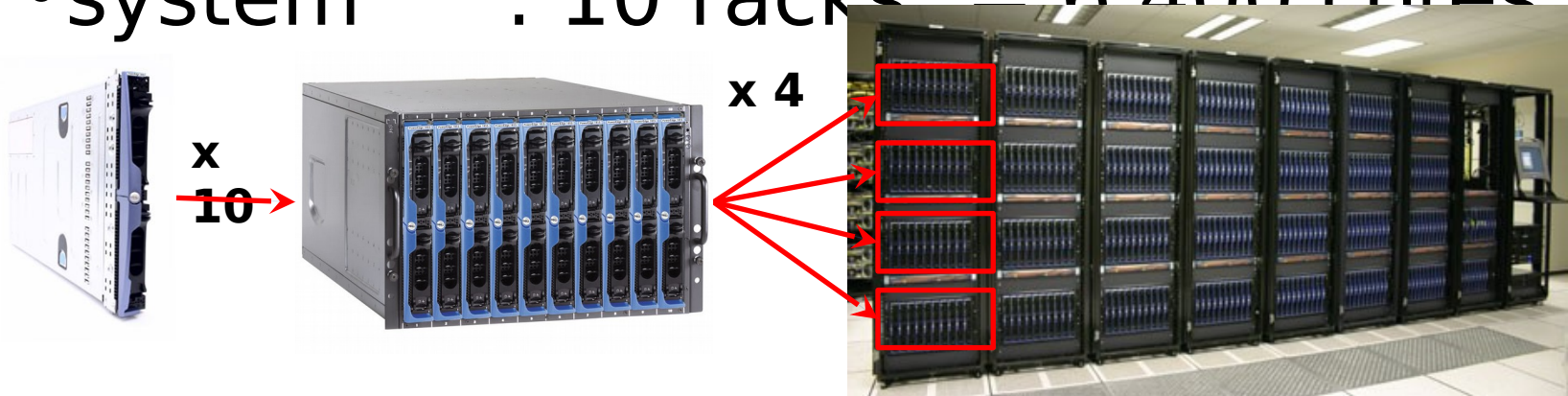
- Supercomputing resources
- Storage
- Visualization
- Data management
- Network infrastructure
- Support

# Terminology

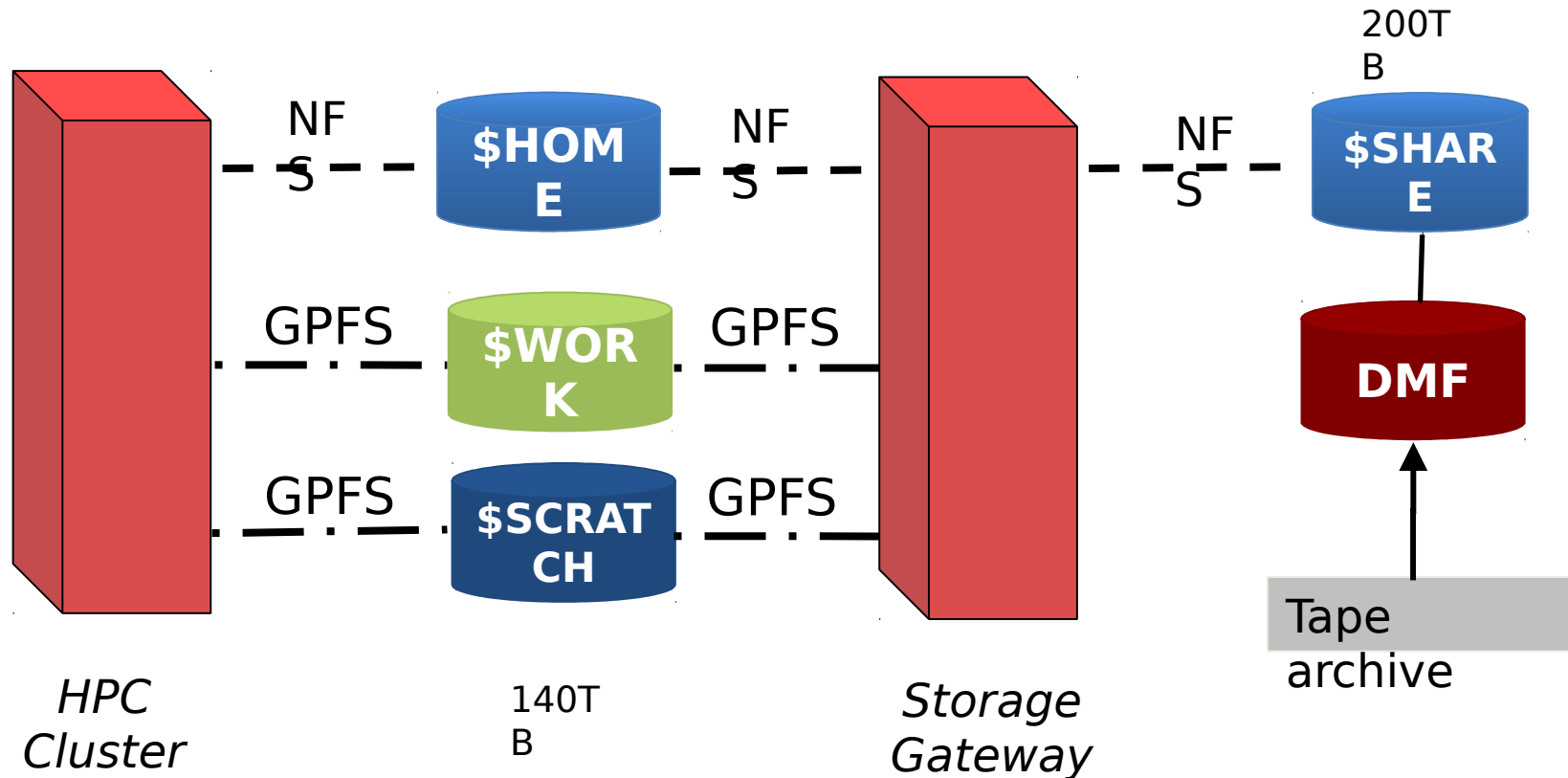
- Core: A computational “unit”
- Socket: A single CPU (“processor”). Includes roughly 4-15 cores.
- Node: A single “computer”. Includes roughly 2-8 sockets.
- Cluster: A single “supercomputer” consisting of many nodes.
- GPU: Graphics processing unit. Attached to some nodes. General purpose GPUs (GPGPUs) can be used to speed up certain kinds of codes.
- Xeon Phi: Intel’s product name for its GPU competitor. Also called “MIC”.

# Blade : Rack : System

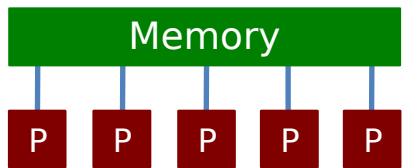
- 1 node : 2 x 8 cores = 16 cores
- 1 chassis : 10 nodes = 160 cores
- 1 rack (frame) : 4 chassis = 640 cores
- system : 10 racks = 6 400 cores



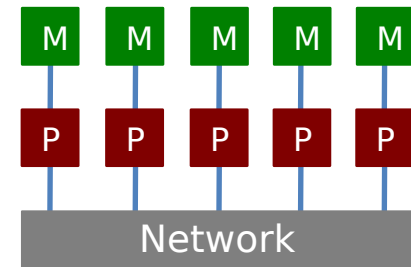
# HPC Storage



# Shared vs. Distributed memory



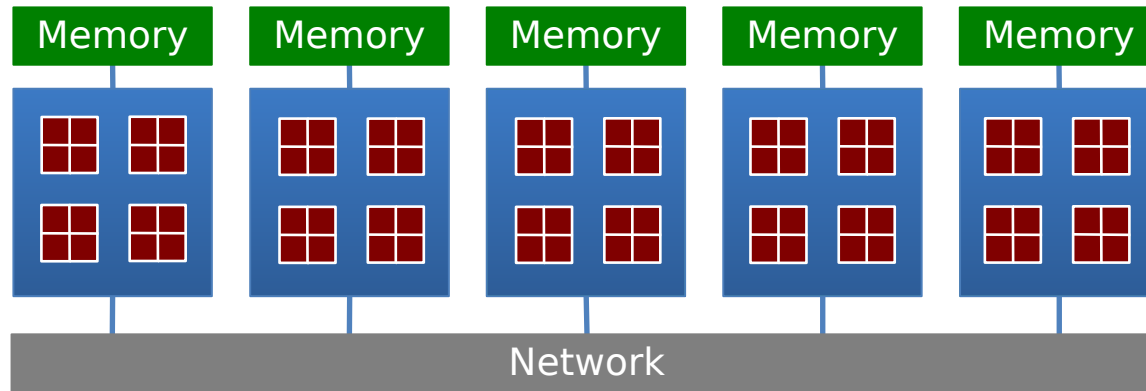
- All processors have access to a pool of shared memory
- Access times vary from CPU to CPU in NUMA systems
- Example: SGI UV, CPUs on same node



- Memory is local to each processor
- Data exchange by message passing over a network
- Example: Clusters with single-socket blades

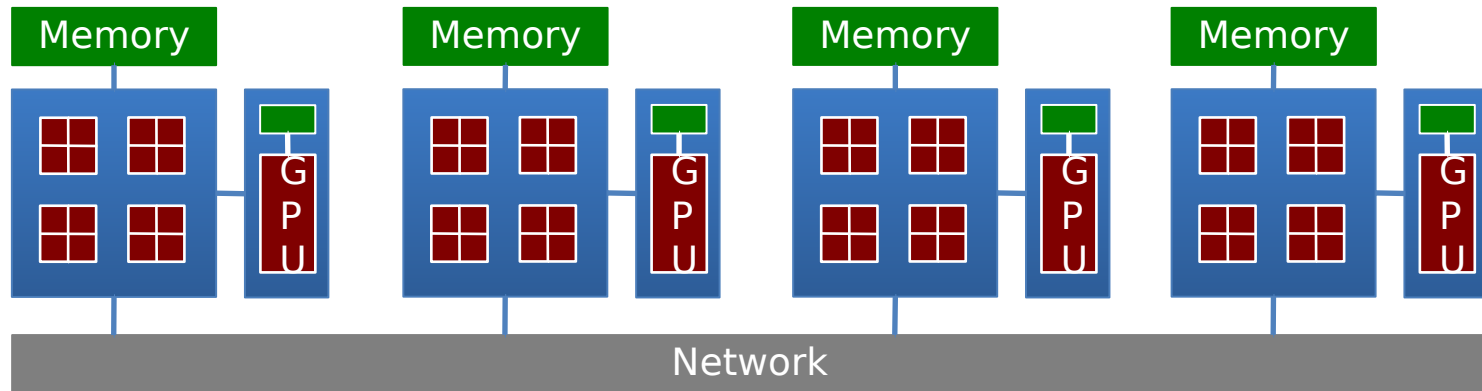


# Multi-core systems



- Current processors place multiple processor cores on a die
- Communication details are increasingly complex
  - Cache access
  - Main memory access
  - Quick Path / Hyper Transport socket connections
  - Node to node connection via network

# Accelerator-based Systems



- Calculations made in both CPUs and GPUs
- No longer limited to single precision calculations
- Load balancing critical for performance
- Requires specific libraries and compilers (CUDA, OpenCL)
- Co-processor from Intel: MIC (Many Integrated Core)

# Submitting a Job

- Submission via a shell script
  - Job description: Resources required, run time, allocation
  - Modules & dependencies
  - Execution statements
- Submit job script: `qsub <job_script>`
- Interactive options:
  - Interactive job: `qsub -I ...`
  - Interactive job with X11 forwarding: `qsub -I -X ...`

# Job Monitoring

- Determine job status, and if pending when it will run

Command	Meaning
<code>checkjob -v JOBID</code>	Get the status and resources of a job
<code>showq</code>	See what jobs are running and cluster utilization
<code>showstart JOBID</code>	Get expected job start time
<code>qdel JOBID</code>	Delete a job

# Job Execution

- Order of job execution depends on a variety of parameters:
  - Submission Time
  - Queue Priority
  - Backfill Opportunities
  - Fairshare Priority
  - Advanced Reservations
  - Number of Actively Scheduled Jobs per User

# Examples: ARC Website

- See the Examples section of each system page for sample submission scripts and step-by-step examples:
  - <http://www.arc.vt.edu/newriver>
  - <http://www.arc.vt.edu/dragonstooth>
  - <http://www.arc.vt.edu/blueridge>
  - <http://www.arc.vt.edu/huckleberry-user-guide>

# A Step-by-Step Example

# Getting Started

- Find your training account (hpcXX)
- Log into NewRiver
  - Mac: `ssh`  
`hpcXX@cascades1.arc.vt.edu`
  - Windows: Use PuTTY
    - Host Name: `cascades1.arc.vt.edu`



# Example: Running MPI Quad

- Or copy from training directory:

```
cp /home/TRAINING/newriver/* ./
```

- Build the code

# Compile the Code

- Intel compiler is already loaded  
`module list`
- Compile command (executable is `mpiqd`)  
`mpicc -o mpiqd mpi_quad.c`
- To use GCC instead, swap it out:  
`module swap intel gcc`

# Prepare Submission Script

- Copy sample script:

```
cp /home/TRAINING/ARC_Intro/br.qsub .
```

- Edit sample script:

- Walltime

- Resource request (nodes/ppn)

- Module commands (add Intel & mvapich2)

- Command to run your job

- Save it (e.g., mpiqd.qsub)

# Submission Script (Typical)

```
#!/bin/bash
#PBS -l walltime=00:10:00
#PBS -l nodes=2:ppn=16
#PBS -q normal_q
#PBS -W group_list=newriver
#PBS -A AllocationName          <--Only for NewRiver/BlueRidge

module load intel mvapich2

cd $PBS_O_WORKDIR
echo "MPI Quadrature!"
mpirun -np $PBS_NP ./mpiqd

exit;
```

# Submission Script (Today)

```
#!/bin/bash
#PBS -l walltime=00:10:00
#PBS -l nodes=2:ppn=8
#PBS -q normal_q
#PBS -W group_list=training
#PBS -A training
#PBS -l advres=NLI_ARC_Intro.26

module load intel mvapich2

cd $PBS_O_WORKDIR
echo "MPI Quadrature!"
mpirun -np $PBS_NP ./mpiqd

exit;
```

# Submit the job

- Copy the files to \$WORK:

```
cp mpiqd $WORK
```

```
cp mpiqd.qsub $WORK
```

- Navigate to \$WORK

```
cd $WORK
```

- Submit the job:

```
qsub mpiqd.qsub
```

- Scheduler returns job number:

```
25770.master.cluster
```

# Wait for job to complete

- **Check job status:**

```
checkjob -v 25770
```

```
showq -u hpcXX
```

- **When complete:**

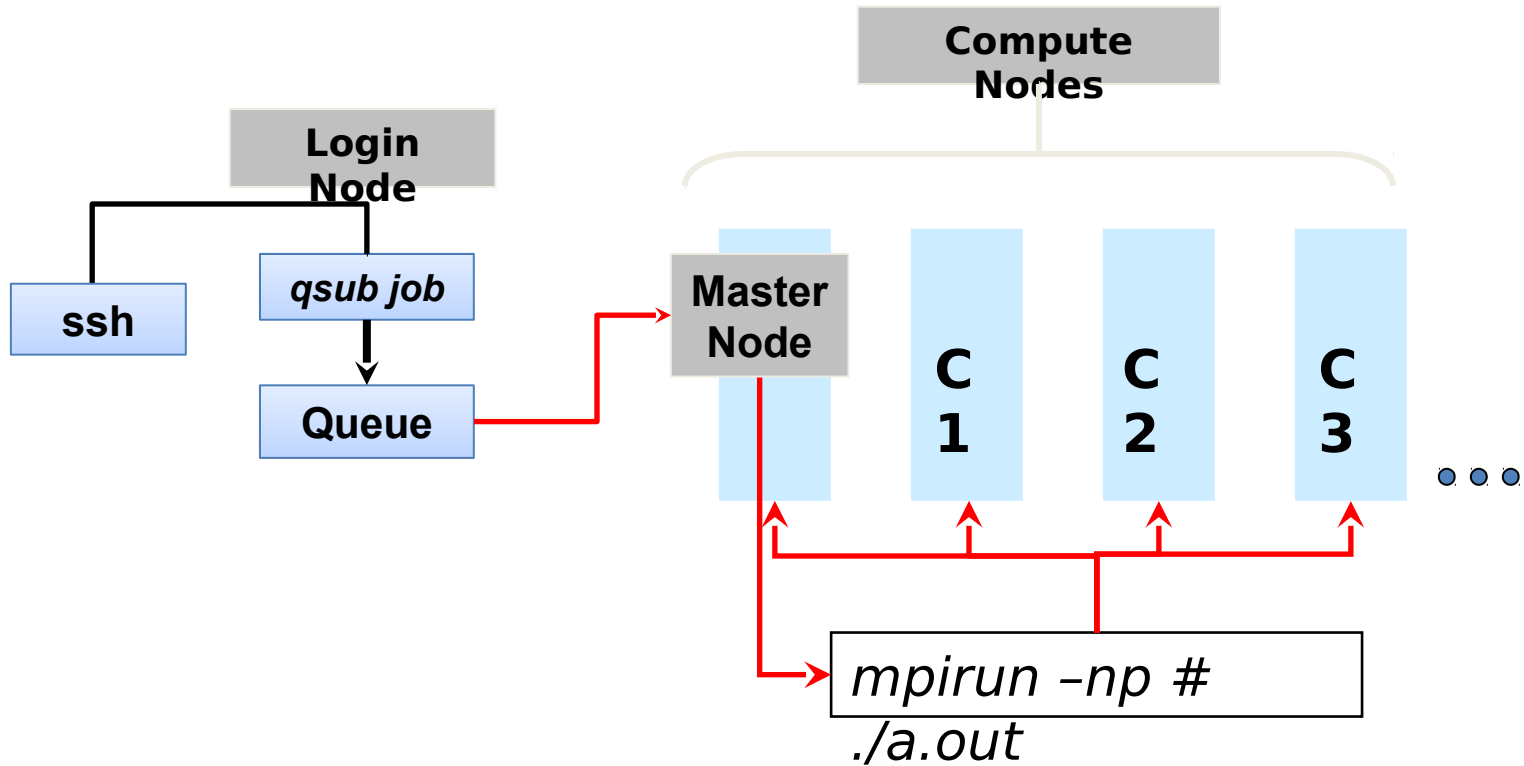
```
-Job output: mpiqd.qsub.o25770
```

```
-Errors: mpiqd.qsub.e25770
```

- **Copy results back to \$HOME:**

```
cp mpiqd.qsub.o25770 $HOME
```

# What is Job Submission?





# Resources

- ARC Website: <http://www.arc.vt.edu>
- Compute Resources & Documentation: <http://www.arc.vt.edu/hpc>
- Storage Documentation: <http://www.arc.vt.edu/storage>
- New Users Guide: <http://www.arc.vt.edu/newusers>
- Frequently Asked Questions: <http://www.arc.vt.edu/faq>
- Linux Introduction: <http://www.arc.vt.edu/unix>
- Module Tutorial: <http://www.arc.vt.edu/modules>
- Scheduler Tutorial: <http://www.arc.vt.edu/scheduler>

# Questions

## **Presenter Name, Email**

Computational Scientist

Advanced Research Computing

Division of IT