# Advanced Research Computing (ARC) Training

**Matthew Brown/Ayat Mohammed/Chris Kuhlman/Sarah Ghazanfari/Sophia Lima**

*Computational Scientists*

*Advanced Research Computing (ARC), Division of Information Technology*

*Virginia Tech*

in collaboration with **other ARC staff members** and **GRAs**

Wednesday, 05 Jun 2024
Summer 2024

# Monitoring Resource Utilization and Job Efficiency

**Chris Kuhlman**

ckuhlman@vt.edu

*Computational Scientist*

*Advanced Research Computing (ARC), Division of Information Technology*

*Virginia Tech*

in collaboration with **other ARC staff members** and **GRAs**

05 Jun 2024
Summer 2024 Series

# Advanced Research Computing (ARC) Trainings, Summer 2024

**via Zoom video conferencing**

**Monday**

- 06/03*: Introduction to Advanced Research Computing (SG)

  Basics of HPC, computer clusters, HPC resources, access to ARC systems

- 06/03*: Connect to ARC Systems and Run your first jobs (AM)

  Connect via Open OnDemand, connect via SSH, cluster and scheduler orientation, run demo jobs

**Tuesday**

- 06/04*: Running code/software on ARC systems in different ways (CK)

  Job environments (modules and Conda), running interactive and batch jobs

- 06/04*: Launching Jobs in Parallel on ARC Clusters (AM)

  MPIRUN vs. SRUN, GNU parallel for load balancing, SRUN for resource detection and binding, "Built-in" or library-based parallelism

**Wednesday**

- 06/05*: Monitoring Resource Utilization and Job Efficiency (CK)

  Acquiring resources, characteristics of compute nodes, overall activity, current loads, job status

# Sign Up Sheet:  Please Sign Up

1. Google sign up sheet is here:
    A. https://docs.google.com/document/d/1VPBIIIupSK4gpSm2DW3w4x4yzDB3QjRQ/edit
2. Please sign in to ensure:
    A. You get credit for the course
    B. Our roster is complete
3. Also, this google sheet has
    A. Commands that we are going to execute together.
    B. Link for these slides
    C. Space for feedback

# Get These Slides

- Slides are available at a link in this file:
    A. https://docs.google.com/document/d/1VPBIIIupSK4gpSm2DW3w4x4yzDB3QjRQ/edit

# Resources

- ARC documentation
  - https://www.docs.arc.vt.edu/
  - READ THIS (No joke; there is vocabulary, computing resources, etc.  Can save a lot of time.)
- Get an account on ARC
  - https://arc.vt.edu/account
- Get a project on ARC (lot more storage)
  - https://coldfront.arc.vt.edu
- Help Desk
  - https://arc.vt.edu/help
- Office hours for with GRAs
  - https://arc.vt.edu/office-hours

# Context, Goals, Feedback

- Context
  - This is an informal workshop
  - Mostly informational about ARC and research computing at VT
  - For new students, faculty, staff, researchers.  And anyone else.
- Goals
  - Create awareness of what types of commands are available to interrogate hardware/software.
  - Learn how to understand your job and all jobs running on a compute node.
- We want to hear your questions
  - Just interrupt the talk
  - Welcome to use chat to ask questions + some time at the end
- Feedback needed to help improve future workshops.  ***PLEASE***
  - One up / one down at the end
  - More detailed feedback

# First Thing's First

VPN needed for connections from off-campus

- https://www.nis.vt.edu/ServicePortfolio/Network/RemoteAccess-VPN.html
- Nearly all ARC services require being on the campus network or VPN
- Use "VT Traffic over SSL VPN" connection
- ColdFront (accounting system) available with or without VPN

Get an ARC account:

- https://coldfront.arc.vt.edu/account/create
- Acceptable Use Policy

# Outline

- Operational models.
- Use of head nodes (or login nodes).
- Seven exercises:
  - Four show commands to use for looking at things like cpu (core) and memory usage.
  - One shows how to inspect your usage and limits on ARC machines.
  - One uses some of these commands while running a simple stress test.
  - One uses a real scenario
- Some of this presentation uses exercises from Workshop S3 and other workshops in the series to request resources.

# HPC Resources at ARC/VT

We use TC today

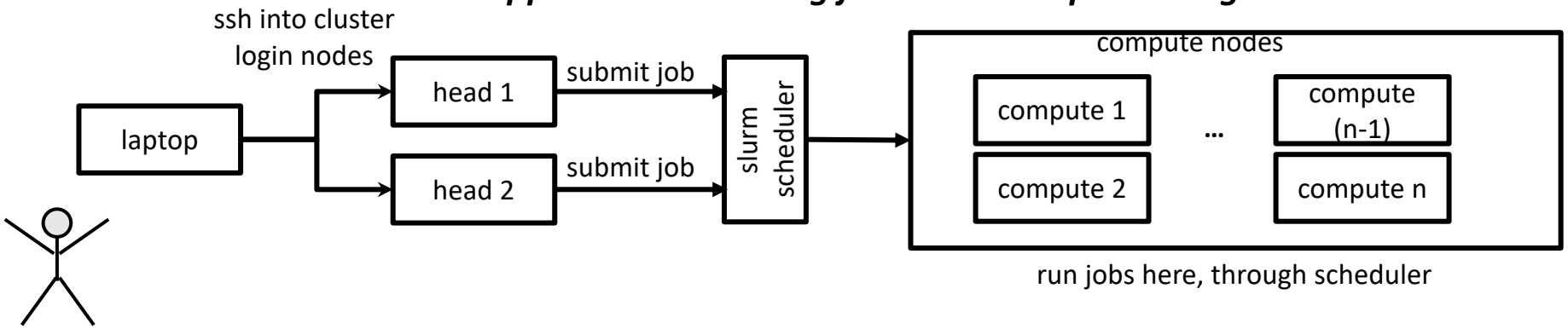
| Cluster | Description | Since |
|---|---|---|
| CUI | Dense GPU + some CPU<br>for projects with controlled data/software | c. 2021 |
| Tinkercliffs | ~~HPC/HTC~~<br>Flagship CPU<br>~~HPE Dense GPU nodes (A100)~~<br>DGX Dense GPU nodes (A100) | c. 2020<br>c. 2021<br>c. 2022 |
| Infer (nearing end of life) | Accelerating inference and ML workloads (T4 GPU)<br>Added P100 GPUs from Newriver<br>Added V100 GPUs from Cascades | c. 2021<br>c. 2016 (EOL)<br>c. 2018 (EOL) |
| OWL (coming soon) | Water-cooled<br>latest generation AMD CPU<br>high mem-per-core<br>DDR5 | c. 2024 |
| Falcon (later in 2024) | GPU node expansion<br>L40S GPUs (20 nodes x4 GPUs)<br>A30 GPUs (32 nodes x4 GPUs) | c. 2024 |

# Operational Models

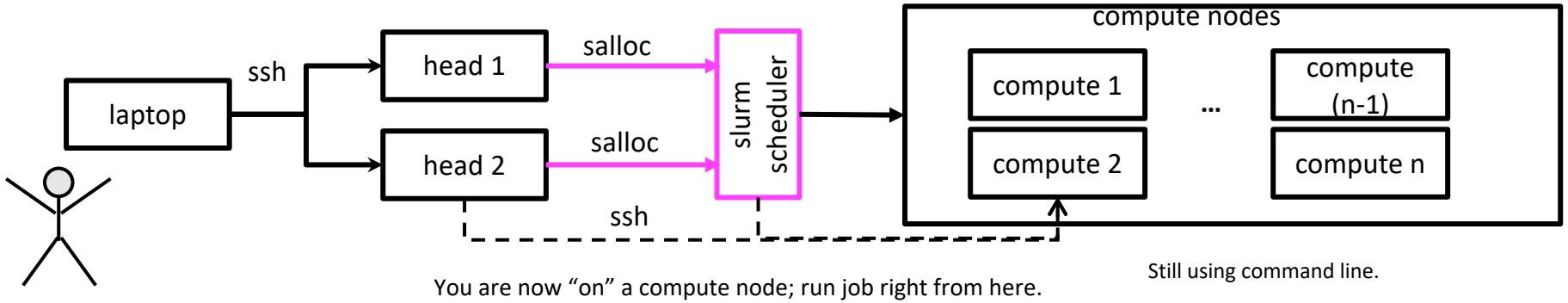
- We are going to review briefly the overall system hardware and your laptop or tower.
    - This will help us to use codes and system (shell, directives) software to set up jobs and run jobs.
    - Different pieces of the software run on different pieces of the hardware.
- If you get these concepts down, your life will be much easier, going forward, in all sorts of ways.
- Real examples and real code require these ideas.

# High Level Operating Environment

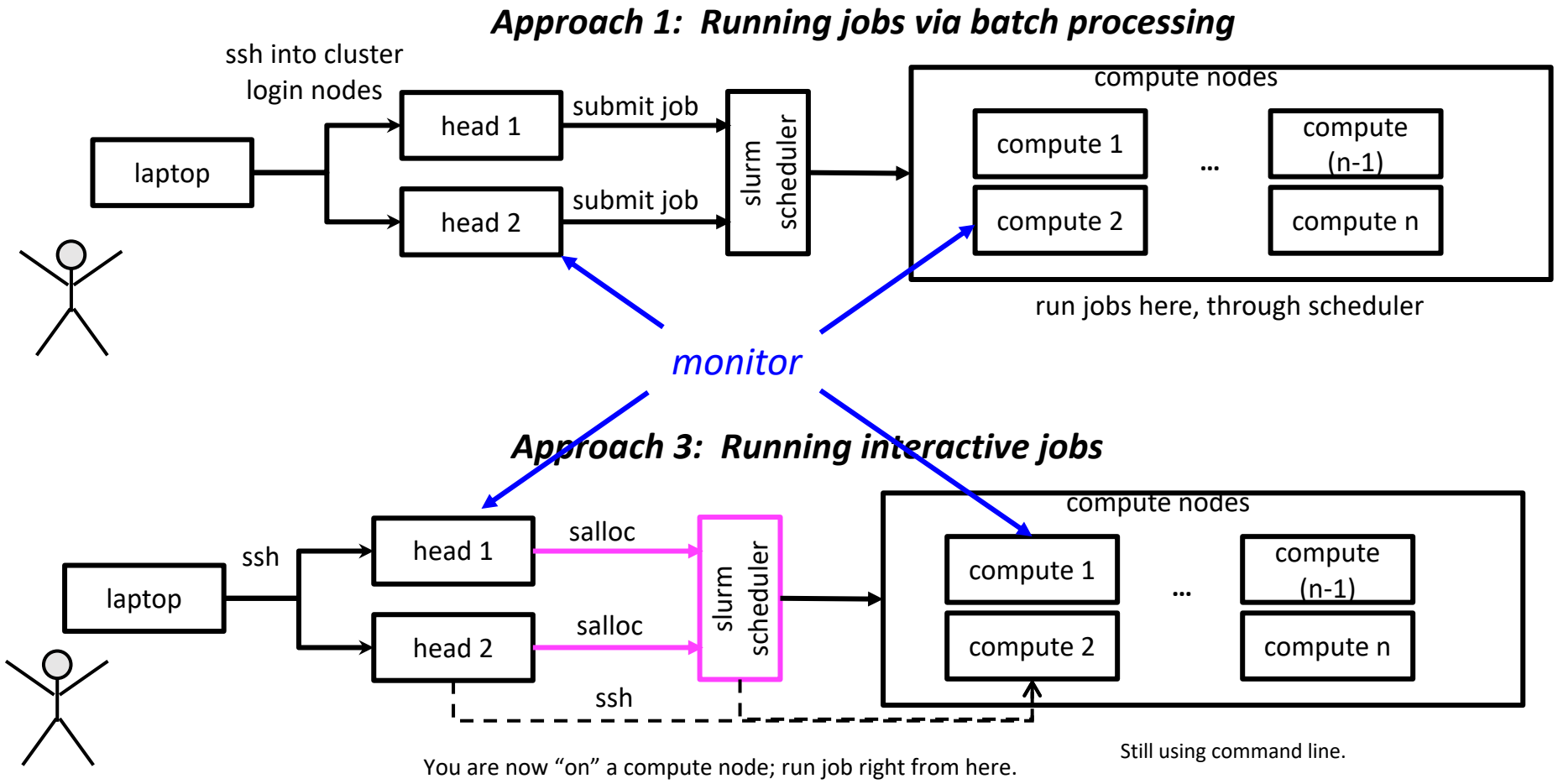
***Approach 1: Running jobs via batch processing***




***Approach 3: Running interactive jobs***



# High Level Operating Environment



# Log In To Tinkercliffs

- Start vpn as you did yesterday, or have done.
- Open a terminal window (preferably 2 or 3) on your tower or laptop.
- Use ssh, in that terminal window, to connect to tinkercliffs, typing:
- ssh <user-name>@tinkercliffs1.arc.vt.edu
- Enter password
- You are now on the tinkercliffs head node 1.

- Repeat this in another (different) terminal window.
- You will have two terminal connections to tinkercliffs.

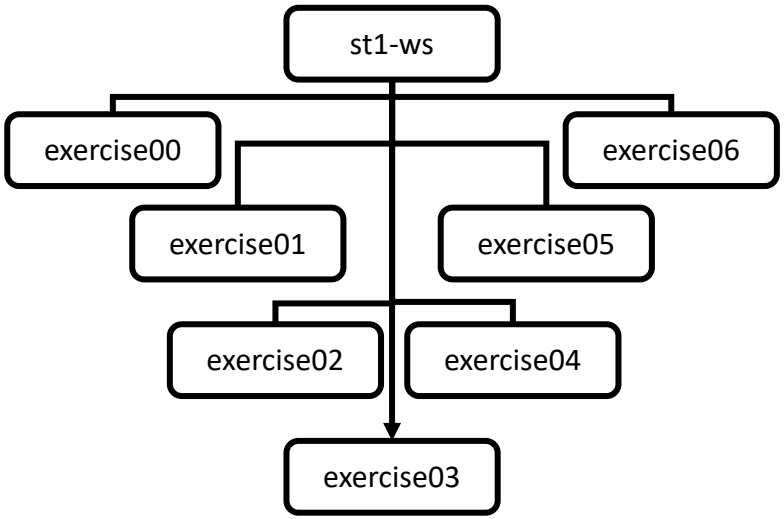
- Repeat this in another (different) terminal window.
- You will have ***three*** terminal connections to tinkercliffs. (3 screens are used for one exercise)

# Summary of Exercises

| Exercise | Goals | |
|---|---|---|
| 0 | **Resource allocation.** salloc to request resources; squeue to see request status; scontrol show job <jobid> to see what resources actually obtained; sacct to see "accounting" info; ssh to log into the allocated compute nodes. | |
| 1 | **Resource allocation; job assignment to cores.** salloc to request resources; module to load a module; source activate to activate a virtual environment; python to run a job; scancel to kill a slurm job (e.g., to give up requested resources); numactl: control which cpu (core) a job runs on. | |
| 2 | **Show compute resources.** numactl: for the machine you are on, shows the nodes, the cores per node, the memory per node. | |
| 3 | **Interrogate load on a node.** Various commands (top, htop, mpstat) to look at in-process/dynamic values of core utilization and memory parameters. | |
| 4 | **Static view of your limits.** Show account usage. And with respect to limit values. (Monthly reset in ARC.) | |
| 5 | **Dynamic views of work on compute nodes.** Use "stress" command to do a stress test. And monitor using htop (variants) and mpstat. | |
| 6 | **Slurm job with monitoring.** Real life calculation and gathering performance data. | |

# Locations of Codes on TC

*directory structure*




- ## After ssh'ing into Tinkercliffs …
- ## go to your home directory on TC
- cd
- ## create a new directory
- mkdir st1-ws
- ## change directory to this new directory
- cd st1-ws
- ## copy tarball from /globalscratch on TC (copy through the ".")
- cp /globalscratch/ckuhlman/arc-workshops-mar-2024/st1.pres.exercises.final.jun.2024.tar.gz .
- ## expand the contents of the tarball.
- ## this will create new directories and put files into them.
- tar xvzf st1.pres.exercises.final.jun.2024.tar.gz
- ## the directory names are the exercise numbers.
- ## files have commands to execute (you can copy and paste them) and code that we will execute.
- cd st1-ws
- ls –lrt *

# Docs on Tinkercliffs

- Landing page
  - https://www.docs.arc.vt.edu/
- Tinkercliffs info
  - https://www.docs.arc.vt.edu/resources/compute/00tinkercliffs.html

# Exercise 00: Requesting Resources and Understanding Allocation

- salloc: a SLURM scheduler command used to provide a Slurm job allocation, which is a set of resources (nodes), possibly with some set of constraints (e.g. number of processors per node)
- Using various “workhorse” commands just to understand a resource request and its allocation
  - salloc (above)
  - scontrol show job
  - squeue
  - sacct
  - ssh
- For sacct, Yale recommends
  - sacct --job=<jobid> -o jobid,user,partition,nodelist,elapsed,state,exitcode,maxrss,reqtres%35,alloctres%35,ntasks
    - Special node: look at maxrss for max memory usage
- # More information on how Slurm will handle resource requests: the manuals
  man < sbatch | srun | salloc >

# Notes

- In S3 (or ARC3) presentation, we requested cluster resources using “interact.”
- Interact command does both:
  - Requests resources (using salloc)
  - Puts user on resources (using ssh)
- In contrast, you can use “salloc” and “ssh” separately.

# Exercise 01:  Requesting Resources and Running Jobs

- Use salloc to allocate a node
- Set up job environment with
  - module
  - source activate
- Run python code from command line
- Run python code, but now use relative specification of a cpu from the cpuset (where cpuset is automatically formed from the salloc command) on which to run the job

# Exercise 02:  Inspecting Basic Properties of Hardware

- numactl
  - Numbers of cpus (cores)
  - Memory per “node”

# Exercise 03: Inspecting Jobs As Running

- top
  - Gives cpu and memory info for each running process on the (compute) node.
  - Meaning of fields here: https://eng.libretexts.org/Bookshelves/Computer_Science/Operating_Systems/Linux_-_The_Penguin_Marches_On_(McClanahan)/08%3A_How_to_Manage_System_Components/4.9%3A_Process_Troubleshooting/4.09.02%3A_Process_Troubleshooting_top_command
  - See next slide for some parameters.
- htop
  - Each cpu (core) utilization
  - Memory usage
- ps -u $USER -o %cpu,rss,args
  - Gives user-level information.
  - RSS is percentage of memory (volatile memory) used.
- mpstat
  - Gives instantaneous, summary information on the cores (cpus) of a node. Cpu usage.
  - Fields for mpstat: https://www.perfmatrix.com/linux-performance-monitoring-mpstat/
- mpstat -A
  - Gives instantaneous, information *for each* core (cpu) of a node. Cpu usage.

# Exercise 03:  Inspecting Jobs As Running (Continued)

- iostat
  - Used for monitoring system input/output statistics for devices and partitions.
  - https://www.geeksforgeeks.org/iostat-command-in-linux-with-examples/
- vmstat
  - "Virtual memory statistics."  Gives memory info.
  - https://www.perfmatrix.com/linux-performance-monitoring-vmstat/
- cpuset
  - Gives cpu (core) numbers allocated on this compute node.
  - Command:  cat /sys/fs/cgroup/cpuset/slurm/uid_`id -u`/job_<jobid>/cpuset.cpus

# Some top Parameters

- **Virtual memory**: a common technique used in a computer's operating system (OS). Virtual memory uses both hardware and software to enable a computer to compensate for physical memory shortages, temporarily transferring data from random access memory (RAM) to disk storage.
- **Shared memory**: amount of shared memory used by the process. It can be used for interprocess communications, but a more common scenario is that this is memory used by shared libraries that an application has linked in.
- **Res memory**: The non-swapped physical memory a task has used.
- **Swap memory**: Memory that is not resident but is present in a task. This is memory that has been swapped out but could include additional non-resident memory. This column is calculated by subtracting physical memory from virtual memory.
- **%CPU**: utilization of a CPU
- **%MEM**: A task's currently used share of available physical memory.

# Exercise 04

- showusage
- quota

# Exercise 05: Running and Monitoring a Stress Test

- Using Linux-provided stress program.
  - Generates load on specified number of cores (cpus) for a specified time.
- Monitor the job in two ways (both are dynamic measurements).
  - With htop: will see cpu loads on 8 cores increase to 100% during the stress test.
  - With mpstat: will see realtime printout of cpu utilization every three seconds (%usr).
- We “pin” or assign the stress program to particular compute node.
- We again monitor with:
  - With htop: will see cpu loads on 8 cores increase to 100% during the stress test.
  - With mpstat: will see realtime printout of cpu utilization every three seconds (%usr).

# Exercise 06: Running and Monitoring a “Real” Job

- Given a graph.
- Goals:
  - Computing the degree distribution of the graph.
  - Monitor job performance and write data to files.

# Putting This All Together for a LONG RUNNING Job

- Basic idea.
  - You start the metadata collection; can be more than one type of invocation.
  - You start your job.
  - When your job finishes, you stop collecting the metadata with 'kill's.
- Detailed example
  - #!/bin/sh
  - echo "Running IOSTAT”
  - iostat 2 >iostat-stdout.txt 2>iostat-stderr.txt &
  - echo "Running MPSTAT”
  - mpstat -P ALL 2 >mpstat-stdout.txt 2>mpstat-stderr.txt &
  - echo "Running VMSTAT”
  - vmstat 2 >vmstat-stdout.txt 2>vmstat-stderr.txt &
  - echo "Running executable”
  - ./a.out 2 1024
  - echo "Done”
  - kill %1
  - kill %2
  - kill %3

mpstat is collecting data for all processors/cpus/cores (-P ALL).

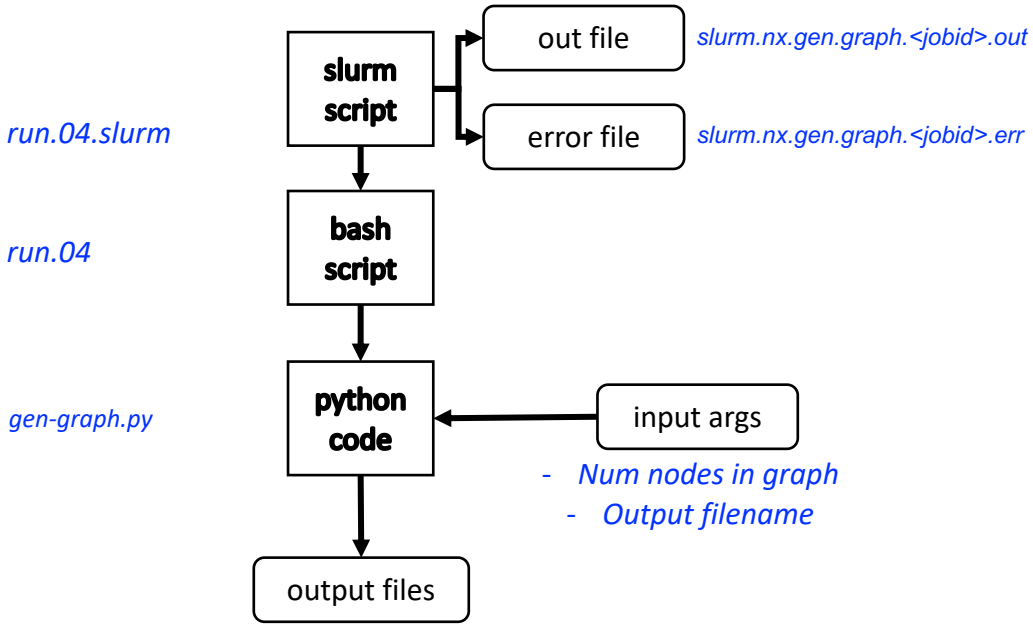
iostat, mpstat, vmstat printing to files every two seconds

The “2>” before the error file name is telling the shell to not print to stderr, but instead to print to the specified file (redirection).

# Exercise 06: Run a Python Code In Batch Mode Using Slurm

- Commands to run job:
  - cd ../exercise06
  - sbatch run.04.slurm
  - Note the unique ID that slurm returns to you.
- That's it; slam dunk.
- How to check status of job
  - squeue –u <your-user-name>
  - Issue this again and again (up-arrow)
- Output file: graph.powerlaw.no.exclusive.n.1000000.out
- Post-process output (next slide).

*What is going on here?*
*At command prompt, type "ls"*
*Now, match the filenames with diagram below.*

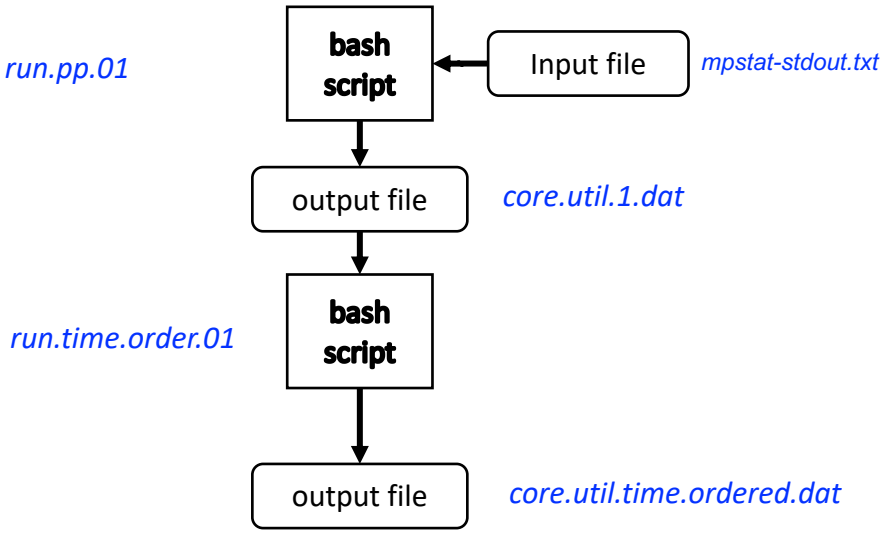



*graph.powerlaw.no.exclusive.n.1000000.out*
*mpstat-stdout.txt*
*others*

# Exercise 06: Post-Process Results

- Take file mpstat-stdout.txt (output from job) as input here.
- Run run.pp.01 on it to produce the output for one particular core (the core that ran the graph generation).
- Run run.time.order.01 on the previous result to alter time so that it appears as a number (not a time stamp) for plotting.

*What is going on here?*
*At command prompt, type "ls"*
*Now, match the filenames with diagram below.*



# Command Summary

- Commands to issue while job is running
  - top
  - htop
  - ps
  - sstat
  - mpstat

- Commands to issue when job is completed
  - seff <jobId>
  - sacct

# Some Warnings

- When you run mpstat for first time, it calculates the idle time since the server has booted up to the point where you have run mpstat.
- But when you run it with intervals, you are getting the value within the defined time amount you specified, here 1 second. And not the entire time since boot up and then to that moment.
- In fact, iostat, vmstat all work in this same way as I told
- Yes, always run these with intervals when collecting data.
- Source: https://serverfault.com/questions/429301/why-does-mpstat-show-different-values-when-i-use-the-interval-setting

# Useful Sites (No Warranty About Their Longevity)

- General
  - https://linux.die.net/man/8/numactl#:~:text=Examples,memory%20interleaved%20on%20all%20CPUs.
  - https://docs.ycrc.yale.edu/clusters-at-yale/job-scheduling/resource-usage/
  - https://man7.org/linux/man-pages/man8/numactl.8.html#:~:text=numactl%20runs%20processes%20with%20a,shared%20memory%20segments%20or%20files.
  - https://linux.die.net/man/8/numactl#:~:text=Examples,memory%20interleaved%20on%20all%20CPUs.
  - https://stackoverflow.com/questions/21311893/runinng-iostat-mpstat-vmstat-along-with-executable
  - https://www.perfmatrix.com/linux-performance-monitoring-vmstat/

- Performance engineering tutorial
  - https://www.perfmatrix.com/performance-engineering-tutorial/
- Performance testing tutorial
  - https://www.perfmatrix.com/performance-testing-tutorial/

# Thanks for Participating

- Google sign up sheet is here:
  - https://docs.google.com/document/d/1uVrupbvN6-2ZsxOFzokp_gLWAaeGocP2/edit
- Please sign in to ensure:
  - You get credit for the course
  - Our roster is complete

# Acknowledgments


- Matt Brown developed an earlier version of this presentation, which heavily informs this work.
- I stole overview slides from Ayat Mohammed and Sarah Ghazanfari.
- I thank them.

# END

# Commands To Cover

- : "seff", "jobload", "htop", "gpumon", "sacct",

- Seff
  - https://stackoverflow.com/questions/24020420/find-out-the-cpu-time-and-memory-usage-of-a-slurm-job
- Sstat
  - https://stackoverflow.com/questions/24020420/find-out-the-cpu-time-and-memory-usage-of-a-slurm-job
- Sacct
  - Post-job statistics
  - https://stackoverflow.com/questions/24020420/find-out-the-cpu-time-and-memory-usage-of-a-slurm-job

TODO

# High Level Operating Environment