Working on Picotte

FITSUM ALEBACHEW

Overview

- What is Picotte?
- Using Picotte:
  - Logging in
  - Transferring files
  - Job scripts
- Important commands

What is Picotte?

Picotte is a high-performance computing cluster

- Operating System: Red-Hat Enterprise Linux 8 64-bit
- Default shell: Bash (Bourne Again Shell)
- Job scheduler: SLURM Workload Manager

Specifications: https://drexel.edu/core-facilities/facilities/research-computing/service/picotte/

Picotte (Nodes)

Picotte has a total of 90 nodes:

- 1 management node
- 1 login node
- 88 compute nodes:
  - 74 def nodes - 48 cores/node, 192 GB RAM/node
  - 12 gpu nodes - 48 cores/node, 192 GB RAM/node, 4 Nvidia Tesla V100-SXM2 32GB GPU devices/node
  - 2 bm nodes - 48 cores/node, 1.5 TB RAM/node

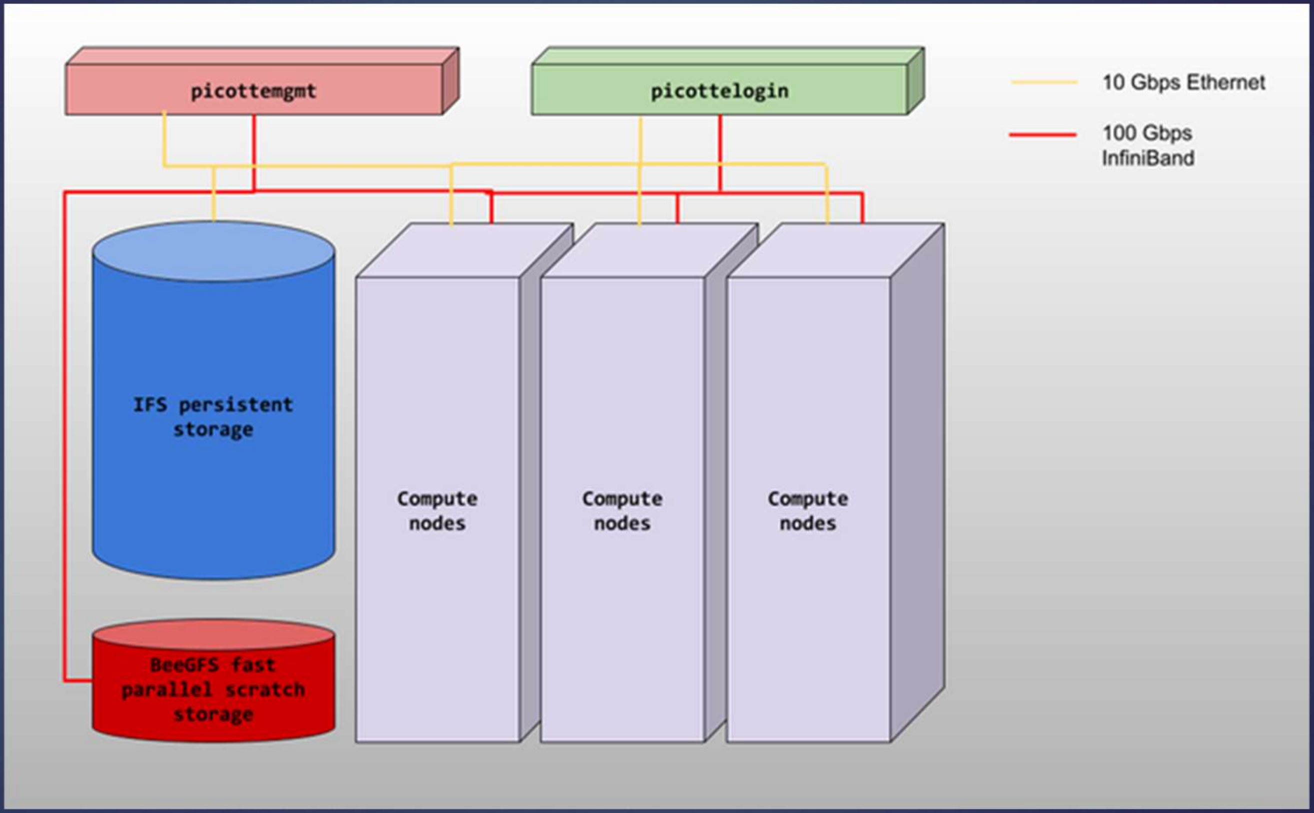
Total of 4224 compute cores, 19.1 TiB RAM

Picotte (Storage)

Picotte has 3 levels of storage, each better suited for different uses:

- Persistent (NFS): for long-lived data (includes /home and /ifs)
  - 649 TB, 10 Gbps Ethernet (big and slow)
  - Main storage for most data, avoid using if you have lots of I/O
- Local Scratch (TMP) : internal drives in nodes (/tmp)
  - 854 GB, 12 Gbps SAS SSD (per node)
  - Fast, but not shared, ideal for single node jobs
  - $TMP variable to access from within a job
- Fast Parallel Shared Scratch (BeeGFS): shared memory b/n nodes (/beegfs)
  - 175 TB, 100 Gbps Infiniband Network
  - Fast and shared, ideal for intensive I/O operations across multiple nodes
  - $BEEGFS_TMPDIR variable to access from within a job

Picotte


picottemgmt
picottelogin
10 Gbps Ethernet
100 Gbps InfiniBand
IFS persistent storage
BeeGFS fast parallel scratch storage
Compute nodes
Compute nodes
Compute nodes

Logging In

- To get access to Picotte, you MUST initially be logged into Drexel's VPN or using the Drexel on-campus Wi-Fi.
- Logging into Drexel's VPN:
  - Download and install Cisco AnyConnect Secure Mobility Client
    - https://vpn.drexel.edu/
  - Launch Cisco AnyConnect Secure Mobility Client
  - Sign in with Drexel credentials

Detailed instructions: https://drexel.edu/it/help/a-z/VPN/

Logging In (cont.)

There are several ways to log into Picotte:

- OpenSSH (Windows) or Terminal (Mac)
  - ssh username@picottelogin@urcf.drexel.edu
- MobaXterm for GUI display
- SSH through Visual Studio Code
- Other SSH clients like PUTTY

More info:

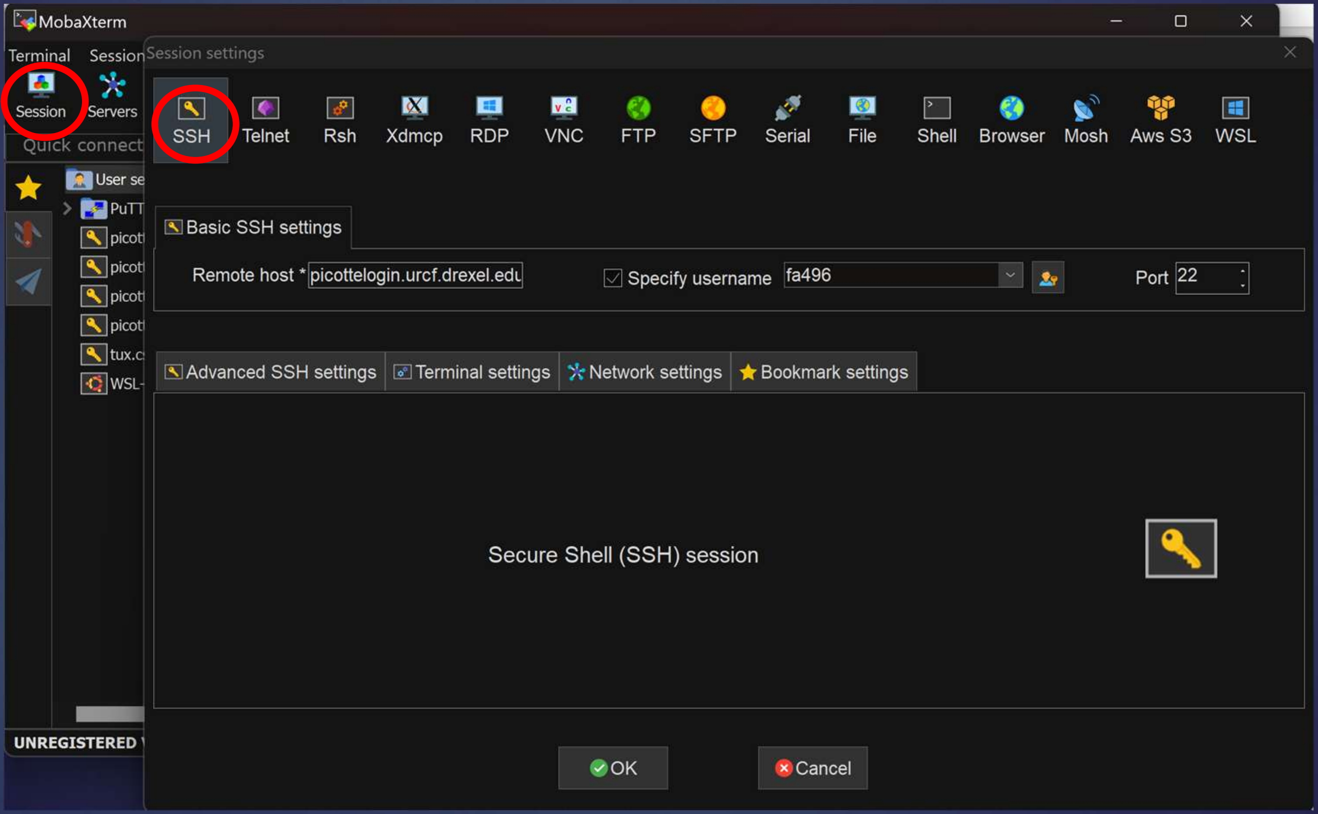
- Windows: https://proteusmaster.urcf.drexel.edu/urcfwiki/index.php/Tips_for_Windows_Users
- Mac: https://proteusmaster.urcf.drexel.edu/urcfwiki/index.php/Tips_for_macOS_Users

Logging in with MobaXterm

- Open MobaXterm application
- Click on 'Session' in the menu tab
- Choose SSH option
- Enter hostname and username
- Enter password

More info: https://mobaxterm.mobatek.net/

Logging in with MobaXterm



Remote host * picottelogin.urcf.drexel.edu

☑ Specify username fa496

Port 22

GUI Displays and Modules

You can use MobaXterm(Windows)/Xquartz(Mac) to display app GUIs like MATLAB, Jupyter Notebook, etc. on your local machine from Picotte:

- Log into Picotte using MobaXterm
- Load the modules for the applications you want to run
  - Modules need to be loaded when using apps installed on Picotte
    - module list - see loaded modules
    - module load <name> - load a module
    - module avail <optional query> - see available modules
- Run the applications from the command line to open a new window containing the GUI.

Logging in with Visual Studio Code

- Install an OpenSSH compatible SSH client on device
- Install ‘Remote Development’ extension pack on VS Code
- Go to the ‘Remote Explorer’ tab on VS Code and click on ‘Add new’
- Enter your username and hostname as ‘ssh username@hostname’
- Enter your password when prompted

Detailed instructions: https://code.visualstudio.com/docs/remote/ssh

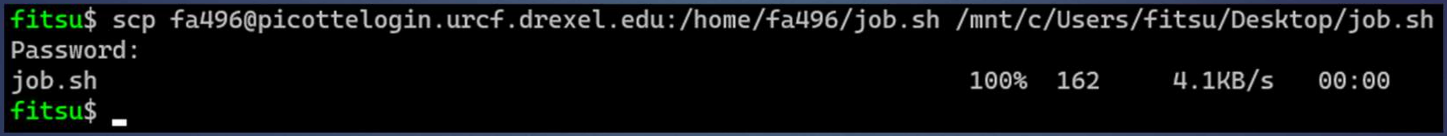
Visual Studio Code Tips

- Extensions can be installed on the remote environment for useful features like language support
- Application configuration can be saved into a workspace file for easier access when logging in again:
  - Having multiple directories open together
  - Setting different options specific to that workspace
  - Resume right where you left off

Transferring Files

There are also several ways to transfer files to and from Picotte:

- Shell commands like scp, sftp, pscp
  - scp <source address> <destination address>
  - address format (remote): username@hostname:<working directory>



```
fitsu$ scp fa496@picottelogin.urcf.drexel.edu:/home/fa496/job.sh /mnt/c/Users/fitsu/Desktop/job.sh
Password:
job.sh                                                                  100%  162     4.1KB/s   00:00
fitsu$ _
```

- MobaXterm/VS Code
  - To copy from Picotte to local machine:
    - Right click on file in explorer and click on download
  - To copy from local machine to Picotte:
    - Open directory in file explorer and drag file from local machine to file explorer

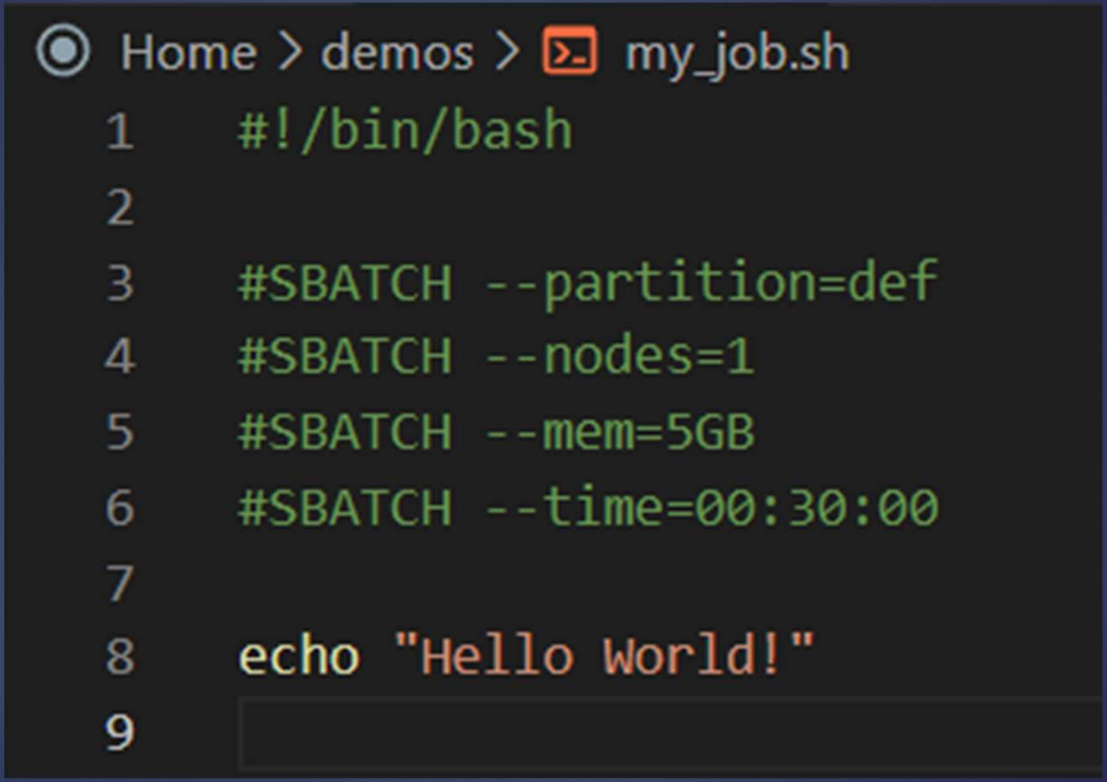
Creating/Submitting Job Scripts

Job scripts are shell scripts that specify the details/options to the job:

- Always start with shebang - #!/bin/bash
- Use ‘#SBATCH’ to set options
- Load modules being used
- Run program executable/command
- Submit your script with the command ‘sbatch myjob.sh’

More info: https://proteusmaster.urcf.drexel.edu/urcfwiki/index.php/Writing_Slurm_Job_Scripts

Creating/Submitting Job Scripts



```
#!/bin/bash

#SBATCH --partition=def
#SBATCH --nodes=1
#SBATCH --mem=5GB
#SBATCH --time=00:30:00

echo "Hello World!"

```

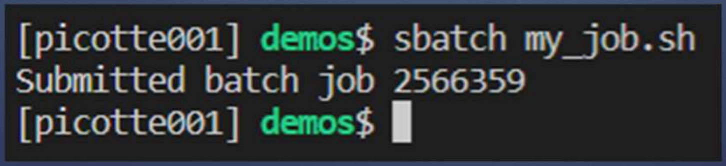
Viewing your Results

- The standard output of the program will be redirected to a new file named ‘slurm-jobId.out’ in the same directory as job script.
- Any file outputs will be saved in the same directory as job script with same name.

SLURM commands

Some important SLURM commands:

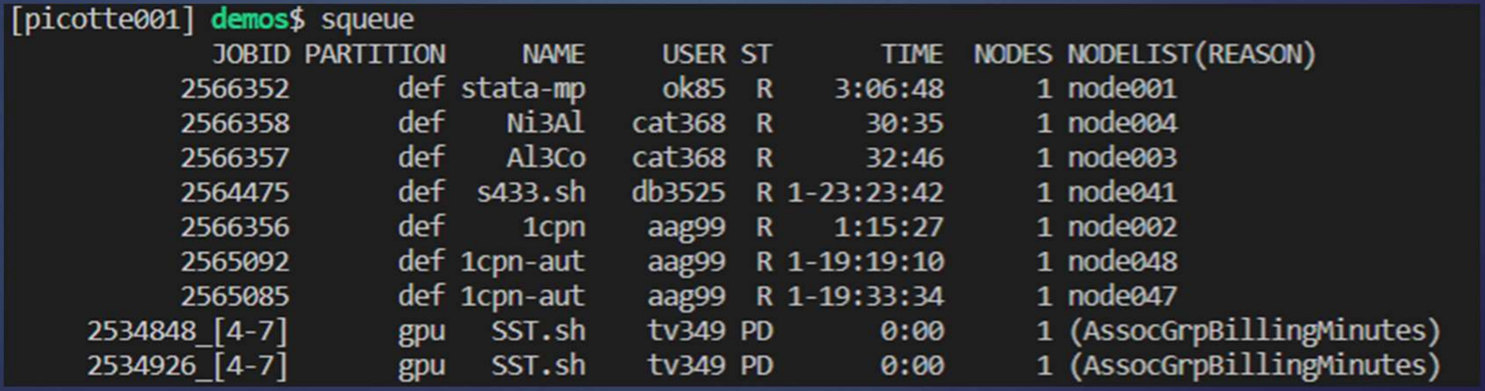
- sbatch: submit a job script to Slurm
  - -p, --partition=par – specify partition to run job (def by default)
  - -N, --nodes=numOfNodes – specify how many nodes to allocate (1 by default)
  - -t, --time=hh:mm:ss – specify a time limit for the job (30min by default)
  - --mem=size – specify required memory per node (4GB by default)
  - --mail-user=user@host – send job status to email (none by default)



```
[picotte001] demos$ sbatch my_job.sh
Submitted batch job 2566359
[picotte001] demos$
```

SLURM commands (cont.)

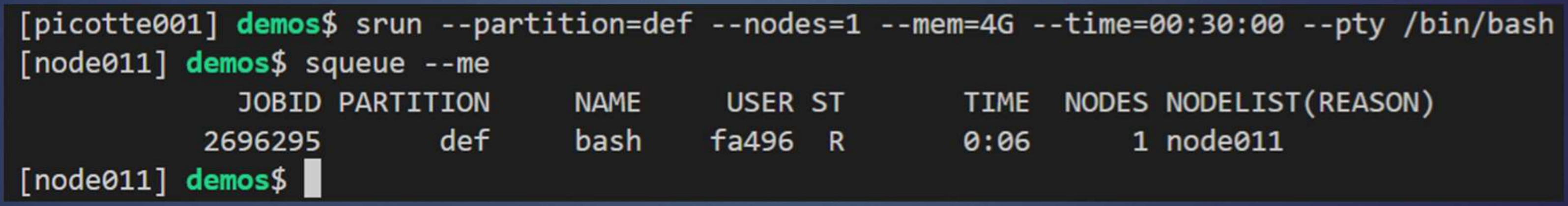
- scancel <jobId>: cancel a job that is pending/running
- squeue: display a list of running jobs
  - --me option to show only your jobs
  - -j <jobId> for a specific job



```
[picotte001] demos$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2566352       def stata-mp     ok85  R    3:06:48      1 node001
           2566358       def   Ni3Al   cat368  R      30:35      1 node004
           2566357       def   Al3Co   cat368  R      32:46      1 node003
           2564475       def  s433.sh  db3525  R 1-23:23:42      1 node041
           2566356       def     1cpn    aag99  R    1:15:27      1 node002
           2565092       def 1cpn-aut    aag99  R 1-19:19:10      1 node048
           2565085       def 1cpn-aut    aag99  R 1-19:33:34      1 node047
     2534848_[4-7]       gpu   SST.sh    tv349 PD       0:00      1 (AssocGrpBillingMinutes)
     2534926_[4-7]       gpu   SST.sh    tv349 PD       0:00      1 (AssocGrpBillingMinutes)
```

SLURM commands (cont.)

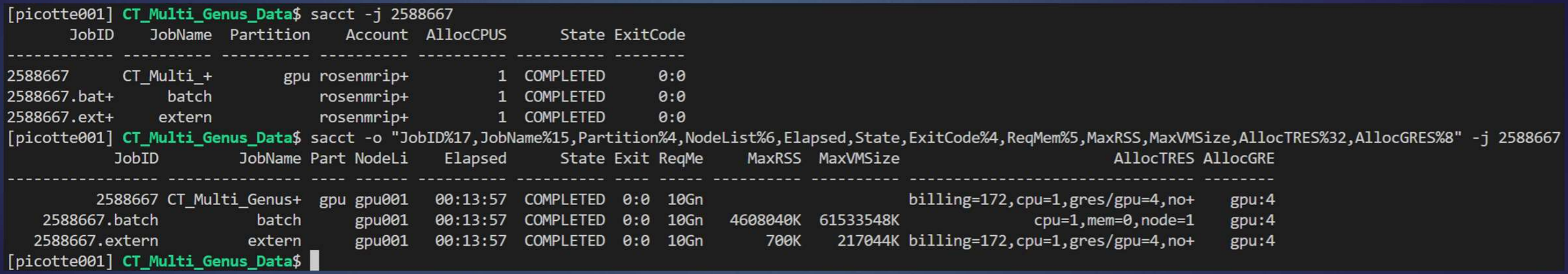
- srun: runs a command/script as a job
  - can be used for interactive sessions
    - srun [OPTIONS] --pty /bin/bash
  - can be used to quickly execute commands on compute nodes



```
[picotte001] demos$ srun --partition=def --nodes=1 --mem=4G --time=00:30:00 --pty /bin/bash
[node011] demos$ squeue --me
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2696295       def     bash    fa496  R       0:06      1 node011
[node011] demos$
```

SLURM commands (cont.)

- sacct: show details about jobs ran by a user (can use -j <jobid> option)
  - Can display all your recent jobs together
  - Can be formatted with the --format (-o) option

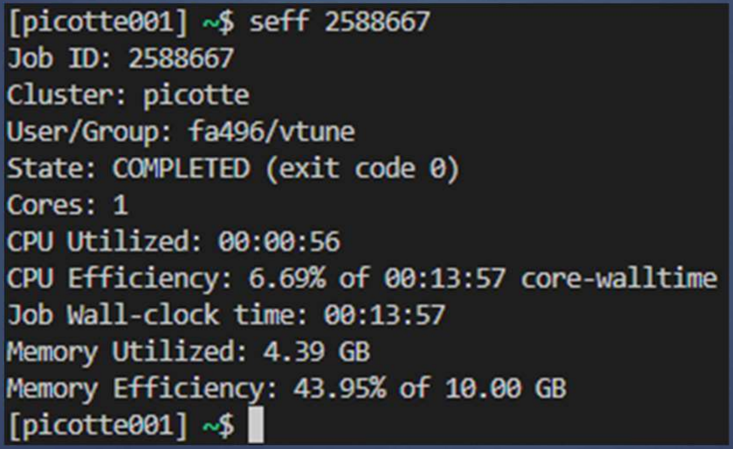


```
[picotte001] CT_Multi_Genus_Data$ sacct -j 2588667
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
2588667      CT_Multi_+        gpu rosenmrip+          1  COMPLETED      0:0
2588667.bat+      batch            rosenmrip+          1  COMPLETED      0:0
2588667.ext+     extern            rosenmrip+          1  COMPLETED      0:0
[picotte001] CT_Multi_Genus_Data$ sacct -o "JobID%17,JobName%15,Partition%4,NodeList%6,Elapsed,State,ExitCode%4,ReqMem%5,MaxRSS,MaxVMSize,AllocTRES%32,AllocGRES%8" -j 2588667
            JobID         JobName Part NodeLi    Elapsed      State Exit ReqMe     MaxRSS  MaxVMSize                        AllocTRES AllocGRE
----------------- --------------- ---- ------ ---------- ---------- ---- ----- ---------- ---------- -------------------------------- --------
          2588667 CT_Multi_Genus+  gpu gpu001   00:13:57  COMPLETED  0:0  10Gn                       billing=172,cpu=1,gres/gpu=4,no+    gpu:4
    2588667.batch           batch      gpu001   00:13:57  COMPLETED  0:0  10Gn   4608040K  61533548K               cpu=1,mem=0,node=1    gpu:4
   2588667.extern          extern      gpu001   00:13:57  COMPLETED  0:0  10Gn       700K    217044K billing=172,cpu=1,gres/gpu=4,no+    gpu:4
[picotte001] CT_Multi_Genus_Data$
```

- MaxRSS (Maximum Resident Set Size) above is the total RAM used by your job

SLURM commands (cont.)

- sinfo: display status of nodes in Picotte
- seff <jobId>: report efficiency statistics on a job



```
[picotte001] ~$ seff 2588667
Job ID: 2588667
Cluster: picotte
User/Group: fa496/vtune
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 00:00:56
CPU Efficiency: 6.69% of 00:13:57 core-walltime
Job Wall-clock time: 00:13:57
Memory Utilized: 4.39 GB
Memory Efficiency: 43.95% of 10.00 GB
[picotte001] ~$
```

More info: https://proteusmaster.urcf.drexel.edu/urcfwiki/index.php/Slurm_Quick_Start_Guide#Commands

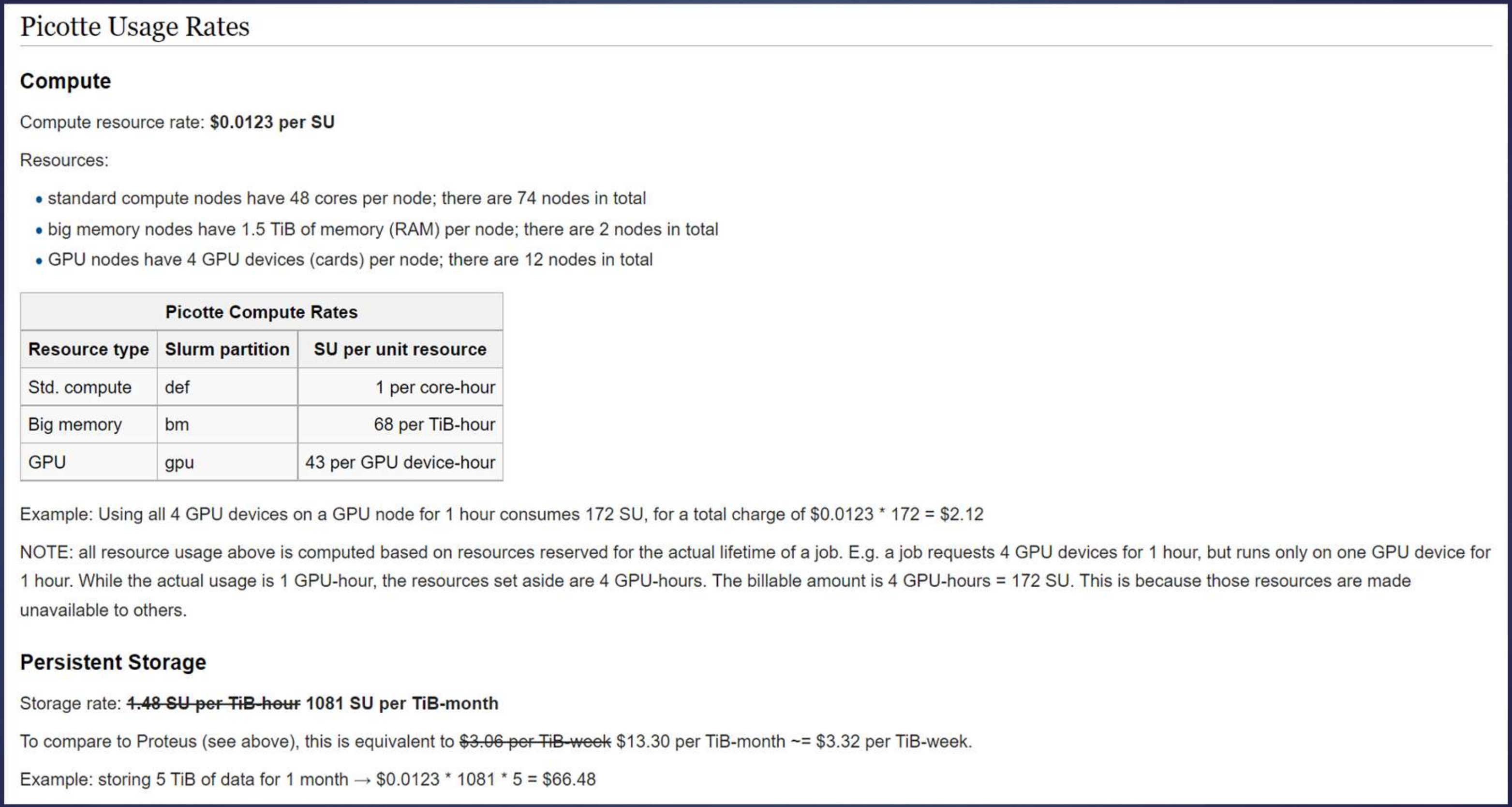
SLURM commands (slurm_util)

These commands have the same behavior and options as their base commands but display output with some added detail.

Run the command “module load slurm_util” to use (consider adding the line to ~/.bashrc file to do it automatically on log-in:

- squeue_detail (squeue_long)
- sinfo_detail
- sacct_detail

Usage Rates



Picotte Usage Rates

Compute

Compute resource rate: **$0.0123 per SU**

Resources:

- standard compute nodes have 48 cores per node; there are 74 nodes in total
- big memory nodes have 1.5 TiB of memory (RAM) per node; there are 2 nodes in total
- GPU nodes have 4 GPU devices (cards) per node; there are 12 nodes in total

Picotte Compute Rates

Resource type	Slurm partition	SU per unit resource
Std. compute	def	1 per core-hour
Big memory	bm	68 per TiB-hour
GPU	gpu	43 per GPU device-hour

Example: Using all 4 GPU devices on a GPU node for 1 hour consumes 172 SU, for a total charge of $0.0123 * 172 = $2.12

NOTE: all resource usage above is computed based on resources reserved for the actual lifetime of a job. E.g. a job requests 4 GPU devices for 1 hour, but runs only on one GPU device for 1 hour. While the actual usage is 1 GPU-hour, the resources set aside are 4 GPU-hours. The billable amount is 4 GPU-hours = 172 SU. This is because those resources are made unavailable to others.

Persistent Storage

Storage rate: **~~1.48 SU per TiB-hour~~ 1081 SU per TiB-month**

To compare to Proteus (see above), this is equivalent to ~~$3.06 per TiB-week~~ $13.30 per TiB-month ~= $3.32 per TiB-week.

Example: storing 5 TiB of data for 1 month → $0.0123 * 1081 * 5 = $66.48

Questions?

- Feel free to attend my office hours every weekday 2 - 3 pm (any changes will be reflected on the URCF wiki main page): https://proteusmaster.urcf.drexel.edu/urcfwiki/index.php/Main_Page#Talks_and_Workshops