RCCS計算機利用の注意点

bias5との比較を中心に

24/June/2024 Hiroyo NISHIDE

RCCSにおけるジョブ管理システム

全ての解析はジョブ管理システムを経由して実行すること

- 基本はPBSの仕様に従う
  - biasではPBSをそのまま採用していた

- RCCSではジョブ投入、ジョブ確認などに独自コマンドを採用している

コマンド対応表　bias5/RCCS ジョブ管理システム

bias5(PBS)コマンド	RCCSコマンド	説明
qsub *file_name*	**jsub *file_name***	ジョブを投入する
qstat -u *my_account*	**jobinfo**	自分のジョブの状態を表示
qstat -Q	**jobinfo -s**	キュー/ジョブタイプの詳細を表示
qstat	**jobinfo -a**	全てのジョブを表示（-g で同グループのみを表示）
qdel *jobID*	**jdel *jobID***	指定したIDのジョブを削除
qhold *jobID*	**jhold *jobID***	指定したIDの待機中ジョブを実行されないようにホールド
qrls *jobID*	**jrls *jobID***	ホールド状態のジョブをリリース（実行待ち状態にする）
tracejob *jobID*	**joblog**	終了したジョブの履歴を表示（CPU点数も表示）

ジョブスクリプトの書き方

biasと全く同じ書き方では
うまく動きません！

- 同じ点
  - 実行したいコマンドをシェルスクリプトに書く
- 異なる点
  - リソース指定の仕方

biasでのスクリプト例

```
#!/bin/bash
#PBS -q medium
#PBS -l mem=8Gb
#PBS -l ncpus=12

cd ${PBS_O_WORKDIR}
diamond blastp --db spo.dmnd --out sce.tab --thread 12 --query myprot.fasta
```

ジョブスクリプトにおけるリソース指定の書き方

- RCCSではキュー名の指定はなく、指定されたリソース量によって投入先ノードが変わる
- **#PBS -l** に続けて「:」で区切りながらリソースを指定する
- mem= で総メモリ量の指定はできない

```
#!/bin/bash
#PBS -l select=1:ncpus=12:mpiprocs=1:ompthreads=12
#PBS -l walltime=4:00:00
source /apl/bio/etc/bio.sh
module load diamond
cd ${PBS_O_WORKDIR}
diamond blastp --db spo.dmnd --out sce.tab --outfmt 6 --query sce_prot.fasta
```

RCCS用スクリプト例

ジョブスクリプトにおけるリソース指定の書き方

```
#!/bin/bash
#PBS -l select=1:ncpus=12:mpiprocs=1:ompthreads=12
#PBS -l walltime=4:00:00
```

- 必ず指定するリソース
  - ノード数：**select=** 1ノードで実行する
  - ノードあたりのコア数：**ncpus=** 12コアを使う
  - ノードあたりのプロセス数：**mpiprocs=** プロセスは1つ
  - プロセスあたりのスレッド数：**ompthreads=** 12スレッドで実行する
  - 計算時間(制限) (時間):(分):(秒)：**walltime=** 最大計算時間は4時間

ompthreads = ncpus x mpiprocs であること

ジョブスクリプトにおけるリソース指定の書き方

- 必要に応じて指定するリソース
  - ジョブタイプ： **jobtype=**

```
#!/bin/bash
#PBS -l select=1:ncpus=64:mpiprocs=1:ompthreads=64:jobtype=largemem
```

  - 大容量メモリ(256Gb以上)を使いたい場合

7.785Gbx64=504Gbメモリ

  - ノードあたりのGPU数：**ngpus=**

```
#!/bin/bash
#PBS -l select=1:ncpus=128:mpiprocs=1:ompthreads=128:ngpus=8
```

  - GPUを使いたい場合

8GPU, 128コア

ジョブ実行:　jsub

- シェルスクリプトを作成

```
#!/bin/sh
#PBS -l select=1:ncpus=12:mpiprocs=1:ompthreads=12
#PBS -l walltime=4:00:00
diamond blastp --db spo.dmnd --out sce.tab --outfmt 6 --query sce_prot.fasta
```

my_script.sh

- jsubコマンドにシェルスクリプトのファイルを渡して実行

```
$ jsub my_script.sh
6439875.ccpbs1
```

- jobinfo でジョブの実行状況を確認

```
$ jobinfo
--------------------------------------------------------------------------------
Queue    Job ID Name              Status CPUs User/Grp       Elaps Node/(Reason)
--------------------------------------------------------------------------------
H(c)     6439875 test.sh           Run       4   ebv/zo0     0:00:20  ccc147
--------------------------------------------------------------------------------
```

キューは今のところ一種類「H」
jobtypeは「c」

jobtype=　ジョブタイプについて

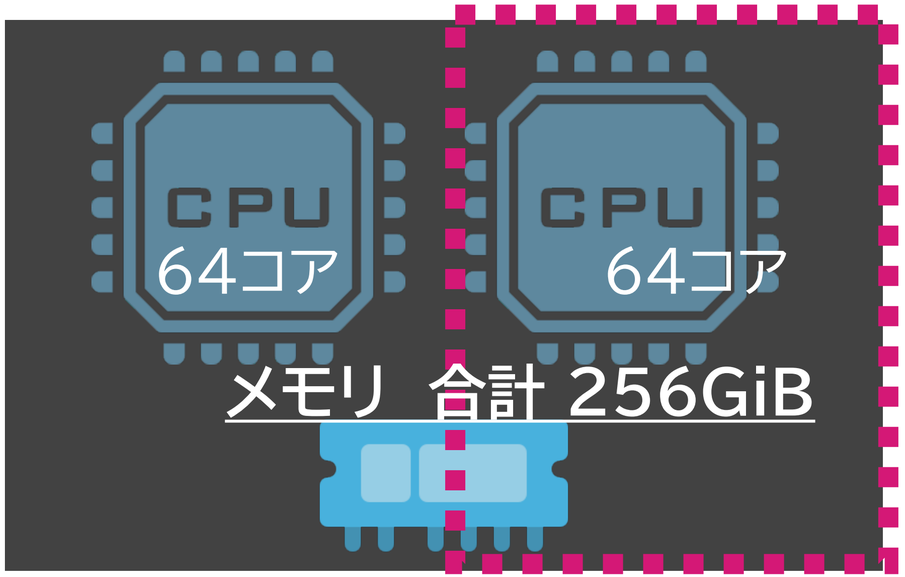
jobtype	計算ノード	メモリ/コア	利用単位	1ジョブあたりの制限	特徴
largemem	TypeF	7.875 Gb	vnode / ノード	1～14 vnode(s) （64～896コア） （ncpus=64 or 128）	大容量メモリ 1/2ノード(1vnode) 以上占有
vnode	TypeC	1.875 Gb	vnode / ノード	1～50 vnode(s) （64～3,200コア） （ncpus=64 or 128）	1/2ノード(1vnode) 以上占有
core	TypeC	1.875 Gb	コア	1～63 コア （ncpus<64）	64コア以下 総メモリ118Gb以下
gpu	TypeG	1.875 Gb	コア	1～48 GPU, 1～16 コア/GPU (ngpus>0)	GPU利用

- largemem 以外の jobtype は指定されたリソースより判定可能なため省略可
- 使えるメモリ量は、指定コア数 x 計算ノードタイプのコア当たりのメモリ量

出典：https://ccportal.ims.ac.jp/manual/queues

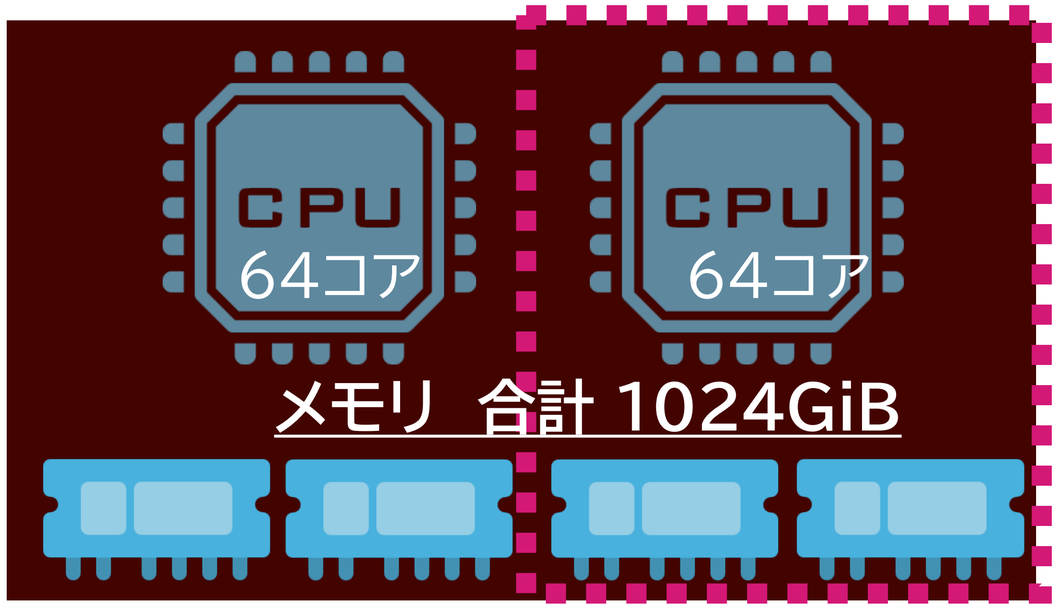
ノードタイプとジョブタイプ

TypeC　　804台
jobtype : core / vnode



Vノード（1/2ノード）
jobtype=vnode で使われる単位

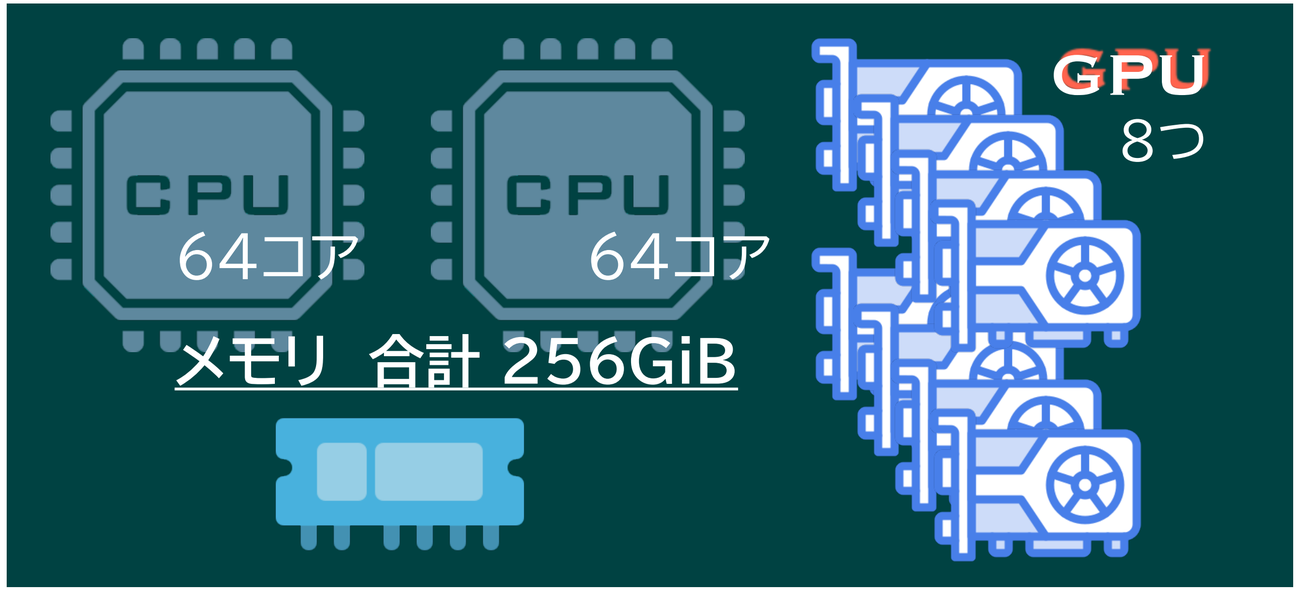
TypeF　　14台
jobtype : largemem



Vノード（1/2ノード）

ノードタイプとジョブタイプ

TypeG　　16台
jobtype : gpu



1ノードに
　8GPU＋128CPUコア
　1GPUあたり16コア

参考:

- 分子生物学アプリケーションにおいて、複数プロセス立ち上げや、複数マシンにまたがった計算を行えるものは非常に少ない
  - select>1 の指定が有効なアプリケーションはほとんどない
- 複数ノード指定が有効なのは、物理マシンをまたがって並列計算できるアプリケーション（MPI等を使う）のみ

- PBSによって制限されたメモリを超えて使おうとして、PBSによりジョブが強制終了されることはない（biasとの相違点）
  - アプリケーション上で足りなくなって落ちることはある
- 指定したwalltime以内に終わらないとPBSによって強制終了

CPU点数

jobtype	CPU キュー係数	GPU キュー係数
largemem	60 点 / (1vnode * 1 時間)	-
vnode	45 点 / (1vnode * 1 時間)	-
core	1 点 / (1コア * 1 時間)	-
gpu	1 点 / (1コア * 1 時間)	60 点 / (1GPU * 1 時間)

- 消費点数は実際のジョブ実行時間で計算される。 walltimeではない

出典：https://ccportal.ims.ac.jp/manual/queues

リソースの指定とCPU点数計算例

- diamond blastx：nr検索、クエリDNA配列：96,706件、クエリファイルサイズ183MB
- 1ノード、16コア指定
- jobtype: **core**

```
#PBS select=1:ncpus=16:mpiprocs=1:ompthreads=16
```

- 所要時間: 73:54:05、typeCノードで実行された
- 消費点数: 1点 x 16コア x 73.9時間 ＝ 1,182点

リソースの指定とCPU点数計算例2

- InterProScan クエリDNA配列：115,079件、クエリファイルサイズ227MB
- 1ノード、64コア指定（1 vnode）
- jobtype: **largemem**

```
#PBS select=1:ncpus=64:mpiprocs=1:ompthreads=64:jobtype=largemem
```

- 所要時間：16:30:22、typeFノードで実行された
- 消費点数：60点 x 1vnode x 16.5時間 ＝ 990点

残りCPU点数の確認

- これまでにグループで消費したCPU点数を表示

```
$ showlim -c
```

- これまでにグループで消費したCPU点数をメンバーごとに表示

```
$ showlim -c -m
```

- 過去のジョブが消費してきた点数を表示

```
$ joblog
```

アレイジョブ

- オプション　**#PBS -J** 開始番号-終了番号
- ジョブファイル中で変数 **${PBS_ARRAY_INDEX}** に番号がセットされ、インクリメントしながら回数分実行される

```
#!/bin/sh
#PBS -J 1-5
#PBS -l select=1:ncpus=4:mpiprocs=1:ompthreads=4
#PBS -l walltime=4:00:00

diamond blastx --threads ${NCPUS} --db nr --outfmt 6 \
 --query ./my.${PBS_ARRAY_INDEX}.fa --out ./out.${PBS_ARRAY_INDEX}.tab
```

- 上記は4コアジョブが5回サブミットされたのと同じ
- 消費点数は5つのジョブそれぞれがかかった時間に依存する
- あまり細かく分割しないこと（大量サブミットしたユーザのジョブは実行が遅れる）

アレイジョブ

```
$ jobinfo
-------------------------------------------------------------------------
Queue   Job ID Name              Status CPUs User/Grp        Elaps Node/(Reason)
-------------------------------------------------------------------------
H(c)   6441620[] test_array.sh  Array      4  ebv/zo0      0:11:39
H(c)   6441620[1] test_array.sh  Run        4  ebv/zo0       0:10:17 ccc035
H(c)   6441620[2] test_array.sh  Run        4  ebv/zo0       0:10:16 ccc035
H(c)   6441620[3] test_array.sh  Run        4  ebv/zo0       0:10:17 ccc035
H(c)   6441620[4] test_array.sh  X          4  ebv/zo0       0:00:04 ccc039
H(c)   6441620[5] test_array.sh  X          4  ebv/zo0       0:00:04 ccc039
-------------------------------------------------------------------------
```

ストレージについて

- 1メンバーあたり500Gb
- それ以上が必要な場合は申請時にその理由を記載
- CPU点数、ディスク容量の追加申請も可能

- グループメンバーのストレージ使用量と残量を表示

```
$ showlim -d -m
```

一時ファイルのためのディレクトリ

- 原則として `/tmp, /var/tmp, /dev/shm` の一時ディレクトリの使用は禁止

- jsubジョブ実行では、環境変数 ${TMPDIR} が 自動で上記以外に設定されている
- 書き込み可能容量 ： 11.9Gb/コア
  - 不足する場合は、コア数を増やす

- biasにおける `/scratch` のように使える領域はありません

▸定期メンテナンス

▸報告書

- メンテナンス等の最新情報は［RCCS］メールニュース を参照のこと
- メンテナンス日は(9:00～17:00)ログインできない
  - 現在は偶数月の1日
  - 不定期に変更あり

- 年度の変わり目ごとに報告書を提出すること
- 2024年6月1日締切

情報の在処など

- 計算科学研究センター（RCCS）Webサイト

  https://ccportal.ims.ac.jp/

- 基生研が編集している移行情報wiki

  https://biaswiki.nibb.ac.jp/menu/index.php/RCCS

- 基礎生物学分野利用に関するお問い合わせ先

  support@nibb.ac.jp