What a GReaT Scheduling Opportunity

Gary Skouson
Institute for Computational and Data Sciences
Penn State University

ICDS Constraints (GReaT)

We're using Slurm

- GReaT (Guaranteed Response Time) allocations – Paid allocations.
  - Groups "purchase" cores/memory/gpus - Guarantee of start time within an hour
  - "unlimited" runtime (More like several weeks…)
  - "roll-over" minutes for unused time
    - Allow expanded usage above paid allocation. (burst)
    - These jobs are eligible for preemption
    - Guaranteed runtime before preemption
  - Track/limit overall account usage to the resources paid for.
- Different CPU/GPU types must be differentiated.
  - "pin" accounts to specific node types.
- We need to allow for "open" usage by all users
  - 48-hour run limit - Preemptable jobs – limited resources

Daily Account Updates

- Service agreements maintained by administrative staff (end up in a database)
- Daily script:
  - Add New accounts (SLA)
    - Added to Slurm with grptres limits and grptresmins limits (CPU, mem, GPU) and node type constraints.
  - Add Time to Existing accounts:
    - Add 1-day of time to existing accounts (e.g. grptresmins cpu,gpu)
  - Clean up "expired" accounts
- Job Submit filter to force paid jobs to proper node types.

What about “roll-over” minutes?

- The grptres limit didn’t allow for “overuse”
- I wanted to change the grptres limit for some jobs to allow users to use more than their allotted cores/gpus.
  - With the understanding that these “burst” jobs can be preempted.
- Introduced **LimitFactor** (included in Slurm 21.08)
  - https://github.com/SchedMD/slurm/commit/7924ddb5d25d03c1bc92a2921d43e6d2350ac824
  - Changes the grptres limit by this factor for jobs in this QOS
    - Grptres cpu=20 with LimitFactor=2 means we can use 40 cores

What isn't GReaT

- I'm not trying to convince anyone that this is the best model for job scheduling.
- This is more of an example of a way we met some interesting and non-standard ***requirements.***
- Odd scheduling requirements can require additional work and code to implement with stock tools.
- Tradeoff between response and utilization still exists.
  - With short enough runtime and small enough jobs, you can increase likelihood of *fairness* with good response time and high utilization.

Thanks

Slurm Details

- Account mytest:
  - Grptres: cpu=192,gres/gpu=8,mem=1.50T
  - Grptresmins: cpu=16588800,gres/gpu=691200
  - Description: [ntype=gc]
- QOS:
  - Open
    - MaxTRESPU: cpu=100,gres/gpu=0,mem=800G,node=10
    - maxjobspu=100
  - Sla – LimitFactor=1
  - Burst2x (etc)
    - Limitfactor=2.0
    - PreemptExemptTime = 1-hour

More Slurm Details

- Partitions:

```
PreemptMode=REQUEUE
PreemptType=preempt/partition_prio
AccountingStorageEnforce=safe

PartitionName=open Default=YES QOS=open MaxTime=48:00:00 PreemptMode=requeue PriorityTier=5
PartitionName=sla-prio QOS=sla DenyAccounts=open PriorityTier=100
PartitionName=burst DenyAccounts=open AllowQOS=burst2x PreemptMode=requeue PriorityTier=50
```


Job Submit Filter

- Force node constraints based on account info
- Force open partition jobs to open qos (and open qos to open partition)
- Try to keep people from submitting jobs that won't run
  - Seems like a losing battle.