Research Computing Resources at UVA

SOMRC Summer Bootcamp
08/07/2018

Byoung-Do (BD) Kim, Ph.D.

SCHOOL of MEDICINE
Research Computing

Outline

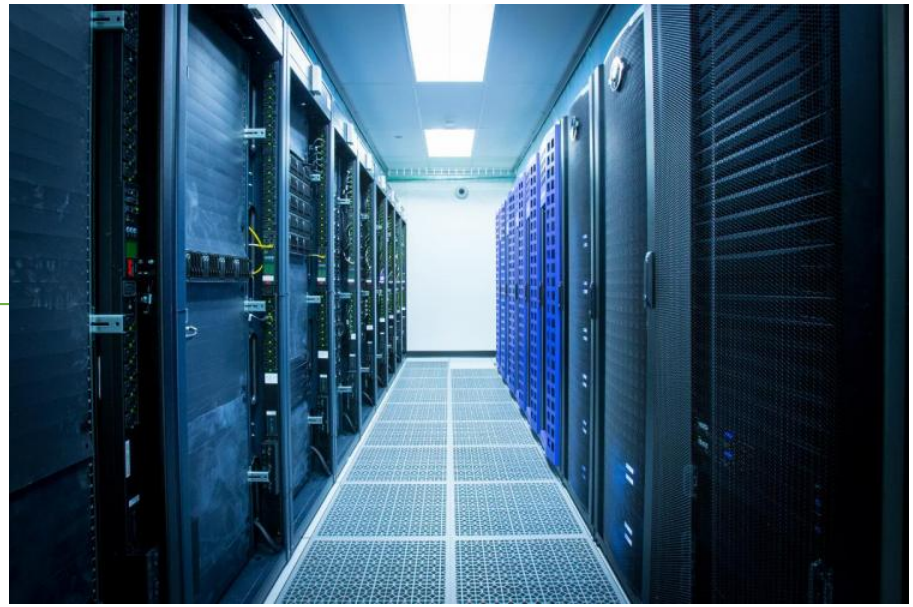
- Intro to SOMRC
- Element of High-Performance Computing
- Overview of Advanced Computing Infrastructure at UVA
- Cloud Solutions using AWS
- Intro to Rivanna HPC System
- Storage Solutions and Data Transfer

SOMRC Services

- User Support
  - SOMRC/CADRE website
    - http://somrc.virginia.edu/
    - http://cadre.virginia.edu/
  - Ticketing System: email to somrc-support@virginia.edu
  - Prefer face-2-face? - Weekly office hour (Wednesday 10am-12pm @HSL)
  - Got a question?- FAQ/KB platform
  - User training & workshops - CADRE Academy
- Research Project Support
  - Cloud solutions
  - High performance computing
  - Data analysis
  - Interdisciplinary research collaborations

Supercomputer?

ssh
client



What is High Performance Computing?

- In our context, it refers to hardware and software tools dedicated to computationally intensive tasks

- Distinction between HPC center (throughput focused) and Data center (data focused) is becoming fuzzy

- Chronological shift of HPC :
  - Supercomputing (special architecture, proprietary)
  - Cluster computing (commodity hardware, Linux OS and open source software)
  - Grid/Cloud computing (from Grid to Cloud)

Performance Metrics

- Number Crunching
  - Measuring FLOPS (Floating Point Operations per Second) by running LINPACK: Top500
- Data Crunching
  - Measuring TEPS (Traversed Edges per Second) by running kernels from graph algorithm: Graph500
- Energy Efficiency
  - Measuring GFLOPS/W: Green500
- Exascale? (goal: 1 Exaflops/20MW by 2020)
  - Top 3 systems are in Japan (Zetta-Scaler based architecture)
  - #1 system runs 17GFlops/W
  - 50GFLOPS/W in 10 years?

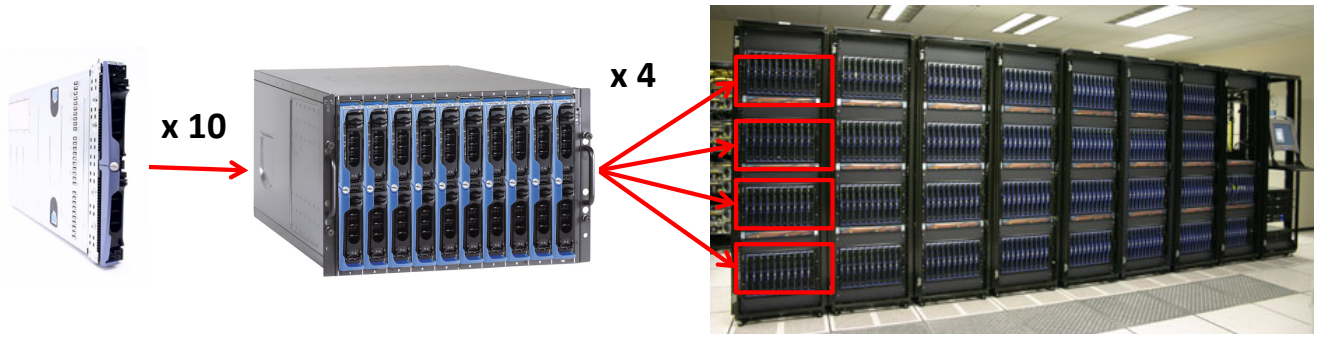
Terminology

- Core – an individual processor on a computer

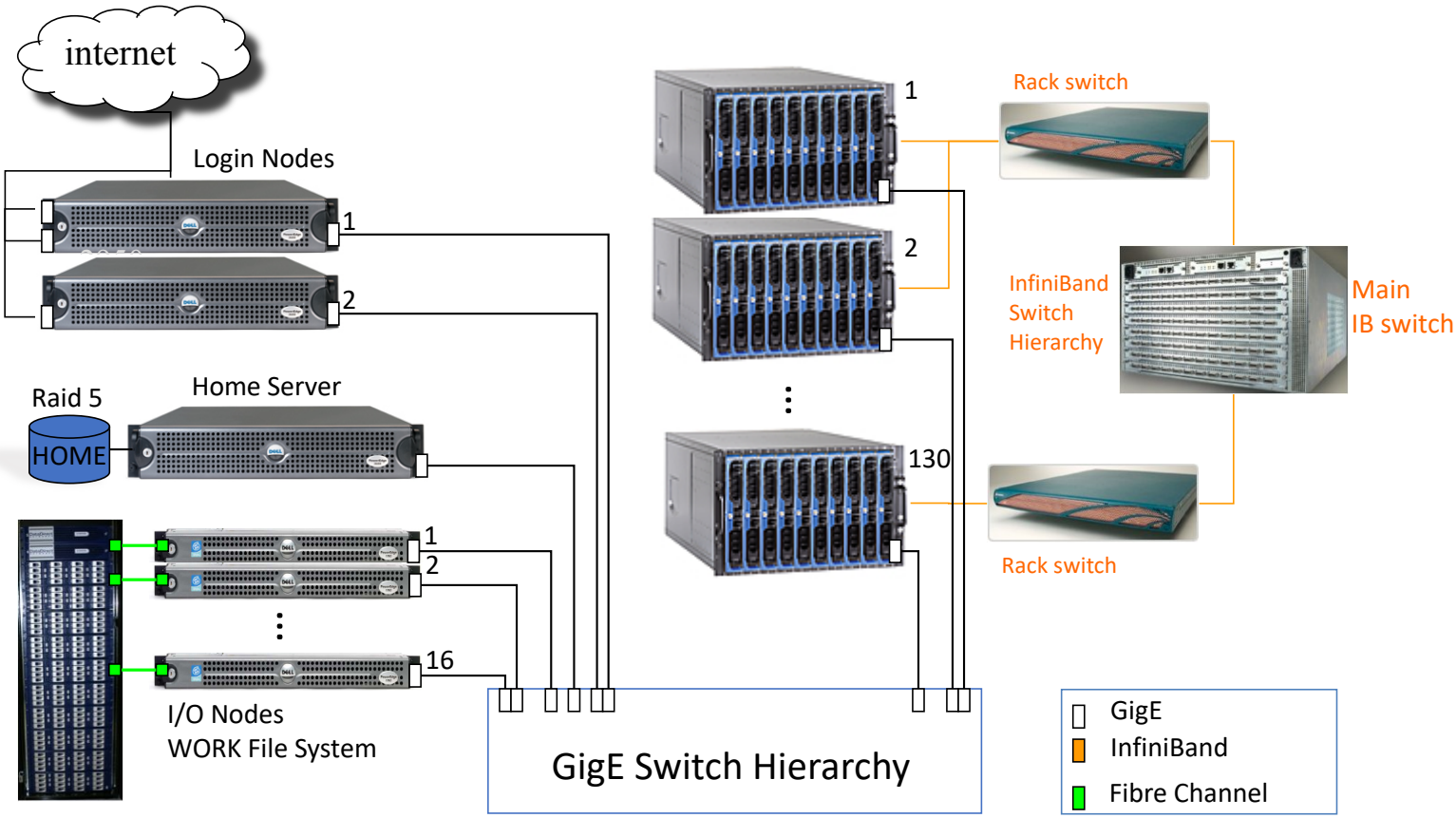
- Node
  - Basic building block of a cluster
  - Usually a specialized computer
  - Two types of nodes:
    - Head Node – computer used for logging on and submitting jobs
    - Compute Node -- computer that does most of the work
- Rack

Blade : Rack : System

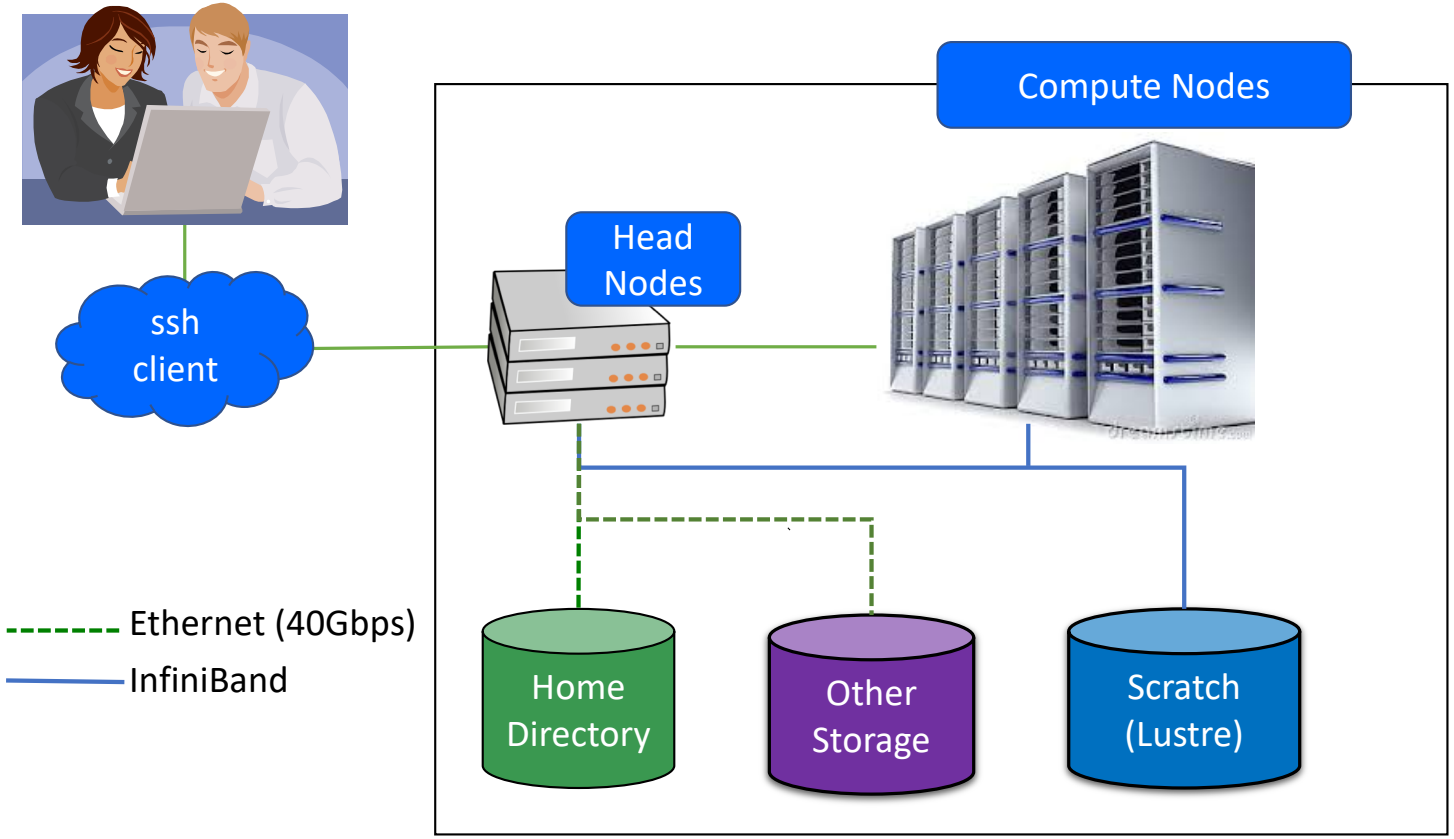
- 1 node (blade) : 10 x 2 cores = 20 cores
- 1 chassis : 10 nodes = 200 cores
- 1 rack (frame) : 4 x 10 nodes = 800 cores
- 10 racks : 8000 cores



General Cluster System Architecture



HPC Cluster Architecture



Hardware in Parallel Computing

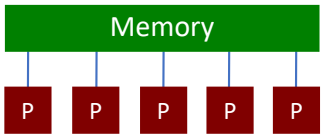
Memory access

- Shared memory
  - SGI Altix, UV systems
  - IBM Power series nodes
- Distributed memory
  - Uniprocessor clusters
- Hybrid
  - Multi-core clusters

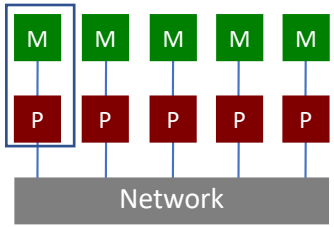
Processor type

- Single core CPU
  - Intel Xeon (Prestonia, Wallatin)
  - AMD Opteron (Sledgehammer, Venus)
- Multi-core CPU
  - Intel Xeon
  - AMD Opteron, Epyc
  - IBM POWER (7, 8, 9…)
- GPU based
  - Nvidia
  - AMD

Shared and distributed memory

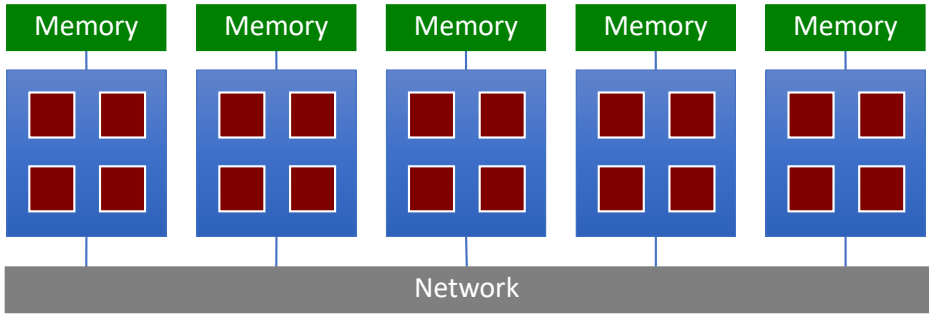


- All processors have access to a pool of shared memory
- Access times vary from CPU to CPU in NUMA systems
- Example: SGI Altix & UV, IBM P6 nodes



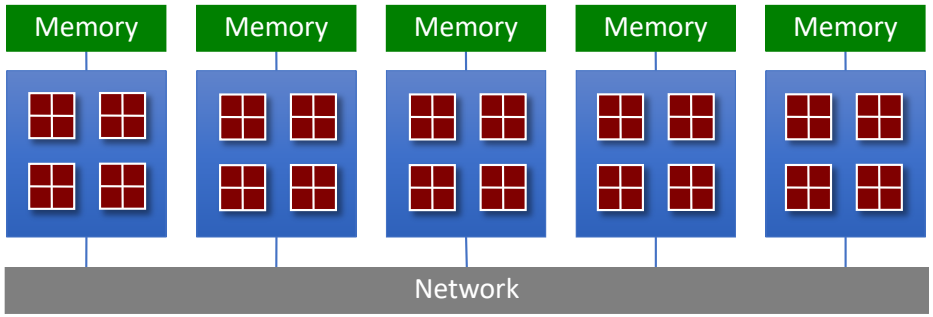
- Memory is local to each processor
- Data exchange by message passing over a network
- Example: Clusters with single-socket blades

Hybrid systems



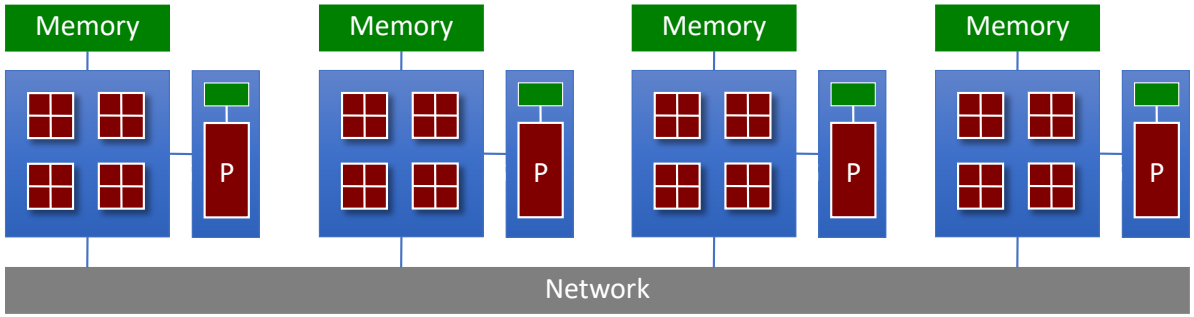
- A limited number of processors N have access to a common pool of shared memory
- To use more than N processors requires data exchange over a network
- Example: Cluster with multi-socket blades

Multi-core systems



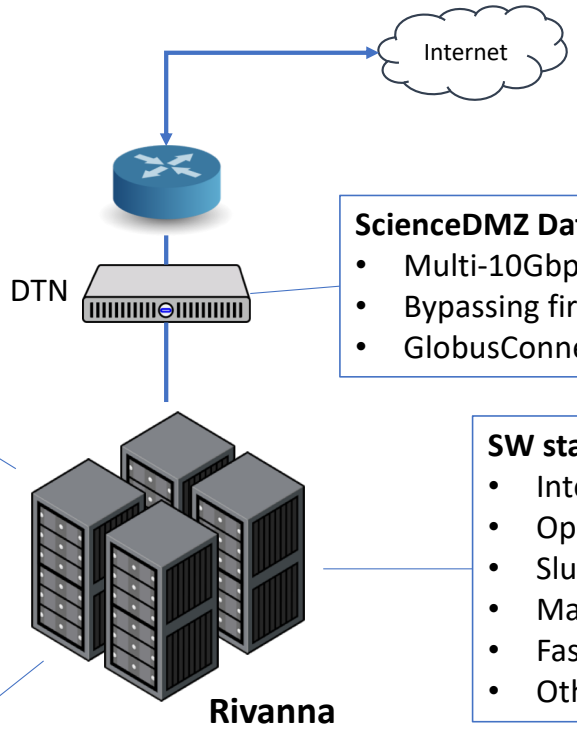
- Extension of hybrid model
- Communication details increasingly complex
  - Cache access
  - Main memory access
  - Quick Path / Hyper Transport socket connections
  - Node to node connection via network

GPGPU Systems



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

Rivanna HPC System



System Spec:
- 4 login nodes
- 265 Compute nodes: ~6000 cores
- 14 GPU nodes
  - 32 K-80's (8x4)
  - 16 P-100's (4x4)
- 8 KNL nodes
  - 64 core/socket
  - 512 total cores
- 5 LargeMem node (1TB/node)
- 56Gbps InfiniBand interconnection

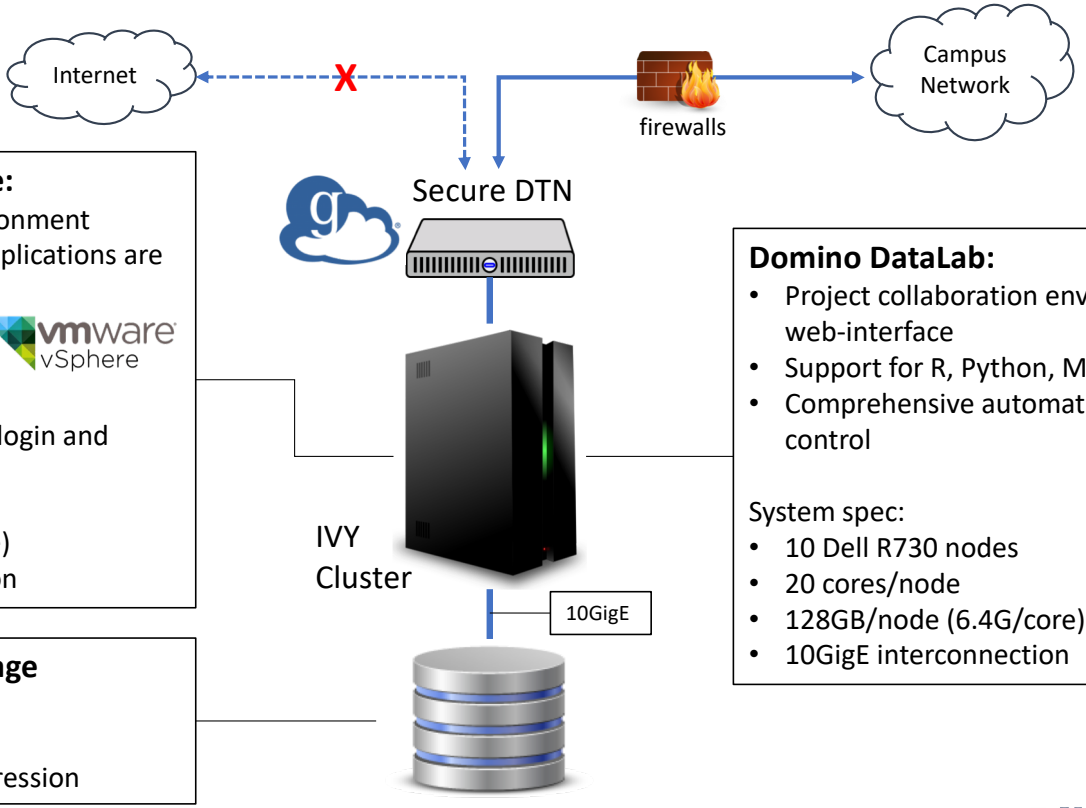
ScienceDMZ Data Transfer Node
- Multi-10Gbps connection
- Bypassing firewalls, high-speed transfer for large files
- GlobusConnect service

SW stack:
- Intel/GCC/PGI compiler
- OpenMPI/Mvapich MPI stacks
- Slurm batch scheduler
- Math libraries (Lapack, boost, etc.)
- FastX GUI interface
- Other applications

File System:
- 50GB /home directory
- 1.4PB /scratch space
- 1.8PB /project

IVY Secure Cluster System

- IVY Secure Private Cloud Platform is designed for the research with highly sensitive data
- VMWare's vCloud + Domino DataLab + Spark (future) configuration



VMWare vCloud suite:
- Virtual Machine environment
- Various biomedical applications are available

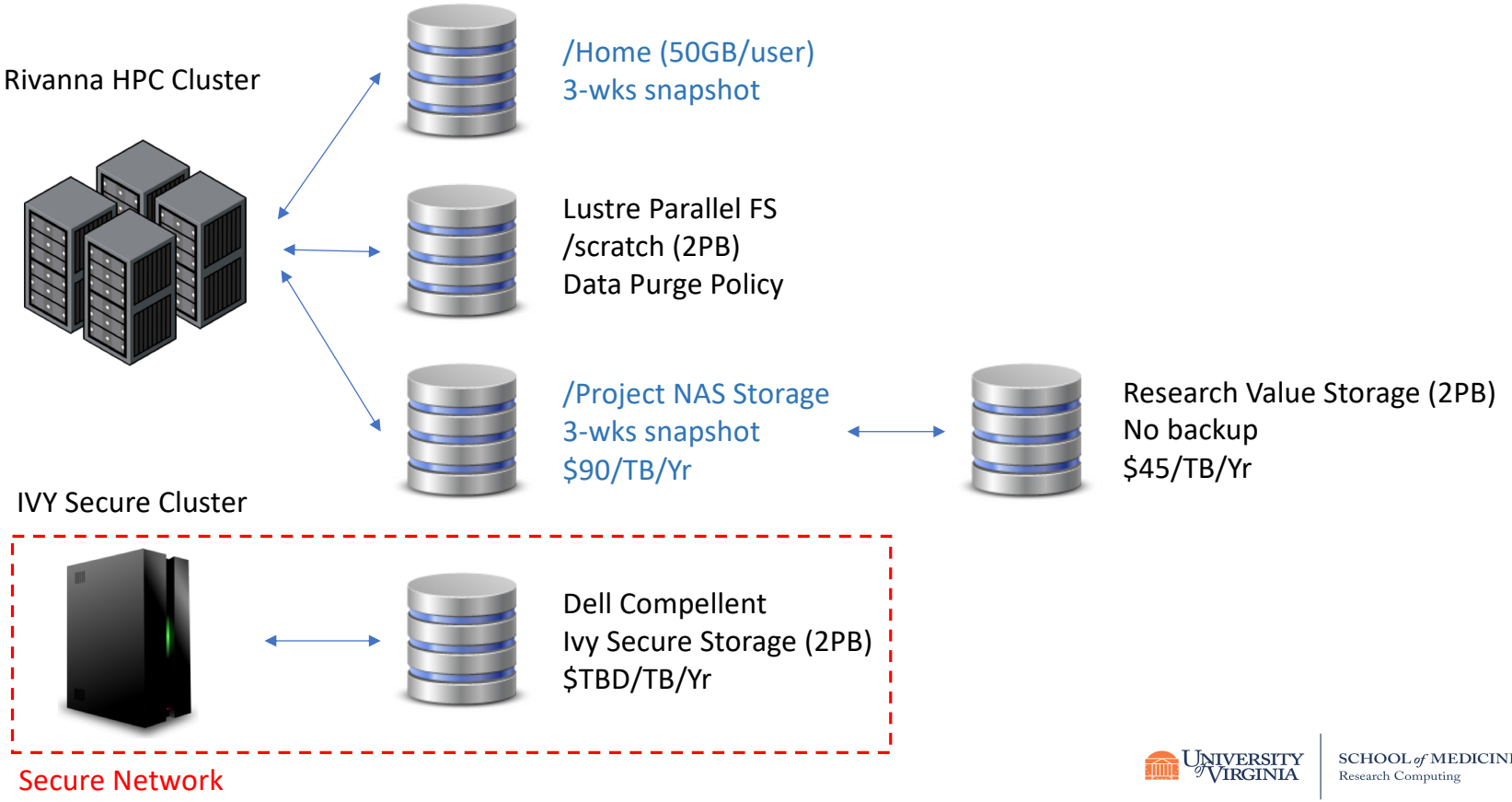
System spec:
- 32 Dell M630 nodes
- (4 nodes are used for login and management)
- 36 cores/node
- 256GB/node (7G/core)
- 10GigE interconnection

Dell Compellent Storage
- 2.1PB total capacity
- 10GigE interface
- Automated data progression

Domino DataLab:
- Project collaboration environment via web-interface
- Support for R, Python, Matlab, etc.
- Comprehensive automatic version control

System spec:
- 10 Dell R730 nodes
- 20 cores/node
- 128GB/node (6.4G/core)
- 10GigE interconnection

UVA Research Computing Storages



Cloud Support

Strategies for Supporting Research

- Complement Rivanna & Grounds with cloud-based services
- Facilitate data sharing, internally and externally
- Create a menu of repeatable solutions
- Develop multi-tenant suite of advanced services

Cloud Infrastructure

Unique Requirements – Provides services typically unavailable on grounds:

- Operating Systems / Hardware / Clustering
- Databases
- Object Storage for external sharing

Unique Lifespans – Ideal for short-lived or fixed-period projects:

- Immediately available, on-demand
- Removes capital expenses
- Can be turned off/on for periodic bursts of research

Removes "Heavy Lifting"

- No need to install and configure services
- No need to maintain and update

Flexible & Resizable

Cloud Support – Solutions

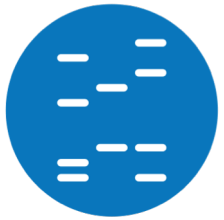
Web Content & Data Sharing

Pipelines

StarCluster

Application Servers & Databases

Data Lakes



Bionformatics Images

Introduction to Rivanna

Warm-Up

- We need to know about
  - Allocations & accounts;
  - Connections to the cluster;
  - Cluster Environment;
  - Modules & queues;
  - SLURM & job submissions

- The remaining slides will cover the basics of these topics.

Allocations

- Rivanna is allocated:
  - At the most basic level, an allocation refers to a chunk of CPU time that you receive in which to run your jobs
- Allocations are measured in service units (SUs), where

  1 SU = 1 core-hour
- However, if all of the memory on a node is requested, the service unit will be doubled (for now)
- Each account will be given a specific number of service units.

Allocations are Finite

- Your group will share an allocation assigned to a PI.
- Do not be the one who uses it up before the end of the semester!
- To the best of your ability:
  - Use good programming practices;
  - Optimize your code;
  - Test your code with small sample sizes until you have ironed out most of the bugs.

Connecting to the Cluster

- You will need an ssh (secure shell) client on your computer.
- For non-graphical access
  - On a Mac or Linux system, use ssh (Terminal application on Macs)

    `ssh -Y Your_ID@rivanna.hpc.virginia.edu`

  - On a Windows system, use `MobaXterm` or `PuTTY`
    - To install MobaXterm use the URL: **`http://mobaxterm.mobatek.net/`**
    - To get a copy of PuTTY, use the URL:

    http://www.chiark.greenend.org.uk/~sgtatham/putty/

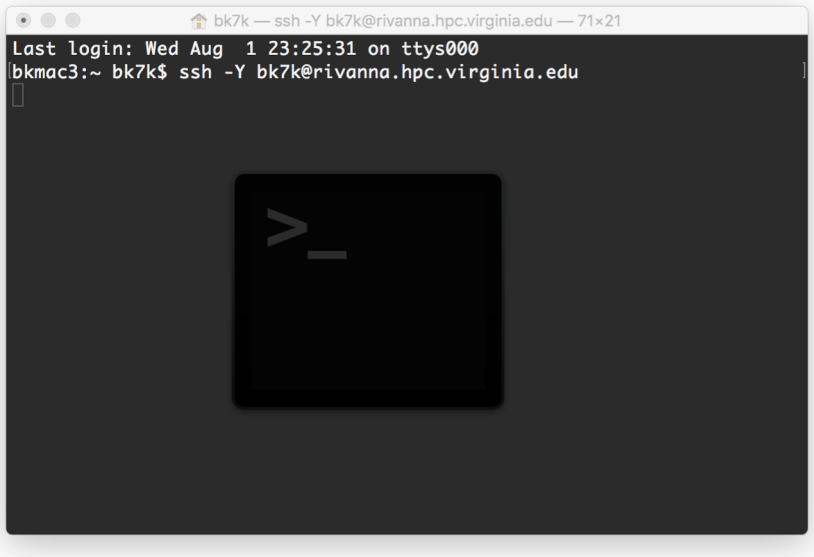
Connecting to the Cluster

- Hostname for the Interactive frontends:
  - Rivanna1.hpc.virginia.edu
  - Rivanna2.hpc.virginia.edu
  - Rivanna3.hpc.virginia.edu

  - rivanna.hpc.virginia.edu (does load-balancing among the three front-ends)

Connecting to the Cluster with ssh

- If you are on a Mac or Linux machine your can connect with ssh.
- Bring up a terminal window and type: ssh –Y *userID*@rivanna.hpc.virginia.edu
- When it prompts you for a password, use your Eservices password.

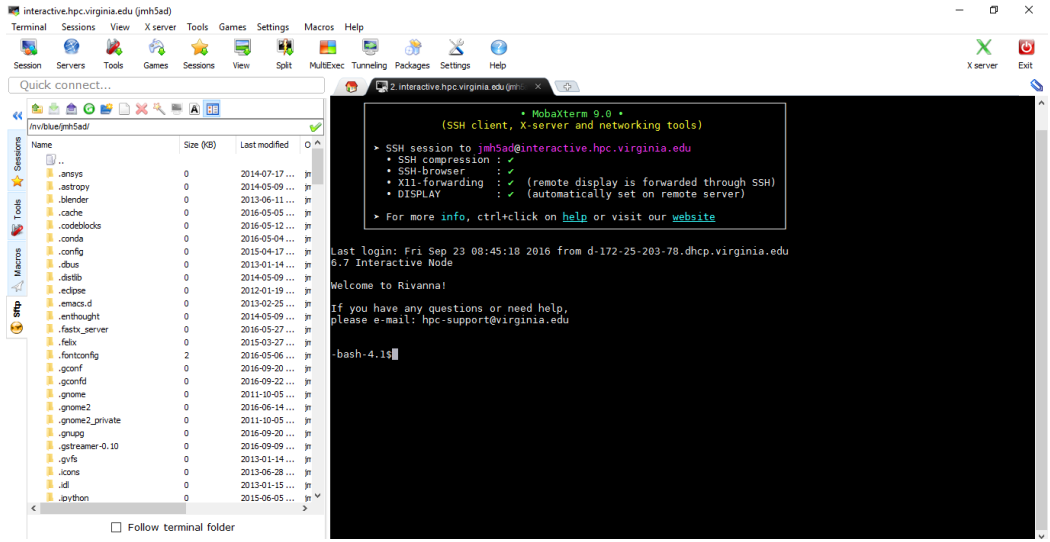


Connecting to the Cluster with MobaXterm

- Start a new SSH session.
- Put in rivanna.hpc.virginia.edu for “Remote Host”. Click the box for “Specify username” and type in your user id. Click on the OK button.
- After typing your password, you should see a split screen.

The right is a terminal window.

The left is a list of files in your remote folder that you can click, drag, and drop onto your desktop.



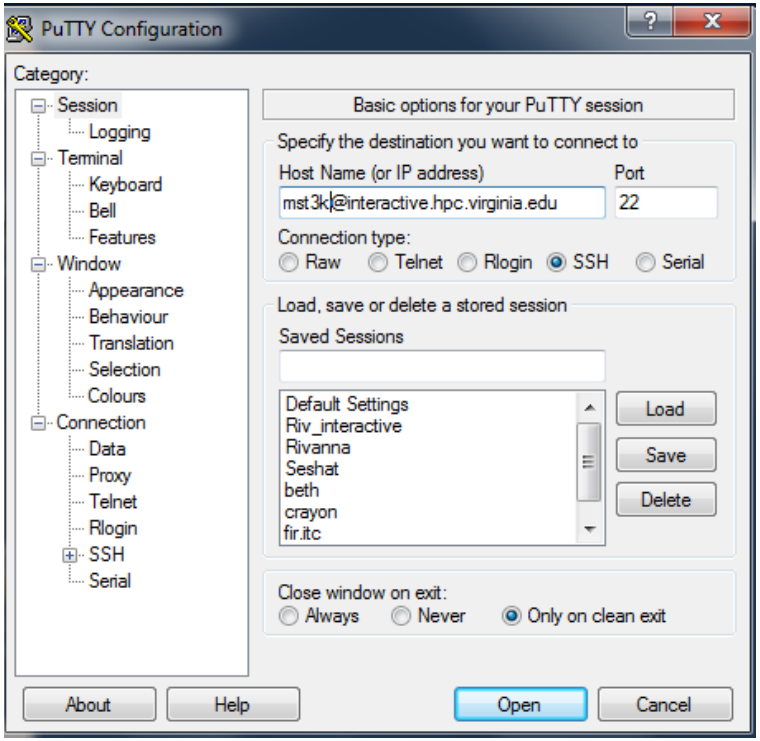
Connecting to the Cluster with PuTTY

- In the textbox for Host Name, type:

  userID@rivanna.hpc.virginia.edu

  and click on the Open button.

- When it prompts you for a password, use your Eservices password.



Graphical Access

- If you absolutely must have a desktop environment, we do have a license for FastX.
  - Clients are available for Windows, Mac, and Linux* systems.
  - But, you must download the installer from Rivanna. Instructions are at http://arcs.virginia.edu/fastx
  - The Linux client may not be compatible with the version of Linux running on your system.

After you have logged in . . .

- A terminal window will appear (except when using FastX).
  - Note: If you log in without graphical capabilities, the mouse or touchpad on your computer is basically useless.

- You will be in your Home directory.
  - To maneuver within your directory, you will need to use Unix/Linux commands.
  - To learn about Unix/Linux file system navigation, see our user guide page

Checking your Allocation

- To see how much core time you have available for running jobs, type `allocations` at the command-line prompt

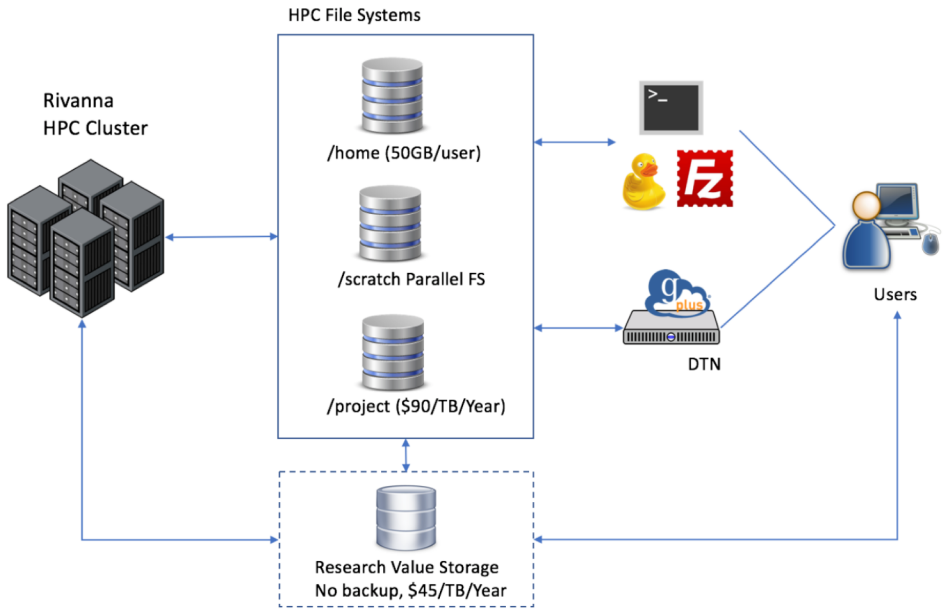
```
$ allocations

  Allocations available to ByoungDo_Kim (bk7k):

   * clslab: less than 5,000 service-units remaining
   * som_support: less than 5,000 service-units remaining
   * somrc: less than 49,495 service-units remaining
   * somrc-hpc-workshop: less than 50,000 service-units remaining
```

File Systems

- /home
  - 50GB/user
  - Free, data protection with 3-wk snapshot
- /scratch
  - Lustre parallel file system
  - Total 1.4PB of space, 100TB/user soft cap
  - Free, data purging policy
- /project
  - Paid service ($90/TB/Yr)
  - Permanent space, data protection with 3-wk snapshot
- Research Value Storage
  - Paid service, $45/TB/Yr
  - No data protection



File Systems (cont'd)

- Rivanna users will get 50GB of /home directory space and the access to /scratch file system for free.
- All Rivanna nodes with a connection to the InfiniBand network can access our Lustre filesystem (a storage system designed specifically for parallel access) with 1.4 PBs of **temporary** storage.
- Each user who can log into Rivanna will have access to 10 TBs of storage, located at **/scratch/<your userID>** .
- Important: This is **NOT permanent** storage.
  - Files older than 90 days may get deleted.
  - Permanent storage can be leased.

- Because the Lustre filesystem is designed for parallel work and is connected to the nodes with InfiniBand, we recommend that you do most of your work in your `/scratch` directory and copy your results to permanent storage for safe-keeping.

Checking your Storage

- To see the amount of scratch space that is available to you, type `sfsq` at the command line prompt.

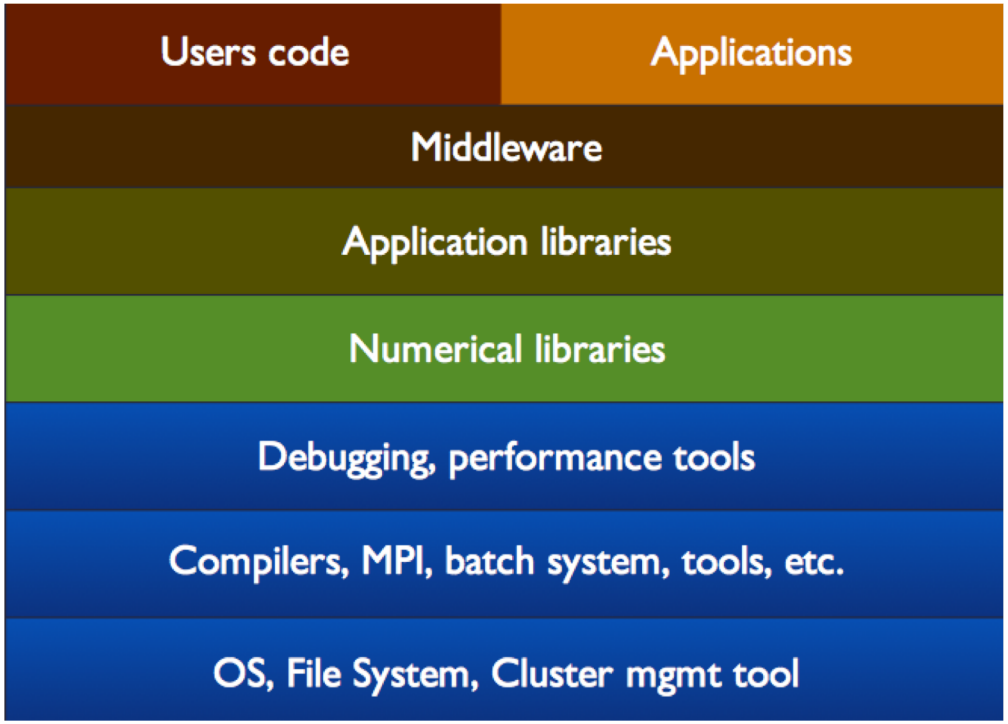
```
-bash4.1-$ sfsq

'scratch' usage status for 'mst3k', last
updated: 2016-09-08 16:26:12

 - ~28/10,000 GBs allocated disk space
 - 153/350,000 files created
 - 151/153 files marked for deletion due to
age limits

To view a list of all files marked for
deletion, please run 'sfsq -l'
```

HPC Cluster S/W Architecture



Modules

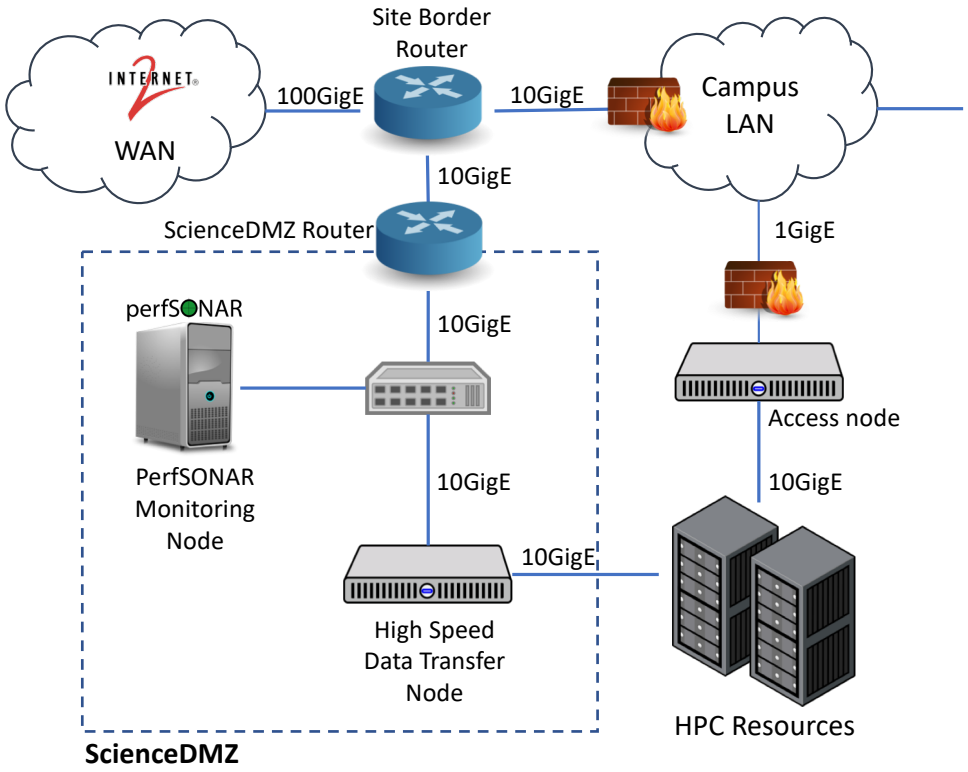
- Any application software that you want to use will need to be loaded with the module load command. For example:
  - module load R
  - module load python

- You will need to load the module any time that you create a new shell
  - Every time that you log out and back in
  - Every time that you run a batch job.

Module Details

- **`module avail`**
  - List all available modules and versions.
- **`module list`**
  - List modules loaded in your environment.
- **`module load mymod`**
  - Load the default module to set up the environment for some software.
  - **`module load mymod/N.M`**
  - Load a specific version N.M of software mymod.
  - **`module load mymod/compiler</N.M>`**
  - For compiler-specific modules, load the compiler and, optionally, the version.
- **`module remove mymod`**
  - Remove the module. Syntax follows `module load`.
- **`module purge`**
  - Clear all modules.

ScienceDMZ Network (FY18)



NSF-CC* Award (2016): Secure ScienceDMZ for high-speed research data transfer and sharing

- ScienceDMZ
  - Network resources dedicated for research data transfer
  - Bypassing firewalls, enabling fast transfer without bottlenecks
  - Dedicated data transfer node will be utilized as a gateway to internet2 network
- Connecting to Internet2 site in D.C. (through UVA Culpeper site)
- GlobusConnect runs for data transfer and sharing service
- https://discuss.rc.virginia.edu/t/globus-connect-data-transfer/345

Data Transfer Options

- Check out SOMRC User Guide
  - Secure Copy (scp)
  - Secure File Transfer Protocol (sftp)
  - 3rd-party software (Cyberduck, Filezilla, etc.)

- Globus Connect (https://www.globus.org/globus-connect-personal)
  - Standard data transfer method in ScienceDMZ
  - High-speed data transfer for research data

Batch Job Submission

- Running jobs on login nodes is not recommended
- The system is a shared resource; hundreds users running their jobs on a limited resource
- Users submit job via scheduler from the head nodes, and the jobs run on the back-end compute nodes
- A scheduler is taking users requests, allocating resources based on fair-share algorithm, executing jobs, and notifying users on the job process
- Rivanna uses SLURM (Simple Linux Utility for Resource Management)

Scheduler: SLURM

- SLURM is the Simple Linux Utility for Resource Management.
  - Open source.
  - Used on about half of the Top 500 systems.

- Documentation on SLURM:
  - https://arcs.virginia.edu/slurm
  - http://slurm.schedmd.com/documentation.html

Partitions (Queues)

- Rivanna has several partitions or queues for job submissions.
  - You will need to specify a partition when you submit a job.

Queue	Max Time per job (day)	Max Node# per job	Max Core# per job	Max Mem per core (GB)	Max Mem per job per node
standard	7	1	20	12	240
parallel	3	120	2400	6	120
largemem	7	1	16	62	992
gpu	3	4	8	12	240
knl	3	8	512	3	192

- To see the partitions that are available to you, type `queues` at the command-line prompt

SLURM Script

- A SLURM script is a bash script with SLURM directives (#SBATCH) and command-line instructions for running your program.

```
#!/bin/bash
#SBATCH --nodes=1                #total number of nodes for the job
#SBATCH --ntasks=1               #how many processes I will run
#SBATCH --time=1-12:00:00        #amount of time for the whole job
#SBATCH --partition=serial       #the queue/partition I will run on
#SBATCH --account=myGroupName #the account/allocation I am using

module load R                    #any modules that my job needs
Rscript myProg.R                 #command-line execution of my job
```

Submitting a SLURM Job

- To submit the SLURM command file to the queue, use the `sbatch` command at the command line prompt.

- For example, if the script on the previous slide is in a file named job_script.slurm, we can submit it as follows:

```
-bash-4.1$ sbatch job_script.slurm
Submitted batch job 18316
```

Checking Job Status

- To display the status of only your jobs, type:

  `squeue –u <your_user_id>`

- Or, simply:

  `jobq`

```
-bash-4.1$ squeue –u mst3k

JOBID   PARTITION   NAME     USER   ST   TIME   NODES  NODELIST(REASON)
18316   serial      job_sci  mst3k  R    1:45    1     udc-aw38-34-1

-bash-4.1$ jobq

JobID  Account      Queue    Status   Cores    Time-Left     Status-info
18316  MyGroupName  serial   running  1(1)      19:59:38     udc-aw38-34-1
```

Deleting a Job

- To delete a job from the queue, use the `scancel` command with the job ID number at the command line prompt:

```
-bash-4.1$ scancel 18316
```

Special Resources

- To request special resources (GPGPU, Lergemem, KNL, etc.), use –p option.
- The GPGPUs are in their own partition, but submitting to this partition is not sufficient to reserve the resource. An additional directive must be specified, **gres** for General Resource Scheduling.

```
#SBATCH -p gpu     #the queue/partition I will run on
#SBATCH -gres=gpu  #reserving a single GPU unit
#SBATCH -gres=gpu:4       #taking 4 GPUS
#SBATCH -gres=gpu:P100:2  #specifying two P-100 GPU
```

Interactive Jobs

- Use the interactive mode for interactive development: ijob -options

```
% ijob -c 1 -A mygroup -p standard --time=1-00:00:00

salloc: Pending job allocation 25394

salloc: job 25394 queued and waiting for resources
```

- Exit when you are done.

Parallel programming models

- Data Parallelism
  - Each processor performs the same task on different data
- Task Parallelism
  - Each processor performs a different task on the same data
- Most applications fall between these two

Data Parallel Programming Example

- One code will run on 2 CPUs
- Program has array of data to be operated on by 2 CPUs so array is split into two parts.

```
program:
…
if CPU=a then
   low_limit=1
   upper_limit=50
elseif CPU=b then
   low_limit=51
   upper_limit=100
end if
do I = low_limit,
upper_limit
   work on A(I)
end do
...
end program
```

CPU A

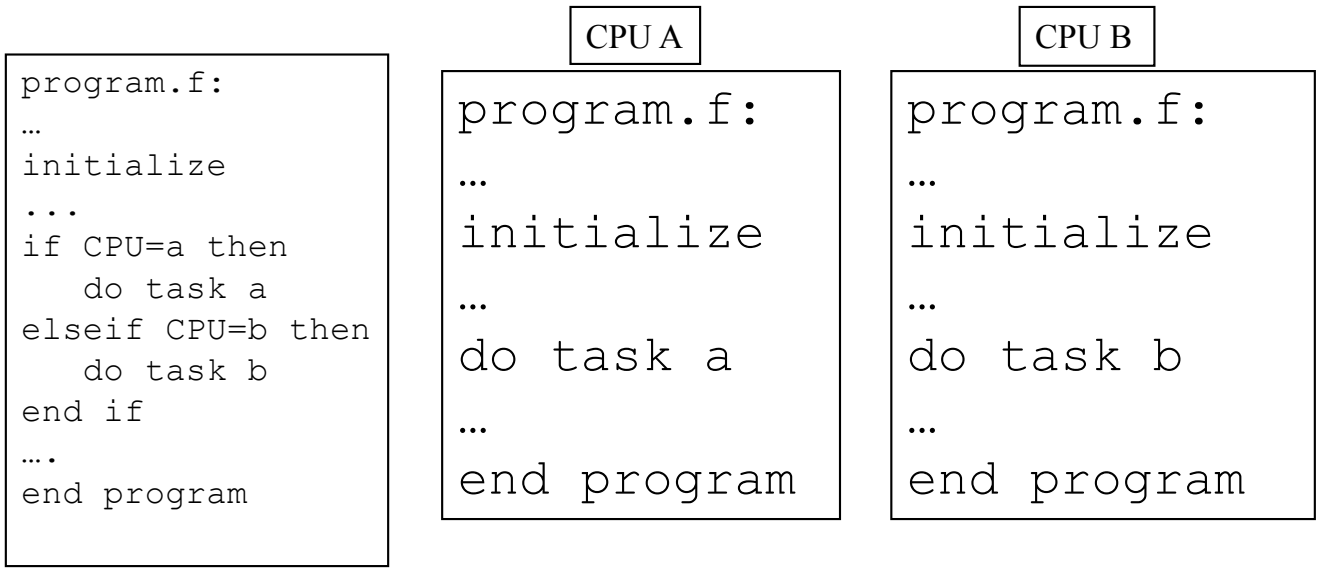
```
program:
…
low_limit=1
upper_limit=50
do I= low_limit,
upper_limit
   work on A(I)
end do
…
end program
```

CPU B

```
program:
…
low_limit=51
upper_limit=100
do I= low_limit,
upper_limit
   work on A(I)
end do
…
end program
```

Task Parallel Programming Example

- One code will run on 2 CPUs
- Program has 2 tasks (a and b) to be done by 2 CPUs



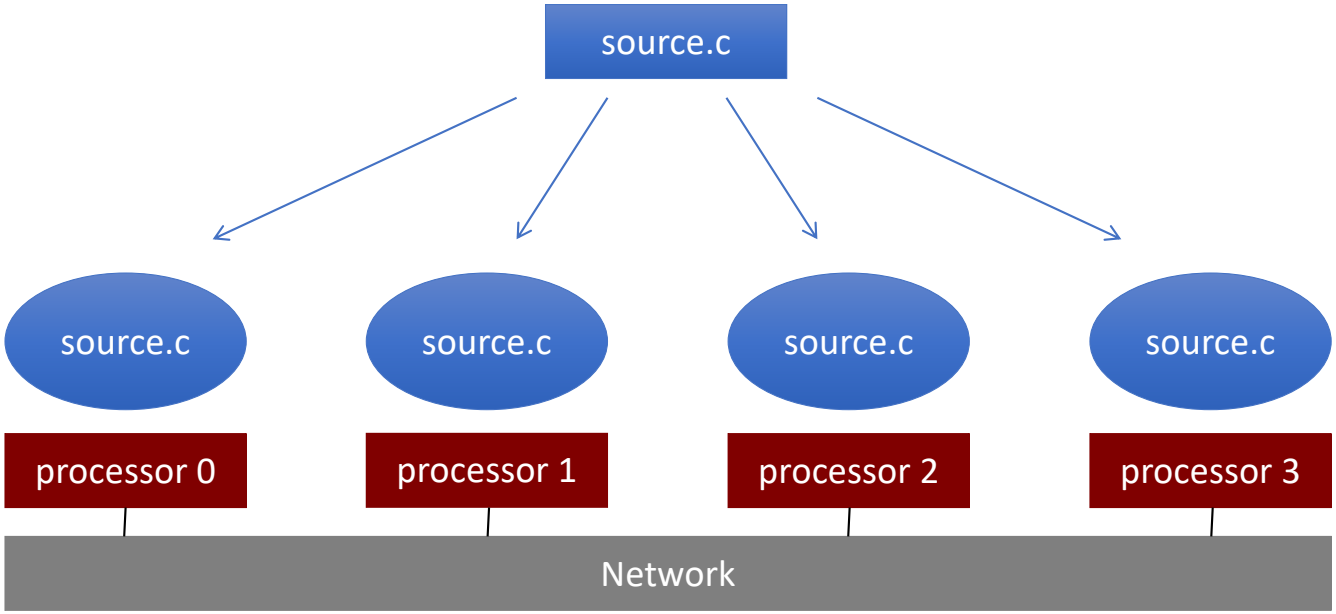
Distributed Memory Programming: MPI

- Distributed memory systems have separate address spaces for each processor
  - Local memory accessed faster than remote memory
  - Data must be manually decomposed
  - MPI is the standard for distributed memory programming (library of subprogram calls)
  - Older message passing libraries include PVM and P4; all vendors have native libraries such as SHMEM (T3E) and LAPI (IBM)

Single Program Multiple Data

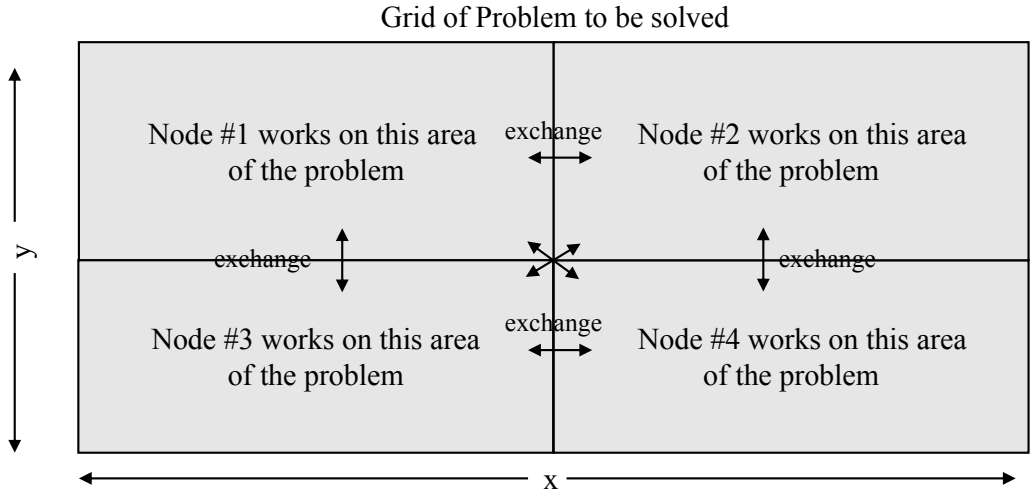
- SPMD: dominant programming model for shared and distributed memory machines.
  - One source code is written
  - Code can have conditional execution based on which processor is executing the copy
  - All copies of code start simultaneously and communicate and sync with each other periodically

SPMD Model


source.c
source.c
source.c
source.c
source.c
processor 0
processor 1
processor 2
processor 3
Network

Data Decomposition

- For distributed memory systems, the 'whole' grid or sum of particles is decomposed to the individual nodes
  - Each node works on its section of the problem
  - Nodes can exchange information



Typical Data Decomposition

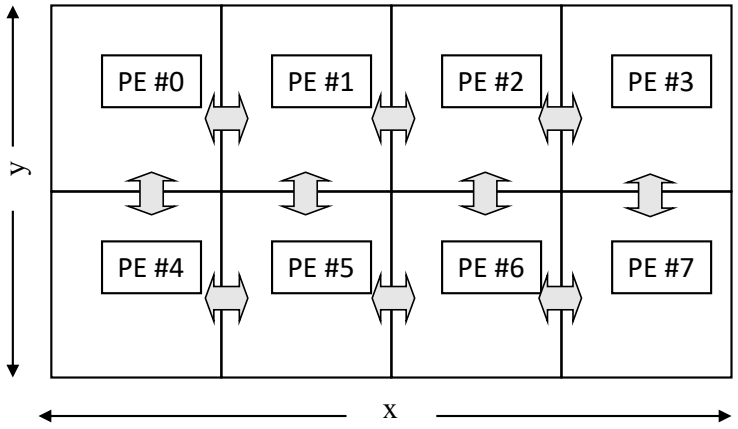
- Example: integrate 2-D propagation problem:

Starting partial differential equation:

$$\frac{\partial \Psi}{\partial t} = D \cdot \frac{\partial^2 \Psi}{\partial x^2} + B \cdot \frac{\partial^2 \Psi}{\partial y^2}$$

Finite Difference Approximation:

$$\frac{f_{i,j}^{n+1} - f_{i,j}^{n}}{\Delta t} = D \cdot \frac{f_{i+1,j}^{n} - 2f_{i,j}^{n} + f_{i-1,j}^{n}}{\Delta x^2} + B \cdot \frac{f_{i,j+1}^{n} - 2f_{i,j}^{n} + f_{i,j-1}^{n}}{\Delta y^2}$$



MPI Example #1

- Every MPI program needs these:

```
#include <mpi.h> /* the mpi include file */
int main(int argc, char *argv[])
{
  /* Initialize MPI */
  ierr = MPI_Init(&argc, &argv);
/* How many total PEs are there */
  ierr = MPI_Comm_size(MPI_COMM_WORLD, &nPEs);
/* What node am I (what is my rank? */
  ierr = MPI_Comm_rank(MPI_COMM_WORLD, &iam);
  ...
  ierr = MPI_Finalize();
}
```

MPI Example #2

```
#include
#include "mpi.h"

int main(int argc, char *argv[])
int argc;
char *argv[];
{
     int myid, numprocs;
     MPI_Init(&argc,&argv);
     MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
     MPI_Comm_rank(MPI_COMM_WORLD,&myid);
     /* print out my rank and this run's PE size*/
     printf("Hello from %d\n",myid," of ",numprocs);
     MPI_Finalize();
}
```