Introduction to Kamiak
Training Workshop

Aurora Clark, 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 22, 2022
What you will learn today

- What is Kamiak
- How to run jobs on Kamiak
  - Submit batch jobs
  - Interactive compute session
  - Types of jobs
- Exercises
  - Logging into Kamiak
  - Transferring files to and from Kamiak
  - Running batch jobs
  - Running an interactive compute session
  - Running job arrays
  - Using scratch storage
  - Using snapshots
What is Kamiak

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

Nodes: 151
Cores: 4,196
Memory: 42 TB
Storage: 1.2 PB
GPU cores: 93,056
Kamiak Storage

- Kamiak has 4 types of storage available to users

  /home/your.name          100GB per user
  /data/lab/pi.labname     500GB per PI lab (e.g., /data/clark)

  Extra storage is available for rent from CIRC service center

  /scratch                  Temporary storage, 2-week lifetime, 10TB limit per user
  /local                    Temporary storage on node, 2-week lifetime, ~400GB
Running Jobs on Kamiak

- **Nodes** are grouped into **partitions**, each owned by a PI or college.
- All nodes also belong to shared **kamiak** partition, available to all users.
- You submit a **job** to a partition asking for **nodes**, **tasks**, and **cores**.
- Job gets added to a partition’s **queue** to wait until resources are available.
- **Slurm** job scheduler decides **who goes first, who gets what, who gets bumped**.
- Investors have priority access to the nodes they own.
- Will **preempt** job in backfill if investor’s job needs its cores.
- Applications only run in parallel if built to do so.
- Resource requirements differ for each app.
There are two ways to run jobs on Kamiak

• **sbatch** `myJob.sh** Batch job submission**
  - Says which partition to submit to (default is kamiak)
  - 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

---

Do not run compute or compile jobs on login nodes, use **sbatch** or **idev** instead
Types of Jobs

• **Single node**
  - Single program instance
  - Multithreading over multiple cores
  - Threads share memory

```
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=10
export OMP_NUM_THREADS=
$SLURM_CPUS_PER_TASK
```

• **Multiple nodes**
  - Each task is a program instance
  - Tasks do not share memory
  - Communicate by message-passing

```
#SBATCH --nodes=2
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=10
```

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

```
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --gres=gpu:tesla:1
```

See samples in: `/opt/apps/samples/advanced`
Introduction to Kamiak  
Follow Along

Logging into Kamiak 

Open a terminal window  
Terminal >> New Window  (for Windows, Start >> Ubuntu)  

Log into Kamiak  
ssh your.name@kamiak.wsu.edu  
extit

Transferring Files to and from Kamiak  

Make sure you are on your laptop, not logged into Kamiak  

Copy from Kamiak to your laptop  
scp -r your.name@kamiak.wsu.edu:/opt/apps/samples/training .  
↑ Recursive, copies all files  
Is -l training  

Copy from your laptop to Kamiak  
scp -r training your.name@kamiak.wsu.edu:~/  
↑ From my laptop  
Synchronize folder contents (copies changed or added files, does not delete)  
rsync -ravx training/ your.name@kamiak.wsu.edu:~/training  
↑ All files  
↑ From laptop  
↑ To Kamiak

Submitting Batch Jobs to Kamiak  

Log back into Kamiak  
ssh your.name@kamiak.wsu.edu  

One-time setup only for this training  
cd training  
source training_only_setup.sh  

Create/edit a job script  
cat myJob.sh
Submit the job script to the job queue
   sbatch  myJob.sh       # To test: sbatch --test-only myJob.sh

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

See output
   cat myJob*.out

Cancel the job
   scancel jobNumber

View past and active jobs
   sacct -u your.name     # Past job history
   scontrol show job jobNumber   # Job details

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, GPUs

Interactive Jobs

Create interactive session on a compute node
   idev -N 1 --ntasks=1 --cpus-per-task=2 -t 360

Module commands set up app environment
   module avail       # Shows available apps for loaded compiler
   module help python3/3.9.5 # See app-specific instructions, resources differ for each app
   module load python3/3.9.5 # Loads specific version (recommended)
   module list        # See loaded modules

Do not run compute jobs on the login nodes
Run the app (use srun only for multiple nodes, runs program once for each task)

```python
python3 -i
    print("Hello World!")
    exit()
```
srun -I python3 helloWorld.py  # Use srun -I to avoid hanging if resources not available
exit

Job Arrays

*Placeholder to create instances of a job as resources become available*

```bash
#SBATCH --array=1-5  # Creates 5 job instances, one for each index 1,2,3,4,5
```

*Each instance is an individual job with the same resources* (index is $SLURM_ARRAY_TASK_ID)

```bash
cat jobArray.sh
sbatch jobArray.sh
squeue -u your.name
cat output/myJobArray*.out
scancel jobNumber
```

Using Scratch Storage

*Create a scratch directory that expires in two weeks*

```bash
export myscratch="$(mkworkspace)"  # Can use inside or outside a job script
echo $myscratch
```

*List your scratch allocations*

```bash
isworkspace
```

*Can optionally delete contents when done*

```bash
rm -r -I $myscratch/*
```

Snapshots

*Three days of read-only backups of home and data folders*

```bash
ls /home/.snapshots
ls /home/.snapshots/daily.2022-03-21_0000/your.name
ls /data/.snapshots/daily.2022-03-21_0000
```
Using Available Software on Kamiak

module avail  # Available modules compatible with compiler
module load python3/3.9.5  # 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

Getting Help

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

Support & Zoom Help Desk Hours

User’s Guide / Kamiak Cheat Sheet
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 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 compute jobs or installs 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/lab 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
  • 64-cores Intel Xeon Gold, 512GB memory
  • Optional Nvidia A100 GPU’s
  • Optional large-memory, 1-2TB

• For price quotes, please submit a service request
  For detailed node descriptions, please see *hpc.wsu.edu/kamiak-hpc/becoming-an-investor/*
• We will be sending out a 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
#!/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 python3           # Load software module from Kamiak repository
srun python3 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"
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

```
kamiak$ idev -N 1 --ntasks=1 -cpus-per-task=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: 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.
salloc: Granted job allocation 1160832
Allocated nodes: cn32
```

```
cn32$ module avail          # Shows available apps for loaded compiler
# Module commands set up app environment
```

```
cn32$ module help python3/3.9.5 # See any app-specific instructions
# (Resources differ for each app)
```

```
cn32$ module load python3/3.9.5 # Loads specific version (recommended)
```

```
cn32$ module list            # See loaded modules
Currently Loaded Modules:
  1) intel/20.2  2) StdEnv    3) python3/3.9.5
```
Interactive Jobs

```bash
cn32$ python3 -i
Python 3.9.5 (default, Jun 2 2021, 10:10:20)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>> exit()

cn32$ srun -I python3 helloWorld.py
Hello World! from cn32

cn32$ exit
exit
exit
salloc: Relinquishing job allocation 1160832

kamiak$
```

Use `srun -I` to avoid hanging if resources are not available
#!/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 python3

srun python3 helloWorld.py -w "inputs/data_${SLURM_ARRAY_TASK_ID}.txt"

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