Slurm ユーザーマニュアル

目次

1 Slurm Workload Manager について

Slurm Workload Manager（以下、Slurm）は、大小さまざまな Linux クラスタを対象にした、オープンソースの、耐故障性のある、高度に大規模対応している、クラスタ管理およびジョブスケジューリングのシステムです。Slurm により、ユーザーは一定期間ジョブを流すためのリソース（計算ノード）に対する排他的および非排他的なアクセスが可能になります。Slurm は確保されたノードに対するジョブの起動・実行・監視のフレームワークを提供し、また、ペンディング（保留中）のジョブのキューの管理によってリソースの競合を調停します。Slurm のオプショナルなプラグインとして、アカウンティング、先行予約、ギャングスケジューリング、バックフィル、トポロジ最適化されたリソース選定、リソース上限設定、ジョブの優先順位といった機能が提供されています。

本マニュアルでは Slurm のインストール内容、使用方法について概説します。

2　Slurm インストール概要

(1)　パッケージ

公式サイト（ https://slurm.schedmd.com/ ）から配布されている Slurm のソースコードを使用します。

(2)　インストールディレクトリ

`/usr/local/slurm-バージョン`

※バージョン には 20.11.4 等の Slurm バージョンが入ります。

(3)　管理ユーザー

`slurm`

(4)　Slurm デーモン群

以下のデーモンから構成されています。これらは OS 起動時に自動起動します。

- `slurmd` … クラスタの各計算ノードで起動します。
- `slurmctld` … クラスタの管理ノードで起動します。
- `slurmdbd` … クラスタの管理ノードで起動します。

(5)　環境設定ファイル

Slurm を使用するための環境設定は各ユーザーのホームディレクトリのファイルで行われています。tcsh をご使用の場合は `~/.cshrc` 、bash をご使用の場合は `~/.bashrc` ファイル内で `/home/.common` 以下に用意した Slurm の環境設定スクリプトを実行します。

なお、標準設定では root は Slurm の環境が設定されていないのでご注意下さい。root で Slurm のコマンドを使用する際は以下コマンドで Slurm 環境をセットして下さい。

tcsh の場合は以下コマンドを実行します。

```
# source␣/home/.common/00-slurm.csh
```

bash の場合は以下コマンドを実行します。

```
# .␣/home/.common/00-slurm.sh
```

3　コマンド例

本項では Slurm で使用するコマンドを概説します。

(1)　sinfo␣-s

パーティション（計算ノードの仮想的なグループ）の情報を出力します。

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

出力は以下の通りです。

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

詳細は Slurm 公式サイトの sinfo ページを参照ください。

(2) sbatch

Slurm にジョブを投入します。予め、ジョブスクリプトを作成しておく必要があります。sbatch コマンドにジョブスクリプトを指定して実行することで、ジョブがキューイングされ実行されます。

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

ジョブ投入に成功すると、Submitted batch job に続いてジョブ ID が出力されます。

よく使われるオプションを以下に示します。

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。

sbatch コマンドのオプションはジョブスクリプト冒頭の #SBATCH で始まる行として指定することもできます。

詳細は Slurm 公式サイトの sbatch ページを参照ください。

(3) squeue

ジョブキューの状態を出力します。

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

出力は以下の通りです。

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）

ST には次表に示す種類があります。

省略名	名称	意味
BF	BOOT_FAIL	起動失敗（典型的にはハードウェア故障等による）
CA	CANCELLED	キャンセルされた
CD	COMPLETED	終了コード 0 ですべてのプロセスが終了した
CF	CONFIGURING	資源が割り当てされたが、資源が利用可能になるのを待機している
CG	COMPLETING	完了処理中だが、おそらく一部のノードの一部のプロセスがまだ動作中
DL	DEADLINE	終了日時に達して強制終了された
F	FAILED	非 0 の終了コードやその他の失敗条件で終了した
NF	NODE_FAIL	一つ以上の割り当てられたノードの故障で終了された
OOM	OUT_OF_MEMORY	メモリ不足となった
PD	PENDING	資源割り当てを待っている
PR	PREEMMPTED	他のジョブによる割り込みで終了された
R	RUNNING	割り当て中
RD	RESV_DEL_HOLD	停止中
RF	REQUEUE_FED	複数クラスタ連携によって再キューイングされた
RH	REQUEUE_HOLD	停止中のジョブが再キューイングされた
RQ	REQUEUED	完了中のジョブが再キューイングされた
RS	RESIZING	ジョブのサイズが変更中
RV	REVOKED	他のクラスタがジョブを起動したので、当該クラスタからジョブが除去されている
SI	SIGNALING	シグナルを受けている
SE	SPECIAL_EXIT	特別な状態で再キューイングされた。これはユーザーによって指定されることがある
SO	STAGE_OUT	ファイルをステージアウトしている
ST	STOPPED	割り当てられたものの、ジョブが SIGSTOP シグナルによって停止している。CPU コアはこのジョブによって捕まれている
S	SUSPENDED	割り当てられたものの、実行が中断されている。CPU コアは他のジョブに開放される
TO	TIMEOUT	実行時間制限によってジョブが強制終了された

詳細は Slurm 公式サイトの squeue ページを参照ください。

(4) sstat

実行中のジョブの情報を出力します（実行中である必要があります）。CPU、タスク、ノード、Resident Set Size（RSS）、Virtual Memory（VM）に関する情報が出力されます。-o オプションを使って出力させる項目をカスタマイズすることができます。

```
$ sstat␣-j␣ジョブ ID␣-o␣出力する項目[,項目]...
```

出力できる項目には以下などがあります。出力可能な項目の一覧は-e オプションで表示できます。

出力項目	内容
JobID	ジョブ ID
MaxVMSize	ジョブのすべてのタスクでの仮想メモリサイズの最大値
MaxRSS	ジョブのすべてのタスクでの物理メモリ消費サイズの最大値
MaxPages	ジョブのすべてのタスクのページフォールトの最大数
MaxDiskRead	ジョブのすべてのタスクの最大読み込みバイト数
MaxDiskWrite	ジョブのすべてのタスクの最大書き込みバイト数

詳細は Slurm 公式サイトの sstat ページを参照ください。

(5) scancel

scancel コマンドにジョブ ID を指定すると、投入したジョブをキャンセルします。

```
$ scancel␣123
```

詳細は Slurm 公式サイトの scancel ページを参照ください。

(6) sacct

実行完了したジョブの情報を出力します。

```
$ sacct
       JobID    JobName  Partition    Account   AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
1234            sample    cluster1    account1          2  COMPLETED      0:0
```

出力は以下の通りです。

カラム	内容
JobID	ジョブ ID
JobName	ジョブの名称
Partition	パーティションの名称
Account	ユーザーが紐づいているアカウントの名称
AllocCPUS	割り当てられた CPU コア数
State	ジョブの状態
ExitCode	終了コード（コロンに続くのはプロセス終了を引き起こしたシグナル）

詳細は Slurm 公式サイトの sacct ページを参照ください。

4 ジョブの投入

4.1 逐次ジョブの投入方法

逐次ジョブのジョブスクリプトの例を次に示します。

```
#! /bin/bash
#SBATCH -p cluster1          ·····  (1)
#SBATCH -n 1                 ·····  (2)
#SBATCH -J myjob             ·····  (3)
#SBATCH -o stdout_%J.txt     ·····  (4)
#SBATCH -e stderr_%J.txt     ·····  (5)
./a.out                      ·····  (6)
```

(1) パーティション名を指定しています。
(2) このジョブのタスク数を指定しています。逐次ジョブではプロセス数が 1 なので 1 です。
(3) ジョブ名を指定しています。
(4) 標準出力の保存先を指定しています。
(5) 標準エラー出力の保存先を指定しています。
(6) プログラムを実行しています。

このように作成したジョブスクリプト（ここではファイル名を./myjob.sh とします）を、次のように sbatch コマンドで投入します。

```
$ sbatch ./myjob.sh
```

4.2 スレッド並列ジョブの投入方法

スレッド並列ジョブのジョブスクリプトの例を次に示します。

```
#! /bin/bash
#SBATCH -p cluster1           ・・・・・・・・・・ (1)
#SBATCH -n 1                  ・・・・・・・・・・ (2)
#SBATCH -c 16                 ・・・・・・・・・・ (3)
#SBATCH -J myompjob           ・・・・・・・・・・ (4)
#SBATCH -o stdout_%J.txt      ・・・・・・・・・・ (5)
#SBATCH -e stderr_%J.txt      ・・・・・・・・・・ (6)
export OMP_NUM_THREADS=16     ・・・・・・・・・・ (7)
./a.out                       ・・・・・・・・・・ (8)
```

(1) パーティション名を指定しています。
(2) このジョブのタスク数を指定しています。スレッド並列ジョブではプロセス数が 1 なので 1 です。
(3) 確保する必要のある CPU コア数を指定しています。この例では 16 スレッド並列で動かすために必要な 16 コアを指定しています。
(4) ジョブ名を指定しています。
(5) 標準出力の保存先を指定しています。
(6) 標準エラー出力の保存先を指定しています。
(7) OpenMP のスレッド並列数を環境変数 OMP_NUM_THREADS に指定しています。
(8) プログラムを実行しています。

このように作成したジョブスクリプト（ここではファイル名を./myompjob.sh とします）を、次のように sbatch コマンドで投入します。

```
$ sbatch ./myompjob.sh
```

4.3 MPI 並列ジョブの投入方法

MPI 並列ジョブのジョブスクリプトの例を次に示します。

```
#! /bin/bash
#SBATCH -p cluster1          ・・・・・・・・・・ (1)
#SBATCH -n 4                 ・・・・・・・・・・ (2)
#SBATCH -J mympijob          ・・・・・・・・・・ (3)
#SBATCH -o stdout_%J.txt     ・・・・・・・・・・ (4)
#SBATCH -e stderr_%J.txt     ・・・・・・・・・・ (5)
mpirun -n 4 ./a.out          ・・・・・・・・・・ (6)
```

(1) パーティション名を指定しています。
(2) このジョブのタスク数を指定しています。MPI 並列ジョブではプロセス数を指定します。この例では 4 を指定しています。
(3) ジョブ名を指定しています。
(4) 標準出力の保存先を指定しています。
(5) 標準エラー出力の保存先を指定しています。
(6) MPI を用いてプログラムを実行しています[1]。

このように作成したジョブスクリプト（ここではファイル名を./mympijob.sh とします）を、次のように sbatch コマンドで投入します。

```
$ sbatch ./mympijob.sh
```

[1] 一部の MPI では、Slurm で確保されたノードの情報が、MPI のホストファイルに自動的に引き継がれない場合があります。その場合、sbatch ページの OUTPUT ENVIRONMENT VARIABLES に列挙されている環境変数（SLURM_JOB_NODELIST 等）を用いて MPI のホストファイルを作成して MPI に引き渡してください。[]を用いて纏められたノードリストを展開するには scontrol show hostnames コマンドが便利です（$ scontrol show hostnames 'node[1-4]'）。

4.4 GPU 利用ジョブの投入方法

Slurm は Gres というプラグインをオプションで追加設定することで GPU リソースのスケジューリングに対応します。GPU を利用するジョブのジョブスクリプトの例を次に示します。

```
#! /bin/bash
#SBATCH -p cluster1            ·····  (1)
#SBATCH -n 1                   ·····  (2)
#SBATCH --gpus-per-node 8      ·····  (3)
#SBATCH -J mygpujob            ·····  (4)
#SBATCH -o stdout_%J.txt       ·····  (5)
#SBATCH -e stderr_%J.txt       ·····  (6)
./a.out                        ·····  (7)
```

(1) パーティション名を指定しています。
(2) このジョブのタスク数を指定しています。この例では 1 を指定しています。
(3) ノードあたりの利用 GPU 数を指定しています。この例では 8 を指定しています。
(3) ジョブ名を指定しています。
(4) 標準出力の保存先を指定しています。
(5) 標準エラー出力の保存先を指定しています。
(6) MPI を用いてプログラムを実行しています。

このように作成したジョブスクリプト（ここではファイル名を./mygpujob.sh とします）を、次のように sbatch コマンドで投入します。

```
$ sbatch ./mygpujob.sh
```

付録A　HPC システムズ お問い合わせ先


弊社ホームページ　http://www.hpc.co.jp/support_index.html

サポート案内やお問い合わせの多い内容など様々な情報を掲載しております。
是非ご活用ください。

HPC システムズ株式会社
〒108-0022　東京都港区海岸 3-9-15 LOOP-X 8 階

HPC 事業部


【営業】03-5446-5531　【サポート】03-5446-5532

お電話によるサポート受付は祝日、弊社指定休日を除く月曜日から金曜日の 9:30～17:30 とさせて頂きます。


【FAX】03-5446-5550


【電子メール】hpcs_support@hpc.co.jp