Running Jobs on Comet
(a practical guide)

Mary Thomas (mthomas@sdsc.edu)

April 19, 2019

Part 1

Outline

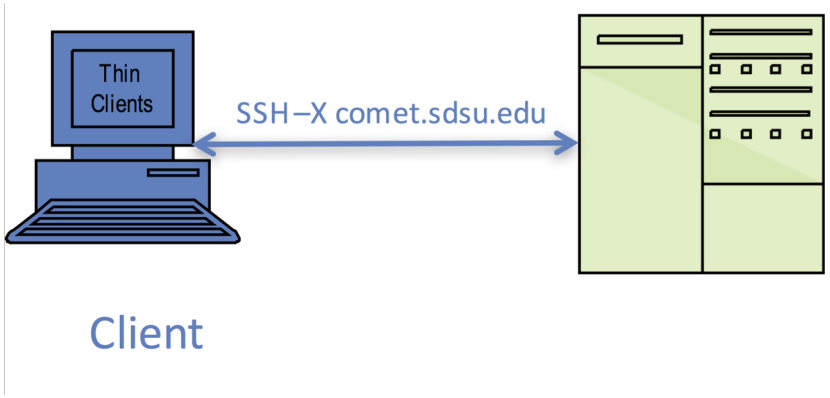
- **Getting Started/Comet System Environment**
- **Comet Overview**
- **Compiling and Linking Code**
- **Running Parallel Jobs**
  - Running OpenMP Jobs
  - Running MPI Jobs
  - Running Hybrid MPI-OpenMP Jobs
  - Running GPU/CUDA Jobs
- **Final Comments**

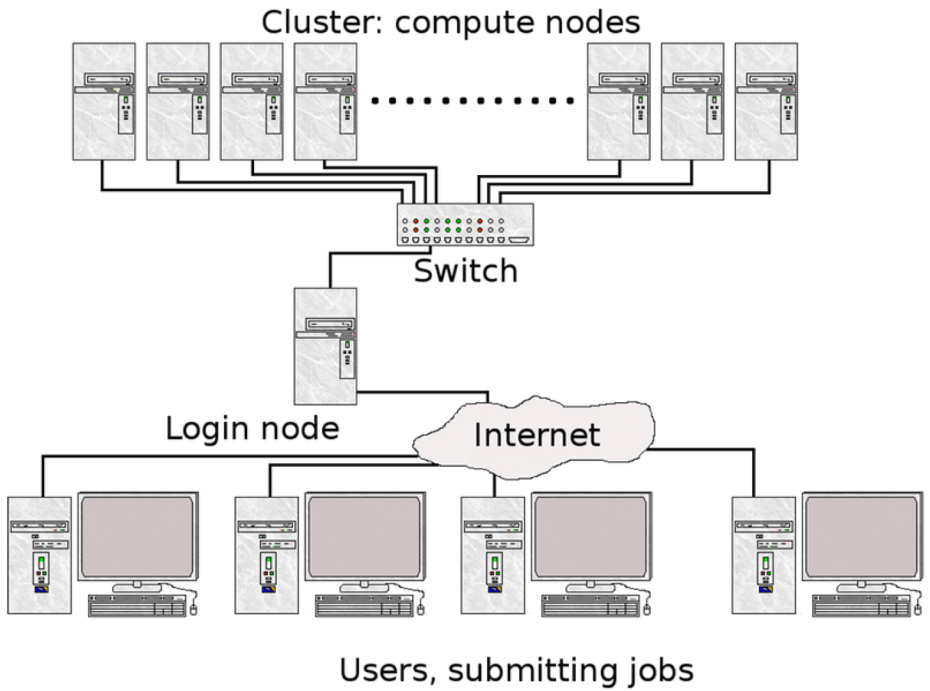
Getting Started

Basic Information

- Comet User Guide:
  - https://www.sdsc.edu/support/user_guides/comet.html
- Online repo for companion tutorial/webinar information:
  - https://github.com/sdsc-training/webinars
  - You must be familiar with running basic Unix commands: see the basic_skills and getting_started links in the webinar directory cited above.
- You must have a comet account in order to access the system. To obtain a trial account:
  - http://www.sdsc.edu/support/user_guides/comet.html#trial_accounts
- More training events listed at SDSC:
  - https://www.sdsc.edu/education_and_training/training.html

System Access: Logging On





- Linux/Mac – Use available *ssh* clients.
- Putty, Cygwin - *ssh* clients for windows:
  - http://www.chiark.greenend.org.uk/~sgtatham/putty/
- Login hostname for SDSC Comet: comet.sdsc.edu (198.202.113.252)
- Passwordless login tutorial:
  - https://www.tecmint.com/ssh-passwordless-login-using-ssh-keygen-in-5-easy-steps/

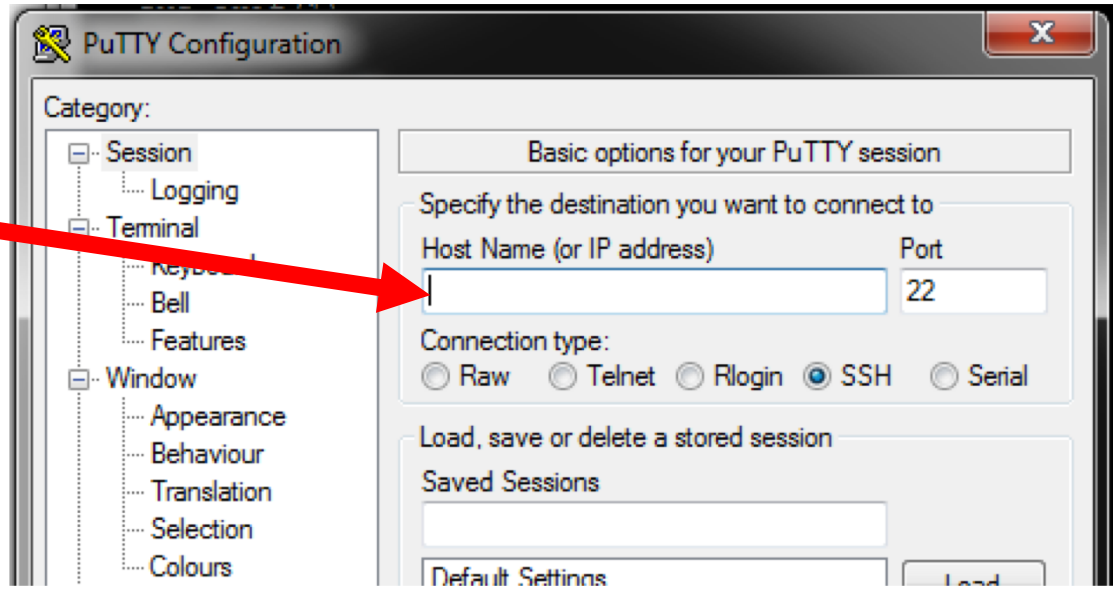
Logging into Comet

Mac/Linux:

```
ssh username@comet.sdsc.edu
```

Windows (PuTTY):

comet.sdsc.edu



Example of a terminal connection:

```
[$USER@wireless-169-228-105-171:~] ssh  comet.sdsc.edu
Warning: No xauth data; using fake authentication data for X11 forwarding.
Last login: Mon Jan  7 15:01:50 2019 from wireless-169-228-105-171.ucsd.edu
Rocks 6.2 (SideWinder)
Profile built 16:45 08-Feb-2016

Kickstarted 17:27 08-Feb-2016

                      WELCOME TO
      ________________  __  _____________
        -----/ ____/ __ \/  |/  / ____/_  __/
          --/ /   / / / / /|_/ / __/   / /
           / /___/ /_/ / /  / / /___  / /
           \____/\____/_/  /_/_____/ /_/

******************************************************************************

[1] Example Scripts: /share/apps/examples

[2] Filesystems:

     (a) Lustre scratch filesystem : /oasis/scratch/comet/$USER/temp_project
         (Preferred: Scalable large block I/O)

     (b) Compute/GPU node local SSD storage: /scratch/$USER/$SLURM_JOBID
         (Meta-data intensive jobs, high IOPs)

     (c) Lustre projects filesystem: /oasis/projects/nsf

     (d) /home/$USER : Only for source files, libraries, binaries.
         *Do not* use for I/O intensive jobs.

[3] Comet User Guide: http://www.sdsc.edu/support/user_guides/comet.html
******************************************************************************
[$USER@comet-ln3:~]
```

Obtaining Tutorial Example Code

- Create a test directory hold the comet example files (e.g. comet-examples)
- Copy the **PHYS244 directory** from the/share/apps/examples directory to your 'comet-examples' directory
- This tutorial will focus on examples in bold.

```
[$USER@comet-ln2 ~]$ mkdir comet-examples
[username@comet-ln3 ~]$ cp -r /share/apps/examples/PHYS244/ comet-examples/
[$USER@comet-ln3:~/comet-examples] ls -al PHYS244/
total 230
drwxr-xr-x 16 user use300 16 Aug  5 19:02 .
drwxr-xr-x  5 user use300  6 Aug  5 19:02 ..
drwxr-xr-x  2 user use300  5 Aug  5 19:02 COMPILER_EXAMPLES
drwxr-xr-x  2 user use300 14 Aug  6 00:56 CUDA
drwxr-xr-x  2 user use300 11 Aug  5 19:02 debug
drwxr-xr-x  3 user use300  3 Aug  5 19:02 HADOOP
drwxr-xr-x  2 user use300  6 Aug  6 00:12 HYBRID
drwxr-xr-x  2 user use300  6 Aug  5 19:02 LOCALSCRATCH
drwxr-xr-x  2 user use300  5 Aug  5 19:02 LOCALSCRATCH2
drwxr-xr-x  2 user use300  9 Nov 25 17:29 MKL
drwxr-xr-x  4 user use300  7 Aug  6 09:55 MPI
drwxr-xr-x  2 user use300  8 Aug  5 19:02 OpenACC
drwxr-xr-x  2 user use300  8 Aug  5 23:25 OPENMP
drwxr-xr-x  3 user use300  5 Aug  5 19:02 pytorch
drwxr-xr-x  4 user use300  4 Aug  5 19:02 SPARK
drwxr-xr-x  4 user use300  5 Aug  5 19:02 TensorFlow
```

Comet Overview

HPC for the “long tail of science:"

Img Src taken from:
https://www.cc.gatech.edu/~echow/ipcc/hpc-course/HPC-networks.pdf

Stone Soupercomputer: cheapest cost/flop=$0
~20 MFlops
https://web.archive.org/web/20031121211117/http://stonesoup.esd.ornl.gov/

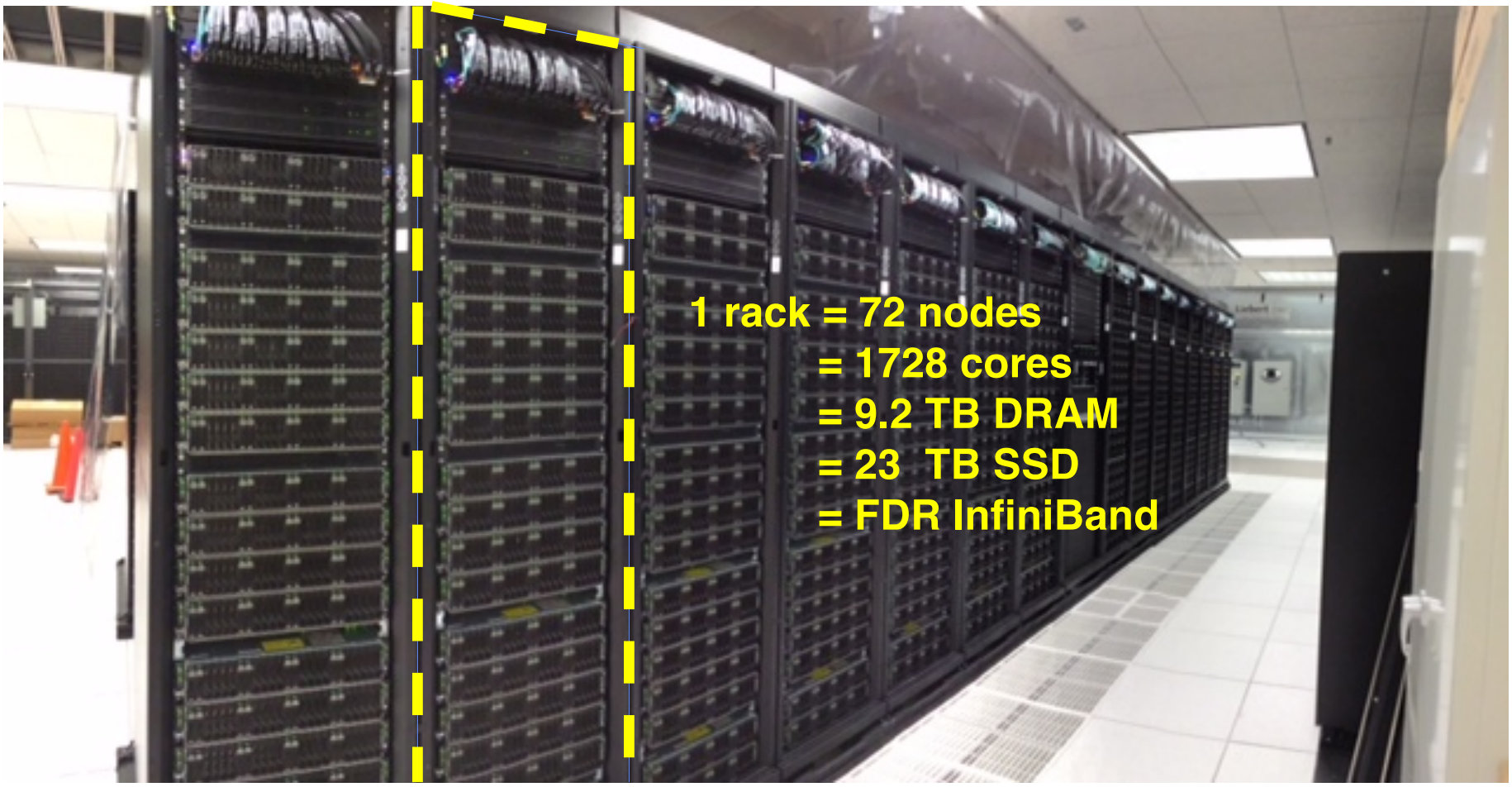
Comet: HPC for the “long tail of science:"

- “Long Tail” - majority of computational research is performed *at modest scale*: large number jobs that run for less than 48 hours, but can be computationally intensive and generate large amounts of data.
- Comet is an NSF-funded system available through the eXtreme Science and Engineering Discovery Environment (XSEDE) program.
- Advanced computing environment: supports science gateways, interactive computing, Jupyter notebooks, containers.

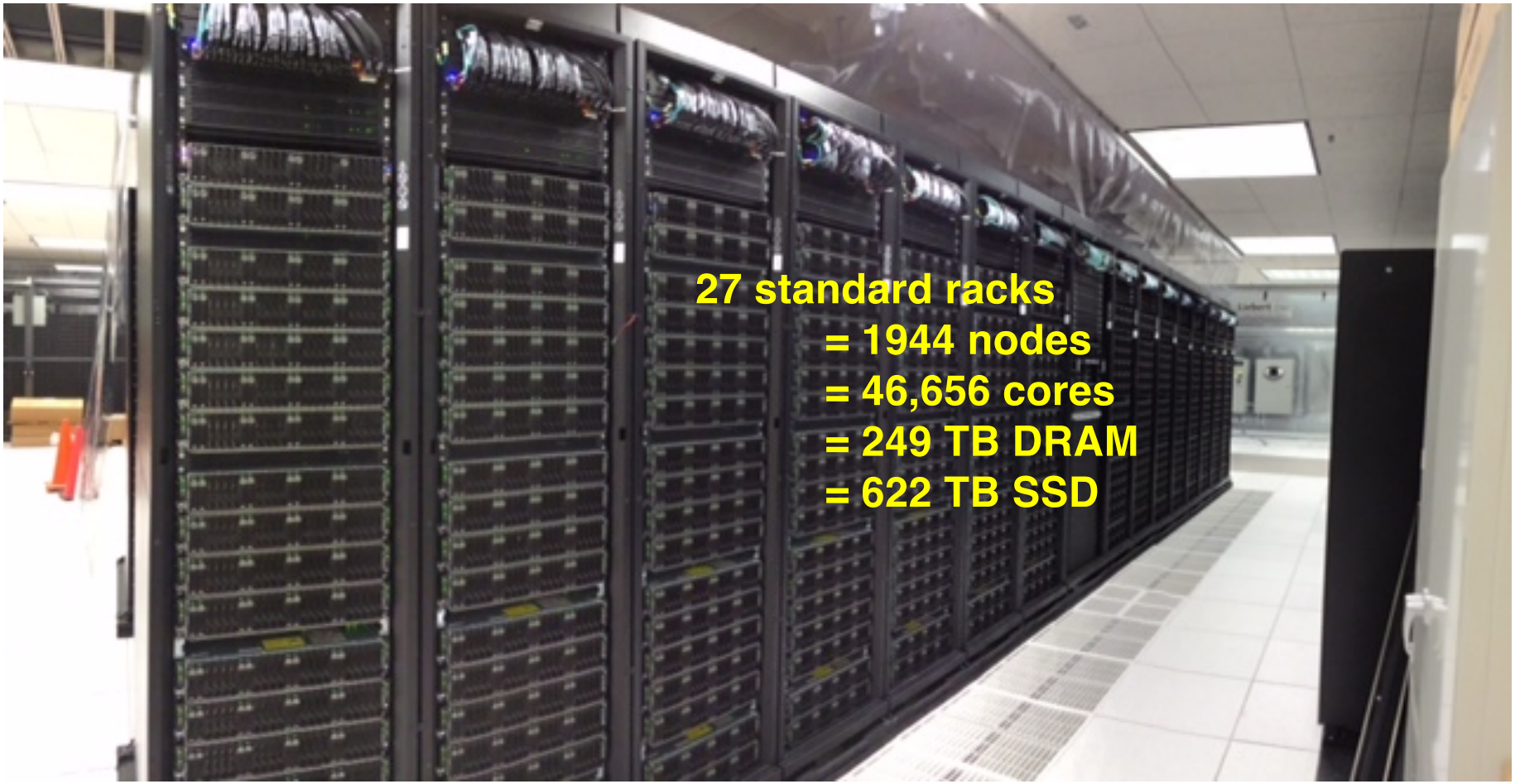
Comet: System Characteristics

- **Total peak flops ~2.76 PF**
- **Dell primary integrator**
  - Intel Haswell processors w/ AVX2
  - Mellanox FDR InfiniBand
- **1944 Standard compute nodes (46,656 cores)**
  - Dual CPUs, each 12-core, 2.5 GHz
  - 128 GB DDR4 2133 MHz DRAM
  - 2*160GB GB SSDs (local disk)
- **72 GPU nodes**
  - 36 nodes same as standard nodes *plus* Two NVIDIA K80 cards, each with dual Kepler3 GPUs
  - 36 nodes with 2 14-core Intel Broadwell CPUs plus 4 NVIDIA P100 GPUs
- **4 large-memory nodes**
  - 1.5 TB DDR4 1866 MHz DRAM
  - Four Haswell processors/node
  - 64 cores/node
- **Hybrid fat-tree topology**
  - FDR (56 Gbps) InfiniBand (bisection)
  - Rack-level (72 nodes, 1,728 cores) full bisection bandwidth
  - 4:1 oversubscription cross-rack
- **Performance Storage**
  - 7.6 PB, 200 GB/s; Lustre
  - Scratch & Persistent Storage segments
- **Durable Storage**
  - 6 PB, 100 GB/s; Lustre
  - Automatic backups of critical data
- **Home directory storage**
- **Gateway hosting nodes**
- **Virtual image repository**
- **100 Gbps external connectivity to Internet2 & ESNet**

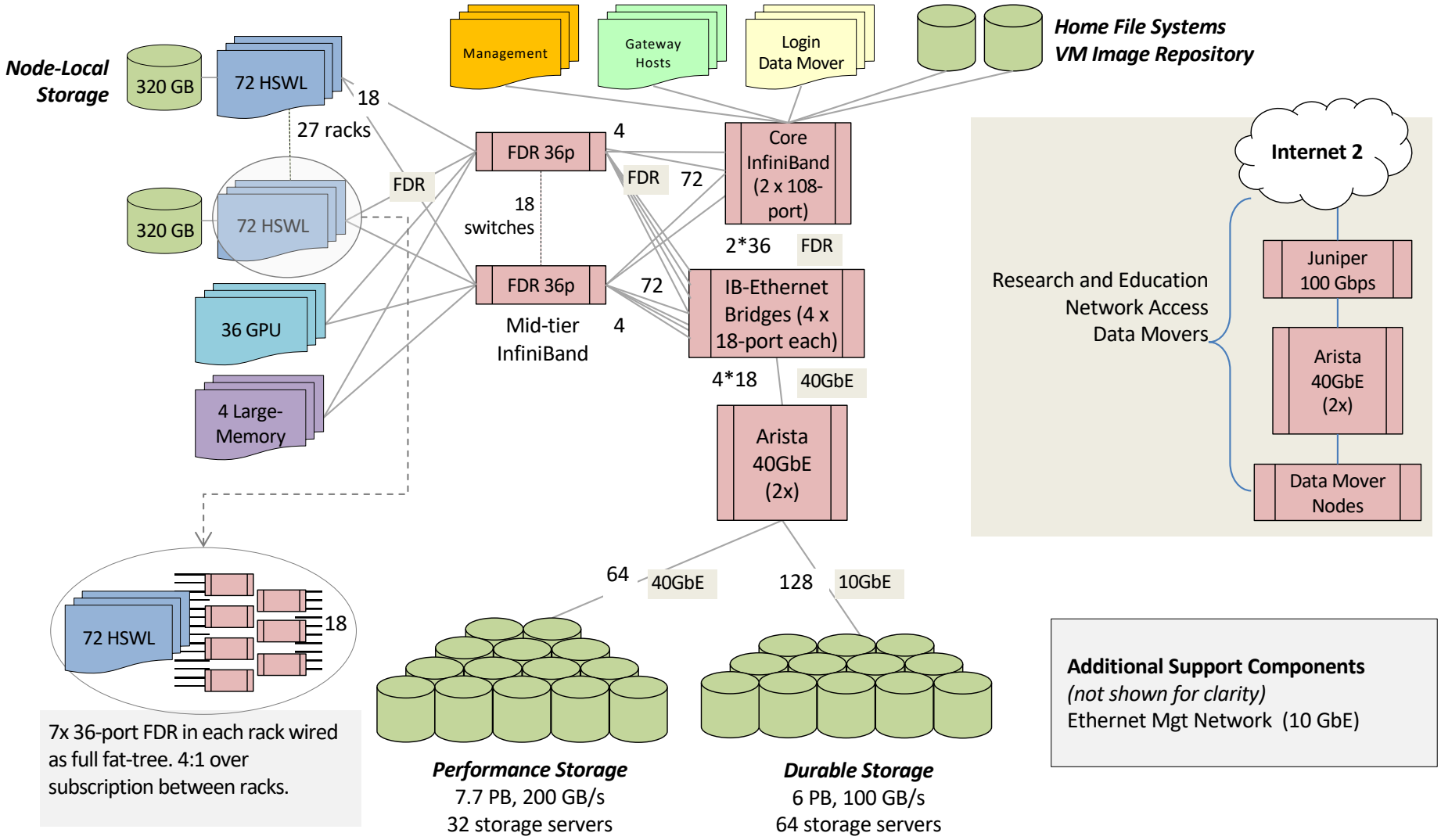
~67 TF supercomputer in a rack



And 27 single-rack supercomputers



Comet Network Architecture
InfiniBand compute, Ethernet Storage



Comet: Filesystems

- **Lustre filesystems – Good for scalable large block I/O**
  - Accessible from all compute and GPU nodes.
  - /oasis/scratch/comet - 2.5PB, peak performance: 100GB/s. Good location for storing large scale scratch data during a job.
  - /oasis/projects/nsf - 2.5PB, peak performance: 100 GB/s. Long term storage.
  - ***Not good for large # of small files or small block I/O***. Please. Don't.
- **SSD filesystems**
  - /scratch local to each native compute node – 210GB on regular compute nodes, 285GB on GPU, large memory nodes, 1.4TB on selected compute nodes.
  - SSD location is good for writing small files and temporary scratch files. Purged at the end of a job.
- **Home directories (/home/$USER)**
  - Source trees, binaries, and small input files.
  - ***Not good for large scale I/O.***

Comet File Systems

Path	Purpose	User Access Limits	Lifetime
$HOME	NFS storage; Source code, important files	100 GB	Backed-up
/oasis/scratch/comet/ $USER/temp_project	Global/Parallel Lustre FS; temp storage for distributed access	500 GB	No backup
/oasis/projects/nsf	Global/Parallel Lustre FS; project storage	~2.5 PB total	Backed-up
/scratch/$USER/$SL URM_JOB_ID	Local SSD on batch job node fast per-node access	210 GB per compute node, 286GB on GPU, Large memory nodes	Purged after job ends

Managing the Environment with Modules

Comet: System Environment

- **Modules are used to manage environment for users.**
- **Default environment:**

```
$ module li
Currently Loaded Module files:
  1) intel/2013_sp1.2.144   2) mvapich2_ib/2.1        3) gnutools/2.69
```

- **Listing available modules:**

```
$ module av
------------------------ /opt/modulefiles/mpi/.intel -------------------------
intelmpi/2016.3.210(default) mvapich2_ib/2.1(default)
mvapich2_gdr/2.1(default)    openmpi_ib/1.8.4(default)
mvapich2_gdr/2.2
--------------------- /opt/modulefiles/applications/.intel ---------------------
atlas/3.10.2(default)     lapack/3.6.0(default)     scalapack/2.0.2(default)
boost/1.55.0(default)     mxml/2.9(default)         slepc/3.6.2(default)
…
…
```

Modules: Managing the User Environment

A few popular commands

Command	Description
module list	List the modules that are currently loaded
module avail	List the modules that are available
module display <module_name>	Show the environment variables used by and how they are affected
module show <module_name>	Same as display
module unload	Remove from the environment
module load	Load into the environment
module swap	Replace with in the environment

Module Command Examples

- Default environment: list, li

```
[$USER@comet-ln3:~/comet-examples]  module li
Currently Loaded Module files:   1) intel/2013_sp1.2.144   2) mvapich2_ib/2.1       3) gnutools/2.69
```

- List available modules: available, avail, av

```
[$USER@comet-ln3:~/comet-examples] module av

---------------------------------- /opt/modulefiles/applications/.intel -----------------------------
-----
atlas/3.10.2(default)     hdf5/1.8.14(default)      papi/5.4.1(default)       sundials/2.6.2(default)
boost/1.55.0(default)     ipm/2.0.3(default)        parmetis/4.0.3(default)   superlu/4.2(default)
fftw/2.1.5                lapack/3.6.0(default)     pdt/3.20(default)         tau/2.23(default)
fftw/3.3.4(default)       mxml/2.9(default)         petsc/3.6.3(default)
trilinos/11.12.1(default)
gsl/1.16                  netcdf/3.6.2              scalapack/2.0.2(default)
gsl/2.1(default)          netcdf/4.3.2(default)     slepc/3.6.2(default)
hdf4/2.11(default)        p3dfft/2.7.4(default)     sprng/2.0b(default)

$ module av ------------------------ /opt/modulefiles/mpi/.intel -------------------------
intelmpi/2016.3.210(default) mvapich2_ib/2.1(default)
mvapich2_gdr/2.1(default)    openmpi_ib/1.8.4(default) mvapich2_gdr/2.2 --------------------
/opt/modulefiles/applications/.intel ---------------------
atlas/3.10.2(default)     lapack/3.6.0(default)     scalapack/2.0.2(default)
boost/1.55.0(default)     mxml/2.9(default)         slepc/3.6.2(default)

… MORE….
```

Module Command Examples

- Load a module, and show what it does

```
[$USER@comet-ln3:~/comet-examples] env
HOSTNAME=comet-ln3.sdsc.edu
IPPROOT=/opt/intel/composer_xe_2013_sp1.2.144/ipp
INTEL_LICENSE_FILE=/opt/intel/composer_xe_2013_sp1.2.144/licenses:/opt/intel/licenses:/root/
intel/licenses
TERM=xterm-256color
SHELL=/bin/bash
HISTSIZE=5000
GDBSERVER_MIC=/opt/intel/composer_xe_2013_sp1.2.144/debugger/gdb/target/mic/bin/gdbserver
SSH_CLIENT=169.228.105.171 58704 22
[SNIP]
HOME=/home/user
ROLLSROOT=/opt/rocks/share/devel/src/roll
MPIHOME=/opt/mvapich2/intel/ib
FFTWHOME=/opt/fftw/3.3.4/intel/mvapich2_ib
SDSCHOME=/opt/sdsc
PYTHONPATH=/opt/sdsc/lib
LOGNAME=user
QTLIB=/usr/lib64/qt-3.3/lib
CVS_RSH=ssh
SSH_CONNECTION=169.228.105.171 58704 198.202.113.252 22
MODULESHOME=/usr/share/Modules
MKL_ROOT=/opt/intel/composer_xe_2013_sp1.2.144/mkl
LESSOPEN=||/usr/bin/lesspipe.sh %s
INFOPATH=/opt/intel/composer_xe_2013_sp1.2.144/debugger/gdb/intel64/share/info/:/opt/intel/c
omposer_xe_2013_sp1.2.144/debugger/gdb/intel64_mic/share/info/
DISPLAY=localhost:42.0
INCLUDE=/opt/intel/composer_xe_2013_sp1.2.144/mkl/include
INTELHOME=/opt/intel/composer_xe_2013_sp1.2.144
G_BROKEN_FILENAMES=1
BASH_FUNC_module()=() {  eval `/usr/bin/modulecmd bash $*`
}
_=/bin/env
```

Module: check Environment

Once you have loaded the modules, you can check the system variables that are available for you to use.

```
[$USER@comet-ln3:~/comet-examples] module load fftw/3.3.4
[$USER@comet-ln3:~/comet-examples]
[$USER@comet-ln3:~/comet-examples] module li
Currently Loaded Modulefiles:
  1) intel/2013_sp1.2.144   2) mvapich2_ib/2.1        3) gnutools/2.69          4)
fftw/3.3.4
[$USER@comet-ln3:~/comet-examples] module show fftw/3.3.4
-------------------------------------------------------------------
/opt/modulefiles/applications/.intel/fftw/3.3.4:

module-whatis            fftw
module-whatis            Version: 3.3.4
module-whatis            Description: fftw
module-whatis            Compiler: intel
module-whatis            MPI Flavors: mvapich2_ib openmpi_ib
setenv                   FFTWHOME /opt/fftw/3.3.4/intel/mvapich2_ib
prepend-path             PATH /opt/fftw/3.3.4/intel/mvapich2_ib/bin
prepend-path             LD_LIBRARY_PATH /opt/fftw/3.3.4/intel/mvapich2_ib/lib
prepend-path             LIBPATH /opt/fftw/3.3.4/intel/mvapich2_ib/lib
-------------------------------------------------------------------
```

Using Script to load modules

Control and guarantee the current working environment. In order for the commands run inside a script (child shell) to change the parent shell, you must use the *source* command.

```
[comet-ln3:~] source ./loadgpuenv.sh
[comet-ln3:~] module list
Currently Loaded Modulefiles:
  1) gnutools/2.69   2) cuda/7.0
[comet-ln3:~] which nvcc
/usr/local/cuda-7.0/bin/nvcc
[mthomas@comet-ln3:~] which mpirun
/usr/bin/which: no mpirun in (/opt/gnu/gcc/bin:usr/local/bin……..)

[comet-ln3:~] source loadintelenv.sh
[comet-ln3:~] module list
Currently Loaded Modulefiles:
  1) gnutools/2.69  2) intel/2013_sp1.2.144 3) mvapich2_ib/2.1
[mthomas@comet-ln3:~] which mpirun
/opt/mvapich2/intel/ib/bin/mpirun
[mthomas@comet-ln3:~] which nvcc
/usr/bin/which: no nvcc in (/opt/gnu/gcc/bin:usr/local/bin……..)
```

```
comet-ln3:~] cat loadgpuenv.sh
#!/bin/bash
module purge
module load gnutools
module load cuda
```

```
[comet-ln3:~] cat loadintelenv.sh
module purge
module load gnutools
module load intel mvapich2_ib
```

Module: command not found

- Sometimes encountered when switching from one shell to another or attempting to run the module command from within a shell script or batch job.
- Module command may not be inherited to the shell
- To keep this from happening, execute the following command:
  - command line (interactive shells)
    - source /etc/profile.d/modules.sh
  - OR add to your shell script (including Slurm batch scripts)

Compiling & Linking

Compiling & Linking: Topics

- Supported Compiler Types
- Intel Compiling
- PGI Compiling
- GNU Compiling
- GPU Compiling

Supported Compiler Types

- **Comet compute nodes support several parallel programming models:**
  - **MPI**: Default Intel Compiler**: intel/2013_sp1.2.144**;
    - Versions 2015.2.164 and 2016.3.210 available.
    - Other options: openmpi_ib/1.8.4 (and 1.10.2), Intel MPI, mvapich2_ib/2.1
    - mvapich2_gdr: GPU direct enabled version
  - **OpenMP & Pthreads**:
    - All compilers (GNU, Intel, PGI) have OpenMP flags.
  - **GPU** nodes: support CUDA, OpenACC.
  - **Hybrid** modes are possible (see examples below).

Suggested Compilers

- Default/Suggested Compilers to used based on programming model and languages:

	Serial	MPI	OpenMP	MPI + OpenMP
Fortran	ifort	mpif90	ifort -openmp	mpif90 -openmp
C	icc	mpicc	icc -openmp	mpicc -openmp
C++	icpc	mpicxx	icpc -openmp	mpicxx -openmp

- In this tutorial, we include hands-on examples that cover many of the cases in the table:
  (1) MPI
  (2) OpenMP
  (3) HYBRID
  (4) Local scratch

Using the Intel Compilers

- Intel compilers and MVAPICH2 MPI implementation will be loaded by default.
- If you have modified your environment, you can reload by executing the module purge & load commands at the Linux prompt, or placing the load command in your startup file (`~/.cshrc or ~/.bashrc`)

```
[$USER@comet-ln2:~] module purge
[$USER@comet-ln2:~] module list
No Modulefiles Currently Loaded.
[$USER@comet-ln2:~] module load gnutools
[$USER@comet-ln2:~] module list
Currently Loaded Modulefiles:
  1) gnutools/2.69
[$USER@comet-ln2:~] module load intel mvapich2_ib
[$USER@comet-ln2:~] module list
Currently Loaded Modulefiles:
  1) gnutools/2.69    2) intel/2013_sp1.2.144   3) mvapich2_ib/2.1
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] which mpicc
/opt/mvapich2/intel/ib/bin/mpicc
```

```
[comet-ln3:~] source loadintelenv.sh
[comet-ln3:~] module list
Currently Loaded Modulefiles:
  1) gnutools/2.69  2) intel/2013_sp1.2.144 3) mvapich2_ib/2.1
[mthomas@comet-ln3:~] which mpirun
/opt/mvapich2/intel/ib/bin/mpirun
```

Using the Intel Compilers

- For Intel Advanced Vector Extensions (AVX2) support, compile with the -xHOST option.
  - https://en.wikipedia.org/wiki/Advanced_Vector_Extensions (128/256bit SIMD, Vector ops (MPI broadcast, gather, …)
  - Note that -xHOST alone does not enable aggressive optimization, so compilation with -O3 is also suggested.
  - The -fast flag invokes -xHOST, but should be avoided since it also turns on interprocedural optimization (-ipo), which may cause problems in some instances.
- Intel Math Kernal Lib (MKL) libraries are available as part of the "intel" modules on Comet.
  - Once this module is loaded, the environment variable MKL_ROOT points to the location of the mkl libraries.
  - The MKL link advisor can be used to ascertain the link line (change the MKL_ROOT aspect appropriately).

Using the Intel Compilers

- In the example below, we are working with the HPC examples that can be found in PHYS244/MKL :

```
[$USER@comet-14-01:~]cd comet-examples/PHYS244/MKL
[$USER@comet-14-01:~/comet-examples/PHYS244/MKL]  pwd
/home/user/comet-examples/PHYS244/MKL
[$USER@comet-14-01:~/comet-examples/PHYS244/MKL] ls -al
total 25991
drwxr-xr-x  2 user use300        9 Nov 25 17:20 .
drwxr-xr-x 16 user use300       16 Aug  5 19:02 ..
-rw-r--r--  1 user use300      325 Aug  5 19:02 compile.txt
-rw-r--r--  1 user use300     6380 Aug  5 19:02 pdpttr.c
-rwxr-xr-x  1 user use300 44825440 Nov 25 16:55 pdpttr.exe
-rw-r--r--  1 user use300      188 Nov 25 16:57 scalapack.20294236.comet-07-27.out
-rw-r--r--  1 user use300      376 Aug  5 19:02 scalapack.sb
```

Using the Intel Compilers

The file `compile.txt` contains the full command to compile the `pdpttr.c` program statically linking 64 bit scalapack libraries on Comet:

```
[$USER@comet-14-01:~/comet-examples/PHYS244/MKL] cat compile.txt
mpicc -o pdpttr.exe pdpttr.c  -I$MKL_ROOT/include
${MKL_ROOT}/lib/intel64/libmkl_scalapack_lp64.a -Wl,--start-group
${MKL_ROOT}/lib/intel64/libmkl_intel_lp64.a
${MKL_ROOT}/lib/intel64/libmkl_core.a
${MKL_ROOT}/lib/intel64/libmkl_sequential.a -Wl,--end-group
${MKL_ROOT}/lib/intel64/libmkl_blacs_intelmpi_lp64.a -lpthread -lm
```

Verify your environment, then compile the command:

```
[$USER@comet-14-01 :~/comet-examples/PHYS244/MKL] source ~/loadintelenv.sh
[$USER@comet-14-01 :~/comet-examples/PHYS244/MKL] module list
Currently Loaded Modulefiles:
  1) gnutools/2.69          2) intel/2013_sp1.2.144   3) mvapich2_ib/2.1
[$USER@comet-14-01:~/comet-examples/PHYS244/MKL] mpicc -o pdpttr.exe pdpttr.c  -I$MKL_ROOT/include
${MKL_ROOT}/lib/intel64/libmkl_scalapack_lp64.a -Wl,--start-group
${MKL_ROOT}/lib/intel64/libmkl_intel_lp64.a ${MKL_ROOT}/lib/intel64/libmkl_core.a
${MKL_ROOT}/lib/intel64/libmkl_sequential.a -Wl,--end-group
${MKL_ROOT}/lib/intel64/libmkl_blacs_intelmpi_lp64.a -lpthread -lm
```

For more information on the Intel compilers run: [ifort | icc | icpc] -help

Using the PGI Compilers

- PGI (formerly The Portland Group, Inc.), was a company that produced a set of commercially available Fortran, C and C++ compilers for high-performance computing systems.
- It is now owned by NVIDIA.
- PGI compilers can be loaded by executing the following commands at the Linux prompt or placing in your startup file (~/.cshrc or ~/.bashrc).
- For AVX support, compile with -fast

```
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] module purge
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] module load gnutools
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] module load pgi
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] module load mvapich2_ib
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] module list
Currently Loaded Modulefiles:
  1) gnutools/2.69     2) pgi/17.5          3) mvapich2_ib/2.1

[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] which mpicc
/opt/mvapich2/pgi/ib/bin/mpicc
```

- For more information on the PGI compilers run: man [pgf90 | pgcc | pgCC]

Recommended PGI Compilers

	Serial	MPI	OpenMP	MPI+OpenMP
Fortran	pgf90	mpif90	pgf90 -mp	mpif90 -mp
C	pgcc	mpicc	pgcc -mp	mpicc -mp
C++	pgCC	mpicxx	pgCC -mp	mpicxx -mp

- PGI supports the following high-level languages:
  - Fortran 77, 90/95/2003, 2008 (partial)
  - High Performance Fortran (HPF)
  - ANSI C99 with K&R extensions
  - ANSI/ISO C++
  - CUDA Fortran
  - OpenCL
  - OpenACC
  - OpenMP

Using the GNU Compilers

- The GNU compilers can be loaded by executing the following commands at the Linux prompt or placing in your startup files (~/.cshrc or ~/.bashrc)

```
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] module purge
Unloading compiler-dependent module gnutools/2.69
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] module load gnutools
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] module load gnu
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] module load openmpi_ib
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL]
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL] which mpicc
/opt/openmpi/gnu/ib/bin/mpicc
[$USER@comet-ln2:~/comet-examples/PHYS244/MKL]
```

- For AVX support, compile with -mavx.
- Note that AVX support is only available in version 4.7 or later, so it is necessary to explicitly load the gnu/4.9.2 module until such time that it becomes the default.

Using the GNU Compilers

Table of recommended GNU compilers:

	Serial	MPI	OpenMP	MPI+OpenMP
Fortran	gfortran	mpif90	gfortran -fopenmp	mpif90 -fopenmp
C	gcc	mpicc	gcc -fopenmp	mpicc -fopenmp
C++	g++	mpicxx	g++ -fopenmp	mpicxx -fopenmp

Running Jobs On Comet

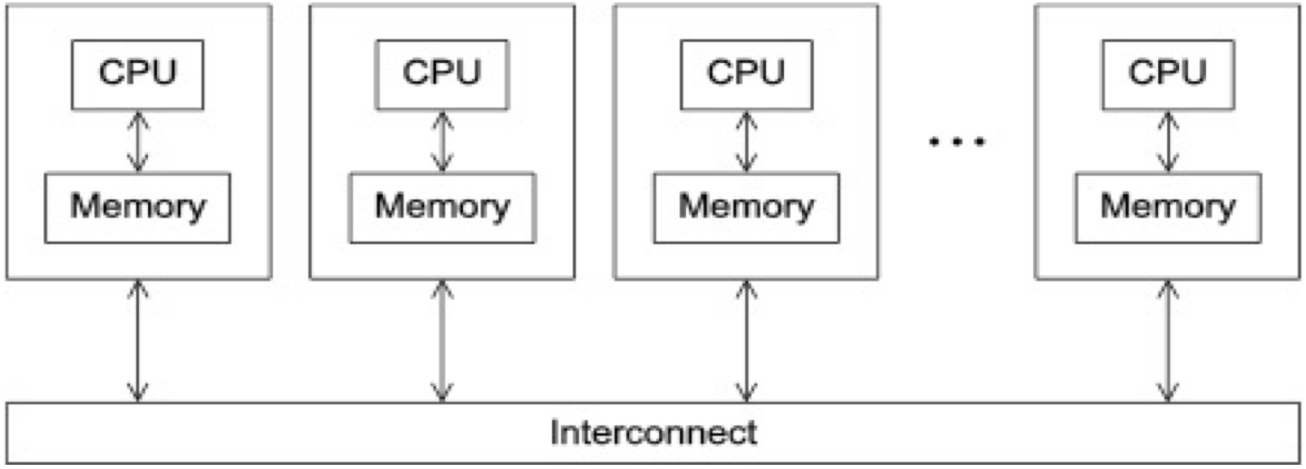
Factors Impacting Job Execution

- **Parallel Models:**
  - Impacts language used, libraries, performance.
- **How you choose to run the job:**
  - Command line execution
  - Batch/queuing System -- Comet uses the Simple Linux Utility for Resource Management (SLURM):
    - Batch Queue
    - Interactive jobs
- **Data I/O choices (topic of upcoming Webinar):**
  - https://www.sdsc.edu/education_and_training/training.html

Parallel Models: Memory

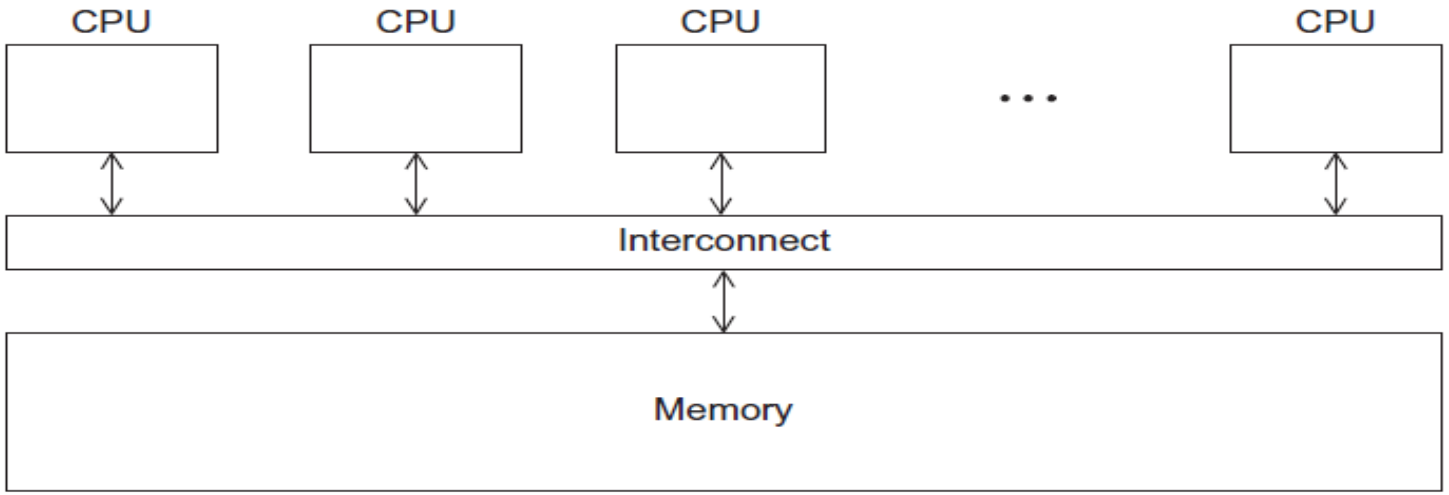
- **Distributed Memory**
- **Shared Memory**
- **Implemented in several languages:**
  - FORTRAN, C, Python, OOPs (sort-of)
- **Large number of libraries and API's**
- **Adds to compilation/linking complexity**

Distributed Memory



- Programs that run asynchronously, pass messages for communication and coordination between resources.
- Examples include: SOA-based systems, massively multiplayer online games, peer-to-peer apps.
- Different types of implementations for the message passing mechanism: HTTP, RPC-like connectors, message queues
- HPC historically uses the **Message Passing Interface (MPI)**

Parallel Models: Shared Memory



- **CPUs all share same localized memory (SHMEM);**
  - Coordination and communication between tasks via interprocessor communication (IPC) or virtual memory mappings.
- **May use: uniform or non-uniform memory access (UMA or NUMA); cache-only memory architecture (COMA).**
- **Most common HPC API's for using SHMEM:**
  - Portable Operating System Interface (POSIX); Open Multi-Processing (OpenMP) designed for parallel computing – best for multi-core computing.

Running Jobs on Comet

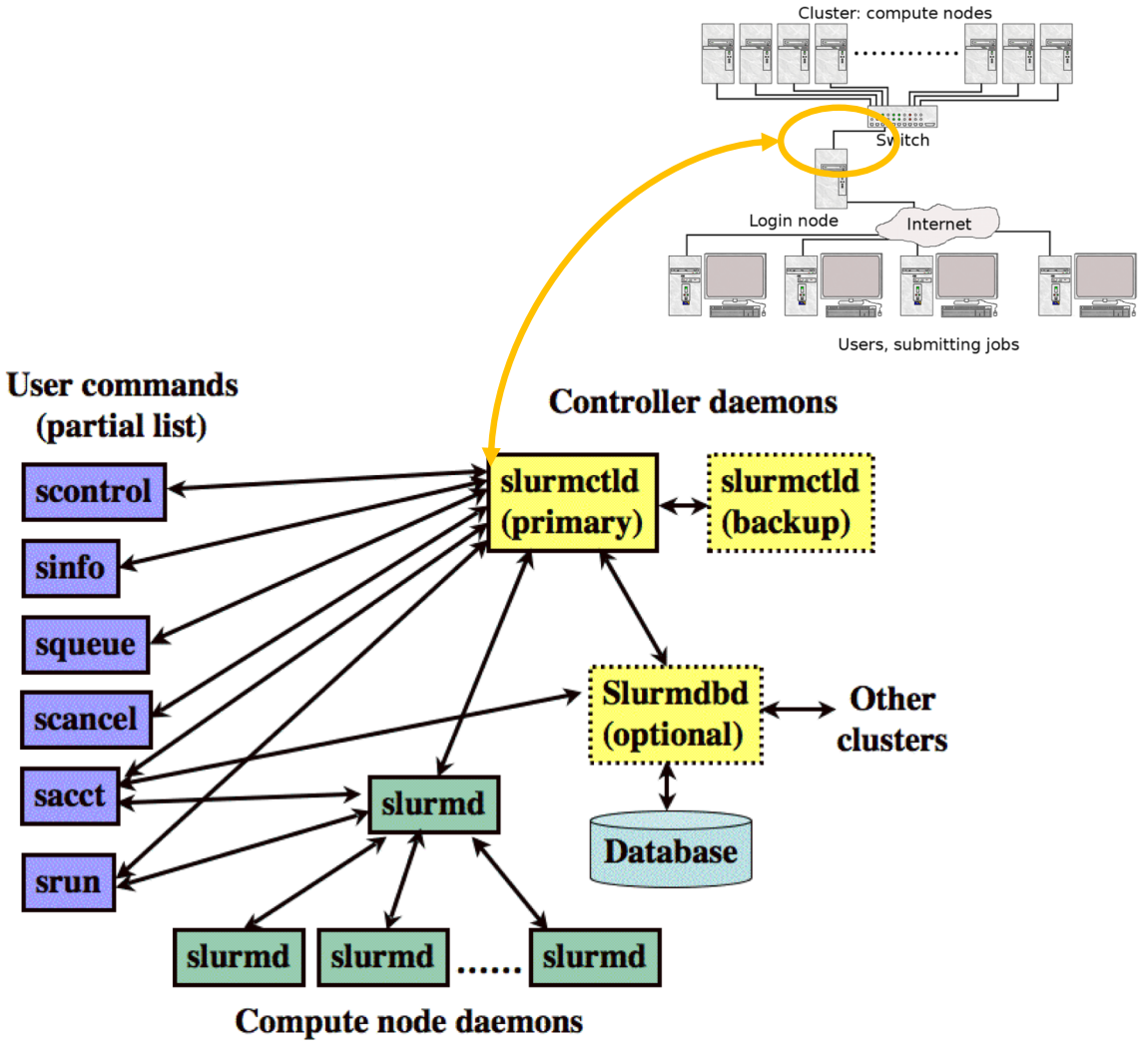
- **Important note: Do not run on the login nodes - even for simple tests.**
- **All job runs must be via the Slurm scheduling infrastructure.**
  - **Interactive Jobs**: Use **srun** command to obtain nodes for 'live' interactive access:

    srun --pty --nodes=1 --ntasks-per-node=24 -p debug -t 00:30:00 --wait 0 /bin/bash
  - **Batch Jobs**: Submit batch scripts from the login nodes. Can choose:
    - Partition (details on upcoming slide)
    - Time limit for the run (maximum of 48 hours)
    - Number of nodes, tasks per node
    - Memory requirements (if any)
    - Job name, output file location
    - Email info, configuration

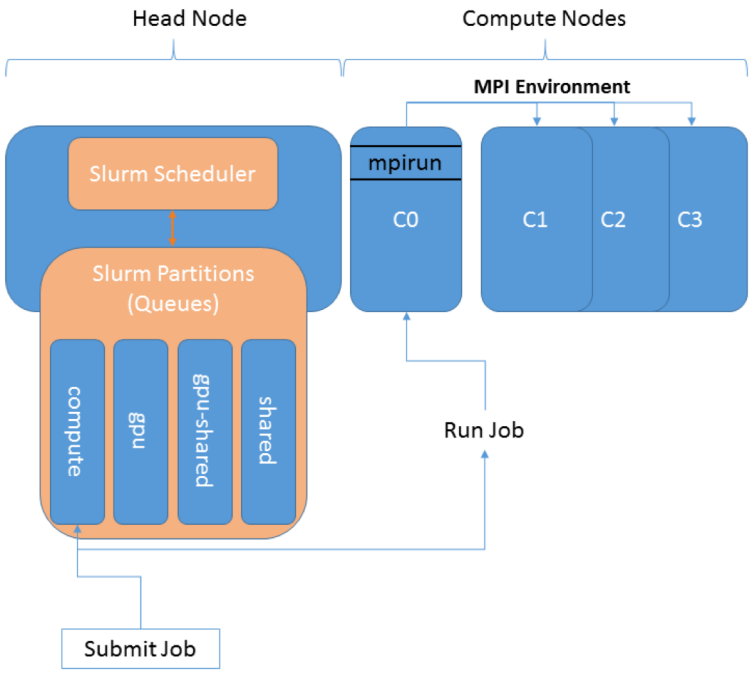
Slurm Resource Manager

Simple Linux Utility for Resource Management

- "Glue" for parallel computer to schedule and execute jobs
- Role: Allocate resources within a cluster
  - Nodes (unique IP address)
  - Interconnect/switches
  - Generic resources (e.g. GPUs)
  - Launch and otherwise manage jobs
- Functionality:
  - Prioritize queue(s) of jobs;
  - decide when and where to start jobs;
  - terminate job when done;
  - Appropriate resources;
  - manage accounts for jobs



Slurm Partitions on Comet



Specified using -p option in b
script. For example:
#SBATCH -p gpu

Queue Name	Max Walltime	Max Nodes	Comments
compute	48 hrs	72	Used for access to regular compute nodes
gpu	48 hrs	4	Used for access to the GPU nodes
gpu-shared	48 hrs	1	Used for shared access to a partial GPU node
shared	48 hrs	1	Single-node jobs using fewer than 24 cores
large-shared	48 hrs	1	Single-node jobs using large memory up to 1.45 TB
debug	30 mins	2	Used for access to debug nodes

Common Slurm Commands

- **Submit jobs using the sbatch command:**

  $ sbatch mycode-slurm.sb

  Submitted batch job 8718049

- **Check job status using the squeue command:**

  $ squeue -u $USER

```
             JOBID PARTITION     NAME     USER      ST       TIME  NODES  NODELIST(REASON)
           8718049   compute      mycode user    PD       0:00      1           (Priority)
```

- **Once the job is running:**

  $ squeue -u $USER

```
             JOBID PARTITION     NAME     USER    ST       TIME  NODES  NODELIST(REASON)
           8718064     debug      mycode user    R        0:02      1        comet-14-01
```

For Fun:

- **Join the UCSD Supercomputing Club:**
  - http://supercomputingclub.ucsd.edu/
  - https://training.sdsc.edu/scc-training-schedule-dev
    - Free pie….
  - Rasberry PI^3 event tomorrow, 4/12/19 @ 3pm
- **Check out the Student Cluster Competition Activity @ SDSC:**
  - https://training.sdsc.edu/scc (kickoff on 4/12/19 @1pm)
    - Working with the new ARM architecture (RISC)
  - Seeking a few grad students interested in mentoring ☺
- **Take a tour of SDSC! SC Club on 4/19/19**

References

- **Comet User Guide**
  - https://www.sdsc.edu/support/user_guides/comet.html#compiling
- **SDSC Training Resources**
  - https://www.sdsc.edu/education_and_training/training.html
  - https://github.com/sdsc-training/webinars
  - Comet shared apps/examples; can be found in
    - /share/apps
- **XSEDE Training Resources**
  - https://www.xsede.org/for-users/training
  - https://cvw.cac.cornell.edu/comet/