Scheduling By Trackable Resources

Morris Jette and Dominik Bartkiewicz
SchedMD

Slurm User Group Meeting 2018

Thanks to NVIDIA for sponsoring this work

Goals

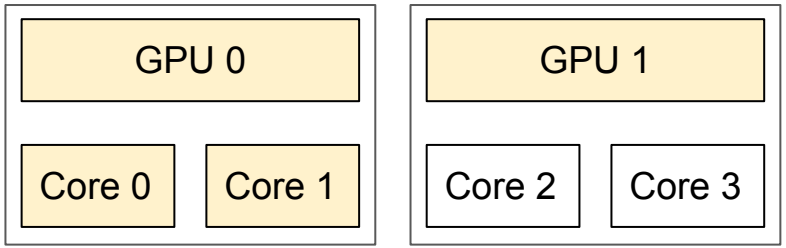
- More flexible scheduling mechanism
  - Especially for clusters with significant GPU resources
- Greater control over GPU resources
  - Task binding
  - Frequency control
- Improved performance of scheduling logic

Current Shortcomings (part 1 of 3)

- GPUs allocated as a fixed count of per allocated node
  - 2 GPUs per node OK
  - 4 GPUs on one node and 2 GPUs on another node not possible
- No controls over GPU frequency/power or per-task binding
- No consideration of GPUs with NVLink (high speed communications between GPUs and GPUs/CPUs) to select preferred GPUs for co-scheduling

Current Shortcomings (part 2 of 3)

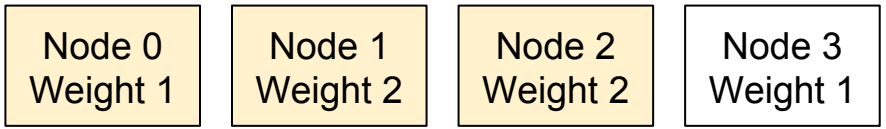
- First CPUs with adjacent GPUs are identified
- Next specific CPUs are selected for the job allocation
- Finally GPUs are selected for the job allocation
  - Favors use of CPUs and GPUs that are on same socket
  - May not be possible since GPUs are selected after CPUs



Cores 0 and 1 might be selected for a job allocation requiring 2 GPUs rather than selecting one core on each socket with a GPU

Current Shortcomings (part 3 of 3)

- Node scheduling weights used in sub-optimal fashion
  - If job allocation can't be satisfied with lowest weight nodes then all nodes with the lowest and next lowest weight considered for use without allocating as many of the lowest weight nodes first



A 3-node job allocation might not use all of the lowest weight nodes

New select/cons_tres Plugin

- “cons_tres” represents “Consumable TRES”
- “TRES” represents “Trackable RESources”
- All functionality provided by “cons_res” plugin is also supported by “cons_tres” (e.g. CR_LLN, CR_PACK_NODES, CR_SOCKET, etc.)
- Addresses all of the previously cited shortcomings
- New “gpu” job options only supported the cons_tres plugin
  - No other select plugin recognizes the new GPU options

New Job Submit Options

Same options apply to salloc, sbatch and srun commands

- --cpus-per-gpu= CPUs required per allocated GPU
- -G/--gpus= GPU count across entire job allocation
- --gpu-bind= Task/GPU binding option
- --gpu-freq= Specify GPU freq, memory freq, voltage
- --gpus-per-node= Works like “--gres=gpu:#” option today
- --gpus-per-socket= GPUs per allocated socket
- --gpus-per-task= GPUs per spawned task
- --mem-per-gpu= Memory per allocated GPU

New Configuration Parameters

Parameters available globally and on per-partition basis. The command line options override these default values.

- DefCpusPerGPU= Default CPUs count per allocated GPU
- DefMemPerGPU= Default memory size per allocated GPU

Data Structure Changes (part 1 of 2)

The new “GPU” options are translated by job submit commands to “tres” options and reported by “scontrol show job”, sview, and squeue accordingly.

- *--gpus-per-node=4* translates to *tres-per-node=gpu:4*
- *--mem-per-gpu=12* translates to *mem-per-tres=gpu:12*

Data Structure Changes (part 2 of 2)

- Internally cons_tres tracks resources on a per-socket basis and accumulates co-located GPUs and cores when possible
- Cons_tres also uses a set of per-node core-bitmaps to track resources rather than a cluster-wide core-bitmap
  - Supports changing core count on a node without restarting slurmctld daemon (important for cloud)

Examples of Use

$ sbatch --ntasks=16 --gpus-per-task=2 my.bash

$ sbatch --ntasks=8 --ntasks-per-socket=2 --gpus-per-socket=tesla:4 my.bash

$ sbatch --gpus=16 --gpu-freq=low,verbose --gpu-bind=closest --nodes=2 my.bash

$ sbatch --gpus=gtx1080:8,gtx1060:2 --nodes=1 my.bash

Conflicting Options (part 1 of 2)

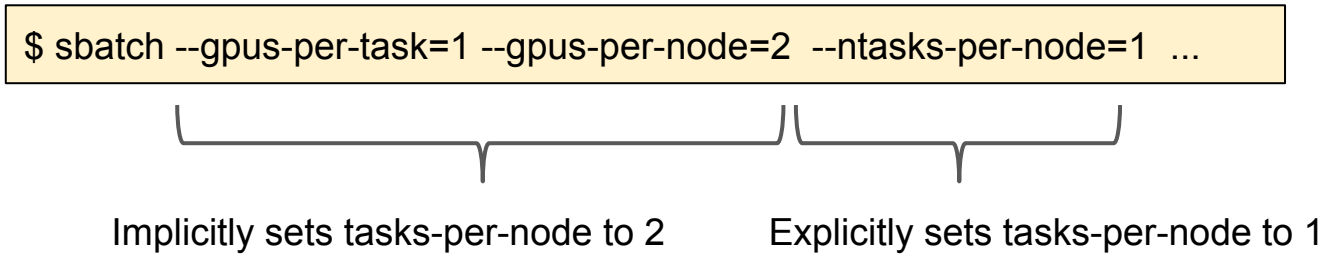
- Given the multitude of options, it is possible to submit a job with conflicting options
  - Many conflicting options are possible
  - In most cases the job will be rejected

```
$ sbatch --gpus-per-task=1 --cpus-per-gpu=2  --cpus-per-task=1  ...
```

Implicitly sets cpus-per-task to 2

Explicitly sets cpus-per-task to 1

Conflicting Options (part 2 of 2)



Deployment Schedule

November 2018 - Full functionality available in pre-release of Slurm version 19.05, unsupported

May 2019 - Slurm version 19.05 to contain full functionality with support

February 2020 - Slurm version 20.02 *might* remove the cons_res plugin

Future Work

The infrastructure is quite generic. It could be easily extended to support concepts like licenses per node, task, etc.

Questions?