センター入門講習会

～高性能演算サーバ PRIMERGY CX400（tatara）～

2016年6月6日

この資料は以下のWebページからダウンロードできます．
https://www.cc.kyushu-u.ac.jp/scp/users/lecture/

並列プログラミング入門講習会のご案内

- スーパーコンピュータの性能を引き出すには・・・
  - **並列化**が不可欠！
- 並列プログラミング入門講習会を6/14,15に開催
  - 自動並列・OpenMP　6/14（火）
  - MPI　6/15（水）
- 受講特典
  - お試しアカウントを提供
- 応募方法
  - URLのページにて案内

https://www.cc.kyushu-u.ac.jp/scp/users/news/2016/375.html

本講習会のスケジュール

- 九州大学情報基盤研究開発センターのシステム
- 高性能演算サーバで利用可能なソフトウエア
- 高性能演算サーバへの接続
- 高性能演算サーバにおけるプログラムのコンパイル、実行

九州大学情報基盤研究開発センターのシステム

- 「全国共同利用」施設
  - システムは九州大学にあるが、利用対象者は全国の研究者
- 利用方法
  - 基本的には、個別に申請、支払い
  - 計算機利用法などの情報：
    https://www.cc.kyushu-u.ac.jp/scp/

ラインナップ

本講習会の対象

計算機名	Fujitsu FX10	Fujitsu CX400	HITACHI HA8000	HITACHI SR16000
				
ノード数	1,152	1,476	965	1
総CPUコア数	18,432	23,616	23,160	256
総理論性能	272TF	966TF	712TF	8.2TF
総主記憶容量（ノード当たり）	36TB (32GB)	185TB (128GB)	241TB (256GB)	16TB
アクセラレータ		NVIDIA Tesla K20m, K20Xm	Intel Xeon Phi 5110P	
ディスク容量	230TB	4PB	3.5PB	550TB
運用予定期間	2012.7-2017.9	2012.9-2017.9	2013.12-2017.9	2013.12-2017.9

理論演算性能

- 高性能演算サーバの理論演算性能　966 Tera FLOPS
  - 理論演算性能？
    - 計算機が持つ演算装置全てを同時に使用できた場合の、理論的なピーク性能
  - FLOPS?
    - FLoating point Operations Per Second の略
    - 1秒間に何回の実数計算を行えるか。

FLOPS ＝
プロセッサの周波数 x 同時実行可能演算数 x プロセッサ数

例）4演算同時実行可能な 1GHzのプロセッサ 1,000個によるシステム
⇒ 1GHz x 4 x 1,000 = 4,000GFLOPS = 4 Tera FLOPS

G: Giga(=10^9)
T: Tera(= 10^{12}),
P: Peta(= 10^{15}),
E: Exa(= 10^{18})

理論演算性能と実際の性能の違い

- 理論演算性能:
  全ての演算器が休むことなく働き続けることが前提

- 実際のプログラムの性能:
  様々な要因で演算器が休止
  - メモリからのデータ到着待ち
  - 他のプロセスの計算完了待ち
    - プロセス間の負荷の不均衡
  - 通信の完了待ち
  - ファイル入出力待ち

理論演算性能による比較は、ほとんど意味が無い

実際のプログラムの性能？

- プログラムによって傾向が変わる。
  例えば。。。
  - 仕事を複数のプロセッサに分担させるのが難しい計算：
    とても高速なプロセッサ 1個による計算機が有利
  - たくさんのプロセッサに分担させることができる計算：
    低速なプロセッサを多数搭載した計算機が有利

どのプログラムを使って比較するか？

Top500 List

- 最も有名な、スーパーコンピュータ性能比較リスト
  http://www.top500.org
  - 稼働中のスーパーコンピュータの上位500台を掲載。

- **LINPACKベンチマークプログラム**の性能で順位付け
  - 連立一次方程式の解を求める計算
  - 比較的、理論性能に近い性能が出る
    - キャッシュヒット率が高い、通信が少ない、負荷が均等

- 他の計算機との比較や傾向の分析などが容易
  - 1993年からほとんど同じ条件で更新を継続。
  - 世界中のほぼ全てのスーパーコンピュータが登録。

スーパーコンピュータ開発競争に利用

最新情報： 2015年11月

来月、更新予定

- **1位　Tianhe-2(China)　33.9 PFLOPS**
  2位　Titan (USA)　17.6 PFLOPS
  3位　Sequoia (USA)　17.1 PFLOPS
  4位　K Computer(Japan)　10.5 PFLOPS

- 国別合計:
  1位 USA　41.3%　(172.6 PFLOPS)
  2位 China　21.2%　(88.7 PFLOPS)
  3位 Japan　9.2%　(38.4 PFLOPS)
  4位 Germany　7.1%　(29.9 PFLOPS)
  5位 France　2.9%　(12.3 PFLOPS)
  - 以下, UK, Saudi Arabia, Switzerland, Korea, Italy, ...

QUARTETTO (= CX400 + HA8000)

Quadruple Technologies	Intel Ivy Bridge, Intel Sandy Bridge, NVIDIA Kepler, Intel Xeon Phi
Quadruple Technics	Kyushu University, Hitachi Ltd., Fujitsu Ltd., NVIDIA Co.
Total Peak Performance	1.5 PFLOPS
Total Memory	431 TiB
HPL Performance	**1.018PFLOPS**
Top500 Rank	**#77 (Nov. 2015)**

	Germany	12C 2.5GHz, Infiniband FDR Bull, Atos Group			
77	Research Institute for Information Technology, Kyushu University Japan	**QUARTETTO** - HA8000-tc HT210/PRIMERGY CX400 Cluster, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, NVIDIA K20/K20x, Xeon Phi 5110P Hitachi/Fujitsu	222,072	1,018.0	1,502.2
78	Texas Advanced Computing Center/Univ.	**Lonestar 5** - Cray XC40, Xeon E5-2690v3	30,048	1,017.0	1,250.0

アクセラレータ

- 特定の計算を、通常の CPUより低い電力で高速に処理できる装置
  - 通常、計算機の拡張スロットに増設して使用
  - 特に消費電力当たり性能の高さから、導入事例増加

- アクセラレータの違い
  - NVIDIA Tesla
    - GPUと共通の構造で一般の計算を行う "GPGPU"計算
    - CUDA等の専用言語でのプログラミングが必要
  - Intel Xeon Phi
    - 低速で低消費電力の CPUを多数並べて並列計算
    - 従来の並列プログラミングインタフェース（OpenMP等）を使用可能

本講習会のスケジュール

- 九州大学情報基盤研究開発センターのシステム
- 高性能演算サーバで利用可能なソフトウエア
- 高性能演算サーバへの接続
- 高性能演算サーバにおけるプログラムのコンパイル、実行

高性能演算サーバで利用可能なアプリケーション

高性能アプリケーションサーバでも利用可能なもの

ソフトウェア名		機能
Gaussian	o	非経験的分子軌道計算プログラム
CHARMM		生体高分子システムモデリングパッケージ
Molpro		非経験的分子軌道計算ソフトウェア
GAMESS	o	非経験的分子軌道法／密度汎関数理論計算プログラム
MSC.Marc		非線形構造解析プログラム
MSC Nastran, Patran		汎用構造解析有限要素法プログラム
ANSYS CFX, Fluent, ICEM, CFD, WorkBench		汎用有限体積法熱流体解析ソフトウェア
VASP	o	固体電子状態計算・バンド計算プログラム
SCIGRESS		古典分子動力学ソフトウェア
MATLAB		数値計算ソフトウェア
AMBER	o	モデリング／分子力学／動力学計算シミュレーションプログラム
IDL		データ解析・可視化プログラミング言語
WRF	o	メソスケール数値気候予測モデル

高性能演算サーバで利用可能なライブラリ

高性能アプリケーションサーバでも利用可能なもの

ライブラリ名		機能
SSL II		Fortran用汎用数値計算ライブラリ
C-SSL II		C, C++用汎用数値計算ライブラリ
LAPACK, BLAS	o	線形計算ライブラリ
ScaLAPACK	o	並列版線形計算ライブラリ
FFTW	o	高速フーリエ変換ライブラリ
NAG		数値計算ライブラリ
PETSc	o	偏微分方程式用数値計算ライブラリ
HDF5	o	階層型データフォーマットライブラリ

高性能演算サーバで利用可能なプログラミング環境

高性能アプリケーションサーバでも利用可能なもの

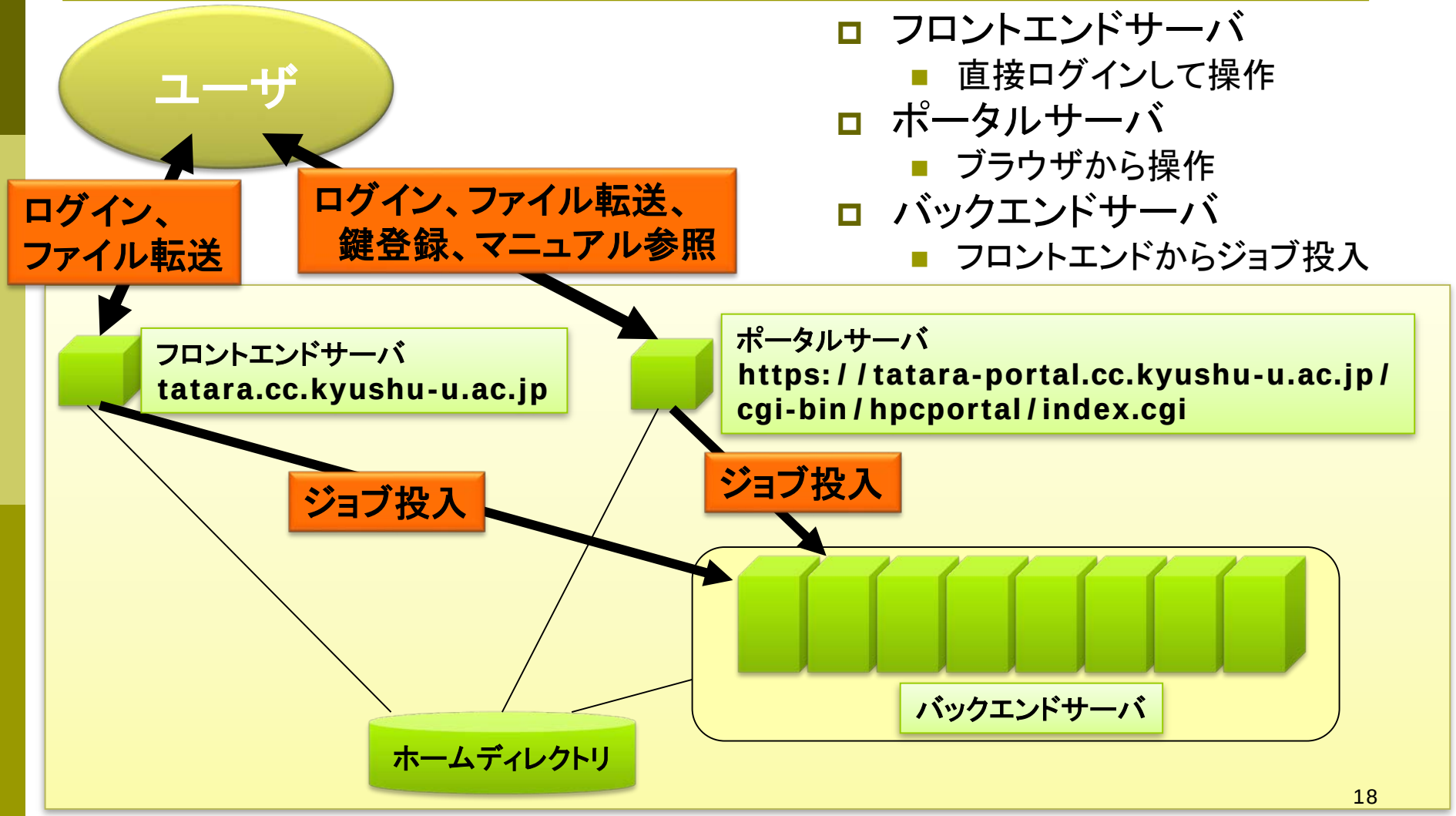
プログラミング環境名		機能
Technical Computing Suite		富士通製Fortran,C,C++コンパイラ
Intel Composer XE 2013	o	Intel製Fortran,C,C++コンパイラ
PGI Fortran		PGI製Fortranコンパイラ
Unified Parallel C	o	分散メモリ型並列C言語
Coarray Fortran	o	分散メモリ型並列Fortran
OpenMP	o	共有メモリ型並列プログラミングモデル
MPIライブラリ	o	分散メモリ型並列プログラミング用通信ライブラリ
CUDA		GPU向けC/C++言語開発環境
CUDA Fortran		GPU向けFortran開発環境
OpenACC		アクセラレータ向けの共通プログラミングインタフェース

本講習会のスケジュール

- 九州大学情報基盤研究開発センターのシステム
- 高性能演算サーバで利用可能なソフトウエア
- 高性能演算サーバへの接続
- 高性能演算サーバにおけるプログラムのコンパイル、実行

高性能演算サーバへのアクセス

- フロントエンドサーバ
  - 直接ログインして操作
- ポータルサーバ
  - ブラウザから操作
- バックエンドサーバ
  - フロントエンドからジョブ投入



高性能演算サーバの利用
ポータルサーバ経由

- ブラウザから以下にアクセス
  https://tatara-portal.cc.kyushu-u.ac.jp/cgi-bin/hpcportal/index.cgi

- 高性能演算サーバの
  ユーザ名とパスワードを入力

- ファイル操作、編集、
  コンパイル、ジョブ投入、
  各種マニュアル参照、等が可能
  - 詳細は、「HPC Portal」のマニュアルを参照

高性能演算サーバの利用
フロントエンドサーバ経由

- まず、自分の公開鍵をポータルサーバで登録
  - 以下のページを参照

https://www.cc.kyushu-u.ac.jp/scp/system/general/CX/how_to_use/01_login.html

- ターミナルソフトで tatara.cc.kyushu-u.ac.jp にログイン

- ログイン後は通常の Linuxサーバと同様の操作

実習 1
ログインまでの流れ

- 以下のページに従い、PuTTYでログインしてください

https://www.cc.kyushu-u.ac.jp/scp/system/general/CX/how_to_use/01_login.html

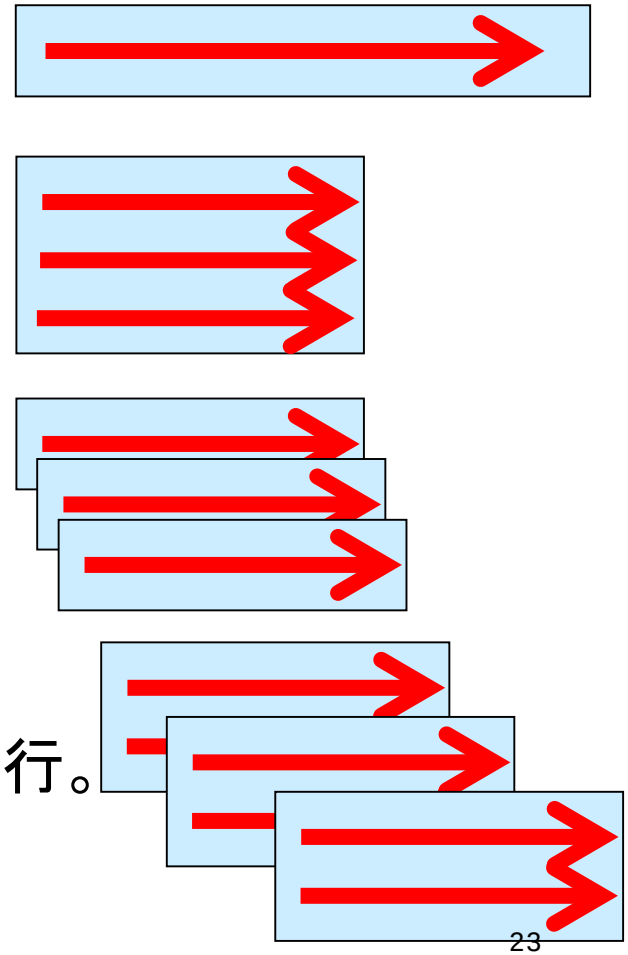
  - 鍵の作成
  - 鍵のアップロード
  - SSHでログイン

本講習会のスケジュール

- 九州大学情報基盤研究開発センターのシステム
- 高性能演算サーバで利用可能なソフトウエア
- 高性能演算サーバへの接続
- 高性能演算サーバにおけるプログラムのコンパイル、実行

プログラム実行の4形態

- 逐次
  - 従来の実行。1CPUコアを利用。
- スレッド並列
  - 共有メモリでの並列実行。
    計算ノード内での実行。
- MPI （プロセス並列）
  - 分散メモリでの並列実行。
    主に複数ノードで利用。
- ハイブリッド並列
  - 複数のスレッド並列をさらに並列に実行。
    主に複数ノードで利用。



主なコンパイルコマンド一覧

言語	逐次	スレッド並列	MPI	ハイブリッド並列
C	**fcc**	**fcc -Kparallel** **fcc -Kopenmp**	**mpifcc**	**mpifcc -Kparallel** **mpifcc -Kopenmp**
C++	**FCC**	**FCC -Kparallel** **FCC -Kopenmp**	**mpiFCC**	**mpiFCC -Kparallel** **mpiFCC -Kopenmp**
Fortran	**frt**	**frt -Kparallel** **frt -Kopenmp**	**mpifrt**	**mpifrt -Kparallel** **mpifrt -Kopenmp**

実際にコンパイルする際は、-Kfastオプションを付与することを推奨します。
C言語のスレッド並列(自動並列)プログラムのコンパイル例
$ fcc –Kfast –Kparallel sample.c
(-Kオプションは以下のようにカンマ区切りで並べることも可能)
$ fcc –Kfast,parallel sample.c

基本コマンド（逐次処理）

- コンパイル

  コンパイルコマンド　オプション　ソースプログラム

  例)

  ```
  tatara$ frt example.f90
  ```

  実行ファイル **a.out** を作成

  ```
  tatara$ frt -o example example.f90
  ```

  作成する実行ファイルの名前を example に変更

コンパイル時の主なオプション

Fortran

-c	オブジェクトファイルの作成までを行う
-o ファイル名	作成するファイル名を変更
-Free	自由形式
-Fixed	固定形式
-Kfast	最適化(推奨オプション)
-Kparallel	自動並列化
-Kopenmp	OpenMPによる並列化
-Haefosux	コンパイル時及び実行時に引数の整合性、添え字式、部分列式の値、未定義変数の参照、配列式の形状適合等を検査

コンパイル時のオプション
C, C++

-c	オブジェクトファイルの作成まで
-o ファイル名	作成するファイル名を変更
-Kfast	最適化(推奨オプション)
-Kparallel	自動並列化
-Kopenmp	OpenMPによる並列化
-Xg	GNU Cの言語仕様に基づいてコンパイル C99規格と同時に指定する場合、 -noansi も追加

数値計算ライブラリ

- 数値計算ライブラリとは？
  - 科学技術計算でよく用いられる計算（連立一次方程式の求解，固有値計算，FFT等）を集めたもの
  - C言語やFortranのプログラムから手続き呼び出しの形で利用できる
  - 一般に高速なアルゴリズムを採用し，さらに計算機に合わせたチューニングが適用されている

手軽に高速なプログラムを作成可能

高性能演算サーバで利用可能な数値計算ライブラリ

- BLAS
  - ベクトル演算や行列演算ライブラリ
- LAPACK
  - 線形代数ライブラリ
- ScaLAPACK
  - 線形代数メッセージパッシング並列ライブラリ
- SSL II, C-SSL II, SSL II/MPI
  - 線形計算、固有値固有ベクトル、非線形計算、極値問題、補間・近似、変換、数値微積分、微分方程式、特殊関数、疑似乱数 等のサブルーチン
- 利用法：以下のページを参照

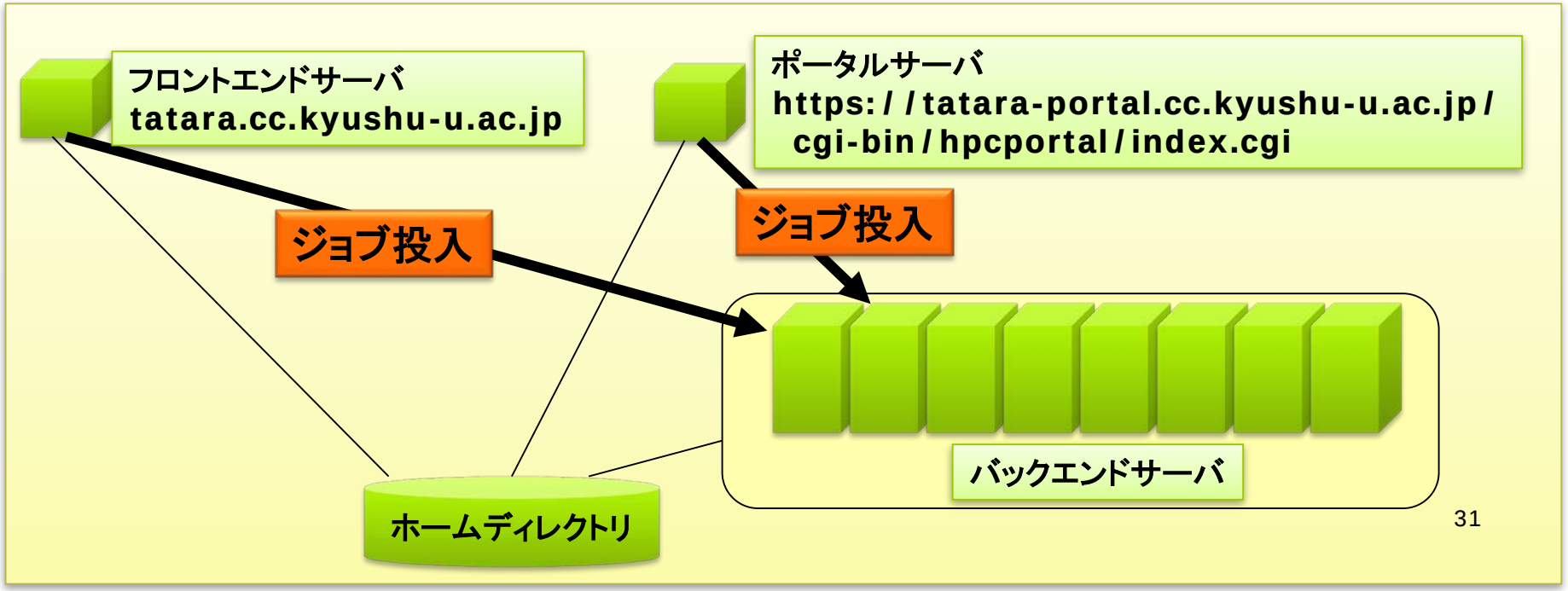
https://www.cc.kyushu-u.ac.jp/scp/system/general/CX/how_to_use/05_software.html

高性能演算サーバで利用可能なメモリ

- 1ノード当たり 113GB
  - 物理的な搭載メモリ量:
    128GB
  - 差分（15GB） = システムが使用する領域

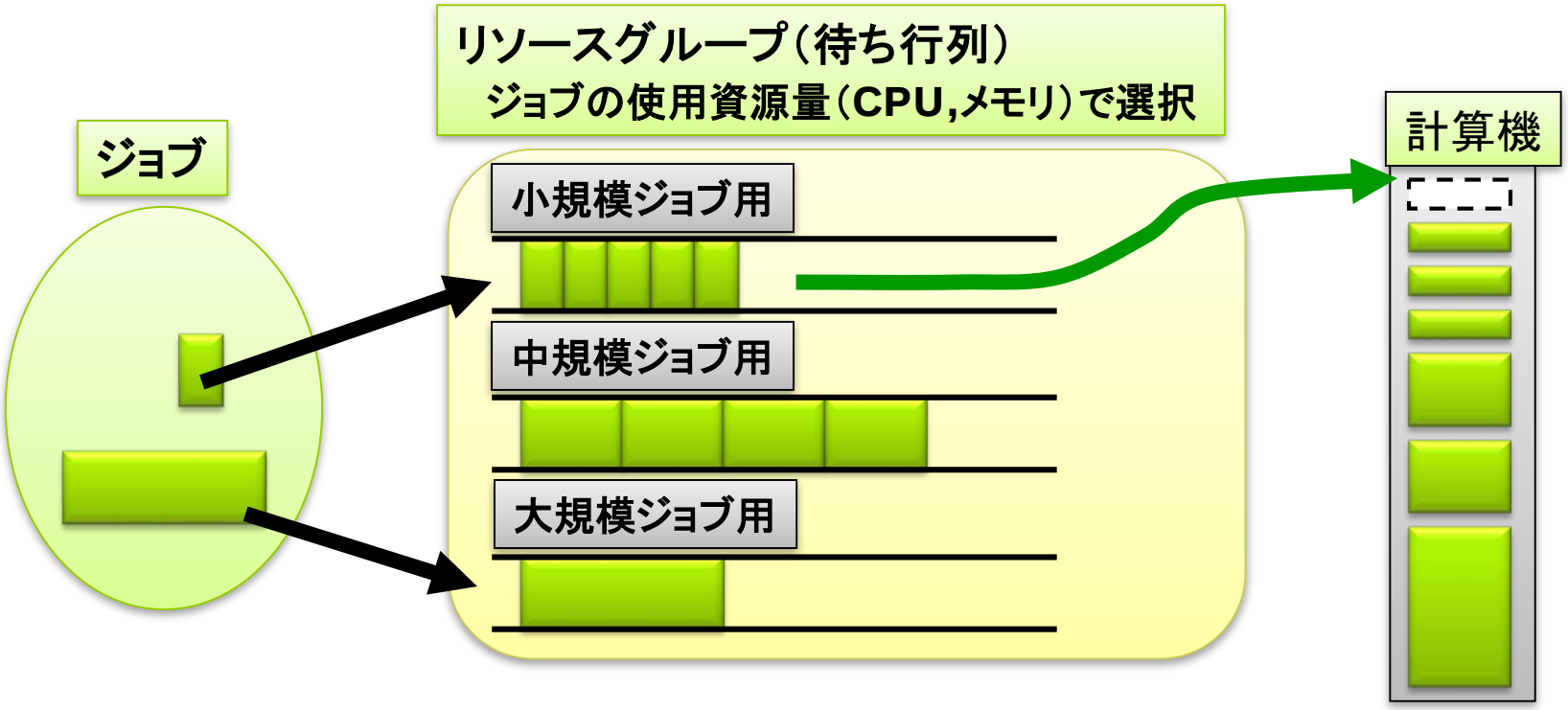
高性能演算サーバでのプログラムの実行

- フロントエンドサーバ上で直接実行
  - 制限：16コア、メモリ 2GB、計算時間 1時間
- バックエンドサーバへジョブとして投入



バッチシステムの仕組み

- 処理してほしい内容を記述したファイルを投入
  - ジョブとして受付
- 資源の空き状況に応じて順に処理される
  - 要求内容や空き状況によっては先を越されることも



バッチシステムの必要性

- 対話的な利用の限界：
  - 負荷が計算機で同時に処理可能な量を超えると，資源が空くまではコマンドの実行不可
  - 次にいつ資源が空くか不明

→ ジョブ実行要求を交通整理する仕組みが必要：バッチシステム

  - 要求された処理内容をジョブとして登録
  - 資源の空き状況に応じて自動的に実行開始

バッチ処理に用いるコマンド

- バッチジョブの投入
  pjsub
- バッチジョブの状況確認
  pjstat
- バッチジョブのキャンセル
  pjdel

ジョブの投入
pjsub

- 利用法:

pjsub オプション ジョブスクリプトファイル名

  - オプション:
    使用する資源等に関する指定
    - いつも同じ指定をするのであれば、ジョブスクリプトファイルの中に記述しても良い。(pjsubコマンドでの指定が優先)
  - ジョブスクリプトファイル:
    バッチシステムに依頼する処理の内容を記述したファイル
    - 詳細は後述
  - 例) test.sh という名前の
    ジョブスクリプトファイルを投入

```
tatara$ pjsub test.sh
[INFO] PJM 0000 pjsub Job 1234 submitted.
```

ジョブ ID

pjsubコマンドの主なオプション

オプション	説明
-o *filename*	標準出力ファイル名 (デフォルトの出力先：ジョブスクリプトファイル名.oジョブID)
-e *filename*	標準エラー出力ファイル名 (デフォルトの出力先：ジョブスクリプトファイル名.eジョブID)
-j	標準エラー出力と標準出力を同じファイルに書き出す
-L rscgrp=*name*	リソースグループ *name* にジョブを投入
-L elapse=*h:m:s*	ジョブの最大実行時間を *h*時間*m*分*s*秒に制限(h，mは省略可)
-L vnode=*limit*	使用する仮想ノード数の最大値
--mpi proc=*procs*	MPIのプロセス数
-s -S	ジョブの所要時間や使用メモリ量等の詳細情報を書き出す -S はノード毎の情報も合わせて出力 (デフォルトの出力先：ジョブスクリプトファイル名.iジョブID)
--no-stging	ステージング機能を使わない

投入可能なリソースグループ

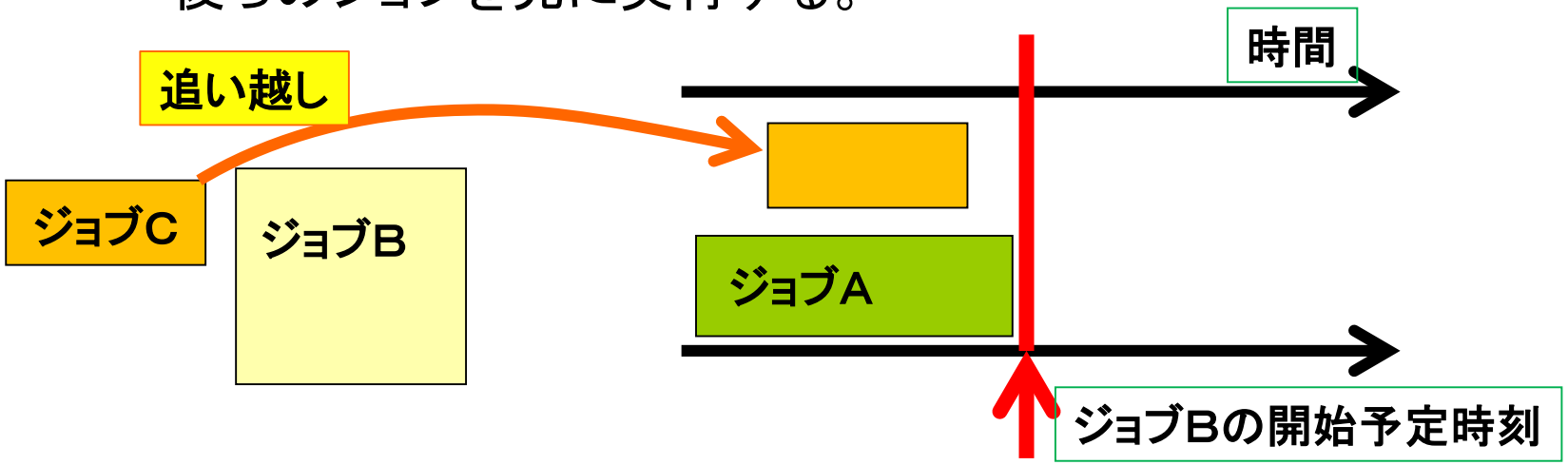
- 一般利用者として利用

リソースグループ名	ノード数	コア数	メモリサイズ	計算時間	備 考
cx-dbg	32	16×32	113GB×32	1時間	デバッグ専用
cx-single	1	16	113GB×1	1週間	シングルノードジョブ専用
cx-small	16	16×16	113GB×16	2日間	16ノードまで利用可能
cx-middle	64	16×64	113GB×64	1日間	64ノードまで利用可能
cx-large	128	16×128	113GB×128	12時間	128ノードまで利用可能
cx-g-dbg	16	16×16	113GB×16	1時間	Tesla K20m デバッグ専用
cx-g-single	1	16	113GB	1週間	Tesla K20m 1ノード
cx-g-small	16	16×16	113GB×16	2日間	Tesla K20m 16ノードまで
cx-gx-small	16	16×16	113GB×16	2日間	Tesla K20Xm 16ノードまで

ジョブの最大実行時間と「バックフィルスケジューリング」

- ジョブの最大実行時間
  - デフォルト値：リソースグループ毎の最大実行可能時間
  - （ジョブが完了する範囲で）出来るだけ短くすると、前のジョブを追い越して実行する可能性が上がる：「バックフィルスケジューリング」の効果

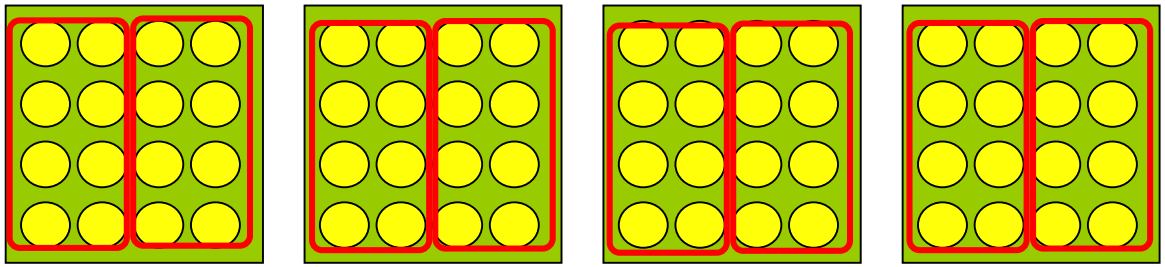
- バックフィルスケジューリング：
  - 前のジョブが資源不足で待たされている場合、そのジョブの開始時刻を遅らせないで実行できるなら後ろのジョブを先に実行する。



ノード数とプロセス数とスレッド数

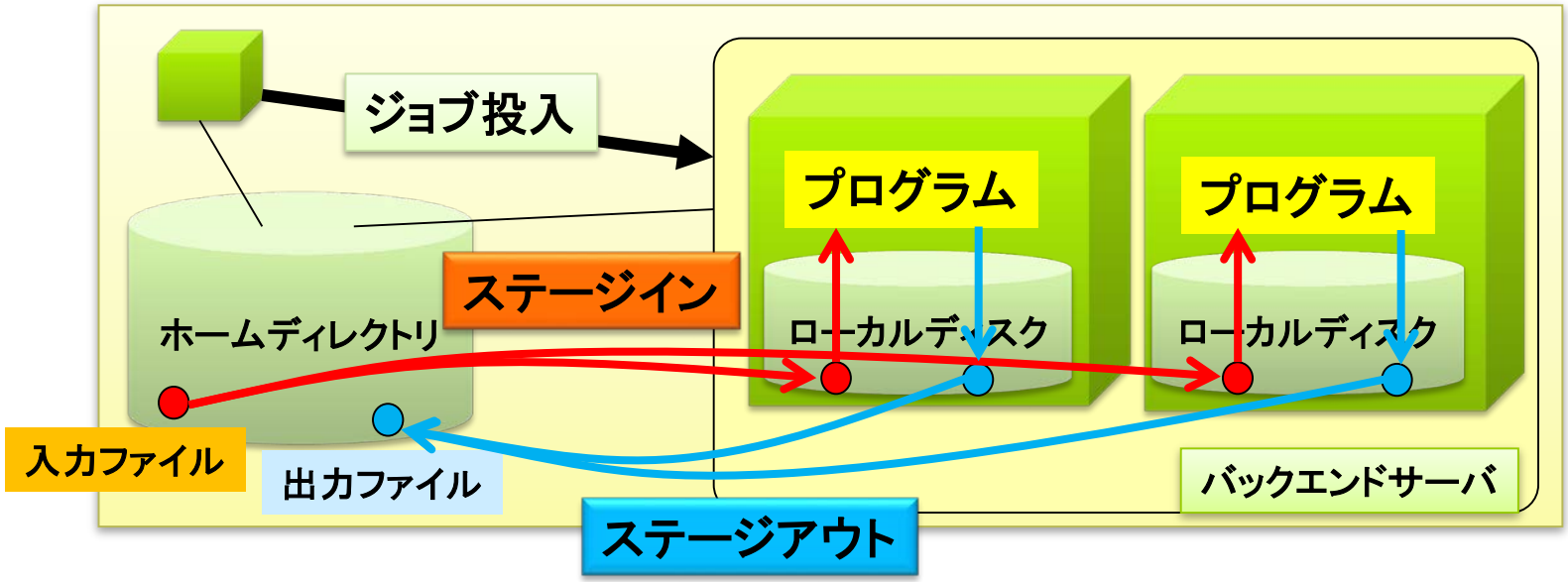
- プロセス数 x スレッド数 ≦ ノード数 * 16
  - デフォルト値:
    ノード数 = 1, プロセス数 = ノード数, スレッド数 = 16
  - プロセス数: MPIで並列実行する際に指定
    - ノード内で複数プロセス実行可能
  - スレッド数: 1プロセスで実行するスレッドの数
    - 最大 16スレッド

例）4ノードで 8プロセス（8スレッド/プロセス）を実行する場合



ステージング

- バックエンドサーバのローカルディスクを使用
  - 事前に入力ファイルをホームからコピー（ステージイン）
  - 終了後に出力ファイルをホームにコピー（ステージアウト）
  - コピーするファイルは --stgin, --stgoutオプションで指定
    （詳細は、ユーザガイドの「ジョブ運用ソフトウェアエンドユーザ向けガイド」を参照）
- ステージングの効果
  - ステージインとステージアウトの時間が必要
  - 同じファイルに対して何度も入出力を繰り返す場合に有効



ジョブの状況 pjstat

- 現在投入されているジョブの情報を表示
  - 表示内容:

```
$  pjstat

ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      0      0      0      1      0      0      0      0       1
s      0      0      0      0      1      0      0      0      0       1

JOB_ID     JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE
1032       test       NM RUN k70043a  10/19 11:44:17  0000:10:00 8
```

ジョブID　ジョブスクリプトファイル名　状態　ユーザID　実行開始日時　制限時間

  - オプションを指定しなければ，自分が投入したジョブのみ表示
  - --all オプション: 実行中の全ジョブを表示

ジョブのキャンセル pjdel

- 利用法:
  pjdel ジョブID
  - pjstatで表示されるジョブIDを指定
  - 例） 12345 というジョブIDのジョブをキャンセル

```
tatara$ pjdel 12345
```

ジョブスクリプトファイルの構成

- ジョブのオプション
  ＋ 通常のシェルスクリプト

```
#!/bin/sh

ジョブのオプション

実行コマンド
```

  - 注意：
    実行開始時のカレントディレクトリは
    ジョブ投入時のカレントディレクトリ

ジョブスクリプトファイルの例 (逐次処理)

```
#!/bin/sh

#PJM -L "vnode=1"
#PJM -L "vnode-core=1"
#PJM -L "rscgrp=cx-lecture"
#PJM -L "elapse=10:00"
#PJM --no-stging
#PJM -j
#PJM -X

./a.out
```

オプションは、pjsubコマンドに指定しても、ジョブスクリプトに記述しても良い

ジョブスクリプトファイルの例 (自動並列によるスレッド並列処理)

```
#!/bin/sh

#PJM -L "vnode=1"
#PJM -L "vnode-core=16"
#PJM -L "rscgrp=cx-lecture"
#PJM -L "elapse=10:00"
#PJM --no-stging
#PJM -j
#PJM -X

export PARALLEL=16
./a.out
```

自動並列の場合、スレッド数を環境変数 PARALLELで指定

ジョブスクリプトファイルの例 (OpenMPによるスレッド並列処理)

```
#!/bin/sh

#PJM -L "vnode=1"
#PJM -L "vnode-core=16"
#PJM -L "rscgrp=cx-lecture"
#PJM -L "elapse=10:00"
#PJM --no-stging
#PJM -j
#PJM -X

export OMP_NUM_THREADS=16
./a.out
```

OpenMPの場合、スレッド数を
環境変数 OMP_NUM_THREADSで指定

ジョブスクリプトファイルの例 (MPI並列処理)

```
#!/bin/sh

#PJM -L "vnode=64"
#PJM -L "vnode-core=1"
#PJM -L "rscgrp=cx-lecture"
#PJM -L "elapse=10:00"
#PJM --no-stging
#PJM -j
#PJM -X

mpiexec -n 64 ./a.out
```

プロセス数
（最大値＝ノード数 * コア数(16)）

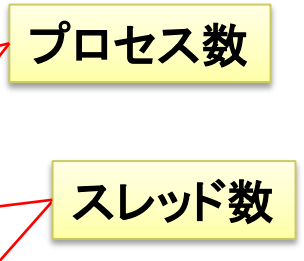
最大値より少なければ、各ノードに
　プロセス数 / ノード数
を割り当て.

ジョブスクリプトファイルの例 (ハイブリッド並列処理(MPI+自動並列))

```
#!/bin/sh

#PJM -L "vnode=16"
#PJM -L "vnode-core=4"
#PJM -L "rscgrp=cx-lecture"
#PJM -L "elapse=10:00"
#PJM --no-stging
#PJM -j
#PJM -X

export PARALLEL=4
mpiexec -n 16 ./a.out
```

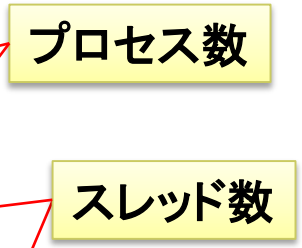


ジョブスクリプトファイルの例 (ハイブリッド並列処理(MPI+OpenMP))

```
#!/bin/sh

#PJM -L "vnode=16"
#PJM -L "vnode-core=4"
#PJM -L "rscgrp=cx-lecture"
#PJM -L "elapse=10:00"
#PJM --no-stging
#PJM -j
#PJM -X

export OMP_NUM_THREADS=4
mpiexec -n 16 ./a.out
```

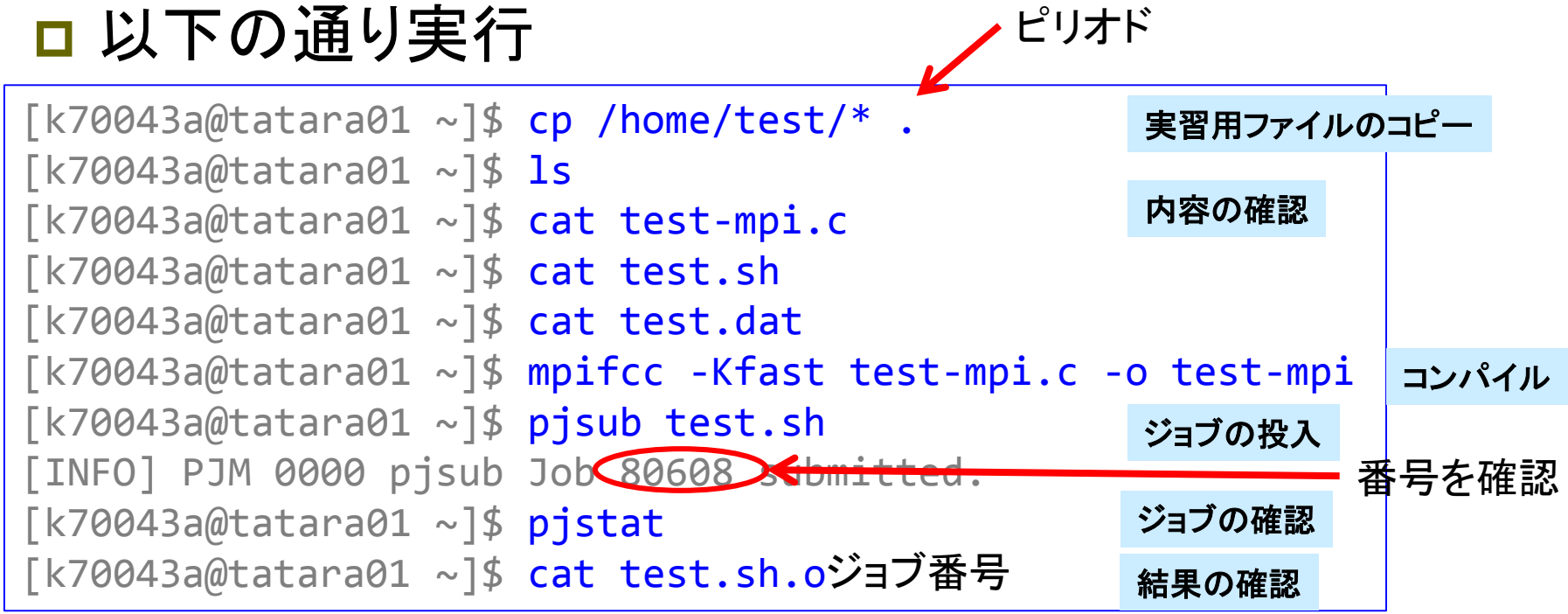


実習 2
高性能演算サーバにおけるプログラムの実行

- 高性能演算サーバにPuTTYでログイン
- サンプルプログラムとデータファイルをコピー
  - /home/test　配下のファイルをホームディレクトリにコピー
- サンプルプログラム test-mpi.c をコンパイル
- 作成された実行ファイルをジョブとして投入

直接ログインする場合の手順

- PuTTY でログイン
- 以下の通り実行



Intel コンパイラの利用法

- 準備
  - 以下を実行

```
$ source /home/etc/intel.sh
```

  - MPIを用いる場合、さらに、ホームディレクトリに .mpd.conf という名前のファイルを作成
    - ファイルの内容：以下の一行のみ
      - 文字列には、任意の文字列を記述

```
secretword=文字列
```

    - その後、.mpd.conf のアクセス権を以下により変更

```
$ chmod 600 ~/.mpd.conf
```

Intel コンパイラの利用法

- コンパイルコマンド

	言語	コマンド	自動並列	OpenMP
非MPI	Fortran	ifort	-parallel	-openmp
	C/C++	icc		
MPI	Fortran	mpiifort		
	C/C++	mpiicc		

- ジョブスクリプト
  - 多少書き換えが必要
  - MPI+OpenMPのスクリプト例
  - 詳細は webページ

```
#!/bin/bash
#PJM -L "rscgrp=cx-lecture"
#PJM -L "vnode=4"
#PJM -L "vnode-core=16"
#PJM -P "vn-policy=abs-unpack"
#PJM -L "elapse=10:00"
#PJM -j
#PJM -X
#PJM --no-stging

source /home/etc/intel.sh

export I_MPI_PERHOST=4
export I_MPI_FABRICS=shm:ofa
export I_MPI_PIN_DOMAIN=omp
export I_MPI_PIN_CELL=core
export OMP_NUM_THREADS=4
export KMP_STACKSIZE=8m
export KMP_AFFINITY=compact

mpdboot -n 4 -f ${PJM_O_NODEINF} -r /bin/pjrsh
mpiexec -n 16 ./a.out
mpdallexit
```

マニュアル、問い合わせ窓口

- ポータルサーバにログイン後、以下のマニュアルを閲覧可能
  - HPCポータルの操作方法
  - プログラミング言語，ライブラリ
    - C/C++, Fortran, MPI, SSL II, XP Fortran, BLAS等
    - プログラミング支援ツール、デバッガ、プロファイラ
  - アプリケーション
    - IDL, Nastran, Patran
  - ユーザガイド
    - ジョブ運用ソフトウェア利用法
- メールによる問い合わせ窓口：
  request@iii.kyushu-u.ac.jp