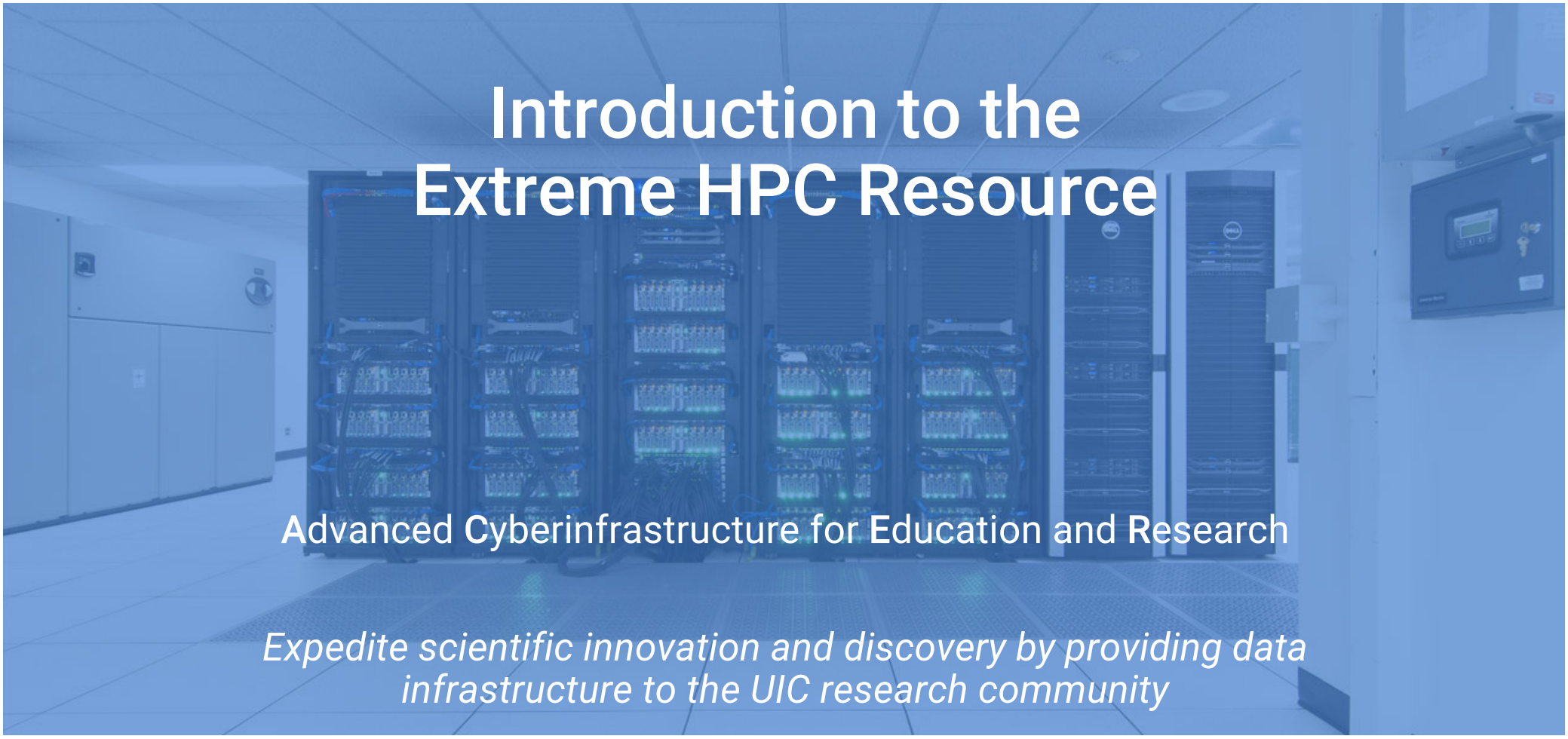


# Introduction to the Extreme HPC Resource

Advanced Cyberinfrastructure for Education and Research

*Expedite scientific innovation and discovery by providing data infrastructure to the UIC research community*

UIC ACCC



UIC ACER ADVANCED CYBERINFRASTRUCTURE FOR EDUCATION AND RESEARCH

# Outline

- Extreme Architecture & Topology
- Requesting an Extreme Account
- Accessing the Cluster via SSH & SCP
- Basic Linux Commands
- Using Cluster Storage
- Resource & Scheduler Concepts
- Job Submission
- Creating a Submit Script
- Monitoring Jobs
- MPI Program Example
- ACER Staff

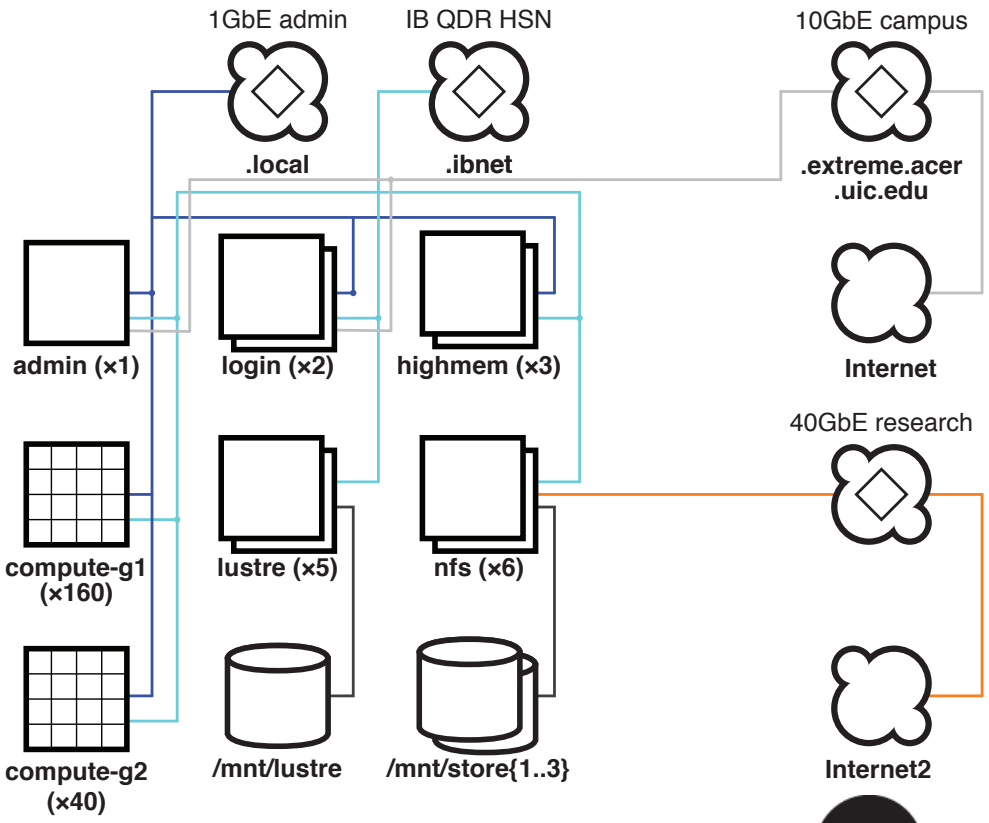
# Extreme Architecture & Topology

- High Performance Computing (HPC) Cluster
  - Dell manufactured
  - Intel x86_64 architecture
  - Compute & high memory nodes (×203)
    - compute-g1 (×160) : 2×8-core 2.6 GHz Intel Xeon E5-2670, 128 GB RAM
    - compute-g2 (×40) : 2×10-core 2.5 GHz Intel Xeon E5-2670 v2 , 128 GB RAM
    - highmem-g1 (×3) : 4×8-core 2.60 GHz Intel Xeon E5-4650L, 1 TB RAM
  - Head nodes (×3)
    - login (×2), admin : 2×8-core 2.6 GHz Intel Xeon E5-2670, 32 GB RAM
- High Speed Network (HSN)
  - Infiniband QDR fabric (40 Gb/s)
  - Fat tree topology with approximately 1:1 blocking factor

# Extreme Architecture & Topology

- Storage
  - Persistent NFS with High Availability (NFS-HA)
    - General purpose storage
    - 3× filesystems, each with 262 TB capacity (formatted)
    - Each filesystem has an active/standby pair of server nodes
    - Mounted on /mnt/store1, /mnt/store2, /mnt/store3
  - Intel Enterprise Edition for Lustre
    - High performance Lustre filesystem
    - 175 TB capacity (formatted)
    - Nodes: 1× management, 2× Object Storage Server (OSS), 2× Metadata Server (MDS)
    - Mounted on /mnt/lustre

# Extreme Architecture & Topology



# Requesting an Extreme Account

- Research faculty, staff, etc.
  - Verify allocated resources exist with PI or equivalent
  - Visit https://acer.uic.edu/computing-resources/big-data/request-access/
  - Select “Extreme/Condo Cluster” and complete form
- Student accounts
  - Sponsored by professor/instructor
  - Account creation will be processed after add/drop deadline

# Accessing the Cluster via SSH & SCP

- Login
  - Username is your UIC NetID
  - Password is usual “ACCC Common Password”
- Using Secure Shell (SSH) on Mac, Linux, Unix:
  - Command requires terminal application window; connect to either:

```
ssh netid@login-1.extreme.acer.uic.edu     # or
ssh netid@login-2.extreme.acer.uic.edu
```

- Using SSH on Windows
  - Requires third party application, such as **PuTTY**
  - PuTTY is available for free from http://www.putty.org/
    - Within PuTTY select SSH radio button under Session
    - Use **login-*n*.extreme.acer.uic.edu** in Host Name field
    - Will be prompted for username and password

# Accessing the Cluster via SSH & SCP

- File transfers can be done via Secure Copy (**SCP**), which uses the same credentials as SSH.
- Using SCP on Mac, Linux, and Unix:
  - `scp -r source_path dest_path`
  - Use `-r` when recursively copying contents of a directory
  - The remote path (either source or destination) is formatted as `netid@login-n.extreme.acer.uic.edu:/path`
- Using SCP on Windows:
  - PuTTY web site provides an analogous **PSCP** application which uses similar syntax to SCP.
  - `pscp.exe` is CLI-based; run within `cmd.exe` utility.

# Accessing the Cluster via SSH & SCP

**Useful SSH Flags:**

- To allow an **X11 GUI application** from within SSH:
  - `ssh -Y netid@login-n.extreme.acer.uic.edu`
  - *The X11 server is already available on Linux and Unix workstations. **XQuartz** is freely available for the Mac; **Xming** is freely available for Windows.*

- Using SSH in **verbose mode** (helpful when debugging):
  - `ssh -v netid@login-n.extreme.acer.uic.edu`

# Basic Linux Commands: Filesystem Navigation

- **`cd`**
  - The **c**hange **d**irectory command is used to navigate the file system:
    - **`cd *path*`**
- **`ls`**
  - The **l**i**s**t directory command is used to gain information on files and subdirectories (folders) in a directory. A good example with useful flags:
    - **`ls -lah`**
    - **`l`** = long (detailed), **`a`** = all (including hidden dot files), **`h`** = human-readable file sizes
- **`pwd`**
  - The **p**rint **w**orking **d**irectory command is used to determine the directory in which you are currently located with respect to the file system hierarchy. This command is useful for determining where you are if you get lost navigating the file system.
- **`mkdir`**
  - By default, this command will **m**a**k**e a new sub**dir**ectory (folder) relative to the directory in which you are currently located. You may however provide an absolute path to where you wish a subdirectory to be made.
    - **`mkdir test1`** `# creates test1 relative to current directory`
    - **`mkdir /export/home/*netid*/test2`** `# creates absolute path`

# Basic Linux Commands: File I/O and Editing

- **`cat`**
  - outputs (and con**cat**enates) the contents of one or more files to standard output (by default the screen)
  - **`cat`** ***`filename`*** *`filename …`*
- **`grep`**
  - Searches input **g**lobally for a **r**egular **e**xpression and **p**rints matching patterns (**regular expressions** are special character strings using a versatile search syntax)
  - **`grep`** ***`regex filename`***
- **`vi`** and **`nano`**
  - text-based file editors
  - **`vi filename`** —or— **`nano filename`**
  - **`nano`** is more intuitive; **`vi`** is more powerful

# Basic Linux Commands: File Permissions

- **`chmod` *`mode filename`*** *`filename …`*
  - **ch**anges **mod**es of **r**ead, **w**rite, and e**x**ecute file permissions, as well as other persistent setuid and ownership settings (“sticky bit”).
  - Permissions are ordered by **u**ser, **g**roup, & **o**ther user access.
  - Current ownership can be determined using the long directory listing (**`ls -l`**); e.g., `-rwxr-x---` indicates the user can read, write, and execute, the group can read and execute, and world has no access (the leading **-** means no persistence flags are set).
  - Mode parameters can be relatively changed ***m±n*** (where ***m*** is one or more of **u**, **g**, or **o**ther; **+** to add or **-** to remove; and ***n*** is one or more of **r**, **w**, or **x**),; e.g., **`chmod ug+x`** ***`filename`*** gives the user and group execute permission.
  - Mode parameters can be absolutely changed using octal-encoded values; e.g., **`chmod 0750`** ***`filename`*** specifies `-rwxr-x---`
  - N.b.: directories must have the execute bit set to be accessible.

# Basic Linux Commands: File Ownership

- **`chown` *`owner filename filename …`***
  - Changes the user (UIC NetID) owning the specified file(s) and/or directory(ies); wildcards such as `*` can be used (with caution).
  - ***`owner`*** can be the user only or user`:`group (colon-separated).
- **`chgrp` *`group filename filename …`***
  - Changes just the group ownership of a file or a directory.
  - On Extreme, the group is either "**`domain users`**" or a group name provided by support staff.

# Basic Linux Commands: Software & Modules

- **`which` *`executable`***
  - This command shows the default path to the specified ***executable***. Extreme provides multiple versions of software packages through modules; the output of this command may change depending on the module(s) loaded. When testing interactively, use this command to verify you are running the version you want.
- **`module` *`parameters …`***
  - The **`module`** command is used to manage software packages on the cluster.
- **`module avail`**
  - Lists all the software modules available on the cluster.
- **`module load` *`modulename`***
  - This command loads the software package into your path. Keep in mind you must use this command in your submit scripts in order to call software packages.
- **`module list`**
  - This command displays active modules listed in the order they were loaded.
- **`module unload` *`modulename`***
  - This command removes the specified software package from your path.

# Basic Linux Commands: MPI Modules Example

There are multiple implementations of the Message Passing Interface (MPI), a standard parallel computing framework. The environment has to be consistent between the building and execution of an MPI application. In this example, note how the path to **`mpirun`** (an MPI execution harness) changes depending on the module loaded.

```
$ which mpirun
/opt/openmpi/bin/mpirun

$ module load compilers/intel
$ which mpirun
/export/share/compilers/intel/impi/4.1.1.036/intel64/bin/mpirun

$ module load tools/mpich-3.0.4-icc
$ which mpirun
/export/share/tools/mpich-3.0.4-icc/bin/mpirun
```

# Storage on the Cluster

- Home directories are on **Persistent NFS Storage**
  **`/export/home/`*`netid`***
- … as are **lab shares**; contact ACER support for further details.

- Fast temporary **scratch** filesystem is on **Lustre**
  **`/mnt/lustre/`*`netid`***

# Basic Job Scripting: Example Submit Script

```
#!/bin/bash
#PBS -l mem=20gb
#PBS -l walltime=20:00:00
#PBS -l nodes=1:ppn=8
#PBS -j oe
#PBS –m abe
#PBS -M email_address
#PBS -N jobname
#PBS -d /export/home/netid/work_dir

sleep 30
```

# Basic Job Scripting: #PBS Headers

**`#!/bin/bash`**

- Always specify the shell that the job script uses.

**`#PBS -l mem=20gb`**

- This optional line tells the cluster how much memory your job intends to use and ensures that there is enough memory on the assigned nodes when you submit.

**`#PBS -l walltime=20:00:00`**

- This line tells the cluster how  long the job should run (HH:MM:SS).

**`#PBS -l nodes=1:ppn=8`**

- This line specifies the number of nodes (physical compute nodes) and then the number of cores (processors) the job will need to run. The assigned resource is a product of these values.

# Basic Job Scripting: #PBS Headers

**`#PBS -j oe`**

- Allows the user to join and otherwise manipulate the standard **o**utput and standard **e**rror into a single file.

**`#PBS -m abe`**

- Requests a status email when a job **b**egins, **e**nds, or **a**borts.

**`#PBS -M email_address`**

- Provides an email address in conjuction with the status email flag above.

**`#PBS -N jobname`**

- Names the job with a custom label to allow a user to make it easily identifiable in the list of jobs (e.g., provided by **`qstat`**).

**`#PBS -d /export/home/netid/work_dir`**

- Specifies the initial working directory for your job, generally where your job's data and/or program reside.

# Job Submission

- Submit the job using a submit script requesting **one node**:
  **`qsub -l nodes=1 submit_script`**
- Now request all **20** cores (**p**rocessors **p**er **n**ode) on **two nodes**:
  **`qsub -l nodes=2:ppn=20 submit_script`**

- The **`-l`** (lowercase L) precedes a comma-separated resource **l**ist. **`qsub`** can accept multiple "**`-l …`**" argument pairs. You may specify these parameters at the command line or in the script.

- Request an **i**nteractive job with **`-I`** (capital i):
  **`qsub -I -l nodes=2:ppn=16`**

# Job Monitoring

- **showq**
  - Displays information on all jobs that are active, queued, or blocked.
  - To only display only your jobs, pipe the output through **grep**:

    **showq | grep *netid***
- **qstat**
  - An alternative queue monitoring application.
- **checkjob**
  - Good for gaining detailed information on an individual job or determine why it failed to run.

    **checkjob -v *jobid***
    - Provides detailed information on the specified job and any error messages.

    **checkjob -v -v *jobid***
    - Provides not only detailed information on the specified job and error messages, but displays the output of your submitted script.
- **qdel**
  - To use this command to cancel one of your jobs , use:

    **qdel *jobid***

# Basic Job Scripting: Invalid Resource Requests

~~`#PBS -l mem=256gb`~~

If you ask for a node with more than 128GB of memory the job will never run as each node only has 128 GB of RAM.

~~`#PBS -l walltime=720:00:00`~~

The maximum walltime is 240 hours (or 10 days). If you submit a job with a walltime longer than the maximum, the the job will not run.

~~`#PBS -l nodes=1:ppn=128`~~

The **ppn** value cannot exceed the per-compute core count. Extreme has 16-core "Generation 1" and 20-core "Generation 2" compute nodes. The default **batch** queue uses G1 nodes; **edu_shared** uses G2; users should inquire about the node generation for other queues. A valid equivalent to a 1 × 128 configuration would be **nodes=8:ppn=16**.

# MPI Sample Program on Extreme

```
#include <mpi.h>
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv) {

        int world_size;
        int rank;
        char hostname[256];
        char processor_name[MPI_MAX_PROCESSOR_NAME];
        int name_len;

        MPI_Init(&argc, &argv); // Initialize the MPI environment
        MPI_Comm_size(MPI_COMM_WORLD, &world_size);  // get the total number of processes
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);        // get the processor rank number
        MPI_Get_processor_name(processor_name, &name_len);  // get the processor name

        gethostname(hostname, 255);                  // non-MPI function to get the hostname
        printf("Hello world!  I am process number: %d from processor %s on host %s out of %d processors\n", rank, processor_name, hostname, world_size);

        MPI_Finalize();

        return 0;
}
```

# Sample Script to Run MPI Program

```
#!/bin/bash
#PBS -l nodes=2:ppn=20,walltime=1:00
#PBS -N MPIsample
#PBS -q edu_shared
#PBS -m abe
#PBS -M netid@uic.edu
#PBS -e mpitest.err
#PBS -o mpitest.out
#PBS -d /export/home/netid/MPIsample

module load tools/mpich2-1.5-gcc

mpirun -machinefile $PBS_NODEFILE -np $PBS_NP ./mpitest
```

# Sample Script to Run MPI Program

0. Copy (**R**ecursively) the directory containing the sample code and job script to your home directory.

```
cp -R /export/share/classes/cs-ece566/MPIsample ~
```

1. Load MPICH2 module before compiling the program.

```
module load tools/mpich2-1.5-gcc
```

2. Compile the program

```
cd ~/MPIsample
mpicc -o mpitest mpitest.c
```

3. Modify job script email address & working directory and the job script to the queue:

```
qsub mpitest.pbs
```

# Notes on MPI Versions

- There are different implementations of the Message Passing Interface; Extreme has various implementation and releases:
  - OpenMPI — *Extreme default*
  - MPICH — *UIC classes have frequently used MPICH2*
  - Intel MPI — *commercial implementation with Intel x86_64 architecture optimizations*
- Implementation releases (versions) are often *backwards compatible;* e.g., MPICH3 **`mpirun`** can execute code compiled with MPICH2 **`mpicc`**
- The different implementations' tools are usually ***not*** compatible with one another; e.g., the Intel MPI **`mpirun`** cannot execute code compiled with OpenMPI **`mpicc`**

# About ACER

The **Advanced Cyberinfrastructure for Education and Research** is a division of the Academic Computing and Communications Center at the University of Illinois at Chicago.

**ACER Staff**

- Himanshu Sharma, *Director of ACER*
- Greg Cross, *Technical Lead*
- Jay Moreau, *HPC Systems Administrator*
- Balpreet Singh, *Student Support Staff*
- Sanjay Andonissamy, *Student Support Staff*

For more information, visit http://www.acer.uic.edu/