# Slurm Basics

## Summer 2022

Timothy H. Kaiser, Ph.D
tkaiser2@nrel.gov

https://github.nrel.gov/tkaiser2/slides

https://github.com/timkphd/slides.git

# To cover...

- Slurm, what is it, does it do?
- Basic commands for:
  - Submitting
  - Getting status
  - Killing jobs
  - Node information
  - History of jobs
  - Seeing priority
  - Queues
  - Simple run scripts
  - Running interactively
  - Some important options
  - Links
  - More scripts as time allows

# How are HPC platforms laid out?

- Login node
- Compute nodes
- Service nodes
  - File system
  - Monitoring
  - Management
- All connected together via a network
- Eagle: https://www.nrel.gov/hpc/eagle-system-configuration.html

# Eagle

- Login nodes:
  - el1.hpc.nrel.gov
  - el2.hpc.nrel.gov
  - el3.hpc.nrel.gov
- Compute nodes:
  - 2589
  - r1xxxx - r10xxxx

# On small systems...

- Login nodes might also be the file system and management node
- My home machine (Runs the version of slurm soon to be installed on Eagle/Vermilion/Swift)


- pie
  - Login
  - Service (slurm)
  - File Server
- 3 Compute nodes
  - pi0-pi2
  - 4 cores / 4 GB
- Web Server
- Switch

# Slurm, what is it, does it do?

- Simple Linux Utility for Resource Management
- It manages compute resources
- It allows for fair sharing of resources
- It is the software that allows you, and others to run jobs on compute nodes

https://slurm.schedmd.com/quickstart.html

# Some definitions

- Node:
  - What is normally thought of as a computer that you could, in theory, login. Slurm manages the nodes and assigns jobs to them.
- Partition:
  - A collection of nodes often that have similar characteristics (memory, disk, core,gpus) or just grouped for management purposes.
- Every compute node is part of one or more partitions.

# Slurm, what is it, does it do?

- Typical usage:
  - You create a "batch" script that says:
    - What you want to run
    - Resources required
    - How long
  - You tell slurm to "run" your script
  - Slurm will run your job on compute nodes (after some waiting for resource to become available)

# Running Batch Scripts

- A batch script is submitted to **slurm**
  - Command to submit scripts
    - **sbatch myscript**
  - Options you add to the batch line overrule what you have in the script
    - **sbatch --nodes=1 myscript**
- The scheduler decides where and when to run your job
  - Each job is given a number
  - Jobs are also given a name so that you can track them in the system
  - Each job has a priority that goes up over time
  - Wait for nodes to become available
  - Start the job and let it run until it finishes or runs out of time

# A Simple Slurm Script hostname

```
#!/bin/bash

#SBATCH --job-name="atest"
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --time=00:02:00
#SBATCH -o stdout
#SBATCH -e stderr
#SBATCH --export=ALL
#SBATCH --account=hpcapps
#SBATCH --partition=debug
##SBATCH --mail-type=ALL
##SBATCH --mail-user=joeuser@nrel.gov

#-----------------------
cd /scratch/$USER
srun hostname
```

Scripts contain comments designated with a # that are interpreted by SLURM.

and normal shell commands

Lines with ## are ignored by slurm

Go to your scratch directory

srun - run a command in parallel

# A Simple Slurm Script for a MPI job

| | |
|---|---|
| #!/bin/bash | This is a bash script |
| #SBATCH --job-name="atest" | Give our job a name in the scheduler |
| #SBATCH --nodes=2 | We want 2 nodes |
| #SBATCH --ntasks-per-node=8 | We expect to run 8 tasks/node |
| #SBATCH --time=00:02:00 | We want the node for 2 minutes |
| #SBATCH --output=stdout | Output will go to a file “stdout” |
| #SBATCH --error=stderr | Errors will go to a file “stderr” |
| #SBATCH --export=ALL | Pass current environment to nodes |
| ##SBATCH --mail-type=ALL | Send email on abort,begin,end |
| ##SBATCH --mail-user=auser@nrel.gov | Address for email |
| #SBATCH --account=hpcapps | Your account (not hpcapps) |
| #SBATCH –partition=debug | The scheduler partition |
| #---------------------- | Just a normal “comment” |
| cd /scratch/$USER | Go to this directory first |
| srun hostname | Run hostname on 8 cores / node (2 nodes) |

# Submitting and Output

```
[tkaiser2@el1 system]$ sbatch myscript
...
...
[tkaiser2@el1 system]$ cat stdout
r102u34
r102u34
r102u34
r102u34
r102u34
r102u34
r102u34
r102u34
r102u34
r102u35
r102u35
r102u35
r102u35
r102u35
r102u35
r102u35
r102u35
[tkaiser2@el1 system]$
```

# Time format

```
#SBATCH --time=dd-hh:mm:ss

                Days (followed by -)
           Hours (followed by :)
      Minutes (followed by :)
                          Seconds
All are optional except Seconds

These are "legal"
--time=48:00:00
--time=120:00
--time=47:59:59
--time=2-00:00:00
--time=10:00
```

# Mail

--mail-type=

- NONE
- BEGIN
- END
- FAIL
- ALL
- TIME_LIMIT_90

--mail-user=**A_VALID_EMAIL_ ADDRESS_PLEASE**

# Standard Output/Error

- If you don't specify
  - #SBATCH -o stdout
  - #SBATCH -e stderr
- Output goes to:
  - slurm-xxx.out/err where xxx is the job number.
- You can create unique files for each job by including the following in your "file name":

```
%j      jobid of the running job.
%u      User name.
%x      Job name.
```

```
#SBATCH -o X_%j_%u_%x.out  gives us  X_9366638_tkaiser2_atest.out
```

# A Script to build/run MPI job

```
#SBATCH --job-name="atest"
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --time=00:02:00
#SBATCH -o /scratch/%u/%j.out
#SBATCH -o /scratch/%u/%j.err
#SBATCH --partition=debug
#SBATCH --account=hpcapps
##SBATCH --mail-type=ALL
##SBATCH --mail-user=joeuser@nrel.gov
```

Scripts contain comments designated with a # that are interpreted by SLURM and normal shell commands

```
#---------------------
cd /scratch/$USER
module load intel-mpi
curl https://raw.githubusercontent.com/timkphd/examples/master/slurm/phostone.c -o phostone.c
mpicc -fopenmp phostone.c -o  phostone
srun ./phostone -F
echo "Second run:"
srun --nodes=1 --ntasks=2 --ntasks-per-node=2 ./phostone -F > phostone.out
```

1. We go to this directory
2. Load MPI module
3. Download (Glorified hello world)
4. Build
5. Run this MPI program on 16 cores

"Override" the tasks/node and # tasks

Pipe output to a file

# Output

```
[tkaiser2@el2 slurm]$ cat phostone.out
MPI VERSION Intel(R) MPI Library 2019 Update 7 for Linux* OS

task    thread             node name  first task    # on node  core
0008      0000               r102u35        0008         0000  0011
0000      0000               r102u34        0000         0000  0015
0001      0000               r102u34        0000         0001  0000
0002      0000               r102u34        0000         0002  0017
0003      0000               r102u34        0000         0003  0003
0004      0000               r102u34        0000         0004  0013
0005      0000               r102u34        0000         0005  0014
0006      0000               r102u34        0000         0006  0016
0007      0000               r102u34        0000         0007  0012
0009      0000               r102u35        0008         0001  0003
0010      0000               r102u35        0008         0002  0004
0011      0000               r102u35        0008         0003  0000
0012      0000               r102u35        0008         0004  0005
0013      0000               r102u35        0008         0005  0015
0014      0000               r102u35        0008         0006  0001
0015      0000               r102u35        0008         0007  0002
[tkaiser2@el2 slurm]$
```

# More about srun

- sbatch will get you a collection of nodes
- srun will actually launch your program in parallel
- In most cases you will use srun instead of mpirun/ mpiexec
- How many instances:
  - Defaults to what is given in your script header

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
```

  - Can also:
    - `srun --tasks-per-node=4 --ntasks=8  myprogrm`
    - `Many options See: man srun`

# Slurm defines Variables:

```
SLURM_MPI_TYPE=pmi2
SLURM_STEP_ID=0
SLURM_NODEID=0
SLURM_TASK_PID=29272
SLURM_PRIO_PROCESS=0
SLURM_CPU_BIND_VERBOSE=quiet
SLURM_SUBMIT_DIR=/scratch/tkaiser2
SLURM_CPUS_PER_TASK=4
SLURM_STEPID=0
SLURM_SRUN_COMM_HOST=192.168.1.1
SLURM_PROCID=0
SLURM_JOB_GID=131364
SLURM_CPU_BIND=....
SLURM_ACCOUNT=hpcapps
SLURMD_NODENAME=c1-28
SLURM_TASKS_PER_NODE=32
SLURM_NNODES=1
SLURM_LAUNCH_NODE_IPADDR=192.168.1.1
SLURM_STEP_TASKS_PER_NODE=32
SLURM_JOB_NODELIST=c1-28
SLURM_CLUSTER_NAME=swift
SLURM_NODELIST=c1-28
SLURM_NTASKS=32
SLURM_UMASK=0002
SLURM_JOB_CPUS_PER_NODE=128
SLURM_TOPOLOGY_ADDR=c1-28
```

```
SLURM_WORKING_CLUSTER=swift:192.168.1.4:6817:9472:101
SLURM_STEP_NODELIST=c1-28
SLURM_JOB_NAME=atest
SLURM_SRUN_COMM_PORT=34097
SLURM_JOBID=143033
SLURM_CONF=/etc/slurm.conf
SLURM_JOB_QOS=normal
SLURM_TOPOLOGY_ADDR_PATTERN=node
SLURM_CPUS_ON_NODE=128
SLURM_JOB_NUM_NODES=1
SLURM_JOB_UID=131364
SLURM_JOB_PARTITION=debug
SLURM_PTY_WIN_ROW=60
SLURM_CPU_BIND_LIST=....
SLURM_JOB_USER=tkaiser2
SLURM_PTY_WIN_COL=146
SLURM_NPROCS=32
SLURM_SUBMIT_HOST=swift-login-1.swift.hpc.nrel.gov
SLURM_JOB_ACCOUNT=hpcapps
SLURM_STEP_LAUNCHER_PORT=34097
SLURM_PTY_PORT=37177
SLURM_JOB_ID=143033
SLURM_CPU_BIND_TYPE=mask_cpu:
SLURM_STEP_NUM_TASKS=32
SLURM_STEP_NUM_NODES=1
SLURM_LOCALID=0
```

```
mkdir -p $SLURM_JOB_NAME/$SLURM_JOB_ID
cd       $SLURM_JOB_NAME/$SLURM_JOB_ID
cat $0 > script.$SLURM_JOB_ID
```

# Getting Status and Information

- squeue
- sinfo
- scontrol show node
- sacctmgr

# squeue

- Shows what is running or waiting to run
- By default it shows everyone's jobs
- Just show yours:
  - squeue -u $USER
- Squeue has many potential output fields
- The --format option allows to to select fields

# squeue: default output

```
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
461291  debug    benchmar  slarsen PD       0:00      1 (ReqNodeNotAvail)
581892  standard  install  joetrip PD       0:00      1 (Priority)
581891  parallel  install  joetrip PD       0:00      1 (Priority)
489530  parallel  install  joetrip PD       0:00      1 (Priority)
489529  parallel  install  joetrip PD       0:00      1 (Priority)
488899  parallel  install  joetrip PD       0:00      1 (Priority)
594990  parallel  install  joetrip PD       0:00      1 (Priority)
594989  parallel  install  joetrip PD       0:00      1 (Priority)
594988  parallel  install  joetrip PD       0:00      1 (Priority)
582732      test   jaguar    toads  R    5:51:25      2 c9-[45-47]
582728      test   jaguar    toads  R    5:55:47      1 c4-49
582726      test   jaguar    toads  R    5:57:24      1 c4-49
582712      test   jaguar    toads  R    6:08:34      1 c8-8
581890      test  install  joetrip PD       0:00      1 (Resources)
```

# squeue with a format

I have an alias "sq" for squeue with a format:

```
alias sq='squeue -u $USER --format="\"%10A%15I%15L%6D%20S%15P%15r%20V%N\""'
```

```
[tkaiser2@el1 runior]$ sq
JOBID     TIME_LIMIT TIME_LEFT  NODES START_TIME          PARTITION  REASON           SUBMIT_TIME          NODELIST
9382137   1:00:00    1:00:00    2     N/A                 debug      QOSMaxJobsPerUs  2022-06-03T10:42:54
9382131   1:00:00    1:00:00    2     N/A                 debug      QOSMaxJobsPerUs  2022-06-03T10:42:37
9382129   1:00:00    59:42      2     2022-06-03T10:42:39 debug      None             2022-06-03T10:42:22 r102u[34-35]
9382130   1:00:00    59:42      2     2022-06-03T10:42:39 debug      None             2022-06-03T10:42:29 r104u33,r105u33
[tkaiser2@el1 runior]$
```

squeue -i 10 -u $USER

Run squeue every 10 seconds

# scancel

- Cancel (kill/stop) your jobs
  - scancel 9382130
    - Kills a particular job(s)
  - scancel -u $USER
    - Kills all of your jobs

# sinfo

## Show information about partitions and nodes

```
[tkaiser2@vs-login-1 ENV]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
sm           up 1-00:00:00      1 drain  vs-sm-0016
sm           up 1-00:00:00      4  down  vs-sm-[0001,0024,0028,0032]
sm           up 1-00:00:00      3  drain vs-sm-[0002-0003,0026]
sm           up 1-00:00:00     24   idle vs-sm-[0004-0015,0017-0023,0025,0027,0029-0031]
gpu          up 1-00:00:00      2  drain vs-gpu-[0004,0006]
gpu          up 1-00:00:00      4  alloc vs-gpu-[0001-0003,0005]
lg           up 1-00:00:00      1  alloc vs-lg-0001
lg           up 1-00:00:00      1  down  vs-lg-0004
lg           up 1-00:00:00      2  drain vs-lg-[0010,0017]
lg           up 1-00:00:00     10  alloc vs-lg-[0002-0003,0006-0008,0012-0014,0016,0018]
std          up 1-00:00:00      2  drain vs-std-[0003,0026]
std          up 1-00:00:00      1  drain vs-std-0014
std          up 1-00:00:00     47  alloc vs-std-[0001-0002,0004-0006,0008,0010-0011,0013]
t            up    4:00:00      2  drain vs-t-[0004,0011]
t            up    4:00:00      4  drain vs-t-[0006,0009,0012-0013]
t            up    4:00:00      9   idle vs-t-[0001-0003,0005,0007-0008,0010,0014-0015]
[tkaiser2@vs-login-1 ENV]$
```

# sinfo

## Show information about partitions and nodes

```
[tkaiser2@vs-login-1 ENV]$ sinfo -p lg
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
lg           up 1-00:00:00      1 alloc* vs-lg-0001
lg           up 1-00:00:00      1  down* vs-lg-0004
lg           up 1-00:00:00      4   comp vs-lg-[0005,0009,0011,0015]
lg           up 1-00:00:00      2  drain vs-lg-[0010,0017]
lg           up 1-00:00:00     10  alloc vs-lg-[0002-0003,0006-0008,0012-0014,0016,0018]
[tkaiser2@vs-login-1 ENV]$
[tkaiser2@vs-login-1 ENV]$
[tkaiser2@vs-login-1 ENV]$ sinfo -N -n vs-t-0001
NODELIST   NODES PARTITION STATE
vs-t-0001      1         t idle
[tkaiser2@vs-login-1 ENV]$
[tkaiser2@vs-login-1 ENV]$
[tkaiser2@vs-login-1 ENV]$ sinfo -N  | grep idle    or    (sinfo -N -t idle)
vs-sm-0004       1        sm idle
vs-sm-0005       1        sm idle
vs-sm-0027       1        sm idle
vs-sm-0030       1        sm idle
vs-sm-0031       1        sm idle
vs-t-0001        1         t idle
vs-t-0014        1         t idle
vs-t-0015        1         t idle
[tkaiser2@vs-login-1 ENV]$
```

# scontrol

- Many functions, only two you will most likely ever use:
  - Hold/release a job
    - `scontrol uhold  9382268`
    - `scontrol release  9382268`
  - Get detailed information about a node(s)/job(s)
    - `scontrol show node`
    - `scontrol show job ######`

# scontrol show node

```
[tkaiser2@el1 runior]$ scontrol show node r103u01
NodeName=r103u01 Arch=x86_64 CoresPerSocket=18
   CPUAlloc=36 CPUTot=36 CPULoad=26.87
   AvailableFeatures=hy
   ActiveFeatures=hy
   Gres=gpu:v100:2
   NodeAddr=r103u01 NodeHostName=r103u01 Version=21.08.5
   OS=Linux 3.10.0-1062.9.1.el7.x86_64 #1 SMP Fri Dec 6 15:49:49 UTC 2019
   RealMemory=751616 AllocMem=0 FreeMem=718868 Sockets=2 Boards=1
   State=ALLOCATED ThreadsPerCore=1 TmpDisk=24000000
   Weight=1 Owner=N/A MCS_label=N/A
   Partitions=gpu,gpu-stdby,gpul,gpul-stdby,bigscratch
   BootTime=2022-04-07T08:39:15 SlurmdStartTime=2022-04-07T08:44:41
   LastBusyTime=2022-06-02T07:55:17
   CfgTRES=cpu=36,mem=734G,billing=36,gres/gpu=2
   AllocTRES=cpu=36,gres/gpu=2
   CapWatts=n/a
   CurrentWatts=0 AveWatts=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

# scontrol show job

```
[tkaiser2@r4i7n35 tkaiser2]$ scontrol show job 9443283
JobId=9443283 JobName=interactive
   UserId=tkaiser2(131364) GroupId=tkaiser2(131364) MCS_label=N/A
   Priority=248099184 Nice=0 Account=hpcapps QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
   RunTime=00:00:17 TimeLimit=01:00:00 TimeMin=N/A
   SubmitTime=2022-06-08T12:28:31 EligibleTime=2022-06-08T12:28:31
   AccrueTime=2022-06-08T12:28:31
   StartTime=2022-06-08T12:28:36 EndTime=2022-06-08T13:28:36 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2022-06-08T12:28:36 Scheduler=Main
   Partition=debug AllocNode:Sid=el3:27289
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=r4i7n35
   BatchHost=r4i7n35
   NumNodes=1 NumCPUs=36 NumTasks=4 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=36,node=1,billing=36
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=[hy|ehy] DelayBoot=00:00:00
   OverSubscribe=NO Contiguous=0 Licenses=(null) Network=(null)
   Command=(null)
   WorkDir=/lustre/eaglefs/scratch/tkaiser2
   Switches=1@2-00:00:00
   Power=
[tkaiser2@r4i7n35 tkaiser2]$
```

# sacct

```
[tkaiser2@vs-login-1 ENV]$ sacct -S 2022-01-04 -E 2022-01-06 -u $USER
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
50005503     rfm_Hello+        gpu    hpcapps         60  COMPLETED      0:0
50005503.ba+      batch              hpcapps         30  COMPLETED      0:0
50005503.0   HelloTest+              hpcapps         60  COMPLETED      0:0
50005504     rfm_Hello+        gpu    hpcapps         60 CANCELLED+      0:0
50005504.ba+      batch              hpcapps         30  CANCELLED     0:15
50005504.0   HelloTest+              hpcapps         60  CANCELLED     0:15
50005505     rfm_Hello+         lg    hpcapps        120  COMPLETED      0:0
50005505.ba+      batch              hpcapps         60  COMPLETED      0:0
50005505.0   HelloTest+              hpcapps        120  COMPLETED      0:0
50005506     rfm_Hello+         lg    hpcapps        120  COMPLETED      0:0
50005506.ba+      batch              hpcapps         60  COMPLETED      0:0
50005506.0   HelloTest+              hpcapps        120  COMPLETED      0:0
50005507     rfm_Hello+         lg    hpcapps        120  COMPLETED      0:0
50005507.ba+      batch              hpcapps         60  COMPLETED      0:0
50005507.0   HelloTest+              hpcapps        120  COMPLETED      0:0
50005508     rfm_Hello+         lg    hpcapps        120  COMPLETED      0:0
50005508.ba+      batch              hpcapps         60  COMPLETED      0:0
50005508.0   HelloTest+              hpcapps        120  COMPLETED      0:0
[tkaiser2@vs-login-1 ENV]$
```

Show history of my jobs between January 4 and January 6

# sacct

- Like most commands sacct has a format option
- Things important to me:
  - JobID
  - Start
  - Nodelist
  - Starting directory
- I have a bash function (recent) that
  - Takes days range (not date)
    - Calls sacct with the above requested
    - Does some filtering

# recent - sacct

```
[tkaiser2@vs-login-1 ENV]$ recent 4 1
50022417 2022-05-31T12:18:06  vs-sm-0025   /home/tkaiser2
50022418 2022-05-31T12:22:11  vs-sm-0025   /home/tkaiser2
50022433 2022-06-01T14:30:09  None assigned    /projects/hpcapps/tkaiser2/runior
50022434 2022-06-01T14:30:38  None assigned    /projects/hpcapps/tkaiser2/runior
50022435 2022-06-01T14:31:26  vs-sm-[0017-0020]/projects/hpcapps/tkaiser2/runior
50022436 2022-06-01T14:33:51  vs-sm-[0017-0020]/projects/hpcapps/tkaiser2/runior
50022437 2022-06-01T14:37:40  vs-sm-[0017-0020]/projects/hpcapps/tkaiser2/runior
50022438 2022-06-01T14:39:13  vs-sm-[0017-0020]/projects/hpcapps/tkaiser2/runior
50022439 2022-06-01T14:42:13  vs-sm-[0029-0030]/projects/hpcapps/tkaiser2/runior
50022440 2022-06-01T14:44:31  vs-sm-[0030-0031]/projects/hpcapps/tkaiser2/runior
50022441 2022-06-01T14:46:29  vs-sm-[0030-0031]/projects/hpcapps/tkaiser2/runior
50022442 2022-06-01T14:50:40  vs-sm-[0030-0031]/projects/hpcapps/tkaiser2/runior
50022443 2022-06-01T14:54:33  vs-sm-[0030-0031]/projects/hpcapps/tkaiser2/runior
50022444 2022-06-01T14:58:43  vs-sm-[0030-0031]/projects/hpcapps/tkaiser2/runior
50022445 2022-06-01T15:06:47  vs-sm-[0030-0031]/projects/hpcapps/tkaiser2/runior
50022446 2022-06-02T14:47:12  None assigned    /projects/hpcapps/tkaiser2/runior
50022447 2022-06-01T16:29:33  vs-sm-[0029-0030]/projects/hpcapps/tkaiser2/runior
50022448 2022-06-01T16:31:14  vs-sm-[0017-0018]/projects/hpcapps/tkaiser2/runior
50022449 2022-06-01T16:33:26  vs-sm-0025   /projects/hpcapps/tkaiser2/runior
[tkaiser2@vs-login-1 ENV]$
```

# Saccntmgr

Another command with many functions but you will most likely only use it to find out what accounts you have on a machine

```
[thk2@el1 ENV]$ sacctmgr show associations user=$USER format=account%15
        Account
---------------
        hpcapps
    continental
       mobility
           msoc
            wks
       naermpcm
[thk2@el1 ENV]$
```

```
alias accounts='sacctmgr show associations user=$USER format=account%15'
```

# sprio

## List priorities of all jobs

```
[tkaiser2@el1 runior]$ sprio | sort   -nk3,3 | head
             JOBID PARTITION    PRIORITY       SITE        AGE  FAIRSHARE    JOBSIZE  PARTITION        QOS
           9382310 short        10548997          0      28627     764730    5931990    3823650          0
           9382310 weto         10548997          0      28627     764730    5931990    3823650          0
           9382294 standard     12768014          0      52474    6773323    2118568    3823650          0
           9382290 standard     12779495          0      63955    6773323    2118568    3823650          0
           9382116 long         18474450          0     217989     873977   13558835    3823650          0
           9382116 weto         18474450          0     217989     873977   13558835    3823650          0
           9382077 long-stdb    45164944          0     288873   40967679      84743    3823650          0
           9382076 long-stdb    45165121          0     289050   40967679      84743    3823650          0
           9382075 long-stdb    45165247          0     289176   40967679      84743    3823650          0


[tkaiser2@el1 runior]$ sprio | sort  -r -nk3,3 | head
             JOBID PARTITION    PRIORITY       SITE        AGE  FAIRSHARE    JOBSIZE  PARTITION        QOS
           9274677 standard-   429027472          0   30589200  318346174   76268449    3823650          0
           9274676 standard-   429027472          0   30589200  318346174   76268449    3823650          0
           9274675 standard-   429027472          0   30589200  318346174   76268449    3823650          0
           9274674 standard-   429027472          0   30589200  318346174   76268449    3823650          0
           9274673 standard-   429027472          0   30589200  318346174   76268449    3823650          0
           9274672 standard-   429027472          0   30589200  318346174   76268449    3823650          0
           9274671 standard-   429027472          0   30589200  318346174   76268449    3823650          0
           9274670 standard-   429027472          0   30589200  318346174   76268449    3823650          0
           9274669 standard-   429027472          0   30589200  318346174   76268449    3823650          0
           9274668 standard-   429027472          0   30589200  318346174   76268449    3823650          0
```

priority - NREL specific command to show the priority of your jobs

# Running Interactively

- Sometimes you want to grab a node or nodes and run commands
- Almost like running on login node except you are actually logged in to a compute node and you have access to all of the nodes you have allocated
- After you get started you can run simple commands or run in parallel with srun
- The command to do this is salloc
- Note: The way this works has recently changed

# Salloc

- salloc
  - Used to allocate resources for a job in an interactive mode. Typically this is used to allocate resources and spawn a shell. The shell is then used to execute srun commands to launch parallel tasks.

```
salloc --nodes=2 --ntasks=72 --time=01:00:00 --partition=debug --account=hpcapps
```

# Grab nodes and run with salloc

```
[thk@el1 ~]$ salloc --nodes=2 --ntasks=72 --time=01:00:00 --partition=debug --account=hpcapps
salloc: Pending job allocation 9382710
salloc: job 9382710 queued and waiting for resources
salloc: job 9382710 has been allocated resources
salloc: Granted job allocation 9382710
salloc: Waiting for resource configuration
salloc: Nodes r102u[34-35] are ready for job
[thk@r102u34 ~]$
[thk@r102u34 ~]$
[thk@r102u34 ~]$ printenv  SLURM_NODELIST
r102u[34-35]
[thk@r102u34 ~]$
[thk@r102u34 ~]$
[thk@r102u34 ~]$ srun -n 4 --distribution=block -l  hostname
1: r102u34
0: r102u34
2: r102u35
3: r102u35
[thk@r102u34 ~]$
```

# More sbatch /salloc options

- Any option specified with #sbatch can be overwritten with an option on the command line
  - `sbatch --time=1-00:00:00 my script`
- `You can ask for nodes with a specific amount of memory, scratch disk, gpus`
- `These options are specific to Eagle but other machines will have similar options`
  - `--mem=750000 #750 Gbytes (big memory nodes)`
  - `--tmp=1500000 1.5 Tbytes of scratch`

# Examples of --mem and --tmp

```
[tkaiser2@el3 scratch]$ salloc --nodes=1 --time=01:00:00 --partition=debug --account=hpcapps
...
salloc: Nodes r2i7n35 are ready for job
[tkaiser2@r2i7n35 scratch]$ free
              total        used        free      shared  buff/cache   available
Mem:       97,259,020     2325684    87243676     7541448     7689660    87022456
Swap:              0           0           0
[tkaiser2@r2i7n35 scratch]$ df | grep scratch
/dev/sda1                  976,282,876          33088    976249788   1% /tmp/scratch
[tkaiser2@r2i7n35 scratch]$
[tkaiser2@r2i7n35 scratch]$
[tkaiser2@r2i7n35 scratch]$ exit
exit

[tkaiser2@el3 scratch]$ salloc --nodes=1 --time=01:00:00 --partition=debug --account=hpcapps --mem=750000
...
salloc: Nodes r102u34 are ready for job
[tkaiser2@r102u34 scratch]$ free
              total        used        free      shared  buff/cache   available
Mem:      791,956,620     6564420   780424112     4830904     4968088   778842520
Swap:              0           0           0
[tkaiser2@r102u34 scratch]$ exit
exit


[tkaiser2@el3 scratch]$ salloc --nodes=1 --time=01:00:00 --partition=debug --account=hpcapps --tmp=1500000
salloc: Pending job allocation 9443739
...
[tkaiser2@r102u34 scratch]$ df | grep scratch
/dev/sda                  1,561,994,912          34880    1561960032   1% /tmp/scratch
[tkaiser2@r102u34 scratch]$
```

https://www.nrel.gov/hpc/eagle-job-partitions-scheduling.html

# Ask for GPU nodes

## --gres=gpu:2

```
salloc: Relinquishing job allocation 9443751
[tkaiser2@el3 scratch]$ salloc --nodes=1 --time=01:00:00 --partition=debug --account=hpcapps
...
salloc: Nodes r3i7n35 are ready for job
[tkaiser2@r3i7n35 scratch]$ echo "number of GPUs on node:" `/nopt/nrel/apps/cuda/gpucount`
number of GPUs on node: 0
[tkaiser2@r3i7n35 scratch]$ exit


[tkaiser2@el3 scratch]$ salloc --nodes=1 --time=01:00:00 --partition=debug --account=hpcapps --gres=gpu:2
...
salloc: Nodes r103u21 are ready for job
[tkaiser2@r103u21 scratch]$ echo "number of GPUs on node:" `/nopt/nrel/apps/cuda/gpucount`
number of GPUs on node: 2
[tkaiser2@r103u21 scratch]$
```

If you land on a node that has GPUs but don't specify the --gres command you will not see the GPUs.

As of 06/09/22 on Vermilion gres in not currently required/supported

# Partitions

## Eagle

```
sinfo | awk '{print $1}' | sort -u
amo
bigmem
bigscratch
csc
debug
gpu
gpul
long
short
standard
weto
weto-stdby
```

Also *-stdby

https://www.nrel.gov/hpc/eagle-job-partitions-scheduling.html

- Some machines require you to request a partition
- Eagle will try to put you in the proper partition based on requested resources

# Some Links

- https://nrel.github.io/HPC
  - Work in progress NREL HPC documentation
- https://www.nrel.gov/hpc/
  - Main NREL HPC documentation
- https://github.com/NREL/HPC/tree/master/slurm
  - My slurm script examples
- https://www.nrel.gov/hpc/eagle-job-partitions-scheduling.html
  - Eagle partition/scheduling Information
- https://www.nrel.gov/hpc/eagle-system-configuration.html
  - Eagle configuration Information
- https://slurm.schedmd.com/quickstart.html
  - Slurm Docs

# man pages

- All of the commands from today have man pages
  - man sbatch
  - man squeue
  - man srun
  - man sinfo
  - man scancel
  - man sacctmgr

https://slurm.schedmd.com/quickstart.html

# Functions

## recent - shows recent jobs

Put these in your .bashrc file

```
recent ( ) {
#    echo $#;
    if (($# == 2)); then
        sacct -S `date --date="$1 days ago" +"%Y-%m-%d"` \
              -E `date --date="$2 days ago" +"%Y-%m-%d"` \
              -u $USER \
              --format=JobID,start,nodelist%25,WorkDir%125 \
              | sed "s/ \+ /\t/g" | egrep --color=auto -v "\.[0-9]|\.ba" \
              | egrep -v "\+ "
    fi;
    if (($# == 1)); then
        sacct -S `date --date="$1 days ago" +"%Y-%m-%d"` \
              -u $USER \
              --format=JobID,start,nodelist%25,WorkDir%125 \
              | sed "s/ \+ /\t/g" | egrep --color=auto -v "\.[0-9]|\.ba" \
              | egrep -v "\+ "
    fi;
    if (($# == 0)); then
        sacct -S `date --date="14 days ago" +"%Y-%m-%d"` \
              -u $USER \
              --format=JobID,start,nodelist%25,WorkDir%125 \
              | sed "s/ \+ /\t/g" | egrep --color=auto -v "\.[0-9]|\.ba" \
              | egrep -v "\+ "
    fi
}
```

```
alias accounts='sacctmgr show associations user=$USER format=account%15'
```

```
alias sq='squeue -u $USER --format='\''%10A%15l%15L%6D%20S%15P%15r%20V%N'\'''
```

# Functions

Put this in your .bashrc file Eagle

## myalloc - does a salloc with reasonable settings

```
function myalloc() {
    ACC=hpcapps
    case $# in
    "0" )
      salloc --nodes=1 --time=01:00:00 --account=$ACC --partition=debug
      ;;
    "1" )
     if [ "$1" = "gpu" ] ; then
         salloc --nodes=1 --time=1:00:00 --account=$ACC --partition=debug --gres=gpu:2
      else
         if [ "$1" -lt "3" ] ; then
           salloc --nodes=$1 --time=1:00:00 --account=$ACC --partition=debug
         else
           salloc --nodes=$1 --time=4:00:00 --account=$ACC --partition=short
         fi
      fi
      ;;
    "2" )
     salloc --nodes=$1 --time=$2 --account=$ACC --partition=short
     ;;
    * )
     echo "myalloc [nodes] [time]"
     echo "   OR"
     echo "salloc --nodes=N --time=hh:mm:ss --account=$ACC --partition=mypartition"
    esac
}
```

# myalloc

On Eagle

```
myalloc                  1 node      1 hr
myalloc 2                2 nodes     1 hr
myalloc 3                3 nodes     4 hrs
myalloc 2 10:00:00 2 nodes    10 hrs
myalloc gpu              1 gpu node 1 hr
```

# Functions

Put this in your .bashrc file Vermilion

```
function myalloc() {
    ACC=hpcapps
    case $# in
     "0" )
       salloc --nodes=1 --time=4:00:00 --account=$ACC --partition=t
       ;;
     "1" )
      if [ "$1" = "gpu" ] ; then
           salloc --nodes=1 --time=4:00:00 --account=$ACC --partition=gpu
       else
           salloc --nodes=$1 --time=4:00:00 --account=$ACC --partition=sm
      fi
       ;;
     "2" )
      salloc --nodes=$1 --time=$2 --account=$ACC --partition=lg
      ;;
     * )
      echo "myalloc [nodes] [time]"
      echo "   OR"
      echo "salloc --nodes=N --time=hh:mm:ss --account=$ACC --partition=mypartition"
     esac
}
```

# Functions

Put this in your .bashrc file Swift

```
function myalloc() {
     ACC=hpcapps
     case $# in
    "0" )
      salloc --nodes=1 --time=04:00:00 --account=$ACC --partition=debug
      ;;
    "1" )
      if [ "$1" -lt "3" ] ; then
        salloc --nodes=$1 --time=4:00:00 --account=$ACC --partition=debug
      else
        salloc --nodes=$1 --time=2-00:00:00 --account=$ACC --partition=parallel
      fi
      ;;
    "2" )
     salloc --nodes=$1 --time=$2 --account=$ACC --partition=parallel
     ;;
    * )
     echo "myalloc [nodes] [time]"
     echo "   OR"
     echo "salloc --nodes=N --time=hh:mm:ss --account=$ACC --partition=mypartition"
    esac
}
```