プログラミング講習会
（最適化・並列化プログラミング含む）

京都大学化学研究所スーパーコンピュータシステム

2025年7月3日

福本　淳司

Hewlett Packard Enterprise

© Copyright 2019 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice.

Contents

10:00 – 10:30

1. システム構成
   - ◆システムの概要
   - ◆利用負担金
   - ◆アーキテクチャ
   - ◆計算サーバ構成概要
   - ◆開発環境および環境設定
2. コンパイルと実行

10:30 – 11:30

3. ジョブスケジューラ：PBS

11:30 – 12:00

4. 最適化・並列化手法
   - ◆最適化・並列化手順の概要
   - ◆最適化）コンパイラオプション
   - ◆並列化）アムダールの法則
   - ◆並列化）OpenMPおよびMPIの特徴

午前中の内容を踏まえて
計算化学アプリを実行してみよう

13:00 – 14:20

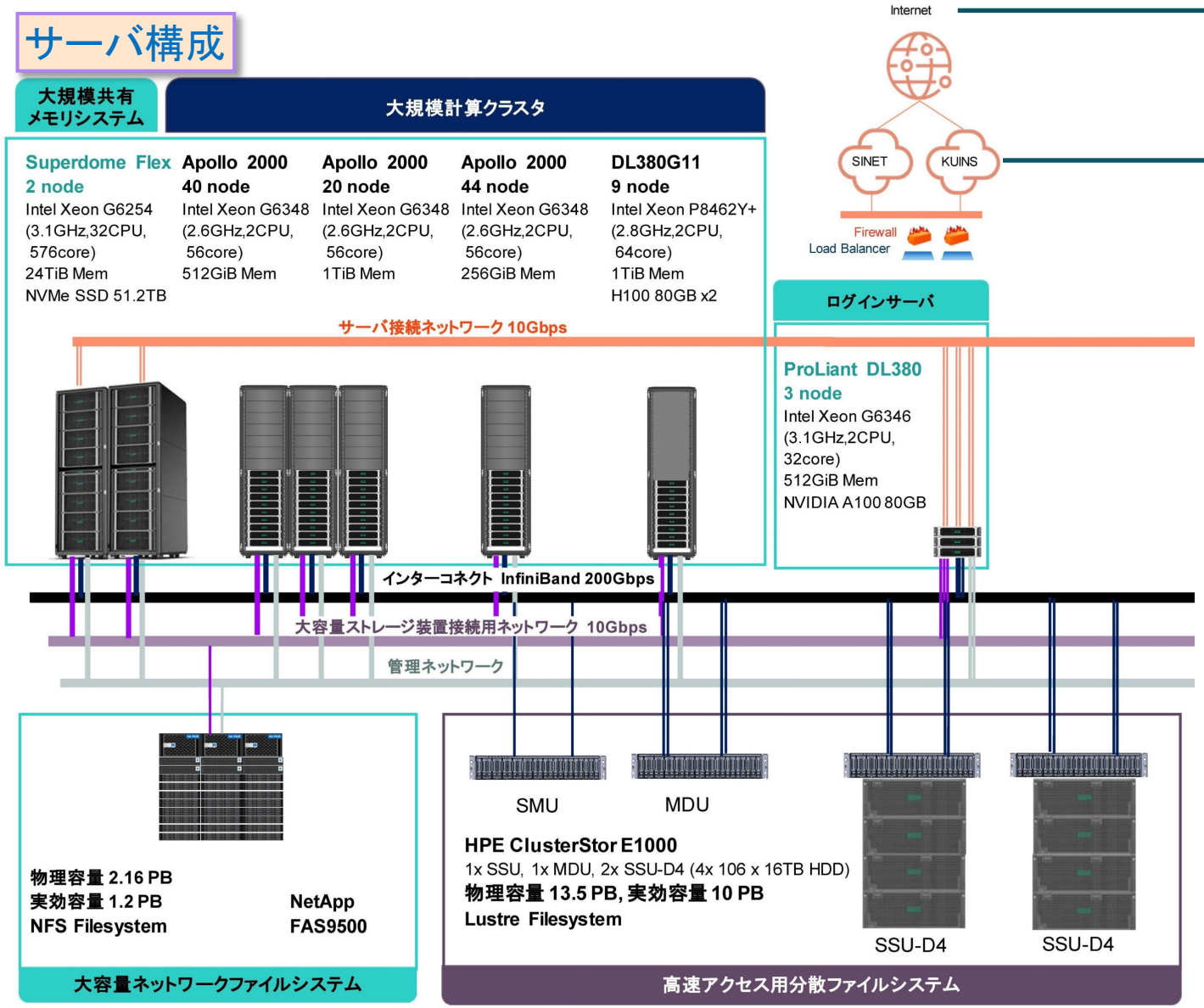
5. 計算化学アプリケーション
   - ◆アプリケーションの概要
   - ◆ジョブ実行方法（Gaussianの場合）
   - ◆ジョブ実行方法（LAMMPSの場合）

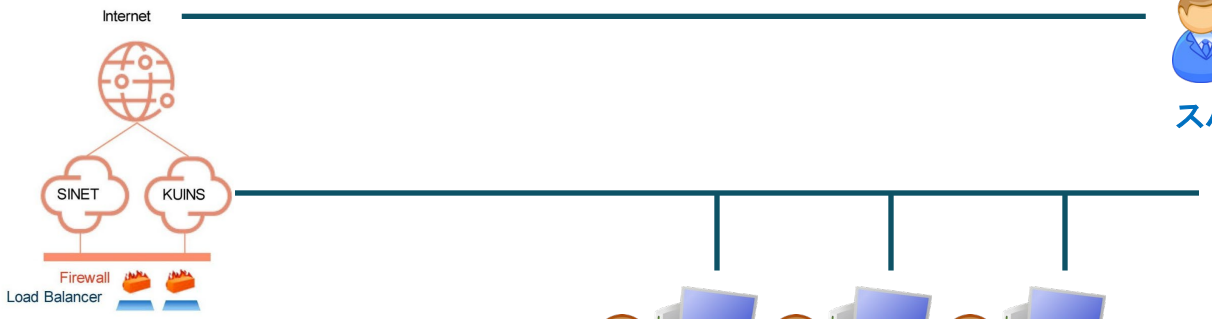
講習会の内容を一部変更

目的
- 最適化・並列化の手法を知る
- 必ずしも並列数を増やしても高速化できるとは限らない
- バッチスクリプトでの並列化の指定方法を知り、アプリケーション使用時に活用する

1. システム構成 > 京都大学化学研究所スーパーコンピュータシステム

サーバ構成





スパコンユーザ（学外）

スパコンユーザ（学内）

スパコンシステムの特徴

- 計算化学を中心とした豊富なアプリケーション
- 200を超えるバイオツールとバイオデータベース
- 一定時間までは従量課金、それ以降は定額となる料金体系
- 全ユーザ：300名弱（うち計算サーバ利用者：150名前後）
- 京都大学はもちろん、他の学術機関、民間、個人利用も可能

スパコンシステムホームページ： https://www.scl.kyoto-u.ac.jp
新規利用申請： https://www.scl.kyoto-u.ac.jp/Apply_new/apply_new_shinsei.html
システム紹介： https://www.scl.kyoto-u.ac.jp/Servers/intro.html
アプリケーション： https://www.scl.kyoto-u.ac.jp/Appli/
利用負担金：https://www.scl.kyoto-u.ac.jp/Apply/apply_futankin.html

1. システム構成 > 利用負担金

課金単価表

システムのサービスにかかる費用を以下にお知らせします。

課金対象	計算ルール
基本料金	1,000円/月
計算サーバ(CPU)[注1,2]	0.00220円/秒
CPU時間に対する課金の上限額[注3]	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
ディスク基本料金[注4] (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[注5,6] (最大900GBまで追加可能)
クラウドストレージ拡張サービス	+300GB 1,000円/年度[注5,6] (最大900GBまで追加可能)
MS/DS利用サービス[注3,注7]	支払責任者の所属 京都大学化学研究所：1,500円/月 上記以外：3,000円/月

※ 1GB = 1024 MB です。

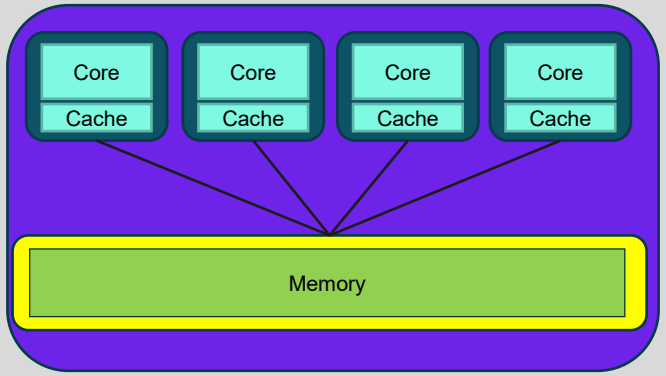
8core使用するジョブを1週間実行した場合、
CPUに対する課金は1万円程度
（0.0022円/秒 * 1週間 * 8core=10,644円）

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。
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。
3. 支払責任者の所属によってCPU時間に対する課金の上限額およびMS/DSの利用負担金が変わります。
4. ディスク基本料金（ホーム領域）は、ホーム領域のみに対する課金です。メールスプール領域や一時領域(/aptmp) は含まれません。
5. ディスク拡張サービスおよびクラウドストレージ拡張サービスは、本サービス申請月に当該料金が発生します。年度途中で拡張サービスの利用を取りやめた場合でも、当該料金の一部もしくは全部の返還はできません。ご了承ください。
6. ディスク拡張サービスおよびクラウドストレージ拡張サービスは年度を超えることはできません。翌年度も継続したい場合には、継続利用申請書にその旨を記載してください。その場合、4月に当該料金が発生します。
7. MS/DSとは、Materials Studio/Discovery Studioです。どちらか一方しか利用しない場合でも料金は変わりません。
8. 支払責任者が京都大学化学研究所の所属でない方であっても、京都大学化学研究所の国際共同利用・共同研究課題公募にて課題が採択された方は 支払責任者が京都大学化学研究所所属の場合と同等の課金体系が適用されます。
9. 作成したアカウントの共有利用はできません。利用者ごとにアカウントの取得をお願いします。

1. システム構成 > 並列計算機アーキテクチャ

現在のスパコンシステム

SMP：共有メモリ型
(Symmetric Multi-Processing)

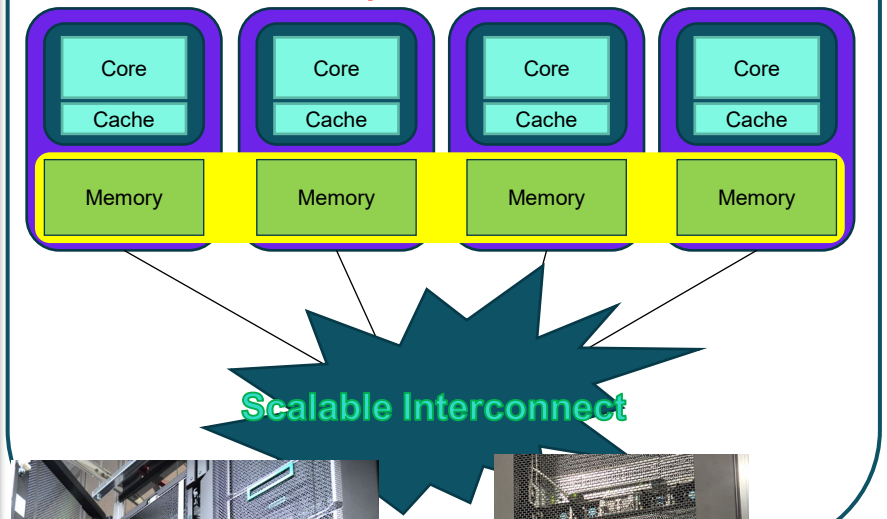


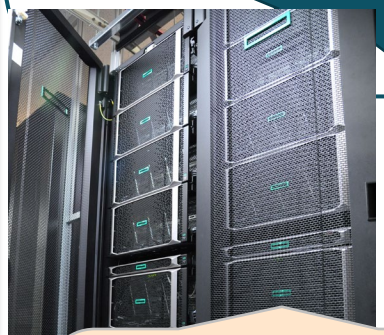
30年ほど前に設置されていたスパコン

【特徴】
- 1台の巨大なコンピュータ
- 計算リソース拡大は困難
- 単一の共有メモリ

ccNUMA：分散共有メモリ型
（Cache Coherent Non-Uniform Memory Access）

HPE Superdome Flex

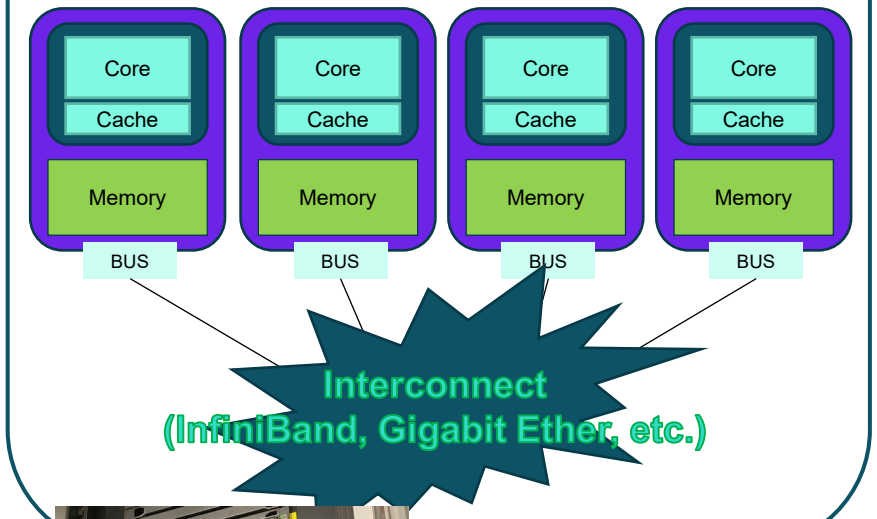


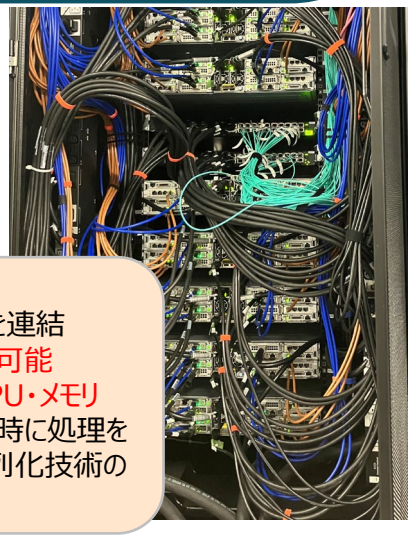


【特徴】
- CPU・メモリの単位を連結したコンピュータ
- 比較的容易に計算リソース拡張が可能
- OSは単一
- ローカルにメモリを持つが論理的な共有が可能
- キャッシュ整合性(CC)：他のCPUによる読み書き操作を常に監視。他のCPUが該当アドレスに書き込みを行ったら、自分のキャッシュのコピーを無効化。

Cluster：分散メモリ型

HPE Apollo 2000





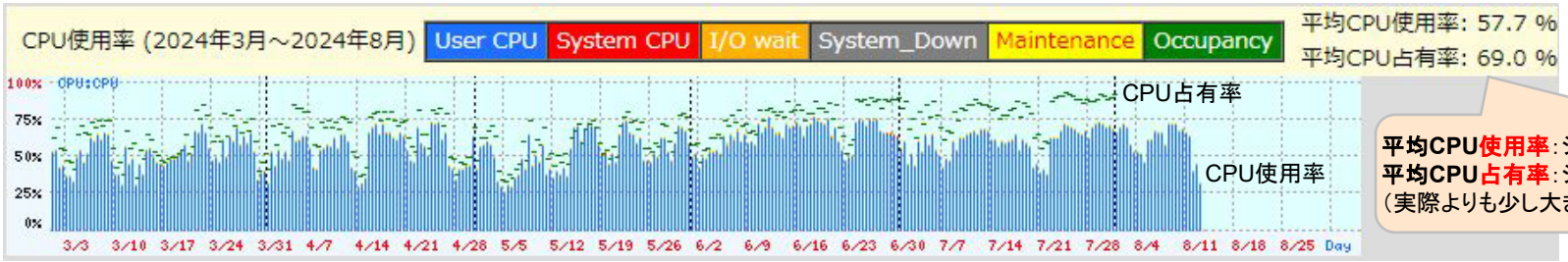
【特徴】
- 複数の小規模コンピュータを連結
- 容易に計算リソース拡大が可能
- それぞれが独立したOS・CPU・メモリ
- 連結されたコンピュータが同時に処理を行うにはMPIと呼ばれる並列化技術の利用が不可欠

1. システム構成 > 計算サーバ 構成概要

大規模共有メモリシステム

大規模メモリを使ったジョブを実行したい場合に適した計算サーバ

システム	HPE Superdome Flex （SDF）
ノード数	2ノード
CPU	Intel Xeon Gold 6254
CPUクロック周波数	3.1GHz
ノード当りのソケット数	32ソケット
ノード当りのコア数	576コア
ノード当りのメモリ容量	24TB
コア当たりのメモリ容量	**42GB**
OS	Red Hat Enterprise Linux 8.7
ジョブ当たりの最大使用可能メモリ	**12TB**



平均CPU使用率：ジョブがCPUリソースを**使用した割合**
平均CPU占有率：ジョブ実行時にCPUリソースを**確保した割合**
（実際よりも少し大きめに確保）

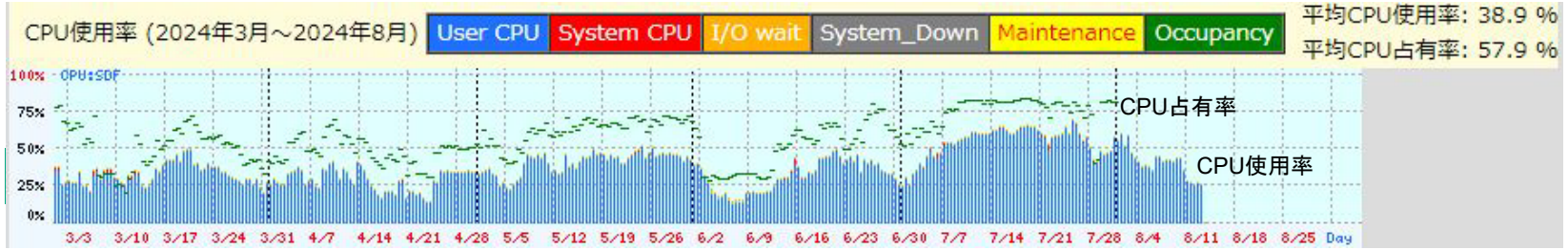
1. システム構成 > 計算サーバ 構成概要

大規模計算クラスタ

SDFと比べてノード当たりのコア数やメモリサイズは小さいため、小規模～中規模のジョブ実行に適したシステム

（SDFとは異なるアーキテクチャであるため、バイナリに完全な互換性がないことに注意）

システム	HPE Apollo 2000			HPE DL380G11
	CPUノード			GPUノード
ノード数	44	40	20	9
CPU	Intel Xeon Gold 6248			Intel Xeon Platinum 8462Y+
CPUクロック周波数	2.6GHz			2.8GHz
ノード当りのソケット数	2ソケット			2ソケット
ノード当りのコア数	56コア			64コア
ノード当りのメモリ容量	256GB	512GB	1,024GB	1,024GB
コア当りのメモリ容量	4.6GB	9.2GB	18.3GB	16.0GB
ノード当りのGPU	-			NVIDIA H100 80GB x 2
OS	Red Hat Enterprise Linux 8.7			



ノード当たりのメモリ容量は異なるが、バッチスクリプト内で必要なメモリ容量を指定することでジョブスケジューラが自動的にどの計算ノードでジョブを実行するべきか判断する。

1. システム構成 > 開発環境

大規模計算クラスタにおける開発環境は以下の通りです。大規模共有メモリシステムにおいてもGPUを除く環境はほぼ同様ですが、詳しくは module コマンドを使って、利用できるバージョンについてご確認ください。
また、大規模計算クラスタと大規模共有メモリシステムにおいては**アーキテクチャが異なる点にご注意**ください。

OS	Red Hat Enterprise Linux 8.7
C/C++/Fortran コンパイラ	Intel oneAPI Compiler 2022以降 GNUコンパイラ 4.7以降
数値演算ライブラリ	Intel MKL 2022以降 OpenBLAS 0.3.6, 0.3.17, 0.3.25, 0.3.26 FFTW 3.3.10 scaLAPACK 2.2.2
MPI	HPE MPI 2.20 Intel MPI 2021以降 MPICH 3.2, 3.3, 4.2.2 Open MPI 1.10.7, 2.1.6, 3.0.5, 3.1.5, .0.3, 4.1.6. 5.0.3 MVAPICH2 2.3.7-1 MVAPICH 3.0
GPU （利用できるGPUはNVIDIA H100）	CUDA 10.0以降 NVIDIA HPC SDK 20.9以降
性能解析ツール	Intel VTune 2022以降

1. システム構成 > 環境設定コマンド(moduleコマンド)

(参照)「アプリケーション一覧」→「module コマンド」
https://www.scl.kyoto-u.ac.jp/Appli/module.html

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(注:大文字・小文字は区別されます)。
$ 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。アプリケーションの実行環境をリセットしたい場合に使用。

バッチスクリプトの中で module コマンドを使用する際は以下を含めてください。

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)

演習　1. module コマンド

ゴール：　module コマンドを使ってコンパイラもしくはアプリケーションの環境設定の手順を学ぶ

2. コンパイルと実行

2. コンパイルと実行 > コンパイラコマンド

DPC++（Data Parallel C++）コンパイラ：
データ並列プログラミングをサポートし、
C++をベースに開発されたコンパイラ

言語	GNUコンパイラ	Intelコンパイラ	
		Classic	DPC++
C	gcc	icc	icx
C++	g++	icpc	icpx
Fortran	gfortran	ifort	ifx
		従来のIntel C/C++コンパイラで、高い信頼性とIntelプロセッサ向けの最適化が特徴。既存のプロジェクトでの使用に適している。Intel oneAPI 2025以降は提供されず。	Intelが新たに開発した次世代のC/C++コンパイラ。**LLVMベースの技術を使用**

コンパイラオプション

- 一覧表示： –help
- バージョン情報：-V

LLVM（Low Level Virtual Machine）とは

- 本来 C と C++ 言語向けに開発されたフリーの**オープンソース・コンパイラー基盤**です。さまざまなプログラミング言語のコンパイラやツールを構築するための基盤を提供。
- Intel oneAPIの他、NVIDIA CUDA Compiler, AMD AOCC, Clang等で採用

LLVMを利用することのメリット

- コードを**高いレベルで最適化**します。これにはループ最適化、インライン展開、デッドコード削除などが含まれ、実行時のパフォーマンスを向上
- データ並列プログラミングに特化した**新しい構文と機能**を提供
- Classicコンパイラでコンパイル可能なソースコードの場合であってもDPC++の並列実行モデルを活用できる場合、DPC++コンパイラが並列処理を効果的に最適化できる可能性あり

コンパイラのマニュアル

以下のURLからインテルコンパイラのマニュアルをご参照ください。
http://www.scl.kyoto-u.ac.jp/Appli/intel.html

2. コンパイルと実行 > Intel oneAPI コンパイラ (Fortran)

Classicコンパイラ（ifort, mpiifort, etc.) / DPC++コンパイラ(ifx, mpiifx, etc.)

-O3: コンパイラの最適化オプションの1つ。高度な最適化により、プログラムの実行速度を向上させるための設定です。

実際の利用時には、コマンド実行前にmoduleコマンドでの環境設定が必要です。

- シリアルプログラム

```
$ ifort –O3 prog.f     /     ifx –O3 prog.f                    (コンパイル)
$ ./a.out                                                      (実行)
```

- OpenMPプログラム

```
$ ifort -O3 -qopenmp prog_omp.f    /     ifx -O3 -qopenmp prog_omp.f      (コンパイル)
$ setenv OMP_NUM_THREADS 4                                                (スレッド並列数の設定)
$ ./a.out                                                                 (実行)
```

スレッド並列数やMPIプロセス数を増やすことでより多くのコアを使って、プログラムの実行速度を向上させることができます。

- MPIプログラム

```
$ mpiifort -O3 prog_mpi.f    /       mpiifx -O3 prog_mpi.f      (コンパイル)
$ mpirun -np 4 ./a.out                                          (MPIプロセス数の指定および実行)
```

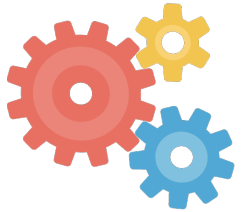
- ハイブリッド(MPI+OpenMP)プログラム

```
$ mpiifort -O3 -qopenmp prog_hyb.f     /    mpiifx -O3 -qopenmp prog_hyb.f    (コンパイル)
$ setenv OMP_NUM_THREADS 8                                                    (スレッド並列数の設定)
$ mpirun -np 4 ./a.out                                                        (MPIプロセス数の指定および実行)
（または）
$ mpirun –np 4 –genv OMP_NUM_THREADS=8 ./a.out                                (スレッド並列数とMPIプロセス数の指定および実行)
```

必要なコア数は、スレッド並列数 x MPIプロセス数であるため、この場合は32コアが必要です。

演習　2. コンパイルと実行

ゴール：　すでに用意されたFortranのソースコードをコンパイルし、ジョブの実行手順を確認する。
また、使用するコア数や並列化手法を変えてジョブを実行する手順も併せて確認する。



3. ジョブスケジューラ：PBS

PBS：Portable Batch System

ジョブスケジューラとは、多くの利用者が共有する計算資源を効率的かつ公平に
利用ができるよう、ジョブの投入・待機・実行・終了などを制御するシステム

3. PBS > PBSとは

ジョブスケジューラのイメージ図

ジョブスケジューラとは:

- 自分が行いたい計算の内容を記述しジョブを投入すれば、**スケジューラが空いている計算リソースを自動的に見つけて実行してくれる**
- 多くのユーザが利用する場合、**ある一部のユーザが計算リソースを占有することが無いよう**、計算リソースを公平に管理することができる

限られた計算リソースを公平かつ効率よく利用するためのジョブ管理ソフトウェア

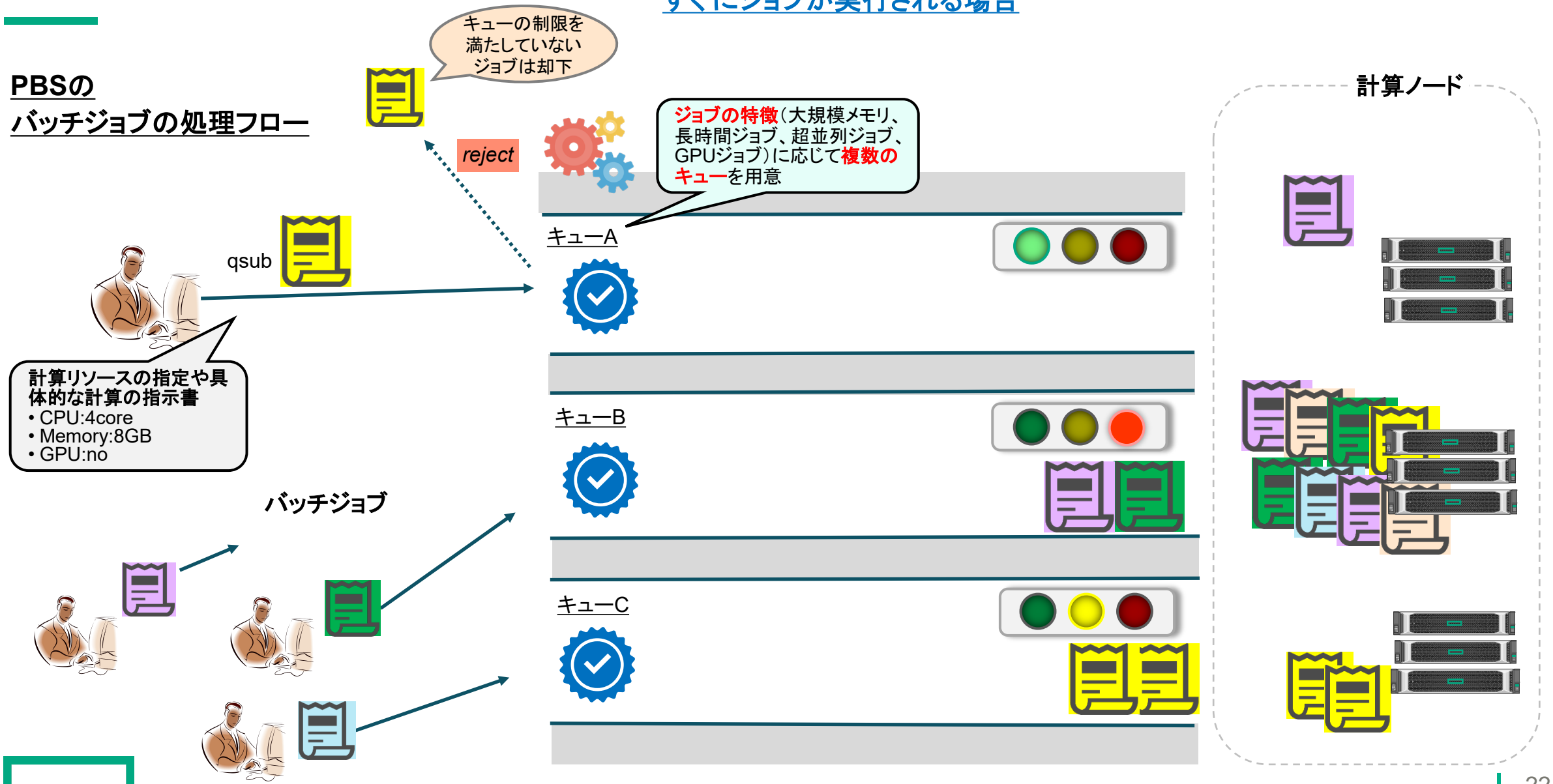
ジョブスケジューラには、PBS, Torque, Slurm, Grid Engine, LSF 等様々なソフトウェアがあります。スパコンシステムではアルテアエンジニアリング社によるPBSを採用しています。

バッチジョブ：バッチ処理（大量のジョブを一括で処理すること）によって実行されるジョブ

キュー：実行待ちのジョブを順番に管理するための待ち行列のこと。用途や必要なリソースに応じて複数のキューが設けられることもあります。

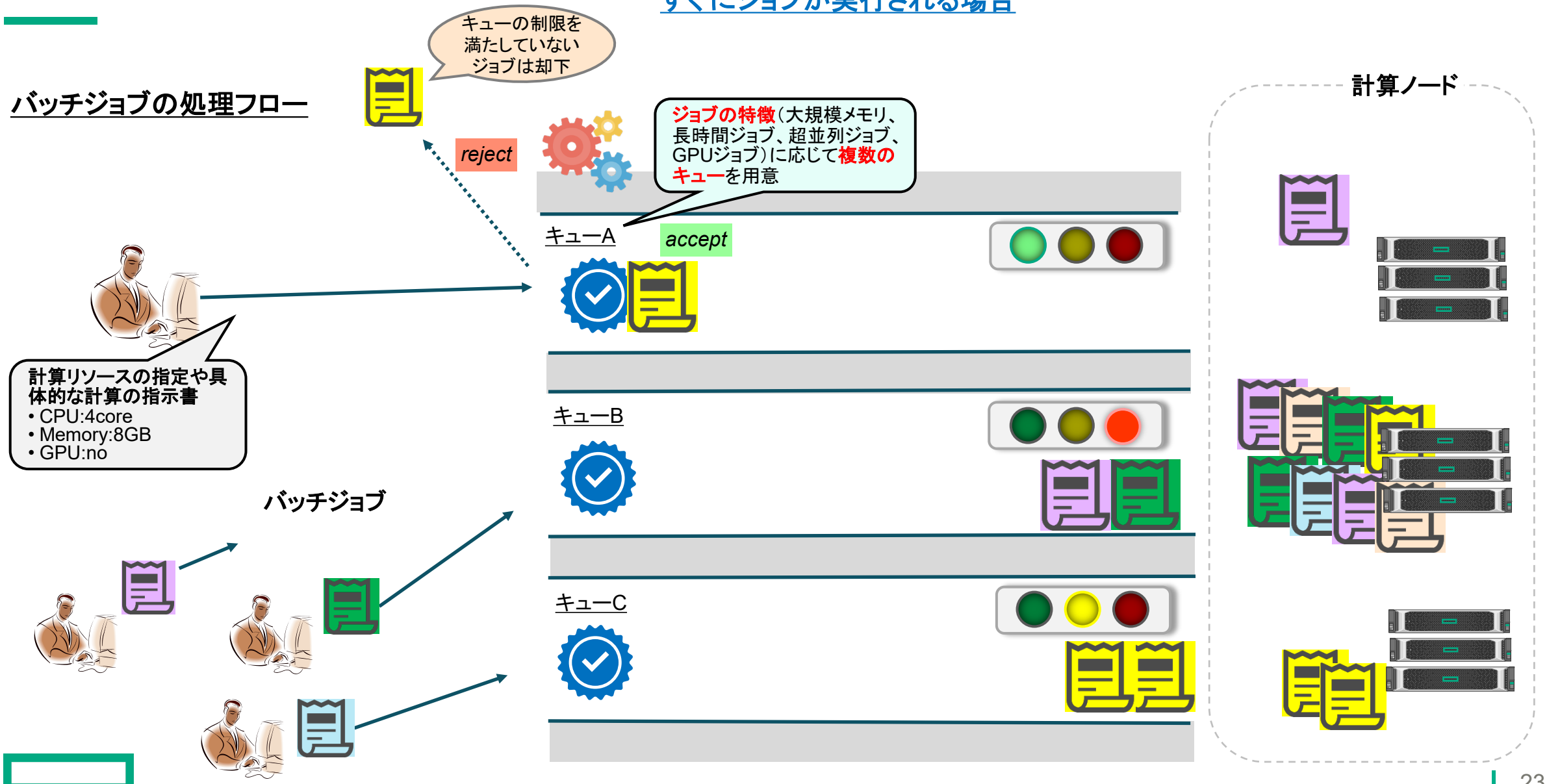
3. PBS > PBSとは

PBSの
バッチジョブの処理フロー



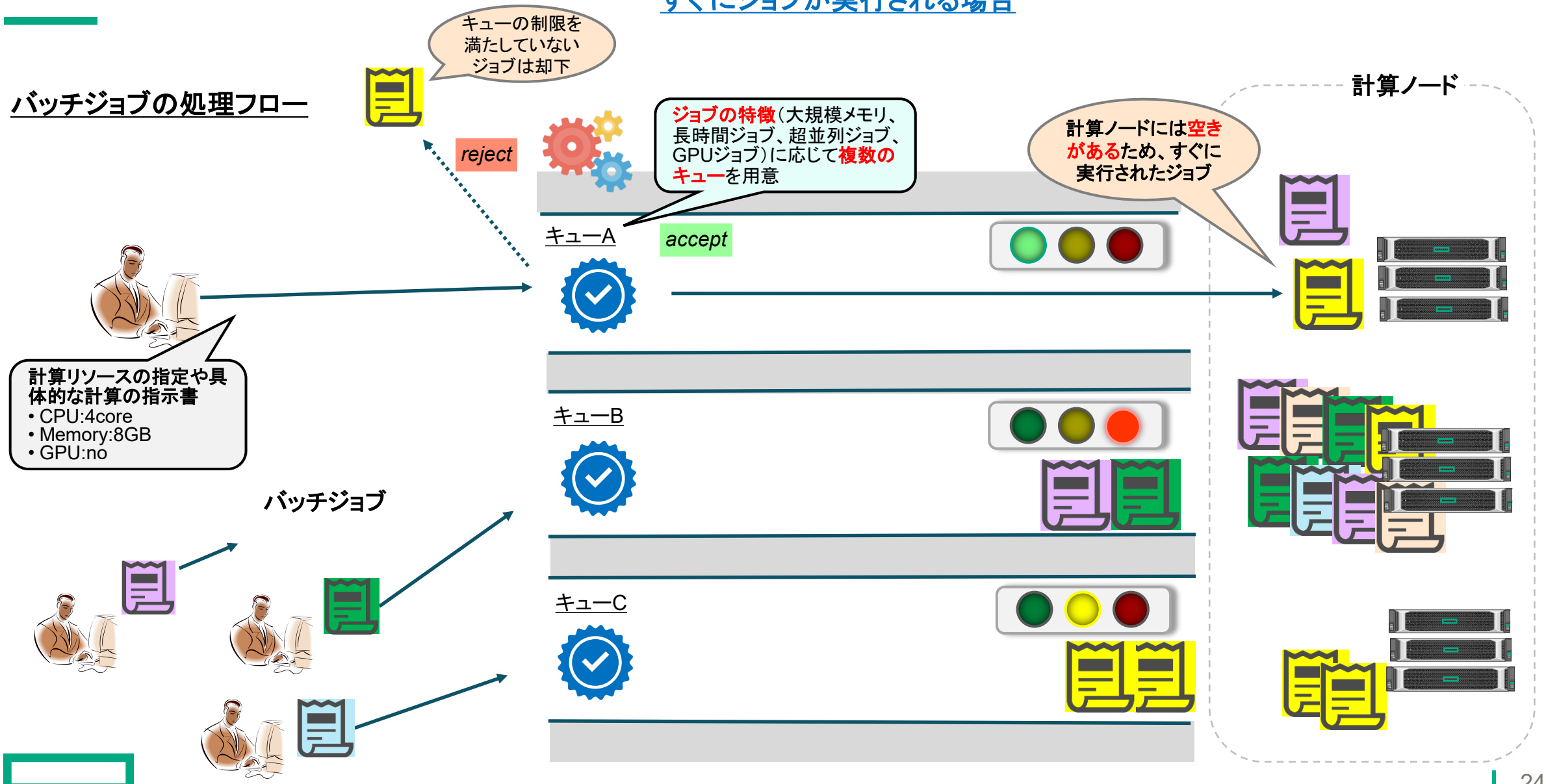
3. PBS > PBSとは

バッチジョブの処理フロー



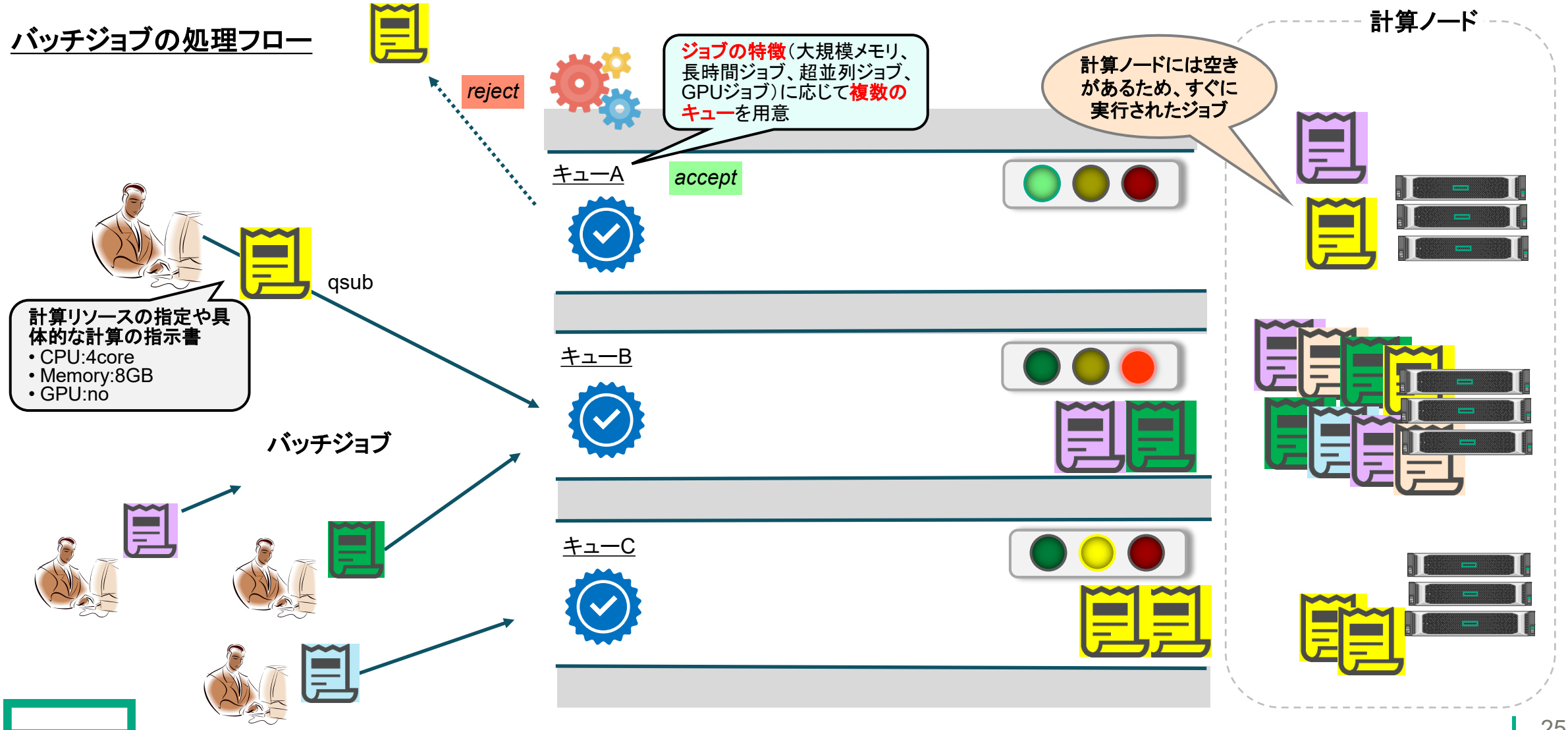
3. PBS > PBSとは

バッチジョブの処理フロー



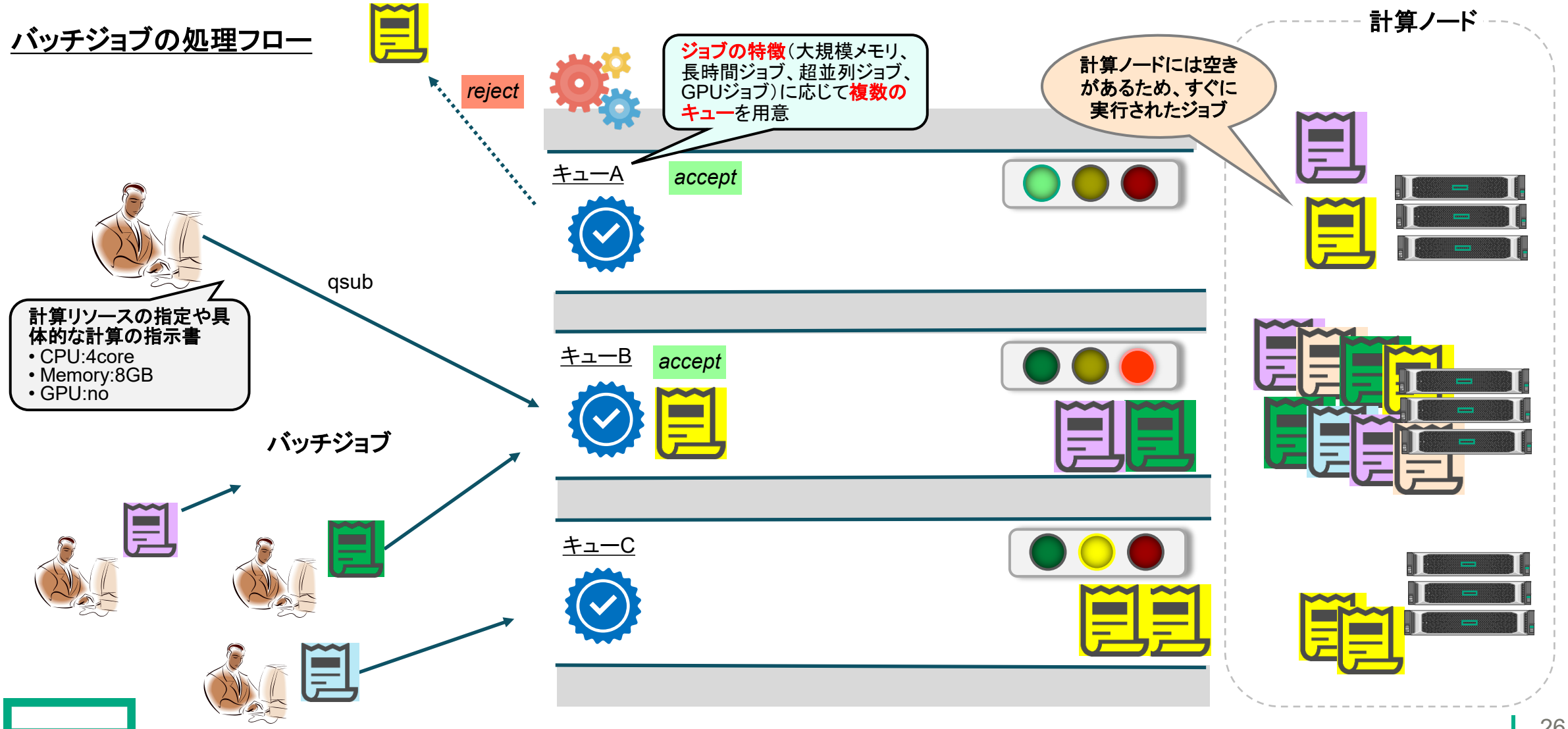
3. PBS > PBSとは

計算リソース不足の場合



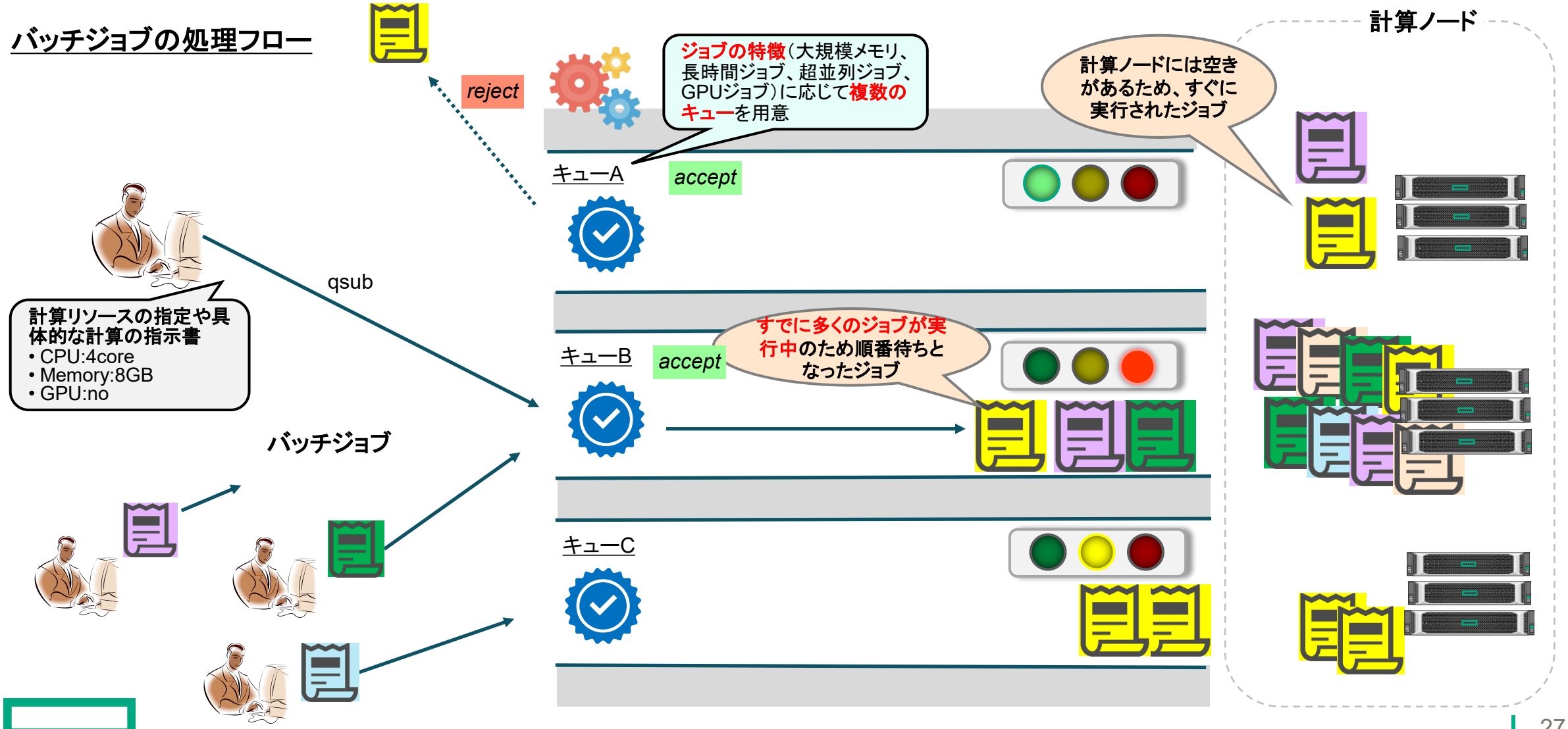
3. PBS > PBSとは

計算リソース不足の場合



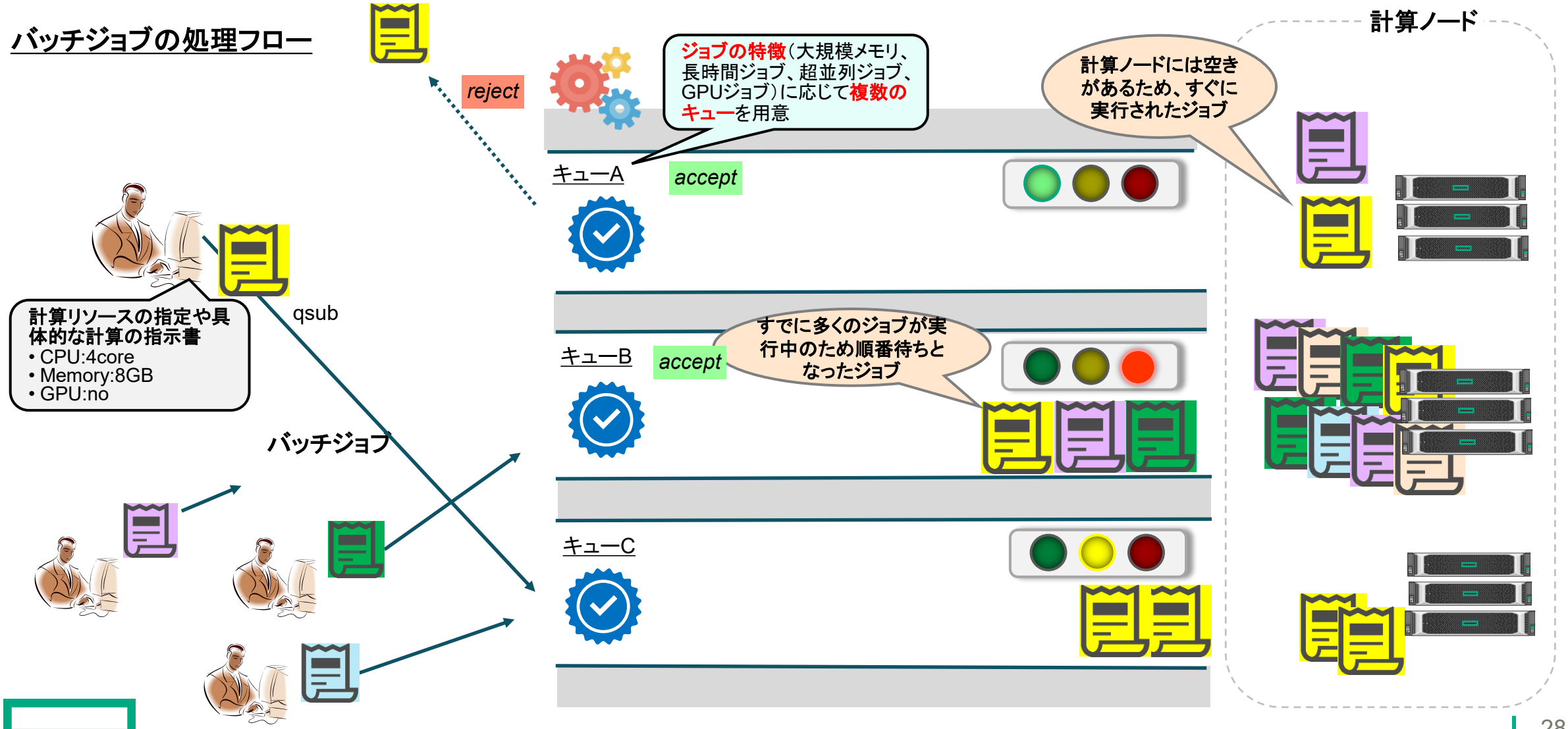
3. PBS > PBSとは

計算リソース不足の場合



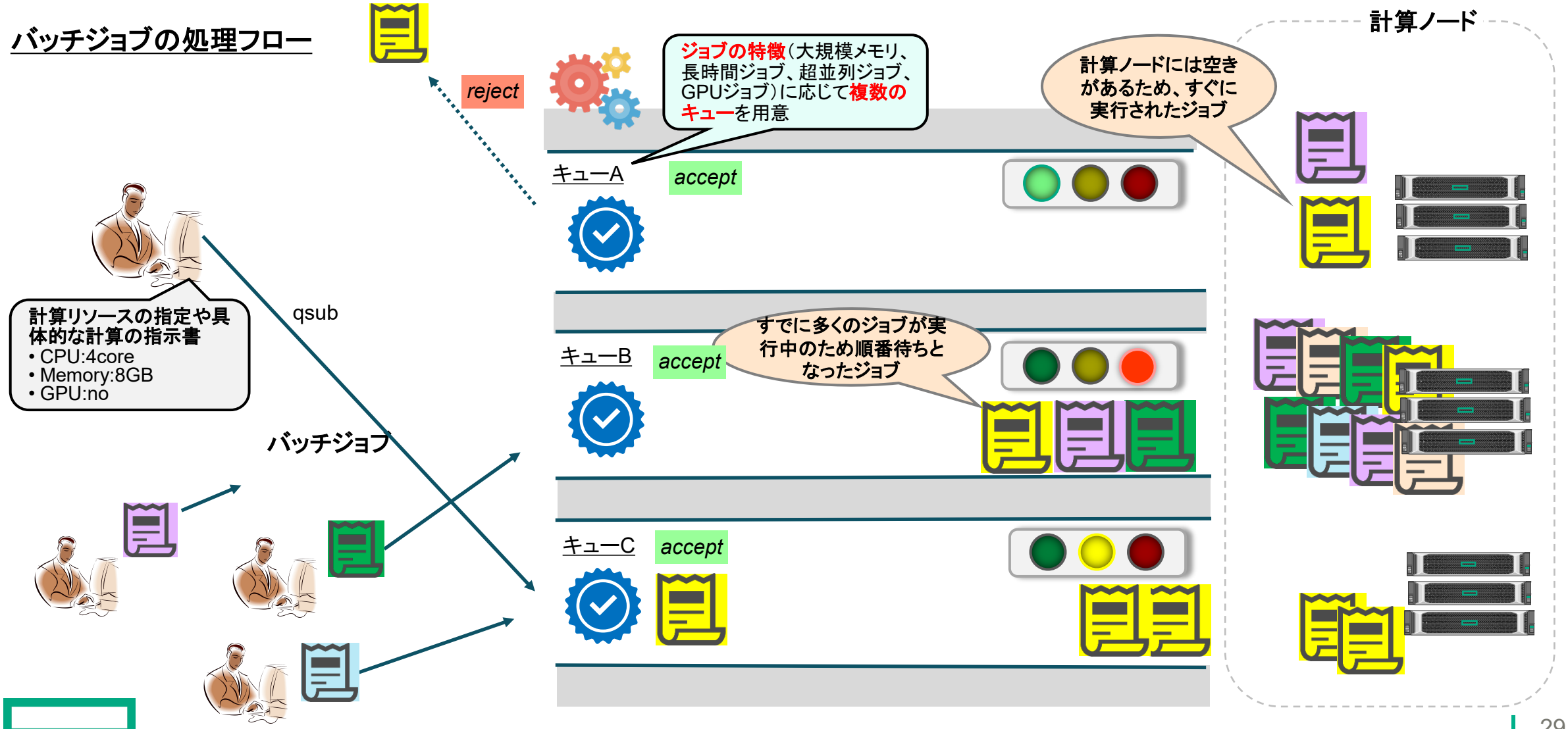
3. PBS > PBSとは

割り当てリソース不足の場合



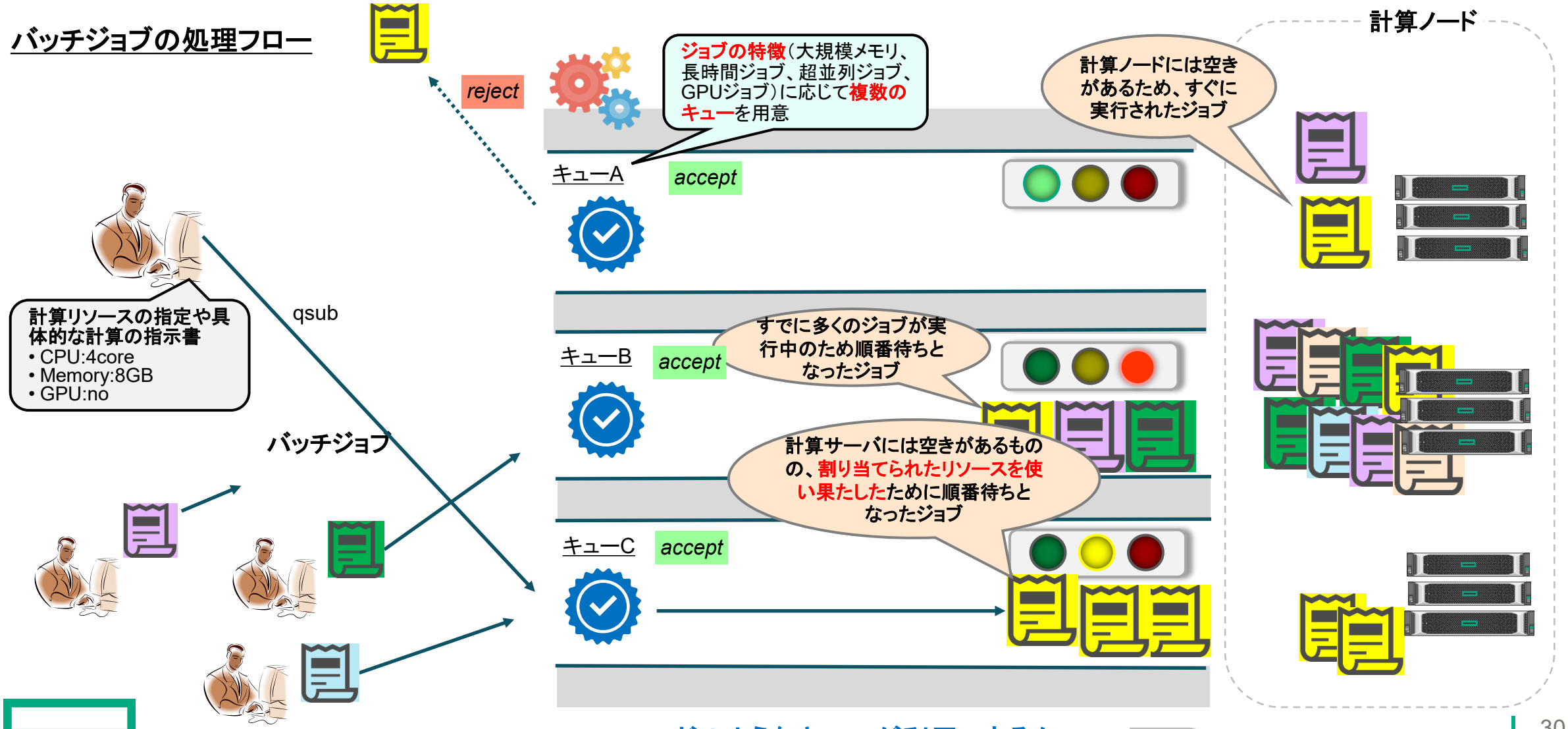
3. PBS > PBSとは

割り当てリソース不足の場合

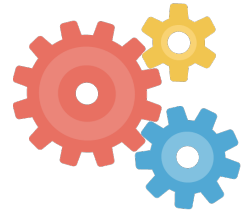


3. PBS > PBSとは

バッチジョブの処理フロー



3. PBS > キュー構成



バッチジョブのキュー構成

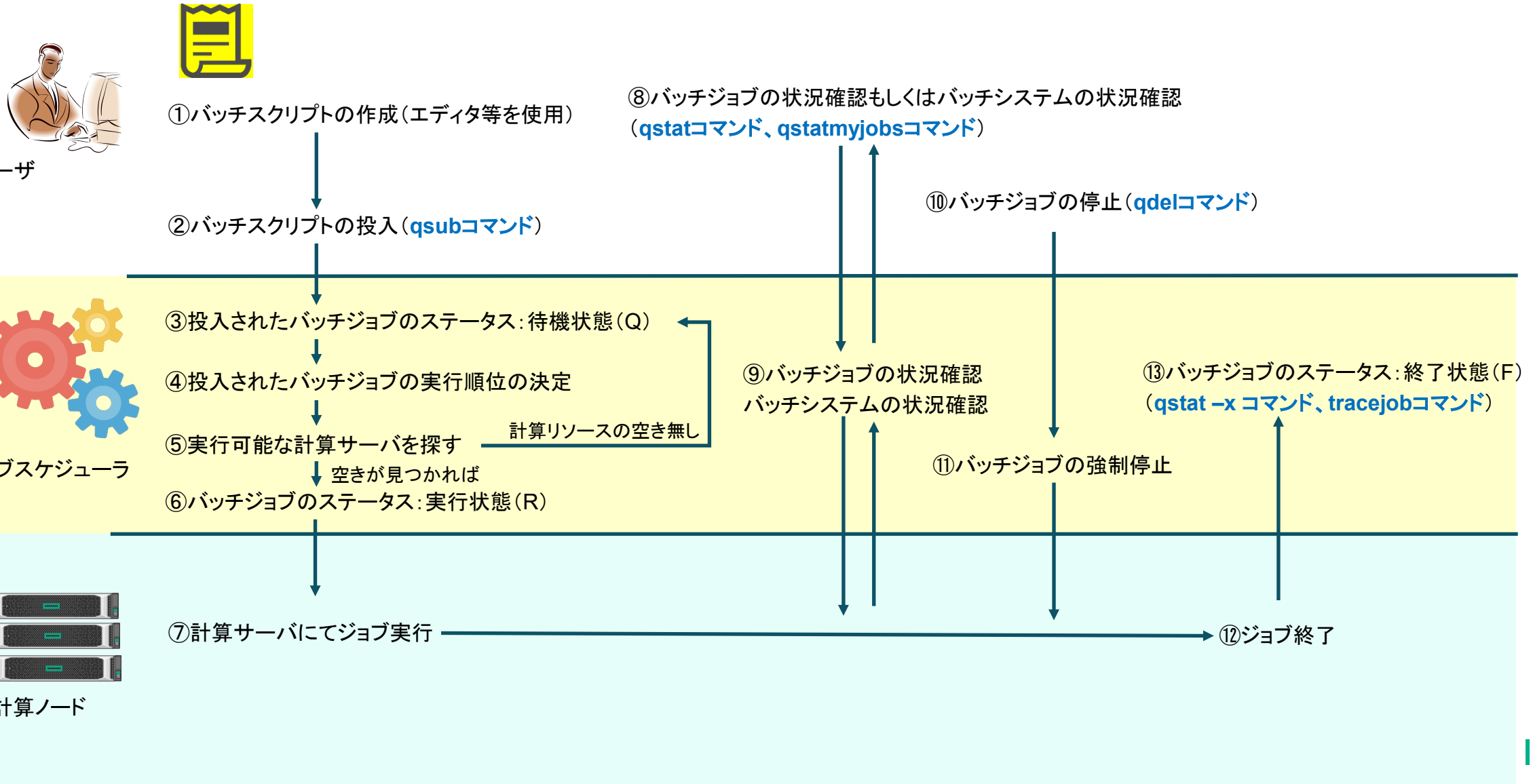
キュー名	SMALL	APC	APG	SDF
計算サーバ	クラスタ	クラスタ	クラスタ	大規模メモリ
キューの実行順位	70	50	80	90
ジョブあたりの最大コア数 (デフォルト)	12 (1)	56 (※) (1)	64 (1)	144 (18)
ジョブあたりの最大メモリ (デフォルト)	48 GB (4 GB)	980 GB (※) (4 GB)	980 GB (4 GB)	12 TB (768 GB)
ジョブあたりの最大経過時間 (デフォルト)	12 h (6 h)	制限なし (120 days)	制限なし (120 days)	制限なし (120 days)
ジョブあたりの最大GPU数	-	-	2	-
ユーザあたりの最大ジョブ数 (ソフトリミット)	-	-	-	8 (4)
ユーザあたりの最大使用コア数 (ソフトリミット)	96	-	-	288 (144)
ジョブの特徴	小規模	中～大規模	GPU	大規模メモリ

スパコンシステムにおいてユーザが利用できる計算リソース	制限値
ユーザあたりの合計コア数（ソフトリミット）	500 (300)
ユーザあたりの合計メモリ（ソフトリミット）	18 TB (3 TB)
ユーザあたりの合計GPU数（ソフトリミット）	4 (1)

【注意点】
- 「(デフォルト)」とは、値を指定しない場合に自動的に設定される値です。
- 「(ソフトリミット)」とは、それ以上の計算リソースは利用可能であるものの、そのジョブの優先順位が低く設定される閾値です。具体的には、ソフトリミットを越えていない待機ジョブの実行可否チェックをすべて終えた後に、ソフトリミットを越えたジョブの実行可否チェックを行います。
- GPUを利用する場合には、APGキューをご利用ください。
- GPU利用可能な枚数はユーザ当たり4枚です。
- （※）APCキューでは複数ノードをまたぐジョブを実行することが可能です。その場合、上記記載よりも多くのコア数およびメモリの利用が可能です。
- SDFキューにのみ、当該キューにおけるジョブ実行数や使用コア数に制限があります。

参照）HOME > 使い方と注意事項 > 計算サービス > ジョブ投入システム
https://www.scl.kyoto-u.ac.jp/Attention_main/batch_sys.html

3. PBS > バッチジョブの作成から実行・状況確認・終了までの全体フロー



3. PBS > バッチスクリプト

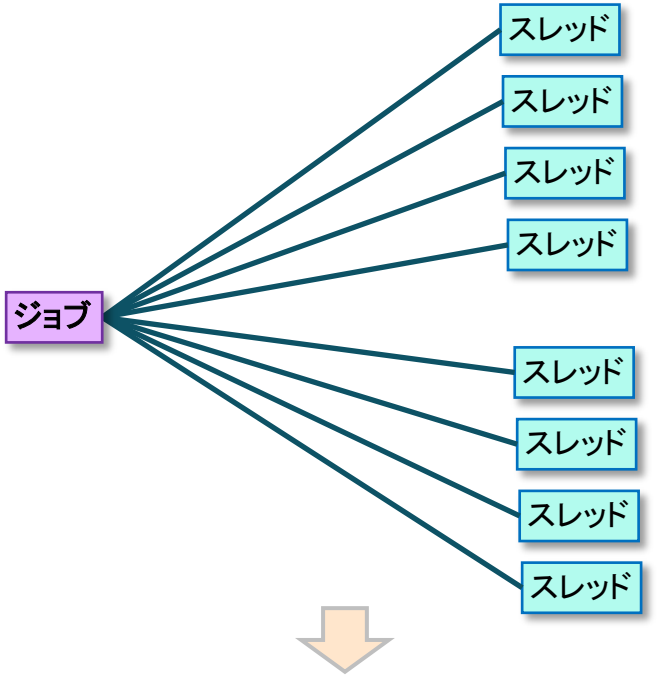
```
#!/bin/csh                                  ← シェルを指定（必須）。#!/bin/bash も可。
#PBS -N my-job                              ← 投入するジョブ名の指定（推奨）
#PBS -q APC                                 ← 投入するキューを指定（必須）
#PBS -l select=1:ncpus=8:ompthreads=8:mem=4gb   ← 必要なリソースの要求（8コア,4GBメモリ）（必須）
```

バッチジョブの設定

```
source /etc/profile.d/modules.csh           ← moduleコマンド利用時に必要な環境設定
cd ${PBS_O_WORKDIR}                         ← qsub コマンド実行ディレクトリへ移動

./a.out                                     ← ジョブ実行
```

ジョブ実行

上記ではompthreads=8の代わりに環境変数 OMP_NUM_THREADS にてスレッド数を指定することも可能です。



-l select=1:ncpus=8:ompthreads=8

主なオプション（以下のオプションは、qsubコマンドのオプションで指定して実行することも可能）

- -N　ジョブ名の指定（推奨）
- -q　ジョブを投入するキューの指定（必須）
- -l（エル）　ジョブ実行に必要なリソースの要求（チャンク数、プロセッサ数、MPIプロセス数、スレッド数、メモリ容量等）（必須）
  リソース指定の区切りは：（コロン）　例）-l select=1:ncpus=12:mpiprocs=3:ompthreads=4:mem=4gb
- -l（エル）walltime　ジョブの経過時間（実際の時間）の最大値　※デフォルト値よりも大きな値を指定したい場合に使用
- -o　標準出力ファイルのPATHの指定
- -e　標準エラー出力ファイルのPATHの指定
- -j oe　標準出力と標準エラー出力をまとめて出力

3. PBS > バッチスクリプト

```
#!/bin/csh                  ← シェルを指定（必須）。#!/bin/bash も可。
#PBS -N my-job              ← 投入するジョブ名の指定（推奨）
#PBS -q APC                 ← 投入するキューを指定（必須）
#PBS -l select=1:ncpus=12:mpiprocs=3:ompthreads=4:mem=4gb ← 必要なリソースの要求（12コア,4GBメモリ）（必須）
```

バッチジョブの設定

```
source /etc/profile.d/modules.csh   ← moduleコマンド利用時に必要な環境設定
module load mpi

cd ${PBS_O_WORKDIR}                 ← qsub コマンド実行ディレクトリへ移動

mpirun ./a.out                      ← ジョブを実行
```

ジョブ実行

※） MPI並列を実行する場合、-l select にて mpiprocs の指定を行ってください。

主なオプション（続き・・・）

-l（エル） ジョブ実行に必要なリソースの要求（チャンク数、プロセッサ数、メモリ容量等）（必須）

リソース指定の区切りは：（コロン） 例）-l select=1:ncpus=12:mpiprocs=3:ompthreads=4:mem=4gb

主な指定リソース

select=(チャンク数) ※同一ノード内で実行するジョブは1を指定（ノードをまたぐMPIジョブを実行する場合には適切な数値を指定）。

ncpus=(コア数) ※ スレッド数 x MPIプロセス数

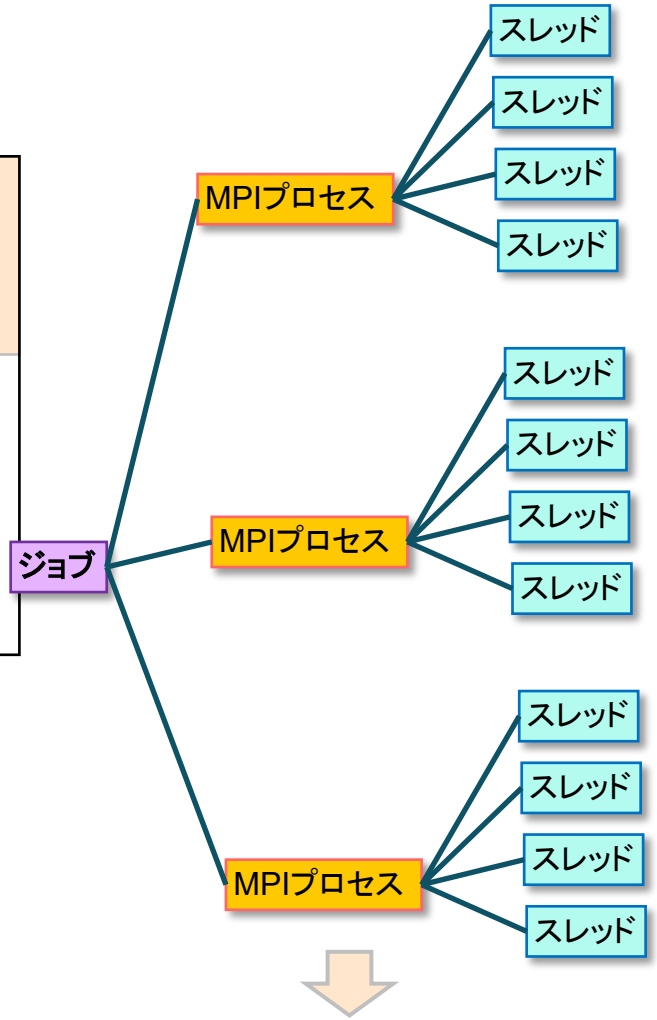
mpiprocs=(MPIプロセス数) ※プロセス間通信での並列（MPI, sockets等）に指定

ompthreads=(スレッド数) ※スレッド並列（pthreadsやOpenMP等）に指定。ompthreadsの指定は環境変数OMP_NUM_THREADSを使用して指定することと同じ。

mem=(最大物理メモリ容量) ※デフォルト値よりも大きな値を指定したい場合に使用。ここで**指定されたメモリをジョブが超えた場合、PBSによってそのジョブは強制終了されます。**そのため、例えば、Gaussianジョブでは入力ファイル内に使用メモリを記述しますが、その値よりも少し大きめの値を設定されることをお勧めします。

ngpus=(GPU数の指定) ※APGキュー（GPUジョブ）の場合にのみ使用

host=(計算ノードのホスト名) ※指定された計算ノードにおいてジョブ実行（通常はPBSによって実行ノードの管理を行うため、ユーザ側では指定しない。）



-l select=1:ncpus=12:mpiprocs=3:ompthreads=4

selectで指定する**チャンク数**とは正確には**プロセスの塊の数**を指します。同一ノード内でジョブを実行する場合は1となります。もし、複数ノードを使用したジョブを実行されたい場合には、改めてスパコンシステムにてお問い合わせください。詳しい設定方法をお知らせします。

3. PBS > よく利用するコマンドとその利用方法

■ ジョブの状況確認

$ qstat [オプション] [Job ID]

- 主なオプション

-a	# 比較的詳しいジョブ情報
-r	# 実行中のジョブだけを表示
-x	# 終了したジョブも表示
-n1 [Job ID]	# ジョブが実行されているホスト名を1行にまとめて表示
-t [Job ID]	# アレイジョブのサブジョブ毎に表示
-f [Job ID]	# full format

意図せずステータスがHとなりジョブが実行されない場合はスパコンシステムまでお知らせください。

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
W: 待機中
H: 保留状態
T: 移動中
S: 中断中
B: 実行中(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 …]

CPU時間

```
[fukumoto@fe1 check]$ qstat
Job id            Name             User              Time Use S Queue
----------------  ---------------- ----------------  -------- - -----
3154162.fe3-adm   Amber_Check      fukumoto          00:01:06 R APC

[fukumoto@fe1 check]$ qstat -a

fe3-adm:
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
3154162.fe3-adm fukumoto APC      Amber_Che* 15741*   1   8    4gb 2880: R 00:05

[fukumoto@fe1 check]$ qstat -n1

fe3-adm:
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
3154162.fe3-adm fukumoto APC      Amber_Che* 15741*   1   8    4gb 2880: R 00:05 cs40-adm/2*8
```

経過時間

実行ノード

```
[fukumoto@fe1 check]$ qstat -f 3154162.fe3-adm
Job Id: 3154162.fe3-adm
    Job_Name = Amber_Check
    Job_Owner = fukumoto@fe1-adm
    resources_used.cpupercent = 709
    resources_used.cput = 00:19:45
    resources_used.mem = 1195728kb
    resources_used.ncpus = 8
    resources_used.vmem = 7341848kb
    resources_used.walltime = 00:02:31
    job_state = R
    queue = APC
    server = fe3-adm
    Checkpoint = u
    ctime = Thu Aug 15 14:25:32 2024
    Error_Path = fe1-adm:/user1/scl7/fukumoto/appli_check/amber/check/Amber_Che
                 ck.e3154162
    exec_host = cs40-adm/2*8
    exec_vnode = (cs40-adm:ncpus=8)
    Hold_Types = n
    Join_Path = oe
    Keep_Files = n
    Mail_Points = a
    mtime = Thu Aug 15 14:28:06 2024
    Output_Path = fe1-adm:/user1/scl7/fukumoto/appli_check/amber/check/Amber_Ch
                 eck.o3154162
    Priority = 0
    qtime = Thu Aug 15 14:25:32 2024
     (途中省略)
    stime = Thu Aug 15 14:25:33 2024
    session_id = 1574199
    sandbox = PRIVATE
    jobdir = /scratch/pbs_jobdir/pbs.3154162.fe3-adm.x8z
    substate = 42
    Variable_List = PBS_O_HOME=/user1/scl7/fukumoto,PBS_O_LANG=en_US.UTF-8,
                 PBS_O_LOGNAME=fukumoto,
     (途中省略)
                 PBS_O_WORKDIR=/user1/scl7/fukumoto/appli_check/amber/check,
                 PBS_O_SYSTEM=Linux,PBS_O_QUEUE=APC
    comment = Job run at Thu Aug 15 at 14:25 on (cs40-adm:ncpus=8)
    etime = Thu Aug 15 14:25:32 2024
    run_count = 1
    Submit_arguments = test.qsub
    project = _pbs_project_default
    Submit_Host = fe1-adm
```

3. PBS > よく利用するコマンドとその利用方法

- キューの状況確認
  $ qstat ［オプション］［キュー名］
  - 主なオプション
    -Q　　# 全キューに対するステータスごとのジョブ数等を表示
    -Qf ［キュー名］　# ある特定のキューの制限値を確認

キューに対する制限やデフォルト値等はSCLホームページをご確認いただくか、もしくは後で紹介するqstatmyjobsコマンドを利用することで確認できます。

```
[fukumoto@fe1 training]$ qstat -Q
Queue              Max   Tot Ena Str   Que   Run   Hld   Wat   Trn   Ext Type
---------------- ----- ----- --- --- ----- ----- ----- ----- ----- ----- ----
SMALL                0     2 yes yes     0     2     0     0     0     0 Exe*
APC                  0   538 yes yes   115   413     0     0     0     0 Exe*
APG                  0    13 yes yes     0    11     1     0     0     0 Exe*
SDF                  0    41 yes yes     2    37     0     0     0     0 Exe*
```

ジョブのステータス
Que: 実行待ち
Run: 実行中
Ext: 終了処理中
F: 終了したジョブ
Wat: 待機中のジョブ
Hld: 保留状態
Trn:　移動中
S: 中断中
B: 実行中（アレイジョブ）
X: 終了したアレイジョブのサブジョブ

```
fe1{appadm}1019: qstat -Qf APC
Queue: APC
    queue_type = Execution
    Priority = 50
    total_jobs = 538
    state_count = Transit:0 Queued:115 Held:0 Waiting:0 Running:413
Exiting:0 Begun:10
    resources_max.mem = 9800gb
    resources_max.ncpus = 560
    resources_max.ngpus = 0
    resources_default.mem = 4gb
    resources_default.ncpus = 1
    resources_default.walltime = 2880:00:00
    default_chunk.Qlist = APC
    resources_assigned.mem = 42307026944kb
    resources_assigned.mpiprocs = 684
    resources_assigned.ncpus = 3626
    resources_assigned.nkaas = 2
    resources_assigned.nodect = 413
    resources_assigned.nviptree = 0
    enabled = True
    started = True
```

演習　3. PBS

ゴール：　バッチスクリプトの書き方を理解し、およびジョブ投入・状況確認の手順を確認する

3. PBS > qstatmyjobsコマンド

▪ユーザの使用コア数・メモリ量、システムの利用状況の確認

qstatmyjobsコマンド（＊）が利用できます。

（＊）PBSによって提供されるコマンドではなく、
スパコンシステム独自のコマンドです。

qstatmyjobsコマンドの実行例

```
% 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   1966( 63%)     0/UNLTD     12   12      0/96        48     48     0/UNLTD       -    -       -/-         6 12
APC     1854( 65%)     1/UNLTD     56   56  56/UNLTD       240    980   224/UNLTD       -    -       -/-      2880 UNLTD
APG      260( 54%)     0/UNLTD     55   64   0/UNLTD       450    980     0/UNLTD       1    2   0/UNLTD      2880 UNLTD
SDF      344( 65%)     3/8          -  144    60/288         -  12288  2400/UNLTD       -    -       -/-      2880 UNLTD
=====================================================================================================================
TOTAL:          JOBS)  4/UNLTD        CPUS) 116/1100          MEM)  2624/18432          GPUS)  0/4

fe3-adm:
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
3148165.fe3-adm appadm   SDF      MS_83CXZ   17743*   1  20  800gb 2880: R 188:0 sdf1/2*0
3148166.fe3-adm appadm   SDF      MS_83QZ5   17783*   1  20  800gb 2880: R 188:0 sdf1/7*0
3148167.fe3-adm appadm   SDF      MS_84U01   17814*   1  20  800gb 2880: R 188:0 sdf1/17*0
3153564.fe3-adm appadm   APC      MS_KGQ5F   967389   1  56  224gb 2880: R 05:27 cs06-adm/0*56
```

UNLTD: unlimited (無制限)

現在実行中のジョブのみを表示
(qstat -rn1)

- use% で各バッチキューの混み具合がわかります。→ 指定するキューの参考（例えば、APC/SDFのどちらのキューにジョブを投入するか）
- 各キューですぐにジョブが確保できるリソースがわかります。→ 指定する計算リソースの参考
- 自分がすでに使用しているリソースを確認できます。→ 今後、どれだけのリソースが利用できるか把握できます。
- コア数は少なくてもよいからメモリをより多く確保できるリソースの指定方法を確認したい場合は –m オプションを利用してください。
  % qstatmyjobs -m

3. PBS > qstatmyjobsコマンド

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、未使用（ジョブが実行されていない）のコア数
(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	ユーザが最大実行可能なジョブ数
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	すぐに実行開始できる、ジョブの最大コア数指定時に、指定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

3. PBS > qstatmyjobsコマンド

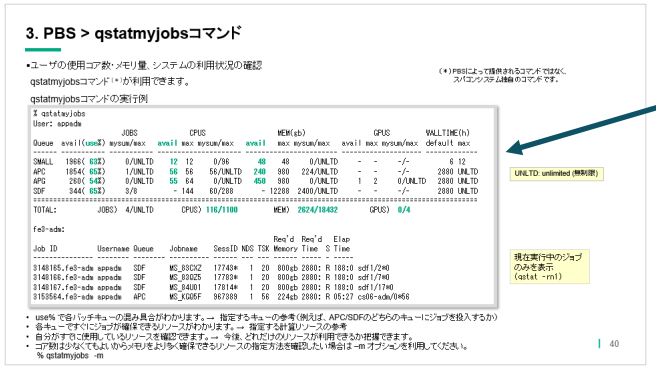
```
fe1{appadm}1001: qstatmyjobs -h
/usr/appli/freeware/bin/qstatmyjobs [-u user] [-c|-m] [-q|-j|-t] [-a] [-d] [-l] [-h]
 -u : specify a user (for administrators)
 -c : display the maximum number of available cores and
      the maximum available memory size in that case (default)
 -m : display the maximum size of available memory and
      the maximum available cores in that case
 -q : display status information only for queues.
 -j : display status information for queues, jobs and job arrays (not subjobs) (default)
 -t : display status information for queues, jobs, job arrays, and subjobs
 -a : display system queues as well as normal queues (for administrators)
 -d : debug mode (for administrators)
 -l : debug mode with load average check (for administrators)
 -h : print help (this message)
```

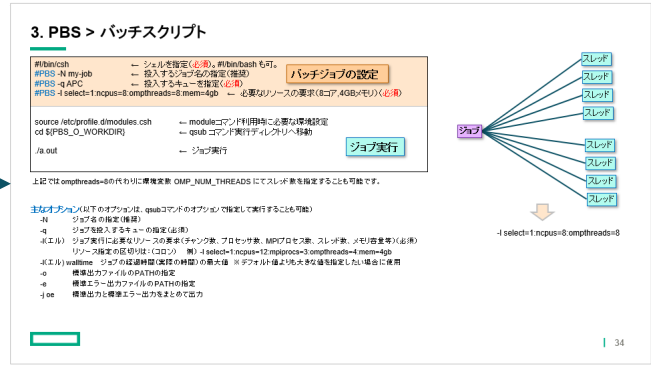
CPUSの avail の表示では、最も多くのコア数が利用できるノードを探して、そのコア数を表示する。この時、MEM(gb) の avail の表示では、当該ノードのメモリが表示される。もし、当該ノードが複数あった場合には、それら複数のノードの中で最も利用できるメモリサイズが大きいノードの空きメモリの値が表示される。

CPUSの場合とは逆に、MEM(gb) の avail の表示では、最も多くのメモリが利用できるノードを探して、そのメモリサイズを表示する。この時、CPU の avail の表示では、当該ノードのコア数が表示される。もし、当該ノードが複数あった場合には、それら複数のノードの中で最も利用できるコア数が大きいノードの空きコア数が表示される。

3. PBS > qstatmyjobsコマンド

qstatmyjobsコマンドの使用目的 → ジョブが必要とする計算リソースが確保できるかを確認したい





例） 8core/4GBメモリが必要なジョブ

- ユーザの利用可能な計算リソースに、空きがあるかどうかを確認する
- 指定したキューにてジョブが実行可能な計算リソースに空きがあるかどうかを確認する（空きがない状態でジョブ投入するとジョブ待ちが発生）

3. PBS > インタラクティブバッチジョブの実行

- インタラクティブバッチジョブとは
  - リアルタイムでジョブを操作できるタイプのジョブ。ノードにログイン後、**コマンドラインで対話的に操作ができます**。
- インタラクティブバッチジョブの利用目的
  - 自分のジョブの状況をコマンドで確認したい。(ps, top 等)
  - 対話形式のジョブやアプリケーションを長時間にわたって実行したい。(python, R, mathematica等)
- インタラクティブバッチジョブの実行
  - qsub -I -q SMALL –l select=1:ncpus=1
    インタラクティブバッチジョブでは -I(大文字のアイ) を指定(Interactive)
    インタラクティブバッチジョブを実行すると、入出力インタフェースがジョブ投入ウィンドウに返されます。
  - インタラクティブバッチジョブを終了するには exit を入力します。
  - 実際にジョブが実行されているかどうかにかかわらず、常に計算リソースを確保し続けますので、不要になりましたら、速やかに exit をお願いします。

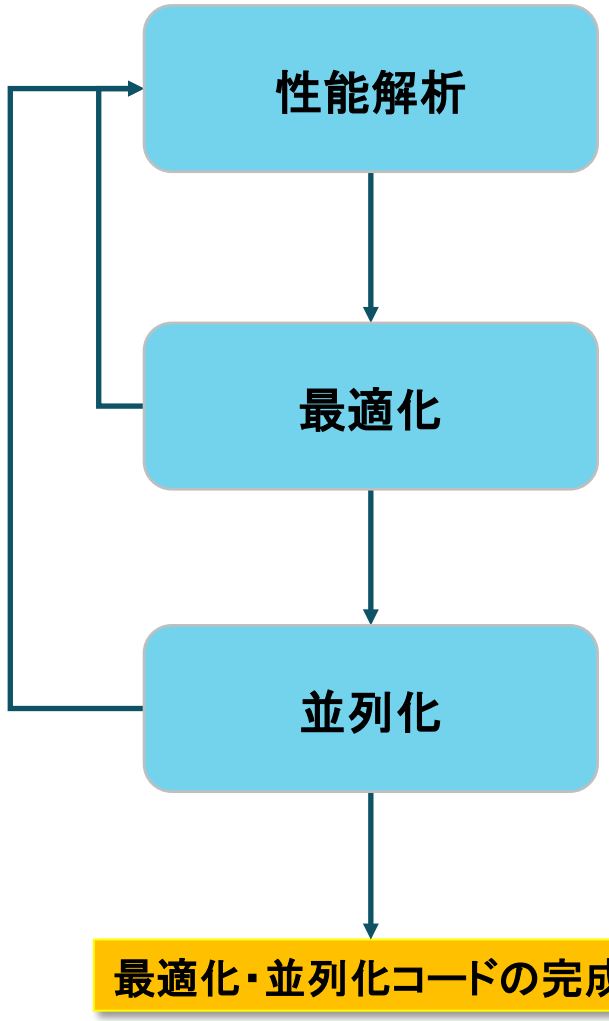
```
fe1{appadm}1021: qsub -I -q SMALL -l select=1:ncpus=4:mem=8gb
qsub: waiting for job 3154269.fe3-adm to start
qsub: job 3154269.fe3-adm ready

cd /scratch/pbs_jobdir/pbs.3154269.fe3-adm.x8z
[appadm@cs41 ~]$ cd /scratch/pbs_jobdir/pbs.3154269.fe3-adm.x8z
[appadm@cs41 pbs.3154269.fe3-adm.x8z]$
```

4. 最適化・並列化手法

- 最適化・並列化手順の概要
- 最適化）コンパイラオプション
- 並列化）アムダールの法則
- 並列化）OpenMPおよびMPIの特徴

4. 最適化・並列化手法 ＞ 最適化・並列化手順

◆アプリケーションプログラムの高速化を検討する際は、
一般に次のような手順で最適化・並列化を行います。



一般には、この手順を繰り返すことによって高い性能が得られます。

計算時間の大部分を占めている処理

性能解析ツールを使用して、プログラムの**ボトルネック**になっている部分やその**原因を特定**

まずは逐次実行での高速化を目指す

- **プログラムの最適化**
  - ✓ コンパイラオプションの適用
  - ✓ 最適化された数値計算ライブラリへの置換（Intel MKL, OpenBLAS, FFTW 等）
  - ✓ コード修正による最適化（実装アルゴリズムの変更、ループ入替によるメモリアクセス効率の向上等）

- **プログラムの並列化**
  - ✓ 自動並列化
  - ✓ OpenMP
  - ✓ MPI
  - ✓ Hybrid (MPI + OpenMPなど)

複数のプロセスを使って高速化を目指す

4.最適化・並列化手法 > 推奨するコンパライオプション

- デフォルトで設定されている主なオプション

オプションの種類	オプション	オプションのレベル
最適化レベル	-O2	パフォーマンス向上のための最適化を行ないます。
特定のプロセッサ向けの最適化	-msse2	インテルプロセッサ向けにSSE2およびSSE命令を生成し、SSE2対応のインテルXeonプロセッサ向けの最適化をします。

- 推奨するオプション

オプションの種類	オプション	オプションのレベル
最適化レベル	-O3	-O2に加えプリフェッチ、スカラー置換、ループ変換、およびメモリアクセス変換などのより強力な最適を有効にします。
特定のプロセッサ向けの最適化	-xCORE-AVX512	AVX512、AVX2、AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2、SSE 命令を生成します。AVX512 命令セット対応のインテル プロセッサー向けに最適化します。
コンパイルを行ったプロセッサで最も高いレベルの最適化	-xHOST	コンパイルをしたプロセッサで利用可能な、最も高いレベルの命令を生成し、そのプロセッサ向けの最適化を行います。

特に、特定のプロセッサ向けの最適化オプションは、実行速度の変化を確認してください（遅くなることもあります）。

プリフェッチ：メモリ上のデータをCPUが実際に必要とする前に、あらかじめキャッシュメモリに読み込む技術
スカラー置換：メモリに頻繁にアクセスする変数や配列の要素を、一時的にCPUレジスタやローカル変数に置換する技術
ループ交換：多重ループ（入れ子になったループ）の内外のループを入れ替える
メモリアクセス変換：プログラムのデータのメモリアクセスパターンの最適化（配列や構造体のメモリ配置の変更等）

4. 最適化・並列化手法　> アムダールの法則

- プログラムを並列化することのメリットは、実行時間(ターンアラウンドタイム) が短縮されることです。
- 「並列化によるスピードアップ s」とは、下式のように、並列化せずに実行した場合の実行時間 T_1 と、並列数 N で実行した場合の実行時間 T_N の比であると定義します。
- 言い換えると、並列化によって実行時間は何倍短縮されたのか、が s になります。

$$s \equiv \frac{T_1}{T_N}$$

並列化していない場合の実行時間が100秒のジョブ。
使用する並列数を増やした場合の実行時間とスピードアップの例

N	1	2	4	8	16
T_N	100(=T_1)	52	30	20	14
s	1	1.9	3.3	5.0	7.1

並列数を2、4と増やしても、性能は2倍、4倍とはならない場合が多い。

なぜ理想的な性能が出ないのか？

16並列でおよそ7倍高速化

4. 最適化・並列化手法　> アムダールの法則

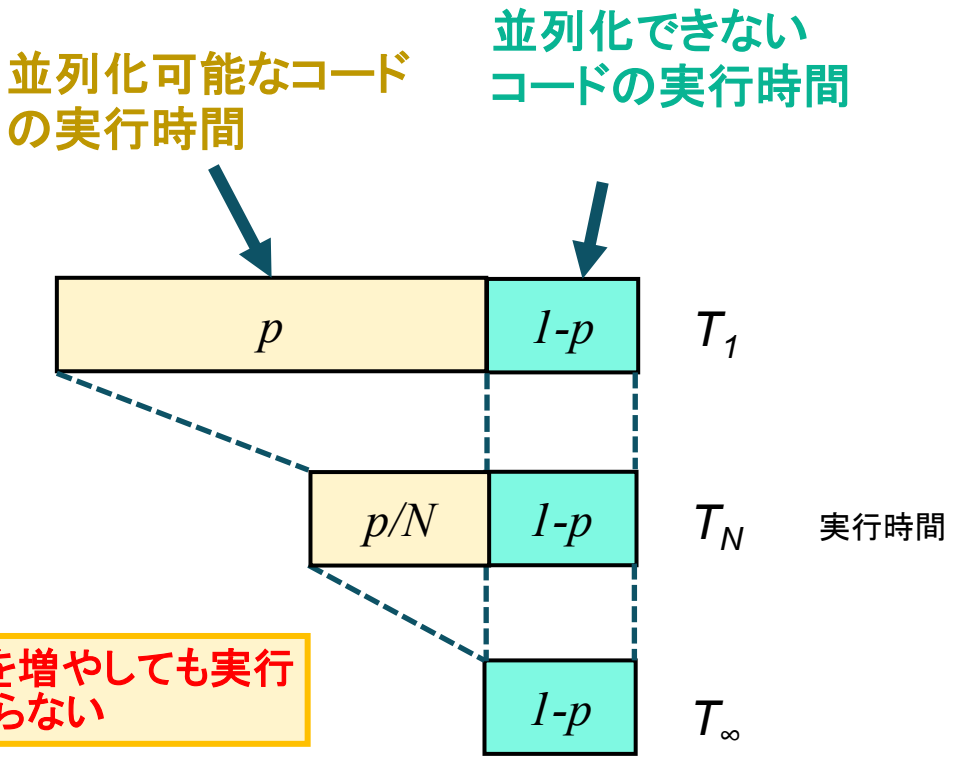
- あるプログラムを逐次実行した際の実行時間のうち、並列化できる部分の割合（並列化率）をp (0 ≦ p ≦ 1) とします。このとき、並列数 N で実行した場合のスピードアップ s は、並列化のオーバーヘッド等を無視できるとすると、以下の式(アムダールの法則）に従うことが知られています。

$$s = T_1/T_N = \frac{T_1}{\frac{p}{N}T_1 + (1-p)T_1} = \frac{1}{\frac{p}{N} + (1-p)} \quad (0 \le p \le 1)$$

上式を p について解くと

$$p = \frac{1 - \frac{1}{s}}{1 - \frac{1}{N}}$$

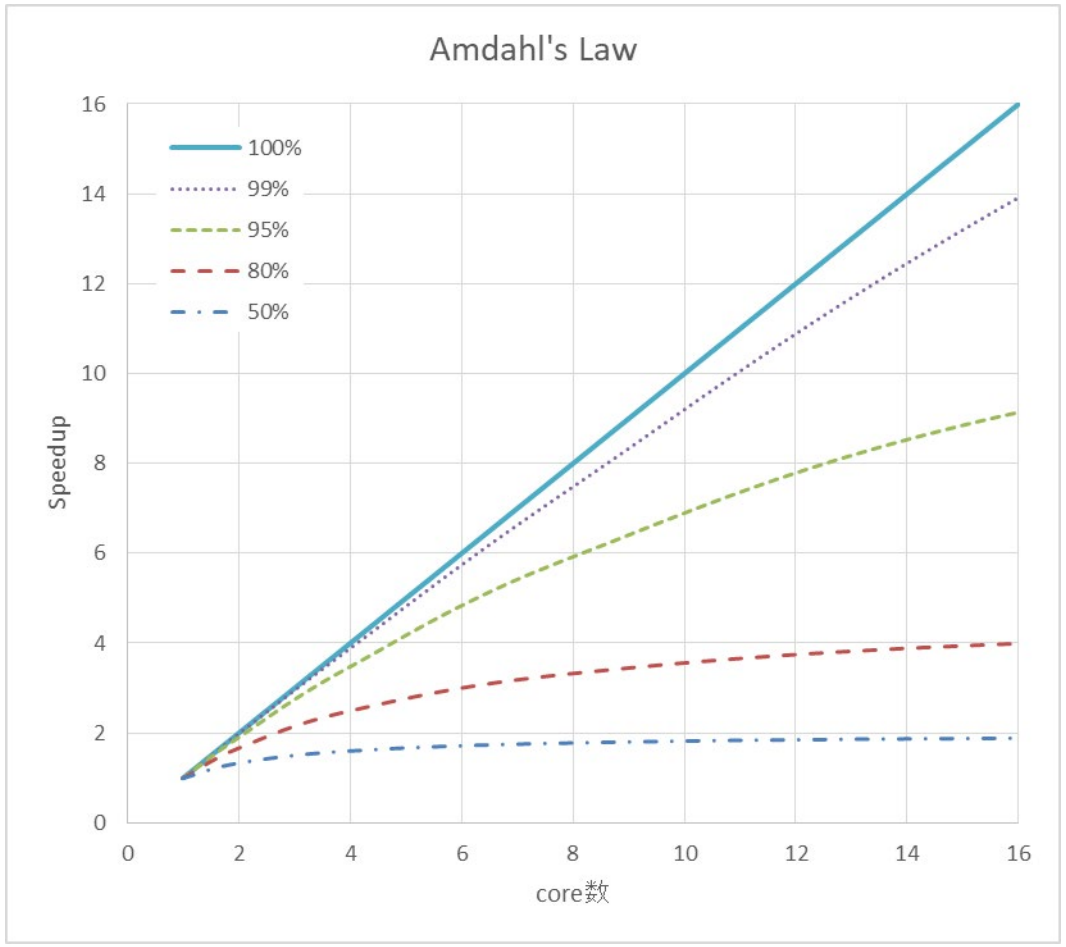
先ほどのケース:
N=16, s=7.0（16並列で7倍高速化）
⇒ p = 0.91 : 並列化率91%

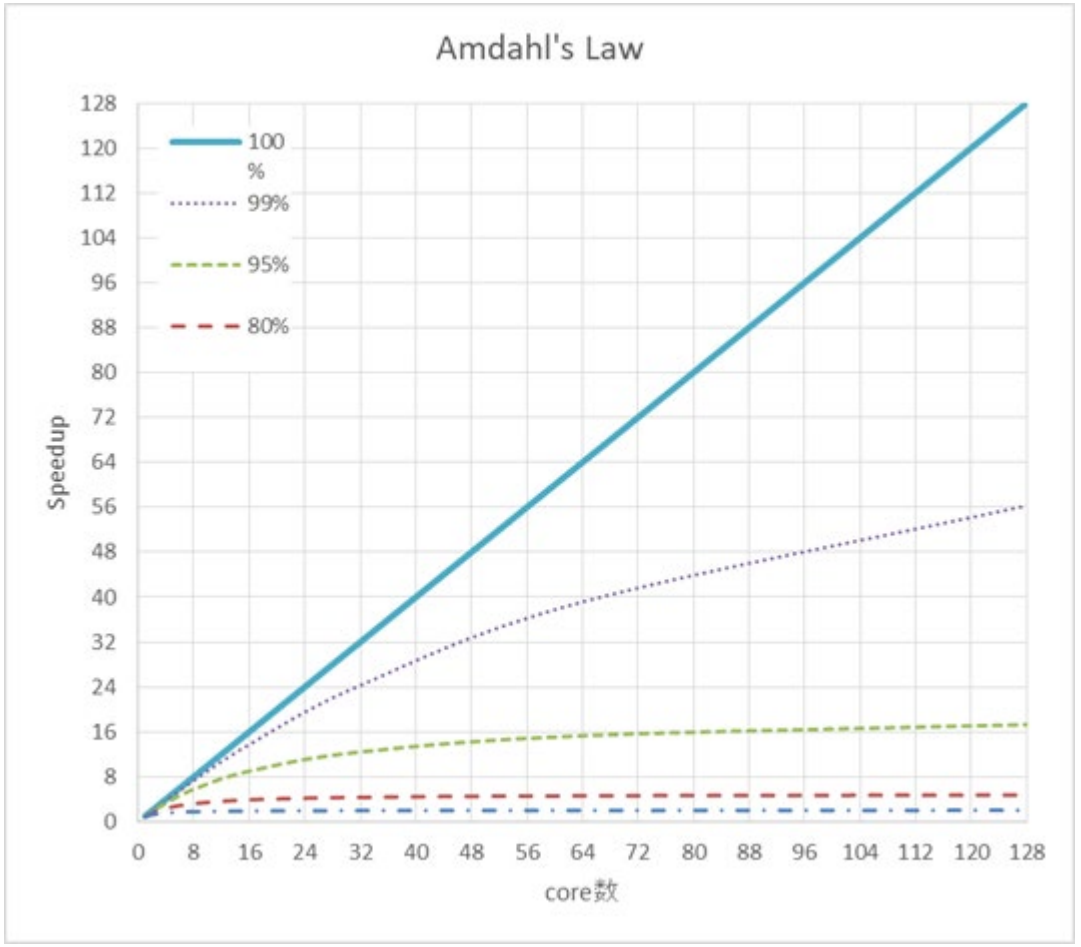


どんなに並列数を増やしても実行時間は 0 にはならない

4. 最適化・並列化手法　> アムダールの法則

- アムダール の法則によるスピードアップの理論値





多くのコアを使って性能向上を行うためには、ソースコードの大部分を並列化させることが必要
あまり並列化できていないプログラムでは、コア数をむやみに増やしてもリソースを無駄に消費してしまう

4. 並列化・並列化手法　> 自動並列化、OpenMP、MPI

性能向上のカギは並列化

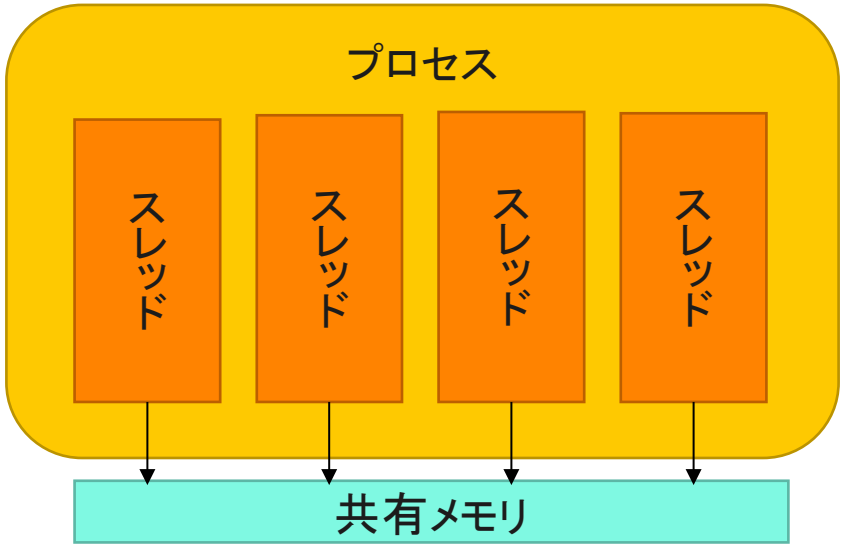
自動並列化
- マルチスレッドの並列化
- コンパイラによる最適化と組み合わせた並列化
- **コンパイルオプションによる簡単な操作**
- 並列化診断メッセージによるレポート

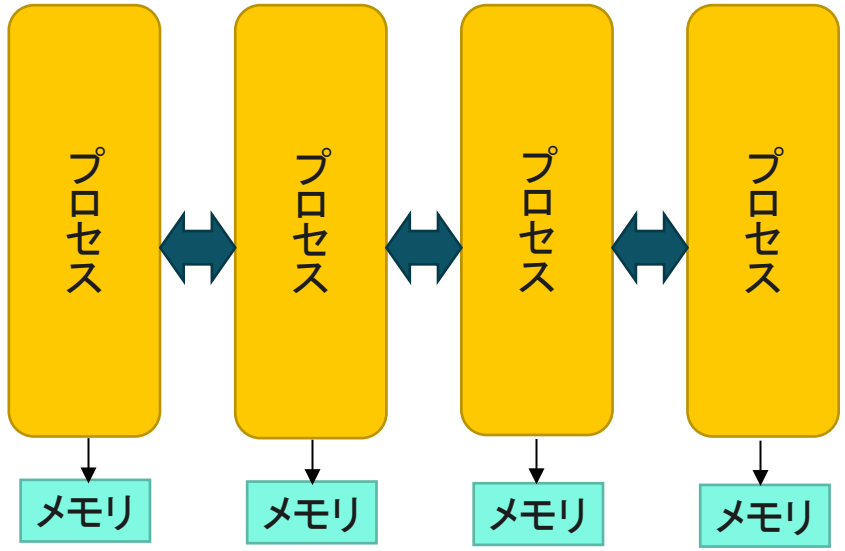
OpenMP
- **共有メモリモデル**

⇒ノードをまたいだ並列化は不可
- 主に**ループレベルの並列化**
- 段階的な適用が可能で導入が容易
- 高度なスケーラビリティを得るためには**粒度を大きくする工夫**が必要

MPI
- **分散メモリモデル**
- 領域分割等による並列化
- 最初から**プログラム全体の並列化**が必要

⇒並列化を意識したコーディングが重要
- 粗粒度の並列化
- スケーラビリティを得るためには高速なネットワークインターコネクトや遅延隠蔽が必要





プロセス：**独立した実行単位**で他のプロセスとは分離されています。
スレッド：同じプロセス内で**リソースを共有**しながら実行される実行単位です。

5. 計算化学アプリケーション

◆アプリケーションの概要
◆ジョブ実行方法（Gaussianの場合）
◆ジョブ実行方法（LAMMPSの場合）

5. 計算化学アプリケーション ＞ アプリケーションの概要

計算化学

名称に★がついているアプリケーションはGPU対応版も利用可能です。

分類	名称	バージョン	モジュール名	主なコマンド	キュー	利用範囲
化合物DB	CSD	2025.1	CSD	cq	-	京都大学内（宇治）
計算化学統合パッケージ	Materials Studio★	2025	ms	-	全て	京都大学内（宇治・桂・吉田）
	Discovery Studio★	2025	-	-	全て	京都大学内（宇治・桂・吉田）
	Materials Science Suite★	2025.1	-	-	全て	京都大学内（宇治）
量子化学	ADF	2025.102	adf	ams	全て	京都大学内（宇治）
	CASTEP (STFC版)	24.1	castep	castep.serial castep.mpi	APGを除く全て	非営利
	CP2K	2025.1	cp2k	cp2k	APGを除く全て	制限なし
	GAMESS	2023.9.30	gamess	gamess	全て	制限なし
	Gaussian16	G16c01	g16	rung16	全て	制限なし
	GaussView6	6.1	g16	gv	fe1	制限なし
	Gaussian/GaussView サイトライセンス	-	-	-	-	京都大学内（宇治）
	Molpro	2025.2	molpro	molpro	全て	京都大学内（宇治）

このリストにないアプリケーションを使用されたい場合はその旨スパコンシステムまでお知らせください。無償なアプリケーションであれば、可能な範囲で導入させていただきます。

	MOPAC7	7	mopac/7	runmopac	全て	制限なし
	MOPAC2016	2016	mopac/2016	mopac	全て	非営利
	NWChem ★	7.2.3	nwchem	nwchem	全て	制限なし
	octopus(外部)	13.0	octopus	octopus	全て	制限なし
	Q-Chem	6.3.0	qchem	qchem	全て	非営利
	QUANTUM ESPRESSO ★	7.3.1	qe	cp.x,pw.x	全て	制限なし
	TURBOMOLE	7.9	turbomole	-	全て	京都大学内(化研)
	Yambo(外部)	5.0.4	yambo	yambo	全て	制限なし
分子動力学	AMBER ★	24	amber	pmemd,sander	全て	非営利
	Gromacs ★	2024.4	gromacs	gmx,gmx_d, gmx_mpi,gmx_mpi_d	全て	制限なし
	LAMMPS ★	29 Aug 24 update1	lammps	lmp_intel_cpu, lmp_kokkos_omp,lmp_kokkos_mpi_only, lmp_intel_gpu,lmp_kokkos_gpu	全て	制限なし
	NAMD ★	3.0	namd	namd3	全て	非営利
X線構造解析	XPLOR-NIH(外部)	3.8	xplor-nih	xplor	fe1	非営利
	CNSsolve(外部)	1.3	cnssolve	cns_solve	fe1	非営利
	XDS(外部)	Jun 30,2023	xds	xds,xds-viewer,xdsgui,xdsstat,xdsme	fe1	非営利
可視化・描画	Avogadro2(外部)	1.100.0	avogadro2	avogadro2	fe1	制限なし
	gnuplot(外部)	6.0.2	gnuplot	gnuplot	fe1	制限なし
	grace	5.1.25	-	xmgrace	fe1	制限なし
	molden	6.2	molden	molden	fe1	非営利
	VMD ★(外部)	1.9.4, 2.0.0a5	vmd	vmd	fe1	制限なし
	xcrysden(外部)	1.6.2	xcrysden	xcrysden	fe1	制限なし
ツール関連	OpenBabel(外部)	3.1.1	openbabel	obabel,obgui	全て	制限なし

5. 計算化学アプリケーション ＞ アプリケーションの概要

分類	名称	概要	特徴
化合物DB	CSD	有機化合物・有機金属化合物の結晶構造データベース	Web版およびアプリケーション版（Windows/MacOS/Linux） 利用にあたっては事前に使用許諾書を提出し、アクティベーションコードを受け取ってください。
計算化学統合パッケージ	Materials Studio	主に材料設計をターゲットとしたソフトウェア	すべての計算モジュールが利用可能なサイトライセンス契約（Windowsや計算サーバで利用可能）/サーバ・クライアント方式で利用可能。要申請・要追加費用
	Discovery Studio	主に生体高分子をターゲットとしたソフトウェア	すべての計算モジュールが利用可能なサイトライセンス契約（Windowsや計算サーバで利用可能）/サーバ・クライアント方式で利用可能。要申請・要追加費用
	Materials Science Suites	材料化学系モデリングプログラム	Windows版あり/サーバ・クライアント方式で利用可能。宇治キャンパスでの利用のみ。要申請
量子化学計算	ADF	密度汎関数法 (DFT) を用いた量子化学計算ソフトウェア	Windows版あり/宇治キャンパスでの利用のみ
	CASTEP（STFC版）	材料特性を第一原理に基づいて計算するためのソフトウェア。Materials Studio の計算モジュールである CASTEPと同じ。	CASTEPのアカデミックフリー版。Science and Technology Facilities Council (STFC) での利用が認められた研究グループのみ。
	GAMESS	量子化学計算ソフトウェア	sockets版とMPI版を提供。sockets版は同一ノード内での並列実行をサポート。
	Gaussian16 GaussView	量子化学計算ソフトウェア	スパコンシステムが保有するNVIDIA H100では動作しない。（A100までをサポート） ライトライセンス契約により宇治キャンパス内であればスパコンユーザではなくてもWindows版/Mac版/Linux版のGaussian/GaussViewのインストールが可能
	TURBOMOLE	量子化学計算ソフトウェア	ライセンス数に限りがあるため、化研所属かつ研究室あたり1ユーザのみ使用可能。要申請

5. 計算化学アプリケーション ＞ アプリケーションの概要

分類	名称	概要	特徴
分子動力学	AMBER	分子力学および分子動力学計算パッケージ	逐次実行版：sander, pmemd MPI版：sander_MPI, pmemd_MPI GPU版：pmemd_cuda
	Gromacs	分子力学および分子動力学計算パッケージ	thread-MPI版： gmx (倍精度版：gmx_d) MPI版： gmx_mpi (倍精度版：gmx_mpi_d) GPU使用時には倍精度版は使用できません
	LAMMPS	分子力学および分子動力学計算パッケージ	CPU版/GPU版 INTELパッケージおよびKOKKOSパッケージ利用可能（詳細は後述） 特にGPUではKOKKOSパッケージ利用で性能向上となる可能性が高い
	NAMD	分子力学および分子動力学計算パッケージ	CPU版/GPU版
ツール関連	OpenBabel	ファイルフォーマット変換ツール	obabel -L formats にて対応可能なフォーマットのリストが表示される

5. 計算化学アプリケーション ＞ ジョブ実行方法（Gaussianの場合）

- 利用可能なバージョンおよびコマンド

  Gaussianバージョン：G16Rev.C01
  コマンド：rung16, formchk, cubegen 等

  ※ 最新版はRev.C02 であるもののこれはNVIDIA A100対応のパッケージであり、スパコンシステムは NVIDIA H100であるため利用不可

- 利用するための環境設定

```
$ module load g16/c01
```

- ジョブ実行方法

```
$ rung16 input.com output.log
```

  引数に入力ファイルおよび出力ファイルを指定

- バッチスクリプト作成時の注意点
  - ✓ 入力ファイルに記載するプロセス数とバッチスクリプトにて記載するプロセス数を一致
  - ✓ 入力ファイルに記載するメモリサイズよりもバッチスクリプトに記載するメモリサイズを**少し大きめ**に記載する
    ⇒ バッチスクリプトで指定したメモリをジョブが超えた場合、他のジョブの異常終了もしくは性能低下を招く可能性があるため、PBSによってジョブが強制的に停止させられる

- その他の注意点
  - ✓ Gaussianのサンプルファイルの格納場所：/usr/appli/g16/c01/g16/tests/com
  - ✓ Gaussianジョブのスクラッチファイルは共有領域である /scratch に作成されます。領域としてはかなり大きめにとっていますので、一時的に数百GBに達するファイルでも利用可能です。また課金対象外です。
  - ✓ formchk等のユーティリティコマンドが利用できる最大メモリサイズは6GBです。
  - ✓ 宇治キャンパス内の研究室であれば、スパコンシステムのユーザでなくとも、Windows版/Mac版/UNIX版のGaussianおよびGaussViewをご自身の端末や研究室のサーバにインストールすることができます。（スパコンシステムまでご連絡いただければダウンロード方法をお伝えします）
  - ✓ スパコンシステムの計算サーバで利用できるGaussianについては、その利用目的やユーザに制限はありません。例えば、商用利用目的で企業ユーザが利用することも可能です。

5. 計算化学アプリケーション ＞ ジョブ実行方法(Gaussianの場合)

入力ファイルの作成例

入力ファイル:test0397.com

```
%nproc=16
%mem=1gb
%chk=test0397.chk

#p rb3lyp/3-21g force test scf=novaracc

Gaussian Test Job 397:
Valinomycin force

0,1
0,-1.3754834437,-2.5956821046,3.7664927822
...
...
```

注意点

- 必要な計算リソースは入力ファイルの先頭部分に記述
- %nproc: 使用コア数、%mem: 使用メモリ、%chk: チェックポイントファイルのファイル名

バッチスクリプト:g16.qsub

```
#!/bin/csh
#PBS -N g16
#PBS -q SMALL
#PBS -l select=1:ncpus=16:mem=2gb

source /etc/profile.d/modules.csh
module load g16/c01
cd $PBS_O_WORKDIR

/usr/bin/time rung16 test0397.com test0397.log
```

```
$ qsub g16.qsub
$ qstat
```

注意点

- バッチスクリプト内で指定する値は入力ファイル内で指定するメモリサイズよりも**大きく設定**
- ncpusは入力ファイル内の値と一致させる
- ompthreadsやmpiprocsの指定はしない

演習　5. 計算化学（Gaussian）

ゴール：　Gaussianジョブの投入・状況確認の手順を確認する

5. 計算化学アプリケーション ＞ ジョブ実行方法（LAMMPSの場合）

- 利用可能なコマンド一覧

LAMMPSバージョン：2408.1

CPU版		GPU版	
INTELパッケージ利用	KOKKOSパッケージ利用	GPUパッケージ利用	KOKKOS（H100対応版）パッケージ利用
lmp_intel_cpu	lmp_kokkos_omp, lmp_kokkos_mpi_only	lmp_intel_gpu, lmp_intel_gpu_single, lmp_intel_gpu_double	lmp_kokkos_gpu, lmp_kokkos_gpu_single, lmp_kokkos_gpu_double

ompはハイブリッド版(OMPスレッドおよびMPI)、mpi_onlyはMPI版

GPU版については単精度版および倍精度版が利用可能

- 利用するための環境設定

```
$ module load lammps/2408.1
```

- 最適化のためのパッケージ

INTELパッケージ：Intel の汎用CPUにて最適化
⇒ スパコンシステムのサーバはすべてIntel CPUですので性能向上が期待できます。

KOKKOSパッケージ：マルチデバイス（CPU・GPU）向けの並列実行を、1つのコードベースで効率的に実装するためのパッケージ
⇒ NVIDIA H100 に対応しているため、GPUでの性能向上が特に期待できる。

- 利用可能なパッケージ一覧

AMOEBA	ASPERE	ATC	AWPMD	BOCS	BODY
BPM	BROWNIAN	CG-DNA	CG-SPICA	CLASS2	COLLOID
COLVARS	COMPRESS	CORESHELL	DIELECTRIC	DIFFRACTION	DIPOLE
DPD-BASIC	DPD-MESO	DPD-REACT	DPD-SMOOTH	DRUDE	EFF
ELECTRODE	EXTRA-COMMAND	EXTRA-COMPUTE	EXTRA-DUMP	EXTRA-FIX	EXTRA-MOLECULE
EXTRA-PAIR	FEP	GPU★	GRANULAR	H5MD	INTEL
INTERLAYER	KOKKOS★★	KSPACE	LATBOLTZ	LEPTON	MACHDYN
MANIFOLD	MANYBODY	MC	MDI	MEAN	MESONT
MGPT	MISC	ML-IAP	ML-RANN	ML-SNAP	ML-UF3
MOFFF	MOLECULE	MOLFILE	NETCDF	OPENMP	OPT
ORIENT	PERI	PHONON	PLUGIN	PLUMED	POEMS
PTM	PYTHON	QEQ	QMMM	QTB	REACTION
REAXFF	REPLICA	RIGID	SHOCK	SMTBQ	SPH
SPIN	SRD	TALLY	UEF	VORONOI	YAFF

★：GPU版のみ　　★★：バージョン2024.8のみ

5. 計算化学アプリケーション ＞ ジョブ実行方法（LAMMPSの場合）

バッチスクリプトの作成例

```
#!/bin/csh
#PBS -q APC
#PBS -N title
#PBS -l select=1:ncpus=8:mpiprocs=4

source /etc/profile.d/modules.csh
module load lammps/2408.1
setenv OMP_NUM_THREADS 2

cd $PBS_O_WORKDIR

mpirun -np 4 lmp_intel_cpu -sf intel -in input_file
```

スレッド数

MPIプロセス数

CPU版(INTELパッケージ利用)

- lmp_intel_cpu バイナリ
- -sf intel の指定
- 環境変数 OMP_NUM_THREADS にてスレッド数を指定
- mpirun –np # にてMPIプロセス数を指定
- 左図の場合
  - ✓スレッド数：2，MPIプロセス数：4
  - ✓使用する全コア数は 2x4 = 8　より ncpus=8

※ CPU版でもKOKKOSパッケージの利用は可能

```
#!/bin/csh
#PBS -q APG
#PBS -N title
#PBS -l select=1:ncpus=1:ngpus=1

source /etc/profile.d/modules.csh
module load lammps/2408.1

cd $PBS_O_WORKDIR

lmp_kokkos_gpu -k on g 1 -sf kk -pk kokkos -in input_file
```

GPU数

GPU版(KOKKOSパッケージ利用)

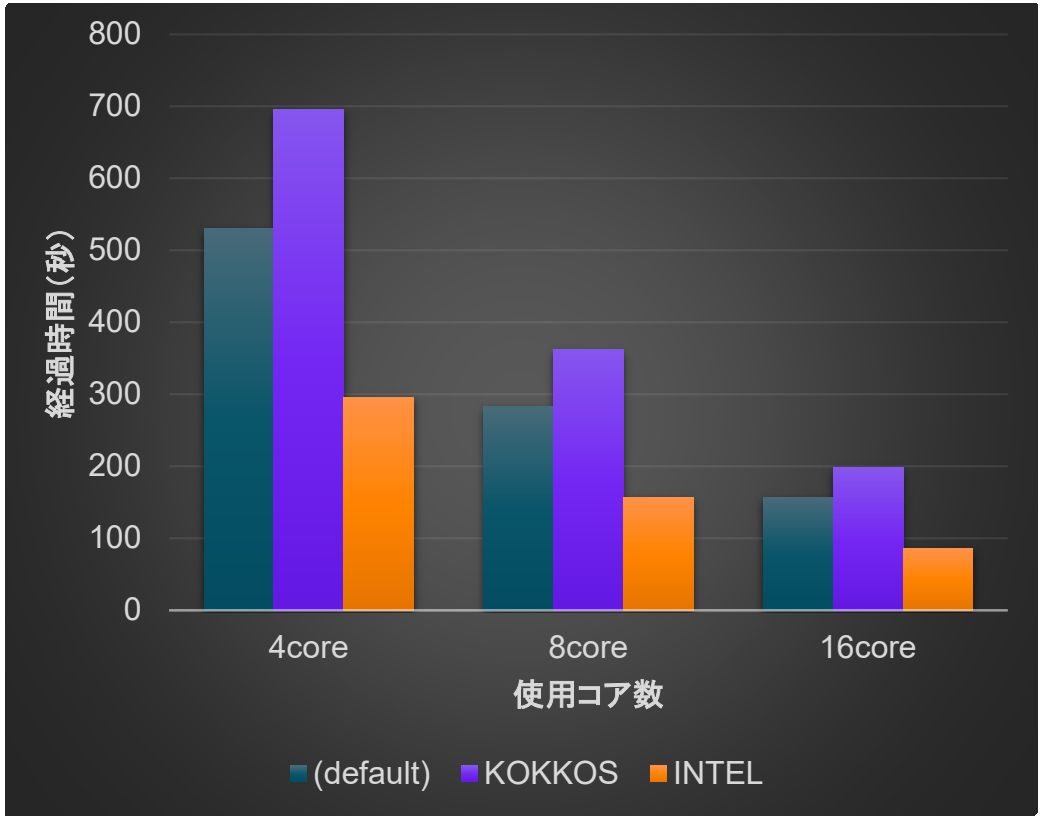
- lmp_kokkos_gpu バイナリ
- -sf kk –pk kokkos の指定
- -k on g # にてGPU数を指定（kはKOKKOSの意味）
- 左図の場合
  - ✓GPU数：1，全プロセス数：1 より ncpus=1

※ GPU版でもGPUパッケージの利用は可能
※ オプションの指定方法についてはオンラインマニュアルなどを参照ください

5. 計算化学アプリケーション ＞ ジョブ実行方法（LAMMPSの場合）

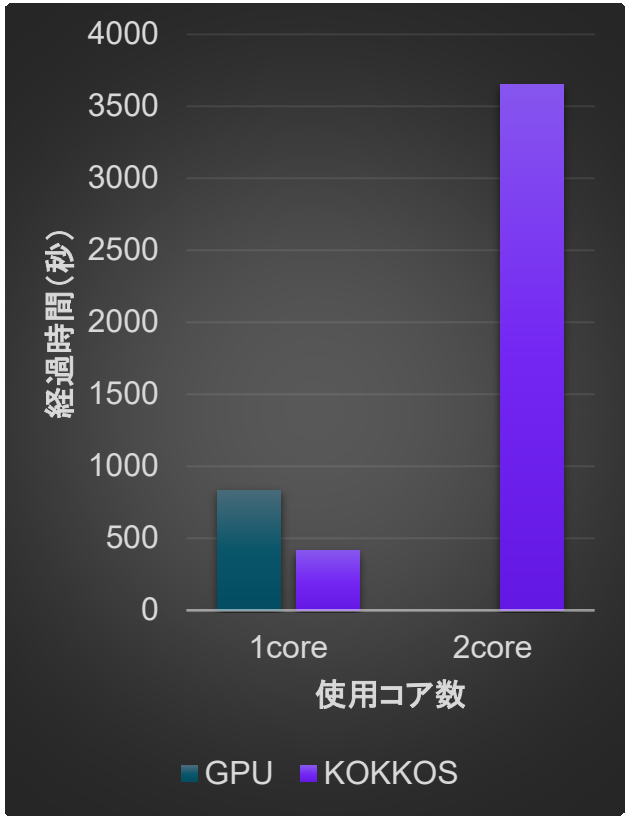
- ロドプシンを用いたMDシミュレーション
  (モデル：NPT, 温度：300K, タイムステップ：2fs, 計算ステップ：10,000 ⇒ 20ps のMD計算)

CPUを使用したベンチマーク



※ この例では、CPUとGPUの計算条件が異なるため、それらの経過時間を直接比較できません。

GPUを使用したベンチマーク



CPUのみの計算ではINTELパッケージの利用がお勧め

GPU使用時はKOKKOSパッケージの利用がお勧め

上記では近接リストの構築方法は、KOKKOSでは neigh half となっていますが、(default) や INTEL パッケージでは近接リストの構築方法の指定ができないため正確な比較はできません。あくまでも参考程度に参照ください。

演習　5. 計算化学（LAMMPS）

ゴール：　LAMMPSジョブの投入・状況確認の手順を確認する

Thank you