AOBA-B

プログラム開発・実行環境

利用手順書

日本電気株式会社

2021 年 4 月 6 日

2.0 版

＜＜　改版履歴　＞＞

版数	改版日	改版理由	改版者
初版	2020.10.17	初版として作成	NEC
2.0	2021.4.6	AOCC のバージョンを 3.0.0 に変更 Intel コンパイラ 2019 を Intel OneAPI2021 に変更	NEC

目次

1. AOBA-B 実行プログラム開発・実行環境

AOBA-B は、AMD 社製 EPYC CPU を 2 基搭載したノード 68 台で構成しています。
CPU 当り 64 コアを搭載しており、ノード内並列では、128 共有メモリ並列までのプログラム実行および、複数ノードを使用した MPI 並列プログラム実行が行えます。

AOBA-B で実行するプログラムの開発・実行環境として、AMD 社製 EPYC CPU に対応した複数の環境が提供されています。

開発環境	ノード内並列	MPI 実行環境	科学技術計算ライブラリ	備考
AOCC3.0.0	OpenMP	OpenMPI4.0.5	AOCL	既定値
GCC10.1.0	ループ並列 OpenMP	OpenMPI4.0.2	AOCL	
InteloneAPI 2021	自動並列 OpenMP	IntelMPI2021	IntelMKL	移行用

2. AOCC3.0.0 によるプログラム開発・実行（既定値環境）

1) プログラム のコンパイル

① ノード内実行プログラム

コンパイラコマンド： flang(Fortran プログラム)
clang(C プログラム)
clang++(C++プログラム)

実行形式：実行オブジェクト a.out を、カレントディレクトリに作成する場合

```
<コンパイラコマンド>  <コンパイラオプション> <プログラムソースファイル名>
例)
$ flang –O3 sample.f
```

コンパイラオプション例：

-march=znver2	EPYC CPU 用にコンパイルすることを指定する
-O*n*	最適化レベルを指定する
fast	最大限の最適化を適用する
3	積極的に最適化を適用する
2	(既定値) 標準的な最適化を適用する

1	最低限の最適化を適用する
0	すべての最適化を無効化する
-fopenmp	OpenMP 並列化を利用する

AOCC のコンパイルオプションの詳細は以下を参照下さい。
https://developer.amd.com/amd-aocc/

② MPI 実行プログラム

コンパイラコマンド： mpifort(Fortran プログラム)
mpicc(C プログラム)
mpic++(C++プログラム)

実行形式：実行オブジェクト a.out を、カレントディレクトリに作成する場合

```
<コンパイラコマンド>　<コンパイラオプション> <プログラムソースファイル名>
例)
$ mpifort –O3 sample.f
```

コンパイラオプション例：
ノード内実行プログラム作成時と同じオプションが利用できます。

2) ジョブスクリプト作成と実行

作成した実行プログラムは、バッチ処理環境(NQSV：NEC Network Queuing System V)に対し、バッチリクエストという形式で実行を依頼します。
バッチリクエストで実行を依頼するためには、プログラムの実行を記述した、ジョブスクリプトを作成します。

① ノード内実行プログラム用ジョブスクリプト

バッチリクエストを処理するための、実行時間などのパラメータを定義して、実行プログラムの実行を記述します。

例(sh 形式の場合)：

```
#!/bin/sh
#PBS -l elapstim_req=2:00:00          ····· (a)
#PBS –b 1                             ····· (b)
#PBS –v OMP_NUM_THREADS=128           ····· (c)
#PBS –A kadai1                        ····· (d)
cd $PBS_O_WORKDIR                     ····· (e)
./a.out                               ····· (f)
```

記述詳細：

(a)　：最大実行経過時間を指定(hh:mm:ss 形式)

(b)　：使用ノード数を指定，ノード内実行プログラムの場合には 1 固定

(c)　：OpenMP 並列実行プログラムの場合、並列数を指定

(d)　：課金先のプロジェクトコードを指定(任意)，未指定の場合はデフォルトのプロジェクトコードに課金

(e)　：カレントディレクトリへ移動

(f)　：実行プログラム名を指定

② MPI 実行プログラム用ジョブスクリプト

MPI 実行プログラムは、mpirun コマンドで実行プログラムを実行します。
バッチリクエストを処理するための、実行時間などのパラメータを定義して、mpirun コマンドで実行プログラムの実行を記述します。

例(sh 形式の場合)：

```
#!/bin/sh
#PBS -T openmpi                       ····· (a)
#PBS -l elapstim_req=4:00:00          ····· (b)
#PBS -b 2                             ····· (c)
#PBS -A kakin1                        ····· (d)
cd $PBS_O_WORKDIR                     ····· (e)
mpirun $NQSV_MPIOPTS –np 256 ./a.out  ····· (f)
```

記述詳細：

(a)　：MPI 実行環境を指定，AOCC 開発環境では、openmpi 固定

(b)　：最大実行経過時間を指定(hh:mm:ss 形式)

(c)　：使用ノード数を指定

(d)　：課金先のプロジェクトコードを指定(任意)，未指定の場合はデフォルトのプロジェクトコードに課金

(e)　　：カレントディレクトリへ移動

(f)　　：mpirun コマンドで実行プログラムを指定

-b オプションで指定した使用ノード数に応じて、バッチリクエストのノード割り当てを自動的に設定するための $NQSV_MPIOPTS を指定

-np オプションで MPI 総プロセス数を指定(-b 指定ノード数×128)

作成したジョブスクリプトを、バッチ処理環境(NQSV)に qsub コマンドを使って投入します。

```
qsub −q lx <ジョブスクリプトファイル名>
例)
$ qsub −q lx run.sh
```

3) 利用上の注意

① 旧システムの MPI プログラム実行ジョブスクリプトは、そのままでは実行はできません。ジョブスクリプトを、上記形式への修正をして実行して下さい。

② 旧システムの LX-406 用の実行モジュールは、AOCC3.0.0 環境では実行できません。

③並列コンピュータ MPI プログラム実行の標準的な実行方法(上記例)では、MPI プロセスは、NQSV により割り当てられたノードの先頭ノードの先頭 CPU のコアから順次割り当てられます。

例) 128 MPI プロセス を実行

```
#!/bin/sh
#PBS -T openmpi
#PBS -l elapstim_req=4:00:00
#PBS -b 1
cd $PBS_O_WORKDIR
mpirun $NQSV_MPIOPTS −np 128 ./a.out
```

⇒ 1 ノードで 128 MPI プロセスが実行される。

例) 160 MPI プロセスを実行

```
#!/bin/sh
#PBS -T openmpi
#PBS -l elapstim_req=4:00:00
#PBS -b 2
cd $PBS_O_WORKDIR
mpirun $NQSV_MPIOPTS –np 160 ./a.out
```

⇒ 先頭ノードに 128 プロセス（64 コア/CPU×2CPU）割り当て、次ノードに 32 プロセス割り当てられる。

④ MPI プログラム実行において、ノード毎のプロセス数割り当てを指定する場合には、mpirun コマンドでノード当りのプロセス数の指定を行ないます。

例) 2 ノードで、160 MPI プロセスを、80 プロセス/ノードで実行

```
#!/bin/sh
#PBS -T openmpi
#PBS -l elapstim_req=4:00:00
#PBS -b 2
cd $PBS_O_WORKDIR
mpirun $NQSV_MPIOPTS --map-by ppr:80:node -np 160   ./a.out
```

ppr:<プロセス数>:node　　：　ノード当りの MPI プロセス数を指定

3. GCC10.1.0 によるプログラム開発・実行

1) プログラム のコンパイル

既定値のプログラム開発環境は、AOCC3.0.0 となっています。GCC10.1.0 でプログラムを開発する場合には、開発環境を切り替えるためのスクリプトを実行した上で、コンパイルを行います。

ログインシェルが、bash の場合：

```
$ source /opt/gcc/gcc-10.1.0/setenv_GCC10.1.0.sh
```

ログインシェルが、tcsh の場合：

```
$ source /opt/gcc/gcc-10.1.0/setenv_GCC10.1.0.csh
```

① ノード内実行プログラム

コンパイラコマンド： gfortran(Fortran プログラム)
gcc(C プログラム)
g++(C++プログラム)

実行形式：実行オブジェクト a.out を、カレントディレクトリに作成する場合

```
<コンパイラコマンド>　<コンパイラオプション> <プログラムソースファイル名>
例)
$ gfortran –O3 sample.f
```

コンパイラオプション例：

-march=znver2	EPYC CPU 用にコンパイルすることを指定する
-O*n*	最適化レベルを指定する
fast	最大限の最適化を適用する
3	積極的に最適化を適用する
2	(既定値) 標準的な最適化を適用する
1	最低限の最適化を適用する
0	すべての最適化を無効化する
-ftree-parallelize-loops=n	ループを n スレッドで実行する
-fopenmp	OpenMP 並列化を利用する

GCC のコンパイルオプションの詳細は以下を参照下さい。
https://gcc.gnu.org/onlinedocs/

② MPI 実行プログラム

コンパイラコマンド： mpifort(Fortran プログラム)
mpicc(C プログラム)
mpic++(C++プログラム)

実行形式：実行オブジェクト a.out を、カレントディレクトリに作成する場合

```
<コンパイラコマンド>　<コンパイラオプション> <プログラムソースファイル名>
例)
$ mpifort –O3 sample.f
```

コンパイラオプション例：
ノード内実行プログラム作成時と同じオプションが利用できます。

2) ジョブスクリプト作成と実行

作成した実行プログラムは、バッチ処理環境(NQSV：NEC Network Queuing System V)に対し、バッチリクエストという形式で実行を依頼します。

バッチリクエストで実行を依頼するためには、プログラムの実行を記述した、ジョブスクリプトを作成します。

プログラム実行環境の既定値は、AOCC3.0.0 となっていますので、GCC10.1.0 で開発したプログラムを実行する場合には、ジョブスクリプトに GCC 実行環境を指定する必要があります。

① ノード内実行プログラム用ジョブスクリプト

バッチリクエストを処理するための、実行時間などのパラメータを定義して、実行プログラムの実行を記述します。

例(sh 形式の場合)：

```
#!/bin/sh
#PBS -l elapstim_req=2:00:00          ・・・・・ (a)
#PBS –b 1                             ・・・・・ (b)
#PBS –v OMP_NUM_THREADS=128           ・・・・・ (c)
#PBS –A kakin1                        ・・・・・ (d)
source /opt/gcc/gcc-10.1.0/setenv_GCC10.1.0.sh   ・・・・・ (e)
cd $PBS_O_WORKDIR                     ・・・・・ (f)
./a.out                               ・・・・・ (g)
```

記述詳細：

(a)　：最大実行経過時間を指定(hh:mm:ss 形式)

(b)　：使用ノード数を指定，ノード内実行プログラムの場合には 1 固定

(c)　：OpenMP 並列実行プログラムの場合、並列数を指定

(d)　：課金先のプロジェクトコードを指定(任意)，未指定の場合はデフォルトのプロジェクトコードに課金

(e)　：プログラム実行環境を GCC10.1.0 に切替え

(f)　：カレントディレクトリへ移動

(g)　：実行プログラム名を指定

② MPI 実行プログラム用ジョブスクリプト

MPI 実行プログラムは、mpirun コマンドで実行プログラムを実行します。

バッチリクエストを処理するための、実行時間などのパラメータを定義して、mpirun コマンドで実行プログラムの実行を記述します。

例(sh 形式の場合)：

```
#!/bin/sh
#PBS -T openmpi                    ・・・・・ (a)
#PBS -l elapstim_req=4:00:00       ・・・・・ (b)
#PBS -b 2                          ・・・・・ (c)
#PBS –A kakin1                     ・・・・・ (d)
#PBS –v CMPENV=GCC                 ・・・・・ (e)
source /opt/gcc/gcc-10.1.0/setenv_GCC10.1.0.sh   ・・・・・ (f)
cd $PBS_O_WORKDIR                  ・・・・・ (g)
mpirun $NQSV_MPIOPTS –np 256 ./a.out   ・・・・・ (h)
```

記述詳細：

(a)　：MPI 実行環境を指定，GCC 開発環境では、openmpi 固定

(b)　：最大実行経過時間を指定(hh:mm:ss 形式)

(c)　：使用ノード数を指定

(d)　：課金先のプロジェクトコードを指定(任意)，未指定の場合はデフォルトのプロジェクトコードに課金

(e)　：バッチリクエストの環境を GCC に指定

(f)　：プログラム実行環境を GCC10.1.0 に切替え

(g)　：カレントディレクトリへ移動

(h)　：mpirun コマンドで実行プログラムを指定

-b オプションで指定した使用ノード数に応じて、バッチリクエストのノード割り当てを自動的に設定するための $NQSV_MPIOPTS を指定

-np オプションで MPI 総プロセス数を指定(-b 指定ノード数×128)

作成したジョブスクリプトを、バッチ処理環境(NQSV)に qsub コマンドを使って投入します。

```
qsub –q lx <ジョブスクリプトファイル名>
例)
$ qsub –q lx run.sh
```

3) 利用上の注意

① 旧システムの MPI プログラム実行ジョブスクリプトは、そのままでは実行はできません。ジョブスクリプトを、上記形式への修正をして実行して下さい。

② 旧システムの LX-406 用の実行モジュールは、GCC10.1.0 環境では実行できません。

③ MPI プログラム実行の標準的な実行方法(上記例)では、MPI プロセスは、NQSV により割り当てられたノードの先頭ノードの先頭 CPU のコアから順次割り当てられます。

例) 128 MPI プロセス を実行

```
#!/bin/sh
#PBS -T openmpi
#PBS -l elapstim_req=4:00:00
#PBS -b 1
#PBS –v CMPENV=GCC
source /opt/gcc/gcc-10.1.0/setenv_GCC10.1.0.sh
cd $PBS_O_WORKDIR
mpirun $NQSV_MPIOPTS –np 128 ./a.out
```

⇒ 1 ノードで 128 MPI プロセスが実行される。

例) 160 MPI プロセスを実行

```
#!/bin/sh
#PBS -T openmpi
#PBS -l elapstim_req=4:00:00
#PBS -b 2
#PBS –v CMPENV=GCC
source /opt/gcc/gcc-10.1.0/setenv_GCC10.1.0.sh
cd $PBS_O_WORKDIR
mpirun $NQSV_MPIOPTS –np 160 ./a.out
```

⇒ 先頭ノードに 128 プロセス（64 コア/CPU×2CPU）割り当て、次ノードに 32 プロセス割り当てられる。

③ MPI プログラム実行において、ノード毎のプロセス数割り当てを指定する場合には、mpirun コマンドでノード当りのプロセス数の指定を行ないます。

例) 2 ノードで、160 MPI プロセスを、80 プロセス/ノードで実行

```
#!/bin/sh
#PBS -T openmpi
#PBS -l elapstim_req=4:00:00
#PBS -b 2
#PBS –v CMPENV=GCC
source /opt/gcc/gcc-10.1.0/setenv_GCC10.1.0.sh
cd $PBS_O_WORKDIR
mpirun $NQSV_MPIOPTS --map-by ppr:80:node -np 160   ./a.out
```

ppr:<プロセス数>:node　　：　ノード当りの MPI プロセス数を指定

4. InteloneAPI2021 によるプログラム開発・実行

1) プログラム のコンパイル

既定値のプログラム開発環境は、AOCC3.0.0 となっています。InteloneAPI2021 でプログラムを開発する場合には、開発環境を切り替えるためのスクリプトを実行した上で、コンパイルを行います。

※Intel oneAPI2021 はログインシェル bash のみサポートしています。

※Intel コンパイラ 2019 も利用できます。Intel コンパイラ 2019 はログインシェル bash、csh をサポートしています。

Intel oneAPI 2021（ログインシェル bash のみ）：

```
$ source /opt/oneapi/setvars.sh intel64
```

Intel コンパイラ 2019（ログインシェルが bash の場合）：

```
$ source /opt/intel/bin/compilervars.sh intel64
```

Intel コンパイラ 2019（ログインシェルが tcsh の場合）：

```
$ source /opt/intel/bin/compilervars.csh intel64
```

① ノード内実行プログラム

コンパイラコマンド：　ifort(Fortran プログラム)
　　icc(C/C++プログラム)

実行形式：実行オブジェクト a.out を、カレントディレクトリに作成する場合

```
<コンパイラコマンド>　<コンパイラオプション> <プログラムソースファイル名>
例)
$ ifort –O3 sample.f
```

コンパイラオプション例：

-O*n*		最適化レベルを指定する
	fast	最大限の最適化を適用する
	3	-O2 レベルの最適化に加えて，ループ融合，ループブロッキングのような最適化も適用する
	2	(既定値) 推奨される最適化レベル．ベクトル化を含む多くの最適化を有効にする
	1	オブジェクトサイズを増加させる最適化を無効にする
	0	すべての最適化を無効化する
-parallel		自動並列化機能を利用する
-qopenmp		OpenMP 並列化を利用する
-qopt-report		最適化レポートを作成する

Intel oneAPI のコンパイルオプションの詳細は以下を参照下さい。

https://www.xlsoft.com/jp/products/intel/tech/documents.html#doc-oneapi

② MPI 実行プログラム

コンパイラコマンド：　mpiifort(Fortran プログラム)
mpiicc(C/C++プログラム)

実行形式：実行オブジェクト a.out を、カレントディレクトリに作成する場合

```
<コンパイラコマンド>　<コンパイラオプション> <プログラムソースファイル名>
例)
$ mpiifort –O3 sample.f
```

コンパイラオプション例：

ノード内実行プログラム作成時と同じオプションが利用できます。

2) ジョブスクリプト作成と実行

作成した実行プログラムは、バッチ処理環境(NQSV：NEC Network Queuing System V)に対し、バッチリクエストという形式で実行を依頼します。

バッチリクエストで実行を依頼するためには、プログラムの実行を記述した、ジョブスクリプトを作成します。

プログラム実行環境の既定値は、AOCC3.0.0 となっていますので、Intel OneAPI2021 で開発したプログラムを実行する場合には、ジョブスクリプトに Intel コンパイラ実行環境を指定する必要があります。

① ノード内実行プログラム用ジョブスクリプト

バッチリクエストを処理するための、実行時間などのパラメータを定義して、実行プログラムの実行を記述します。

例(sh 形式の場合)：

```
#!/bin/sh
#PBS -l elapstim_req=2:00:00          ····· (a)
#PBS –b 1                             ····· (b)
#PBS –v OMP_NUM_THREADS=128           ····· (c)
#PBS –A kakin1                        ····· (d)
source /opt/oneapi/setvars.sh intel64 ····· (e)
cd $PBS_O_WORKDIR                     ····· (f)
./a.out                               ····· (g)
```

記述詳細：

(a)　：最大実行経過時間を指定(hh:mm:ss 形式)

(b)　：使用ノード数を指定，ノード内実行プログラムの場合には 1 固定

(c)　：OpenMP 並列実行プログラムの場合、並列数を指定

(d)　：課金先のプロジェクトコードを指定(任意)，未指定の場合はデフォルトのプロジェクトコードに課金

(e)　：プログラム実行環境を Intel OneAPI2021 に切替え

(f)　：カレントディレクトリへ移動

(g)　：実行プログラム名を指定

② MPI 実行プログラム用ジョブスクリプト

MPI 実行プログラムは、mpirun コマンドで実行プログラムを実行します。

バッチリクエストを処理するための、実行時間などのパラメータを定義して、mpirun コマンドで実行プログラムの実行を記述します。

例(sh 形式の場合)：

```
#!/bin/sh
#PBS -T intmpi                          ・・・・・ (a)
#PBS -l elapstim_req=4:00:00            ・・・・・ (b)
#PBS -b 2                               ・・・・・ (c)
#PBS –A kakin1                          ・・・・・ (d)
source /opt/oneapi/setvars.sh intel64   ・・・・・ (e)
cd $PBS_O_WORKDIR                       ・・・・・ (f)
mpirun -np 256 ./a.out                  ・・・・・ (g)
```

記述詳細：

(a)　：MPI 実行環境を指定，Intel コンパイラ開発環境では、intmpi 固定

(b)　：最大実行経過時間を指定(hh:mm:ss 形式)

(c)　：使用ノード数を指定

(d)　：課金先のプロジェクトコードを指定(任意)，未指定の場合はデフォルトのプロジェクトコードに課金

(e)　：MPI プログラム実行環境を InteloneAPI2021 に切替え

(f)　：カレントディレクトリへ移動

(g)　：mpirun コマンドで実行プログラムを指定
　　-np オプションで MPI 総プロセス数を指定(-b 指定ノード数×128)

作成したジョブスクリプトを、バッチ処理環境(NQSV)に qsub コマンドを使って投入します。

```
qsub –q lx <ジョブスクリプトファイル名>
例)
$ qsub –q lx run.sh
```

3) 利用上の注意

① 旧システムの MPI プログラム実行ジョブスクリプトは、そのままでは実行はできません。ジョブスクリプトを、上記形式への修正をして実行して下さい。

② MPI プログラム実行の標準的な実行方法(上記例)では MPI プロセスは、NQSV により割り当てられたノードの先頭ノードの先頭 CPU のコアから順次割り当てられます。

例) 128 MPI プロセス を実行

```
#!/bin/sh
#PBS -T intmpi
#PBS -l elapstim_req=4:00:00
#PBS -b 1
source /opt/oneapi/setvars.sh intel64
cd $PBS_O_WORKDIR
mpirun -np 128 ./a.out
```

⇒ 1 ノードで 128 MPI プロセスが実行される。

例) 160 MPI プロセスを実行

```
#!/bin/sh
#PBS -T intmpi
#PBS -l elapstim_req=4:00:00
#PBS -b 2
source /opt/oneapi/setvars.sh intel64
cd $PBS_O_WORKDIR
mpirun -np 160 ./a.out
```

⇒ 先頭ノードに 128 プロセス（64 コア/CPU×2CPU）割り当て、次ノードに 32 プロセス割り当てられる。

③ ノード毎のプロセス数割り当てを指定する場合には、mpirun コマンドでノード当りのプロセス数の指定を行ないます。

例) 2 ノードで、160 MPI プロセスを、80 プロセス/ノードで実行

```
#!/bin/sh
#PBS -T intmpi
```

```
#PBS -l elapstim_req=4:00:00
#PBS -b 2
source /opt/oneapi/setvars.sh intel64
cd $PBS_O_WORKDIR
mpirun -hostfile ${PBS_NODEFILE} -ppn 80 -np 160 ./a.out
```

ppn　　：　ノード当りの MPI プロセス数を指定

④ Intel コンパイラ，IntelMPI のバージョンアップにより、旧システムで利用していたプログラム(実行オブジェクト）がそのままでは動作しない場合があります。
その場合には、上記に従い、プログラムのコンパイルを行って下さい。

5. ライブラリの利用

1) AOCL(AMD Optimizing CPU Libraries)

AOCL(AMD Optimizing CPU Libraries)は、AMD 社が提供する EPYC CPU に最適化された数値計算ライブラリパッケージです。
AOCC 開発環境、および GCC 開発環境から利用することができます。

パッケージの詳細は、AMD 社のサイトを参照して下さい。
https://developer.amd.com/amd-aocl/

ライブラリの利用方法は、以下を参照して下さい。
AOCL_User_Guide_3.0.pdf

2) IntelMKL

Intel コンパイラ開発環境から利用できる数値計算ライブラリパッケージです。

パッケージの詳細および利用方法は、以下を参照して下さい。
https://www.xlsoft.com/jp/products/intel/perflib/mkl/index.html