2026年度
バイオユーザのための
京大化研スパコンシステム利用法

日本ヒューレット・パッカード合同会社
上原英也

spradm@scl.kyoto-u.ac.jp

0774-38-3265　(内線: 3265)
https://www.scl.kyoto-u.ac.jp/

共催
京都大学化学研究所附属バイオインフォマティクスセンター
NPOバイオインフォマティクス・ジャパン

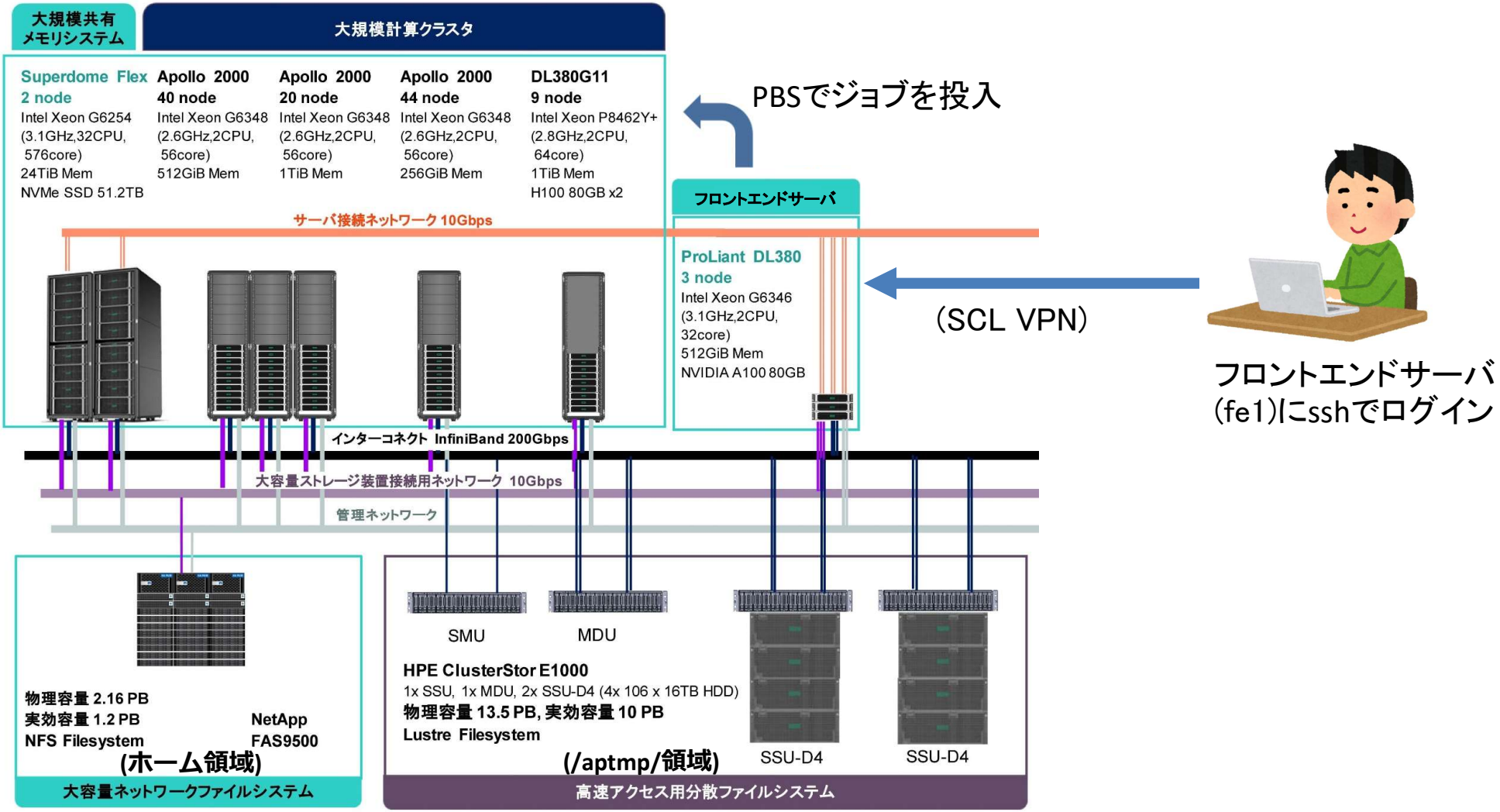
ver 1.0.0

内容

1 システム概要

1.1 システム構成図

（参）「システムの紹介」



1.2 サーバスペック

（参）「システムの紹介」→「大規模共有メモリシステム」
「システムの紹介」→「大規模計算クラスタ」

	大規模共有メモリ システム	大規模計算クラスタ (CPUノード)	大規模計算クラスタ (GPUノード)	フロントエンド サーバ
機種	HPE Superdome Flex	HPE Apollo2000 G10+	HPE DL380 G11	HPE DL380 G10+
ノード数	2	104	9	3
ホスト名	sdf1, sdf2	csXX, cmXX, clXX	ch1～ch9	fe1
CPU	Intel Xeon G6254 3.1GHz x 32CPU (18コア/CPU)	Intel Xeon G6348 2.6GHz x 2CPU (28コア/CPU)	Intel Xeon P8462Y+ 2.8GHz x 2CPU (32コア/CPU)	Intel Xeon G6346 3.1GHz x 2CPU (16コア/CPU)
コア数/ノード	576	56	64	32
メモリ/ノード	24TiB	256GiB (cs01～cs44) 512GiB (cm01～cm40) 1TiB (cl01～cl20)	1TiB	512GiB
GPU	-	-	NVIDIA H100 x 2 (80GB)	NVIDIA A100 (80GB)
SSD	50TB	-	-	-
OS	Red Hat Enterprise Linux(＊) 8.7			

(＊) x86_64 (amd64) Linux用 binaryプログラムが動作します。

2 利用にあたって

2.1 新規利用申請 (アカウントの取得)

(参)「各種手続き」→「新規利用申請」

- 支払責任者が京都大学教職員であるか否かによって申請手順が異なります。

- スパコンシステム(計算サーバ)で計算される場合
  - 支払い責任者が京都大学教職員:
    - 「新規利用申請書を作成する」→「利用形態」で「計算サーバ使用」を選択
  - 支払い責任者が京都大学教職員以外
    - 「新規利用申請用紙(PDF)」の「計算サーバの利用」の「有」にチェック
- 支払責任者が申請者となります。
- 【新規利用申請書作成時の注意点】をお読み下さい。
  - お問い合わせ
    tel: 0774-38-3265　(内線: 3265)
    email: spradm@scl.kyoto-u.ac.jp

2.1 新規利用申請 (アカウントの取得) (続き)

（参）「各種手続き」→「新規利用申請」

- 研究成果報告書（計算サーバの利用者）
  - 以下に該当する方はご提出は不要です
    - 所属が企業（営利目的でのご利用）
    - 新規利用申請が1月以降
    - 計算サーバもしくはアプリケーションを全く利用しなかった(※)
    - 特許出願中もしくは出願準備中などにより、研究内容を非公開としたい(※)

    (※) この場合には、その旨をスパコンシステムまでお知らせください。

2.2 利用負担金

(参)「各種手続き」→「利用負担金」

課金対象	計算ルール
基本料金	1,000円/月
計算サーバ(CPU時間)[注1,2)]	0.00220円/秒
CPU時間に対する課金の上限額	支払責任者の所属 京都大学化学研究所：40,000円/月 京都大学（化学研究所は除く）：60,000円/月 学術機関（京都大学は除く）：80,000円/月 民間機関：200,000円/月 ＊CPUノード 約10,000円／台／日
ディスク基本料金 (ホーム領域)	無料 (100GBまで利用可能)
クラウドストレージサービス	無料 (100GBまで利用可能)
オプションサービス	
ディスク拡張サービス(ホーム領域)	+300GB 1,000円/年度 (最大900GBを追加可能)
クラウドストレージ拡張サービス	+300GB 1,000円/年度 (最大900GBを追加可能)

- 注1) 並列プログラムで同時に複数のCPUを利用した場合には、各CPUでのCPU時間の合計を課金の対象とします。
- 注2) GPUを利用した計算の場合、計算ジョブの経過時間（秒）をCPU時間と換算して課金します。
- 計算一時領域 (/aptmp/(ユーザ名)/) は課金されません(容量制限もなし)。
- 「各種手続き」→「利用負担金の参照」で利用金額等をご確認頂けます。

2.3 スパコンシステムログイン手順

(参)「使い方と注意事項」→「パソコンからの使い方」
「使い方と注意事項」→「計算サービス」→「ホスト名とログイン」

- スパコンシステムを利用するには フロントエンドサーバ fe1 にログインして下さい。
  - (Linux, Mac) ターミナルで

```
$ ssh アカウント名@fe1.scl.kyoto-u.ac.jp
```

(× login.scl.kyoto-u.ac.jp)

(Windows) ターミナルソフト(コマンドプロンプト、MobaXterm, TeraTerm, Putty, Rlogin, 等)

  - fe1の GUIアプリケーション(seaview, LibreOffice, gnuplot, FigTree, IGV 等)を利用する場合
    - (Linux) ssh -Y (もしくは ssh -Y -C)でログイン。
    - (Mac) Xquartz をインストールして、ssh -Y (もしくは ssh -Y -C)でログイン。
    - (Windows) MobaXterm or Xming をインストール
      (参)「使い方と注意事項」→「パソコンからの使い方」→「WindowsでX11フォワーディング」

2.3 スパコンシステムログイン手順 (続き)

(参)「使い方と注意事項」→「基本サービス」→「ホスト名とログイン」

- ログインシェル(コマンドラインインターフェース)の変更
  - csh, tcsh (初期設定), bash, zsh を選択できます。

```
$ ldapchsh bash          # bashに変更する場合 (フロントエンドサーバ fe1 で実行して下さい)
```

- 京大ネットワーク(KUINS)以外の外部ネットワークからフロントエンドサーバにログインする場合は、まず最初にVPN でスパコンシステムのネットワークに接続して下さい。
  - (Windows, Linux, Mac) https://vpn.scl.kyoto-u.ac.jp/
  - (Mac) Mac App Store から F5Access をインストール

2.3 スパコンシステムログイン手順 (続き)

（参）「使い方と注意事項」→「パソコンからの使い方」→「Windowsでファイル転送」
（参）「使い方と注意事項」→「パソコンからの使い方」→「MacOSXでファイル転送」

- 自PC⇔フロントエンドサーバ間でファイル転送するには以下のソフト・コマンドをご利用下さい。
  (Windows) WinSCP, FileZilla, MobaXterm
  (Mac) Cyberduck, FileZilla
  (Linux) scp, sftp, rsync

  - 改行コードに注意！
    Windowsで作成したテキストファイルをスパコンシステムにコピーして実行しようとすると、
    改行コードの問題でエラーになることがあります。
    （例）
    '¥r': コマンドが見つかりません
    ^M: コマンドが見つかりません

  - テキストファイルの改行コードをLinux用に変換
    $ dos2unix (ファイル)
    もしくは
    $ nkf -w -Lu --overwrite (ファイル)　　# 改行コード変換 + 文字コード変換(UTF-8)

2.3 スパコンシステムログイン手順 (続き)

（参）「使い方と注意事項」→「基本サービス」→「ホスト名とログイン」

- フロントエンドサーバではジョブ当たり、30分・8GBのCPU時間制限・メモリ制限があります。
- 計算サーバで対話的に作業するにはPBSのインタラクティブバッチジョブ（§ 4.7)を利用して下さい
- 原則として、計算サーバ（スパコン本体）には直接sshでログインしないで下さい
  （ジョブが正しく実行されているかtop 等で確認するくらいであれば可）。

2.4 ディスク領域

（参）「使い方と注意事項」→「計算サービス」→「ディスク領域」

	ホーム領域	計算一時領域	SSD
場所	~（ユーザ名）/	/aptmp/（ユーザ名）/	${TMPDIR}/
ファイルシステム	NFS	Lustre	xfs
サーバ間共有	○	○	✖
課金	有	無	-
容量制限	有	無	-
Snapshot	有	無	-
備考			SDFキューでのみ利用可 (§4.11)

2.4.1 ホーム領域 (ホームディレクトリ)

（参）「使い方と注意事項」→「計算サービス」→「ディスク領域」

- Snapshot
  - ホームディレクトリ内の各ディレクトリに .snapshot/(日時)/ という隠しディレクトリがあり、その日時でのファイルを参照できます。

```
$ ls .snapshot/
daily.2026-05-14_0010  weekly.2026-05-10_0015  weekly.2026-05-03_0015
```

  - 誤って削除・修正したファイルを .snapshot/ から復元できます(普通に cp でコピー)。
  - .snapshot/ は ls -a で見えません！（タブ補完も効きません）
  - .snapshot/ の中のファイルは削除・変更できません。

- ディスク使用量の確認 (quota)

```
$ quota -s
Disk quotas for user lect-1 (uid 10151):
     Filesystem  space  quota  limit  grace  files  quota  limit  grace
fas9500-03_NFS:/HOME/user1/
                72366M  81920M   100G         6870   1800k  2000k
```

2.4.2 計算一時領域 (/aptmp/(ユーザ名)/)

(参)「使い方と注意事項」→「計算サービス」→「ディスク領域」

- 基本的には計算時はこの領域を使用して下さい
- 課金・容量制限・保存期限はありません
- 空き容量が少なくなってきた場合はファイルの整理(圧縮・削除)をお願いする場合があります
  - ディスク使用量だけでなく、ファイル数にもご注意下さい！
- ディスク使用量の確認(注)

```
$ lfs-quota -h
Disk quotas for usr lect-1 (uid 10151):
     Filesystem    used   quota   limit   grace   files   quota   limit   grace
       /lustre/  512.1G      0k      0k       -  203327       0       0       -
```

ディスク使用量

総ファイル数

(注) lfs-quota コマンドは /lustre/ ファイルシステムでの全使用量を出力するため、
/aptmp/ の使用量とずれが生じることがあります(/scratch/ を使用している場合)

- Snapshotの機能はありません(バックアップはご自身で取得して下さい)

2.4.3ディレクトリサイズ・ファイル数の確認 (1/2)

- ディレクトリサイズ（ディスク使用量）
  - du コマンド

```
$ du -hs (ディレクトリ)/              # (ディレクトリ)/の全サイズ
$ du -hS (ディレクトリ)/              # (ディレクトリ)/の中の各ディレクトリ直下の合計ファイルサイズ
$ du -hS (ディレクトリ)/ | sort -h    # ディレクトリのサイズでソート
```

  - mpifileutils（大量のファイルを含むディレクトリ用） New!

```
$ module load mpifileutils
$ mpirun -np 8 dwalk (directory)/ --no-atime
…
[2026-04-27T18:45:34] Walked 101117373 items in 2613.485 seconds (38690.629 items/sec)
[2026-04-27T18:45:35] Items: 101117373
[2026-04-27T18:45:35]   Directories: 10894223
[2026-04-27T18:45:35]   Files: 84023121
[2026-04-27T18:45:35]   Links: 6200029
[2026-04-27T18:45:35] Data: 86.231 TiB (1.076 MiB per file)
```

2.4.3ディレクトリサイズ・ファイル数の確認 (2/2)

- ファイル数
  - du コマンド

```
$ du --inodes -s (directory)/                # (directory)/ 下の総ファイル・ディレクトリ数.
$ du --inodes -S -t 10000 (directory)/    # 直下にファイルが10,000個以上あるディレクトリだけ出力
```

  - find コマンド

```
$ find (directory)/ | wc -l                # (directory)/ 下の総ファイル・ディレクトリ数
$ find (directory)/ -type f | wc -l     # (directory)/ 下の総ファイル数
$ find (directory)/ -type d | wc -l    # (directory)/ 下の総ディレクトリ数
```

  - mpifileutils (大量のファイルを含むディレクトリ用)

```
$ module load mpifileutils
$ mpirun -np 8 dwalk (directory)/ --lite (--no-atime)
…
[2026-04-27T18:45:34] Walked 101117373 items in 2613.485 seconds (38690.629 items/sec)
[2026-04-27T18:45:35] Items: 101117373
[2026-04-27T18:45:35]   Directories: 10894223
[2026-04-27T18:45:35]   Files: 84023121
[2026-04-27T18:45:35]   Links: 6200029
```

2.5 ウェブアプリケーション

（参）「使い方と注意事項」→「基本サービス」→「メールシステムの利用」
「使い方と注意事項」→「基本サービス」→「クラウドストレージ」

（注）ウェブメール、クラウドストレージページ内の
「ログアウト」ボタンは機能しません。

2.6 システム稼働状況

- スパコンシステムホームページのトップページ

← ノード毎の詳細情報
（要スパコンアカウント）

- リアルタイムでサーバの使用状況を確認: qstatmyjobs コマンド（§ 4.9）

```
fe1{lect-1}1001: qstatmyjobs -q
User: lect-1
                      JOBS            CPUS                    MEM(gb)                  GPUS          WALLTIME(h)
Queue  avail(use%) mysum/max   avail max mysum/max   avail   max mysum/max   avail max mysum/max  default max
------ ----------- ---------- ----------------------------- ------------------------ ------------------- -----------
SMALL    870( 85%)    0/UNLTD    12  12     0/96       48    48     0/UNLTD     -   -     -/-          6 12
APC      702( 87%)    0/UNLTD    21  56     0/UNLTD    12   980     0/UNLTD     -   -     -/-       2880 UNLTD
APG      480( 16%)    0/UNLTD    64  64     0/UNLTD   980   980     0/UNLTD     2   2     0/UNLTD   2880 UNLTD
SDF      608( 39%)    0/8         - 144     0/288      - 12288     0/UNLTD     -   -     -/-       2880 UNLTD
================== ========== ============================= ======================== =================== ===========
TOTAL:          JOBS)  0/UNLTD       CPUS)   0/500          MEM)     0/18432        GPUS)  0/4
```

3 アプリケーションの利用

3.1 moduleコマンド

（参）「アプリケーション一覧」→「module コマンド」

- スパコンシステムのアプリケーションは module コマンドで管理されています。

```
$ module avail                          # 利用できるアプリケーションの module を表示。
$ module avail -L                       # 各アプリケーションの最新版だけを表示。
$ module avail bl                       # 名前が bl から始まる module だけを表示。

$ module load [-s] (module名)           # アプリケーションの module を読み込む。そのアプリケーションが
                                          依存する別のモジュールも併せてload されることがあります。
                                          -sオプションを付けると読み込み時のメッセージが表示されません。

$ module list                           # 現在読み込んでいる module の確認。

$ module switch [-f] (module名1) (module名2)   # 読み込んでいる module の切り替え(同一アプリの
                                                 場合はmodule名１は省略可)。エラーが出てswitch
                                                 できない場合は -f オプションを付けて下さい。

$ module unload (module名)              # 指定した module を破棄。
$ module purge                          # 全て破棄。
```

3.1 moduleコマンド (続き)

- シェルスクリプト（PBSジョブスクリプト）の中で module コマンドを使用する際は
  source /etc/profile.d/modules.sh　　（sh/bashスクリプト）
  source /etc/profile.d/modules.csh　　（csh/tcshスクリプト）
  を含めて下さい。
- 同時に読み込むモジュールの組み合わせによっては、アプリケーションが動作不良を起こすことがありますので、アプリケーションの実行後にその都度 module unload (or purge)するようにして下さい。

```
#!/bin/sh
source /etc/profile.d/modules.sh

module load prog1
prog1 xxxx
module purge

module load prog2
prog2 yyyy
module purge
```

3.2 バイオインフォマティクスアプリケーション

（参）「アプリケーション一覧」→「バイオインフォマティクス」

- インストールされているバイオインフォマティクスアプリケーション
  https://www.scl.kyoto-u.ac.jp/Appli/#biotool
- ゲノムネットサービスで利用しているバイオインフォマティクスアプリケーション

名称	概要	モジュール名
BLAST+	ホモロジー検索	blast+
Clustal Omega	マルチプルアラインメント	clustal-omega
ClustalW2	マルチプルアラインメント	clustalw2
DBGET	統合データベース検索システム	dbget
ETE Toolkit	系統樹作成	ete
FASTA	ホモロジー検索	fasta
FastTree	系統樹作成	FastTree
FastTreeMP	系統樹作成(FastTreeのOpenMP並列化版)	FastTreeMP
ghostx	ホモロジー検索	ghostx
ghostz	ホモロジー検索	ghostz
HMMER	モチーフ検索	hmmer
MAFFT	マルチプルアラインメント	mafft
MUSCLE	マルチプルアラインメント	muscle
PHYLIP	系統樹作成	phylip
PRRN	マルチプルアラインメント	prrn
raxml	系統樹作成	raxml
SIMCOMP	化合物構造検索	simcomp
ssearch	ホモロジー検索	ssearch
SUBCOMP	化合物部分構造検索	subcomp

3.3 バイオインフォマティクスデータベース

（参）「アプリケーション一覧」→「バイオインフォマティクスDB」

- バイオインフォマティクスデータベース: **/db/(種別)/(データベース名)/**
  - 種別: blast, bowtie, bowtie2, dbget, diamond, fasta, ghostx, hmmer, motif, rpsblast
  - データベース名: genbank, refseq, mgenes, ncbi, swissprot, trembl, pfam, 等
    (例) NCBI nr のBLASTファイル: /db/blast/ncbi/nr.*
  - データベース更新情報: /db/dbinfo.txt
- /db/ の実体は /lustre/db/YYYYMMDD/ にあります。
  - 古いものは、.tpxz 形式で圧縮して保持しています。
  - ファイルの抽出手順は
    /lustre/db/00_How_to_extract_files.txt
    /lustre/db/extract.sh
    を参照下さい。
- NCBI, EBI, PDBj のFTPサイト(一部)のミラー
  /db/ftp.ncbi.nih.gov/ # genbank, refseq, genomes, taxonomy など
  /db/ftp.ebi.ac.uk/ # uniprot, uniref など
  /db/ftp.pdbj.org/
  /db/logan/

3.3 バイオインフォマティクスデータベース(続き)

- 利用可能なバイオインフォマティクスデータベース

(*) academic use only

データベース名	概要	種別	ディレクトリ名
GenfBank	塩基配列データベース	dbget, blast, fasta	genbank
GenPept	塩基配列データベース	dbget, blast, fasta, diamond, ghostx	genpept
RefSeq	塩基配列・アミノ酸配列データベース	dbget, blast, fasta, diamond, ghostx	refseq
MGENES	メタゲノム情報	dbget, blast, fasta, diamond, ghostx	mgenes
NR-NT	重複を取り除いた塩基配列データベース	dbget, blast, fasta	nr-nt
NR-AA	重複を取り除いたアミノ酸配列データベース	dbget, blast, fasta, diamond, ghostx	nr-aa
NCBI BLASTデータベース	NCBIで公開されているBLASTデータベース (nr, nt, swissprot, refseq_protein, taxdb等)	blast, fasta, diamond, ghostx	ncbi
UniProt/Swiss-Prot	アミノ酸配列データベース	dbget, blast, fasta, diamond, ghostx	swissprot
UniProt/TrEMBL	アミノ酸配列データベース	dbget, blast, fasta, diamond, ghostx	trembl
UniRef	アミノ酸配列データベース	blast, fasta, diamond, ghostx	uniref
dbEST	EST(Expressed Sequence Tags)配列	blast, fasta	dbest
dbGSS	GSS(Genome Survey Sequences)配列	blast, fasta	dbgss
dbSTS	STS(Sequence Taged Sites)配列	blast, fasta	dbsts
Silva (*)	リボソームRNA配列データベース	dbget, blast, fasta	silva
RDP	リボソームRNA配列データベース	dbget, blast, fasta	rdp
PR2	リボソームRNA配列データベース	dbget, blast, fasta	pr2
PDBSTR	PDBのアミノ酸配列	blast, fasta, diamond, ghostx	pdbstr
Pfam	タンパク質ドメインファミリー	dbget, hmmer	pfam
NCBI CDD	NCBI Conserved Domain Database	dbget, rpsblast	ncbi-cdd

4. バッチシステム PBS

4.1 バッチシステム (ジョブスケジューラー)とは

(参)「使い方と注意事項」→「計算サービス」→「ジョブ投入システム」

- 共有の計算機システムにおいて、複数のユーザができるだけ公平に計算機リソース(CPU, GPU, メモリ)を利用できる様にジョブのスケジューリングを行います。

- 当スパコンシステムではバッチシステムとして Altair社の PBS Professional を使用しています。

- 実行したいジョブをファイル(ジョブスクリプトファイル)に書いて、それをフロントエンドサーバ fe1 から qsub コマンドで投入します。

- ユーザの使用可能コア数・メモリ

同時実行ジョブの合計コア数	500
同時実行ジョブの合計メモリ	18TB

- 同時に投入できるジョブ数: 10,000 (アレイジョブのサブジョブを含む)

4.2 ジョブの投入

（参）「使い方と注意事項」→「計算サービス」→「ジョブ投入システム」→「ジョブ投入オプション」

$ qsub　[オプション]　（ジョブスクリプトファイル）

- 主なオプション
  - -q xxxx　　キューの指定 (SMALL, APC, APG, SDF)
    ＊キューによって使用できるマシン、リソース、ジョブの優先度が異なります
  - -l xxxx　　ジョブの実行に必要なリソースを指定
    - 主なリソース
      - » -l select=1:ncpus=(使用コア数):mem=(使用メモリ)
        (ncpus, mem を指定しなかった場合はキューのデフォルト値になります)
      - » GPUを使う場合: -l select=1:ncpus=(使用コア数):mem=(使用メモリ):ngpus=1
      - » ジョブの最大経過時間を指定: -l walltime=(時間):(分):(秒)
  - -N xxxx　　ジョブ名の指定
  - -o xxxx　　標準出力ファイルのパス
  - -e xxxx　　標準エラー出力ファイルのパス
  - -j oe　　標準出力と標準エラー出力をまとめて出力
- qsubコマンドでジョブを投入すると Job ID が発行されます
  - ジョブがうまく実行できない等の問い合わせの際は Job ID をご連絡下さい

4.3 キュー一覧

（参）「使い方と注意事項」→「計算サービス」→「ジョブ投入システム」

キュー名	SMALL	APC	APG(※3)	SDF
計算サーバ:台数	クラスタ(CPU): 104	クラスタ(CPU): 101	クラスタ(GPU): 9	大規模共有メモリ: 2
ジョブの特徴	小規模	中～大規模	GPU	大規模メモリ
キューの実行順位	70	50	100	90
ジョブあたりの制限				
最大コア数 (デフォルト)	12 (1)	56 (※2) (1)	64 (1)	144 (18)
最大メモリ (デフォルト)	48GB (4 GB)	980 GB (※2) (4 GB)	980 GB (4 GB)	12 TB (768 GB)
最大経過時間 (デフォルト)	12 h (6 h)	制限なし (2880 h)	制限なし (2880 h)	制限なし (2880 h)
ユーザあたりの制限				
最大ジョブ数 (ソフトリミット)	-	-	4 (1)	8 (4)
合計コア数 (ソフトリミット)	96	500	-	288 (144)
合計メモリ(ソフトリミット)	-	9.8 TB	-	14 TB (6 TB)

(※1) MPIで複数ノードを使用したジョブの場合、1ジョブで500コア・9.8TBまで使用可
(※2) GPUは1ジョブ当たり最大2枚, 1ユーザ当たり最大4枚までという制限があります。

4.3 キューー一覧(続き)

キュー名	cdb
計算サーバ	gds1, gds2
ジョブの特徴	GenomeNet開発
キューの実行順位	50
ジョブあたりの制限	
最大コア数 (デフォルト)	58 (1)
最大メモリ (デフォルト)	980GB (20 GB)
最大経過時間 (デフォルト)	制限なし (720 h)
ユーザあたりの制限	
最大ジョブ数	32
合計コア数	32
合計メモリ	1.3TB

- cdb キューはGenomeNet開発者用のキューです。
- gdsX のホームディレクトリはfe1のホームディレクトリとは異なります(/aptmp/ は共通)。
- PBS(cdbキュー)を使用しないでgdsXで直接実行されたジョブは30分のCPU時間制限があります。
- gdsX の64コア・1TBメモリのうち、cdbキューは合計58コア・980GB まで使用できます。
- 1時間以上計算時間がかかる大規模ジョブを流す必要がある時は、必ず、他の利用者に案内し了解をとって下さい。
- jupyterなどを走らせるときはインタラクティブバッチジョブ(“qsub -I -q cdb …”)を使用して下さい。コアを占有しないために、夜間など利用しない時間帯はジョブを終了して下さい(or -l walltime=... を指定しておく)。

4.4 ジョブスクリプトの例

（参）「使い方と注意事項」→「計算サービス」→「ジョブスクリプトの例」

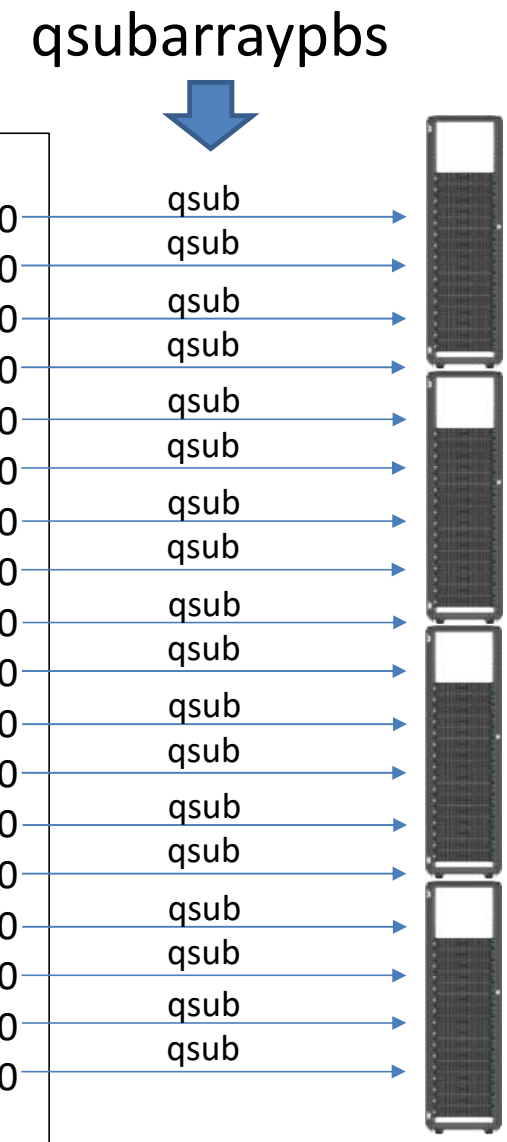
```
#!/bin/sh
#PBS -q APC                                  ←キューを指定
#PBS -l select=1:ncpus=10:mem=40gb           ←使用コア数、メモリサイズを指定
#PBS -N test                                 ←ジョブ名を指定
#PBS -o test.out                             ←標準出力ファイル
#PBS -e test.err                             ←標準エラー出力ファイル

source /etc/profile.d/modules.sh             ← moduleコマンドを使えるようにする
module load blast+/2.15.0                    ← blast+ を読み込み
cd $PBS_O_WORKDIR/                           ← qsubを実行したディレクトリに移動

blastp -db db/nr -query query.fa -out result.out -outfmt 7 -num_threads 10
```

- qsubコマンドのオプションはジョブスクリプト中で
  #PBS xxxx
  で指定することができます。
- 並列化されているプログラムを実行する時は、プログラムのオプションで使用コア数を明示的に指定して下さい。
  （要注意！）diamond, megahit, traitrelax, pigz, juicer

4.5 大規模入力ファイルの処理

（参）「アプリケーション一覧」→「バイオインフォマティクス」→「qsubarraypbs」

```
$ cat blastp.com
blastp -db db/nr -query query01.fa -out result01.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query02.fa -out result02.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query03.fa -out result03.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query04.fa -out result04.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query05.fa -out result05.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query06.fa -out result06.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query07.fa -out result07.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query08.fa -out result08.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query09.fa -out result09.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query10.fa -out result10.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query11.fa -out result11.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query12.fa -out result12.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query13.fa -out result13.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query14.fa -out result14.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query15.fa -out result15.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query16.fa -out result16.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query17.fa -out result17.out -outfmt 7 -num_threads 10
blastp -db db/nr -query query18.fa -out result18.out -outfmt 7 -num_threads 10
…
```



4.5 大規模入力ファイルの処理(続き)

（＊）PBSのコマンドではなく、当スパコンシステム独自のコマンドです
（参）「アプリケーション一覧」→「バイオインフォマティクス」→「qsubarraypbs 」

$ qsubarraypbs(*) [オプション]　（ジョブを列挙したファイル）

- PBSのアレイジョブの機能を使って、ファイルに書かれた多数のジョブを分散実行します。

- オプション（基本的には qsubコマンドと同じ）
  - -q（queue名）　　… SMALL, APC, APG, SDF（必須）
  - -l select=1:ncpus=(1行分のジョブ当たりのコア数):mem=(1行分のジョブ当たりの使用メモリ)
  - --max（同時実行数）
  - その他のqsub オプション（-N, -r, -v, -l, -p, -P, -W）

- ジョブを列挙したファイル
  - 1行につき1ジョブのコマンドを書いて下さい（; や && で区切って1行に複数のコマンドを列挙してもOKです）。
  - コマンドの出力をファイルにリダイレクトする場合は sh/bash 形式で書いて下さい。
    （コマンド） 1>xxx.out 2>xxx.err
  - # から始まる行はコメント行で無視されます（#PBS は機能しません）。

4.5 大規模入力ファイルの処理(続き)

- 補足
  - qsubarraypbsコマンドの実行前に、実行するアプリケーションを module load しておいて下さい。
    - SDF queueを利用する場合

      $ module load SDF
      $ module load (program)
      $ qsubarraypbs -q SDF -l select=1:ncpus=xxx:mem=xxx ….

      # SDF用のbinaryがあるプログラム
      $ module load SDF
      $ module avail　で　/usr/appli/sdf/modulefiles/ にあるもの

  - qsubarraypbsの内部で cd $PBS_O_WORKDIR/ されます。
  - ディレクトリ pbslog/ にエラーメッセージ、エラーになったジョブ等が出力されます。
  - PBSのメール通知機能(-M, -m オプション)は使用できません。
  - (ジョブを列挙したファイル) は10,000行以内に収まるようにして下さい (PBSのアレイジョブの制限)
- 同時実行数の変更

  $ qalter -Wmax_run_subjobs=(同時実行本数) (JobID)

  (例)
  $ qalter -Wmax_run_subjobs=10　2225228[].fe3-adm

4.6.1 分割ジョブの作成: 入力ファイルの分割

- ファイルを指定した行数で分割

  $ wc -l （入力ファイル）　　# 全行数をカウント

  $ split -l (分割行数) -d -a 4 (入力ファイル) (出力ファイル名)

  　　-d -a 4 … (出力ファイル名) に4ケタ数字の連番を付加

- FASTAファイルの分割

  $ fasta_split （FASTAファイル）（分割数）　　# 配列数が均等になるように分割

  $ module load fasta-splitter

  $ fasta-spliltter --n-parts (分割数) (FASTAファイル)　# ファイルサイズが均等になる様に分割

  $ fasta-spliltter --part-size (分割数サイズ) (FASTAファイル)

  $ module load seqkit

  $ seqkit split -O ./ -p (N) (FASTAファイル)　　# 配列数が均等になるようにN分割

  $ seqkit split -O ./ -s (N) (FASTAファイル)　　# N配列毎に分割

  $ seqkit split2 -O ./ -l (N) (FASTAファイル)　　# N残基毎に分割

- FASTQファイルの分割

  $ module load fastq-splitter

  $ fastq-spliltter --n-parts (分割数) (FASTQファイル)　# 分割数(--n-parts) or 分割サイズ(--part-size)を指定

4.6.2 分割ジョブの作成: 連番が付いたジョブの作成

- seqコマンド : 連番の数字を出力するコマンド
  - -w: 0でパディング
- xargs: 標準入力から読み込んだ文字列を {} に代入して
  コマンドを実行(-i オプション)

```
$ seq -w 1 10 | xargs -i echo "command sample{}.pep output{}.txt“ >com.txt
$ cat com.txt
command sample01.pep output01.txt
command sample02.pep output02.txt
command sample03.pep output03.txt
command sample04.pep output04.txt
command sample05.pep output05.txt
command sample06.pep output06.txt
command sample07.pep output07.txt
command sample08.pep output08.txt
command sample09.pep output09.txt
command sample10.pep output10.txt
```

4.6.3 分割ジョブの作成: 入力ファイルのリストからジョブを作成

- parallel: 標準入力から読み込んだ文字列を代入してコマンドを実行 (xargsより高機能)

```
$ find /db/fasta/mgenes/T*.pep | parallel -k --dry-run "{} {.} {/} {//} {/.} {#}"
/db/fasta/mgenes/T30001.pep /db/fasta/mgenes/T30001 T30001.pep /db/fasta/mgenes T30001 1
/db/fasta/mgenes/T30002.pep /db/fasta/mgenes/T30002 T30002.pep /db/fasta/mgenes T30002 2
/db/fasta/mgenes/T30003.pep /db/fasta/mgenes/T30003 T30003.pep /db/fasta/mgenes T30003 3
/db/fasta/mgenes/T30004.pep /db/fasta/mgenes/T30004 T30004.pep /db/fasta/mgenes T30004 4
/db/fasta/mgenes/T30005.pep /db/fasta/mgenes/T30005 T30005.pep /db/fasta/mgenes T30005 5
...
```

{}	{.}	{/}	{//}	{/.}	{#}
そのまま表示	拡張子を取り除いて表示	ファイル名	ディレクトリ名	ファイル名（拡張子なし）	通し番号

以下のコマンドも同じ結果を与えます。

```
$ parallel -k --dry-run "{} {.} {/} {//} {/.} {#}" ::: /db/fasta/mgenes/T*.pep
```

例)

```
$ find /db/fasta/mgenes/T*.pep | parallel -k --dry-run "command {} 1>{/.}.out 2>{/.}.err" >com.txt
$ cat com.txt
command /db/fasta/mgenes/T30001.pep 1>T30001.out 2>T30001.err
command /db/fasta/mgenes/T30002.pep 1>T30002.out 2>T30002.err
command /db/fasta/mgenes/T30003.pep 1>T30003.out 2>T30003.err
…
```

4.7 インタラクティブバッチジョブ

（参）「使い方と注意事項」→「計算サービス」→「ジョブスクリプトの例」

- PBSのインタラクティブバッチジョブの機能を利用すると、計算ノードに
  ログインして対話的に作業を行うことができます。
  - qsub に –I オプション(「I」ntaractive)を付けて下さい。
  - インタラクティブバッチジョブの利用は基本的にはSMALL キュー だけにして下さい
    (ログインしっぱなしになるのを防ぐため)。

```
[appadm@fe1]$ qsub -I -q SMALL -l select=1:ncpus=10:mem=30gb -l walltime=12:00:00
qsub: waiting for job 205274.fe3-adm to start
qsub: job 205274. fe3-adm ready


cd /scratch/pbs_jobdir/pbs.205274.fe3.x8z
[appadm@cs18]$ cd /scratch/pbs_jobdir/pbs.205274. fe3-adm.x8z
[appadm@cs18]$ cd $PBS_O_WORKDIR
```

4.8 ジョブの確認・削除

(参)「使い方と注意事項」→「計算サービス」→「投入ジョブ・キュー情報の参照」
「使い方と注意事項」→「計算サービス」→「ジョブ制御コマンド」

- ジョブ・キューのステータスの確認
  $ qstat [オプション] (Job ID or キュー名)
  - 主なオプション
    - -x # 終了したジョブも表示 (7日前まで)
    - -f [Job ID] # full format
    - -t [Job ID] # アレイジョブのサブジョブ毎に表示
    - -n1 [Job ID] # ジョブが実行されているホスト名を表示
    - -r # 実行中のジョブだけを表示
    - -q, -Q # 全queueのステータスを表示

＊ ジョブのステータス
Q: 実行待ち
R: 実行中
E: 終了処理中
F: 終了したジョブ
H: 保留状態
S: 中断中
B: 実行中のアレイジョブ
X: 終了したアレイジョブのサブジョブ

- ジョブの削除
  $ qdel [Job ID] [Job ID …]

- ジョブの実行の保留・保留解除 (qsubarraypbsで投入したジョブを一時的に止めたい場合など)
  $ qhold [Job ID] # ジョブの実行を保留(ステータスが Q のジョブのみ)
  $ qrls [Job ID] # ジョブの保留を解除

4.9 qstatmyjobsコマンド(＊)

(参)「使い方と注意事項」→「計算サービス」→「投入ジョブ・キュー情報の参照」

- ユーザの使用コア数・メモリ量、キューの利用状況の確認

```
$ qstatmyjobs
User: appadm
                          JOBS            CPUS                    MEM(gb)                 GPUS          WALLTIME(h)
Queue  avail(use%) mysum/max   avail max mysum/max   avail   max mysum/max   avail max mysum/max  default max
------ ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
SMALL   1193( 76%)     0/UNLTD    12  12     0/96       48      48     0/UNLTD     -   -     -/-          6 12
APC     1025( 78%)    19/UNLTD    36  56   380/UNLTD   780     980   1800/UNLTD     -   -     -/-       2880 UNLTD
APG       56( 89%)     0/UNLTD    24  64     0/UNLTD   765     980      0/UNLTD     2   2     0/UNLTD   2880 UNLTD
SDF      305( 69%)     1/8         - 144    72/288       - 12288   2600/UNLTD     -   -     -/-       2880 UNLTD
=============================================================================================================
TOTAL:          JOBS) 20/UNLTD       CPUS) 452/500          MEM) 4400/18432        GPUS)  0/4

fe3-adm:
                                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
14421.fe3-adm   appadm   APC      MS_FRJZR   15345*   1  30  120gb 2880: R 59:15 cl11-adm/0*30
14425.fe3-adm   appadm   APC      MS_FSI7Q   18073*   1  30  120gb 2880: R 59:13 cl12-adm/0*30
15084.fe3-adm   appadm   APC      MS_H5QR8   32758*   1  30  120gb 2880: R 46:13 cs10-adm/0*30
…
```

(＊)PBSのコマンドではなく、当スパコンシステム独自のコマンドです

4.9 qstatmyjobsコマンド(続き)

- 最もメモリが空いているノードを探す(-m オプション)

```
$ qstatmyjobs -m
User: appadm
                        JOBS            CPUS                    MEM(gb)                 GPUS          WALLTIME(h)
Queue  avail(use%) mysum/max   avail max mysum/max   avail   max mysum/max   avail max mysum/max  default max
------ ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
SMALL   1193( 76%)     0/UNLTD    12  12     0/96       48    48     0/UNLTD     -   -     -/-          6 12
APC     1025( 78%)    19/UNLTD    12  56   380/UNLTD   880   980  1800/UNLTD     -   -     -/-       2880 UNLTD
APG       56( 89%)     0/UNLTD    24  64     0/UNLTD   765   980     0/UNLTD     2   2     0/UNLTD   2880 UNLTD
SDF      305( 69%)     1/8         - 144    72/288       - 12288  2600/UNLTD     -   -     -/-       2880 UNLTD
================================================================================================================
TOTAL:          JOBS) 20/UNLTD       CPUS) 452/500          MEM) 4400/18432        GPUS)  0/4

fe3-adm:
                                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
14421.fe3-adm   appadm   APC      MS_FRJZR   15345*   1  30  120gb 2880: R 59:15 cl11-adm/0*30
14425.fe3-adm   appadm   APC      MS_FSI7Q   18073*   1  30  120gb 2880: R 59:13 cl12-adm/0*30
15084.fe3-adm   appadm   APC      MS_H5QR8   32758*   1  30  120gb 2880: R 46:13 cs10-adm/0*30
...
```

4.9 qstatmyjobsコマンド(続き)

項目	説明
Queue	キュー名
avail	キューが利用可能な全コア数のうち、利用可能（ジョブが実行されていない）コア数
(use%)	キューが利用可能な全コア数のうち、使用中（ジョブが実行中）のコア数の割合(%)
JOBS: mysum	ユーザが実行中のジョブの総数
JOBS: max	ユーザが実行可能な最大ジョブ数
CPUS: avail	ジョブがすぐに実行開始できる最大コア数
CPUS: max	そのキューで指定可能な最大コア数
CPUS: mysum	ユーザが実行中のジョブの合計コア数
CPUS: max	ユーザあたり、実行可能なジョブの最大合計コア数
MEM(gb): avail	ジョブの最大コア数指定時に、すぐに実行開始可能な最大メモリサイズ。なお、もしジョブの最大コア数よりも少ないコア数を指定すると、より多くのメモリを指定しても、そのジョブがすぐに実行開始されることがあります。
MEM(gb): max	そのキューで指定可能な最大メモリサイズ
MEM(gb): mysum	ユーザが実行中のジョブの合計メモリサイズ
MEM(gb): max	ユーザあたり、実行可能なジョブの最大合計メモリサイズ
GPUS: avail	ジョブがすぐに実行開始できる最大GPU数
GPUS: max	そのキューで指定可能な最大GPU数
GPUS: mysum	ユーザが実行中のジョブの合計GPU数
GPUS: max	ユーザあたり、実行可能なジョブの最大合計GPU数
WALLTIME(h): default	何も指定しない場合に設定される経過時間
WALLTIME(h): max	指定可能な最大経過時間
fe3-adm: 以下	実行中のユーザジョブ情報(qstat -fnrt1) 左からジョブID、ユーザ名、バッチキュー、ジョブ名、セッションID、chunk数、コア数、メモリ、最大経過時間、ステータス、経過時間、実行ノード

4.10 終了したジョブの実行情報の確認

(参)「使い方と注意事項」→「計算サービス」→「投入ジョブ・キュー情報の参照」

$ tracejob [オプション] [Job ID]

- 主なオプション
  -n（日数） … 何日遡ってログファイルを確認するか

```
$ tracejob -n 2 191784.fe3-adm

Job: 191784.fe3-adm
…
04/19/2024 00:46:35  S    Exit_status=0 resources_used.cpupercent=149 resources_used.cput=00:08:18
                          resources_used.mem=39432kb resources_used.ncpus=4 resources_used.vmem=416888kb
                          resources_used.walltime=00:05:38
```

Exit_status	ジョブの終了ステータス（0でない場合はエラーが発生しています。Exit_status=-27 はメモリ超過でPBSにkillされたことを表します。）
resources_used.cpupercent	平均CPU使用率
resources_used.mem	使用メモリ量
resources_used.ncpus	qsub時に指定したncpus
resources_used.walltime	ジョブの実行時間

(注) アレイジョブの場合は、後述の PbsExitStatus コマンドを使用して下さい。

4.10 終了したジョブの実行情報の確認(続き)

(参)「アプリケーション一覧」→「バイオインフォマティクス」→「PbsExitStatus」

- (アレイ)ジョブの実行情報の確認
  $ PbsExitStatus（*） [オプション] [Job ID]
  - -e ： Exit_status=0 でないもの(エラーが発生しているもの)だけを表示
  - -m ：使用メモリ(resource_used.mem) が最大のものを表示
  - -c ：出力行数をカウント
  - -t ：walltimeの合計を出力

例)

```
$ PbsExitStatus -e 213483[].fe3-adm
…
20240502:05/02/2024 13:22:44;0010;Server@fe3-adm;Job;213483[3583].fe3-adm;Exit_status=271
resources_used.cpupercent=99 resources_used.cput=00:17:02 resources_used.mem=33733660kb
resources_used.ncpus=2 resources_used.vmem=34207856kb resources_used.walltime=00:17:17
```

（＊）PBSのコマンドではなく、当スパコンシステム独自のコマンドです

4.11 SSDの利用 (SDFキュー限定)

- PBSでジョブを投入すると、ジョブの開始時に一時ディレクトリ
  /scratch/pbs_tmpdir/(JobID)/
  が自動的に作成されます。
  このディレクトリは ${TMPDIR}/ で利用することができます。

/scratch/ … SSD (sdf1, sdf2)
… Lustreファイルシステム
(PCクラスター)

- ジョブが終了すると ${TMPDIR}/ は自動的に削除されます。

- デノボアセンブリングツールなどIO負荷の高いアプリケーションを SDFキューで実行する際は、
  一時ディレクトリやファイルの出力先に ${TMPDIR} を指定して下さい。

- 使用例1) コマンドの一時ディレクトリを指定するオプションで ${TMPDIR}/を指定
  - spades --tmp-dir=${TMPDIR} …
  - megahit --tmp-dir ${TMPDIR} …
- 使用例2) コマンドの出力先に ${TMPDIR}/ を指定、もしくは ${TMPDIR}/ の中で計算を実行
  - mkdir ${TMPDIR}/output/
    flye -o ${TMPDIR}/output/ …
    mv ${TMPDIR}/output/ /aptmp/xxx/ # 計算結果を /aptmp/ に移動

5. Appendix

5.1 condaコマンド利用時の注意(1/2)

- Anaconda, Miniconda コマンドを使用してユーザ様ご自身で各種アプリケーションをインストールすること自体は問題ありませんが、”conda init”を実行した際に、以下の様な設定が ~/.bashrc ファイルに追記される場合があります。

```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/usr/appli/freeware/miniconda/3.6/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/usr/appli/freeware/miniconda/3.6/etc/profile.d/conda.sh" ]; then
        . "/usr/appli/freeware/miniconda/3.6/etc/profile.d/conda.sh"
    else
        export PATH="/usr/appli/freeware/miniconda/3.6/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
```

- この設定は、スーパーコンピュータ環境にインストールされているアプリケーションと競合し、予期しないエラーが発生する場合があります。

- つきましては、次項の手順に従い、conda の設定を ~/.bashrc ではなく、別のファイル（例: ~/.bashrc.conda）に記載して頂くようお願い申し上げます。

5.1 condaコマンド利用時の注意(2/2)

1. ~/.bashrc の設定内容を確認する

   ~/.bashrc の condaの設定部分（# >>> conda initialize >>> から # <<< conda initialize <<< の間の内容）を切り取り、~/.bashrc.conda などの別ファイルに貼り付けて保存してください。

2. ~/.bashrc.conda を使用して conda の環境を有効化する

   conda コマンドを使用する必要がある場合は、下記コマンドを実行して設定を読み込んでください。
   $ source ~/.bashrc.conda