HPC SOLUTIONS

東京大学 物性研究所 御中

PC クラスタ
ユーザマニュアル

第 1.01 版

2011 年 9 月 1 日

2012 年 7 月 20 日更新

株式会社 HPC ソリューションズ

本書では、一般ユーザがクラスタに導入されているソフトウェアを利用できるまでの環境設定、利用方法を説明します。シェルは bash、MPI は OpenMPI を基本とし説明します。

1 Intel Composer XE2011

コンパイラとして Intel CompserXE2011 が導入されており、デフォルトで出来る様設定されています。サーバノード(ホスト名 psi)でのみコンパイラを動作させることが出来ます。その他のリモートノード(ホスト psi001～009)では動作させないで下さい。

1.1 環境設定

新規ユーザー登録時から bash シェル環境下では別途環境設定することなく利用可能となってますが、bash 以外のシェルを利用する場合は別途環境設定を施工する必要があります。
以下に示す形で bash シェル下での環境変数が設定されていますので、利用シェル環境に応じて施工して下さい。

＊ 当機における bash 環境では設定済であるため、.bashrc への追記作業は不要です。

```
source /opt/intel/composerxe/bin/compilervars.sh intel64
                        [ または ia32]
```

- **Compiler,mkl,ipp 等全ての環境が設定されます。**
- **32bit 環境を利用する場合、intel64 ではなく"ia32"と指定します。**
- **環境設定ファイル(.bashrc)に追記する事でログオン時に自動で読み込み・設定されます。また、Intel Compiler で Build した mpi プログラム実行時にリモートノードへ環境設定を反映させる必要があるため、必須設定となります。**
- **環境設定ファイル(.bashrc)を変更した場合、環境を反映させるために再度ログオンするか上記スクリプトを実行します。**

1.2 プログラムのコンパイル

コンパイラコマンドは、以下の通りです。

コマンド	説明
ifort	Fortran コンパイラ
icc	C コンパイラ
icpc	C++コンパイラ

使用したいコンパイラコマンドで、オプションを使用してコンパイルを行います。
以下にコマンドの使用方法例を示します。

icc [コンパイラオプション] ファイル [ファイル 2]…
icpc [コンパイラオプション] ファイル [ファイル 2]…
ifort [コンパイラオプション] ファイル [ファイル 2]…

また、コンパイルオプションについては、icc -help などを利用してください。

コンパイル例:

C コンパイラで実行モジュール mpitest を作成します。

```
$ icc -o mpitest mpitest.c
```

2 OpenMPI

MPI に OpenMPI Ver.1.4.3 を導入しています。以下のディレクトリ配下に導入しています。

/opt/mpi/openmpi/143/{Build コンパイラ名}

{Build コンパイラ名}

- **gnu … Gnu Compiler Ver4.4.4 Build 版(CentOS6 標準)**
- **intel … Intel Compiler Build 版**
- **gcc424 … Gnu Compiler Ver4.2.4 Build 版**

2.1 環境設定

以下の設定を環境設定ファイル(.bashrc)に記述します。
利用用途に応じて Build コンパイラの箇所を変更します。

Intel コンパイラで Build した openmpi 環境の設定例:

```
export PATH=/opt/mpi/openmpi/143/intel/bin:$PATH
export LD_LIBRARY_PATH=/opt/mpi/openmpi/143/intel/lib:
/opt/mpi/openmpi/143/intel/lib/openmpi:$LD_LIBRARY_PATH
```

2.2 MPI プログラムのコンパイル

MPI のコンパイラコマンドは、以下の通りです。

コマンド	説明
mpif77	Fortran77 コンパイラ
mpif90	Fortran90 コンパイラ
mpicc	C コンパイラ
mpicxx	C++コンパイラ

使用したいコンパイラ(Gnu、Intel)の MPI コンパイラコマンドで、そのコンパイラ(Gnu、Intel)のオプションを使用してコンパイルを行います。
以下に mpicc コマンドの基本的な使用方法を示します。

mpicc [コンパイラオプション] [ソース・オブジェクトファイル] [リンカオプション]

コンパイル例:

C コンパイラで実行モジュール testmpi を作成。

```
$ mpicc -o mpitest mpitest.c
```

2.3 MPI プログラムの実行

以下、バッチジョブスケジューラを利用しない実行方法を説明しますが、基本事項で説明するバッチジョブスケジューラのご利用を推奨します。
プログラムの実行はリモートノード(psi001～009)でのみ許可しています。サーバノード(psi)では実行しない様、注意して下さい。また、対象リモートノード上で他の計算が実行されている場合は性能低下が起こりますのでリソースの空きに注意して下さい。

1) 利用するホストのホスト名を、実行させるプロセス数分記述したマシンファイルを作成します。

マシンファイル作成例 1:単体のリモートノードの場合

- Machines というファイルを作成
- ホスト psi002、プロセス数 4

```
psi002
psi002
psi002
psi002
```

マシンファイル作成例 2:複数のリモートノードの場合

- machines2 というファイルを作成
- ホスト psi001、psi002、各プロセズ数 4

```
psi001
psi001
psi001
psi001
psi002
psi002
psi002
psi002
```

2) mpiexec コマンドで、MPI プログラムを実行します。

mpiexec -np [プロセス数] –machinefile [マシンファイル名]
実行プログラム

実行例 1:単体のリモートノードの場合

実行プログラム mpitest をホスト psi002、プロセス数 4 で実行。

```
$ mpiexec -np 4 –machinefile machines ./mpitest
myid = 00: psi002.issp.u-tokyo.ac.jp
myid = 01: psi002.issp.u-tokyo.ac.jp
myid = 02: psi002.issp.u-tokyo.ac.jp
myid = 03: psi002.issp.u-tokyo.ac.jp
```

実行例 2:複数のリモートノードの場合

実行プログラム mpitest をホスト psi001、psi002、各ホストプロセス数 4、合計 8 プロセズで実行。

```
$ mpiexec –np 8 –machinefile2 machines ./mpitest
myid = 04: psi002.issp.u-tokyo.ac.jp
myid = 05: psi002.issp.u-tokyo.ac.jp
myid = 00: psi001.issp.u-tokyo.ac.jp
myid = 06: psi002.issp.u-tokyo.ac.jp
myid = 01: psi001.issp.u-tokyo.ac.jp
myid = 07: psi002.issp.u-tokyo.ac.jp
myid = 02: psi001.issp.u-tokyo.ac.jp
myid = 03: psi001.issp.u-tokyo.ac.jp
```

3 Torque

バッチジョブスケジューラを利用することにより、リソースの空き状況を気にする事無く自動でのジョブの実行・管理をしてくれます。

3.1 環境設定

環境設定ファイル(.bashrc)に以下の設定を施し、反映させます。

```
export PATH=/opt/torque/bin:$PATH
export LD_LIBRARY_PATH=/opt/torque/lib:$LD_LIBRARY_PATH
```

3.2 キュークラス

利用形態別にキューを 3 つ(small,middle,large) 設定しています。small クラスが default クラスと設定となっていますので、クラスの指定をしないと default クラスが使用されますので注意して下さい。なお、利用できる CPU 数、最大実行時間、実行数などは下記の表の通りです。

クラス	最大 CPU 数	ノード数	最大実行時間	ユーザー毎の実行可能総本数
small (=default)	12	1	24 時間	6
middle	24	2	24 時間	3
large	48	4	1 時間	1

キュー構成は随時変更されますので、最新のキュー構成情報はログイン時のメッセージや**下記のコマンド**で確認してください。

cat /etc/motd

3.3 スケジューラーの利用方法

3.3.1ジョブの投入・実行

バッチジョブとして投入・実行する為には、スケジューラ投入用のジョブスクリプトを作成する必要があります。
以下、基本的な設定例を示します。

設定例 1: ノード数 1、プロセス数 1、small キュー、シングルジョブ

```
[hpcs@psi ~]$ cat test.sh
#!/bin/sh
#PBS -l nodes=1:ppn=1
#PBS -q small
#PBS -N pbs-test
#PBS -j oe

cd $PBS_O_WORKDIR

./a.out   → 実行ジョブ
```

設定例 2: ノード数 1、プロセス数 12、small キュー、flat MPI

```
[hpcs@psi ~]$ cat test.sh
#!/bin/sh
#PBS -l nodes=1:ppn=12
#PBS -q small
#PBS -N pbs-test
#PBS -j oe

cd $PBS_O_WORKDIR

mpiexec -np $PBS_NP ./a.out   → 実行ジョブ
```

設定例 3: ノード数 2、プロセス数 12、middle キュー、flat MPI

```
[hpcs@psi ~]$ cat test.sh
#!/bin/sh
#PBS -l nodes=2:ppn=12
#PBS -q middle
#PBS -N pbs-test
#PBS -j oe

cd $PBS_O_WORKDIR

mpiexec -np $PBS_NP ./a.out   → 実行ジョブ
```

設定例 4: ノード数 4、プロセス数 4、スレッド数 12、large キュー、hybrid MPI

```
[hpcs@psi ~]$ cat test.sh
#!/bin/sh
#PBS -l nodes=4:ppn=12
#PBS -q large
#PBS -N pbs-test
#PBS -j oe

export OMP_NUM_THREADS=12

cd $PBS_O_WORKDIR

mpiexec -np 4 -pernode ./a.out   → 実行ジョブ
```

ジョブの投入例:

```
[hpcs@psi ~]$ qsub test.sh
118.psi.issp.u-tokyo.ac.jp
```

3.3.2ジョブの確認

投入したジョブを確認する際は qstat コマンドを使用します。

```
[hpcs@psi ~]$ qstat -a
psi:
                                                            Req'd  Req'd   Elap
Job ID              Username Queue    Jobname    SessID NDS   TSK Memory Time  S Time
--------------------       ------------ ---------- ----------------- ----------
-----    ------ ------    -----   - -----
17.psi              admin     default     test          9818  1      1  --      --   R
00:00
18.psi              hpcs     default     test          9919  1      1  --      --   R 00:00
```

3.3.3ジョブの削除

投入したジョブを削除したい場合は、qdel コマンドを使用します。

下記は Job ID:18 のジョブを削除する例です。

```
[hpcs@psi ~]$ qstat -a

psi:
                                                            Req'd  Req'd   Elap
Job ID              Username Queue    Jobname    SessID NDS   TSK Memory Time  S Time
--------------------       ------------ ---------- ----------------- ----------
17.psi              admin     default     test          9818  1      1  --      --   R
00:00
18.psi              hpcs     default     test          9919  1      1  --      --   R 00:00
[hpcs@psi ~]$ qdel 18
[hpcs@psi ~]$ qstat -a

psi:
                                                            Req'd  Req'd   Elap
Job ID              Username Queue    Jobname    SessID NDS   TSK Memory Time  S Time
--------------------       ------------ ---------- ----------------- ----------
17.psi              admin     default     test          9818  1      1  --      --   R
00:00
```

4 FFTW

FFTW3.3 Beta 版を以下のディレクトリ配下にインストールしています。

```
/usr/local/fftw33-b/{Build コンパイラ名}
```

＊ FFTW3.2.2 も別途/usr/local/fftw32 に同様に導入しています。

{Build コンパイラ名}

- **openmpi143/gnu … Gnu Compiler Ver4.4.4＆Gnu 版 openmpi Build 版**
- **openmpi143/intel … Intel Compiler ＆Intel 版 openmpi Build 版**
- **openmpi143/gcc424 … Gnu Compiler Ver4.2.4 ＆gcc424 版 openmpi Build 版**

4.1 環境設定

以下の設定を環境設定ファイル(.bashrc)に記述します。
利用用途に応じて Build コンパイラの箇所を変更します。

gnu コンパイラで Build した fftw 環境の設定例:

```
export LD_LIBRARY_PATH=/opt/fftw33-b/openmpi143/gnu/lib:
$LD_LIBRARY_PATH
```

4.2 利用方法

利用方法は、ソースコードでヘッダファイル fftw3.h をインクルードさせ、リンク時に-lbfftw3 として libbfftw3.a をリンクさせる必要があります。もし、libbfftw3.a が検索できないときは、ライブラリの検索オプション-L を各コンパイラにあったディレクトリを検索オプションに加えてください。

実行例)Gnu 版を利用した場合

```
hpcs@psi~$ gcc -L/usr/local/fftw33-b/openmpi143/gnu/lib -lfftw3 main.c
```

5 PAPI

これまで別途 Kernel に Patch を当てる必要があった PAPI ですが、CentOS 6 から標準で Kernel にハードウエアカウンターが利用可能となった状態で組み込まれております。
組み見込まれているバージョンは、Ver.4.1.0-2 となります。

5.1 利用方法

利用方法は、ソースコードでヘッダファイル papi.h をインクルードさせ、リンク時に-lpapi として libpapi.so をリンクさせる必要があります。

実行例）Gnu 版を利用した場合

```
hpcs@psi~$ gcc -lpapi main.c
```

6 GCC(Gnu Compiler)

OS 標準(gnu-4.4.4)のもの以外に、Ver.4.2.4 を導入しております。
以下のディレクトリ配下にインストールしています。

```
/usr/local/gcc424
```

- **Build 時の指定オプションなし**
- **JAVA 関連、adm 関連を有効にするとエラーとなり Build 不可。システム標準の Ver.4.4.4 からのバージョンダウン扱いとなり、それに伴いシステムの JAVA、gnat のバージョンをダウンさせる必要があります。これらは、システムの根幹に関わる部分であり、ダウンさせる事が難しいためオプション無しでの Build とさせて頂いております。**
- **OS 標準の Ver.4.4.4 には環境設定は必要ありません。ただし Ver.4.2.4→4.4.4 へ利用変更される場合は、Ver.4.2.4 の環境設定を削除して頂く必要があります。**

6.1 環境設定

環境設定ファイル(.bashrc)に以下の設定を施し、反映させます。

```
export PATH=/usr/local/gcc424/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/gcc424/lib:
$LD_LIBRARY_PATH
```

6.2 プログラムのコンパイル<Ver.4.4.4、4.2.4>

コンパイラコマンドは、以下の通りです。

コマンド	説明
gfortran	Fortran コンパイラ
gcc	C コンパイラ
g++	C++コンパイラ

使用したいコンパイラコマンドで、オプションを使用してコンパイルを行います。
以下にコマンドの使用方法を示します。

gcc [コンパイラオプション] ファイル [ファイル 2]…
g++ [コンパイラオプション] ファイル [ファイル 2]…
gfortran [コンパイラオプション] ファイル [ファイル 2]…

コンパイル例:

実行モジュール testmpi を作成する。

```
$ gcc -o testmpi testmpi.c
```

7 Appendix…（mpich2）

本クラスタシステムには MPI として、OpenMPI 以外に mpich2 Ver.1.4 を以下のディレクトリ配下にインストールしています。

```
/opt/mpi/mpich2/1.4/{Buildコンパイラ名}
```

- **gnu … Gnu Compiler Ver4.4.4 Build 版（CentOS6 標準）**
- **intel … Intel Compiler Build 版**
- **gcc424 … Gnu Compiler Ver4.2.4 Build 版**

7.1 環境設定

以下の設定を環境設定ファイル(.bashrc)に記述します。
利用用途に応じて Build コンパイラの箇所を変更します。

Intel コンパイラで Build した mpich2 環境の設定例:

```
export PATH=/opt/mpi/mpich2/1.4/intel/bin:$PATH
export LD_LIBRARY_PATH=/opt/mpi/mpich2/1.4/intel/lib:
$LD_LIBRARY_PATH
```

7.2 MPI プログラムのコンパイル

MPI のコンパイラコマンドは、以下の通りです。

コマンド	説明
mpif77	Fortran77 コンパイラ
mpif90	Fortran90 コンパイラ
mpicc	C コンパイラ
mpicxx	C++コンパイラ

使用したいコンパイラ(Gnu、Intel)の MPI コンパイラコマンドで、そのコンパイラ(Gnu、Intel)のオプションを使用してコンパイルを行います。
以下に mpicc コマンドの基本的な使用方法を示します。

```
mpicc [コンパイラオプション] [ソース・オブジェクトファイル] [リンカオプション]
```

コンパイル例:
C コンパイラで実行モジュール testmpi を作成。

```
$ mpicc -o testmpi testmpi.c
```

7.3 MPI プログラムの実行

本書 2.3 項 openmpi での実行方法と同じです。
バッチジョブスケジューラを利用することを推奨しています。

1) 利用するホストのホスト名を、実行させるプロセス数分記述したマシンファイルを作成します。

マシンファイル作成例 1:単体のリモートノードの場合
- Machines というファイルを作成
- ホスト psi002、プロセス数 4

```
psi002
psi002
psi002
psi002
```

マシンファイル作成例 2:複数のリモートノードの場合
- machines2 というファイルを作成
- ホスト psi001、psi002、各プロセズ数 4

```
psi001
psi001
psi001
psi001
psi002
psi002
psi002
psi002
```

2) mpiexec コマンドで、MPI プログラムを実行します。

```
mpiexec –np [プロセス数] –machinefile [マシンファイル名]
実行プログラム
```

実行例 1:単体のリモートノードの場合
実行プログラム mpitest をホスト psi002、プロセス数 4 で実行。

```
$ mpiexec -np 4 –machinefile machines ./mpitest
myid = 00: psi002.issp.u-tokyo.ac.jp
myid = 01: psi002.issp.u-tokyo.ac.jp
```

```
myid = 02: psi002.issp.u-tokyo.ac.jp
myid = 03: psi002.issp.u-tokyo.ac.jp
```

実行例 2:複数のリモートノードの場合

実行プログラム mpitest をホスト psi001、psi002、各ホストプロセス数 4、合計 8 プロセズで実行。

```
$ mpiexec -np 8 -machinefile2 machines ./mpitest
myid = 04: psi002.issp.u-tokyo.ac.jp
myid = 05: psi002.issp.u-tokyo.ac.jp
myid = 00: psi001.issp.u-tokyo.ac.jp
myid = 06: psi002.issp.u-tokyo.ac.jp
myid = 01: psi001.issp.u-tokyo.ac.jp
myid = 07: psi002.issp.u-tokyo.ac.jp
myid = 02: psi001.issp.u-tokyo.ac.jp
myid = 03: psi001.issp.u-tokyo.ac.jp
```