Introduction to Kamiak
Training Workshop

Eric Roalson, CIRC Director
Peter Mills, Assistant Director
Rohit Dhariwal, Computational Scientist
Roy Obenchain, HPC Systems Administrator
Will Aoki, HPC Systems Administrator
Tim Neumann, Program Coordinator

hpc.wsu.edu/training/slides
hpc.wsu.edu/training/follow-along
hpc.wsu.edu/cheat-sheet

These slides
Handout
Cheat Sheet

March 10, 2021
What is Kamiak

- Kamiak is a **cluster** of many individual computers (called **nodes**) connected by a high-speed network
- Each computer is like your laptop, but with more cores (40) and memory
- Applications can run in **parallel** over many cores and across multiple nodes
- **Speeds up** solving large problems

Nodes: 137
Cores: 3,372
Memory: 33 TB
Storage: 1.2 PB
GPU cores: 60,032
What you will learn today

- Transfer files to and from Kamiak
- Logging into Kamiak
- Basic Linux commands to navigate folders, move files
- How to run jobs on Kamiak
  - Submit batch jobs
  - 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 for nodes or storage
Transferring Files to and from Kamiak

- For Windows, first make sure you have Ubuntu installed. To install Ubuntu, see: hpc.wsu.edu/cheat-sheet

- There are several ways to transfer and synchronize files between your laptop and Kamiak.

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

  Windows alternative: **Winscp**

- **Synchronize**

  `rsync`
Transferring Files - `scp`

- Let’s transfer some files from Kamiak to our machine
  
  *Make sure you are on your laptop, not logged into Kamiak*

**Open a terminal window**

- Terminal >> New Window *(Mac)*
- Start >> Ubuntu *(Windows)*

- `cd`

**Transfer a folder from Kamiak to my laptop**

- `scp -r your.name@kamiak.wsu.edu:/opt/apps/samples/training .`
  
  Recursive, copies all files in folder  
  From Kamiak  
  To current folder on laptop

**Now from my laptop back to Kamiak**

- `scp -r training your.name@kamiak.wsu.edu:~/`
  
  From my machine  
  To my home directory on Kamiak
We can also synchronize folders between Kamiak and your laptop so that they have the same contents.

```
rsync -ravx source destination
```

- Copies any changed files from the source to the destination
- Only adds or replaces files, doesn’t delete them from the destination
- Useful for data backups and mirroring

Common options used with rsync:

- `-r` recursive copy, all subfolders and files
- `-a` archive mode, preserves permissions, ownership
- `-v` verbose
- `-x` don't cross filesystem boundaries
- `-z` compress file data
- `-h` print output in human-readable format
Transferring Files - `rsync`

- Let’s synchronize a folder on Kamik to our laptop
  
  Make sure you are on your laptop, not logged into Kamiak

  - `cd`
  - `rsync -rvx training/ your.name@kamiak.wsu.edu:~/training`

  Recursive, archive, verbose, compress
  From folder on my laptop
  To folder on Kamiak

```
sumo(10011) % rsync -rvx training/ peter.mills@kamiak.wsu.edu:~/training
peter.mills@kamiak.wsu.edu's password:
sending incremental file list

sent 219 bytes received 21 bytes 43.64 bytes/sec
total size is 3,105 speedup is 12.94
sumo(10012) %
```
Logging into Kamiak - ssh

Open a terminal window
• Terminal >> New Window (Mac)
• Start >> Ubuntu (Windows)

Log into Kamiak
• ssh your.name@kamiak.wsu.edu
• exit

Welcome to Kamiak!
Documentation on this system can be found at:
https://hpc.wsu.edu

Use of Kamiak implies your acceptance of Kamiak's End User License Agreement,
which is available at: https://hpc.wsu.edu/eula

To see all the software that is available across all compilers and
MPI stacks, issue: "module spider"

You have access to the following compute partitions:
  adam, beckman, bio534, cahnrs, cahnrs_bigmem, cahnrs_gpu, cas, catalysis_gpu,
  catalysis_long, clark, flicklin, free_gpu, hort503-01-s18, hpc_club, kamiak, katz,
  lee, lofgren, mainlab, pddms, peters, popgenom, ssl, stockle, storfer, tanner,
  taylor, test, vcea

[peter.mills@login-p1n01 ~]$ _
Basic Linux Commands

• There are many basic Linux commands you must master

  cd  *path*  move your current position in the folder tree
  ls  *path*  list contents of a folder (“ls” for current folder)
  cp  *from to*  copy files, for folders use “cp –r from to”
  mv  *from to*  move files and folders
  rm  *todelete*  delete files, for folders use “rm –r –l path”

• Special pathnames  
  Example: cd ~/..
  .. means one level up, the parent folder
  .  means the current folder
  ~  means your home directory

• For other commands see hpc.wsu.edu/cheat-sheet or Ch2 of the tutorial sent in email
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
Running Jobs on Kamiak – Key Concepts

• Kamiak is a **cluster** of computers called **nodes**, each 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
  • Also overarching shared backfill partition (“kamiak”) for all users

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

• Job gets added to a **queue** to wait in line
  • One queue for each partition (i.e., group of nodes)
  • Job waits to run until resources are available

• **Slurm job scheduler** decides:
  • Who goes first, who gets what
  • Who gets bumped (preempted) from backfill if an investor needs node back

• **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 to a job from parts of several nodes

• **Applications will only run in parallel if they are built to do so**
  • Kamiak does not parallelize your program
  • It just runs your program on the node it assigns it to
Shared Backfill Partition ("kamiak")

- Kamiak's nodes are grouped into partitions, each owned by investor
- All nodes also belong to a shared partition (called "kamiak")
  - 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: 180 CPU cores (~ 9 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’s/cores, memory, GPU’s)
  - 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
Submitting Batch Jobs to Kamiak

• Log into Kamiak
  
  `ssh your.name@kamiak.wsu.edu`
  
  `cd training`
  
  `source training_only_setup.sh`  # One-time setup 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 3/10/21 -u your.name`  # Past job history
  
  `scontrol show job jobNumber`  # Job details
Viewing Information about the Cluster

Follow along

- 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
**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

```bash
kamiak$ idev -N 1 --ntasks=1 -cpus-per-task=2 -t 360
Iddev interactively runs commands on a compute node.
See 'man sallocate' for idev options to reserve a job allocation.
To use a GPU within idev: use 'srun yourCommand', e.g. 'srun python -i'.
To use X11 forwarding from a compute node:
  Use 'ssh -Y' or more secure 'ssh -X' to log into Kamiak.
  Within idev, use 'srun --x11' to launch a task with a user interface.
Recommend using 'srun -I' to launch a task without hanging.
Default time is 60 minutes. Use '-t yourMinutes' to override.
sallocate: Granted job allocation 1160832
Allocated nodes: cn32

cn32$ module avail

cn32$ module load python/2.7.15

cn32$ module list
Currently Loaded Modules:
  1) StdEnv
  2) intel/xe_2016_update3
  3) workspace_maker/master
  4) python/2.7.15
```
Interactive Jobs

Use `srun -l` to avoid hanging if resources are not available.

cn32$ python -i
Python 2.7.15 (default, Jun 10 2018, 14:31:07)
[GCC 7.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hello World!"
Hello World!
>>> exit()

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

cn32$ exit
exit
salloc: Relinquishing job allocation 1160832

kamiak$
Job Arrays

- **Template** to create multiple instances of a job
  - `#SBATCH --array=1-5`  
    *Creates 5 instances, one for each index 1,2,3,4,5*

- Each instance is an individual job with the same resources
  - Template holds place in queue, spawns instances as resources become available

- Each instance has a unique index `$SLURM_ARRAY_TASK_ID`
  - Can use index however you want, in different ways

- Example: run the same program over different files *(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
```
Running Job Arrays

kamiak$ cat jobArray.sh

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</td>
<td>_0</td>
<td>kamiak</td>
<td>myJobArr</td>
<td></td>
<td>0:01</td>
<td>1</td>
<td>cn32</td>
</tr>
<tr>
<td>1161089</td>
<td>_1</td>
<td>kamiak</td>
<td>myJobArr</td>
<td></td>
<td>0:01</td>
<td>1</td>
<td>cn32</td>
</tr>
<tr>
<td>1161089</td>
<td>_2</td>
<td>kamiak</td>
<td>myJobArr</td>
<td></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
Other Types of Jobs

• **Single node shared memory**
  - Multithreaded or OpenMP
  - Threads share memory

• **Multiple nodes**
  - MPI for message passing
  - Tasks do not 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/advanced
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

```bash
module avail
# See available modules for compiler
module load python3/3.4.3
# Load specific version (Recommended)
module list
# See loaded modules
module avail python3
# See available python3 modules
module load python3
# Load latest version
module unload python3
# Unload a module
module spider
# See all modules
module whatis anaconda3
# See what a module does
module help anaconda3
# See help for a module
which python3
# See that python is in your path
printenv PATH
# See effects of loading modules
printenv LD_LIBRARY_PATH
```
Kamiak Storage

- Kamiak has 4 types of storage available to users
  - `/home/your.netid` 100GB per user
  - `/data/pi.name` 500GB per PI lab (e.g., `/data/clark`)
    
    *Extra storage is available for rent from CIRC service center*
  - `/scratch` Temporary storage, available to all users
  - `/local` Temporary storage on node, for all users
• Use `mkworkspace` to create a scratch directory that expires in two weeks
  
  ```bash
  export myscratch="$(mkworkspace)"
  ```

  - Can be used inside a job script, for faster staging or a larger storage space

  - Use `mkworkspace -l` to create a scratch directory on a node’s SSD
  ```bash
  export myscratch="$(mkworkspace -l)"
  ```

  • Use `lsworkspace` to list your scratch allocations (can also just use `ls`)
  ```bash
  lsworkspace -v
  ls -ld /scratch/${USER}*
  ```

  • Just use `rm` to delete the contents of your scratch directory when done
  ```bash
  rm -r -I $myscratch/*
  ```
The Center for Institutional Research Computing (CIRC) provides cutting-edge high-performance computing (HPC) resources and expertise to advance computational and data-intensive research across WSU. CIRC’s goal is to propel WSU into a position of national leadership in research computing by providing state-of-the-art HPC hardware and software infrastructure and a world-class team that supports its effective utilization and fosters computational literacy. Our services include support, guidance, and training in the use of the HPC cluster, application development and deployment, and investing in nodes and storage.
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.

2. **“Requested node configuration is not available”**
   - 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.**
   - 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).
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

• 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)

Don’t

• Do not run intensive workloads (computing, installs, or long-running file transfers) 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
Purchasing Nodes and Renting Extra Storage

- All users have access to the backfill queue, /home and /scratch storage, and any /data storage made available by their PI.
- If you need more → **have your PI become an investor**
- Submit a service request to purchase nodes or rent extra storage
  - *Nodes are permanently owned by the investor with a 5-year warranty*
  - *Storage can be rented annually in units of 512GB per year*

- **Standard compute nodes**
  - 40-cores Intel Xeon Gold, 384GB memory
  - 40-cores Intel Xeon Gold, 192GB memory
  - Optional Nvidia A100 GPU’s
  - Optional large-memory, 1500GB

- For price quotes, please submit a service request

For detailed node descriptions, please see [hpc.wsu.edu/kamiak-hpc/becoming-an-investor/](http://hpc.wsu.edu/kamiak-hpc/becoming-an-investor/)
• 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:
  - CIRC Advisory Committee - share your ideas with its members
  - WSU HPC club - 4 nodes purchased through Tech Fee grant
Different ways to use rsync

- `rsync -ravx training your.name@kamiak.wsu.edu:~/`

  Without slash, aligns training to ~/training (copy the folder by name)
  With a slash, aligns training/* to ~/ (copy the contents of the folder)

- `rsync -ravx --delete training/ your.name@kamiak.wsu.edu:~/training`

  Deletes non-matching items in the destination
How Job Submission Works

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

**What**
```
#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)

**Where**
- Kamiak Cluster
  - Node
  - Memory
  - Cores
  - Network
Types of Applications

**Multithreaded** programs take *cpus-per-task*, and run on 1 node

```
#SBATCH --nodes=1 --ntasks-per-node=1 --cpus-per-task=10
myapp
```

Number of cores per running program

Runs the program once

**MPI** programs take *tasks*, and run on multiple nodes

```
#SBATCH --nodes=2 --ntasks-per-node=4 --cpus-per-task=1
srun myapp
```

Runs the program once for each task

```
#SBATCH --nodes=1-2 --ntasks=8 --cpus-per-task=1
srun myapp
```

*Dynamic layout*

Let it compact the task layout over the min-max nodes

1. Total allocated cores = ntasks * cpus-per-task
2. --ntasks-per-node is treated as max tasks per node, if --ntasks is also specified
Different Ways to Use Job Arrays

• You can use a job array index in different ways

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

• 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 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
Transferring Files - `scp`

```
sumo(10110) %scp -r peter.mills@kamiak.wsu.edu:/opt/apps/samples/training .
peter.mills@kamiak.wsu.edu's password:
data_2.txt     100% 29 10.2KB/s 00:00
data_1.txt     100% 29  8.5KB/s 00:00
data_0.txt     100% 29  8.2KB/s 00:00
setup.sh        100% 156 83.6KB/s 00:00
jobArray.sh    100% 1252  590.4KB/s 00:00
myJob.sh        100%  999  504.2KB/s 00:00
helloWorld.py  100%  611  302.3KB/s 00:00
sumo(10111) %
```

```
sumo(10116) %scp -r training peter.mills@kamiak.wsu.edu:~/
peter.mills@kamiak.wsu.edu's password:
helloWorld.py  100%  611  68.0KB/s 00:00
data_0.txt    100%  29  3.9KB/s 00:00
data_1.txt    100%  29  4.6KB/s 00:00
data_2.txt    100%  29  4.9KB/s 00:00
jobArray.sh 100% 1252 116.8KB/s 00:00
myJob.sh     100%  999 110.7KB/s 00:00
setup.sh     100%  156 18.5KB/s 00:00
sumo(10117) %
```
#!/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=1 # Number of tasks per node (max)
#SBATCH --ntasks=1          # Number of tasks (processes) What (cpus)
#SBATCH --cpus-per-task=2   # Number of cores per task (threads)

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

module load python/2.7.15 # Load software module from Kamiak repository
srun python helloWorld.py -w   # Each task runs this program (total 1 times)
                            # Each srun is a job step, and spawns -ntasks

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`

<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>1160269</td>
<td>kamiak</td>
<td>myJob</td>
<td>peter.mi</td>
<td>PD</td>
<td>0:00</td>
<td>1</td>
<td>(Priority)</td>
</tr>
</tbody>
</table>

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>1160269</td>
<td>kamiak</td>
<td>myJob</td>
<td>peter.mi</td>
<td>R</td>
<td>0:02</td>
<td>1</td>
<td>cn93</td>
</tr>
</tbody>
</table>

kamiak$ `squeue -j 1160269`

<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>1160269</td>
<td>kamiak</td>
<td>myJob</td>
<td>peter.mi</td>
<td>R</td>
<td>0:17</td>
<td>1</td>
<td>cn93</td>
</tr>
</tbody>
</table>

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

kamiak$ `scancel 1160269`

See also: man sbatch
Viewing Job History and Details

kamiak$ `sacct -S 10/26/20 -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=4294680391  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
StartTime=2020-02-26T13:57:03  EndTime=2020-02-26T13:58:23  Deadline=N/A
Partition=kamiak  AllocNode:Sid=login-p1n02:33414
NodeList=cn93  BatchHost=cn93
NumNodes=1  NumCPUs=2  NumTasks=1  CPUs/Task=2  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 Available Partitions

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

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

```
kamiak$ squeue -a | more
JOBID PARTITION  NAME   USER ST   TIME NODES NODELIST(REASON)
474239       cas      tp4 craig.te PD       0:00 2 (Resources)
474240       cas      tp5 craig.te PD       0:00 2 (Priority)
474241       cas      tp6 craig.te PD       0:00 2 (Priority)
471077  popgenom BFS.3L.i joel.t.n  R 9-03:34:38 1 cn77
471078  popgenom BFS.2R.i joel.t.n  R 9-03:34:03 1 cn29
471079  popgenom BFS.2L.i joel.t.n  R 9-03:33:14 1 cn29
473678    kamiak cle_6cma tung.ngu  R 6-05:07:11 1 cn7
473722   beckman hydr-vra hong.zho  R 6-04:28:26 2 cn[43-44]
473726   beckman occ-1na3 hong.zho  R 5-23:37:54 2 cn[52-53]
473727    kamiak dbl_Pt21 mareike.  R 2-23:09:49 2 cn[3-4]
```
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=2019-11-28T09:00:39  SlurmdStartTime=2019-11-28T09:01:34
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
#!/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 $HOSTNAME"
module load python
srun python helloWorld.py -w "inputs/data_${SLURM_ARRAY_TASK_ID}.txt"

echo "Completed job array $SLURM_ARRAY_TASK_ID on host $HOSTNAME"
Available Software Modules on Kamiak

---------------------------------- Compilers ----------------------------------

StdEnv  (L)  gcc/5.2.0  intel/xe_2016_update2
gcc/4.9.3  gcc/6.1.0 (D)  intel/xe_2016_update3 (L,D)

------------------------

intel/xe_2016_update3 Software

bazel/0.4.2  fftw/3.3.4  netcdf/4
cmake/3.7.2  gromacs/2016.2_mdrun  nwchem/6.6
corset/1.06  gromacs/2016.2_mpi  (D)  octave/4.0.1
eems/8ee979b  hdf5/1.8.16  siesta/4.0_mpi
eELPA/2016.05.003  lammps/16feb16  stacks/1.44
espresso/5.3.0  mvapich2/2.2

--------------------------------

Other Software

anaconda2/2.4.0  lmod/6.0.1
anaconda2/4.2.0  (D)  lobster/2.1.0
anaconda3/2.4.0  mercurial/3.7.3-1
anaconda3/4.2.0  (D)  music/4.0
bamtools/2.4.1  mzmine/2.23
binutils/2.25.1  netapp/5.4p1
blast/2.2.26  netapp/5.5  (D)
bonnie++/1.03e  octave/4.2.0  (D)
boost/1.59.0  openblas/0.2.18_barcelona
bowtie/1.1.2  openblas/0.2.18_haswell
canu/1.3  openblas/0.2.18  (D)
cast/dbf2ec2  orangefs/2.9.6
cp2k/4.1_pre_openmp  parmetis/4.0.3
Available Software Modules on Kamiak (Cont’d)

---------------------------------- Other Software ----------------------------------
jemalloc/4.4.0 (D) trinity/2.2.0
laszip/2.2.0 valgrind/3.11.0
libgeotiff/1.4.0 workspace_make/master (L,D)
libint/1.1.4 workspace_make/1.1b
libkml/1.3.0 workspace_make/1.2
liblas/1.8.0
libspatialite/4.3.0a zlib/1.2.11
libxsmm/1.4.4

---------------------------------- Licensed Software ----------------------------------
clc_genomics_workbench/6.0.1 gaussian/09.d.01 vasp/5.4.4
clc_genomics_workbench/8.5.1 (D) green/1.0
dl_polly/4.08 stata/14

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".
The End