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

Cynthia Cornelius

Center for Computational Research
University at Buffalo, SUNY
701 Ellicott St
Buffalo, NY 14203
Phone: 716-881-8959
Office: B1-109
cdc at ccr.buffalo.edu

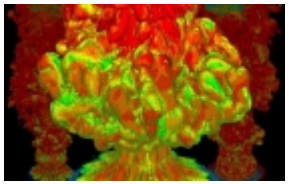
January 2013

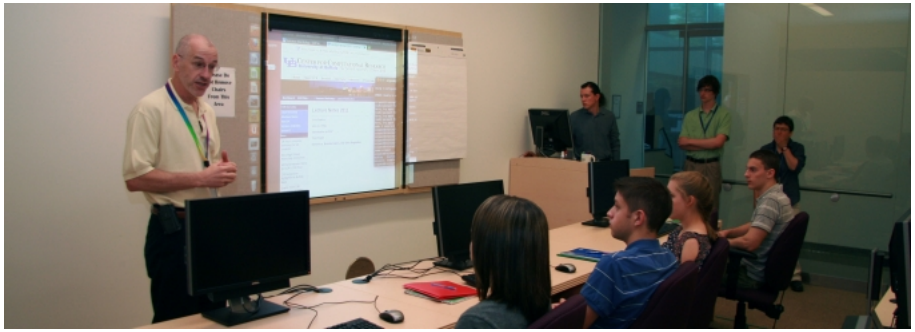
CCR

The Center for Computational Research is the supercomputing facility for the University at Buffalo.

CCR's mission is three-fold:

- Research
- Education
- Outreach





Cluster Resources

- 256 computers 8-cores with 24GB memory
- 372 computers 12-cores with 48GB memory
- 16 computers 32-cores with 256GB memory
- 2 computers 32-cores with 512GB memory

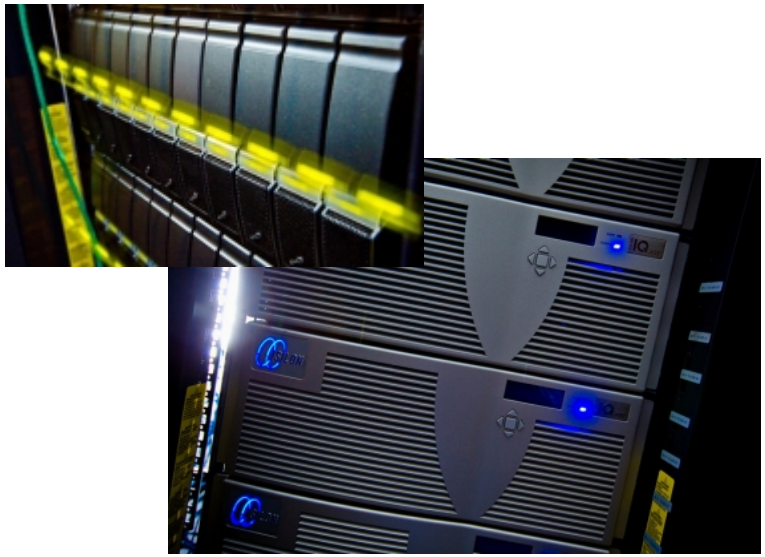
- 32 computers 12 cores with 48GB memory and 2 GPUs

COMING SOON

- 32 16-core machines
- Remote Visualization System

Cluster Resources

- 505TB of networked attached storage
- Backup of home and projects directories



more on cluster resources

- High performance Infiniband fabric network

- Command line interface
- Batch scheduling access to cluster machines

Data Storage

Permanent Data Storage

- Home directory
  - 2GB
  - /user/username
  - Back up
- Projects directory
  - 200GB – 500GB
  - /projects/groupname
  - Group space
  - Requested by Academic Research Advisor
  - Back up

Temporary Data Storage

- Panasas directory
  - /panasas/scratch
  - 215TB
  - Files older than 2 weeks are automatically deleted
  - Available on all nodes
- Scratch directory
  - /scratch
  - 256GB – 3TB
  - Local to compute node
  - Files deleted at end of job

Accessing the Cluster

Login Machine

- The CCR cluster front-end machine is **u2.ccr.buffalo.edu**
- 32-core node with 256GB of memory
- Login to cluster
- Transfer data files to and from cluster data storage
- Command line Linux environment
- Compile code, edit files and submit jobs to cluster

- Compute nodes and storage spaces are not directly accessible from outside the cluster.
- Small computations on login machine
- Accessible from UB network only
  - LAN or UB Secure Wireless on campus
  - Use UB VPN from off campus to connect to UB network

Accessing the Cluster

From Windows Machines

- Login
  - XWin-32 program
  - Allow graphical display back to user's workstation/laptop
  - PuTTY secure shell program
- File transfer
  - Filezilla or WinSCP are graphical file transfer programs
  - scp/sftp available in the PuTTY package
- UB VPN for off-campus access

More on:

- Xwin-32
- Filezilla
- WinSCP
- UB VPN
- User Guide

Accessing the Cluster

From Linux and MAC Machines

- Login
- ***ssh username@u2.ccr.buffalo.edu***
- ***ssh -X username@u2.ccr.buffalo.edu***
  - Allow graphical display back to user's workstation/laptop
- File transfer
  - Filezilla is a graphical file transfer program
  - sftp or scp
- UB VPN for off-campus access

More on:

- ssh/sftp/scp
- UB VPN for Linux
- UB VPN for MAC
- User Guide

Command Line

- List the contents of a directory: ***ls -l***
- Displays ownership and permissions.
- Show details and hidden files: ***ls -la***
- Show details in reverse time order: ***ls -latr***
- Show current directory pathname: ***pwd***
- Create a directory: ***mkdir dirname***

- Change directory: ***cd /pathname***
- Change to one directory above: ***cd ..***
- Change to two directories above: ***cd ../..***
- Change to home directory: ***cd***
- Remove a file: ***rm filename***
- Remove a directory: ***rm -R dirname***

Command Line

- **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, write and execute.
- drwxrwxrwx
- -rwxrwxrwx

- Read (r) permission allows viewing and copying.
- Write (w) permission allows changing and deleting.
- Execute (x) permission allows execution of a file and access to a directory.
- Show permissions: ***ls -l***
- The left most is – for a file and d for a directory.
- The next three are read, write and execute for the user (owner).

Command Line

- The middle three are read, write and execute for the group.
  - These permissions set the access for all members of the group.
- The rightmost three are read, write and execute for other.
- 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***
- Removing all permissions for group and other on a directory and its contents:
  - ***chmod -R go-rwx dirname***

Command Line

- The asterisk is a wild card for many UNIX commands.
  - List all C files: ***ls *.c***
- Files edited on Windows machines can contain hidden characters, which may cause runtime problems.
- Use the file command to show the type of file.
  - If the file is a DOS file, then use the dos2unix command to remove any hidden characters.

- Viewing files: cat, more or less
- ***cat filename*** displays a file to the screen
- ***more filename*** displays a file to the screen with page breaks
- ***less filename*** allows paging forward and back
- There are several editors available.
  - emacs, gedit, nano and vi

Command Line

- Files edited on Windows machines can contain hidden characters, which may cause runtime problems.
- Use the file command to show the type of file.
  - If the file is a DOS file, then use the dos2unix command to remove any hidden characters.
  - ***file filename***
  - ***dos2unix filename***

- Manual pages are available for UNIX commands.
  - ***man ls***
- Change password using the MyCCR web interface
  - More on MyCCR
- See the CCR user guide for more information.

more on the User Guide

CCR Command Line Reference Card

Modules

- Most applications installed on the cluster have a module file that sets paths and variables.
- Loading a module adds the path for the application to path variables and sets variables.
- Unloading a module removes the path and unsets the variables.

- List available modules
  - ***module avail***
  - ***module avail intel***
- Show what a module will do
  - ***module show module-name***
- Load a module
  - ***module load module-name***
- Unload a module
  - ***module unload module-name***

Modules

- List modules currently loaded
  - ***module list***
- Users can create custom modules to point to their own software installations.
- Creating your own module.
  - Create the privatemodules directory in your home directory.
  - Copy a module to the privatemodules directory.
  - Edit the module.
- Using your own module.
  - Load the use.own module.
    - ***module load use.own***
  - Load the module.
    - ***module load your-module-name***

Running on the Cluster

- Computations are submitted to the PBS (Portable Batch System) scheduler as jobs to run on the cluster.
  - Jobs can be script or interactive.
- The compute nodes are assigned to jobs by the scheduler.
- Users are not permitted to use compute nodes outside of the scheduled job.

- The qsub command is used to submit a job to the cluster.
- Specify the number and type of nodes.
  - -lnodes=2:IB2:ppn=8
    - Requests 2 compute nodes and 8 processors per node.
    - Total of 16 processors.
    - The IB2 tag specifies the 8-core nodes.

Running on the Cluster

- -lnodes=4:IB1:ppn=12
  - Requests 4 compute nodes and 12 processors per node.
  - Total of 48 cores.
  - The IB1 tag specifies the 12-core nodes.
- The PBS tags IB1 and IB2 stand for InfiniBand fabrics (networks) 1 and 2, corresponding to 12-core and 8-core nodes.

- Specify the length of time.
  - This is the walltime of the job. The maximum allowed walltime is 72 hours.
- Most jobs will run in the default queue.
  - There is no need to specify this queue.
- Using the `-I` flag on the command line requests an interactive job.

more on cluster nodes and PBS tags

Running on the Cluster

- The cluster has a debug queue for quick testing of codes.
  - The maximum walltime is 1 hour.
  - There are 2 8-core nodes, 2 12-core nodes, and 1 GPU node.
- Use the “-q debug” flag to request the debug queue.

- Submit the job.
  - ***qsub pbs-script***
- Show job status with the qstat command.
  - ***qstat -an -u username***
  - ***qstat jobid***
- A job will have a queued or run state.
  - Jobs in a run state have a node list.
  - The cores are indicated in the node list.

Running on the Cluster

- The first node in the head node of the job.
- The qdel command will delete a job.
  - ***qdel jobid***
- The showstart command will give an estimated start time for a job.
  - ***showstart jobid***
  - The start time is estimated based on the current queue. This value can change.

- The showq command with display the queue status. The running jobs are shown first, followed by the queued or idle jobs.
  - ***showq***
- pipe is helpful with showq: ***showq | more***
- The showbf command will show nodes that currently available.
  - ***showbf -S***
  - ***showbf -S | more***

Hadoop on the Cluster

- There are two methods of running a Hadoop computation on the cluster.
- The first is to use the MyHadoop scripts to set up and run the Hadoop computation.
- The second is an interactive job. We will look at this option later.
- Using the MyHadoop scripts:

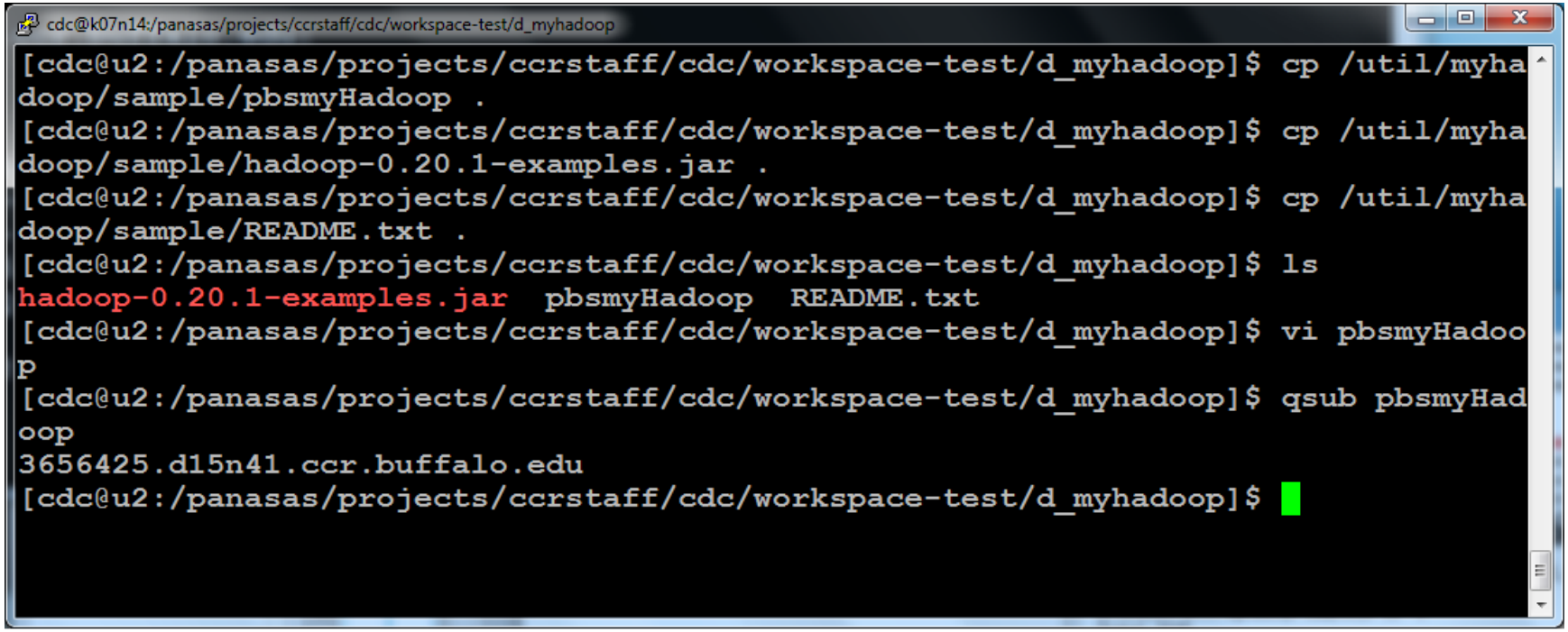
- Change to working directory.
- Copy the pbsmyHadoop, hadoop-0.20.1-examples.jar and README.txt files from the /util/myhadoop/sample directory.
- ***cp /util/myhadoop/sample/pbsmyHadoop .***
- ***cp /util/myhadoop/sample/hadoop-0.20.1-examples.jar .***

Hadoop on the Cluster

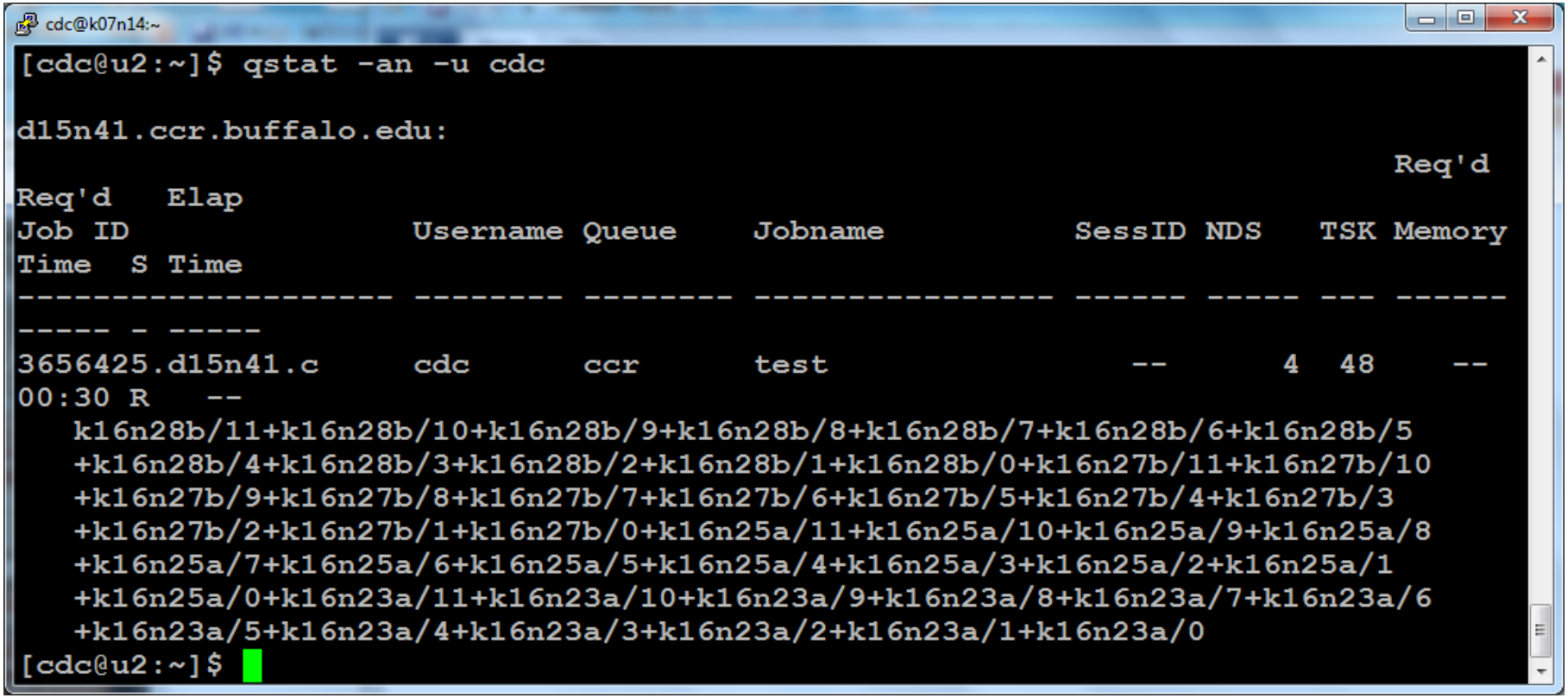
- ***cp /util/myhadoop/sample/README.txt .***
- Edit the pbsmyHadoop file.
- ***emacs pbsmyHadoop***
- Submit the job.
- ***qsub pbsmyHadoop***
- Show the job in the queue.
- ***qstat -an -u username***
- ***qstat -an jobid***

- The config directory is created in the working directory.
- The output file will be in this directory when the job completes.
- Monitor the job the ccrjobviz.pl command.
- ***ccrjobviz.pl jobid***

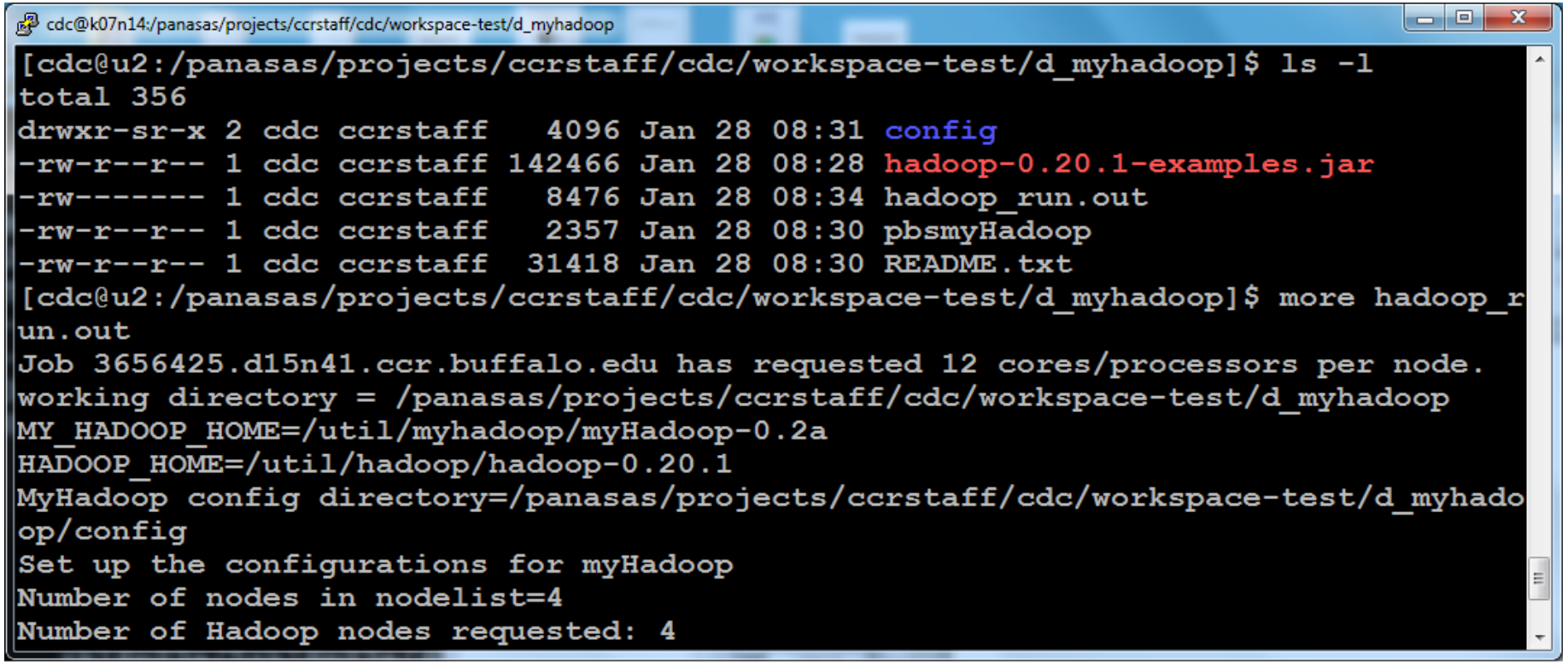
Submitting a Job



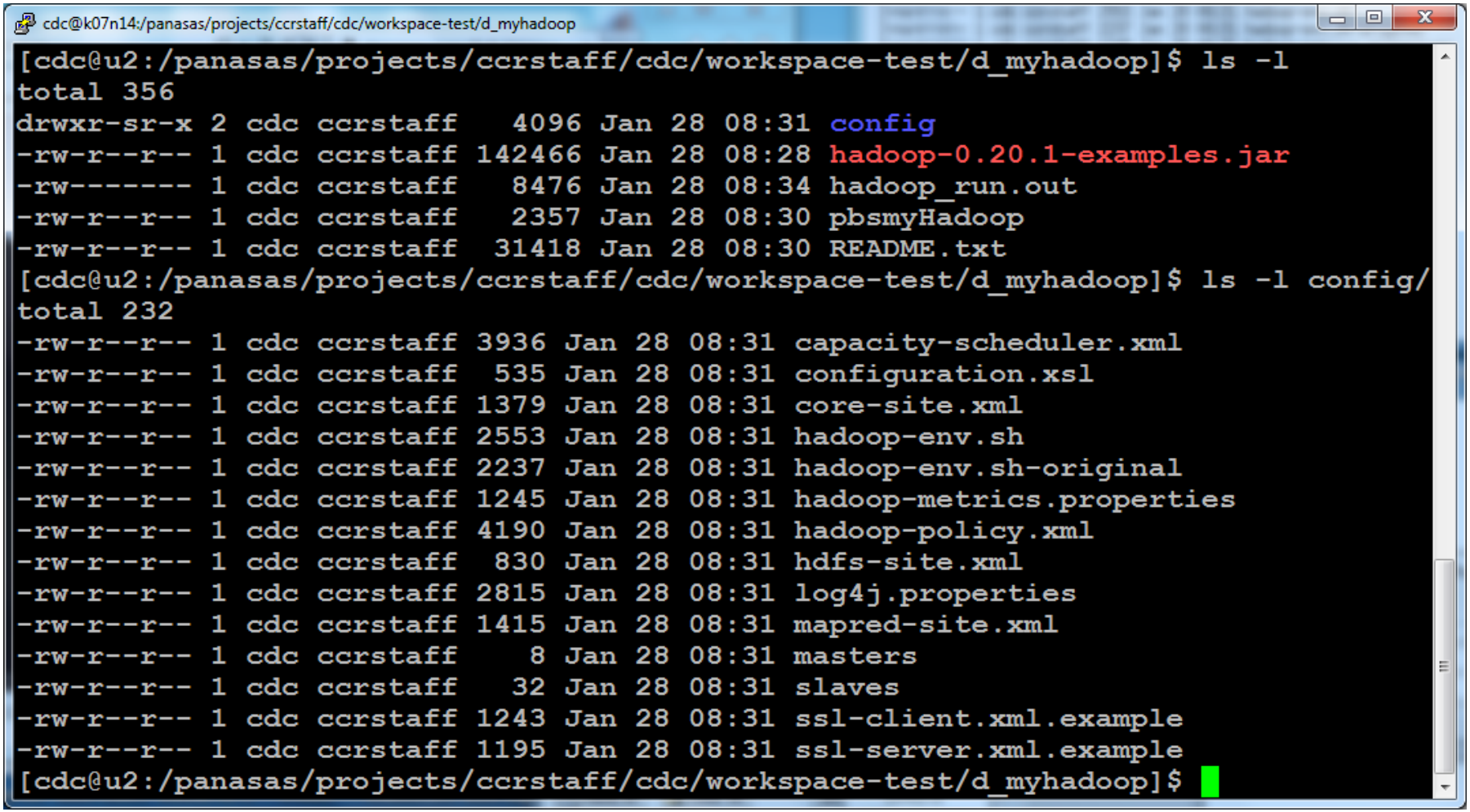
Running a Job



Running a Job



Running a Job



pbsmyHadoop Script page 1

```
#!/bin/bash
#PBS -l nodes=4:IB1:ppn=12
#PBS -l walltime=00:30:00
#PBS -M cdc@buffalo.edu
#PBS -m e
#PBS -N test
#PBS -o hadoop_run.out
#PBS -j oe

cd $PBS_O_WORKDIR
...
```

- specify bash shell
- node request
- time request
- email address
- email on job exit
- name of job
- output file
- stdout and stderr go to same output file
- change to the directory from which the job was submitted.

pbsmyHadoop Script page 2

```
echo "working directory = "$PBS_O_WORKDIR
. $MODULESHOME/init/sh
module load myhadoop/0.2a/hadoop-0.20.1
echo "MY_HADOOP_HOME="$MY_HADOOP_HOME
echo "HADOOP_HOME="$HADOOP_HOME
#### Set this to the directory where Hadoop configs should be generated
# Don't change the name of this variable (HADOOP_CONF_DIR) as it is
# required by Hadoop - all config files will be picked up from here
#
# Make sure that this is accessible to all nodes
export HADOOP_CONF_DIR=$PBS_O_WORKDIR/config
echo "MyHadoop config directory="$HADOOP_CONF_DIR
```

pbsmyHadoop Script page 2

- Display the working directory.
- Initialization modules.
- Load the module file for myhadoop.
- Display the value of the $MY_HADOOP_HOME variable.
- Display the value of the $HADOOP_HOME variable.
- Set the $HADOOP_CONF_DIR to the config directory in the working directory.
- Display the value of the $HADOOP_CONF_DIR variable.

pbsmyHadoop Script page 3

```
### Set up the configuration
# Make sure number of nodes is the same as what you have requested from PBS
# usage: $MY_HADOOP_HOME/bin/pbs-configure.sh -h
echo "Set up the configurations for myHadoop"
# this is the non-persistent mode
NNuniq=`cat $PBS_NODEFILE| uniq | wc -l`
echo "Number of nodes in nodelist="$NNuniq
$MY_HADOOP_HOME/bin/pbs-configure.sh -n $NNuniq -c $HADOOP_CONF_DIR

sleep 15
```

pbsmyHadoop Script page 3

- Create a unique node list using the $PBS_NODEFILE variable.
  - The $PBS_NODEFILE variable contains a list of the compute nodes assigned to the job with each node listed as many times as cores on the node.
  - `cat $PBS_NODEFILE| uniq | wc -l ` means display the nodefile and filter it through the unique command, then count the number of lines that are output
- Run the configure script .
- Wait for 15 seconds.

pbsmyHadoop Script page 4

```
# this is the persistent mode
# $MY_HADOOP_HOME/bin/pbs-configure.sh -n 4 -c $HADOOP_CONF_DIR -p -d /oasis/clo
udstor-group/HDFS

#### Format HDFS, if this is the first time or not a persistent instance
echo "Format HDFS"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR namenode -format

sleep 15

echo "start dfs"
$HADOOP_HOME/bin/start-dfs.sh

sleep 15
```

pbsmyHadoop Script page 4

- Format the HDFS on the compute node. The HDFS (Hadoop Distributed File System) is created using the local disk of each compute node running the job.
- Wait 15 seconds.
- Start the dfs.
- Wait 15 seconds.

pbsmyHadoop Script page 5

```
echo "copy file to dfs"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR
dfs -put README.txt /

sleep 15

echo "ls files in dfs"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR
dfs -ls /

echo "start jobtracker (mapred)"
$HADOOP_HOME/bin/start-mapred.sh

echo "ls files in dfs"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR
fs -ls /
```

pbsmyHadoop Script page 5

- Copy file to dfs.
- Wait 15 seconds.
- List the files in dfs.
- Start the job tracker.
- List the files in dfs.

pbsmyHadoop Script page 6

```
echo "run computation"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR jar
hadoop-0.20.1-examples.jar
 wordcount /README.txt /output

echo "ls files in dfs"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR fs -ls
/output

echo "view output results"
$HADOOP_HOME/bin/hadoop --config $HADOOP_CONF_DIR fs -
cat /output/part-r-00000

echo "stop jobtracker (mapred)"
$HADOOP_HOME/bin/stop-mapred.sh

echo "stop dfs"
$HADOOP_HOME/bin/stop-dfs.sh
```

pbsmyHadoop Script page 6

- Run the computation.
- List the files in dfs.
- View the results.
- Stop the job tracker.
- Stop the dfs.

pbsmyHadoop Script page 7

```
#### Clean up the working directories after job completion
echo "Clean up"
$MY_HADOOP_HOME/bin/pbs-cleanup.sh -n $NNuniq
echo
```

pbsmyHadoop Script page 7

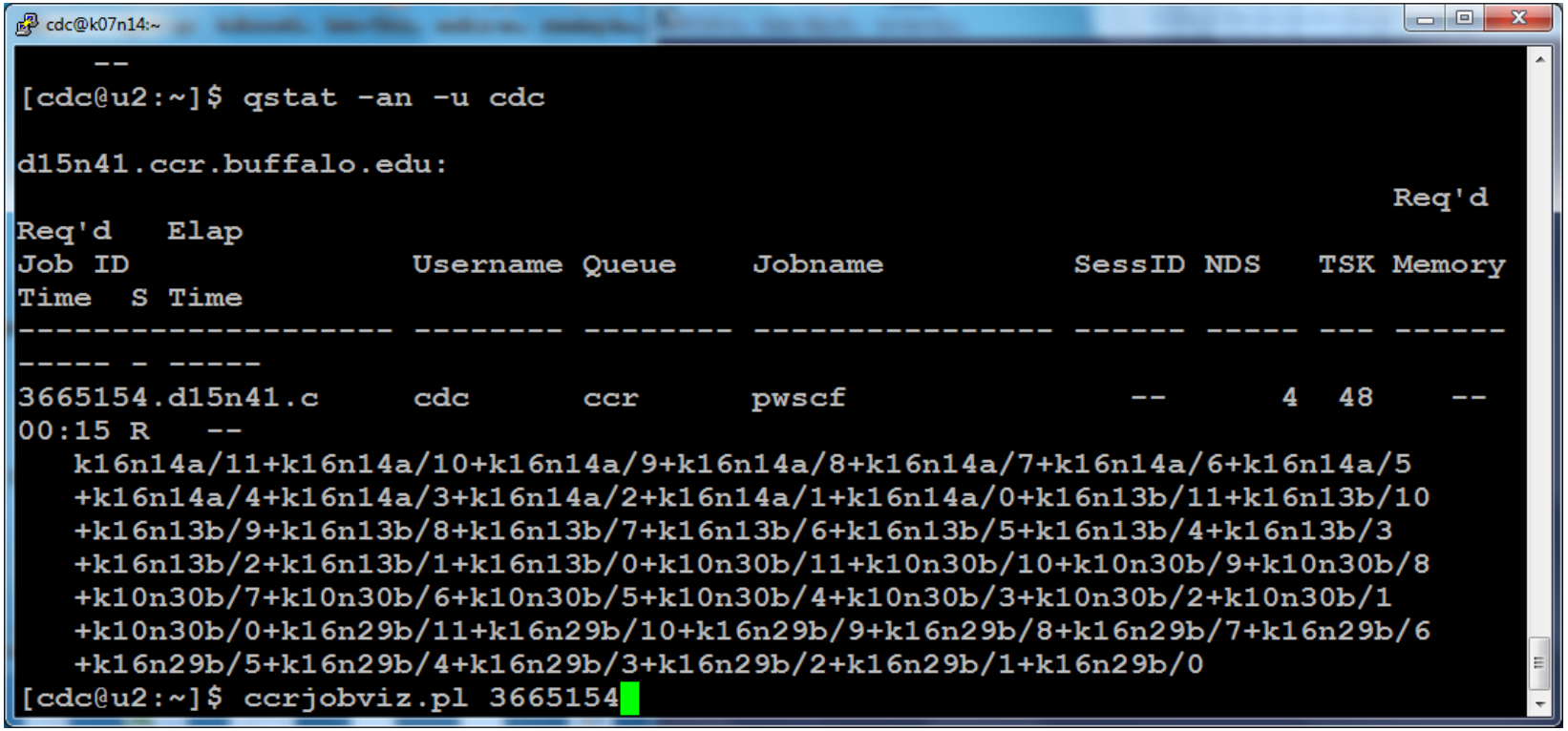
- Run the cleanup script.

Monitoring a Job

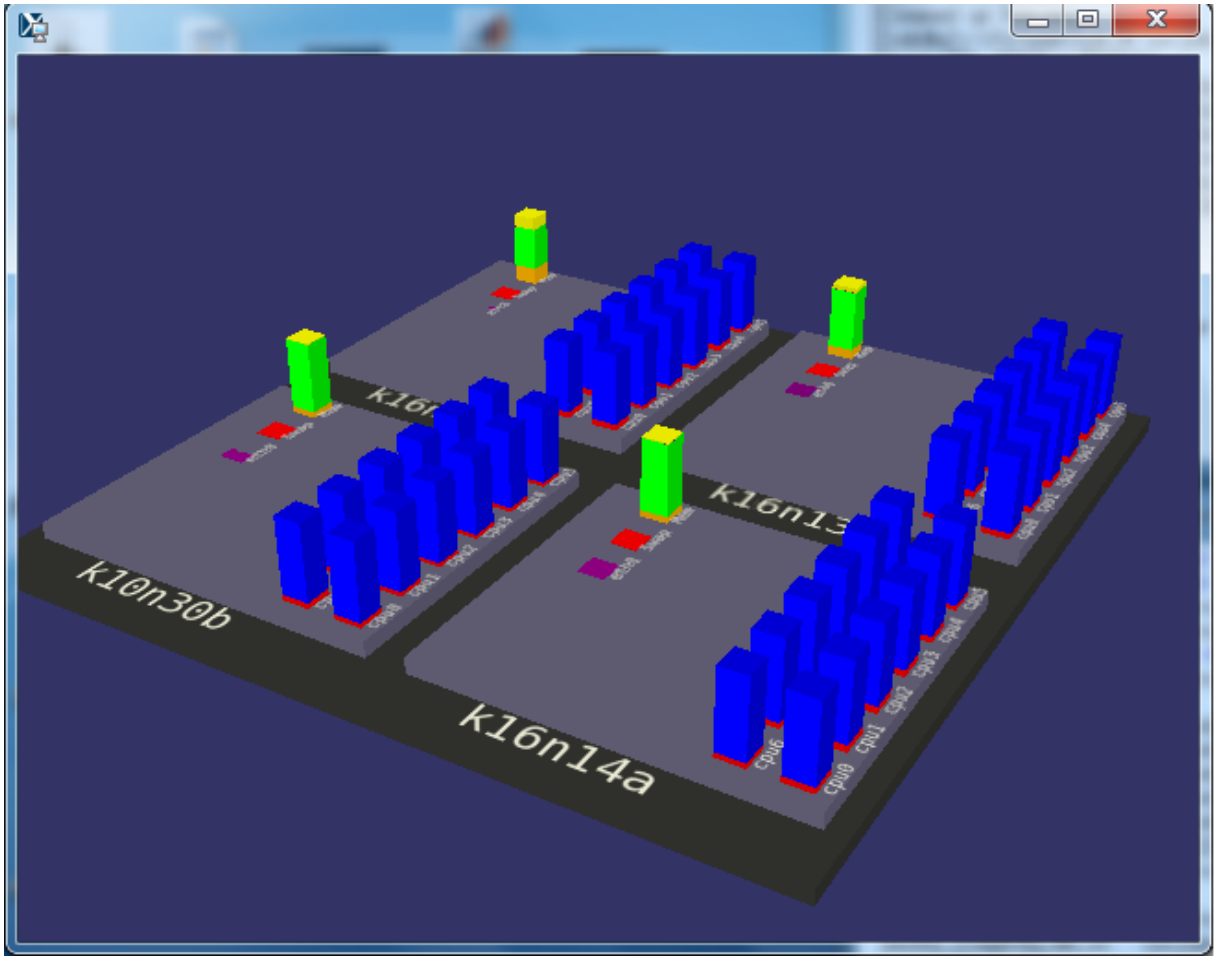
- ccrjobviz.pl is a graphical display used to monitor and evaluate a running job.
  - ***ccrjobviz.pl jobid***
- The top command can used to monitor the processes on a node.
  - Use **qstat -an -u username** or ***qstat -an jobid***
  - ssh into the node: ***ssh nodename***
  - Run top on the node: ***top***

- The first node in the list is the head node of the job, which contains the standard output and error file while the job running.
  - ssh into the node: ***ssh nodename***
  - ***cd /var/spool/pbs/spool***
  - The output file will have the jobid as the file name.
  - ***cat jobid***

Monitoring a Job



Monitoring a Job



Hadoop on the Cluster

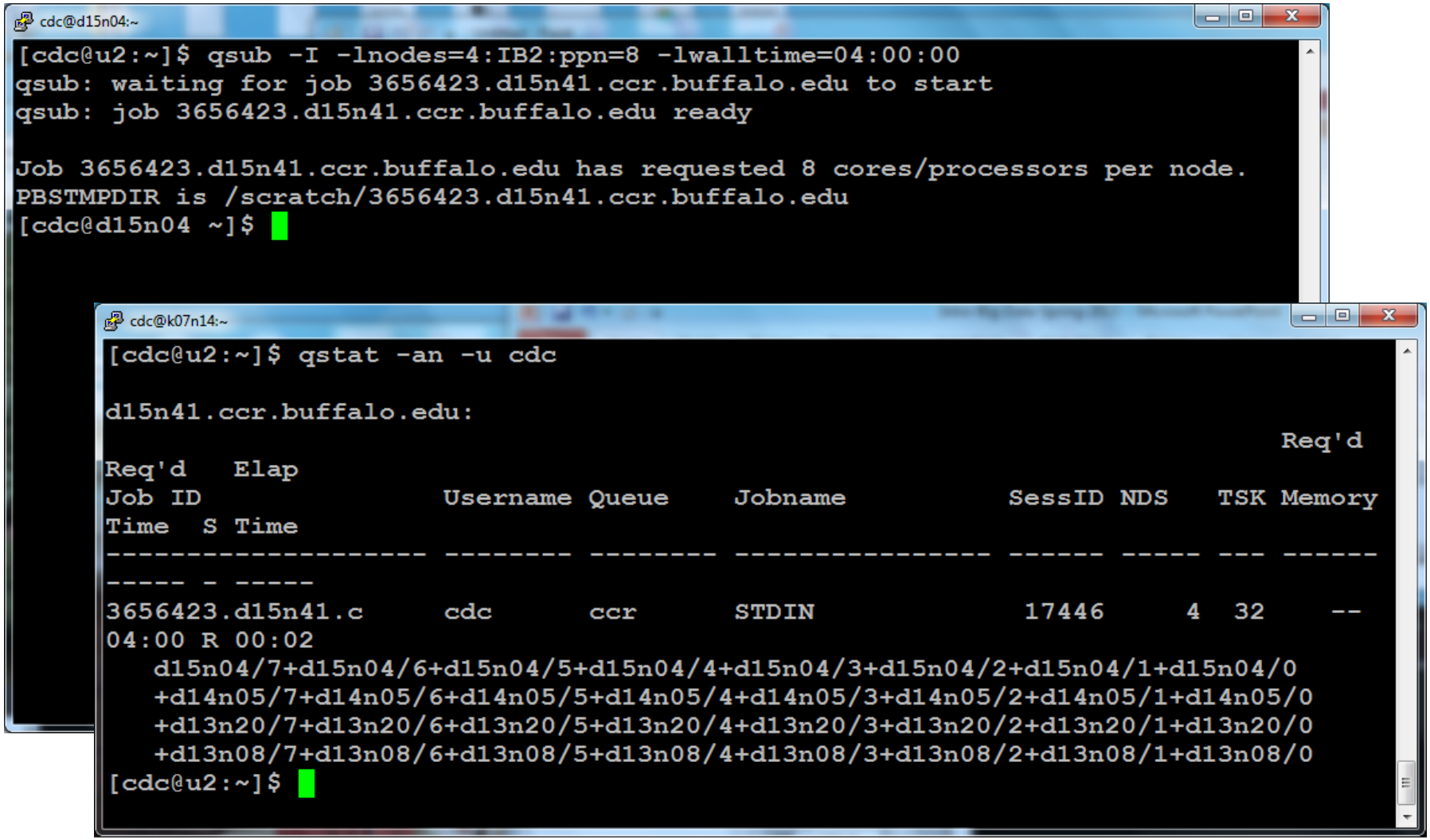
- The second method of running a Hadoop computation is to use an interactive job.
- Submit an interactive job.
- ***qsub -I***

  -lnodes=4:IB2:ppn=8

  -lwalltime=03:00:00
- Once the job starts, configure Hadoop by hand and run the computation.

- This will take more time, however you will learn the specific configuration steps necessary for the hadoop computation.
- Here are the instructions: Running Hadoop - CCR

Submitting an Interactive Job



User Support

Online Resources:

New User Information

User Guide

Tutorials and Presentations

Help:

CCR staff members are available to assist you.

UB Faculty and student can request individual and group training sessions.

more on CCR Help