Introduction to

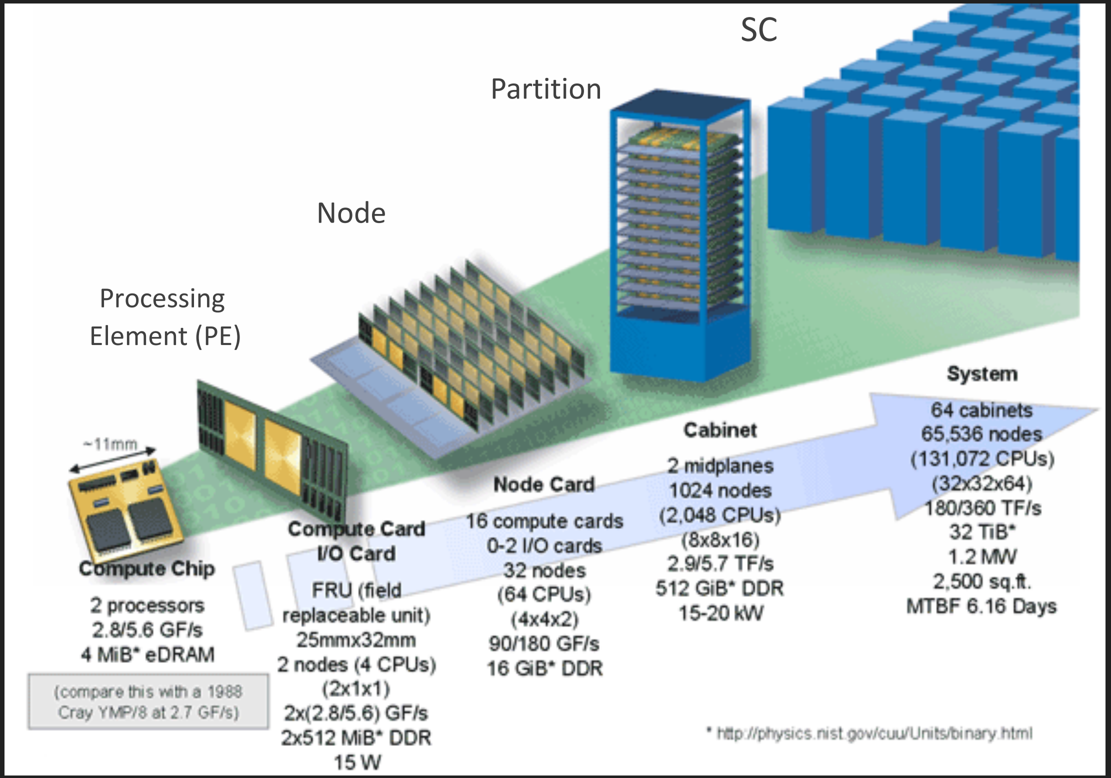
High Performance Computing (HPC)

on Lewis and Clark clusters

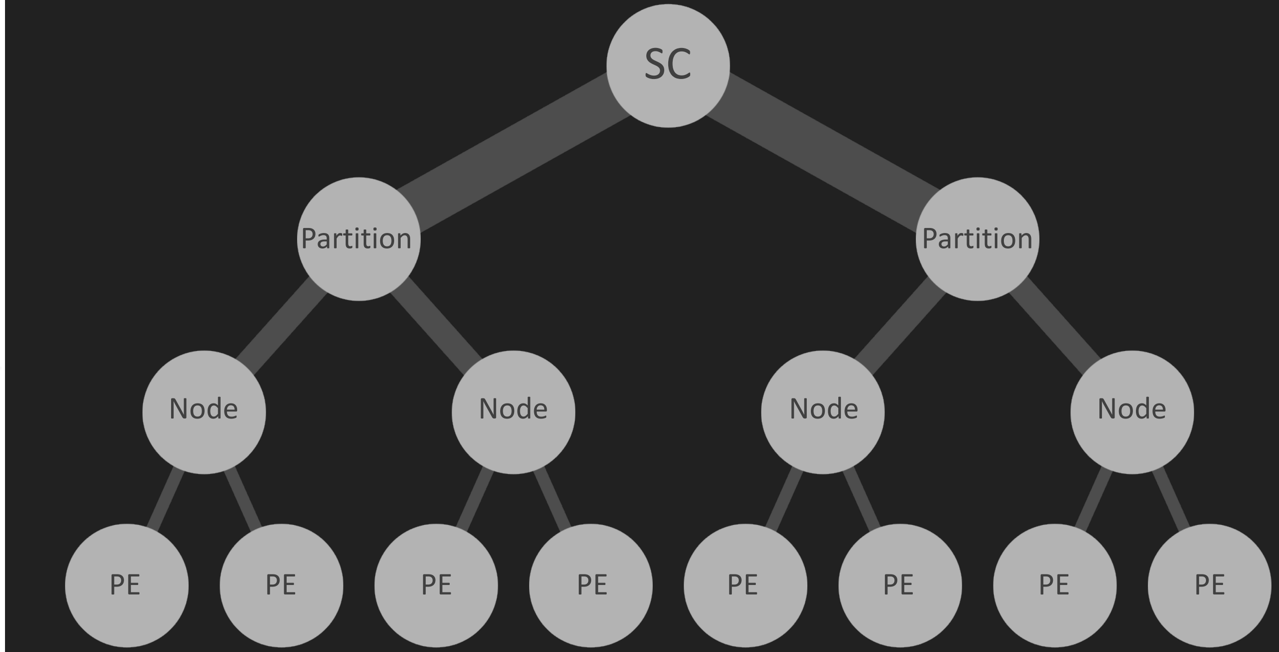
Research Computing Support Services (RCSS)
http://docs.rnet.missouri.edu

RCSS CIE team
Ashkan Mirzaee, Asif Magdoom
Brian Marxkors, Christina Roberts
Predrag Lazic

What is a cluster (supercomputer)?

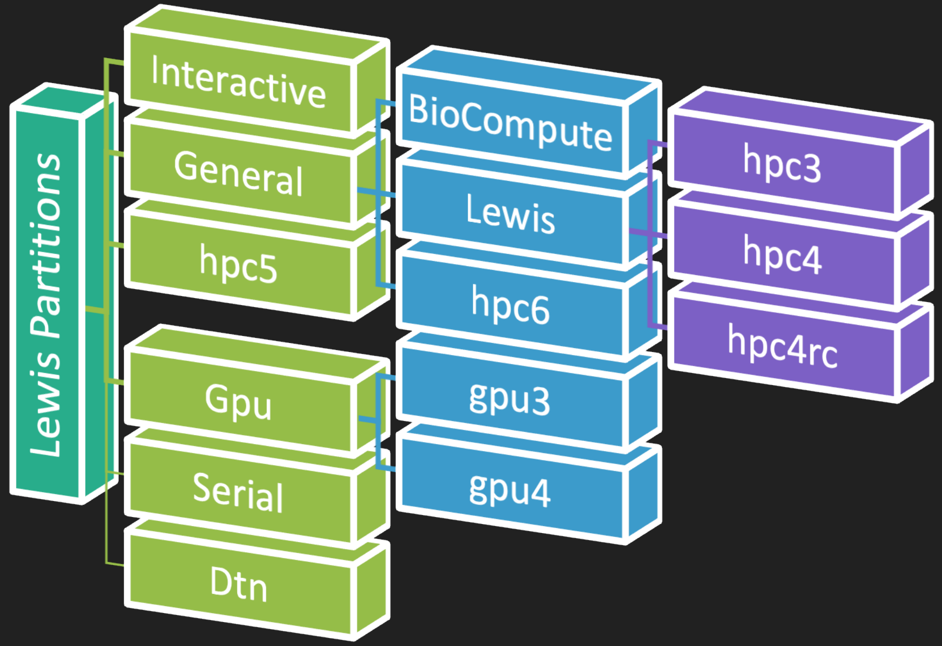


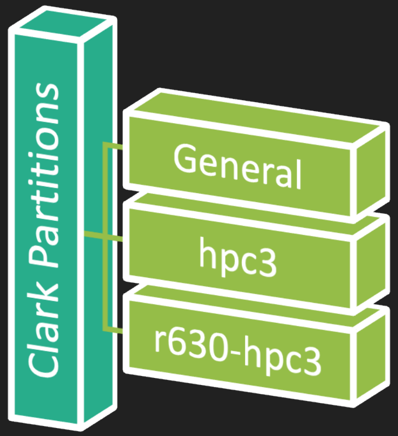
wikimedia.org



- A supercomputer is a computer with a high level of performance as compared to a general-purpose computer
- Purpose: massive parallelization because life is too short!
- The world's fastest 500 supercomputers run Linux-based operating systems

Lewis and Clark





Lewis

- A large-scale cluster for requesting high amount of resources
- Great for parallel programing
- GPU resources
- No cost for MU members for general usage
- Investment option is available to receive more resource (more fairshare)

Clark

- Great for learning and teaching
- No need for registration and it is available to all MU members by MU username and PawPrint
- Usually less crowded - receive resources very fast
- No cost for MU members

Lewis and Clark Partitions

Lewis

Partition Name	Time Limit	Nodes	Cores (per node*)	Cores (total)	Memory in GB (per nodes*)	Processors
Interactive	4:00:00	4	24+	144	251+	Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz+
General	4:00:00	187	24+	5636	122+	Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz+
BioCompute	2-00:00:00	37	56	2072	509	Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
Lewis	2-00:00:00	90	24+	3564	122+	Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz+
hpc3	2-00:00:00	22	24	1296	122+	Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz
hpc4	2-00:00:00	37	28	1260	251+	Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
hpc4rc	2-00:00:00	34	28	1008	251	Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
hpc6	2-00:00:00	61	48	2976	379+	Intel(R) Xeon(R) Gold 6252 CPU @ 2.10GHz
hpc5	2-00:00:00	35	40	1320	379	Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz
Gpu	2:00:00	16	16+	372	122+	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz+
gpu3	2-00:00:00	13	16+	284	122+	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
gpu4	2-00:00:00	3	40+	124	379+	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
Serial	2-00:00:00	1	64	64	1,025	AMD EPYC 7601 32-Core Processor
Dtn	2-00:00:00	2	16+	36	66+	Intel(R) Xeon(R) CPU X5550 @ 2.67GHz+

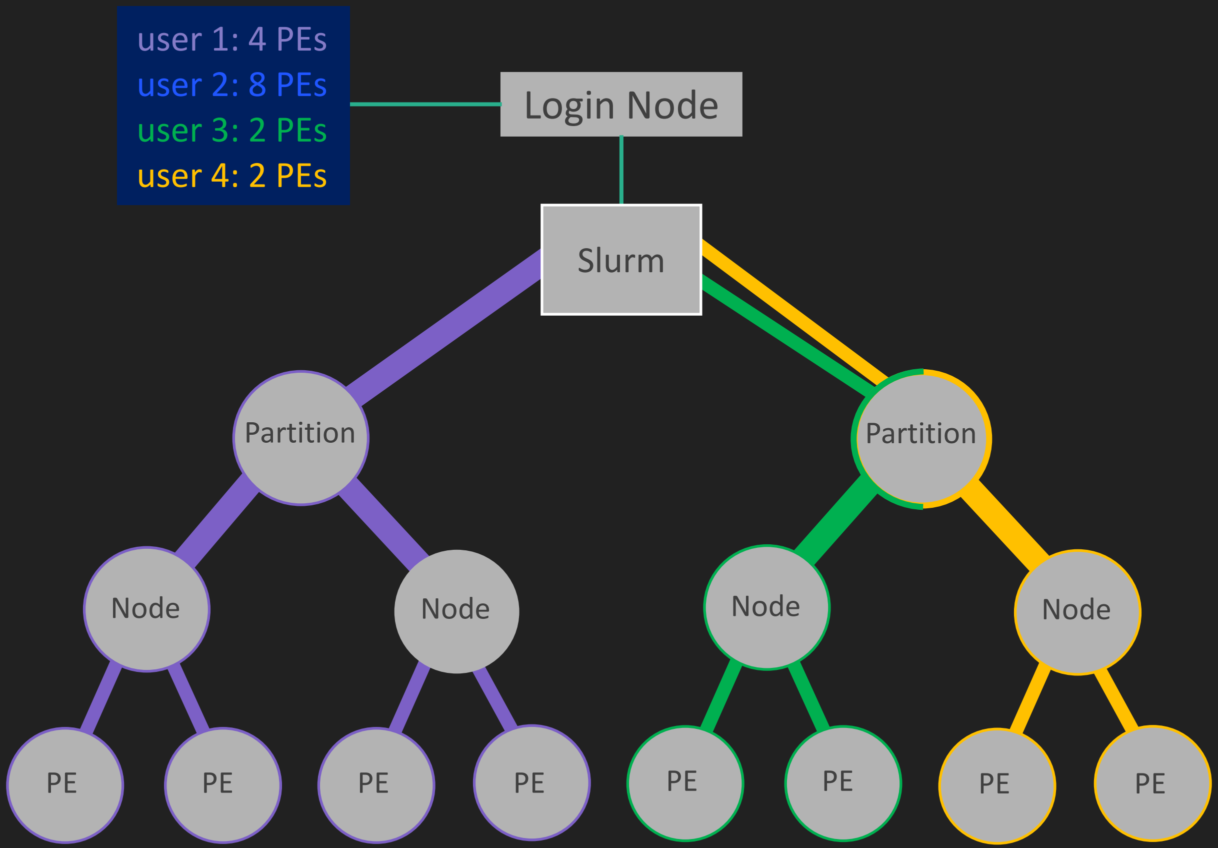
Clark

Partition Name	Time Limit	Nodes	Cores (per node*)	Cores (total)	Memory in MB (per nodes*)	Processors
General	2:00:00	4	24	96	122	Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz
hpc3	2-00:00:00	4	24	96	122	Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz
r630-hpc3	2-00:00:00	4	24	96	122	Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz

* plus sign (+) indicates a mixed environment. The number before the plus represents the minimum
http://docs.rnet.missouri.edu/policy/partition-policy/

SLURM

Slurm is a system for cluster management and job scheduling. All RCSS clusters use Slurm (https://slurm.schedmd.com).

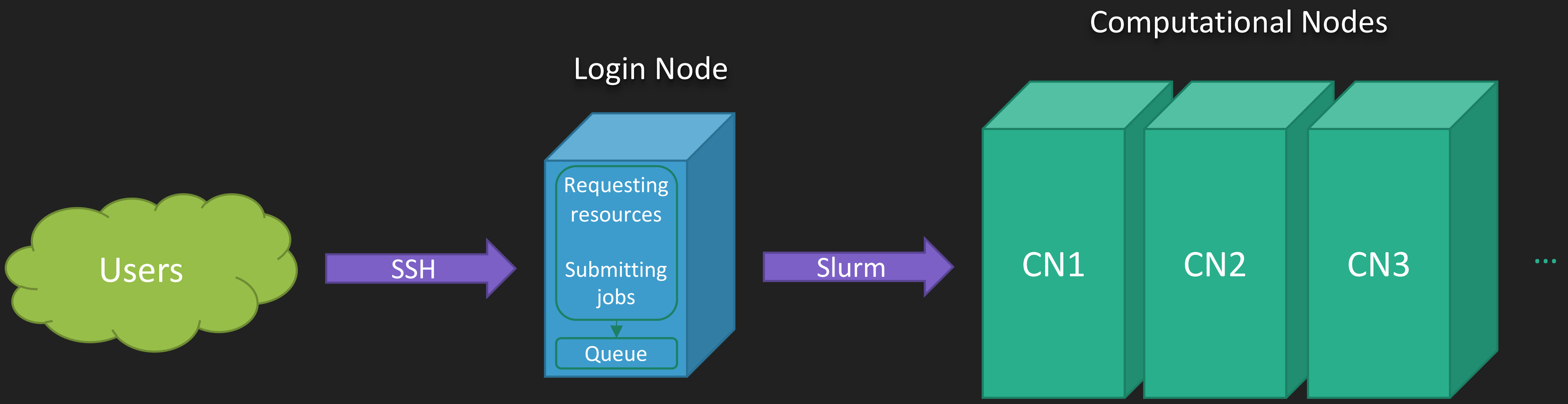


- Slurm is a workload scheduler and has set of tools for submitting and monitoring jobs
- Slurm is a resource management system and has many tools to find available resources in the cluster
- All Slurm commands start with letter “s”. In this course we will learn many of them
- Resource allocation depends on your fairshare i.e. priority in the queue

Login node

All users connect to Clark and Lewis clusters through the login nodes.

```
[user@lewis4-r630-login-node675 ~]$
[user@clark-r630-login-node907 ~]$
```



All jobs must be run using Slurm submitting tools to prevent running on the Lewis login node. Jobs that are found running on the login node will be immediately terminated followed up with a notification email to the user.

Review

Unix commands

man manual	hostname host name	cat concatenate files and print output
cd change directory	nproc number of processers	top information about processes
pwd print working directory	lscpu list CPU architecture	Tab tab completion
ls -la long list of all files	free –h free memories	clear or Ctrl + l clear
mkdir make directory	df disk free	exit or Ctrl + d exit
cp copy	echo $USER echo user id	history history

Text editors

emacs -nw	nano -z	vim
Ctrl-k kill (cut)	Ctrl-k cut	i insert
Ctrl-y yank (paste)	Ctrl-u uncut (paste)	Esc dd delete line
Alt-w copy	Alt-6 copy	Esc yy yank (paste)
Shift-arrow keys select	Alt-m + alt-a Select	Esc u undo Esc p paste
Ctrl-z suspend fg return	Ctrl-z suspend fg return	Esc Ctrl + z suspend fg ret.
Ctrl-x + Ctrl-s save	Ctrl-o write out	Esc + :w write
Ctrl-x + Ctrl-c close	Ctrl-x exit	Esc + :q quit :q! no change

Files

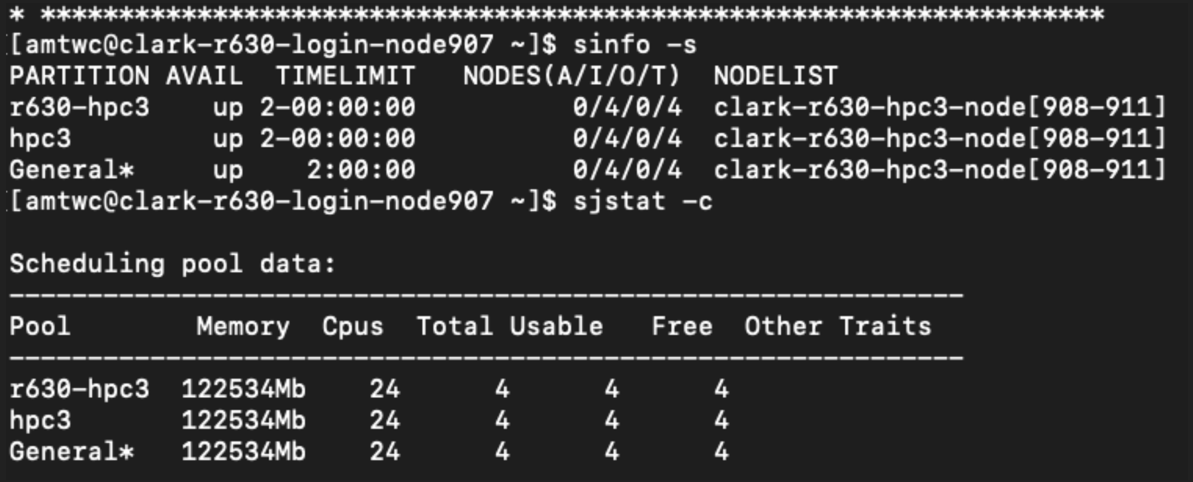
ssh username@lewis.rnet.missouri.edu connect to lewis by ssh
cp -r /storage/hpc/data/plmx7/EXES_FOR_USERS/training ~ copy training files to the home directory

Cluster info

sinfo – sjstat – scontrol - sacctmgr

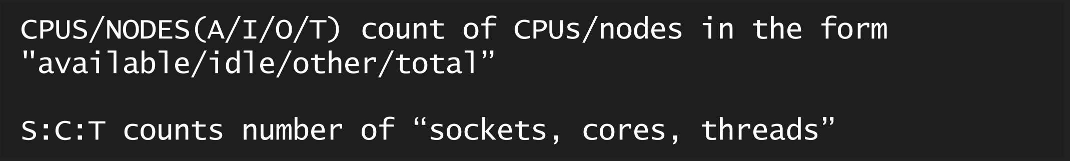
```
sinfo –s summary of cluster resources
sinfo -p <partitoion-name> -o %n,%C,%m,%z compute info of nodes in a partition
sinfo -p Gpu -o %n,%C,%m,%G GPUs information in Gpu partition
sjstat –c show computing resources per node
scontrol show partition <partition-name> partition information
scontrol show node <node-name> node information
sacctmgr show qos format=name,maxwall,maxsubmit show quality of services
./ncpu.py show number of available CPUs and GPUs per node
```

```
-s --summarize
-o --format
-p --partition
```



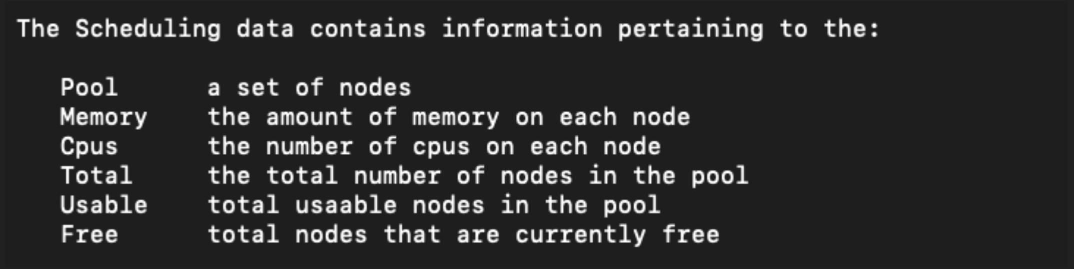
```
* ********************************************************************
[amtwc@clark-r630-login-node907 ~]$ sinfo -s
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
r630-hpc3    up 2-00:00:00          0/4/0/4  clark-r630-hpc3-node[908-911]
hpc3         up 2-00:00:00          0/4/0/4  clark-r630-hpc3-node[908-911]
General*     up    2:00:00          0/4/0/4  clark-r630-hpc3-node[908-911]
[amtwc@clark-r630-login-node907 ~]$ sjstat -c

Scheduling pool data:
-------------------------------------------------------------
Pool        Memory  Cpus  Total Usable   Free  Other Traits
-------------------------------------------------------------
r630-hpc3  122534Mb    24      4      4      4
hpc3       122534Mb    24      4      4      4
General*   122534Mb    24      4      4      4
```



CPUS/NODES(A/I/O/T) count of CPUs/nodes in the form "available/idle/other/total"

S:C:T counts number of "sockets, cores, threads"



```
The Scheduling data contains information pertaining to the:

   Pool      a set of nodes
   Memory    the amount of memory on each node
   Cpus      the number of cpus on each node
   Total     the total number of nodes in the pool
   Usable    total usaable nodes in the pool
   Free      total nodes that are currently free
```

Users info

```
                        sshare – sacctmgr – df – lfs quota
sshare -U show your fairshare and accounts                              -U --Users
sacctmgr show assoc user=$USER format=acc,user,share,qos,maxj your QOS
groups show your groups
df -h /home/$USER home storage quota                                    -h --human-readable
lfs quota -hg $USER /storage/hpc data/scratch storage quota             -g user/group
lfs quota –hg <group-name> /storage/hpc data/scratch storage quota
./userq.py show user’s fairshare, accounts, groups and QOS
```

- Resource allocation depends on your fairshare. If your fairshare is `0.000000` you have used the cluster more than your fair share and will be de-prioritized by the queuing software
- Users have 5GB at their home directory `/home/$USER` and 100GB at `/storage/hpc/data/$USER`
- **Do not** use home directory for running jobs, storing data or virtual environments
- Clark users have 100G on their home storage. The above methods does not apply for Clark
- The RCSS team reserves the right to delete anything in `/scratch` and `/local/scratch` at any time for any reason
- There are no backups of any storage. The RCSS team is not responsible for data integrity and data loss. **You are responsible for your own data and data backup**
- Review our storage policy at http://docs.rnet.missouri.edu/policy/storage-policy/

Note: `$USER` is a default environmental variable that returns your user id, try `echo  $USER`

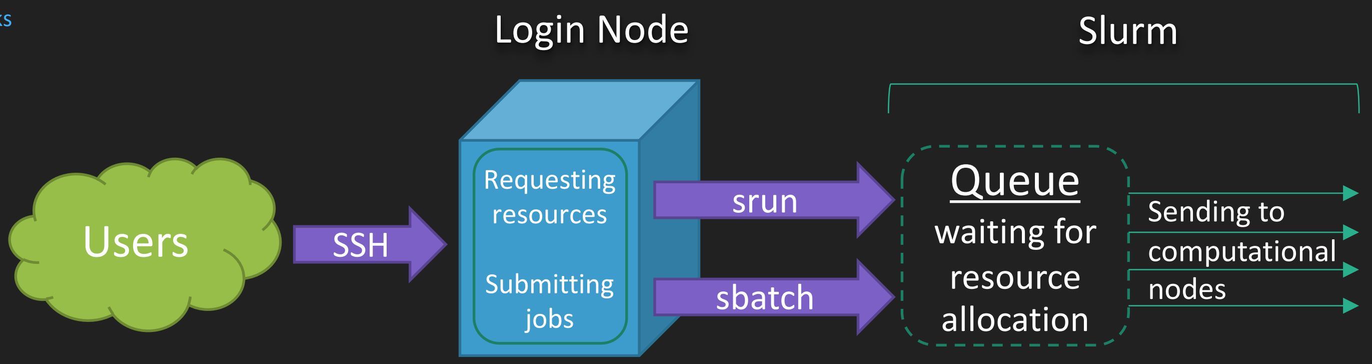
Job submission

All jobs must be run using `srun` or `sbatch` to prevent running on the Lewis login node.

srun: requesting resources to run jobs

```
srun <slurm-options> <software-name/path>
srun --pty /bin/bash requesting a pseudo terminal of bash shell to run jobs interactively
srun -p Interactive --pty /bin/bash requesting a p.t. of bash shell in Interactive Node      -p --partition
srun -p <partition-name> -n 4 --mem 16G --pty /bin/bash req. 4 tasks* and 16G memory      -n --ntasks
srun -p Gpu --gres gpu:1 -N 1 --ntasks-per-node 8 --pty /bin/bash req. 1 GPU and 1 node    -N --nodes
for running 8 tasks on Gpu partition
```

*Slurm by default allocates 1 CPU per tasks



sbatch: submitting jobs

Batch file is a shell script (#!/bin/bash) including Slurm options (#SBATCH) and computational tasks

```
sbatch <batch-file> submitting a batch file
```

After job completion we will receive outputs (slurm-jobid.out)

Slurm options

man srun - man sbatch		Environmental Variables
-p --partition <partition-name>	--pty <software-name/path>	$SLURM_JOB_ID
--mem <memory>	--gres <general-resources>	$SLURM_JOB_NAME
-n --ntasks <number of tasks>	-t --time <days-hours:minutes>	$SLURM_JOB_NODELIST
-N --nodes <number-of-nodes>	-A --account <account>	$SLURM_CPUS_ON_NODE
-c --cpus-per-task <number-of-cpus>	-L --licenses <license>	$SLURM_SUBMIT_HOST
-w --nodelist <list-of-node-names>	-J --job-name <jobname>	$SLURM_SUBMIT_DIR

Example

test.py

```
#!/usr/bin/python3

import os

os.system("""
echo hostname: $(hostname)
echo number of processors: $(nproc)
echo data: $(date)
echo job id: $SLURM_JOB_ID
echo submit dir: $SLURM_SUBMIT_DIR
""")

print("Hello world”)
```

jobpy.sh

```
#!/bin/bash

#SBATCH -p Interactive
#SBATCH -n 4
#SBATCH --mem 8G

python3 test.py
```

sbatch

```
sbatch jobpy.sh
```

srun

```
srun -p Interactive -n 4 --mem 8G --pty bash

python3 test.py
```

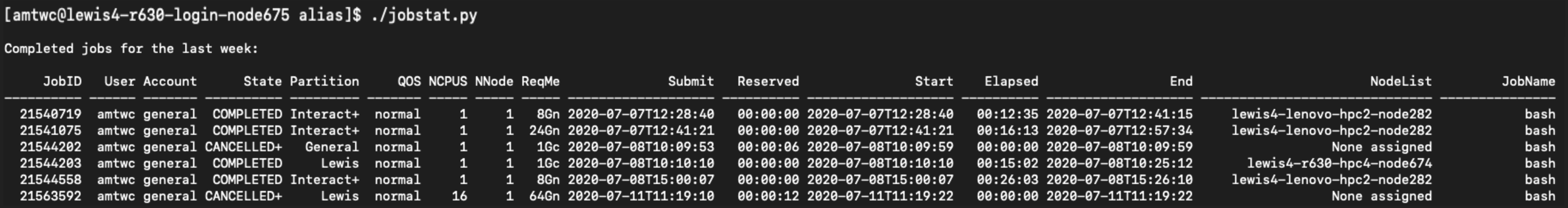
Output

```
hostname: lewis4-lenovo-hpc2-node282
number of processors: 4
data: Sun Jun 28 13:27:39 CDT 2020
job id: 21437062
submit dir: /home/user/training
Hello world
```

Monitor jobs

sacct - squeue - scancel

```
sacct -X show your jobs in the last 24 hours                                    -X --allocations
sacct -X -S <yyyy-mm-dd> show your jobs since a date                            -S --starttime
sacct -X -S <yyyy-mm-dd> -E <yyyy-mm-dd> -s <R/PD/F/CA/CG/CD> show              -s --state
running/pending/failed/cancelled/completing/completed jobs in a preiod of time
sacct -j <jobid> show info about the jobid
squeue -u <username> show a user jobs (R/PD/CD) in the queue                    -u --user
squeue -u <username> --start show estimation time to start pending jobs
scancel <jobid> cancel jobs
./jobstat.py <day/week/month/year> show info about running, pending and completed
jobs of a user within a time period (default is a week)
```



```
[amtwc@lewis4-r630-login-node675 alias]$ ./jobstat.py

Completed jobs for the last week:

     JobID   User Account      State Partition     QOS NCPUS NNode ReqMe              Submit   Reserved               Start    Elapsed                 End                       NodeList         JobName
---------- ------ ------- ---------- --------- ------- ----- ----- ----- ------------------- ---------- ------------------- ---------- ------------------- ------------------------------ ---------------
  21540719  amtwc general  COMPLETED Interact+  normal     1     1   8Gn 2020-07-07T12:28:40   00:00:00 2020-07-07T12:28:40   00:12:35 2020-07-07T12:41:15     lewis4-lenovo-hpc2-node282            bash
  21541075  amtwc general  COMPLETED Interact+  normal     1     1  24Gn 2020-07-07T12:41:21   00:00:00 2020-07-07T12:41:21   00:16:13 2020-07-07T12:57:34     lewis4-lenovo-hpc2-node282            bash
  21544202  amtwc general CANCELLED+   General  normal     1     1   1Gc 2020-07-08T10:09:53   00:00:06 2020-07-08T10:09:59   00:00:00 2020-07-08T10:09:59                  None assigned            bash
  21544203  amtwc general  COMPLETED     Lewis  normal     1     1   1Gc 2020-07-08T10:10:10   00:00:00 2020-07-08T10:10:10   00:15:02 2020-07-08T10:25:12       lewis4-r630-hpc4-node674            bash
  21544558  amtwc general  COMPLETED Interact+  normal     1     1   8Gn 2020-07-08T15:00:07   00:00:00 2020-07-08T15:00:07   00:26:03 2020-07-08T15:26:10     lewis4-lenovo-hpc2-node282            bash
  21563592  amtwc general CANCELLED+     Lewis  normal    16     1  64Gn 2020-07-11T11:19:10   00:00:12 2020-07-11T11:19:22   00:00:00 2020-07-11T11:19:22                  None assigned            bash
```

Monitor CPU and Memory

Compeleted jobs

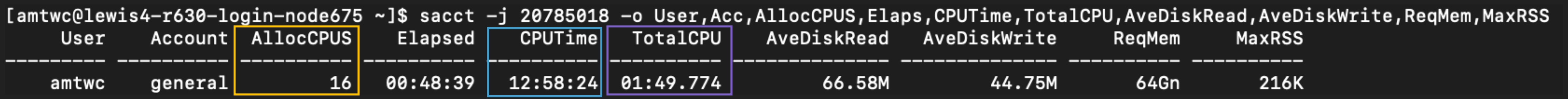
sacct - seff

```
sacct -j <jobid> -o User,Acc,AllocCPUS,Elaps,CPUTime,TotalCPU,AveDiskRead,AveDiskWrite,ReqMem,MaxRSS
```

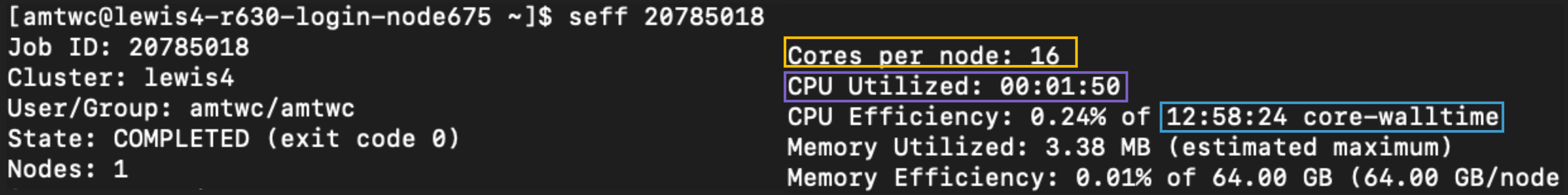
info about CPU and virtual memory for **compeleted** jobs

```
-j --jobs
-o --format
```

`seff <jobid>` show jobs CPU and memory efficiency



```
[amtwc@lewis4-r630-login-node675 ~]$ sacct -j 20785018 -o User,Acc,AllocCPUS,Elaps,CPUTime,TotalCPU,AveDiskRead,AveDiskWrite,ReqMem,MaxRSS
     User    Account  AllocCPUS    Elapsed    CPUTime   TotalCPU    AveDiskRead   AveDiskWrite     ReqMem     MaxRSS
--------- ---------- ---------- ---------- ---------- ---------- -------------- -------------- ---------- ----------
    amtwc    general         16   00:48:39   12:58:24  01:49.774         66.58M         44.75M       64Gn       216K
```



```
[amtwc@lewis4-r630-login-node675 ~]$ seff 20785018
Job ID: 20785018
Cluster: lewis4
User/Group: amtwc/amtwc
State: COMPLETED (exit code 0)
Nodes: 1
                                          Cores per node: 16
                                          CPU Utilized: 00:01:50
                                          CPU Efficiency: 0.24% of 12:58:24 core-walltime
                                          Memory Utilized: 3.38 MB (estimated maximum)
                                          Memory Efficiency: 0.01% of 64.00 GB (64.00 GB/node
```

CPU Efficiency = CPU Utilization / Core-walltime
Core-walltime = Core per node * Elapsed time

Running jobs:

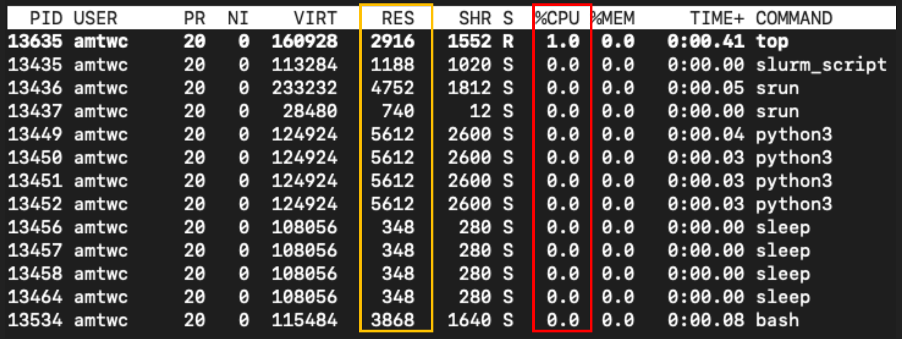
sstat - srun

`sstat <jobid> -o AveCPU,AveDiskRead,AveDiskWrite,MaxRSS` info about CPU and memory for **runing** jobs (srun only)

`srun --jobid <jobid> --pty /bin/bash` attach to a srun/sbatch session and run `top` command to see information about processes

Test

```
emacs -nw test.py
emacs -nw jobpy.sh
Ctrl+x+s to save and Ctrl+x+c to exit
sbatch jobpy.sh
sstat <jobid>
srun --jobid <jobid> --pty bash
top -u $USER press q to exit
```



```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
13635 amtwc     20   0  160928   2916   1552 R   1.0  0.0   0:00.41 top
13435 amtwc     20   0  113284   1188   1020 S   0.0  0.0   0:00.00 slurm_script
13436 amtwc     20   0  233232   4752   1812 S   0.0  0.0   0:00.05 srun
13437 amtwc     20   0   28480    740     12 S   0.0  0.0   0:00.00 srun
13449 amtwc     20   0  124924   5612   2600 S   0.0  0.0   0:00.04 python3
13450 amtwc     20   0  124924   5612   2600 S   0.0  0.0   0:00.03 python3
13451 amtwc     20   0  124924   5612   2600 S   0.0  0.0   0:00.03 python3
13452 amtwc     20   0  124924   5612   2600 S   0.0  0.0   0:00.03 python3
13456 amtwc     20   0  108056    348    280 S   0.0  0.0   0:00.00 sleep
13457 amtwc     20   0  108056    348    280 S   0.0  0.0   0:00.00 sleep
13458 amtwc     20   0  108056    348    280 S   0.0  0.0   0:00.00 sleep
13464 amtwc     20   0  108056    348    280 S   0.0  0.0   0:00.00 sleep
13534 amtwc     20   0  115484   3868   1640 S   0.0  0.0   0:00.08 bash
```

resident set size (RES) = memory KB

```
#SBATCH -n 4
#SBATCH --mem 8G

We are using 0% CPU and less
than 6MB memory
```

Modules

```
module avail/load/unload/list/show/purge
```

```
module avail available modules        module load loaded modules
module show show modules info         module unload unload loaded modules
module list list loaded modules       module purge unload all loaded modules
```

For example let's run R and MATLAB interactively:

R

```
srun -p Interactive --mem 4G --pty /bin/bash
module load R
module list
  1) R/R-3.3.3
R
```

MATLAB

```
srun -p Interactive --mem 4G -L matlab --pty /bin/bash
module load matlab
module list
  1) matlab/matlab-R2020a
matlab -nodisplay
```

Never load modules in the login node. It makes login node slow for all users. **Many modules don't work in the login node.** We can load modules in batch files, for example:

test.R

```
#!/usr/bin/R

for (i in 1:3) {
    cat("Hello world", i,"\n")
}
```

runr.sh

```
#!/bin/bash

#SBATCH -p Interactive
#SBATCH --mem 4G

module load R
Rscript test.R
```

sbatch

```
sbatch runr.sh
```

Output

```
Hello world 1
Hello world 2
Hello world 3
```

What is next:

Version control

- Git https://git-scm.com/book/en/v2

Job dependencies

- Slurm dependency option (--dependency) https://slurm.schedmd.com/sbatch.html
- Snakemake https://snakemake.readthedocs.io/

Virtual Environments

- Anaconda https://conda.io/en/latest/

Software Installation

- Spack https://spack.readthedocs.io/en/latest

Parallel programming

- Scientific languages C/Fortran
  - OpenMP
  - OpenACC for GPU parallelization
  - MPI for massive parallelization
- Python
  - NumPy
  - Numba
  - mpi4py

RCSS Documentation
http://docs.rnet.missouri.edu

XSEDE
https://www.xsede.org/for-users/training

Software Carpentry
https://software-carpentry.org/lessons/

HPC Carpentry
https://hpc-carpentry.github.io

Data Carpentry
https://datacarpentry.org/lessons/

Cornell Virtual Workshop
https://cvw.cac.cornell.edu/topics

Pittsburgh Supercomputing Center (PSC)
https://www.psc.edu/resources-for-users/training/

TACC Learning Portal
https://learn.tacc.utexas.edu/course/

Feedback and Questions
mudoitrcss@missouri.edu