Welcome to Kamiak

1/20/2017 Training Session
Aurora Clark, CIRC Director
Jeff White, Kamiak Linux Admin
Shenting Cui, Computational Scientist
Kamiak

NetApp Storage

login nodes (2)

Compute Nodes

User Interface
Where did this infrastructure come from?

• $3 Mil
  - Founding investing units (CAHNRS, CAS, VCEA)
  - Office of Research
  - Information Technology Services
  - Provost/President

Capability
• Chassis
• Cabling
• Networking
• Base storage
• Base stakeholder (College) nodes
• 2 staff

Sustainable ongoing support is being provided as Kamiak grows through faculty investment
Kamiak

CAS (12 nodes)  CAHNRS (11 nodes)  VCEA (3 nodes)  OR (4 nodes)  Investor (76)

Compute Nodes
Kamiak

- every user /home directory has 10Gb (quota `-s -f /home`)
- extra storage goes into /data

Extra storage:

1) CIRC/ITS Service Center for Storage

144 Tb total, with 63 Tb purchased by faculty investors
(`df --h /data/investor name`)

2) 5-year storage purchased by Colleges:

CAHNRS: ~200 Tb
CAS: ~100 Tb
VCEA: ~25 Tb
OR: ~13 Tb
Introduction

Example Workflow:

User

Login
Create files, move data, etc

Load software modules
Decide resources
Submit Queue

Compute
Pre-process
Post-process
Visualize

Analyze
Share
Move data

This training event
What you will learn today

- Navigating Kamiak using essential linux commands
- How to transfer files on and off Kamiak
- How to find and load software modules
- How to submit jobs and optimize your computational efficiency using the queue
- Best practices to being a good user
- Tips and tricks to installing your own software
- How to get help
- How to invest (nodes or storage)
Brief Linux Review

• If you are on a PC/Mac you will utilize an Xterminal, which mimics the “non-graphical” command-line prompt of a Linux/Unix OS
• Free software: putty (for windows), terminal (for mac – preinstalled)
Brief Linux Review

- Secure shell protocol (ssh) typically used to login to Kamiak

Last login: Tue Jan 17 15:17:56 on ttys000
Auroras-MacBook-Pro:~ auclark$ ssh auclark@kamiak.wsu.edu
auclark@kamiak.wsu.edu's password: 🛠️
Once logged in – there are many basic Linux commands you must master (see Ch2 of tutorial sent in email)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ls -al</code></td>
<td>Display all information about files/directories</td>
</tr>
<tr>
<td><code>pwd</code></td>
<td>Show the path of current directory</td>
</tr>
<tr>
<td><code>mkdir directory-name</code></td>
<td>Create a directory</td>
</tr>
<tr>
<td><code>rm file-name</code></td>
<td>Delete file</td>
</tr>
<tr>
<td><code>rm -r directory-name</code></td>
<td>Delete directory recursively</td>
</tr>
<tr>
<td><code>rm -f file-name</code></td>
<td>Forcefully remove file</td>
</tr>
<tr>
<td><code>rm -rf directory-name</code></td>
<td>Forcefully remove directory recursively</td>
</tr>
<tr>
<td><code>cp file1 file2</code></td>
<td>Copy file1 to file2</td>
</tr>
<tr>
<td><code>cp -r dir1 dir2</code></td>
<td>Copy dir1 to dir2, create dir2 if it doesn’t exist</td>
</tr>
<tr>
<td><code>mv file1 file2</code></td>
<td>Rename source to dest / move source to directory</td>
</tr>
<tr>
<td><code>ln -s /path/to/file-name link-name</code></td>
<td>Create symbolic link to file-name</td>
</tr>
<tr>
<td><code>touch file</code></td>
<td>Create or update file</td>
</tr>
<tr>
<td><code>cat &gt; file</code></td>
<td>Place standard input into file</td>
</tr>
<tr>
<td><code>more file</code></td>
<td>Output contents of file</td>
</tr>
<tr>
<td><code>head file</code></td>
<td>Output first 10 lines of file</td>
</tr>
<tr>
<td><code>tail file</code></td>
<td>Output last 10 lines of file</td>
</tr>
<tr>
<td><code>tail -f file</code></td>
<td>Output contents of file as it grows starting with the last 10 lines</td>
</tr>
<tr>
<td><code>gpg -c file</code></td>
<td>Encrypt file</td>
</tr>
<tr>
<td><code>gpg file.gpg</code></td>
<td>Decrypt file</td>
</tr>
<tr>
<td><code>wc</code></td>
<td>Print the number of bytes, words, and lines in files</td>
</tr>
<tr>
<td><code>xargs</code></td>
<td>Execute command lines from standard input</td>
</tr>
</tbody>
</table>
Once logged in – there are many basic Linux commands you must master (see Ch2 of tutorial sent in email)

**DIRECTORY TRAVERSE**

- `cd ..` => To go up one level of the directory tree
- `cd` => Go to $HOME directory
- `cd /test` => Change to /test directory
Brief Linux Review

• Once logged in – there are many basic Linux commands you must master (see Ch2 of tutorial sent in email)

DIRECTORY TRAVERSE

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd ..</td>
<td>=&gt; To go up one level of the directory tree</td>
</tr>
<tr>
<td>cd</td>
<td>=&gt; Go to $HOME directory</td>
</tr>
<tr>
<td>cd /test</td>
<td>=&gt; Change to /test directory</td>
</tr>
</tbody>
</table>

• There are many, many tutorials online that can help you solidify your Linux command expertise
Brief Linux Review

- Once logged in – there are many basic Linux commands you must master (see Ch2 of tutorial sent in email)

**DIRECTORY TRAVERSE**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd ..</td>
<td>Go up one level of the directory tree</td>
</tr>
<tr>
<td>cd</td>
<td>Go to $HOME directory</td>
</tr>
<tr>
<td>cd /test</td>
<td>Change to /test directory</td>
</tr>
</tbody>
</table>
• If you want to open and edit files, you need to also choose a text editor to use:

• There are many, many tutorials on different text editors as well
There are several ways to transfer and synchronize files across different computers.

**Transfer only**
- `scp` (secure copy), `sftp` (secure file transfer protocol)

**Synchronize**
- `rsync` (once two copies are established on 2 computers, you can only copy the most recent updates to files – this decreases network traffic, good for large amounts of data)
  - Versatile for data backups and mirroring

**Some common options used with rsync commands**

- `-v`: verbose
- `-r`: copies data recursively (but don’t preserve timestamps and permission while transferring data)
- `-a`: archive mode, archive mode allows copying files recursively and it also preserves symbolic links, file permissions, user & group ownerships and timestamps
- `-z`: compress file data
- `-h`: human-readable, output numbers in a human-readable format
Example uses of rsync

- Copying data from your local computer to your home directory on Kamiak

- A file:

```
Auroras-MacBook-Pro:Desktop auclark$ rsync -avz test.xyz auclark@kamiak.wsu.edu:~/
auclark@kamiak.wsu.edu's password:
building file list ... done
test.xyz
```

- A directory

```
Auroras-MacBook-Pro:Desktop auclark$ rsync -avz codes-n-scripts/ auclark@kamiak.wsu.edu:~/
```
File Transfers

- Example uses of rsync
  - Copying files/synching files from Kamiak to your local machine

Auroras-MacBook-Pro:Desktop auclark$ rsync -avz auclark@kamiak.wsu.edu:~/test.xyz
auclark@kamiak.wsu.edu's password:
receiving file list ... done

sent 20 bytes received 102 bytes 11.62 bytes/sec
total size is 3987 speedup is 32.68

Need more than 10Gb for a short period of time?
- Use Scratch!
Scratch Space

All users have access to temporary, fast, scratch storage. Scratch storage is allocated using the mkworkspace function which can be found in the workspace_maker module. All data written to scratch will be deleted one week after the workspace is created.

Let's look at an example of how to create a scratch space allocation.

Creating a Scratch Space

To create a scratch space run the `mkworkspace` command.

```
[myWSU.NID@login-p1n01 ~]$ mkworkspace
Successfully created workspace. Details:
Workspace: /scratch/myWSU.NID_509834
User: myWSU.NID
Group: its_p_sto_qa_hpc_kamiak-my_group
Expiration: 2016-06-13 10:34:14.583285
[myWSU.NID@login-p1n01 data]$ cd /scratch/myWSU.NID_509834/
[myWSU.NID@login-p1n01 myWSU.NID_509834]$ 
```

Notice that our new scratch space was given an expiration date of 2016-06-13, one week after the creation of `/scratch/myWSU.NID_509834/`. On this date, the directory `/scratch/myWSU.NID_509834/` and its contents will be deleted.

*By default, this creates space on the shared 10K disks in the NetApps storage device (runs over 10Gb network)*
Creating a Workspace and Using Scratch

Using the Local SSD

The program `mkworkspace` will allow you to select from different backends using the `-b` flag. To use the local disk, select the `/local` backend.

```
[myWSU.NID@login-p1n02 ~]$ idev --partition=cas --account=cas
Requesting 1 node(s) from cas partition
1 task(s)/node, 20 cpu(s)/task
Time: 0 (hr) 60 (min).
Submitted batch job 75937
Job is pending. Please wait. 0(s)
JOBID=75937 begin on cn14
--> Creating interactive terminal session (login) on node cn14.
--> You have 0 (hr) 60 (min).
--> Assigned Host List : /tmp/idev_nodes_file_myWSU.NID
[myWSU.NID@cn14 ~]$ mkworkspace -b /local -t 0-2:00
Successfully created workspace. Details:
  Workspace: /local/myWSU.NID_600982
  User: myWSU.NID
  Group: its_p_sto_qa_hpc_kamiak-my_group
  Expiration: 2016-06-06 13:12:29.834586
```

Notice that we create the allocation with a 2hr life span (matching the lifespan of our `idev` session). This ensures that the 600GB local disk remains available to all users.
Listing Scratch Allocations

A list of your scratch allocations and their expiration dates can be obtained using the command `lsworkspace`

```
[myWSU.NID@login-p1n02 ~]$ lsworkspace
Workspace: /scratch/myWSU.NID_509834
  Creation host: login-p1n02.kamiak.wsu.edu
  Creation time: 2016-06-06 10:34:14.583205
  User owner: myWSU.NID
  Group owner: its_p_sto_qa_hpc_kamiak-my_group
  Expiration time: 2016-06-13 10:34:14.583285
[myWSU.NID@login-p1n02 ~]$
```
• Now – you should be able to (and practice):
  - Logging into kamiak
  - Decide and learn what text editor you want to use
  - Basic linux commands to navigate directories, move files, etc.
  - Moving files onto and off of kamiak

• What’s next → exploring the available software
Software Modules on Kamiak

How to use environment modules (not kernel modules)

$ module -help               # lots of options and sub_commands
$ module avail            # all modules available on the system
$ module whatis libint/1.1.4  # what the module does
$ module spider module_name  # more info about a module
$ module swap m_old m_new    # unload m_old, load m_new
$ module purge              # unload all modules
$ module unload module_name  # unload a module
$ module load netcdf/4      # demonstrate module dependency

Lmod has detected the following error: Cannot load module "netcdf/4" without these module(s) loaded:
hdf5/1.8.16             # You have to module load hdf5/1.8.16 first

Check the module you loaded:

$ module load octave; module list
$ which octave          # check whether octave is in your path
$ env |$LD_LIBRARY_PATH  # check whether the library you loaded is in your path
$ icc -v               # tell you what icc version is in your path
### Software Modules on Kamiak

<table>
<thead>
<tr>
<th>Compiler Modulefiles</th>
<th>Compiler</th>
<th>Intel XE 2016 Update 3</th>
<th>Other Modulefiles</th>
</tr>
</thead>
<tbody>
<tr>
<td>StdEnv (L)</td>
<td>gcc/4.9.3</td>
<td>gcc/5.2.0</td>
<td>anaconda2/2.4.0</td>
</tr>
<tr>
<td></td>
<td>gcc/5.2.0</td>
<td>gcc/6.1.0 (D)</td>
<td>google_sparsehash/4cb9240</td>
</tr>
<tr>
<td></td>
<td>intel/xe_2016_update2</td>
<td>intel/xe_2016_update3 (L,D)</td>
<td>python3/3.4.3</td>
</tr>
<tr>
<td></td>
<td>corset/1.06</td>
<td>espresso/5.3.0</td>
<td>anaconda2/4.2.0 (D)</td>
</tr>
<tr>
<td></td>
<td>lammps/16feb16</td>
<td>nwchem/6.6</td>
<td>java/oracle_1.8.0_92</td>
</tr>
<tr>
<td></td>
<td>siesta/4.0_mpi</td>
<td>elpa/2016.05.003</td>
<td>r/3.2.2</td>
</tr>
<tr>
<td></td>
<td>hdf5/1.8.16</td>
<td>netcdf/4</td>
<td>r/3.3.0 (D)</td>
</tr>
<tr>
<td></td>
<td>octave/4.0.1</td>
<td></td>
<td>samtools/1.3.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>setrtarg/6.0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>blast/2.2.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bamtools/2.4.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>blast/2.2.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bonnie++/1.03e</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>boost/1.59.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bowtie/1.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>canu/1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cast/dbf2ec2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>clc_genomics_workbench/6.0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>lmod/6.0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bowtie/1.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>canu/1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cast/dbf2ec2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>clc_genomics_workbench/6.0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>lmod/6.0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bowtie/1.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>canu/1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cast/dbf2ec2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>clc_genomics_workbench/6.0.1</td>
</tr>
</tbody>
</table>
Software Modules on Kamiak Cont’d

<table>
<thead>
<tr>
<th>Software Module</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>clc_genomics_workbench</td>
<td>8.5.1 (D)</td>
</tr>
<tr>
<td>netapp</td>
<td>5.5 (D)</td>
</tr>
<tr>
<td>trinity</td>
<td>2.2.0</td>
</tr>
<tr>
<td>cp2k</td>
<td>4.1</td>
</tr>
<tr>
<td>openblas</td>
<td>0.2.18_barcelona</td>
</tr>
<tr>
<td>valgrind</td>
<td>3.11.0</td>
</tr>
<tr>
<td>cuda</td>
<td>7.5</td>
</tr>
<tr>
<td>openblas</td>
<td>0.2.18_haswell</td>
</tr>
<tr>
<td>workspace_maker</td>
<td>master (L,D)</td>
</tr>
<tr>
<td>gaussian</td>
<td>09.d.01</td>
</tr>
<tr>
<td>openblas</td>
<td>0.2.18 (D)</td>
</tr>
<tr>
<td>workspace_maker</td>
<td>1.1b</td>
</tr>
<tr>
<td>gdal</td>
<td>2.1.0</td>
</tr>
<tr>
<td>parmetis</td>
<td>4.0.3</td>
</tr>
<tr>
<td>workspace_maker</td>
<td>1.1</td>
</tr>
<tr>
<td>gdb</td>
<td>7.10.1</td>
</tr>
<tr>
<td>pexsi</td>
<td>0.9.2</td>
</tr>
<tr>
<td>workspace_maker</td>
<td>1.2</td>
</tr>
<tr>
<td>geos</td>
<td>3.5.0</td>
</tr>
<tr>
<td>proj</td>
<td>4.9.2</td>
</tr>
<tr>
<td>git</td>
<td>2.6.3</td>
</tr>
<tr>
<td>python</td>
<td>2.7.10</td>
</tr>
</tbody>
</table>

**Where:**

**L:** Module is loaded

**D:** Default Module in cases there are several versions are available

For example, try:

```shell
$ module load python3; module list
```

# note: not necessary to
# include version number

You can see that python3/3.5.0 is loaded

You can also do module load in job script
• How to get this software to work for you → submitting to the queue
• Kamiak is primarily a **batch** processing system intended to run non-interactive compute **jobs** on the individual compute **nodes** of a **queue/partition**

• Slurm is the batch scheduler and resource manager used to control compute nodes and run jobs on them

• A node is a computer which has resources available for jobs:
  - CPU cores
  - Memory
  - Accelerators (GPU, Xeon Phi)

• A **queue/partition** is a set of nodes

• Slurm does not automatically parallelize or otherwise improve your program, it just runs your program on the node(s) it assigns your job to.
• Follow along (optional):
  - Log into Kamiak
    • Linux or Mac: ssh -Y kamiak.wsu.edu
    • Windows: PuTTY, MobaXterm, etc.

See also:
https://hpc.wsu.edu/users-guide/terminal-ssh
Running Jobs on Kamiak - Slurm

- **sinfo**: Information about the cluster
  - `sinfo --all`
  - What partitions are available?
  - What/how many nodes are in each of them?

- **squeue**: View running and queued jobs
  - `squeue --all`
  - Are my jobs running, pending, or held?

- **scontrol**: View information about aspects of the cluster
  - `scontrol show node sn11`
  - `scontrol show job $job_id`
  - How many GPUs does node sn11 have?
  - How much time is left in my job before its time limit?

- **scancel**: Cancel one or more jobs

- **sbatch**: Submit a new job
  - `sbatch my_code.sh`

- **srun**: Run a parallel job (usually within a batch job)
  - `srun my_mpi_program`
Running Jobs on Kamiak - Slurm

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>474239</td>
<td>cas</td>
<td>tp4</td>
<td>craig.te</td>
<td>PD</td>
<td>0:00</td>
<td>2</td>
<td>(Resources)</td>
</tr>
<tr>
<td>474240</td>
<td>cas</td>
<td>tp5</td>
<td>craig.te</td>
<td>PD</td>
<td>0:00</td>
<td>2</td>
<td>(Priority)</td>
</tr>
<tr>
<td>474241</td>
<td>cas</td>
<td>tp6</td>
<td>craig.te</td>
<td>PD</td>
<td>0:00</td>
<td>2</td>
<td>(Priority)</td>
</tr>
<tr>
<td>468054</td>
<td>kamiak</td>
<td>g09test</td>
<td>e.martin</td>
<td>PD</td>
<td>0:00</td>
<td>2</td>
<td>(launch failed requeued held)</td>
</tr>
<tr>
<td>471077</td>
<td>popgenom</td>
<td>BFS.3L.i</td>
<td>joel.t.n</td>
<td>R</td>
<td>9-03:34:38</td>
<td>1</td>
<td>cn77</td>
</tr>
<tr>
<td>471078</td>
<td>popgenom</td>
<td>BFS.2R.i</td>
<td>joel.t.n</td>
<td>R</td>
<td>9-03:34:03</td>
<td>1</td>
<td>cn29</td>
</tr>
<tr>
<td>471079</td>
<td>popgenom</td>
<td>BFS.2L.i</td>
<td>joel.t.n</td>
<td>R</td>
<td>9-03:33:14</td>
<td>1</td>
<td>cn29</td>
</tr>
<tr>
<td>473678</td>
<td>kamiak</td>
<td>cle_6cma</td>
<td>tung.ngu</td>
<td>R</td>
<td>6-05:07:11</td>
<td>1</td>
<td>cn7</td>
</tr>
<tr>
<td>473722</td>
<td>beckman</td>
<td>hydr-vra</td>
<td>hong.zho</td>
<td>R</td>
<td>6-04:28:26</td>
<td>2</td>
<td>cn[43-44]</td>
</tr>
<tr>
<td>473726</td>
<td>beckman</td>
<td>occ-1na3</td>
<td>hong.zho</td>
<td>R</td>
<td>5-23:37:54</td>
<td>2</td>
<td>cn[52-53]</td>
</tr>
<tr>
<td>473727</td>
<td>kamiak</td>
<td>dbl_Pt21</td>
<td>mareike.</td>
<td>R</td>
<td>2-23:09:49</td>
<td>2</td>
<td>cn[3-4]</td>
</tr>
</tbody>
</table>
Running Jobs on Kamiak - Slurm

```bash
kamiak$ sinfo
```

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>TIMELIMIT</th>
<th>NODES</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>cahnrs</td>
<td>up</td>
<td>7-00:00:00</td>
<td>1</td>
<td>mix cn9</td>
<td></td>
</tr>
<tr>
<td>cahnrs</td>
<td>up</td>
<td>7-00:00:00</td>
<td>10</td>
<td>alloc cn[1-8,10-11]</td>
<td></td>
</tr>
<tr>
<td>cahnrs_bigmem</td>
<td>up</td>
<td>7-00:00:00</td>
<td>1</td>
<td>mix sn4</td>
<td></td>
</tr>
<tr>
<td>cahnrs_gpu</td>
<td>up</td>
<td>7-00:00:00</td>
<td>1</td>
<td>mix sn2</td>
<td></td>
</tr>
<tr>
<td>cas</td>
<td>up</td>
<td>7-00:00:00</td>
<td>2</td>
<td>mix cn[23-24]</td>
<td></td>
</tr>
<tr>
<td>cas</td>
<td>up</td>
<td>7-00:00:00</td>
<td>9</td>
<td>alloc cn[14-15,17-2]</td>
<td></td>
</tr>
<tr>
<td>cas</td>
<td>up</td>
<td>7-00:00:00</td>
<td>1</td>
<td>down cn16</td>
<td></td>
</tr>
<tr>
<td>clark</td>
<td>up</td>
<td>7-00:00:00</td>
<td>5</td>
<td>idle cn[80-84]</td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>up</td>
<td>4:00:00</td>
<td>1</td>
<td>mix cn32</td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>up</td>
<td>4:00:00</td>
<td>1</td>
<td>alloc cn33</td>
<td></td>
</tr>
<tr>
<td>free_gpu</td>
<td>up</td>
<td>7-00:00:00</td>
<td>1</td>
<td>mix sn3</td>
<td></td>
</tr>
<tr>
<td>free_phi</td>
<td>up</td>
<td>7-00:00:00</td>
<td>1</td>
<td>resv sn1</td>
<td></td>
</tr>
<tr>
<td>kamiak*</td>
<td>up</td>
<td>7-00:00:00</td>
<td>1</td>
<td>maint* cn72</td>
<td></td>
</tr>
<tr>
<td>kamiak*</td>
<td>up</td>
<td>7-00:00:00</td>
<td>1</td>
<td>resv sn1</td>
<td></td>
</tr>
<tr>
<td>kamiak*</td>
<td>up</td>
<td>7-00:00:00</td>
<td>27</td>
<td>mix cn[9,12-13,23-]</td>
<td></td>
</tr>
<tr>
<td>kamiak*</td>
<td>up</td>
<td>7-00:00:00</td>
<td>34</td>
<td>alloc cn[1-8,10-11,]</td>
<td></td>
</tr>
<tr>
<td>kamiak*</td>
<td>up</td>
<td>7-00:00:00</td>
<td>30</td>
<td>idle cn[53-71,80,]</td>
<td></td>
</tr>
<tr>
<td>kamiak*</td>
<td>up</td>
<td>7-00:00:00</td>
<td>1</td>
<td>down cn16</td>
<td></td>
</tr>
</tbody>
</table>
Running Jobs on Kamiak – Slurm Backfill

• Backfill vs. investor and sponsor partitions
• Backfill includes all nodes of the cluster
• Preemption
  - Slurm will cancel and resubmit backfill jobs if an investor’s job requires use of the node
  - Example:
    1. You submit a job to the backfill partition, it gets assigned to cn83
    2. cn83 is also in the “clark” partition
    3. A user of the Clark lab submits a job to the clark partition
    4. Slurm attempts to find idle resource in the clark partition to assign
    5. Failing that, Slurm cancels the backfill job and runs the investor job
    6. The backfill job is resubmitted to the backfill partition
• Investors have on-demand access to the nodes they own
• Resource limits
  - Currently: 6 standard compute nodes (120 CPU cores)
To use Kamiak, write submission script to submit a batch job to Slurm defining:

- What resources your job needs (CPU cores, memory, GPU, ...)
- Various constraints and other settings (see `man sbatch`)
- Your program and how to run it

---

```bash
#!/bin/bash

#SBATCH --ntasks=1            # Number of tasks
#SBATCH --cpus-per-task=1     # Number of CPU cores per task
#SBATCH --nodes=1-1           # Number of nodes (min-max)
#SBATCH --gres=gpu:tesla:1    # Generic RESources
#SBATCH --partition=kamiak    # Partition/Queue to use

echo "Starting test GPU job on host $HOSTNAME"

module load cuda
deviceQuery

echo "Completed test GPU job on host $HOSTNAME"

kamiak$ sbatch test_gpu_job.sh
Submitted batch job 296129
```

---

**Note:** Do not run your script, pass it to `sbatch`
Using mkworkspace in a SLURM script

The program mkworkspace was written to work well within a SLURM submission script. By adding the -q flag, one can suppress the all the output except the name of the directory that was created for the allocation. This can then be stored in a local variable and invoked within the script.

```bash
#!/bin/bash
#SBATCH --job-name=WSM     ### Job Name
#SBATCH --partition=free 
#SBATCH --time=0-00:01:00    ### Wall clock time limit in Days-HH:MM:SS
#SBATCH --nodes=1           ### Node count required for the job
#SBATCH --ntasks-per-node=1 ### Number of tasks to be launched per Node

SCRATCHDIR="$(mkworkspace -q)"

my_executable $SCRATCHDIR

rmworkspace -a -f --name=$SCRATCHDIR
```
Running Jobs on Kamiak – Slurm Serial Jobs

```bash
kamiak$ scontrol show job 296129
JobId= 296129  JobName=test_gpu_job.sh
    UserId=my.NID(8003)  GroupId=its_p_sys_ur_kam-its_staff(7000)
    Priority=4294967295  Nice=0  Account=its_staff  QOS=normal
    JobState=COMPLETED  Reason=None  Dependency=(null)
    Requeue=1  Restarts=0  BatchFlag=1  Reboot=0  ExitCode=0:0
    RunTime=00:00:02  TimeLimit=7-00:00:00  TimeMin=N/A
    SubmitTime=2017-01-10T11:34:32  EligibleTime=2017-01-10T11:34:32
    StartTime=2017-01-10T11:34:33  EndTime=2017-01-10T11:34:35
    PreemptTime=None  SuspendTime=None  SecsPreSuspend=0
    Partition=kamiak  AllocNode:Sid=login-p1n02:28486
    NodeList=sn2
    NumNodes=1  NumCPUs=1  NumTasks=1  CPUs/Task=1  ReqB:S:C:T=0:0:*:*:
    TRES=cpu=1,mem=6400M,node=1
    Socks/Node=*  NtasksPerN:B:S:C=1:0:*:*  CoreSpec=*  
    MinCPUsNode=1  MinMemoryCPU=6400M  MinTmpDiskNode=0
    Features=(null)  Gres=gpu:tesla:1  Reservation=(null)
    Command=/home/my.NID/job_scripts/test_gpu_job.sh
    WorkDir=/home/my.NID
    StdErr=/home/my.NID/slurm-296129.out
    StdIn=/dev/null
    StdOut=/home/my.NID/slurm-296129.out
```
kamiak$ cat slurm-296129.out
Starting test GPU job on host sn2
deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Tesla K80"
  CUDA Driver Version / Runtime Version 8.0 / 7.5
  CUDA Capability Major/Minor version number: 3.7
  Total amount of global memory: 11440 MBytes
     (11995578368 bytes)
     (13) Multiprocessors, (192) CUDA Cores/MP: 2496 CUDA Cores
     GPU Max Clock rate: 824 MHz (0.82 GHz)
** output snipped for brevity **
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 8.0, CUDA
Runtime Version = 7.5, NumDevs = 1, Device0 = Tesla K80
Result = PASS
Completed test GPU job on host sn2
Running Jobs on Kamiak – Slurm MPI Jobs

A serial job? That’s fine, but how can I use Kamiak for some parallel work, with MPI?

```
#!/bin/bash

#SBATCH --cpus-per-task=1      # Number of CPU cores per task
#SBATCH --ntasks-per-node=5    # Number of tasks per node
#SBATCH --nodes=2-2            # Number of nodes (min-max)
#SBATCH --constraint=ivybridge # Restrict to Ivybridge nodes (optional)
#SBATCH --partition=kamiak     # Partition/Queue to use
#SBATCH --output=%j.out        # Output (STDOUT)
#SBATCH --error=%j.err         # Error (STDERR)

echo "Starting test MPI job on host $HOSTNAME"

echo "I am job $SLURM_JOBID running on nodes $SLURM_JOB_NODELIST"

# Intel module includes MPI
# gcc with openmpi, mpich, or mvapich2 are also available
module load intel/xe_2016_update3

# My program I compiled on Kamiak
cd $HOME/mpiicc_test
srun ./hello_mpiicc

echo "Completed test MPI job on host $HOSTNAME"

kamiak$ sbatch test_mpi_job.sh
Submitted batch job 613498
```
Running Jobs on Kamiak – Slurm MPI

```
amiak$ squeue -u my.NID
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
613498  kamiak test_mpi my.NID R       0:02      2 cn[3-4]
amiak$ squeue -u my.NID
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
amiak$ cat 613498.out
Starting test MPI job on host cn3
I am job 613498 running on nodes cn[3-4]
Hello world from processor cn3, rank 0 out of 10 processors
Hello world from processor cn4, rank 5 out of 10 processors
Hello world from processor cn3, rank 1 out of 10 processors
Hello world from processor cn3, rank 2 out of 10 processors
Hello world from processor cn3, rank 3 out of 10 processors
Hello world from processor cn3, rank 4 out of 10 processors
Hello world from processor cn4, rank 6 out of 10 processors
Hello world from processor cn4, rank 7 out of 10 processors
Hello world from processor cn4, rank 8 out of 10 processors
Hello world from processor cn4, rank 9 out of 10 processors
Completed test MPI job on host cn3
```
Running Jobs on Kamiak – Slurm Arrays

- Job arrays offer a mechanism for submitting and managing collections of similar jobs quickly.

```bash
#!/bin/bash
#SBATCH --ntasks=5            # Number of tasks
#SBATCH --cpus-per-task=1    # Number of CPU cores per task
#SBATCH --partition=kamiak   # Partition/Queue to use
#SBATCH --array=0-4

echo "Starting test array job on host $HOSTNAME"

echo "I am Slurm job ${SLURM_JOB_ID}, array job ${SLURM_ARRAY_JOB_ID},
and array task ${SLURM_ARRAY_TASK_ID}"

export my_files=("data0.txt", "data1.txt", "data2.txt", "data3.txt",
"data4.txt")
# /path/to/some/app --input=${my_files[$SLURM_ARRAY_TASK_ID]}

echo "Completed test array job on host $HOSTNAME"

kamiak$ sbatch test_array_job.sh
Submitted batch job 471096
```
Running Jobs on Kamiak – Slurm Arrays

```
kamiak$ squeue -u my.NID
JOBID  PARTITION NAME     USER    ST TIME  NODES  NODELIST(REASON)
471096_0 kamiak    test_arr my.NID  R  0:01  1      sn2
471096_1 kamiak    test_arr my.NID  R  0:01  1      sn2
471096_2 kamiak    test_arr my.NID  R  0:01  1      sn5
471096_3 kamiak    test_arr my.NID  R  0:01  1      sn5
471096_4 kamiak    test_arr my.NID  R  0:01  1      sn5

kamiak$ cat slurm-471096_0.out
Starting test array job on host sn2
I am Slurm job 471097, array job 471096, and array task 0
Completed test array job on host sn2

kamiak$ cat slurm-471096_1.out
Starting test array job on host sn2
I am Slurm job 471098, array job 471096, and array task 1
Completed test array job on host sn2

kamiak$ cat slurm-471096_2.out
Starting test array job on host sn5
I am Slurm job 471099, array job 471096, and array task 2
Completed test array job on host sn5
```
Running Jobs on Kamiak – Interactive Jobs

- idev is a tool used to create an interactive job
- idev supports a limited set of Slurm options (see --help)
- SSH to a compute node is also possible if you have a job running on it

```
kamiak$ idev
Requesting 1 node(s) from kamiak partition
1 task(s)/node, 1 cpu(s)/task
Time: 0 (hr) 60 (min).
Submitted batch job 470692
Job is pending. Please wait. 0(s)
JOBID=470692 begin on cn13
--> Creating interactive terminal session (login) on node cn13.
--> You have 0 (hr) 60 (min).
--> Assigned Host List : /tmp/idev_nodes_file_my.NID

cn13$ echo "Hello"
Hello

Removing job 470692
```
Running Jobs on Kamiak – Other Slurm Options

- Time limit
  - \#SBATCH --time=1-00:00 # D-HH:MM
- Delayed start
  - \#SBATCH --begin=now+1hour
  - \#SBATCH --begin=16:00
- Require a specific hardware constraint
  - \#SBATCH --constraint=haswell
- Depend on another job
  - \#SBATCH --dependency=after:$/job_id
- Job name
  - \#SBATCH --job-name=test

See also:
man sbatch
1. My job never ran and it didn’t create output files.
   • Check in the directory where you submitted the job, by default Slurm will place output files there. If you set a specific output file, did the directory it is in exist before the job was submitted? Do you have write access to that directory?

2. “Requested node configuration is not available”
   • Either your resource request is wrong (e.g. asks for more cores per node than exist) or the nodes with enough resources are offline (check `sinfo`).

3. My queue/partition is busy and my jobs are waiting too long.
   • If possible, use smaller jobs which are easier for Slurm to find resources for.
   • Switch to a partition with available resources, such as backfill.

4. My GPU job won’t run, it says there are no CUDA capable devices.
   • Ensure you requested a GPU with the --gres option of `sbatch`.

5. My jobs get cancelled with a memory error.
   • Use the --mem or --mem-per-cpu options of `sbatch` to request more memory.
Running Jobs on Kamiak – Being a Good User

Kamiak is a shared cluster for all of WSU and your access to it is a privilege. Its resources are finite and care must be taken to ensure its continued usefulness for yourself and the greater research community.

Do

- Use /scratch and /local for your computational storage needs when possible
- Cite Kamiak in your work
- Utilize all resources of a node when possible (20 core job on one node is generally better than using 10 cores on two nodes)
- Report issues via Kamiak’s Service Desk
- Utilize checkpointing in your workflow
- Abide by Kamiak’s End User License Agreement and WSU policies

Don’t

- Run intensive workloads (e.g. compiling) on a login node, use idev to get a compute node to work on
- Undersubscribe: Use more resources than you reserved, causing (often severe) performance problems with other jobs (and your own)
- Oversubscribe: Reserve more resources than you use, causing them to be idle but unavailable for other jobs
- Submit thousands of jobs – use job arrays or an alternative
- “Game” or otherwise abuse the scheduler (Slurm)
- Give your password to anyone, ever
What if I want to install my own software?
An example of Installing software package: libxc-2.2.2

$ wget https://www.cp2k.org/static/downloads/libxc-2.2.2.tar.gz
$ tar -zxvf libxc-2.2.2.tar.gz
$ cd libxc-2.2.2
$ more README  # in some cases it is called README.md
$ more INSTALL  # you must read these files before
               # any attempt of installation
$ idev -c 4 --time=4:00:00  # request a idev session
$ ./configure --help  # to see all options available

Note the following statement in README file:
“...... Furthermore, the most important contents of the src directory are:
xc.h  - main header file with all external definitions
util.h - header file with internal definitions
*.f90 *.F90 xc_f.c  - Fortran 90 interface .......

This suggests that we need to use C and Fortran compilers
Tips and Tricks on Software Installs Cont’d

$ mkdir ../libxc-install  # destination dir. for package
$ ./configure --prefix=/home/username/libxc-install --enable-shared CC=icc FC=ifort
$ make -j 4  # compile
$ make check  # check the compilation
$ make install  # install into the destination dir.
$ make clean  # remove files created in compilation
$ make distclean  # remove files created by ./configure
$ ls ../libxc-install  # view installation directory, should have: bin include lib
$ ls ../libxc-install/*  # viewing all sub-directories

Set PATH and LD_LIBRARY_PATH in your .bash_profile file by putting in the following lines:

export PATH=/home/username/libxc-install/bin:$PATH
export LD_LIBRARY_PATH=/home/username/libxc-install/bin:$LD_LIBRARY_PATH
How to become an investor

• Everyone at WSU has access to the backfill queue, 10Gb of storage in /home, and any storage their Unit may allocate to them

• If you need more → become an investor

Service Catalogue

Standard Compute Nodes

<table>
<thead>
<tr>
<th>Processors</th>
<th>Cores per Node</th>
<th>Memory</th>
<th>Local Disk</th>
<th>Networking</th>
<th>Budgetary Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual Intel E5-2660v3</td>
<td>20</td>
<td>128GB 2133MT/s DDR4 ECC</td>
<td>1 x 400GB SSD</td>
<td>2 x 10GB SFP+ single-port FDR infiniband</td>
<td>$7,840 / 5yrs</td>
</tr>
<tr>
<td>Dual Intel E5-2660v3</td>
<td>20</td>
<td>256GB 2133MT/s DDR4 ECC</td>
<td>1 x 400GB SSD</td>
<td>2 x 10GB SFP+ single-port FDR infiniband</td>
<td>$9,550 / 5yrs</td>
</tr>
<tr>
<td>Dual Intel E5-2660v3</td>
<td>20</td>
<td>512GB 2133MT/s DDR4 ECC</td>
<td>1 x 400GB SSD</td>
<td>2 x 10GB SFP+ single-port FDR infiniband</td>
<td>$12,970 / 5yrs</td>
</tr>
</tbody>
</table>
Special Compute Nodes
If you are interested in purchasing a non-standard compute resource, i.e. an accelerator equipped node or a large memory system, please submit a request to our Service Desk to get an up to date budgetary estimate. For reference purposes, below are the budgetary estimates for the specialty nodes currently installed on Kamiak.

<table>
<thead>
<tr>
<th>Processors</th>
<th>Cores per Node</th>
<th>Memory</th>
<th>Accelerator</th>
<th>Local Disk</th>
<th>Networking</th>
<th>Budgetary Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quad Intel E7-4880v2</td>
<td>60</td>
<td>2TB 1600MT/s DDR3 ECC</td>
<td>none</td>
<td>600 GB 10k RPM SAS 6Gbps</td>
<td>2 x 10GB SFP+ dual-port FDR infiniband</td>
<td>$57,300 / 5yrs</td>
</tr>
<tr>
<td>Dual Intel E5-2670v3</td>
<td>24</td>
<td>256GB 2133MT/s DDR4 ECC</td>
<td>Dual Tesla K80 GPU (9984 CUDA cores)</td>
<td>2 x 60GB 6 GBps SSD 2 x 1TB 7.2k rpm SATA</td>
<td>2 x 10GB SFP+ dual-port FDR infiniband</td>
<td>$16,500 / 5yrs</td>
</tr>
<tr>
<td>Dual Intel E5-2670v3</td>
<td>24</td>
<td>256GB 2133MT/s DDR4 ECC</td>
<td>256GB 2133MT/s DDR4 ECC Dual Xeon Phi 5110P coprocessors (120 physical cores)</td>
<td>2 x 60GB 6 GBps SSD 2 x 1TB 7.2k rpm SATA</td>
<td>2 x 10GB SFP+ dual-port FDR infiniband</td>
<td>$16,000 / 5yrs</td>
</tr>
</tbody>
</table>
Storage

All users are provided with a 10GB home directory thanks to the generous contributions of our initial investors. Users may also take advantage of the performant scratch file space to temporarily store data. If you would like additional long term “Project Storage”, this can be purchased through the Service Desk in either 100GB or 500GB increments.

<table>
<thead>
<tr>
<th>Storage Type</th>
<th>Usable Storage Amount</th>
<th>Disk Type</th>
<th>Transfer Rate</th>
<th>Budgetary Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Storage</td>
<td>100GB unit</td>
<td>7.2k RPM</td>
<td>6GBps</td>
<td>$12 /year</td>
</tr>
<tr>
<td>Project Storage</td>
<td>500GB unit</td>
<td>7.2k RPM</td>
<td>6GBps</td>
<td>$56 /year</td>
</tr>
</tbody>
</table>
• We will be sending out a qualtrix survey to get your feedback about this training event (its our first one and we want to get better!)

• An advanced training session will be offered at the end of the semester – let us know in the survey what topics would be of interest

• Other ways to learn more and participate in Kamiak governance:
  - Share your thoughts and ideas with the Kamiak Executive User Group (members – Scott Beckman, Ananth K., Stephen Ficklin, Thomas Badman (student rep))
  - Organize! We are interested in starting a student HPC group...see the amazing things that have been done at the UW one:
    - http://students.washington.edu/hpcc/
Additional Info Beyond the Training Session
Things that need repeating

• Sometimes you may have repetitive tasks – use cron!
  • copying/Synchronizing your files every day at midnight
  • Reporting disk usage to you at regular intervals
  • Checking job outputs on the hour

• Every user has a crontab text file as part of their account, it is modified using the crontab command

• username: man crontab to see the full list of commands

• username: crontab –l to list existing crontab file entries

• Username: crontab –e to edit the existing crontab file (opens up a text editor)
The information you must include is (in order of appearance):

1. A number (or list of numbers, or range of numbers), \( m \), representing the minute of the hour;
2. A number (or list of numbers, or range of numbers), \( h \), representing the hour of the day;
3. A number (or list of numbers, or range of numbers), \( dom \), representing the day of the month;
4. A number (or list, or range), or name (or list of names), \( mon \), representing the month of the year;
5. A number (or list, or range), or name (or list of names), \( dow \), representing the day of the week; and
6. \textit{command}, which is the command to be run, exactly as it would appear on the command line.

\textit{cron} examines crontab entries once every minute.

The time and date fields are:

<table>
<thead>
<tr>
<th>field</th>
<th>allowed values</th>
</tr>
</thead>
<tbody>
<tr>
<td>minute</td>
<td>0-59</td>
</tr>
<tr>
<td>hour</td>
<td>0-23</td>
</tr>
<tr>
<td>day of month</td>
<td>1-31</td>
</tr>
<tr>
<td>month</td>
<td>1-12 (or names; see example below)</td>
</tr>
<tr>
<td>day of week</td>
<td>0-7 (0 or 7 is Sunday, or use names; see below)</td>
</tr>
</tbody>
</table>
The default crontab file looks like this:

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow  command
```

These lines all start with a # because they are comments; they are ignored by cron, and are just there for you to read.
You don’t have to run the command in the crontab – you can have it point to a script

We want our job to run at 5 A.M., which would be minute 0, hour 5, every day of the month, every month, every day of the week. We need to add a line to the bottom of the file which looks like this:

```
0 5 * * * /home/myname/scripts/do-every-day.sh
```