FX10 スーパーコンピュータシステムの利用方法について

システム運用係

既に、スーパーコンピューティングニュース、Web ページでもお知らせしている通り、2012 年 4 月 2 日 (月) 10:00 より FX10 スーパーコンピュータシステムの試験運転を開始します。

本稿では、FX10 スーパーコンピュータシステムの利用方法について簡単にご説明します (記載している内容は、原稿執筆時によるもので実際の運用時とは異なる場合があります)。利用方法の詳細については、利用支援ポータル・Web ページにてお知らせいたします。

なお、試験運転期間中は、システムの設定変更等のため、予告なく運転の停止、運用仕様の変更を行う場合がありますので、予めご了承ください。

1. ログインノードの利用

FX10 スーパーコンピュータシステムでは、ログインノード、インタラクティブノード、計算ノードで、それぞれの特徴に合わせた計算サービスを行っています。

ログインノードは、プログラムの編集・コンパイル、バッチジョブ投入などの利用環境を提供しています。

1.1 鍵登録

FX10 スーパーコンピュータは、公開鍵認証を行っているため、利用に先立ち公開鍵登録を行う必要があります。公開鍵登録は、利用支援ポータル (https://oakleaf-www.cc.u-tokyo.ac.jp) で行います。

利用支援ポータルには、本センターから利用登録時に通知された利用者番号 (ユーザ名) とパスワード (登録する SSH 公開鍵ではありません) を用いて接続します。接続 (認証) が成功すると、「SSH 公開鍵登録」、「パスワード変更」などのメニュー画面が表示されますので、「SSH 公開鍵登録」を選択、公開鍵を登録してください。

1.2 接続先 (ログインノードのホスト名一覧)

FX10 スーパーコンピュータのログインノードは全部で 6 台用意しています。ホスト名は、以下の通りです (表 1)。

表 1. 接続ホスト名一覧

ホスト名	oakleaf-fx.cc.u-tokyo.ac.jp ※以下のホストの何れかに接続します。 また、どのホストに接続しても同じ環境です。負荷分散にご協力ください。 oakleaf-fx-1.cc.u-tokyo.ac.jp oakleaf-fx-2.cc.u-tokyo.ac.jp oakleaf-fx-3.cc.u-tokyo.ac.jp oakleaf-fx-4.cc.u-tokyo.ac.jp oakleaf-fx-5.cc.u-tokyo.ac.jp oakleaf-fx-6.cc.u-tokyo.ac.jp
接続方法	SSH Protocol Version 2
認証方式	鍵による認証 (センター発行のパスワードは SSH ログインには使用しません) 初回は Web による鍵登録が必要です (公開鍵登録は利用支援ポータルで実施)

2. コンパイル

FX10 スーパーコンピュータには、富士通社製コンパイラ (Fortran 77/90、C、C++) と GCC、g95 を用意しています。

ログインノードと計算ノード・インタラクティブノードでは、ハードウェアアーキテクチャが異なる (バイナリ互換がない) ため、プログラムを実行する計算機にあわせてコンパイルする必要があります。そのため、それぞれの言語について、ログインノードではクロスコンパイル環境 (コマンド) を用意しています。

各コンパイルコマンドの詳細については、「2.1 コンパイルコマンド」をご覧ください。

表 2.　エンディアンの違い

	エンディアン	備考
ログインノード	リトルエンディアン	クロスコンパイル環境を提供
計算・インタラクティブノード	ビッグエンディアン	

2.1 コンパイルコマンド

FX10 スーパーコンピュータでコンパイルするコマンドは以下の表の通りです (表 3)。

ログインノードで計算ノード・インタラクティブノード実行ができるバイナリを生成する場合には、クロスコンパイラを使用します。また、バッチジョブやインタラクティブジョブ中でコンパイルする場合にはオウンコンパイラを使用します。

表 3. コンパイルコマンド一覧

言語		クロスコンパイラ (ログインノードで使用)	オウンコンパイラ (計算・インタラクティブノードで使用)
Fortran 77/90		frtpx	frt
	MPI	mpifrtpx	mpifrt
C		fccpx	fcc
	MPI	mpifccpx	mpifcc
C++		FCCpx	FCC
	MPI	mpiFCCpx	mpiFCC

2.2 Fortran 77/90 言語

FX10 スーパーコンピュータで使用できる富士通社製 Fortran 77/90 コンパイラの仕様は以下の通りです (表 4)。また、自動並列化、OpenMP・MPI 実行する場合などは、以下のオプションを指定してください (表 5)。なお、コンパイル時に使用できる最適化オプションなどの詳細については、利用者マニュアルを参照してください。

表 4. 言語仕様 (Fortran 77/90 言語)

言語	コマンド名	規格
Fortran77/90	frtpx (frt)	JIS X 3001-1982 ISO/IEC 1539-1980 JIS X 3001-1:1994 ISO/IEC 1539:1991 JIS X 3001-1:1998 ISO/IEC 1539-1997 JIS X 3001-1:2009 (Fortran 2003 規格) の一部 OpenMP API V3.0 仕様

表 5. Fortran コンパイルコマンド

	クロスコンパイラ (ログインノード)	オウンコンパイラ
逐次実行	frtpx -Kfast	frt -Kfast
自動並列化	frtpx -Kfast , parallel	frt -Kfast , parallel
スレッド並列化 (OpenMP)	frptx -Kfast , openmp	frp -Kfast , openmp
MPI	mpifrtpx -Kfast	mpifrt -Kfast
自動並列化 + MPI	mpifrtpx -Kfast , parallel	mpifrt -Kfast , parallel
OpenMP + MPI	mpifrtpx -Kfast , openmp	mpifrt -Kfast , openmp

2.3 C、C++ 言語

FX10 スーパーコンピュータで使用できる富士通社製 C、C++ コンパイラの仕様は以下の通りです (表 6)。また、自動並列化、OpenMP・MPI 実行する場合などは、以下のオプションを指定してください (表 7、表 8)。なお、コンパイル時に使用できる最適化オプションなどの詳細については、利用者マニュアルを参照してください。

表 6. 言語仕様 (C、C++ 言語)

言語	コマンド名	規格
C	fccpx (fcc)	ISO/IEC 9899:1990 ISO/IEC 9899:1999 OpenMP API V3.0 仕様
C++	FCCpx (FCC)	ISO/IEC 14882:12003 (export キーワードは除く) OpenMP API V3.0 仕様

表 7. C コンパイルコマンド

	クロスコンパイラ (ログインノード)	オウンコンパイラ
逐次実行	fccpx -Kfast	fcc -Kfast
自動並列化	fccpx -Kfast , parallel	fcc -Kfast , parallel
スレッド並列化 (OpenMP)	fcctx -Kfast , openmp	fcc -Kfast , openmp
MPI	mpifccpx -Kfast	mpifcc -Kfast
自動並列化 + MPI	mpifccpx -Kfast , parallel	mpifcc -Kfast , parallel
OpenMP + MPI	mpifccpx -Kfast , openmp	mpifcc -Kfast , openmp

表 8. C++ コンパイルコマンド

	クロスコンパイラ (ログインノード)	オウンコンパイラ
逐次実行	FCCpx -Kfast	FCC -Kfast
自動並列化	FCCpx -Kfast , parallel	FCC -Kfast , parallel
スレッド並列化 (OpenMP)	FCCpx -Kfast , openmp	FCC -Kfast , openmp
MPI	mpiFCCpx -Kfast	mpiFCC -Kfast
自動並列化 + MPI	mpiFCCpx - Kfast , parallel	mpiFCC - Kfast , parallel
OpenMP + MPI	mpiFCCpx - Kfast , openmp	mpiFCC - Kfast , openmp

3. バッチジョブの実行

FX10 スーパーコンピュータでは、インタラクティブジョブ、バッチジョブの実行が行えます。主なジョブ操作コマンドは、以下の通りです (表 9)。

表 9. ジョブ操作コマンド

コマンド名	操作・説明	HA8000、SMP コマンド
pjsub	インタラクティブジョブ、バッチジョブの投入を行う ○ 指定方法 [z30000@oakleaf-fx-1 ~]$ **pjsub オプション スクリプト** ○ インタラクティブジョブの実行 [z30000@oakleaf-fx-1 ~]$ **pjsub --interact** ○ ノード数の指定方法 (例では 96 Node 指定) [z30000@oakleaf-fx-1 ~]$ **pjsub -L "node=96"** または、スクリプト内に以下を記述 **#PJM -L "node=96"** ○ 経過時間の指定方法 (例では 1 時間を指定) [z30000@oakleaf-fx-1 ~]$ **pjsub -L "elapse=1:00:00"** または、スクリプト内に以下を記述 **#PJM -L "elapse=1:00:00"**	qsub
pjdel	インタラクティブジョブ、バッチジョブの削除を行う [z30000@oakleaf-fx-1 ~]$ **pjdel ジョブ ID**	qdel
pjstat	インタラクティブジョブ、バッチジョブの状態確認を行う	qstat

3.1 インタラクティブジョブ実行

インタラクティブジョブを実行する場合には、pjsub コマンドに --interact オプションを指定し、実行します。

```
[z30000@oakleaf-fx-1 ~]$ pjsub --interact                pjsub コマンドでインタラクティブジョブを起動
[INFO] PJM 0000 pjsub Job 12345 submitted.               ジョブ ID は "12345" となる
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 12345 started.
[z30000@e10-087 ~]$                                      インタラクティブジョブの起動

 インタラクティブ利用 (コンパイルや debug 処理など)

[z30000@e10-087 ~]$ exit                                 exit コマンドでインタラクティブジョブの終了
exit
[INFO] PJM 0083 pjsub Interactive job 12345 completed.
[z30000@oakleaf-fx-1 ~]$
```

図 1. インタラクティブジョブの実行例 (1)

```
[z30000@oakleaf-fx-1 ~]$ pjsub --interact -L "node=12"   インタラクティブジョブ (12 Node) を起動
[INFO] PJM 0000 pjsub Job 23456 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 23456 started.
[z30000@e10-087 ~]$ cat mpi.f                            MPI プログラムの確認
  program sample
  include 'mpif.h'

  (中略)

  call MPI_FINALIZE(ierr)
  stop
  end
[z30000@e10-087 ~]$ mpifrt mpi.f                         MPI プログラムのコンパイル (オウンコンパイラ)
[z30000@e10-087 ~]$ mpiexec ./a,out                      MPI プログラムの実行

      実行結果の表示

[z30000@e10-087 ~]$ exit                                 exit コマンドでインタラクティブジョブの終了
exit
[INFO] PJM 0083 pjsub Interactive job 23456 completed.
[z30000@oakleaf-fx-1 ~]$
```

図 2. インタラクティブの実行例 (2)

3.2 バッチジョブ実行

バッチジョブを投入するには、pjsub コマンドを実行します。ここでは、スクリプトファイルを作成してバッチジョブを実行する例を示します。

ジョブの実行形態	例示
逐次ジョブ (1 ノードあたり 1 プロセス実行)	図 3 を参照
スレッド並列ジョブ実行 (1 ノードあたり 1 プロセス、複数スレッド)	図 4 を参照
MPI ジョブ実行 (1 ノードあたり 1 プロセス実行)	図 5 を参照
MPI ジョブ実行 (1 ノードあたり複数プロセス実行)	図 6 を参照
ハイブリッド並列ジョブ実行 (1 ノードあたり複数プロセス、複数スレッド実行)	図 7 を参照

3.2.1 バッチジョブスクリプト例

3.1 にある、主な利用形態ごとのバッチジョブスクリプト例を示します。

```
[z30000@oakleaf-fx-1 ~]$ cat job.sh                バッチジョブスクリプトの確認
#!/bin/sh
#PJM -L "node=1"                                   使用ノード数を "1 ノード" 指定
#PJM -L "elapse=1:00:00"                           経過時間制限値 を "1:00:00 (1 時間)" 指定
#PJM -j                                            標準エラー出力を標準出力へマージ

./a.out                                            a.out プログラムの実行

[z30000@oakleaf-fx-1 ~]$ pjsub job.sh              バッチジョブの投入
[INFO] PJM 0000 pjsub Job 4321 submitted.          ジョブ ID = 4321 でバッチジョブが投入された
[z30000@oakleaf-fx-1 ~]$
```

図 3. バッチジョブの実行例 (1) (逐次ジョブ)

```
[z30000@oakleaf-fx-1 ~]$ cat job.sh                バッチジョブスクリプトの確認
#!/bin/sh
#PJM -L "node=1"                                   使用ノード数を "1 ノード" 指定
#PJM -L "elapse=1:00:00"                           経過時間制限値 を "1:00:00 (1 時間)" 指定
#PJM -j                                            標準エラー出力を標準出力へマージ

export OMP_NUM_THREADS=16                          スレッド数の設定 (OMP_NUM_THREADS,
export PARALLEL=16                                 PARALLEL の指定は必須)
./a.out                                            a.out プログラムの実行

[z30000@oakleaf-fx-1 ~]$ pjsub job.sh
```

図 4. バッチジョブの実行例 (2) (スレッド並列ジョブ)

```
[z30000@oakleaf-fx-1 ~]$ cat job.sh                バッチジョブスクリプトの確認
#!/bin/sh
#PJM -L "node=12"                                  使用ノード数を "12 ノード" 指定
#PJM -L "elapse=1:00:00"                           経過時間制限値 を "1:00:00 (1 時間)" 指定
#PJM -j                                            標準エラー出力を標準出力へマージ

mpiexec ./a.out                                    mpiexec コマンドを使用してプログラムの実行

[z30000@oakleaf-fx-1 ~]$ pjsub job.sh
```

図 5. バッチジョブの実行例 (3) (MPI ジョブ)

```
[z30000@oakleaf-fx-1 ~]$ cat job.sh                バッチジョブスクリプトの確認
#!/bin/sh
#PJM -L "node=12"                                  使用ノード数を "12 ノード" 指定
#PJM -L "elapse=1:00:00"                           経過時間制限値 を "1:00:00 (1 時間)" 指定
#PJM --mpi "proc=192"                              プロセス数の指定 (例では 192 プロセス)
#PJM -j                                            標準エラー出力を標準出力へマージ

mpiexec ./a.out                                    mpiexec コマンドを使用してプログラムの実行

[z30000@oakleaf-fx-1 ~]$ pjsub job.sh
```

図 6. バッチジョブの実行例 (4)(MPI ジョブ)

```
[z30000@oakleaf-fx-1 ~]$ cat job.sh                バッチジョブスクリプトの確認
#!/bin/sh
#PJM -L "node=12"                                  使用ノード数を "12 ノード" 指定
#PJM -L "elapse=1:00:00"                           経過時間制限値 を "1:00:00 (1 時間)" 指定
#PJM --mpi "proc=24"                               プロセス数の指定 (24 プロセス × 8 スレッド)
#PJM -j                                            標準エラー出力を標準出力へマージ

export OMP_NUM_THREADS=8                           スレッド数の設定 (OMP_NUM_THREADS,
export PARALLEL=8                                  PARALLEL の指定は必須)
mpiexec ./a.out                                    mpiexec コマンドを使用してプログラムの実行

[z30000@oakleaf-fx-1 ~]$ pjsub job.sh
```

図 7. バッチジョブの実行例 (5)(ハイブリッド並列ジョブ)

3.2.2 バッチジョブの実行結果ファイル

バッチジョブを実行すると、バッチジョブの標準出力 (-o 指定で出力ファイルの指定)、標準エラー出力 (-e 指定で出力ファイルの指定)、統計情報 (-s 指定) を出力することが出来ます。

標準出力	ジョブ名.o ジョブ ID	(例：test.sh.o12345)	、(例：STDIN.o98765)
標準エラー出力	ジョブ名.e ジョブ ID	(例：test.sh.e12345)	、(例：STDIN.e98765)
統計情報	ジョブ名.i ジョブ ID	(例：test.sh.i12345)	、(例：STDIN.i98765)

なお、シェルスクリプトを使用しないバッチジョブを投入した場合には、STDIN.～ となります。

3.2.3 バッチジョブで設定される主な環境変数

バッチジョブ実行環境で設定される主な環境変数は以下の通りです (表 10)。

表 10. 設定される主な環境変数

環境変数名	説明
PJM_ENVIRONMENT	ジョブ種別を示す。BATCH (バッチジョブ)、INTERACT (インタラクティブジョブ)
PJM_JOBNAME	ジョブ名
PJM_JOBID	ジョブ ID
PJM_O_HOME	ホームディレクトリ
PJM_O_HOST	ジョブ投入ホスト名
PJM_O_WORKDIR	ジョブ投入ディレクトリ

3.2.4 その他、注意事項

本稿では、基本的な利用方法 (簡単なバッチジョブ投入方法等) についてのみ記載しております。また、FX10 スーパーコンピュータシステムでは、ローカルファイルシステムを利用するためのジョブステージング機能や、ジョブの実行ノード形状 (1 次元・2 次元・3 次元) などを指定することも可能となっています (この場合、どの計算ノードでプロセスを生成するかなどの、細かな指定 (設定) が必要となります)。

これら、FX10 スーパーコンピュータ利用の上での基本的な動作、チューニングに関する事項・詳細等については、今後利用支援ポータルなどでお知らせしていく予定です。

3.3 バッチジョブの削除

バッチジョブを削除するには、pjdel コマンドを実行します (図 8)。

3.4 バッチジョブの状態確認、参照

バッチジョブの状態 (実行待ち、実行中など) を確認、参照するには、pjstat コマンドを実行します (図 9)。

```
[z30000@oakleaf-fx-1 ~]$ pjsub job.sh                    バッチジョブの投入
[INFO] PJM 0000 pjsub Job 5432 submitted.                ジョブ ID = 5432 でバッチジョブが投入された
[z30000@oakleaf-fx-1 ~]$
[z30000@oakleaf-fx-1 ~]$ pjstat                          pjstat コマンドでジョブ ID の確認
  (省略)
[z30302@oakleaf-fx-1 ~]$ pjdel 5432                      pjdel コマンドでバッチジョブ (5432) の削除
[INFO] PJM 0100 pjdel Job 5432 canceled.                 バッチジョブの削除メッセージが出力される
[z30000@oakleaf-fx-1 ~]$
[z30000@oakleaf-fx-1 ~]$ pjstat                          pjstat コマンドで削除されたことを確認
```

図 8. バッチジョブの削除例

```
[z30000@oakleaf-fx-1 ~]$ pjsub a.sh                                          バッチジョブの投入
[z30000@oakleaf-fx-1 ~]$ pjstat                                              状態確認
  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL
       0      1      0      0      1      0      0      0      0       2
s      0      1      0      0      1      0      0      0      0       2

JOB_ID      JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE  バッチジョブの実行を
1234        a.sh       NM RUN z30000   02/28 13:13:42  0048:00:00 4             確認
1235        a.sh       NM QUE z30000   02/29 09:00:00  0048:00:00 192
```

【 各項目の説明 】

pjstat コマンドで表示されている内容の主な内容は以下の通り。なお、現在開発中のため運用開始時には表示内容等が変更になっている場合があります。

JOB_ID	ジョブ ID
JOB_NAME	ジョブ名
ND	ジョブモデル (NM: 通常ジョブ、ST: ステップジョブ)
ST	ジョブ状態 (ACC: 受付状態、QUE: 実行待ち、RUN: 実行中)
USER	利用者番号
START_DATE	ジョブ実行開始時刻。ジョブが実行前の場合にはジョブ実行開始予測時刻
ELAPSE_LIM	経過時間制限値
NODE_REQUIRE	ジョブ投入時の指定ノード数

```
[z30000@oakleaf-fx-1 ~]$ pjstat -A                                           -A オプションを付加
  ACCEPT QUEUED  STGIN  READY RUNING RUNOUT STGOUT   HOLD  ERROR   TOTAL   (システム全体の状態を
       0      0      0      0      4      0      0      0      0       5     表示)
s      0      0      0      0      4      0      0      0      0       5

JOB_ID      JOB_NAME   MD ST  USER     START_DATE      ELAPSE_LIM NODE_REQUIRE
1111        ********** NM RUN ******   02/27 09:22:55  0024:00:00 1             他の利用者情報も表示
1222        ********** NM RUN ******   02/27 12:22:55  0024:00:00 1             される (利用者番号等
1233        ********** NM RUN ******   02/27 23:22:55  0024:00:00 1             は表示されない (表示
1234        a.sh       NM RUN z30000   02/28 13:13:42  0048:00:00 4             が * となる))
1235        a.sh       NM QUE z30000   02/29 09:00:00  0048:00:00 192
[z30000@oakleaf-fx-1 ~]$
```

図 9. バッチジョブの状態確認 (イメージ)

4. マニュアル

FX10 スーパーコンピュータのマニュアル (言語、チューニングマニュアル、利用の手引きなど) については、利用支援ポータルにて参照可能とする予定です。主な言語系のマニュアルについては、以下の表を参照してください (表 11)。

表 11. 主なマニュアル (言語系)

Fortran、C、C++、 XPFortran TOOL 関係	Fortran 文法書
	Fortran 使用手引書
	Fortran 翻訳時メッセージ
	C 言語使用手引書
	C++ 言語使用手引書
	C/C++ 最適化メッセージ説明書
	XPFortran 使用手引書
	Fortran/C/C++ 実行時メッセージ
	実行時情報出力機能使用手引書
	プログラミング支援ツール使用手引書
	プロファイラ使用手引書
	高速4倍精度基本演算ライブラリ使用手引書
SSLⅡ (スレッド並列版含む)	SSLⅡ サブルーチン一覧
	SSLⅡの使い方
	ユーザプログラムの翻訳・結合・実行のしかた
	富士通 SSLⅡ 使用手引書 (科学用サブルーチンライブラリ)
	FUJITSU SSLⅡ 拡張機能使用手引書 (科学用サブルーチンライブラリ)
	FUJITSU SSLⅡ 拡張機能使用手引書Ⅱ
	FUJITSU SSLⅡスレッド並列機能 使用手引書 (科学用サブルーチンライブラリ)
	SSLⅡ サブルーチン一覧
C-SSLⅡ (スレッド並列版含む)	C-SSL Ⅱの使い方
	ユーザプログラムの翻訳・結合・実行のしかた
	C-SSL Ⅱ 使用手引書 (科学用関数ライブラリ)
	C-SSL Ⅱスレッド並列機能使用手引書 (科学用関数ライブラリ)
SSL Ⅱ/MPI	ユーザプログラムの翻訳・結合・実行のしかた
	SSL Ⅱ/MPI 使用手引書 (科学用サブルーチンライブラリ)
MPI	MPI 使用手引書

5. 利用相談窓口

FX10 スーパーコンピュータの利用に関する窓口については、従来の 相談窓口 (soudan@cc.u-tokyo.ac.jp) ではなく、専用の窓口を用意する予定です (ベンダーを含めた相談窓口を予定)。メールアドレスなどの詳細が決まり次第お知らせいたします。

FX10 スーパーコンピュータの利用を行う上での基本操作 (鍵登録、ログイン、コンパイル、バッチジョブの実行) について簡単に説明してきました。**詳細については、現在サービス環境の開発中につき変更される可能性があります。**

繰り返しになりますが、最新の情報等については、本センター Web ページ、利用支援ポータルなどでお知らせしていきます。