AOBA-S

プログラム開発・実行環境

利用手順書

日本電気株式会社

2024 年 3 月 29 日

第 4 版

＜＜ 改版履歴 ＞＞

版数	改版日	改版理由	改版者
第１版	2023.8.1	第一版として作成	NEC
第２版	2023.8.4	マニュアル全体のレイアウトおよび文章を変更。 「1.AOBA-S 実行プログラム開発・実行環境」において、VH および VE についての説明を変更。コンパイルの互換性について説明を変更。 「2.プログラム開発環境」において、ロードするモジュールの表を VE、VH 用に変更。モジュール利用についての説明を変更。 「3.プログラム実行環境」において、プログラム用ジョブスクリプト例の変更。	NEC
第３版	2023.8.30	「3.プログラム実行環境」において、環境変数 VE_OMP_NUM_THREADS の説明追加	NEC
第４版	2023.3.29	マニュアル全体の記載および利用コマンドを Intel OneAPI 2023 から 2024 へ変更 「2.プログラム開発環境」のデフォルトで読み込まれる NEC SDK バージョンの記載を NEC SDK を 5.0.2 から 5.2.0 に変更、NEC MPI を 3.4.0 から 3.6.0 に変更、NEC NLC を 3.0.0 から 3.1.0 に変更 「2.プログラム開発環境」へ MODULES_AUTO_HANDLING の説明を追加 「3.プログラム実行環境 2)」へノードあたりのプロセス数の指定方法を追加	NEC

目次

1. AOBA-S 実行プログラム開発・実行環境

1) 概要

AOBA-S は、NEC 社製ベクトルエンジン(VE) 8 基 と VH CPU(VH) 1 基(120 コア:60 物理コア HT 有効、1 ソケット)を搭載したノード 504 台で構成されています。
VE 1 基は、16 コアで構成され、共有メモリ並列では 16 並列までのプログラム実行が行えます。
VH は、120 コアで構成され、共有メモリ並列では 120 並列までのプログラム実行が行えます。
また、VH および VE を複数使用した MPI 並列プログラムの実行可能です。

NEC SDK は VE 上のみでのプログラム実行が可能です。
Intel oneAPI は VH 上のみでのプログラム実行が可能です。

AOBA-S で実行するプログラムの開発には、VE、VH それぞれの専用環境を利用します。

	開発環境	ノード内並列	MPI ライブラリ	数値計算ライブラリ
ベクトルエンジン(VE)	NEC SDK	自動並列 OpenMP	NEC MPI	NEC NLC
ベクトルホスト(VH)	Intel oneAPI	自動並列 OpenMP	Intel MPI	Intel MKL

AOBA-S 演算サーバ(1 ノード)の使用可能なリソースは、以下の通りとなります。

	コア数	メモリ容量
ベクトルエンジン(VE)	16 コア/VE x8 基	96GB/VE x8 基
ベクトルホスト(VH)	120 コア(60 物理コア HT 有効 1 ソケット) ※数値演算時は 60 コア以下の利用を推奨	251GB

※HT: Hyper Threading

2) 利用上の注意

コンパイラを使用する際は、module コマンドにて使用するモジュールとバージョンを指定して環境変数を定義します。使用方法については、以降の各コンパイラの使用方法を参照ください。

3) AOBA-A と AOBA-S の互換性

① 実行ファイルの互換性

VE 向けプログラムについて、AOBA-A でコンパイルされた実行プログラムは AOBA-S で実行可能です。ただし、AOBA-S の環境に合わせて最適化されていませんので、AOBA-S で実行するプログラムは、AOBA-S でコンパイルすることを強く推奨します。

② サーチパスの変更点

AOBA-S と AOBA-A では、モジュールファイル、および、INCLUDE 行、#include で取り込まれるファイル、ライブラリのサーチパスが、以下のように変更されます。

モジュールファイルのサーチパスおよび INCLUDE 行、および、#include で取り込まれるファイルのサーチ(サーチする順番に記載)

AOBA-S	AOBA-A
ソースファイルのあるディレクトリ	←
カレントディレクトリ	←
-I で指定されたディレクトリ	←
-B で指定されたディレクトリ直下の includeディレクトリ	←
環境変数 NFORT_INCLUDE_PATH で指定されたディレクトリ	←
-isystem で指定されたディレクトリ	←
/opt/nec/ve/nfort/バージョン番号/include	←
/opt/nec/ve3/include(※)	/opt/nec/ve/include(※)

※-isysroot が指定されているとき、-isysroot で指定されたディレクトリ直下の include がサーチされます。

ライブラリのサーチパス（サーチする順番に記載）

AOBA-S	AOBA-A
-L で指定されたディレクトリ	←
-B で指定されたディレクトリ	←
環境変数 NFORT_LIBRARY_PATH で指定されたディレクトリ	←
/opt/nec/ve3/nfort/バージョン番号/lib	/opt/nec/ve/nfort/バージョン番号/lib
環境変数 VE_LIBRARY_PATH で指定されたディレクトリ	←
/opt/nec/ve3/lib/gcc	/opt/nec/ve/lib/gcc
/opt/nec/ve3/lib	/opt/nec/ve/lib

③ コンパイラオプションの変更点

AOBA-S、AOBA-A ではコンパイラオプションの既定値が、以下のように変更されます。

AOBA-S	AOBA-A
-march=ve3	-march=ve1

-mfp16-format=ieee	-mfp16-format=none

-march：ターゲットのアーキテクチャを指定する。kind に指定できる値は以下である。

ve1 VE1 以降で使用できるオブジェクトを生成する。

ve3 VE3 以降で使用できるオブジェクトを生成する。(既定値)

-mfp16-format：半精度浮動小数点の形式を指定する。kind に指定できる値は以下である。

-march=ve3 有効時のみ指定可能。

none 半精度浮動小数点の形式を指定しない。

ieee IEEE binary16 形式を使用する。(-march=ve3 有効時の既定値)

bfloat bfloat16 形式を使用する。

④ 半精度浮動小数点の利用

AOBA-S では半精度浮動小数点を使用したオブジェクトファイルを生成し実行できます。

半精度 浮動小数点を使用したオブジェクトファイルは AOBA-A では生成、実行できません。

半精度浮動小数点の種別

半精度浮動小数点の種別は-mfp16-format による半精度浮動小数点形式の指定と

プログラム中の半精度浮動小数点の使用有無で以下のように決まります。

半精度浮動小数点の使用	半精度浮動小数点の形式(-mfp16-format)		
	none	ieee	bfloat
なし	種別なし	種別なし	種別なし
あり	種別なし	binary16	bfloat16

binary16 と bfloat16 の混在リンク

binary16 と bfloat16 のオブジェクトファイルを混在リンクし、一つのオブジェクトファイル、

実行ファイル、共有ライブラリを生成できません。

⑤ 注意事項

・AOBA-S と AOBA-A のオブジェクトファイルの混在した静的ライブラリ、共有ライブラリ、実行ファイルを生成できません。

作成しようとするとリンク時に以下のエラーとなります。

```
/opt/nec/ve/bin/nld: a.o: this object cannot use on ve3.
/opt/nec/ve/bin/nld: failed to merge target specific data of file a.o
```

・AOBA-S で AOBA-A の実行ファイルを実行したとき、正しいトレースバック情報を出力できません。

2. プログラム開発環境

1) module コマンドの利用方法

既定値のプログラム開発環境は、NEC SDK(最新バージョン)です。
Intel コンパイラを利用する場合には、module コマンドを使用します。
module コマンドを使用することで、アプリケーションの利用に必要な環境変数を統一的に設定することができます。

【module コマンドの利用方法】

利用できるアプリケーションの module 名を表示

$ module avail

※表示される Intel oneAPI モジュールの概要について"4) module コマンド補足事項"を参照ください。

アプリケーションの module を読み込む

$ module load (module 名)

※依存関係により複数のモジュールがロードされることがあります。

※Intel OneAPI 2024 では依存関係のあるモジュールがデフォルトでロードされません。
依存関係のモジュールをロードするため、環境変数へ「MODULES_AUTO_HANDLING=1」を設定しています。
該当の設定を無効化する場合には、「MODULES_AUTO_HANDLING=0」を設定してから module コマンドを実行してください。

主要なモジュール

VE 向け開発環境	ロードするモジュール名
ncc(5.2.0)	ncc/5.2.0 (既定値)
necmpi(3.6.0)	necmpi/3.6.0 (既定値)
nfort(5.2.0)	nfort/5.2.0 (既定値)
nlc(3.1.0)	nlc/3.1.0 (既定値)

※ログイン後は上記のモジュールがロードされた状態となっています。

VH 向け開発環境	ロードするモジュール名
Intel oneAPI	compiler/latest
Intel MPI	compiler/latest mpi/latest

Intel MKL	compiler/latest mkl/latest

※同じモジュール名で 32 が末尾についているモジュールは 32bit 版です。

実行例) Intel oneAPI でコンパイル、VH 上でプログラムを実行する場合

```
$ module load compiler/latest
```

実行例) Intel MPI でコンパイル、VH 上でプログラムを実行する場合

```
$ module load compiler/latest mpi/latest
```

実行例) Intel MKL でコンパイル、VH 上でプログラムを実行する場合

```
$ module load compiler/latest mkl/latest
```

現在読み込んでいる module の確認

```
$ module list
```

指定した module を破棄

```
$ module unload (module 名)
```

実行例)

```
$ module unload compiler/latest
```

現在読み込まれている全ての module を破棄

```
$ module purge
```

module load 時に読み込まれる環境変数を確認

```
$ module show (module 名)
```

実行例)

```
$ module show compiler/latest
```

2) VE 用プログラム開発環境　NEC SDK

① VE 内実行プログラム

コンパイラコマンド：　nfort(Fortran コンパイラ)
ncc(C コンパイラ)
nc++(C++コンパイラ)

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)　$ nfort –O3 sample.f

コンパイルの主なオプション

-O4	Fortran 言語仕様を逸脱した副作用を伴う最大限の最適化・自動ベクトル化を適用する
-O3	副作用を伴う最適化・自動ベクトル化、および、多重ループの最適化を適用する
-O2	副作用を伴う最適化・自動ベクトル化を適用する(デフォルト)
-O1	副作用を伴わない最適化・自動ベクトル化を適用する
-O0	最適化、自動ベクトル化、並列化、インライン展開を適用しない
-mparallel	自動並列化を利用する
-fopenmp	OpenMP 並列化を利用する
-mno-parallel-omp-routine	-mparallel と-fopenmp が同時に指定された時、OpenMP ディレクティブを含むルーチンを自動並列化しない
-finline-functions	自動インライン展開を利用する
-fbounds-check	バウンスチェック(配列の上下限のチェック)を行う
-ftrace	性能解析 ftrace 機能用の実行ファイルを作成する
-report-diagnostics	ベクトル化診断リスト(ソースファイル名.L)を出力する
-fdiag-vector=2	詳細なベクトル化診断メッセージを出力する(既定値は、-fdiag-vector=1)
-report-format	ベクトル化、並列化などの最適化情報がソース行とともに出力された編集リスト(ソースファイル名.L)を出力する

NEC SDK のコンパイルオプションの詳細は以下を参照ください。
https://sxauroratsubasa.sakura.ne.jp/Documentation(Japanese)
⇒SDK
⇒C/C++ Compiler ユーザーズガイド
⇒Fortran Compiler ユーザーズガイド

② MPI 実行プログラム

コンパイラコマンド： mpinfort(MPIFortran コンパイラ)
mpincc(MPIC コンパイラ)
mpinc++(MPIC++コンパイラ)

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) $ mpinfort –O3 sample.f

③ MPI プログラムの VE コア割り当て

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

例) 160MPI プロセスを実行

```
#!/bin/sh
#PBS -T necmpi
#PBS -l elapstim_req=4:00:00
#PBS --venode 10
cd $PBS_O_WORKDIR
mpirun –np 160 ./a.out
```

⇒1VE あたり 16 コアのため 160 プロセス(160 コア)を動作させる場合は 10VE を確保します。

1 ノードは 8VE で構成されているため、先頭ノードの 8VE に 128 プロセス（16 コア×8VE）割り当て、次ノードの先頭 2VE に 32(16 コア×2VE)プロセスが割り当てられます。

④ MPI プロセスの出力分離

MPI 実行プログラムでは、各 MPI プロセスの出力が、標準出力、標準エラー出力ファイルのそれぞれ一つのファイルに混在して記録されます。MPI プロセス毎に出力を分離したい場合には、分離するスクリプトファイルを、実行プログラムの前に指定して実行します。

```
mpirun –np 256 /opt/nec/ve/bin/mpisep.sh ./a.out
```

ファイルの分離は、複数の方式があり、環境変数 NMPI_SEPSELECT で指定します。

NMPI_SEPSELECT：

1	各 MPI プロセスの標準出力だけを stdout.uuu:rrr へ保存
2	各 MPI プロセスの標準エラー出力だけを stderr.uuu:rrr へ保存(既定値)
3	各 MPI プロセスの標準出力 および 標準エラー出力を，それぞれ

	stdout.uuu:rrr および stderr.uuu:rrr に保存
4	各 MPI プロセスの標準出力 および 標準エラー出力を，同一のファイル std.uuu:rrr に保存

例)

```
#!/bin/sh
#PBS –q sxs
#PBS -T necmpi
#PBS -l elapstim_req=4:00:00
#PBS --venode 32
#PBS -A kakin1
cd $PBS_O_WORKDIR
export NMPI_SEPSELECT=3
mpirun –np 512 /opt/nec/ve/bin/mpisep.sh ./a.out
```

3) VH 用プログラム開発環境 Intel oneAPI

①Intel oneAPI のための環境設定

module コマンドで必要なモジュールを読み込んで環境を設定します。

module コマンドの使用方法については「1) module コマンドの利用」をご確認ください。

② VH 内実行プログラム

コンパイラコマンド： ifx(Fortran コンパイラ)
icx(C コンパイラ)
icpx(C++コンパイラ)

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) $ module load compiler/latest

※module コマンドはセッションの最初だけ実行します

$ ifx –O3 sample.f

コンパイルの主なオプション

-Ofast	-O3+αの最適化を有効にする
-O3	-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
-O2	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(デフォルト)

-O1	オブジェクトサイズを増加させる最適化を無効にする
-O0	すべての最適化を無効にする
-qopenmp	OpenMP 並列化を利用する
-qopt-report	最適化レポートを作成する
-march=znver3	VH CPU の HW 機能で利用可能な最適化を有効にする

※「-O3」、「-Ofast」、「-march=znver3」等の最適化オプションは、プログラムの演算順序等が変わることがあります。プログラムの計算結果に問題がないことを確認の上ご使用ください。

Intel oneAPI のコンパイルコマンド詳細については以下を参照ください。

https://www.xlsoft.com/jp/products/intel/tech/documents.html

⇒「製品ドキュメント」タブ

⇒インテル® oneAPI (インテル® oneAPI DPC++/C++、Fortran コンパイラー)

⇒バージョン 2024

⇒Fortran コンパイラー・クラシックおよびインテル® Fortran コンパイラー・デベロッパー・ガイドおよびリファレンス (英語)

⇒oneAPI DPC++/C++ コンパイラー デベロッパー・ガイドおよびリファレンス (英語)

② MPI 実行プログラム

コンパイラコマンド： mpiifx(MPIFortran コンパイラ)
mpiicx(MPIC コンパイラ)
mpiicpx(MPIC++コンパイラ)

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) $ module load mpi/latest compiler/latest

※module コマンドはセッションの最初だけ実行します

$ mpiifx –O3 sample.f

詳細は以下を参照ください。

https://www.xlsoft.com/jp/products/intel/cluster/mpi/index.html

⇒「技術情報」タブ

⇒バージョン 2021

⇒デベロッパー・リファレンス・ガイド (英語)

4) module コマンド補足事項

module avail コマンドで表示される Intel oneAPI モジュールの一覧

モジュール名	概要
Advisor	Intel(R) Advisor
Ccl	Intel(R) oneAPI Collective Communications Library
Clck	Intel(R) Cluster Checker
compiler	Intel 64-bit compiler(s).
compiler32	Intel 32-bit compiler(s).
compiler-rt	runtime libraries for Intel 64-bit compiler(s).
compiler-rt32	runtime libraries for Intel 32-bit compiler(s).
debugger	The gdb-oneAPI debugger (an extension of standard gdb).
dev-utilities	Add oneap-cli sample browser to PATH and oneAPI samples include dirs into CPATH.
Dnnl	IntelR oneAPI Deep Neural Network Library
dnnl-cpu-gomp	IntelR oneAPI Deep Neural Network Library
dnnl-cpu-iomp	IntelR oneAPI Deep Neural Network Library
dnnl-cpu-tbb	IntelR oneAPI Deep Neural Network Library
Dpct	Intel(R) DPC++ Compatibility Tool.
Dpl	Intel(R) DPC++ Library.
Icc	Intel 64-bit Classic Compiler (icc)
icc32	Intel 32-bit Classic Compiler (icc)
init_opencl	Submodule for Intel(R) oneAPI DPCPP compiler FPGA environment
inspector	Intel(R) Inspector
intel_ipp_intel64	Intel(R) Integrated Performance Primitives Intel(R) 64 architecture.
intel_ippcp_intel64	Intel(R) Integrated Performance Primitives Cryptography Intel(R) 64 architecture.
Itac	Intel(R) Trace analyzer and Collector
Mkl	Intel(R) oneAPI Math Kernel Library (oneMKL) IA-64 architecture
mkl32	Intel(R) oneAPI Math Kernel Library (oneMKL) IA-32 architecture
Mpi	Intel(R) MPI Library

Tbb	Intel(R) oneAPI Threading Building Blocks
Vtune	Intel(R) VTune(TM) Profiler

詳細については製品ドキュメントをご確認ください。

https://www.xlsoft.com/jp/products/intel/tech/documents.html

⇒「製品ドキュメント」タブ

3. プログラム実行環境

1) VE 用プログラム開発環境　NEC SDK のバッチリクエストによる実行

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。

利用者のログイン環境によっては module の load に失敗することがあります。

投入用スクリプト内で「module load ～」行の前に、以下を記載することで回避できます。

source /etc/profile.d/modules.sh（投入用スクリプトが sh or bash の場合）
source /etc/profile.d/modules.csh（投入用スクリプトが csh の場合）

① VE 内実行プログラム用ジョブスクリプト

例(sh 形式の場合)：

```
#!/bin/sh
#PBS –q sxs                          ・・・・・ (a)
#PBS -l elapstim_req=2:00:00         ・・・・・ (b)
#PBS --venode 1                      ・・・・・ (c)
#PBS –A kadai1                       ・・・・・ (d)

export VE_OMP_NUM_THREADS=8          ・・・・・ (e)
cd $PBS_O_WORKDIR                    ・・・・・ (f)
./a.out                              ・・・・・ (g)
```

記述詳細：

(a)　：リクエスト投入先キュー として AOBA-S 用キューを指定
(b)　：最大実行経過時間を指定(hh:mm:ss 形式)
(c)　：使用 VE 数を指定，VE 内実行プログラムの場合には 1 固定
(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ェクトコードに課金
(e)　：自動並列/OpenMP 並列実行プログラムの場合、並列数を指定
指定が無い場合は、16(既定値)で実行される
(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

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 <ジョブスクリプトファイル名>
例)
$ qsub run.sh
```

② VE 内 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。

例(sh 形式の場合)：

```
#!/bin/sh
#PBS –q sxs                          ····· (a)
#PBS -T necmpi                       ····· (b)
#PBS -l elapstim_req=4:00:00         ····· (c)
#PBS --venode 32                     ····· (d)
#PBS -A kakin1                       ····· (e)
cd $PBS_O_WORKDIR                    ····· (g)
mpirun –np 512 ./a.out               ····· (h)
```

記述詳細：

(a) ： リクエスト投入先キュー として AOBA-S 用キューを指定

(b) ： MPI 実行環境を指定

(c) ： 最大実行経過時間を指定(hh:mm:ss 形式)

(d) ： 使用 VE 数を指定

(e) ： 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

(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(--venode 指定数×16 コアまで)

NEC MPI のプログラム実行方法の詳細は以下を参照ください。

https://sxauroratsubasa.sakura.ne.jp/Documentation(Japanese)

⇒NEC MPI

⇒NEC MPI ユーザーズガイド

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 <ジョブスクリプトファイル名>
例)
$ qsub run.sh
```

③ VE 内自動並列、MPI 併用実行プログラム用ジョブスクリプト

自動並列と MPI を併用した実行プログラムは、mpirun コマンドで実行プログラムを実行します。

例(sh 形式の場合)：

```
#!/bin/sh
#PBS –q sxs                          ····· (a)
#PBS -T necmpi                       ····· (b)
#PBS -l elapstim_req=4:00:00         ····· (c)
#PBS --venode 32                     ····· (d)
#PBS –v VE_OMP_NUM_THREADS=16        ····· (e)
#PBS -A kakin1                       ····· (f)
cd $PBS_O_WORKDIR                    ····· (g)
mpirun –np 32 ./a.out                ····· (h)
```

記述詳細：

(a) ： リクエスト投入先キュー として AOBA-S 用キューを指定

(b) ： MPI 実行環境を指定

(c) ： 最大実行経過時間を指定(hh:mm:ss 形式)

(d) ： 使用 VE 数を指定

(e) ： 自動並列実行プログラムの場合、並列数を指定

(f) ： 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

(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
-np オプションで MPI 総プロセス数を指定(--venode 指定数)

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 <ジョブスクリプトファイル名>
例)
$ qsub run.sh
```

2) VH 用プログラム開発環境　Intel oneAPI のバッチリクエストによる実行

利用者のログイン環境によっては module の load に失敗することがあります。

投入用スクリプト内で「module load　～」行の前に、以下を記載することで回避できます。

source /etc/profile.d/modules.sh（投入用スクリプトが sh or bash の場合）

source /etc/profile.d/modules.csh（投入用スクリプトが csh の場合）

① VH 内実行プログラム用ジョブスクリプト

例(sh 形式の場合)：

```
#!/bin/sh
#PBS -q sxs ・・・・・ (a)
#PBS -l elapstim_req=4:00:00 ・・・・・ (b)
#PBS -b 1 ・・・・・ (c)
#PBS –v OMP_NUM_THREADS=120 ・・・・・ (d)
#PBS –A kakin1 ・・・・・ (e)
module load  compiler/latest ・・・・・ (f)
cd $PBS_O_WORKDIR ・・・・・ (g)
./a.out ・・・・・ (h)
```

記述詳細：

(a) ： リクエスト投入先キュー として AOBA-S 用キューを指定

(b) ： 最大実行経過時間を指定(hh:mm:ss 形式)

(c) ： 使用 VH 数を指定

(d) ： 並列数を指定

(e) ： 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

(f) ： 必要なアプリケーションのモジュールを読み込み
※コンパイル時に利用したモジュールを指定

(g) ： カレントディレクトリへ移動

(h) ： 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。

例(sh 形式の場合)：

```
#!/bin/sh
#PBS -q sxs ・・・・・ (a)
#PBS -T intmpi ・・・・・ (b)
#PBS -l elapstim_req=4:00:00 ・・・・・ (c)
#PBS -b 2 ・・・・・ (d)
#PBS –A kakin1 ・・・・・ (e)
module load compiler/latest mpi/latest ・・・・・ (f)
cd $PBS_O_WORKDIR ・・・・・ (g)
mpirun -np 240 ./a.out ・・・・・ (h)
```

記述詳細：

(a) ： リクエスト投入先キュー として AOBA-S 用キューを指定

(b) ： MPI 実行環境を指定

(c) ： 最大実行経過時間を指定(hh:mm:ss 形式)

(d) ： 使用 VH 数を指定

(e) ： 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

(f) ： 必要なアプリケーションのモジュールを読み込み
※コンパイル時に利用したモジュールを指定

(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
-np オプションで MPI 総プロセス数を指定(-b 指定 VH 数×120 まで)

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 <ジョブスクリプトファイル名>
例)
$ qsub run.sh
```

インテル MPI ライブラリーの利用については以下を参照ください。

https://www.xlsoft.com/jp/products/intel/tech/documents.html

⇒「製品ドキュメント」タブ
⇒インテル® MPI ライブラリー
⇒バージョン 2021
⇒デベロッパー・リファレンス・ガイド (英語)

③ MPI プログラムの CPU コアの割り当て

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。
ノード毎のプロセス数割り当てを指定する場合には、mpirun コマンドでノードあた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 -q sxs
#PBS -T intmpi
#PBS -l elapstim_req=4:00:00
#PBS -b 2
#PBS –A kakin1
module load compiler/latest mpi/latest
cd $PBS_O_WORKDIR
mpirun -ppn 80 -np 160 ./a.out
```

-ppn : ノード当りの MPI プロセス数を指定

3) 会話リクエストによる実行

会話リクエストは、会話的に演算サーバを利用することで、演算サーバ上でプログラムのコンパイラや実行およびデバック実行を行うことができるリクエストです。

【実行方法】

```
qlogin -q inter -T necmpi
          (a)       (b)
```

(a) : キュー名
(b) : MPI 実行環境を指定
※MPI 実行する場合のみ指定する
NEC SDK(VE)時は necmpi、Intel oneAPI (VH)時は intmpi を指定する

会話リクエストの詳細についてはマニュアルを参照ください
https://sxauroratsubasa.sakura.ne.jp/Documentation(Japanese)
⇒NQSV 利用の手引 操作編
　⇒第 3 章 会話リクエストの操作

4. 数学ライブラリの利用

1) NLC ライブラリ(VE 用)

AOBA-S のアーキテクチャに対応した科学技術計算ライブラリ NLC(NEC Numeric Library Collection)を利用できます。

NLC は、広範な分野の数値シミュレーションプログラムの作成を強力に支援する数学ライブラリのコレクションであり、AOBA-S のベクトルエンジンに対応しています。 NLC を用いることにより、難解な数値計算アルゴリズムの詳細に煩わされることなく高度な科学技術計算プログラムを作成することができ、数値シミュレーションプログラム開発の生産性を大幅に改善することができます。

NLC の詳細および利用方法は、以下を参照ください。

https://sxauroratsubasa.sakura.ne.jp/Documentation(Japanese)

⇒SDK

⇒NLC (NEC Numeric Library Collection) ユーザーズガイド

2) Intel MKL(VH 用)

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、以下を参照してください。

https://www.xlsoft.com/jp/products/intel/perflib/mkl/index.html

⇒「技術情報」タブ

⇒バージョン 2024

⇒デベロッパー・リファレンス (英語)