Welcome to Kamiak

2/26/18 Training Session

Aurora Clark, CIRC Director
Peter Mills, Computational Scientist
Jeff White, Systems Engineer
What is Kamiak

• Kamiak is a **cluster** of hundreds of computing nodes
• Each **node** is a computer like your laptop, but with many cores
• Applications can run in **parallel** over many cores and nodes
• **Speeds up** solving large problems
What Does Kamiak Provide?

• Each user has 10GB in their home directory in `/home/your.name`

• Extra storage is available in `/data/investorOrCollege` (/data/clark)
  - CIRC/ITS Service Center for Storage
    ~90 TB purchased by faculty investors
  - 5-year storage purchased by colleges
    CAHNRS ~200 TB    CAS ~100 TB    VCEA ~25 TB    OR ~13 TB
Stats about Kamiak

- CPU Cores: 2,608
- Memory: 24 TB
- Storage: 1.2 PB
- CUDA Cores: 109,824

As of February 2018
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
Specialized training events
What you will learn today

• Logging into Kamiak
• Basic Linux commands
• How to transfer files to and from Kamiak
• How to run jobs on Kamiak
  - Submit jobs
  - Run an interactive compute session
  - Use job arrays
  - Find and load software modules
• How to get help
• Best practices to being a good user
• How to invest in Kamiak (nodes or storage)
Several ways to log into Kamiak

- Linux or Mac: `ssh your.name@kamiak.wsu.edu`
- Windows: PuTTY, Cygwin, MobaXterm, WSL, etc.

See also: https://hpc.wsu.edu/users-guide/terminal-ssh
• Follow along:
  - Log into Kamiak – Linux or Mac
    • Terminal >> New Window
    • `ssh your.name@kamiak.wsu.edu`

See also: https://hpc.wsu.edu/users-guide/terminal-ssh
Logging into Kamiak - Windows - Putty

- **Follow along:**
  - Log into Kamiak – Windows
  - PuTTY, MobaXterm, Cygwin, WSL, etc.

See also: https://hpc.wsu.edu/users-guide/terminal-ssh

Download from: https://www.putty.org
• **Follow along:**
  - Log into Kamiak – Windows
  - **PuTTY**, MobaXterm, Cygwin, WSL, etc.

Be sure to use “your.name” and not “your.name@kamiak.wsu.edu”
Basic Linux Commands

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

FILE COMMANDS

- `ls -al`: Display all information about files/directories
- `pwd`: Show the path of current directory
- `mkdir directory-name`: Create a directory
- `rm file-name`: Delete file
- `rm -r directory-name`: Delete directory recursively
- `rm -f file-name`: Forcefully remove file
- `rm -rf directory-name`: Forcefully remove directory recursively
- `cp file1 file2`: Copy file1 to file2
- `cp -r dir1 dir2`: Copy dir1 to dir2, create dir2 if it doesn’t exist
- `mv file1 file2`: Rename source to dest/move source to directory
- `ln -s /path/to/file-name link-name`: Create symbolic link to file-name
- `touch file`: Create or update file
- `cat > file`: Place standard input into file
- `more file`: Output contents of file
- `head file`: Output first 10 lines of file
- `tail file`: Output last 10 lines of file
- `tail -f file`: Output contents of file as it grows starting with the last 10 lines
- `gpg -c file`: Encrypt file
- `gpg file.gpg`: Decrypt file
- `wc`: Print the number of bytes, words, and lines in files
- `xargs`: Execute command lines from standard input
Basic Linux Commands

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

```bash
[auclark@login-p1n02 ~]$ ls
1cluster1043-MP2.nw  3cluster1051-MP2.nw  lifetime-codes
1cluster1048-MP2.nw  3cluster575-MP2.nw  one-water-clusters.tar.gz
1cluster1051-MP2.nw  3cluster590-MP2.nw  polyhedron-example.tgz
1cluster575-MP2.nw   3cluster680-MP2.nw  slurm-474836.out
1cluster590-MP2.nw   3cluster696-MP2.nw  slurm-idv39008.o82900
1cluster680-MP2.nw   3cluster724-MP2.nw  solB1830.xyz
1cluster696-MP2.nw   3cluster727-MP2.nw  solB592.xyz
1cluster724-MP2.nw   3cluster997-MP2.nw  test
1cluster727-MP2.nw   chemical-networks.R  test.xyz
1cluster997-MP2.nw   ChemNetworks-2.2.updates.tar.gz  three-water-clusters.tar.gz
1solvent-example.tgz  codes  VASP
2solvent-example.tgz  forAbdullah  water-in-ethanol
3cluster1043-MP2.nw  home
3cluster1048-MP2.nw  integrated-solvent-wrap-v4.f90
```
Basic Linux Commands

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

<table>
<thead>
<tr>
<th>DIRECTORY TRAVERSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd ..</td>
</tr>
<tr>
<td>cd</td>
</tr>
<tr>
<td>cd /test</td>
</tr>
</tbody>
</table>

• There are many tutorials online that can help you solidify your linux command expertise
Text Editors

• If you want to open and edit files, you need to also choose a text editor to use:

• There are many tutorials on different text editors as well
Transferring Files to and from Kamiak

- There are several ways to transfer and synchronize files across different computers

  - **Transfer**
    - Linux, Mac: `scp` (secure copy), `sftp` (secure file transfer protocol)
    - Windows: **WinSCP**

  - **Synchronize**
    - Linux or Mac only: **rsync**
Transferring Files - Mac/Linux - scp

- Lets transfer some files from Kamiak to our machine
- Follow along:
  - Terminal >> New Window
  - cd
  - scp your.name@kamiak.wsu.edu:~/.bashrc .

From my home directory on Kamiak
To current directory on my machine

mac% cd
mac% scp peter.mills@kamiak.wsu.edu:~/.bashrc .
peter.mills@kamiak.wsu.edu's password:
.bashrc 100% 2503 1.0MB/s 00:00
mac% ls -l .bashrc
-rw------- 1 phmills staff 2503 Feb 8 13:37 .bashrc
• Now from my machine back to Kamiak, renaming it

• Follow along:
  • `mv .bashrc newbash`
  • `scp newbash your.name@kamiak.wsu.edu:~`

```
mac% mv .bashrc newbash
mac% scp newbash peter.mills@kamiak.wsu.edu:~
peter.mills@kamiak.wsu.edu's password:
newbash 100% 2503  397.3KB/s  00:00
```
Let's transfer an entire folder from Kamiak to our machine.

Follow along:

- `scp -r your.name@kamiak.wsu.edu:/opt/apps/samples/training .`

From a folder on Kamiak
Recursive copy – will go into a directory and get all files
Into the present working directory

Warning: a trailing slash / on the source may insert just the folder members, not the folder.

mac% scp -r peter.mills@kamiak.wsu.edu:/opt/apps/samples/training.
peter.mills@kamiak.wsu.edu's password:
hybrid_hello.sh 100% 1267 26.1KB/s 00:00
mac% ls -l
drwxrwxr-x 17 phmills staff 578 Feb 8 14:02 training/
Transferring Files - WinSCP

- Transfer only – follow along
  - scp (secure copy), sftp (secure file transfer protocol)
  - Windows: WinSCP

Download from: https://winscp.net
Transferring Files - WinSCP

- Transfer only
- scp (secure copy), sftp (secure file transfer protocol)
- Windows: WinSCP

Password:

- Remember password for this session

OK  Cancel  Help
Transferring Files - WinSCP
Transferring Files - WinSCP
Transferring Files - \textit{rsync}

- There are several ways to transfer and synchronize files across different computers

- \textbf{Synchronize – follow along}
  - Linux or Mac only
  - \texttt{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 \texttt{rsync} commands

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

- 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:~/
```

- Current file and location
- Archive, verbose, compress
- Destination location
Transferring Files - rsync

• 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

sent 1551 bytes  received 42 bytes  167.68 bytes/sec
total size is 3987  speedup is 2.50
```
Transferring Files - *rsync*

- Example uses of rsync
  - Copying data **from your local computer to your home directory** on Kamiak
  - A directory

```
$ rsync -avz codes-n-scripts/ auclark@kamiak.wsu.edu:~/
```

- Trailing slash vs. none → contents only vs. dir+contents
What You Have Learned so Far

• Now – you should be able to (and practice):
  - Logging into Kamiak
  - Basic Linux commands to navigate folders, move files, etc.
  - Moving files to and from Kamiak
  - Decide and learn what text editor you want to use

• What’s next → submitting jobs for execution
Running Jobs on Kamiak – Key Concepts

• Kamiak is a **cluster** of computing nodes

• Each **node** is a computer like your laptop, but with many cores

• Each **core** (processor), can execute a different program (**task**)  
  • 20 cores can do 20 different things at once

• Nodes are grouped into **partitions**  
  • Each partition holds nodes owned by a PI or college

• You submit a **job** asking for a number of cores

• Job gets added to a **queue** to wait for resources  
  • One queue for each partition

• **Slurm job scheduler** decides:  
  • Who goes first  
  • Who gets what  
  • Who gets bumped

• **Cores (cpu’s) are the unit of resource allocation**  
  • Each core can only be used by 1 job at any time  
  • Cores can be allocated across parts of several nodes

• Kamiak does **not** parallelize your program  
  • It just runs your program on the cores and nodes it assigns it to
• Kamiak's nodes are grouped into partitions owned by each investor
• All nodes also belong to a shared backfill partition (called "kamiak")
  - **Backfill can be used by all users**
• Investors have priority access to the nodes they own
  - **You will get bumped if an investor’s job can’t find enough idle cores to run, and yours would allow it to start**

• Resource limits for backfill jobs
  - Currently: 120 CPU cores (~ 6 standard compute nodes)
How to Run Jobs on Kamiak

There are two ways to run jobs on Kamiak

• **sbatch myJob.sh**  *Batch job submission*
  - Says what partition to submit to, default is backfill queue
  - Says what resources your job needs (CPU cores, memory, GPU)
  - Says what program to run

• **idev**  *Interactive session on compute node*
  - Puts you on a compute node
  - Just type in commands and see them executed

So I can login to the cluster, but how do I use it?

Do not run compute or compile jobs on login nodes, use **sbatch** or **idev** instead
**How Job Submission Works**

**What**

- `sbatch myJob.sh`
  - Waits until resources available
  - Allocates cores on nodes
  - Runs Job script on first node

**Where**

- `myJob.sh`
  - `#SBATCH --nodes=2`
  - `#SBATCH --ntasks-per-node=2`
  - `#SBATCH --ntasks=4`
  - `#SBATCH --cpus-per-task=1`

- `srun myProgram`
  - Runs myProgram once for each task (total 4 times)
Submitting Batch Jobs to Kamiak

• Log into Kamiak
  
  `ssh your.name@kamiak.wsu.edu`
  
  `cp –r /opt/apps/samples/training .`
  
  `cd training`
  
  `. setup.sh`  # One-time setup only for this training

• Create/edit a job script
  
  `cat myJob.sh`

• Submit the job script to the job queue
  
  `sbatch myJob.sh`  # sbatch --test-only myJob.sh

• View the job queue
  
  `squeue -u your.name`  # Shows pending and running jobs
  
  `squeue –j jobNumber`

• Cancel the job

  `scancel jobNumber`  # Cancel job

• View past and active jobs

  `sacct –S 2/26/18 -u your.name`  # Past job history
  
  `scontrol show job jobNumber`  # Job details
Batch Job Script

```
#!/bin/bash

#SBATCH --partition=kamiak            # Partition/Queue to use
#SBATCH --job-name=myJob             # Job name
#SBATCH --output=myJob_%j.out         # Output file (stdout)
#SBATCH --error=myJob_%j.err          # Error file (stderr)
#SBATCH --mail-type=ALL              # Email notification: BEGIN,END,FAIL,ALL
#SBATCH --mail-user=your.name@wsu.edu  # Email address for notifications
#SBATCH --time=7-00:00:00             # Wall clock time limit Days-HH:MM:SS

#SBATCH --nodes=1                     # Number of nodes (min-max) Where (layout)
#SBATCH --ntasks-per-node=2          # Number of tasks per node (max)
#SBATCH --ntasks=2                    # Number of tasks (processes) What (cpus)
#SBATCH --cpus-per-task=1            # Number of cores per task (threads)

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

module load python
srun python helloWorld.py          # Each srun is a job step, and spawns -ntasks
                                     # Each task runs this program (total 2 times)

echo "Completed job on node $HOSTNAME"
```
Submitting, Viewing, and Canceling Batch Jobs

```
kamiak$ sbatch myJob.sh
Submitted batch job 1160269

kamiak$ squeue -u peter.mills
  JOBID   PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
  1160269  kamiak     myJob     peter.mi PD       0:00      1 (Priority)

kamiak$ squeue -u peter.mills
  JOBID   PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
  1160269  kamiak     myJob     peter.mi  R       0:02      1 cn93

kamiak$ squeue -j 1160269
  JOBID   PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
  1160269  kamiak     myJob     peter.mi  R       0:17      1 cn93

kamiak$ cat myJob_1160269.out
I am job 1160269 running on nodes cn93
Hello World! from cn93
Hello World! from cn93

kamiak$ scancel 1160269
```
### Viewing Job History and Details

```bash
kamiak$ sacct -S 2/11/18 -u peter.mills

<table>
<thead>
<tr>
<th>JobID</th>
<th>JobName</th>
<th>Partition</th>
<th>Account</th>
<th>AllocCPUS</th>
<th>State</th>
<th>ExitCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1160269</td>
<td>myJob</td>
<td>kamiak</td>
<td>circ</td>
<td>2</td>
<td>CANCELLED</td>
<td>0:0</td>
</tr>
<tr>
<td>1160269.bat+</td>
<td>batch</td>
<td>circ</td>
<td>2</td>
<td>CANCELLED</td>
<td>0:20</td>
<td></td>
</tr>
<tr>
<td>1160269.0</td>
<td>python</td>
<td>circ</td>
<td>2</td>
<td>CANCELLED</td>
<td>0:20</td>
<td></td>
</tr>
</tbody>
</table>

kamiak$ scontrol show job 1160269
JobId=1160269  JobName=myJob
  UserId=peter.mills(8316)  GroupId=its_p_sys Ur_kam-circ(7060)  MCS_label=N/A
  Priority=4294967295  Nice=0  Account=circ  QOS=normal
  JobState=TIMEOUT  Reason=TimeLimit  Dependency=(null)
  Requeue=1  Restarts=0  BatchFlag=1  Reboot=0  ExitCode=0:15
  RunTime=00:01:20  TimeLimit=00:01:00  TimeMin=N/A
  Partition=kamiak  AllocNode:Sid=login-p1n02:33414
  NodeList=cn93  BatchHost=cn93
  NumNodes=1  NumCPUs=2  NumTasks=2  CPUs/Task=1  ReqB:S:C:T=0:0:*:*
  TRES=cpu=2,mem=16G,node=1
  Gres=(null)  Reservation=(null)
  Command=/home/peter.mills/training/myJob.sh
  WorkDir=/home/peter.mills/training
  StdErr=/home/peter.mills/training/myJob_1160269.err
  StdIn=/dev/null
  StdOut=/home/peter.mills/training/myJob_1160269.out
```

Viewing Information about the Cluster

- What partitions and nodes are available
  
  `sinfo -a | more`  # Availability (alloc, idle, mix)

- View all running and queued jobs
  
  `squeue -a | more`  # Queued jobs for all partitions

- View node details
  
  `scontrol show node cn93`  # Amount of memory, cpus, GPU's
### Viewing Available Partitions

**kamiak$ sinfo -a | more**

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

See also:  
https://hpc.wsu.edu/queue-list
### Viewing Job Submissions

```
[-terminal]
kamiak$ `squeue -a | more`

<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</th>
<th>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>
<td></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>
<td></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>
<td></td>
</tr>
<tr>
<td>471077</td>
<td>popgenom</td>
<td>BFS.3L.i</td>
<td>joel.t.n</td>
<td>R</td>
<td>03:34:38</td>
<td>1</td>
<td>cn7</td>
<td></td>
</tr>
<tr>
<td>471078</td>
<td>popgenom</td>
<td>BFS.2R.i</td>
<td>joel.t.n</td>
<td>R</td>
<td>03:34:03</td>
<td>1</td>
<td>cn29</td>
<td></td>
</tr>
<tr>
<td>471079</td>
<td>popgenom</td>
<td>BFS.2L.i</td>
<td>joel.t.n</td>
<td>R</td>
<td>03:33:14</td>
<td>1</td>
<td>cn29</td>
<td></td>
</tr>
<tr>
<td>473678</td>
<td>kamiak</td>
<td>cle_6cma</td>
<td>tung.ngu</td>
<td>R</td>
<td>05:07:11</td>
<td>1</td>
<td>cn7</td>
<td></td>
</tr>
<tr>
<td>473722</td>
<td>beckman</td>
<td>hydr-vra</td>
<td>hong.zho</td>
<td>R</td>
<td>04:28:26</td>
<td>2</td>
<td>cn[43-44]</td>
<td></td>
</tr>
<tr>
<td>473726</td>
<td>beckman</td>
<td>occ-1na3</td>
<td>hong.zho</td>
<td>R</td>
<td>03:37:54</td>
<td>2</td>
<td>cn[52-53]</td>
<td></td>
</tr>
<tr>
<td>473727</td>
<td>kamiak</td>
<td>dbl_Pt21</td>
<td>mareike.</td>
<td>R</td>
<td>02:23:09</td>
<td>2</td>
<td>cn[3-4]</td>
<td></td>
</tr>
</tbody>
</table>
```
kamiak$ scontrol show node cn93
NodeName=cn93  Arch=x86_64  CoresPerSocket=14
    CPUAlloc=20  CPUErr=0  CPUTot=28  CPULoad=22.11
AvailableFeatures=broadwell,e5-2660-v4-2.00ghz,avx2
ActiveFeatures=broadwell,e5-2660-v4-2.00ghz,avx2
Gres=(null)
NodeAddr=cn93  NodeHostName=cn93  Version=17.02
OS=Linux  RealMemory=128658  AllocMem=128000  FreeMem=7234  Sockets=2  Boards=1
State=MIXED  ThreadsPerCore=1  TmpDisk=64329  Weight=62000  Owner=N/A  MCS_label=N/A
Partitions=clark,adam,storfer,bio534,kamiak
BootTime=2017-11-28T09:00:39  SlurmdStartTime=2017-11-28T09:01:34
CfgTRES=cpu=28,mem=128658M
AllocTRES=cpu=20,mem=125G
CapWatts=n/a
CurrentWatts=125  LowestJoules=15477  ConsumedJoules=1017346137
ExtSensorsJoules=n/s  ExtSensorsWatts=0  ExtSensorsTemp=n/s
Interactive Jobs

- **idev** creates an interactive session on a compute node
  - Same options as **sbatch**
  - Can also **ssh** to a compute node if you have a job allocated on it

**Examples:**

```
$ idev -N 1 --ntasks-per-node=2 -t 360
Idev interactively runs commands on a compute node.
See 'man salloc' for idev options to reserve a job allocation.
To use a GPU within idev, you need to use 'srun yourCommand'.
To run python interactively with a GPU, use 'srun python -i'.
Recommend using 'srun -I' to launch a task without hanging.
Default time is 60 minutes. Use '-t yourMinutes' to override.
salloc: Granted job allocation 1160832
Allocated nodes: cn32

$ module avail

$ module load python/2.7.10

$ module list
Currently Loaded Modules:
  1) StdEnv
  2) intel/xe_2016_update3
  3) workspace_maker/master
  4) python/2.7.10
```
Interactive Jobs

cn32$ python -i
Python 2.7.10 (default, Jul 19 2017, 16:30:38)
[GCC 6.1.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hello World!"
Hello World!
>>> exit()

cn32$ srun -I python helloWorld.py --nowait
Hello World! from cn32
Hello World! from cn32

cn32$ exit
exit
salloc: Relinquishing job allocation 1160832

kamiak$ 

Use `srun -I` to avoid hanging if resources are not available
Job Arrays

- A **job array** submits a collection of jobs with one script
- Runs the script multiple times with a different **index**
  - Each job array step is scheduled as an individual job
  - Each job array step is allocated the same requested resources
- Can be used to run the same program over **segmented data**
  - Each data file is named with a different index suffix (*data_1.txt*)

```
#SBATCH --array=0-2:1  # Index 0,1,2 in steps of 1
srun myProgram inputs/data_${SLURM_ARRAY_TASK_ID}.txt
```

Please use job arrays instead of submitting hundreds of individual jobs
Different Ways to Use Job Arrays

- You can use a job array index in different ways

- As an index into a set of names or numbers
  - `export names=("0.txt" "1.txt" "2.txt" "3.txt" "4.txt")`
  - `export names=$(ls myFolder/inputs*.txt)` (for known --array total)
  - `srun myProgram ${names[$SLURM_ARRAY_TASK_ID]}`

- As a suffix to a folder or file name
  - `srun myProgram myFolder_${SLURM_ARRAY_TASK_ID}/d.txt`

- As an index into a script to run (only for single core jobs)
  - `srun bash myProgram_${SLURM_ARRAY_TASK_ID}.sh`

- Any combination of the above
#!/bin/bash

#SBATCH --partition=kamiak # Partition/Queue to use
#SBATCH --job-name=myJobArray # Job name
#SBATCH --output=output/myJobArray_%A_%a.out # Output filename, jobid_index
#SBATCH --error=output/myJobArray_%A_%a.err # Error filename, jobid_index
#SBATCH --time=7-00:00:00 # Wall clock time limit Days-HH:MM:SS
#SBATCH --mail-type=ALL # Email notification: BEGIN,END,FAIL,ALL
#SBATCH --mail-user=your.name@wsu.edu # Email address for notifications
#SBATCH --array=0-2:1 # Number of jobs, in steps of 1

#SBATCH --nodes=1 # Number of nodes (min-max)
#SBATCH --ntasks-per-node=1 # Number of tasks per node (max)
#SBATCH --cpus-per-task=1 # Number of cores per task (threads)
#SBATCH --mem-per-cpu=8G # Memory per core (gigabytes)

# Runs this job 3 times, with index SLURM_ARRAY_TASK_ID as 0,1,2
# Split your data into 3 files, name them data_0.txt, data_1.txt, data_2.txt
# Each job array step is scheduled as an individual job
# Each job array step is allocated the above resources (cores, memory)

echo "Starting job array $SLURM_ARRAY_TASK_ID on host $HOSTNAME"

module load python
srun python helloWorld.py "inputs/data_${SLURM_ARRAY_TASK_ID}.txt"

echo "Completed job array $SLURM_ARRAY_TASK_ID on host $HOSTNAME"
Running Job Arrays

```

kamiak$ sbatch jobArray.sh
Submitted batch job 1161089

kamiak$ squeue -u peter.mills

<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>1161089_0</td>
<td>kamiak</td>
<td>myJobArr</td>
<td>peter.mills</td>
<td>R</td>
<td>0:01</td>
<td>1</td>
<td>cn32</td>
</tr>
<tr>
<td>1161089_1</td>
<td>kamiak</td>
<td>myJobArr</td>
<td>peter.mills</td>
<td>R</td>
<td>0:01</td>
<td>1</td>
<td>cn32</td>
</tr>
<tr>
<td>1161089_2</td>
<td>kamiak</td>
<td>myJobArr</td>
<td>peter.mills</td>
<td>R</td>
<td>0:01</td>
<td>1</td>
<td>cn32</td>
</tr>
</tbody>
</table>

kamiak$ cat output/myJobArray*out
Starting job array 0 on host cn32
Processing data file: inputs/data_0.txt
Starting job array 1 on host cn32
Processing data file: inputs/data_1.txt
Starting job array 2 on host cn32
Processing data file: inputs/data_2.txt

kamiak$ scancel 1161089
```
Using Available Software on Kamiak

- Kamiak has many software packages already installed and available for use
- `module` commands set up the environment for your software to run
  - Adds programs to your executable search path
  - Pulls in any software dependencies

```sh
module load python  # load latest version
module load python3/3.5.0  # load specific version
module list  # see loaded modules
module whatis anaconda3  # see what a module does
module help wrf  # see help for a module
module avail  # available compatible modules
module spider  # see all modules
module unload python3  # unload a module
module purge  # unload all modules
which python  # see that python is in your path
printenv PATH  # see effects of loading modules
printenv LD_LIBRARY_PATH
```
## Available Software Modules on Kamiak

### Compilers

<table>
<thead>
<tr>
<th>StdEnv</th>
<th>gcc/5.2.0</th>
<th>intel/xe_2016_update2</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc/4.9.3</td>
<td>gcc/6.1.0 (D)</td>
<td>intel/xe_2016_update3 (L,D)</td>
</tr>
</tbody>
</table>

### intel/xe_2016_update3 Software

<table>
<thead>
<tr>
<th>bazel/0.4.2</th>
<th>fftw/3.3.4</th>
<th>netcdf/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmake/3.7.2</td>
<td>gromacs/2016.2_mdrun</td>
<td>nwchem/6.6</td>
</tr>
<tr>
<td>corset/1.06</td>
<td>gromacs/2016.2_mpi (D)</td>
<td>octave/4.0.1</td>
</tr>
<tr>
<td>eems/8ee979b</td>
<td>hdf5/1.8.16</td>
<td>siesta/4.0mpi</td>
</tr>
<tr>
<td>elpa/2016.05.003</td>
<td>lammps/16feb16</td>
<td>stacks/1.44</td>
</tr>
<tr>
<td>espresso/5.3.0</td>
<td>mvapich2/2.2</td>
<td></td>
</tr>
</tbody>
</table>

### Other Software

<table>
<thead>
<tr>
<th>anaconda2/2.4.0</th>
<th>lmod/6.0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>anaconda2/4.2.0 (D)</td>
<td>lobster/2.1.0</td>
</tr>
<tr>
<td>anaconda3/2.4.0</td>
<td>mercurial/3.7.3-1</td>
</tr>
<tr>
<td>anaconda3/4.2.0 (D)</td>
<td>music/4.0</td>
</tr>
<tr>
<td>bamtools/2.4.1</td>
<td>mzmine/2.23</td>
</tr>
<tr>
<td>binutils/2.25.1</td>
<td>netapp/5.4p1</td>
</tr>
<tr>
<td>blast/2.2.26</td>
<td>netapp/5.5 (D)</td>
</tr>
<tr>
<td>bonnie++/1.03e</td>
<td>octave/4.2.0 (D)</td>
</tr>
<tr>
<td>boost/1.59.0</td>
<td>openblas/0.2.18_barcelona</td>
</tr>
<tr>
<td>bowtie/1.1.2</td>
<td>openblas/0.2.18_haswell</td>
</tr>
<tr>
<td>canu/1.3</td>
<td>openblas/0.2.18 (D)</td>
</tr>
<tr>
<td>cast/dbf2ec2</td>
<td>orangefs/2.9.6</td>
</tr>
<tr>
<td>cp2k/4.1_pre_openmp</td>
<td>parmetis/4.0.3</td>
</tr>
<tr>
<td>Available Software Modules on Kamiak (Cont’d)</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Other Software</td>
<td></td>
</tr>
<tr>
<td>cp2k/4.1_pre_serial</td>
<td></td>
</tr>
<tr>
<td>cp2k/4.1 (D)</td>
<td></td>
</tr>
<tr>
<td>cuda/7.5 (D)</td>
<td></td>
</tr>
<tr>
<td>cuda/7.5.18</td>
<td></td>
</tr>
<tr>
<td>cuda/8.0.44</td>
<td></td>
</tr>
<tr>
<td>cudnn/4_cuda7.0+</td>
<td></td>
</tr>
<tr>
<td>cudnn/5.1_cuda7.5 (D)</td>
<td></td>
</tr>
<tr>
<td>cudnn/5.1_cuda8.0 (D)</td>
<td></td>
</tr>
<tr>
<td>cufflinks/2.2.1</td>
<td></td>
</tr>
<tr>
<td>dislin/11.0</td>
<td></td>
</tr>
<tr>
<td>dropcache/master</td>
<td></td>
</tr>
<tr>
<td>eigan/3.3.2</td>
<td></td>
</tr>
<tr>
<td>freetype/2.7.1</td>
<td></td>
</tr>
<tr>
<td>freexl/1.0.2</td>
<td></td>
</tr>
<tr>
<td>gdal/2.0.0</td>
<td></td>
</tr>
<tr>
<td>gdal/2.1.0 (D)</td>
<td></td>
</tr>
<tr>
<td>gdb/7.10.1</td>
<td></td>
</tr>
<tr>
<td>geos/3.5.0</td>
<td></td>
</tr>
<tr>
<td>git/2.6.3</td>
<td></td>
</tr>
<tr>
<td>globus/6.0</td>
<td></td>
</tr>
<tr>
<td>google_sparsehash/4cb9240</td>
<td></td>
</tr>
<tr>
<td>grass/7.0.5</td>
<td></td>
</tr>
<tr>
<td>gsl/2.1</td>
<td></td>
</tr>
<tr>
<td>idev-ng/master</td>
<td></td>
</tr>
<tr>
<td>java/oracle_1.8.0_92</td>
<td></td>
</tr>
<tr>
<td>jemalloc/3.6.0</td>
<td></td>
</tr>
<tr>
<td>jemalloc/4.4.0 (D)</td>
<td></td>
</tr>
<tr>
<td>perl/5.24.1</td>
<td></td>
</tr>
<tr>
<td>pexsi/0.9.2</td>
<td></td>
</tr>
<tr>
<td>proj/4.9.2</td>
<td></td>
</tr>
<tr>
<td>python/2.7.10</td>
<td></td>
</tr>
<tr>
<td>python3/3.4.3</td>
<td></td>
</tr>
<tr>
<td>qgis/2.14.15</td>
<td></td>
</tr>
<tr>
<td>qscintilla/2.9.4</td>
<td></td>
</tr>
<tr>
<td>qscintilla/2.10 (D)</td>
<td></td>
</tr>
<tr>
<td>r/3.2.2</td>
<td></td>
</tr>
<tr>
<td>r/3.3.0</td>
<td></td>
</tr>
<tr>
<td>r/3.4.0 (D)</td>
<td></td>
</tr>
<tr>
<td>rampart/0.12.2</td>
<td></td>
</tr>
<tr>
<td>samtools/1.3.1</td>
<td></td>
</tr>
<tr>
<td>settarg/6.0.1</td>
<td></td>
</tr>
<tr>
<td>shore/0.9.3</td>
<td></td>
</tr>
<tr>
<td>shoremap/3.4</td>
<td></td>
</tr>
<tr>
<td>singularity/2.3.1</td>
<td></td>
</tr>
<tr>
<td>smbnetfs/0.6.0</td>
<td></td>
</tr>
<tr>
<td>sratoolkit/2.8.0</td>
<td></td>
</tr>
<tr>
<td>superlu_dist/4.3</td>
<td></td>
</tr>
<tr>
<td>svn/2.7.10</td>
<td></td>
</tr>
<tr>
<td>tcl-tk/8.5.19</td>
<td></td>
</tr>
<tr>
<td>tiff/3.9.4</td>
<td></td>
</tr>
<tr>
<td>tophat/2.1.1</td>
<td></td>
</tr>
<tr>
<td>towhee/7.2.0</td>
<td></td>
</tr>
<tr>
<td>trinity/2.2.0</td>
<td></td>
</tr>
</tbody>
</table>
### Available Software Modules on Kamiak (Cont’d)

<table>
<thead>
<tr>
<th>Other Software</th>
<th>Licensed Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>jemalloc/4.4.0</td>
<td>clc_genomics_workbench/6.0.1</td>
</tr>
<tr>
<td>laszip/2.2.0</td>
<td>clc_genomics_workbench/8.5.1</td>
</tr>
<tr>
<td>librgeotiff/1.4.0</td>
<td>dl_polly/4.08</td>
</tr>
<tr>
<td>libint/1.1.4</td>
<td>gaussian/09.d.01</td>
</tr>
<tr>
<td>libkml/1.3.0</td>
<td>vasp/5.4.4</td>
</tr>
<tr>
<td>liblas/1.8.0</td>
<td>green/1.0</td>
</tr>
<tr>
<td>libspatialite/4.3.0a</td>
<td>stata/14</td>
</tr>
<tr>
<td>libxsmm/1.4.4</td>
<td>zlib/1.2.11</td>
</tr>
</tbody>
</table>

Where:

- **D**: Default Module
- **L**: Module is loaded

Use "module spider" to find all possible modules. Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".
Kamiak Storage

- Kamiak has 4 types of storage available to users
  - `/home` 10GB of storage per user
  - `/data` Investor-purchased storage
  - `/scratch` Temporary storage, available to all users
  - `/local` Temporary fast SSD storage on each node, for all users
- Use `mkworkspace` to create a scratch directory that expires in two weeks
  
  ```sh
  export my_scratchdir = "$(mkworkspace -q)"
  ```
  
  - Can be used inside a job script, for larger storage space

- Use `mkworkspace -b /local` to create a scratch directory on a node’s SSD
  
  ```sh
  export my_scratchdir = "$(mkworkspace -q -b /local -t 0-2:00)"
  ```
  
  - The 2 hour time limit is optional, for example for an idev session

- Use `lsworkspace` to list your scratch allocations and creation dates
  
  - Scratch storage by default is on the HDD-disks on the NetApp

- Use `rmworkspace` to delete your scratch directory when done
  
  ```sh
  rmworkspace -a -f --name=$my_scratchdir
  ```
Other Types of Jobs

• **MPI message passing**
  - Runs on multiple nodes
  - Tasks do not share memory

• **OpenMP shared memory**
  - Runs on single node
  - Threads share memory

• **GPU (Graphics Processing Unit) accelerators**
  - Thousands of tiny pixel cores, and matrix processors
  - Offloads kernel function to run over many data points
  - Requires CUDA, OpenCL, OpenACC

See samples in: /opt/apps/samples/training/other
The newly-established Center for Institutional Research Computing (CIRC) is coordinating many of WSU’s research computing efforts. CIRC’s goal is to propel WSU into a position of national leadership in the field of research computing by unifying the research community around shared and strategic institutional goals. By leveraging cyber-infrastructure resources across the WSU system, CIRC is creating a regional computing hub that enhances research across all colleges and campuses.

Explore this site to learn more about Kamiak, WSU’s high performance computing cluster, and other available computing resources.
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 --mem or --mem-per-cpu options of sbatch to request more memory.
   - sacct -u your.name -o jobid,reqmem,maxrss,state,nodelist to see how much memory you used (maxRSS is per task)
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**

- Cite Kamiak in your work
- Report issues via Kamiak’s Service Desk
- Abide by Kamiak’s End User License Agreement and WSU policies
- Use accurate resource requirements (CPU, time, memory)
- Use /scratch and /local for your computational storage when possible

**Don’t**

- Do not run intensive workloads (computing or compiling) on a login node, use `sbatch` or `idev` to run on a compute node
- Do not submit thousands of jobs – use job arrays
- Do not give your password to anyone, ever
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
- Put in a service request asking for a quote (Tim N.)

Standard Compute Nodes

Please note that these prices are estimates and are subject to change.

If you are planning on purchasing a node or applying for grant funding, please contact our Service Desk for up to date pricing and availability, or alternate configurations.

<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 Xeon Silver 4116</td>
<td>24</td>
<td>192 GB 2666 MT/s DDR4</td>
<td>1 x 400GB SSD</td>
<td>2 x 10GB SFP+ single-port FDR InfiniBand</td>
<td>$9,800 / 5yrs</td>
</tr>
<tr>
<td>Dual Intel Xeon Gold 6138</td>
<td>40</td>
<td>320 GB 2666 MT/s DDR4</td>
<td>1 x 400GB SSD</td>
<td>2 x 10GB SFP+ single-port FDR InfiniBand</td>
<td>$14,900 / 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 E5-2660v4</td>
<td>28</td>
<td>256GB 2133MT/s DDR4 ECC</td>
<td>Dual Tesla K80 GPU (9984 CUDA cores)</td>
<td>400 GB Solid State Drive @ 6 Gbps</td>
<td>2 x 10GB SFP+ dual-port FDR infiniband</td>
<td>$17,500 / 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.

Other training sessions are planned throughout the year – 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))
- WSU HPC club → 4 nodes purchased through Tech Fee grant
- WSU HPC club officers Thomas Badman, Connor Wytko