ユーザーマニュアル

開発環境編

目次

1　はじめに

この度は当社計算機をお買い求め頂きまして誠にありがとうございます。

本マニュアルは当社計算機で使用する開発環境について概説します。当社の計算機は製品ごとにインストールされている環境は異なり、通常ではここで紹介する全ての開発環境をインストールしておりません。

本マニュアルをご覧の際は計算機にインストールされているパッケージをご確認の上、ご使用になる開発環境の項目をご確認頂くようお願い致します。

2 Intel Parallel Studio XE

2.1 インストールディレクトリ

2019 年現在、Intel Compiler に代表される、Intel 社製アプリケーション開発用ソフトウェアは、Intel Parallel Studio XE という開発パッケージに統合されています。

Intel Parallel Studio XE には複数の開発用ソフトウェアが同梱されていますが、本文章では、Intel Compiler 及び、Intel Compiler に同梱されている MKL に絞って解説を行っていきます[1]。

Intel Parallel Studio XE のパッケージはインテル レジストレーション・センターのホームページで配布されています。ライセンスをお持ちの場合、Intel Parallel Studio XE は以下のサイトからパッケージをダウンロードすることができます。
https://registrationcenter.intel.com/regcenter/register.aspx

Intel Parallel Studio XE は毎年更新され、発売された西暦年が Version となっています。また、そこに含まれる Intel Compiler は、Intel Parallel Studio XE の Version（西暦年）の末尾２桁が Major Version となります。
例： Intel Parallel Studio XE 2018 → Intel Compiler 18.0.X　X は update の回数

計算機にセットアップされている Intel Parallel Studio XE は、以下のディレクトリにインストールされています。

パッケージ	ディレクトリ
Intel Parallel Studio XE 20XX 例） Intel Parallel Studio XE 2018	/opt/intel/psxe20XX /opt/intel/psxe2018

このインストールディレクトリは、Intel Parallel Studio XE のデフォルトとは異なっております。変更した理由は、複数の Intel Compiler（ Parallel Studio XE ）を混在させた場合に、一部の環境変数が重複し、不具合が発生するためです[2]。

[1] Intel Parallel Studio XE が Cluster Edition の場合、IntelMPI も同梱されていますが、こちらについては、3 章にて解説を行っていきます。
[2] Compiler の MAN の参照先 が重複し､使用している Version の設定が反映されなくなります。

2.2 Intel Compiler の使用方法

Intel Compiler を使うには以下コマンドを使用します。出荷設定では root 以外の全ユーザーが使用できます。

コマンド	内容
ifort	Fortran 言語用コンパイラ
icc	C 言語用コンパイラ
icpc	C++ 用コンパイラ

コンパイラのバージョンは、"-V "オプションで確認できます。例： ifort -V

出荷設定では icc / ifort コマンドに以下のエイリアス設定をしています。icc / ifort コマンドを使用する際、以下オプションが自動で適用されます。

オプション	内容
-D_FILE_OFFSET_BITS=64	プログラム上で 2GB を超えるサイズのファイルを取り扱うために使用します。 (icc で標準使用します。)
-D_LARGEFILE_SOURCE	プログラム上で 2GB を超えるサイズのファイルを取り扱うために使用します。 (icc で標準使用します。)
-assume buffered_io	ファイルに書き込みを行う際、ディスクにすぐに書き込まず、バッファに蓄積することを指示します。 (ifort で標準使用します。)

Intel Composer でよく使用するコンパイルオプションです。

オプション	内容
-O0 -O1 -O2 -O3	ソースコードの最適化を制御します。 -O0 ～ -O3 のうち、１つを選択して指定します。 明示的に指定しない場合（デフォルト設定）は -O2 となります。 一般的に、-O に続く数字が大きい程パフォーマンスが向上しますが、数値精度に問題が出る場合があります。 -O0：最適化なし。パフォーマンスが非常に低下するので注意が必要となります。 -O1：一部の最適化を有効化します。 -O2：デフォルト設定。ベクトル化が有効となります。 -O3：-O2 より強力な最適化を行います。
-parallel	作成された実行ファイルがスレッド並列で動作するようになります。
-qopenmp	OpenMP を使用するプログラムをコンパイルする際に使用します。 Intel Compiler の旧 Version では、-openmp というオプション名でしたが変更になりました。
-mcmodel	使用するメモリーモデルを指定します。 (例) -mcmodel=large 2GB 以上のサイズのデータが使用できます。
-fp-model	浮動小数点数の扱い方を指定します。 (例) -fp-model strict 縮約を無効にし、浮動小数点数環境の変更を可能にするプロパティを有効にします。

2.3 コンパイラバージョンの変更

Intel Compiler の環境設定を各ユーザーのホームディレクトリのファイルで行っています。

具体的には、ユーザーのシェルスクリプトが bash の場合[3]は `~/.bashrc` 、 tcsh の場合は `~/.cshrc` ファイル内で、使用コンパイラのパッケージを設定しています。

複数のコンパイラがインストールされている場合、使用するコンパイラのパッケージを変更するには、次のように変更します。

・ユーザーのシェルが bash の場合は `~/.bashrc` 内の COMPILER を修正します。

例） `Intel Compiler を Version 18.0 から 17.0 に変更する場合`

修正前

```
COMPILER=INTEL18.0
#COMPILER=INTEL17.0
#COMPILER=INTEL15.0
#COMPILER=PGI17
#COMPILER=PGI16
#COMPILER=PGI15
```

修正後

```
#COMPILER=INTEL18.0
COMPILER=INTEL17.0
#COMPILER=INTEL15.0
#COMPILER=PGI17
#COMPILER=PGI16
#COMPILER=PGI15
```

・ユーザーのシェルが tcsh の場合は `~/.cshrc` 内の COMPILER を修正します。

例） `Intel Compiler を Version 18.0 から 17.0 に変更する場合`

修正前

```
set COMPILER=INTEL18.0
#set COMPILER=INTEL17.0
#set COMPILER=INTEL15.0
#set COMPILER=PGI17
#set COMPILER=PGI16
#set COMPILER=PGI15
```

修正後

```
#set COMPILER=INTEL18.0
set COMPILER=INTEL17.0
#set COMPILER=INTEL15.0
#set COMPILER=PGI17
#set COMPILER=PGI16
#set COMPILER=PGI15
```

[3] 各ユーザーが使用するシェルは `"echo $0"` コマンド等で確認できます。

次回ユーザーがログインした時からコンパイラのバージョンが変わります。
バージョンが変更されたことを、念のため "ifort -V" や "icc -V" でご確認下さい。

コンパイラバージョン変更時の注意

上記のインテルコンパイラのバージョンを変更すると、環境変数 LD_LIBRARY_PATH 等が変わります。コンパイラのバージョン変更後に以前コンパイルしたプログラムを動作すると、実行ファイルが正しいライブラリを参照できず、正常に動作しない場合があります。その際は実行するプログラムをコンパイルし直すか、使用するコンパイラのバージョン毎に新しいユーザーを作成するなどをしてご対応下さい。

2.4 Intel Math Kernel Library

Intel Compiler には Intel Math Kernel Library (MKL)パッケージが同梱されています。
MKL とは、ベクトル演算関数、高速フーリエ変換 (FFT)等、科学技術計算向けで良く使用される処理を高速で処理するライブラリ群で、特に Intel 社製 CPU で実行する場合に非常に高い性能を示します。

MKL の blas / lapack 等を使用する場合のコンパイル例は以下のとおりです。

(例 1) Intel LP64 インターフェースを使用したダイナミックリンクでのコンパイル

```
ifort myprog.f -L${MKLPATH} -I${MKLINCLUDE}
-lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread -lm
```

(例 2) Intel LP64 インターフェースを使用したスタティックリンクでのコンパイル

```
ifort myprog.f -L${MKLPATH} -I${MKLINCLUDE}  -Wl,--start-group
${MKLPATH}/libmkl_intel_lp64.a ${MKLPATH}/libmkl_intel_thread.a
${MKLPATH}/libmkl_core.a -Wl,--end-group -liomp5 -lpthread -lm
```

※${MKL_PATH}、${MKLINCLUDE}は使用する Intel Compiler のバージョンで変わります。

MKL はビルド状況に応じて、オプションやリンクするライブラリを選択、変更する必要があり、リンクオプションは非常に複雑となります。

Intel は公式サイトで、状況に応じたリンクオプションをナビゲートするサイト、
Intel® Math Kernel Library Link Line Advisor サイトを開設しています。
具体的なリンクオプションは以下のサイトを参照ください。

https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor

2.5 ドキュメント

Intel Parallel Studio XE のマニュアルは以下にインストールされています。
使用方法や仕様の詳細が知りたい場合には参照ください。

パッケージ	マニュアルのディレクトリ
Intel Parallel Studio XE 20XX 例）Intel Parallel Studio XE 2018	/opt/intel/psxe20XX/documentation_20XX /opt/intel/psxe2018/documentation_2018

以上のディレクトリにあるドキュメントですが、基本的に Intel 社のオンラインマニュアルを参照する形式をとっています。そのため、計算機がインターネットに接続できない環境下では参照できる情報が多くないことにご注意ください。

また、Intel Parallel Studio XE の代理店のエクセルソフト(株)のサイトより、各種ドキュメントが配布されています。以下 URL をご参照下さい。

https://www.xlsoft.com/jp/products/intel/studio_xe/index.html#documents

2.6 サポート期間

当社で販売している Intel Parallel Studio XE のライセンスは 1 年間のサポート期間があります。保障期間内は以下の権利があります。

(1) Intel Parallel Studio XE のパッケージがバージョンアップした際、新しいパッケージを入手して計算機にインストールすることができます。新しいパッケージはインテル レジストレーション・センターから配布されます。
(2) Intel 社のテクニカルサポート対応が受けられます。

ライセンスが切れた場合は上記の対応が受けられなくなります。サポート期限後も引き続きこれらの対応を受ける場合は有償でライセンス更新を行う必要があります。
サポート期限が切れた後も既に計算機にインストールされているコンパイラを使用することや、コンパイルしたプログラムを使用することは可能です。

Intel Parallel Studio XE のサポート期間はインテル レジストレーション・センターでログインして確認できます。また、計算機で使用されているライセンスファイルにも記述されています。ライセンスファイルは以下をご確認下さい。

・ライセンスファイルは `/opt/intel/licenses` 以下に置かれています。

確認箇所は以下の例を参考にしてください。
・ファイル例：(以下例では 2019/2/19 までです)

```
PACKAGE IC45FB71A INTEL 2019.0219 5B5F03F3893A COMPONENTS="ArBBL ¥
```

3 IntelMPI

IntelMPI とは、並列・分散プロセス間のメッセージ機能を提供する MPI 規格を実装したソフトウェアの一つです。特徴としては、

・MPICH3、MVAPICH2 をベースに Intel 社が開発を行っている有償ソフトウェア
・MPI-3.1 規格に準拠している
・特に Intel 社製ハードウェアに向けた調整がなされている。

という点が挙げられます。

過去に MPI-1 規格に準拠した mpich や lam、MPI-2 規格に準拠した mpich2、openmpi で動作していたソースコードは、ほとんどの場合、修正が必要無くそのままコンパイルできます。

なお、過去に mpich や openmpi 等でビルドを行ったアプリケーションバイナリはそのままでは IntelMPI 環境で動作しません。IntelMPI で実行するためには、ソースコードから IntelMPI でリビルドを行う必要があります。

本項では計算機にセットアップされている IntelMPI について概説します。

3.1 インストールディレクトリ

現在、IntelMPI は、Intel Parallel Studio XE とディレクトリ構造が統一されています[4]。IntelMPI の Major Version と、Intel Parallel Studio XE の Version（西暦年）は共通しているため、例えば、IntelMPI 2018 は Intel Parallel Studio XE 18.0 とディレクトリ構造が同じとなっています。

計算機にセットアップされている IntelMPI は、2.1 章で説明している Intel Parallel Studio XE との整合性をとるため、以下のディレクトリにインストールされています。

パッケージ	ディレクトリ
IntelMPI 20XX 例） IntelMPI 2018	/opt/intel/psxe20XX /opt/intel/psxe2018

[4] IntelMPI Version 2017 以降

3.2 IntelMPI のコンパイルコマンド

以下のコマンドで、MPI を使用したソースコードのコンパイルを行うことが可能です。
IntelMPI は、内部で使用するコンパイラの種類によってコンパイルコマンドが異なっており、今まで慣例的に使用されてきた mpif90 や mpicc を使用すると、Intel Compiler が使用されない点にご注意ください。

コマンド	内容
mpiifort	Intel Fortran Compiler (ifort)を使用した Fortran 用コマンド
mpiicc	Intel C Compiler (icc) を使用した C 用コマンド
mpiicpc	Intel C/C++ Compiler (icpc) を使用した C++用コマンド
mpif77	GNU Fortran compiler g77 を使用した Fortran 用コマンド
mpif90	GNU Fortran compiler gfortran を使用した Fortran 用コマンド
mpicc	I_MPI_CC 変数等で指定している場合は、指定の C Compiler 指定の無い場合は、GNU C compiler (gcc) を使用した C 言語用コマンド
mpigcc	GNU C compiler (gcc)を使用した C 言語用コマンド
mpigxx	GNU C/C++ compiler (g++)を使用した C/C++言語用コマンド

コンパイラ時のオプションについては、基本的に内部で使用されるコンパイラのオプションをそのまま使用できます。

ただし、static リンクについては注意が必要となります。
近年、OS のシステムの関係や、またはアプリケーション開発の都合上、ライブラリのリンク時に shared リンク（ dynamic リンク）を使用するケースが非常に多くなっています。

コンパイラ時のオプションで static オプションを使用することは可能ですが、アプリケーションが shared リンクを想定しており対応が難しいケースや、リンクに必要なライブラリ群が shared リンク用のパッケージしかないというケースもあり、全てのライブラリを static でリンクしたバイナリを作成する事が困難である事例も数多く見受けられます。

このような場合、static コマンドを使用してアプリケーションのビルドを行っても、作成したノード外で実行する、ノード環境を変更（更新）するなどのケースでは、アプリケーションが正常に動作しない場合があります。

3.3 実行コマンド

mpirun コマンドを使用してプログラムを実行します。形式は以下の通りです。

```
mpirun -np [並列数] [option] [実行ファイル]
```

[並列数] には、並列計算に使用するプロセスの数が入ります。通常、その数の CPU コアが使用されます。例えば、4core を使用する並列計算の場合、mpirun -np 4 … のように指定します。

[option] の項目は実行時のオプションです。オプションの詳細な解説は、OS 上の man コマンドで man mpirun と実行するか、IntelMPI のドキュメントでご確認下さい。

[実行ファイル] 指定時にはパスの指定を正確に行う必要があります。特に同名の実行ファイルが複数ある場合や、ノード間でプロセスを起動する場合には、安全に実行するため、絶対パスで指定することを強く推奨します。

例えば、プログラム test が/usr/local/bin/ に存在し、-i オプションで入力ファイル、-o オプションで出力ファイルを指定できるケースを考えます。入力ファイルに /home/hpc/test_data/inp を使用し、出力先として /home/hpc/test_out/out を指定したい場合、下記のようなコマンドラインになります。

```
mpirun -np 4 /usr/local/bin/test -i /home/hpc/test_data/inp
-o /home/hpc/test_out/out
```

また、上記の例でカレントディレクトリにプログラム test を置いたときは下記のようなコマンドラインになります。

```
mpirun -np 4 ./test -i /home/hpc/test_data/inp -o /home/hpc/test_out/out
```

3.4 計算実行時における計算ノードの指定

IntelMPI においては、計算に使用するノードは実行時に都度指定する事が必要になります。ノードの指定は実行時のコマンドで行います。

カレントディレクトリにあるプログラム `test` を、計算ノード node01 で 1 プロセス、計算ノード node02 で 3 プロセス実行させる場合は下記のコマンドとなります。

```
mpirun -np 1 -host node01 ./test : -np 3 -host node02 ./test
```

並列計算に使用する複数のノードに対して、均等に計算プロセスを割り振って実行する場合には、-ppn オプションを使用する事も可能です。
カレントディレクトリのプログラム `test` を、計算ノード node01、node02、node03、node04 でそれぞれ 2 プロセスずつ、計 8 プロセスで実行させる場合、下記のコマンドとなります[5]。

```
mpirun -ppn 2 -hosts node01,node02,node03,node04 ./test
```

なお、IntelMPI では実行時に各ノードに割り振られるプロセスは、CPU（core）に Bind される設定がデフォルトとなっています。
プロセスを CPU（core）に Bind するオプションを明示的に設定する必要はありません。

[5] -np オプションと -ppn オプションの同時使用は誤動作する場合があるため、推奨しません。

計算に使用するノードの指定をファイルで設定する事も可能です。
まず、計算に使用するノードと、そこに割り振るプロセスを指定したファイルを作成します。
例えば、node01 に 2 プロセス、node02 に 4 プロセス、node03 に 2 プロセスを実行させる場合の設定ファイルは以下のようになります。

```
node01:2
node02:4
node03:2
```

設定ファイルを作成した後、-machinefile オプションで、作成したファイルを指定します。
この設定ファイルをカレントディレクトリに machinelist という名前で保存し、
カレントディレクトリのプログラム `test` を実行させる場合、下記のコマンドとなります

```
mpirun -machinefile ./machinelist ./test
```

この場合、設定ファイルに書かれた割り振り方を使用して並列計算が行われます。
（ 具体例で示した machinelist の場合は、計 8 プロセスでの並列計算が実行されます。）

また、-machinefile オプションは、-np オプションと組み合わせることも可能です。
具体的には以下のように指定します。

```
mpirun -machinefile ./machinelist -np 4 ./test
```

この２つのオプションを共存した場合、 -np で指定された並列数分のプロセスを、
-machinefile で指定した設定ファイルの上から順番に割り振る形で計算が行われます。
上記の例の場合、node01、node02 それぞれで 2 プロセスずつ、計 4 プロセスでの並列計算が実行されます。

前述しましたが、IntelMPI では、-machinefile で指定する設定ファイルを、計算実行時に自動的に参照させる機能を持っておりません。
ジョブスケジューラーを使用しない場合、お手数ですが計算を実行する際、毎回指定して下さるようお願いします。

3.5 並列計算時の Tips

実行するプログラムの特性によっては、実行プロセスのノードへの割り振り方がプログラムのパフォーマンスに大きな影響を与えることがあります。以下に、その例を示します。

(例) CPU コアーメモリ間の大量のデータ転送により律速しているプログラム

CPU コアの高い処理性能を十分に発揮させるには、それに見合った速度でデータをメモリから取り出せなくてはなりません。アルゴリズム上、メモリから CPU コア（のレジスタ）へのデータ転送速度が性能支配的なプログラムの場合、各プロセスが使用可能な転送速度を勘案しながらノードへのプロセス群の割り当てを選択することが重要です。

ノード上では、ノード上で稼働するプロセス群が、CPUーメモリ間のデータ転送速度を分け合って使用します。そのため、各プロセスが使用できる転送速度は、稼働しているプロセス数分の 1 になります。

例えば、各ノードに 8core を有する 4 ノードクラスタ環境で、計 16 プロセス並列を行う場合について、以下の２種類の実行例を考えます。

実行例 A：2 ノード使用し、各ノードで 8 個ずつプロセス起動する場合
実行例 B：4 ノード使用し、各ノードで 4 個ずつプロセス起動する場合

CPUーメモリ間のデータ転送速度全体を 100 とおいた場合、それぞれにおける、各 CPU コアーメモリ間のデータ転送速度は次となります。

	実行例 A	実行例 B
node01	100/8 = 12.5	100/4 = 25
node02	100/8 = 12.5	100/4 = 25
node03		100/4 = 25
node04		100/4 = 25

実行例 A は、実行例 B に比べ、各プロセスの CPU-メモリ間のデータ転送速度が２倍となります。

CPU コアーメモリ間の大量のデータ転送により律速しているプログラムでは、このデータ転送速度がプログラム全体の計算速度に大きく影響しますので、注意深くプロセス群の割り当てを選択することが重要です。

以上の例は一般論ですので、実際の動作についてはお客様ご自身でご確認いただいた上でご使用下さい。

3.6 ジョブスケジューラーとの連携方法

ジョブスケジューラー(LSF / (Open)PBS / GridEngine 等) を使用して MPI のプログラムをジョブとして投入すると、ジョブスケジューラーが自動でプログラムを実行するノードを選択します。

ジョブスケジューラーを使用して MPI ジョブを投入する方法につきましては、各ジョブスケジューラーのマニュアルをご確認下さい。

4 PGI Professional Edition

PGI コンパイラは The Portland Group Inc 社製コンパイラです。科学技術計算分野において非常に多くの実績があり、同分野において採用される事の多いコンパイラの１つです。

PGI コンパイラの特色の１つとして、GPU を使用するようなソースコードへの対応に力を入れている点が挙げられ、OpenACC 規格や、NVIDIA 社の CUDA ライブラリとの連携がスムーズに行われるように設計されています。

2019 年 10 月現在、PGI コンパイラには無償版の PGI Community Edition と、有償版の PGI Professional Edition がありますが、この章では後者のセットアップ構成について概説します。

4.1 インストールパッケージ

PGI Professional Edition のパッケージは代理店のソフテック社より配布されています。以下 URL からダウンロードしたパッケージをインストールしております。

http://www.softek.co.jp/SPG/ftp.html

PGI Professional Edition は `/opt/pgi` ディレクトリにインストールしています。CUDA OpenMPI 等の追加モジュールもすべて一緒にインストールしています。

PGI のライセンスサーバーは OS ブート時に自動で起動します[6]。

4.2 使用するコマンド

PGI Professional Edition では以下コマンドを使用します。
出荷設定では root 以外の全ユーザーが使用できます。

コマンド	内容
`pgf77`	Fortran 77 用
`pgfortran(,pfg95,pgf90)`	Fortran 77/ 90 / 95 /2003 用 内部的には３つのコマンドは同一
`pgcc`	C 言語用
`pgc++`	C++ 用

[6] ライセンスサーバー起動スクリプト：`/etc/rc.d/init.d/lmgrd-pgi`

4.3 環境設定

PGI Professional Edition を使用するための環境設定は各ユーザーのホームディレクトリのファイルで行っています。bash をご使用の場合は ~/.bashrc ファイル、tcsh をご使用の場合は ~/.cshrc ファイル内で、PGI コンパイラのシェル環境設定ファイルをロードします。

~/.bashrc もしくは~/.cshrc ファイル内に変数 COMPILER があります。変数 COMPILER に PGI16 を選択していると、ユーザーは PGI Professional Edition Version16 を使用できます。

4.4 コンパイラバージョンの変更

PGI Professional Edition の使用環境は各ユーザーのホームディレクトリのファイルで設定されます。使用するコンパイラのパッケージを変更する際は次のファイルの COMPILER の変数の値を変更します。

・ユーザーのシェルが bash の場合は ~/.bashrc 内の COMPILER を修正します。

例） Intel Compiler Version 18.0 から PGI Professional Edition 17.0 に変更する場合

修正前

```
COMPILER=INTEL18.0
#COMPILER=INTEL17.0
#COMPILER=INTEL15.0
#COMPILER=PGI17
#COMPILER=PGI16
#COMPILER=PGI15
```

修正後

```
#COMPILER=INTEL18.0
#COMPILER=INTEL17.0
#COMPILER=INTEL15.0
COMPILER=PGI17
#COMPILER=PGI16
#COMPILER=PGI15
```

・ユーザーのシェルが tcsh の場合は ~/.cshrc 内の COMPILER を修正します。

例） Intel Compiler Version 18.0 から PGI Professional Edition 17.0 に変更する場合

修正前

```
set COMPILER=INTEL18.0
#set COMPILER=INTEL17.0
#set COMPILER=INTEL15.0
#set COMPILER=PGI17
#set COMPILER=PGI16
#set COMPILER=PGI15
```

修正後

```
#set COMPILER=INTEL18.0
#set COMPILER=INTEL17.0
#set COMPILER=INTEL15.0
set COMPILER=PGI17
#set COMPILER=PGI16
#set COMPILER=PGI15
```

※各ユーザーが使用するシェルは "echo $0" コマンド等で確認します。

次回ユーザーがログインした時からコンパイラのバージョンが変わります。バージョンが変更されたことを、念のため "pgf90 -V" や "pgcc -V" でご確認下さい。
(.cshrc や.bashrc の記述の詳細については付録 A をご参照下さい。)

4.5 ドキュメント

計算機の以下のファイルから、PGI のオンラインドキュメントを参照できます。

```
/opt/pgi/linux86-64/"Version"/doc/docmentation.html
```

※ " Version" にはインストールしたパッケージのバージョンが入ります。

PGI コンパイラ代理店のソフテック社がホームページでドキュメントを公開しています。以下 URL にコンパイラについての FAQ や TIPS 等様々な情報が掲載されております。

http://www.softek.co.jp/SPG/Pgi/support.html

4.6 ライセンスファイル

PGI Professional Edition のライセンスファイルは `/opt/pgi` ディレクトリにあります。この中の `license.dat` ファイルにサポート期限の日時が記述されています。この期間の間、PGI コンパイラのアップデートを行うことや、開発元へのサポート対応が可能です。この期限を過ぎても、現在セットアップされている PGI Professional Edition を使用することはできます。

4.7 Gaussian ソースコード版での使用

Gaussian 社が販売している量子化学計算 Gaussian16、Gaussian 09 のソース版を Linux EM64T プラットフォームで使用される場合、Gaussian 社は PGI コンパイラを使用することを推奨しています。また、Gaussian 社では Gaussian のリビジョン毎に Gaussian 社がサポートする PGI Professional Edition のバージョンを指定しています。

Gaussian ソース版の使用目的で販売しました計算機では、PGI Professional Edition は最新のものではなく以下バージョンのものをセットアップしております。

Gaussian Version＋Revision	PGI Professional Edition バージョン
Gaussian16 C01 AVX, AVX2	PGI F77 18.10
Gaussian16 C01 SSE4, legacy	PGI F77 17.7
Gaussian16 B01	PGI F77 17.7
Gaussian16 A03	PGI F77 16.5
Gaussian09 E01	PGI F77 15.10

5 CUDA

CUDA は NVIDIA 社が提供する GPU を演算処理で使用するための統合開発環境です[7]。
本項では計算機にセットアップされている CUDA について概説します。

5.1 インストールパッケージ

CUDA の各パッケージを以下ディレクトリにインストールしています。

バージョン	ディレクトリ
CUDA Driver	.run ファイルをダウンロードし、インストール
CUDA Toolkit	`CUDA Version X.Y` の場合 `/usr/local/cuda-X.Y` 上記ディレクトリへのシンボリックリンクとして `/usr/local/cuda` が作成されます。

CUDA の各パッケージは出荷するシステムに合わせた、最新の安定版をインストールしております。セットアップされている CUDA のバージョンは以下コマンドで確認できます。

CUDA Driver ・・・・・・ `nvidia-smi` コマンド
CUDA Toolkit ・・・・・・ `nvcc` コマンド (root 以外のユーザーで使用可能)

5.2 使用するコマンド

GPU 用ソースファイルを CUDA でコンパイルする場合には、以下のコマンドを使用します。出荷設定では root 以外の全ユーザーが使用できます。

コマンド	内容
`nvcc`	C 言語 / C++ 用 コンパイラ（CUDA）

[7] http://developer.nvidia.com/cuda-downloads にて配布

5.3 環境設定

CUDA を使用するための環境設定は各ユーザーのホームディレクトリのファイルで行っています。bash をご使用の場合は ~/.bashrc ファイル、tcsh をご使用の場合は ~/.cshrc ファイル内で、CUDA のためのシェル環境設定ファイルをロードしています。

ロードされている CUDA のシェル環境設定ファイルは以下となります。

X.Y は CUDA の Version を表します。

bash 用　/home/.common/600-CUDA X.Y.sh

tcsh 用　/home/.common/600-CUDA X.Y.csh

例：CUDA Version 9.0 の場合　bash 用　/home/.common/600-CUDA9.0.sh

tcsh 用　/home/.common/600-CUDA9.0.csh

5.4 GPU の動作確認

GPU の状態は以下のようにしてご確認下さい。

(1) nvidia-smi コマンド

計算機に搭載している各 GPU の GPU 温度・GPU 使用率・メモリ使用率等の情報を取得できます。nvidia-smi には以下のようなオプションがあります。

オプション	内容
-q	GPU の全ての情報を出力します
-q -d XXX XXX：対象	XXX で指定した情報のみ出力します。 このオプションで指定可能な項目は以下の通りです UTILIZATION, ECC, TEMPERATURE, POWER, CLOCK, COMPUTE, PIDS, PERFORMANCE, SUPPORTED_CLOCKS, PAGE_RETIREMENT, ACCOUNTING, ENCODER_STATS
-i 0	複数の GPU が搭載されている場合、0 番として認識している GPU に対して処理を実行します
-l 2	2 秒おきに出力しつづけます。 Ctrl+C で出力を止められます。

(2) deviceQuery

nvidia-smi と同様に GPU の状態を出力します。
deviceQuery プログラムは CUDA Toolkit パッケージ内のサンプルプログラムのひとつで以下ディレクトリにあります。

/usr/local/cuda/samples/1_Utilities/deviceQuery

(3) dmesg コマンド、 /var/log/Xorg.log ファイル

それぞれの出力のうち、NVRM から始まる行が NVIDIA の GPU カードのログとなります。

NVIDIA のログのみを抽出するには、root ユーザーで以下の様にコマンドを入力します。

dmesg コマンドの場合： dmesg | grep NVRM

/var/log/Xorg.log ファイルの場合： cat /var/log/Xorg.log | grep NVRM

以下の例のように、ログ中に “ fail “ や “ Error “ 等の文字列が出力されている場合、GPU の使用時に不具合が発生している可能性があります。

GPU 使用時に不具合がある場合の dmesg の 1 例

```
# dmesg | grep NVRM
The system logs include some disturbing looking NVRM messages however:
NVRM: RmInitAdapter failed! (0x25:0xffffffff:1011)
NVRM: rm_init_adapter(1) failed
allocation failed: out of vmalloc space - use vmalloc=<size> to increase size.
NVRM: RmInitAdapter failed! (0x25:0xffffffff:1011)
NVRM: rm_init_adapter(1) failed
allocation failed: out of vmalloc space - use vmalloc=<size> to increase size.
NVRM: RmInitAdapter failed! (0x25:0xffffffff:1011)
NVRM: rm_init_adapter(1) failed
```

5.5 CUDA プログラミングガイド

PGI コンパイラを用いて CUDA プログラムを開発する場合には、PGI コンパイラ代理店のソフテック社が購入者向けに載せているテクニカル情報・コラムが有用です。

ソフテック PGI テクニカル情報・コラム（ソフテック社お客様専用）
http://www.softek.co.jp/SPG/Pgi/TIPS/para_guide.html

PGI CUDA Fortran 構文を用いた GPGPU プログラム例と、そのコンパイル・リンク方法は、次のページに解説されています。

PGI CUDA Fortran のコンパイルオプション
http://www.softek.co.jp/SPG/Pgi/TIPS/opt_cudaF.html

また、OpenACC ディレクティブを用いた GPGPU プログラム例と、そのコンパイル・リンク方法は、次のページに解説されています。

PGI アクセラレータ・コンパイル用のオプション
http://www.softek.co.jp/SPG/Pgi/TIPS/opt_accel.html

6 トラブルシューティング

現時点で判明している不具合は特にありません。

付録A

A.1 シェル環境設定ファイルの配置

弊社でセットアップをした各アプリケーションのシェル環境設定ファイルは /home/.common ディレクトリに置かれています。設定ファイルは、bash 用(拡張子が.sh)と csh 用(拡張子が.csh)の 2 種類が用意されています。ディレクトリ内は、コンパイラや MPI のバージョンに応じて、以下のように配置されています。

① /home/.common/INTEL*、/home/.common/PGI* ディレクトリ
このディレクトリはコンパイラ名・コンパイラバージョンごとに作成されており、Intel Parallel Studio XE や PGI Compiler 用のシェル環境設定ファイルが配置されます。さらに、そのコンパイラを用いてコンパイル・リンクを行ったアプリケーション用のシェル環境設定ファイルが配置されます。

② /home/.common/INTEL*/IntelMPI ディレクトリ
Intel Parallel Studio XE が設定されている環境上において、IntelMPI を使用するためのシェル環境設定ファイルが配置されます。さらに、Intel Parallel Studio XE 、IntelMPI を用いてコンパイル・リンクを行ったアプリケーションのシェル環境設定ファイルが配置されます。

③ /home/.common ディレクトリ
このディレクトリ直下には、コンパイラ・MPI に依存しないアプリケーションのシェル環境設定ファイルが配置されます。

A.2 シェル環境設定ファイルの読み込みの流れ

各ユーザーの ~/.cshrc と ~/.bashrc では変数 COMPILER と変数 MPI が定義されています。そして、変数 COMPILER で指定したコンパイラ、および変数 MPI で指定した MPI に関するシェル環境設定ファイルが、ファイル名昇順で /home/.common 以下から読み込まれます。読み込まれる順番は、前述 A.1 の箇条書きの ①、次に MPI に応じて ②、最後に③の順になります。

付録B

B.1　HPC システムズ お問い合わせ先


弊社ホームページ　http://www.hpc.co.jp/support_index.html

サポート案内やお問い合わせの多い内容など様々な情報を掲載しております。
是非ご活用ください。

HPC システムズ株式会社
〒108-0022　東京都港区海岸 3-9-15 LOOP-X 8 階

HPC 事業部


【営業】03-5446-5531　【サポート】03-5446-5532

お電話によるサポート受付は祝日、弊社指定休日を除く月曜日から金曜日の 9:30～17:30 とさせて頂きます。


【FAX】03-5446-5550


【電子メール】hpcs_support@hpc.co.jp