Slurmを用いた計算実行方法

HPCシステムズ株式会社 HPC事業部

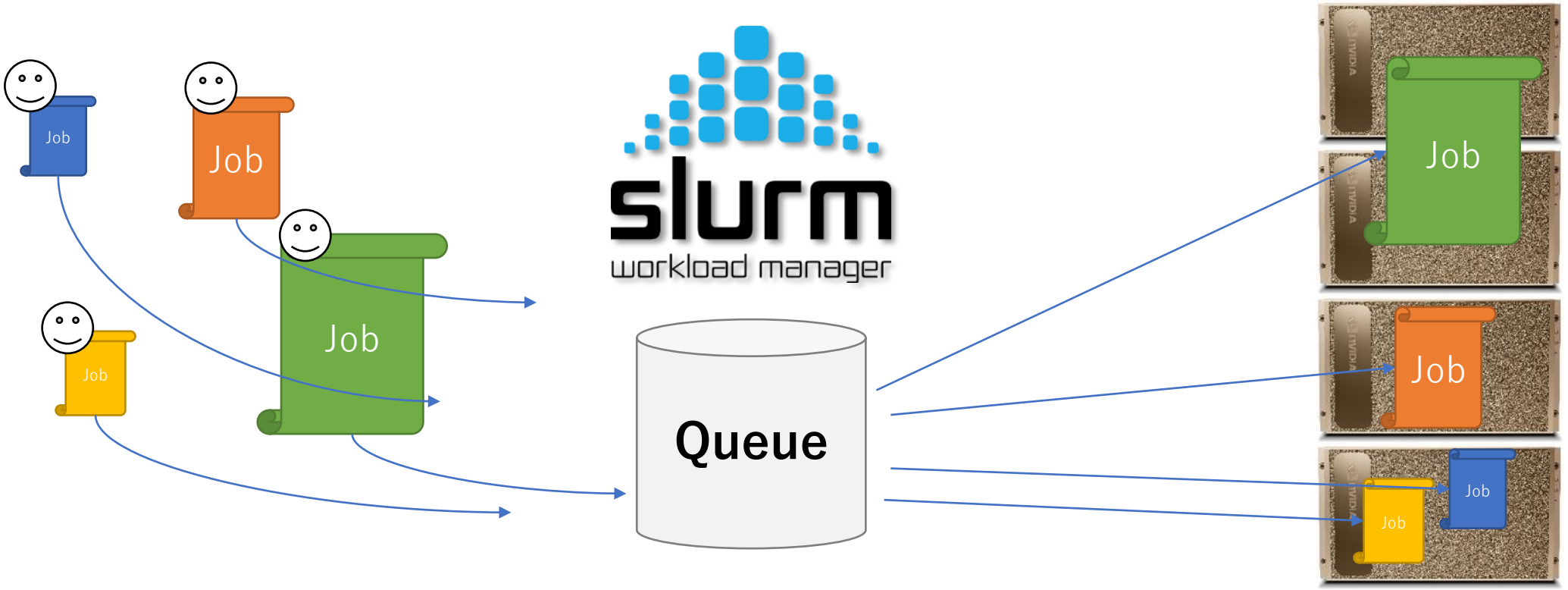
2021/03/25作成

目次

- Slurm 概要
- 主要なSlurmコマンドの説明
- Python環境構築とプログラム実行のながれ
- 補足事項・注意事項

Slurm

計算機の稼働率を高めながら共有するために
実行させたい計算(ジョブ)をいったん受け取って、
適切な計算機・適切なタイミングで実行するシステム



Slurm用語集

ジョブ	ユーザーがノードに実行させたいコマンド列
ノード	１台の計算機
パーティション	１台以上のノードからなる仮想的な集合
ジョブスクリプト	ジョブをBashスクリプト形式で記述したファイル
投入する	実行させたいジョブをSlurmに登録すること
キャンセルする	Slurmに投入されたジョブを取り消すこと
キュー	Slurmがジョブを溜めている記憶領域
ジョブスケジューリング	どのジョブをいつどのノードで実行するかの計画を作ること

sbatch コマンド：ジョブ投入

```
$ sbatch ジョブスクリプト
Submitted batch job 123
```

ジョブ投入に成功するとジョブIDが出力されます。

オプション例	概要
sbatch -J "my job name"	ジョブに任意のジョブ名をつけます。
sbatch -p パーティション名	指定したパーティションにジョブを投入します。
sbatch -N ノード数	ノード数を指定します。
sbatch -n タスク数	ジョブ全体で立ち上げるタスク数を指定します。
sbatch -c コア数	1タスクあたりに必要とするCPUコア数を指定します。デフォルトは1です。1タスクが複数ノードを跨がらないようにノード割り当てが行われます。
sbatch -o ./stdout_%j.log	標準出力を保存するファイル名を指定します。%jはジョブIDに置換されます。デフォルトはslurm-%j.outです。
sbatch -e ./stderr_%j.log	標準エラー出力を保存するファイル名を指定します。%jはジョブIDに置換されます。デフォルトはslurm-%j.outです。

ジョブスクリプトの作成例

- 1ノード(-N)
- 1プロセス(-n)
- 1スレッド(-c)
- 1GBメモリ(--mem=)

を要求

```
#! /bin/bash
#SBATCH -p sv-dgx
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -c 1
#SBATCH --mem=1G
#SBATCH -J 01node-01core-01GB
#SBATCH -o 01node-01core-01GB_%J_out.txt
#SBATCH -e 01node-01core-01GB_%J_err.txt

./myapp
```

その他のジョブスクリプトオプションは
`$ man sbatch` を参照

ジョブスクリプトの作成例

- 1ノード
- 1プロセス
- 20スレッド（20コア）
- メモリ30GB

を要求

```
#! /bin/bash
#SBATCH -p sv-dgx
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -c 20
#SBATCH --mem=30G
#SBATCH -J 01node-20core-30GB
#SBATCH -o 01node-20core-30GB_%J_out.txt
#SBATCH -e 01node-20core-30GB_%J_err.txt

./myapp
```

ジョブスクリプトの作成例

- 1ノード
- 2プロセス
- 20スレッド（40コア）
- メモリ30GB

を要求

```
#! /bin/bash
#SBATCH -p sv-dgx
#SBATCH -N 1
#SBATCH -n 2
#SBATCH -c 20
#SBATCH --mem=30G
#SBATCH -J 01node-40core-30GB
#SBATCH -o 01node-40core-30GB_%J_out.txt
#SBATCH -e 01node-40core-30GB_%J_err.txt

./myapp
```

ジョブスクリプトの作成例

- 2ノード
- 2プロセス
- 20スレッド（40コア）
- メモリ30GB/ノード

を要求

```
#! /bin/bash
#SBATCH -p sv-dgx
#SBATCH -N 2
#SBATCH -n 2
#SBATCH -c 20
#SBATCH --mem=30G
#SBATCH -J 02node-40core-30GB
#SBATCH -o 02node-40core-30GB_%J_out.txt
#SBATCH -e 02node-40core-30GB_%J_err.txt

./myapp
```

ジョブスクリプトの作成例

- 1ノード
- 1プロセス
- 1スレッド（1コア）
- メモリ1GB
- 1GPU/ノード

を要求

```
#! /bin/bash
#SBATCH -p sv-dgx
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -c 1
#SBATCH --mem=1G
#SBATCH --gpus-per-node=1
#SBATCH -J 01node-01core-01GB-1GPU
#SBATCH -o 01node-01core-01GB-1GPU_%J_out.txt
#SBATCH -e 01node-01core-01GB-1GPU_%J_err.txt

./myapp
```

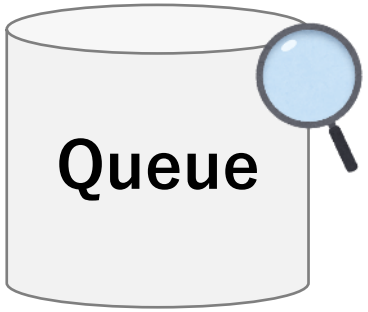
アプリケーション側からは、
要求した数のGPUだけがCUDAデバイスとして見えます。
ただしnvidia-smiコマンドでは全部見えます。

sinfo -s コマンド：ノード稼働状況の監視


```
$ sinfo -s
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
cluster1*    up   infinite          1/3/0/4 node[1-4]
```

カラム	内容
PARTITION	パーティションの名称
AVAIL	パーティションの状態（upまたはinact）
TIMELIMIT	最大実行時間
NODES(A/I/O/T)	ノードの状態（allocated/idle/other/total）
NODELIST	パーティションに割り当てられたノード（[]でまとめて表記）

squeue コマンド：ジョブキューの監視




```
$ squeue
     JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
       123  cluster1    myjob      hpc R       0:02      2 node[1-2]
```

カラム	内容
JOBID	ジョブID
PARTITION	パーティションの名称
NAME	ジョブの名称
USER	投入したユーザー
ST	ジョブの状態の省略名
TIME	ジョブに使われた時間（日-時間:分:秒）
NODES	ジョブに割り当てられたノードの数
NODELIST(REASON)	ジョブに割り当てられたノード（ペンディング状態のジョブは括弧内に待機理由）

実際のジョブ実行の様子

```
$ sinfo -s
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
sv-dgx*      up   infinite          0/4/0/4 sv-dgx[01-04]
$ squeue
     JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

```
$ sbatch 01node-20core-30GB.sh
Submitted batch job 187
$ sinfo -s
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
sv-dgx*      up   infinite          1/3/0/4 sv-dgx[01-04]
$ squeue
     JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
       187    sv-dgx 01node-2 dgxadmin  R       0:03      1 sv-dgx01
```

実演（別資料）

その他のコマンド

- scancel コマンド：
  投入されたジョブをキャンセル

```
$ scancel 123
```

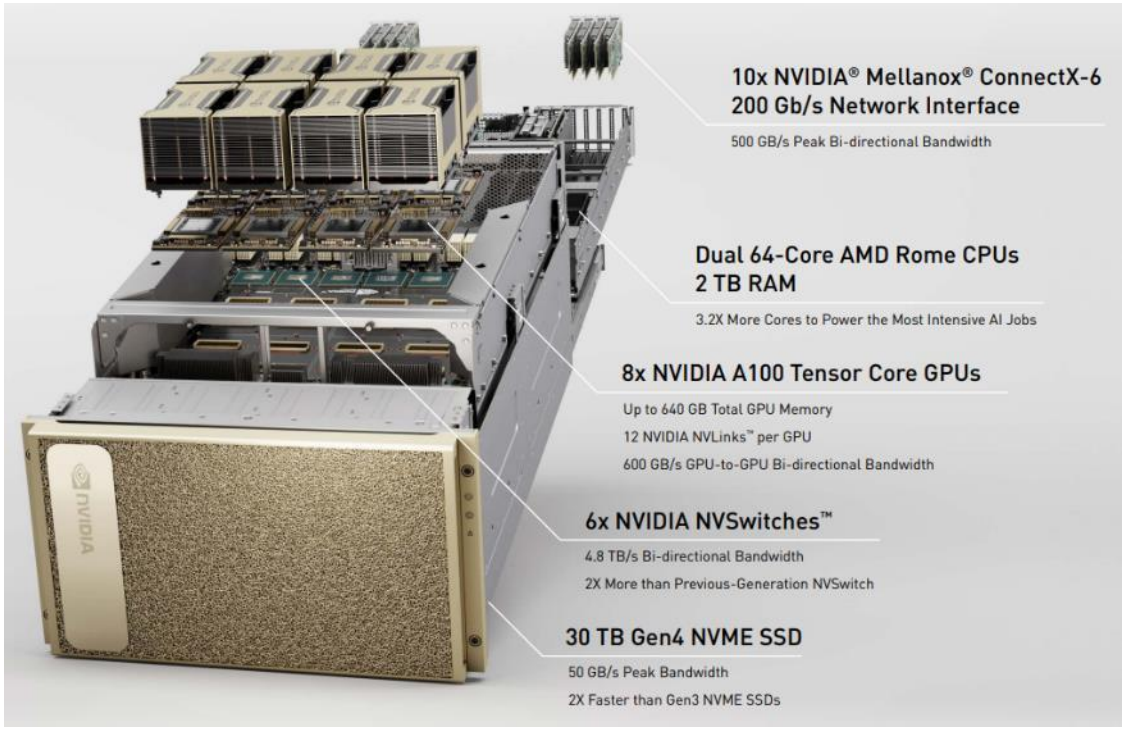
- sacct コマンド：
  実行完了したジョブ情報を出力

```
$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
187          01node-20+     sv-dgx       test         20  COMPLETED      0:0
187.batch         batch                  test         20  COMPLETED      0:0
```

sbatchで投入したジョブにはbatchというジョブステップが付与される

参考）DGX A100のスペック

- 1ノードあたり
  - CPU: 128コア（64コアCPUが2基）
  - メモリ: 2TB
  - GPU: A100(80GB) が8基
  - ストレージ: 30TB NVMe SSD



このノードにジョブから要求された以上の資源があるならばSlurmはジョブを実行します。
資源の空きが無ければ、別のノードを検討します。
それでも資源の空きが無ければ、待ちとなります。

ジョブスケジューリング方針：Backfill

Fig. 1. Examples of FCFS and FCFS + Backfill.

キューの先頭の待機ジョブの開始時刻が変わらない範囲で、後続のジョブを先に実行する

これによりノード全体の稼働率を高められる

ジョブスクリプトの注意点

#SBATCH --mem=xxxG を極力指定してください

- 指定しない場合、無限大のメモリを消費するジョブとして扱われます
- つまり、CPUやGPUが空いていても他のジョブを同時に
  実行できなくなります　⇒稼働率低下の原因になります

```
#! /bin/bash
#SBATCH -p sv-dgx
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -c 1
#SBATCH --mem=1G
```

参考情報

- Slurm各種コマンドのマニュアル
  https://slurm.schedmd.com/man_index.html

- Slurmドキュメント一覧
  https://slurm.schedmd.com/documentation.html

- Slurmのライセンス(GNU GPL v2)
  https://slurm.schedmd.com/disclaimer.html