Introduction to Running Computations on the High Performance Clusters at the Center for Computational Research

L. Shawn Matott

Center for Computational Research
University at Buffalo, SUNY
701 Ellicott St
Buffalo, NY 14203

ccr-help at ccr.buffalo.edu

Fall 2014

Part One

CCR Resources

The Center for Computational Research provides high performance computing resources to the University at Buffalo.

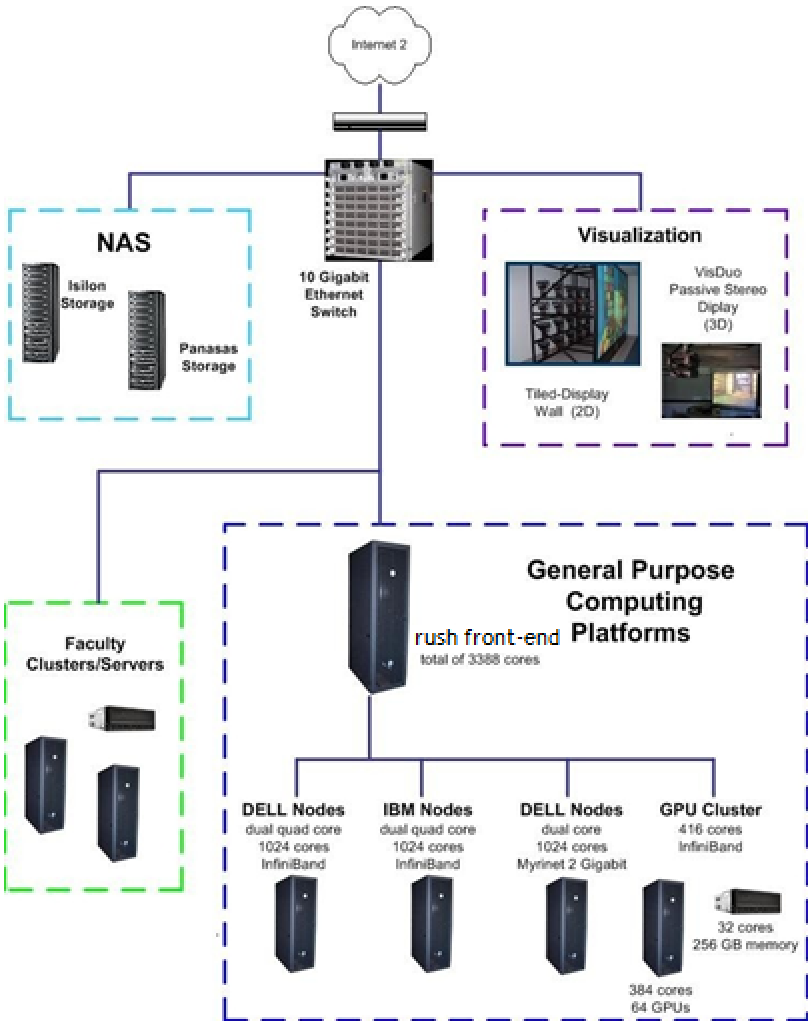
- ✦Supporting faculty research and classroom education, as well as local business and University collaborations.
- ✦High performance and high through-put cluster.
- ✦High performance remote visualization.

CCR Cluster

- Over 8,000 cores
- Mix of 8, 12, 16 and 32 compute nodes.
  - A compute node is a linux machine with memory, disks, and cores (cpus), as well as both ethernet and Infiniband network connections.
- Infiniband and ethernet networks
- Panasas and Isilon storage
- Linux - CentOS release 6.5

More information on the CCR cluster specifications.

CCR Cluster



Coming soon: an Intel Xeon-Phi node

CCR Cluster Compute Nodes

- 372 **12-core** nodes with 48GB and Qlogic IB
- 256 **8-core** nodes with 24GB and Mellanox IB
- 32 **16-core** nodes with 128GB and Mellanox IB
- 16 **32-core** nodes with 256GB and Qlogic IB
- 2 **32-core** nodes with 512GB and Qlogic IB
- 32 **12-core** nodes, each w/ 2 Nvidia Fermi **GPUs**, 48GB and Mellanox IB
- 2 **12-core** remote visualization nodes with 256GB, 2 Nvidia Fermi GPUS
- More on Compute Nodes

CCR Remote Visualization

- The remote visualization machines have Nvidia Tesla GPUs, 12 cores and 256 GB of memory.
- The visualization is enhanced with the use of NICE Remote Visual software.
  - NICE provides a desktop to the use.
  - Several software packages can make use of cluster compute nodes.
- More information on using CCR Remote Visualization.

Storage

- Home directories are **/user/username** on the front-end login machine and all compute nodes.
  - The default size of a home directory is **2GB**.
  - There is a daily back-up of home directories.
  - The iquota command shows the usage and quota.
- All users may use the **/panasas/scratch** file system. This file system is available on all compute nodes and the front-end.
  - There is no usage quota on /panasas/scratch
  - There is no backup of files on /panasas/scratch
  - Files are periodically deleted

Storage

- All compute nodes have a local /scratch.
  - Computations can use the local scratch space.
- UB faculty can request a **/projects** directory for the research group.
- The /projects directories are accessible from all compute nodes and the front-end machine.
- The quota ranges from 200GB to 500GB.
  - There is no charge.
- Storage space greater than 500GB has a one time fee. Contact ccr-help for more information.

Login to Front-end Machine

- The front-end machine to the cluster is **rush.ccr.buffalo.edu**
  - 32-core node with 256GB of memory.
- Accessible from **UB network only**. Use the Cisco AnyConnect Client to connect to the UB network from off campus. The UBit software webpage has download links and installation instructions.
- Users on campus should setup UB-Secure on their laptops for connecting through wireless.
- Only secure protocols, such as ssh and sftp, are permitted to access the front-end.

Login to Front-end Machine

- Only small test computations can run on the front-end machine.
  - **CPU time limit of 15 minutes.** Processes that exceed the time limit are automatically terminated.
  - Be sure to monitor the processes using the top command.
- In general, computations run on the cluster as jobs submitted to the SLURM scheduler.
- The compute nodes and storage cannot be directly accessed from outside the cluster.

Login to Front-end machine

- Login from Linux or Mac
- ssh -X rush.ccr.buffalo.edu
- ssh -X UBITusername@rush.ccr.buffalo.edu
  - The -X flag enables a graphical display. This is optional.

- Windows users must install X-Win32 or PuTTY for the secure login, which can be downloaded from the UBit software webpage.
- The **X-Win32** program allows for a graphical display.

Accessing the Cluster

Details for X-Win32 setup

- ✦Select Manual in the configuration window.
- ✦Select ssh
- ✦Connection name: rush
- ✦Host: rush.ccr.buffalo.edu
- ✦Login: your UBITusername
- ✦Command: /usr/bin/xterm -ls
- ✦Password: CCR password
- ✦Note: Be sure to run only one X-Win32.

Accessing the Cluster

- sftp and scp are command line transfer commands. See the CCR user manual for more information.
- There are several graphical file transfer applications. Be sure to specify secure file transfer (sftp). Most can be downloaded from the UBit software webpage.
- Filezilla is graphical file transfer program, which is available for Windows, Linux and MAC computers.
- WinSCP for Windows.
- Cyberduck for MACs.

Get Help - Change Password

- Send email to ccr-help@buffalo.edu for assistance.
- Use the web form to request assistance.

- Use the MyCCR web interface to change or reset the CCR password.
- Login with UBitusername and Ubit password.
- MyCCR

Command Line Environment

- The rush cluster is a command line UNIX environment.
- The user's login executes a shell. This is a process that accepts and executes the commands typed by the user.
- By default the shell is set to **bash**.
- The **.bashrc** file is executed on login.
  - This script resides in the user's home directory.
  - Variables and paths can be set in the .bashrc file.
  - It is also executed when a job starts on a compute node.

Command Line Environment

- List contents of current directory: **ls**
- Long listing: **ls -l**
- Long list and hidden files: **ls -la**
- Reverse time order: **ls -latr**
- Show current directory pathname: **pwd**
- Create a directory: **mkdir** directoryname
- Change to given pathname: **cd** /pathname
- Change to one directory above: **cd ..**
- Change to two directories above: **cd ../..**
- Change to home directory: **cd**

Command Line Environment

- Remove a file: **rm filename**
- Remove a directory: **rm -R directoryname**
  - If the directory is empty: rmdir dirname
- Copy a file: **cp oldfile newfile**
- Display file to the screen: **cat filename**
- Display with page breaks: **more filename**
  - Press space bar to page through the file.
  - Press enter to advance one line.
  - Forward direction only.
- "Control c" to quit.

Command Line Environment

- All directories and files have permissions.
- Users can set the permissions to allow or deny access to their files and directories.
- Permissions are set for the user (**u**), the group (**g**) and everyone else (**o**).
- The permissions are read (**r**), write (**w**) and execute (**x**).
- **Read** (r) permission allows viewing and copying.
- **Write** (w) permission allows changing and deleting.

Command Line Environment

- **Execute** (x) permission allows execution of a file and access to a directory.
- Show permissions with “ls -l”.
  - This will also show the user and group associated with the file or directory.
- drwxrwxrwx
- -rwxrwxrwx
  - d indicates a directory
  - -**rwxrwxrwx** user permissions
  - **-rwxrwxrwx** group permissions
  - **-rwxrwxrwx** other (world) permissions

Command Line Environment

- Change permissions with the chmod command.
- **chmod ugo+rwx** filename
- **chmod ugo-rwx** filename
- ugo is user, group and other
- rwx is read, write and execute
- + add a permission
- - removes a permission
- Adding write permission for group: **chmod g+w filename**

Command Line Environment

- Removing all permissions for group and other on a directory and its contents: **chmod -R go-rwx dirname**
- The grep command is used to search for a pattern: **grep pattern file**
- Show the disk space: **df -h**
- Show the estimated disk space usage: **du -s -h`ls` | sort -rn | more**
- The pipe symbol **|** is used to indicate that the output of the first command should be used as input to the second.

Command Line Environment

- There are several editors available: emacs, nano and vi
  - emacs can provide a graphical interface
- Files edited on Windows machines can contain hidden characters, which may cause runtime problems.
- Use the file command to show the type of file: **file filename**
- Use the dos2unix command to remove any hidden characters: **dos2unix filename**

Command Line Environment

- The asterisk (*) is a wild card for many UNIX commands.
- List all C files: **ls ***.c
- Show the type of file: **file filename**
- Manual pages are available for UNIX commands: **man ls**
- More information on UNIX commands: [CCR Reference Card]

Command Line Environment

- Modules are used to set paths and variables for applications installed on the cluster.
- List all available modules: **module avail**
- List modules currently loaded: **module list**
- Show what a module will do: **module show module-name**
- Load a module: **module load module-name**
- Unload a module: **module unload module-name**
- Example: module avail intel
  - List the Intel compilers

Compilers

- The GNU compilers and libraries are part of the Operating System and are in the default path.
- C compiler: gcc
- C++ compiler: g++
- Fortran compilers: g77, gfortran
- gfortran is a Fortran 95 compiler.
- Example:
  - gcc -o hello-gnu hello.c
  - ./hello-gnu

Compilers

- The Intel compilers can take advantage of the processor and core architecture and features.
- Load the module for the intel compiler: module load intel
- C compiler: icc
- C++ compiler: icpc
- Fortran compiler: ifort
- Example:
  - icc -o hello-intel hello.c
  - ./hello-intel

Hello World

- Hello World C program:

```
#include <stdio.h>

 int main(void)
 {
   printf("Hello, world!\n");
   return 0;
 }
```

Hello World

- Hello World C++ program:

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello, World!" << endl;
   return 0;
}
```

Hello World

- Hello World FORTRAN program:

```
program hello
   write(*,*) 'Hello, World!'
end program hello
```

Part Two

What is SLURM?

- SLURM is an acronym for Simple Linux Utility for Resource Management.
- SLURM is a workload manager that provides a framework for job queues, allocation of compute nodes, and the start and execution of jobs.
- SLURM is a comprehensive resource manager.
  - Individual node resources, such as GPUs or number of threads on a core, can be scheduled with SLURM.

SLURM Partitions

- The production partition for the CCR cluster is **general-compute**. This is the default partition.
- The **debug** partition has 7 nodes. Jobs can run for a maximum of 1 hour.
  - 4 8-core nodes
  - 2 12-core nodes
  - 1 16-core node with 2 GPUs.
- The **gpu**, **largemem**, **supporters** and **viz** partitions have specific requirements.

SLURM Partitions

- These partitions have a higher priority than the general-compute partition.
- Nodes in these partitions are also in the general-compute partition.
- The **gpu** and **largemem** partitions are available to all users.
- The guideline for submitting a job to the **gpu** partition is that the computation must make use of a GPU.

SLURM Partitions

- The guideline for submitting a job to the **largemem** partition is that the computation must require more than 100GB of memory.
- The **supporters** partition is a higher priority partition for research groups that have made contributions to CCR.
- The **viz** partition is dedicated to the remote visualization compute nodes.

SLURM Commands

- **squeue** – shows the status of jobs.
- **sbatch** – submits a script job.
- **salloc** – submits an interactive job.
- **srun** – runs a command across nodes.
- **scancel** – cancels a running or pending job.
- **sinfo** – provides information on partitions and nodes.
- **sview** – graphical interface to view job, node and partition information.
- **slurmjobvis** –graphical job monitoring tool.

squeue example

squeue -u cdc

JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)

4832 general-c hello_te cdc R 0:20 2 f16n[10-11]

- Job status:
  - **R** – job is running.
  - **PD** – job is waiting for resource.
    - Reasons are usually (Resources) or (Priority).
  - Others commons reasons are CA (cancelled) and CD (completed).

sinfo example

sinfo -p general-compute

PARTITION AVAIL TIMELIMIT NODES STATE NODELIST

general-comput* up 3-00:00:00 264 idle d07n07s[01-02],d07n08s[01-02], …

- Node states:
  - **alloc** – all cores are in use.
  - **mix** – some cores are available.
  - **idle** – node is free. All cores are available.
  - **down**- node is down.
  - **drained** – node is offline.

sinfo example

More detailed sinfo query:

sinfo --exact --partition=general-compute --format="%15P %5a %10A %.4c %6m %6G %16f %t %N" | more

PARTITION AVAIL NODES(A/I) CPUS MEMORY GRES FEATURES STATE NODELIST

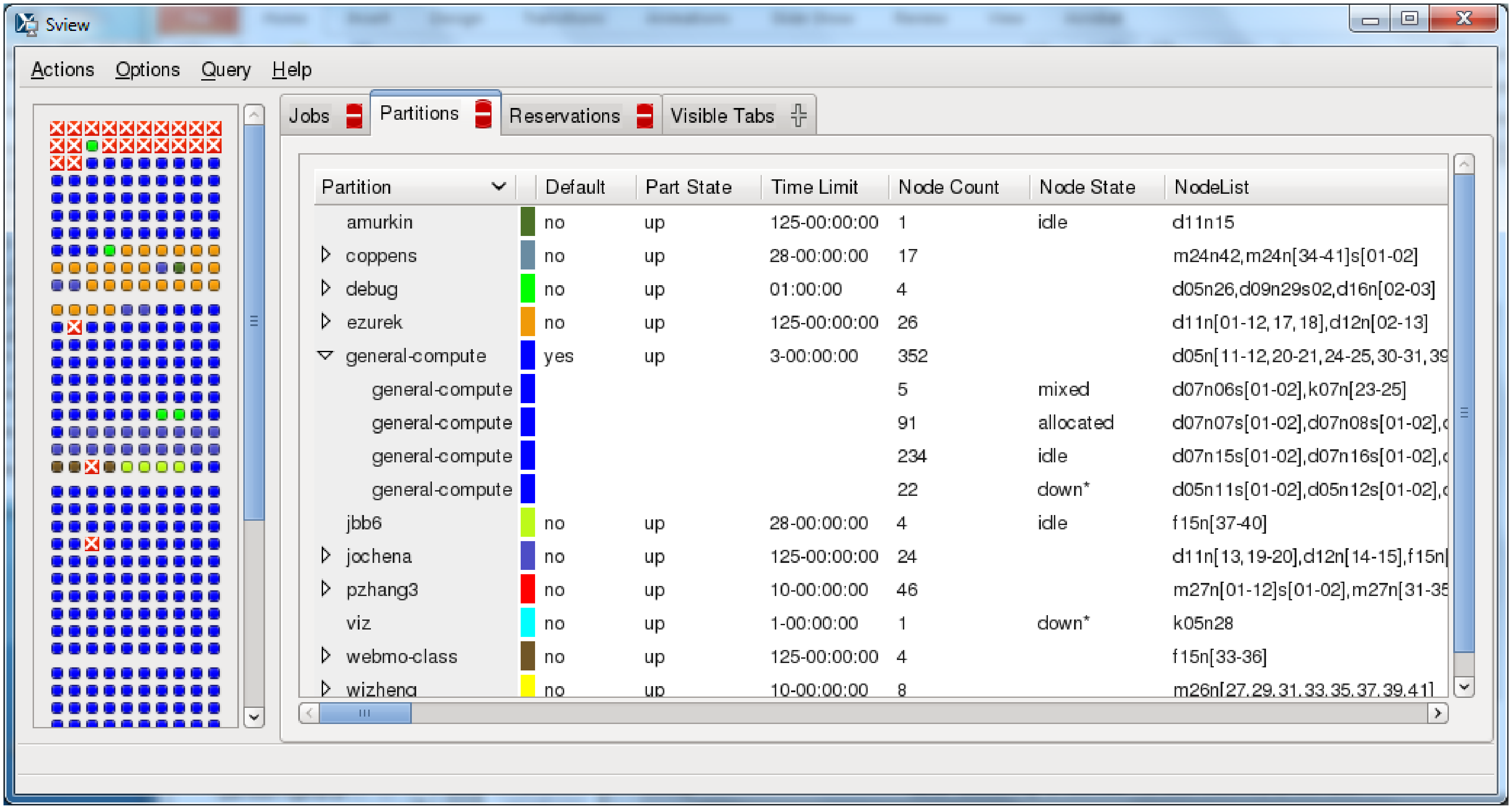
general-comput* up 0/0 12 48000 gpu:2 IB,CPU-X5650 down* d05n11s[01-02],d05n12s[0102],d05n20s[01-02],d05n21s[01-02],d05n24s[01-02],d05n25s[01-02],d05n30s[01-02],d05n31s[01-02],d05n39s[01-02],d05n40s[01-02]

...

general-comput* up 0/151 12 48000 (null) IB,CPU-E5645 idle k13n17

s[01-02],k13n18s[01-02],k13n19s[01-02],k13n23s[01-02],k13n24s[01-02],

sview example



STUBL

- STUBL = SLURM Tools and UBiLities
- UB CCR Customizations of SLURM commands
  - A bit more user-friendly
  - Examples:
    - snodes – what nodes are available?
    - fisbatch – interactive job
    - sranks – what is my jobs priority?
    - sjeff, sueff – How efficient are my jobs?
    - stimes – When will my job start?
  - slurmhelp – list all SLURM and STUBL commands
  - Type any command followed by “--help” for usage info.

Commonly used SLURM variables

- $SLURM_JOBID
- SLURM_JOB_NODELIST
  - Node list in SLURM format; for example f16n[04,06].
- $SLURM_NNODES
  - Number of nodes
- $SLURMTMPDIR
  - /scratch/jobid
  - local to the compute node
- $SLURM_SUBMIT_DIR
  - Directory from which the job was submitted
- **NOTE!** Jobs start in the $SLURM_SUBMIT_DIR.

sbatch example

Submit a job: **sbatch slurmHelloWorld**

Here is the SLURM script:

```
#!/bin/sh
##SBATCH --partition=debug
#SBATCH --time=00:15:00
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
##SBATCH --mem=24000
#SBATCH --job-name="hello_test“
#SBATCH  --output=test.out
#SBATCH --mail-user=usename@buffalo.edu
#SBATCH --mail-type=ALL
echo "SLURM_JOBID="$SLURM_JOBID
echo “SLURM_JOB_NODELIST"=$SLURM_JOB_NODELIST
echo "SLURM_NNODES"=$SLURM_NNODES
echo "SLURMTMPDIR="$SLURMTMPDIR
```

sbatch example

```
echo "working directory = “$SLURM_SUBMIT_DIR
module load intel/14.0
module load intel-mpi/4.1.3
module list
ulimit -s unlimited
#
#export I_MPI_DEBUG=4
#NPROCS=`srun --nodes=${SLURM_NNODES} bash -c 'hostname' |wc -l`
#echo "NPROCS="$NPROCS
echo "Launch helloworld with srun"
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
srun ./helloworld

#
echo "All Done!"
```

sbatch example - annotated

#!/bin/sh
#SBATCH is a directive to SLURM
is a comment
Specify the partition; default is general-compute
##SBATCH --partition=debug
Request 15 minutes
#SBATCH --time=00:15:00
Request compute nodes
#SBATCH --nodes=2
Specify 8 cores on a node
#SBATCH —ntasks-per-node=8
Specify memory limit of 24000MB per node
##SBATCH —mem=24000

sbatch example - annotated

Specify name of job

#SBATCH —job-name="hello_test“

Specify name of output file for stdout and stderr

#SBATCH --output=test.out

Specify email address

#SBATCH --mail-user=usename@buffalo.edu

Send email on start and end of job

#SBATCH --mail-type=ALL

echo "SLURM_JOBID="$SLURM_JOBID

echo “SLURM_JOB_NODELIST"=$SLURM_JOB_NODELIST

echo "SLURM_NNODES"=$SLURM_NNODES

echo "SLURMTMPDIR="$SLURMTMPDIR

$SLURMTMPDIR is a directory created in /scratch on the compute nodes for this job

sbatch example - annotated

echo "working directory = “$SLURM_SUBMIT_DIR

$SLURM_SUBMIT_DIR is the directory from which the job was submitted

The job will always start in the $SLURM_SUBMIT_DIR

Load Intel Compiler and Intel-MPI

module load intel/14.0

module load intel-mpi/4.1.3

module list

Unlimit the stack

ulimit -s unlimited

#

Set debug for MPI if needed; maximum level 99

#export I_MPI_DEBUG=4

sbatch example - annotated

Count number of processors in job; not used by srun, however issuing srun forces SLURM to run the prologue script to setup the nodes for the job

```
NPROCS=`srun --nodes=${SLURM_NNODES} bash -c 'hostname' |wc -l`
#echo "NPROCS="$NPROCS
echo "Launch helloworld with srun”
```

Must export PMI library for srun task launcher; comment out if using mpirun or mpiexec

```
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
```

Run the code

```
srun ./helloworld

#
echo "All Done!"
```

Hello World MPI Code

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

int main( argc, argv )
   int  argc;
   char **argv;
{
   int rank, size;
   int len;
   char procname[MPI_MAX_PROCESSOR_NAME];
   MPI_Init( &argc, &argv );
   MPI_Comm_size( MPI_COMM_WORLD, &size );
   MPI_Comm_rank( MPI_COMM_WORLD, &rank );
   MPI_Get_processor_name(procname,&len);
   printf( "Hello world from process %d of %d on node  %s\n", rank, size, procname);
   MPI_Finalize();
   return 0;
}
```

Compiling MPI Code

- Load the Intel-MPI module.
  - module load intel-mpi/4.1.3
- Compiling Hello World with GNU C:
  - mpicc -o helloworld helloworld.c
- Compiling with Intel C:
  - module load intel/14.0
  - mpiicc -o helloworld helloworld.c

Submit Job

- Run on rush front-end:

```
[cdc@rush:~]$ mpirun -np 2 ./helloworld
Hello world from process 0 of 2 on node  k07n14
Hello world from process 1 of 2 on node  k07n14
[cdc@rush:~]$ slurmHelloWorld
```

- Submit a job: sbatch slurmHelloWorld
- Check status of job with squeue
- Once the job starts running:
  - use slurmjobvis to monitor the job
  - login to the compute node with ssh and run top

Task Launching

- Use mpirun when testing on the rush front-end.
- srun will execute a command across nodes.
  - Typically, this is the best choice for launching a parallel computation on more than 1 compute node.
- Use srun when submitting a SLURM job (e.g. from within an sbatch script).
- Intel-MPI mpirun and mpiexec are SLURM aware, however srun is the most robust task launcher

Part Three

Node Sharing

- Compute nodes are shared among different jobs and users.
- Tasks are limited to the number of cores and memory specified.
- The integration of CPUSETS and SLURM makes this possible.
  - CPUSET is a Linux kernel level mechanism that can be used to control access to individual cores.
- The default memory limit per core is 3GB.
  - Use the --mem flag to request more memory for a job.

Node Sharing

- --mem=24000
  - Requests 24GB per node.
- --mem-per-core=16000
  - Requests 16GB on a core; use for serial job.
- Jobs exceeding the memory limit will be terminated by the resource manager.
- Check the General Compute Cluster webpage for node memory and core details.
- The --exclusive flag will request the nodes as dedicated. The nodes will not be shared.

Interactive Job

- The salloc command requests the nodes.
- Once the nodes have been allocated to the job, then the user can login to the compute node.
  - The user is **not** logged into the compute node when the job starts.
- Typically, srun is used to execute commands on the allocated nodes.
- ssh can only be used to login to nodes assigned to a given user’s job.

Interactive Job

- The **fisbatch** script can be used to submit and run an interactive job.
- fisbatch --partition=debug --time=00:15:00 --nodes=1 --ntasks-per-node=8
  - Once the job starts, you will be automatically logged into the node.
  - Failures or errors can cause the job to not launch properly, but still be listed in squeue. In that case, use scancel jobid to remove the job.
- The **salloc** command can also be used to submit an interactive job.

Example of an Interactive Job

[cdc@rush:~]$ **salloc --partition=general-compute --nodes=1 --time=01:00:00 --exclusive**

salloc: Granted job allocation 54124

[cdc@rush:~]$ **export | grep SLURM**

declare -x SLURM_JOBID="54124"
declare -x SLURM_JOB_CPUS_PER_NODE="8"
declare -x SLURM_JOB_ID="54124"
declare -x SLURM_JOB_NODELIST="d07n35s01"
declare -x SLURM_JOB_NUM_NODES="1"
declare -x SLURM_NNODES="1"
declare -x SLURM_NODELIST="d07n35s01“
…

[cdc@rush:~]$ exit

exit
salloc: Relinquishing job allocation 54124
salloc: Job allocation 54124 has been revoked.
[cdc@rush:~]$

Note: salloc does not login to compute node!

Example of an Interactive Job

[cdc@rush ~]$ salloc --partition=general-compute --nodes=1 --time=01:00:00 --exclusive &

[1] 14269

[cdc@rush ~]$ salloc: Granted job allocation 4716

[cdc@rush ~]$

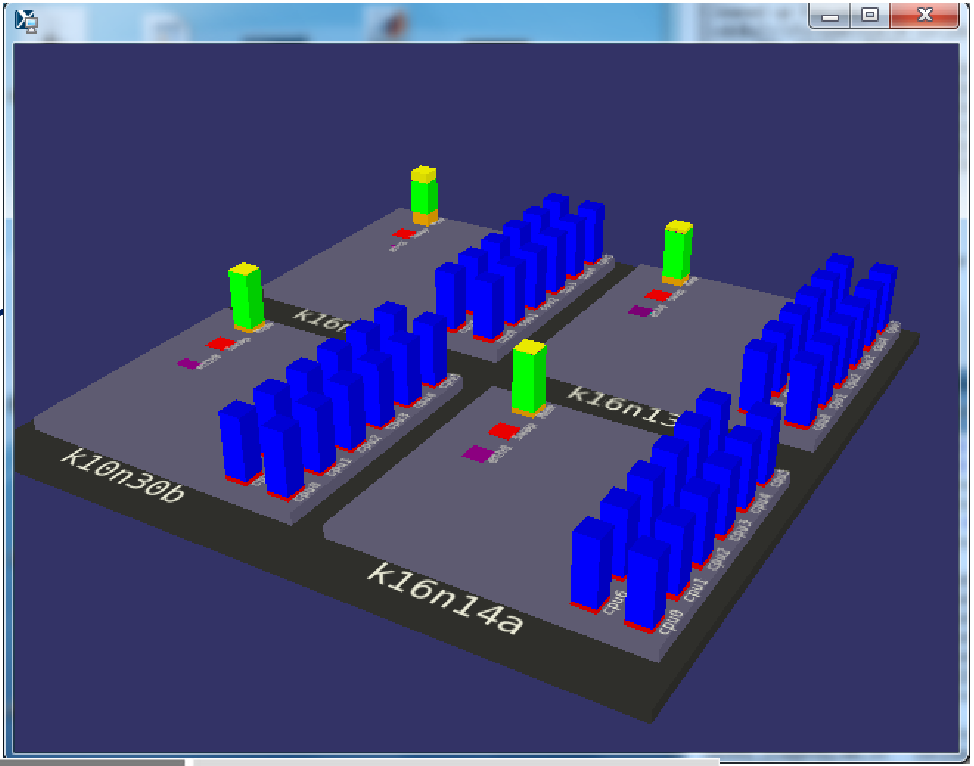
Note!

Placing the salloc in the background allows the allocation to persist.

The user is **not** logged into the compute node when the job starts.

Job monitoring

- The NEW **slurmjobvis** is a graphical display of the activity on the node. CPU, memory, network, as well as GPU utilization are displayed.
- This an improved version of ccrjobvis.
- User can also login using ssh to the compute nodes in the job.



More Information and Help

- [CCR SLURM] web page
  - Most update information for running on the cluster.
  - Sample scripts, code, READMEs, pdfs and extensive instructions.
- More sample SLURM scripts can be found in the /util/slurm-scripts directory on rush.
- Compute Cluster web page
- Remote Visualization web page
- Users can get assistance by sending an email to ccr-help@ccr.buffalo.edu.

Part 4

- The following CCR reference card gives a handy review of important and useful commands.