Procmgr To Slurm

- Configless setup
- Slurm Feature (Constraint)
- Heterogeneous jobs and constraint
- Slow start-up time for heterogeneous jobs
- Running job steps in parallel
- Update 20240312

TODO:

1. job management: support restart, scancel with signal? support restart an individual process, **show completed jobs**.
2. check for existing user/platform. Note: --dependency flag can be used to check for unique jobname but jobs are still queued. Better to exit if the same jobname is found. See the unique format here: ls ~tmoopr/.psdaq/. Show details of conflicting jobs. Job comment (right now is unique) is used to check for existing jobs for ALL users.
3. **procstat like unbuffered output style. Maybe https://portal.supercomputing.wales/index.php/index/slurm/interactive-use-job-arrays/x11-gui-forwarding/. Note: you can x11-forward using for example,**

   slurm.conf

   ```
   salloc -n1 --x11 srun -n1 --x11 xterm -hold -e "python test_run.py"
   ```

   This --x11 in srun also works with sbatch when $DISPLAY is exported correctly. See lcls2/psdaq/psdaq/slurm for how it's implemented.
4. check if slurm avoids weka cores. It looks like slurm tries to avoid weka cores automatically.
5. **Multi-threading process. This was possible in the past but possibly with recent changes, it's not working.**
6. How to identify what a resource is for drp (Bandwidth/ Memory/ physical cores for each process).
7. Documentation/How to
8. **Testing goals: TMO, RIX, other long-live processes, high rate 71kHz**
9. slurm.conf: configless setup still not complete, set MaxTime=UNLIMITED, no limit on memory, no hyper-threading, make sure slurm version is consistent
10. **Automatically add back power-cycled nodes.**
11. **Check why collect_host=drp-srcf-mon001 doesn't work.**

    ```
    2024-05-28 15:23:33,483 None-DaqControl[38248]: <E> getPlatform() Exception: Operation cannot be
    accomplished in current state
    ```

12. New 'cores' field in cnf file:

    ```
    { host: 'drp-srcf-cmp004', cores: '10' id:'tmo_fim1_0',   flags:'spu',  env:epics_env, cmd:drp_cmd0+' -l
    0x10 -D wave8 -k epics_prefix=MR3K4:FIM:W8:01'},
    ```

    For processes that start with 'drp ', add -W with #cores - 4 (pgpreader, collector, ebreceiver, & filewriter).

For ami-node_*

```
{ host:worker_node, id:f'ami-node_{N}', flags:'s', env:epics_env,
                    cmd:f'ami-node --hutch {hutch} --prometheus-dir {prom_dir} -N {N} -n
{ami_workers_per_node} -H {ami_manager_node} --log-level warning worker -b {heartbeat_period} psana://shmem=
{hutch}' }
```

1.

Note on multi-user access:

- Conflicts over activedet temp (yaml) files?
- AMI temp or user-created files

Slurm manages resources (CPUs, GPUs)

First person to talk to is Ric

In principle we have our own customizable slurm installation on drp-srcf-*, but might still need work/tweaking from IT.

Can we extend slurm to manage our resources, determined by kcu firmware? e.g.

- rix timing system fiber connection

- cameralink node
- high-rate timing nodes
- low-rate timing nodes (epics)
- hsd nodes
- generic (wave8) nodes

Slurm features to look into (could also look into other tools (airflow?) if necessary):

- job arrays
- job steps
- heterogeneous job support

Conceptually want:

- "sbatch tmo.cnf" (instead of "procmgr start tmo.cnf")
- tmo.cnf has
    - typical timing system on cmp028: "drp -P tmo -D ts ..."
    - typical control.py on mon001: "control -P tmo ..."
    - special localhost: "control_gui -P tmo -B DAQ:NEH" (this is unusual because it runs on the localhost and has a gui, only localhost processes have gui's with procmgr)
- critical features (if we could do this we may be able to replace procmgr) *underlined = done:
    - keep node allocations and command lines hardwired (like existing .cnf)
    - demo: run control/timing processes
    - need a replacement for procstat (could start with a command line version, do gui later)
    - if one job crashes, don't want the whole job to exit
    - different daq processes need different environment (pva detectors in particular, you can see this in the cnf)
    - would like python, not a bash interface for the users (maybe reuse existing cnf?)
    - need per-process log files
        - error of each process should go to that process log file --> process error (printout) goes to the per-process log while sigkill goes to slurm output
        - header of the log file (see example from procgmr)
    - support flags (p - attach plaform no., x - window, s - delay (ignored/use slurm), u (ignored))
    - two people can't run the tmo daq at the same time (use slurm detail like nodelist and jobname or comment to check if another user trying to use the same detectors/ timing). Note: it looks like with --dependency=singleton, the same user can only run one 'jobname' at a time. Preferred solution is to print out error message and exit (no pending job) when the same job name is submitted.
    - option to run each command in the cnf file as a single job or single job step
    - multithread process is allowed even when single cpu is asked by slurm. yes, per top, we see more than one drp processes running on different cores when job step is created with -n1.
    - (this already exists, currently written into ~tmoopr/.psdaq, may need that permissions work right since now daq could run as "cpo"?) "activedet": remembering the previously selected detectors? could we use "sacct"? maybe we could add info to sacct? currently control.py puts the info in files in directories like ~tmoopr/.psdaq. UPDATE: It looks like control_gui stores the values (selected detectors) directly in configdb using zmq. We probably don't need anything new for this. See below for more detail (when running control_gui with --loglevel DEBUG):
- nice-to-have features:
    - support interactive log file access like procstat
    - support live xterm for processes like procstat, and with .cnf "x" flag
    - support different conda envs for different detectors
- the "dream":
    - understand which nodes are camlink nodes ("resource management")
    - dynamically allocate requested types of nodes
    - (hard, do as a second step?) would change the BOS connections so the right detectors were connected to the allocated nodes

Note from Ric 3/1/2024:

I thought that maybe the first thing to try would be to figure out how to launch a process that brings up a GUI, e.g., groupca or xpmpva, or maybe even start simpler with xeyes or xclock. The main idea was to test the ability of telling slurm that the process that you want to run is an X11 application, which I read in the docs it can do.
The next thing might be to try to bring up the DAQ using slurm and thus thinking about what the slurm description file would look like. Can we use something like the .cnf? Can we automatically convert the .cnfs to whatever slurm requires? Or do we need to start from scratch? For this step I'm thinking we would still have to specify everything, like the node each process runs on.
The last thing I looked into a little bit was the idea of defining resources to slurm. For this I thought I'd need some setup to try things out on, which resulted in Jira ECS-4017 (I don't think anything was done though). Chris Ford was also working on this project and he suggested setting up a virtual machine with a private slurm setup I could tinker with (I haven't figured out how to do that, yet). Anyway, the idea of the resources is that based on what each DRP needs (e.g., detector type, KCU firmware type, a GPU, X11, etc.), resources would be defined to slurm so that when you launch a DAQ, it would allocate the nodes according to the resources needed and start the processes on them. Perhaps at some point in the future we could even have it modify the connections in the BOS to connect a detector to an available host that has the right KCU firmware, thus making RIX hosts available to TMO and vice versa. I think that's about as far as I got. Let me know if you have questions. I have to take Rachel to a doctor's appointment at 1 so I think I'll be out until 3 or so. We can talk later, if you prefer. I'll take a look at the link as soon as I can. Feel free to add the above to that if you think it would be helpful.

Configless setup

To share the slurm.conf file on all compute nodes, we can modify the slurm.conf file on the control node with following parameter:

slurm.conf
`SlurmctldParameters=enable_configless`

Run scontrol reconfig to enable the change. On each compute node, add the following argument to slurmd in sysconfig

/etc/sysconfig/slurmd

```
SLURMD_OPTIONS="--conf-server psslurm-drp"
```

Restart the compute node with

```
systemctl restart slurmd
```

Slurm Feature (Constraint)

We can specify a list of features (drp processes such as timing_0, teb0, etc.) for each compute node in slurm.conf. The following showing an example of Node description section in /etc/slurm/slurm.conf on psslurm-drp.

slurm.conf

```
# Nodes description
# DRP SRCF as analysis cluster
NodeName=drp-srcf-cmp035 RealMemory=128000 Sockets=1 CoresPerSocket=64 ThreadsPerCore=1 CoreSpecCount=3
Feature=timing_0,teb0,control
NodeName=drp-srcf-cmp036 RealMemory=128000 Sockets=1 CoresPerSocket=64 ThreadsPerCore=1 CoreSpecCount=3
Feature=control_gui
```

After saving the file, run

slurm.conf

```
scontrol reconfig
```

This will update the cluster feature table. You can view the available features by running:

slurm.conf

```
monarin@drp-srcf-cmp036 ~ sinfo -o "%30N  %10c  %10m  %35f  %10G "
NODELIST                       CPUS        MEMORY      AVAIL_FEATURES                      GRES
drp-srcf-cmp[031,043]          64          128000      (null)                              (null)
drp-srcf-cmp035                64          128000      timing_0,teb0,control               (null)
drp-srcf-cmp036                64          128000      control_gui                         (null)
```

To submit a job requiring a feature, you can use --constraint flag:

slurm.conf

```
#!/bin/bash
#SBATCH --partition=drpq
#SBATCH --job-name=main
#SBATCH --constraint=timing_0,teb0,control --ntasks=3
```

Heterogeneous jobs and constraint

Heterogeneous job concept allows use to spread allocations of physical cores unevenly. For drp processes, we can do this by grouping all the processes that run on the same node in one heterogeneous group (one node per group).

With slurm constraint (or feature), a node can be marked with a list of supporting drp processes. E.g.

Node	AvailableFeatures

drp-srcf-cmp035	timing,teb,control
drp-srcf-cmp036	control

Creating a heterogeneous job with groups that share the same node is NOT possible for slurm (slurm handles heterogeneous jobs with backfill and all the job steps within it must be fulfilled prior to the start of the job). To make sure that we create heterogeneous groups that do not overlap with each other, we need to pack drp processes onto available nodes.

Slow start-up time for heterogeneous jobs

Slurm backfill scheduler will delay heterogeneous jobs initiation attempts until after the rest of the queue has been processed (see https://slurm.schedmd.com/slurm.conf.html). To override this and set priority for heterogeneous jobs as highest, we need to add the following in slurm.conf.

slurm.conf

```
SchedulerParameters=bf_hetjob_immediate,bf_hetjob_prio=max
```

The setting above helps speed up the StartTime for heterogeneous jobs down to around 1-2 s (from 20s or more). However, the slow StartTime still can ben observed when we start the same heterogenous job right after canceling one.

slurm.conf

```
monarin@psslurm-drp ~ scontrol show jobid 489853
JobId=489853 HetJobId=489853 HetJobOffset=0 JobName=main
   HetJobIdSet=489853-489854
   UserId=monarin(12682) GroupId=xu(1106) MCS_label=N/A
   Priority=130 Nice=0 Account=(null) QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:11 TimeLimit=12:00:00 TimeMin=N/A
   SubmitTime=2024-04-22T15:00:48 EligibleTime=2024-04-22T15:00:48
   AccrueTime=2024-04-22T15:00:48
   StartTime=2024-04-22T15:01:04 EndTime=2024-04-23T03:01:04 Deadline=N/A
   PreemptEligibleTime=2024-04-22T15:01:04 PreemptTime=None
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-04-22T15:01:04
   Partition=drpq AllocNode:Sid=drp-srcf-cmp036:15356
   ReqNodeList=drp-srcf-cmp035 ExcNodeList=drp-srcf-cmp036
   NodeList=drp-srcf-cmp035
   BatchHost=drp-srcf-cmp035
   NumNodes=1 NumCPUs=3 NumTasks=3 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=3,node=1,billing=3
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=(null)
   WorkDir=/cds/home/m/monarin/lcls2/psdaq/psdaq/slurm
   StdErr=/cds/home/m/monarin/2024/04/22_15:00:48_drp-srcf-cmp036:slurm.log
   StdIn=/dev/null
   StdOut=/cds/home/m/monarin/2024/04/22_15:00:48_drp-srcf-cmp036:slurm.log
   Power=
   NtasksPerTRES:0

JobId=489854 HetJobId=489853 HetJobOffset=1 JobName=main
   HetJobIdSet=489853-489854
   UserId=monarin(12682) GroupId=xu(1106) MCS_label=N/A
   Priority=126 Nice=0 Account=(null) QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:11 TimeLimit=12:00:00 TimeMin=N/A
   SubmitTime=2024-04-22T15:00:48 EligibleTime=2024-04-22T15:00:48
   AccrueTime=2024-04-22T15:00:48
   StartTime=2024-04-22T15:01:04 EndTime=2024-04-23T03:01:04 Deadline=N/A
   PreemptEligibleTime=2024-04-22T15:01:04 PreemptTime=None
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-04-22T15:01:04
   Partition=drpq AllocNode:Sid=drp-srcf-cmp036:15356
   ReqNodeList=drp-srcf-cmp036 ExcNodeList=drp-srcf-cmp035
   NodeList=drp-srcf-cmp036 SchedNodeList=drp-srcf-cmp036
   BatchHost=drp-srcf-cmp036
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1,node=1,billing=1
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=(null)
   WorkDir=/cds/home/m/monarin/lcls2/psdaq/psdaq/slurm
   StdErr=/cds/home/m/monarin/2024/04/22_15:00:48_drp-srcf-cmp036:slurm.log
   StdIn=/dev/null
   StdOut=/cds/home/m/monarin/2024/04/22_15:00:48_drp-srcf-cmp036:slurm.log
   Power=
   NtasksPerTRES:0
```

The problem is more amplified when --constraint flag is used instead of --nodelist. We observed a wait time of up to 22s in the case of starting a heterogeneous job withe constraint right after cancelling one.

slurm.conf

```
monarin@psslurm-drp ~ scontrol show jobid 489849
JobId=489849 HetJobId=489849 HetJobOffset=0 JobName=main
   HetJobIdSet=489849-489850
   UserId=monarin(12682) GroupId=xu(1106) MCS_label=N/A
   Priority=130 Nice=0 Account=(null) QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:01:09 TimeLimit=12:00:00 TimeMin=N/A
   SubmitTime=2024-04-22T14:53:55 EligibleTime=2024-04-22T14:53:55
   AccrueTime=2024-04-22T14:53:55
   StartTime=2024-04-22T14:54:22 EndTime=2024-04-23T02:54:22 Deadline=N/A
   PreemptEligibleTime=2024-04-22T14:54:22 PreemptTime=None
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-04-22T14:54:22
   Partition=drpq AllocNode:Sid=drp-srcf-cmp036:15356
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=drp-srcf-cmp035
   BatchHost=drp-srcf-cmp035
   NumNodes=1 NumCPUs=3 NumTasks=3 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=3,node=1,billing=3
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=timing_0&teb0&control DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=(null)
   WorkDir=/cds/home/m/monarin/lcls2/psdaq/psdaq/slurm
   StdErr=/cds/home/m/monarin/2024/04/22_14:53:55_drp-srcf-cmp036:slurm.log
   StdIn=/dev/null
   StdOut=/cds/home/m/monarin/2024/04/22_14:53:55_drp-srcf-cmp036:slurm.log
   Power=
   NtasksPerTRES:0

JobId=489850 HetJobId=489849 HetJobOffset=1 JobName=main
   HetJobIdSet=489849-489850
   UserId=monarin(12682) GroupId=xu(1106) MCS_label=N/A
   Priority=126 Nice=0 Account=(null) QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:01:09 TimeLimit=12:00:00 TimeMin=N/A
   SubmitTime=2024-04-22T14:53:55 EligibleTime=2024-04-22T14:53:55
   AccrueTime=2024-04-22T14:53:55
   StartTime=2024-04-22T14:54:22 EndTime=2024-04-23T02:54:22 Deadline=N/A
   PreemptEligibleTime=2024-04-22T14:54:22 PreemptTime=None
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-04-22T14:54:22
   Partition=drpq AllocNode:Sid=drp-srcf-cmp036:15356
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=drp-srcf-cmp036 SchedNodeList=drp-srcf-cmp036
   BatchHost=drp-srcf-cmp036
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1,node=1,billing=1
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=control_gui DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=(null)
   WorkDir=/cds/home/m/monarin/lcls2/psdaq/psdaq/slurm
   StdErr=/cds/home/m/monarin/2024/04/22_14:53:55_drp-srcf-cmp036:slurm.log
   StdIn=/dev/null
   StdOut=/cds/home/m/monarin/2024/04/22_14:53:55_drp-srcf-cmp036:slurm.log
   Power=
   NtasksPerTRES:0
```

Running job steps in parallel

Here's an example script that shows how we can run job steps in parallel (with &).

job_step.sh

```
#!/bin/bash

#SBATCH --job-name parallel
#SBATCH --output slurm-%j.out
#SBATCH --ntasks=3                  ## number of tasks (analyses) to run
#SBATCH --cpus-per-task=2          ## the number of threads allocated to each task
#SBATCH --time=0-00:10:00

# Execute job steps
srun --ntasks=1 --nodes=1 --cpus-per-task=$SLURM_CPUS_PER_TASK bash -c "sleep 2; echo 'hello 1'" &
srun --ntasks=1 --nodes=1 --cpus-per-task=$SLURM_CPUS_PER_TASK bash -c "sleep 4; echo 'hello 2'" &
srun --ntasks=1 --nodes=1 --cpus-per-task=$SLURM_CPUS_PER_TASK bash -c "sleep 8; echo 'hello 3'" &
wait
```

For s3df, this works as you can see that the three job steps start at the same time.

```
(ps-4.6.3) sacct -j 41486534 --format=JobID,Start,End,Elapsed,REQCPUS,ALLOCTRES%30
JobID                    Start                 End    Elapsed  ReqCPUS                      AllocTRES
------------ ------------------- ------------------- ---------- -------- ------------------------------
41486534     2024-03-14T17:58:03 2024-03-14T17:58:12   00:00:09        6  billing=6,cpu=6,mem=6G,node=1
41486534.ba+ 2024-03-14T17:58:03 2024-03-14T17:58:12   00:00:09        6            cpu=6,mem=6G,node=1
41486534.ex+ 2024-03-14T17:58:03 2024-03-14T17:58:12   00:00:09        6  billing=6,cpu=6,mem=6G,node=1
41486534.0   2024-03-14T17:58:04 2024-03-14T17:58:06   00:00:02        2            cpu=2,mem=2G,node=1
41486534.1   2024-03-14T17:58:04 2024-03-14T17:58:08   00:00:04        2            cpu=2,mem=2G,node=1
41486534.2   2024-03-14T17:58:04 2024-03-14T17:58:12   00:00:08        2            cpu=2,mem=2G,node=1
```

For drp nodes, running this script shows that the second or more job steps wait for the previous one to finish.

```
(ps-4.6.3) monarin@drp-srcf-eb001 (master) slurm sacct -j 625799 --format=JobID,Start,End,Elapsed,REQCPUS,
ALLOCTRES%30
       JobID               Start                 End    Elapsed  ReqCPUS                      AllocTRES
------------ ------------------- ------------------- ---------- -------- ------------------------------
625799       2024-03-14T18:11:36 2024-03-14T18:11:45   00:00:09        6         billing=6,cpu=6,node=1
625799.batch 2024-03-14T18:11:36 2024-03-14T18:11:45   00:00:09        6             cpu=6,mem=0,node=1
625799.0     2024-03-14T18:11:36 2024-03-14T18:11:38   00:00:02        6            cpu=6,mem=6G,node=1
625799.1     2024-03-14T18:11:38 2024-03-14T18:11:42   00:00:04        6            cpu=6,mem=6G,node=1
625799.2     2024-03-14T18:11:42 2024-03-14T18:11:45   00:00:03        6            cpu=6,mem=6G,node=1
```

The example script above came from this page https://hpc.nmsu.edu/discovery/slurm/tasks/parallel-execution/. It also mentioned about hyperthreading and how this impacts slurm scheduling. I tried both changing --cpus-per-task=1 and --hint=nomultithread but nothing changed.

Update 20240312

- We think job steps (srun) will be able to satisfy the features (listed below) that we need to replace procmgr.
- Job array isn't a good option because it seems to fit with single cmd running in parallel model (we have different command/ environment for each process).
- We tested following features for srun:
    1. If one step dies, what happen? other steps continue
    2. Each job step can have independent --output and --error to send their output and error to. Note that anything related to slurm error will be written sbatch --output file.
    3. Can we use squeue/scontrol to check job step details? –> yes. Below shows job details of a single job with three job steps.

```
(ps-4.6.3) sacct -j 41308291
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
41308291       parallel     milano  lcls:data          6    RUNNING      0:0
41308291.ba+      batch             lcls:data          6    RUNNING      0:0
41308291.ex+     extern             lcls:data          6    RUNNING      0:0
41308291.0       hello2             lcls:data          2    RUNNING      0:0
41308291.1       hello3             lcls:data          2    RUNNING      0:0
41308291.2       hello1             lcls:data          2    RUNNING      0:0
```

4. Can we start/cancel individual steps? you can cancel individual step

```
(ps-4.6.3) scancel 41308291.1
(ps-4.6.3) sacct -j 41308291
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
41308291       parallel     milano  lcls:data          6    RUNNING      0:0
41308291.ba+      batch             lcls:data          6    RUNNING      0:0
41308291.ex+     extern             lcls:data          6    RUNNING      0:0
41308291.0       hello2             lcls:data          2    RUNNING      0:0
41308291.1       hello3             lcls:data          2 CANCELLED+      0:9
41308291.2       hello1             lcls:data          2    RUNNING      0:0
```

   You can only restart the entire job:

```
(ps-4.6.3) scontrol requeue 41308291
```

5. (DREAM) Can we specify resource (eg. ask for a gpu node)? yes

```
srun --partition ampere --account lcls:data -n 1 --time=00:10:00 --gpus=1 --pty /bin/bash
```

   Note on viewing slurm cluster info:

```
monarin@sdfiana002 ~ sinfo -o "%20N  %10c  %10m  %95f  %10G "
NODELIST              CPUS        MEMORY
AVAIL_FEATURES
GRES
sdfrome[003-123]      128         512000      CPU_GEN:RME,CPU_SKU:7702,CPU_FRQ:2.00
GHz                                                   (null)
sdfmilan[001-072,101  128         512000      CPU_GEN:RME,CPU_SKU:7713,CPU_FRQ:2.00
GHz                                                   (null)
sdfampere[001-023]    128         1024000     CPU_GEN:RME,CPU_SKU:7542,CPU_FRQ:2.10GHz,GPU_GEN:AMP,
GPU_SKU:A100,GPU_MEM:40GB,GPU_CC:8.0        gpu:a100:4
sdfturing[001-016]    48          191552      CPU_GEN:SKX,CPU_SKU:5118,CPU_FRQ:2.30GHz,GPU_GEN:TUR,
GPU_SKU:RTX2080TI,GPU_MEM:11GB,GPU_CC:7.5   gpu:geforc
more info about sinfo format:
https://slurm.schedmd.com/sinfo.html#SECTION_EXAMPLES
```

6. (DREAM) Can we swithc the BOS connection when the resources have been allocated"