MPICH User's Guide

Version 4.0.2

Japanese, translated by DeepL

Source: https://www.mpich.org/static/downloads/4.0.2/mpich-4.0.2-userguide.pdf

1 Introduction

本書は、MPICH がすでにインストールされていることを前提に書かれています。MPICH のインストール方法については、MPICH インストーラガイド、または MPICH のトップ・レベルのディレクトリにある README を参照してください。本書は、MPI アプリケーションのコンパイル・リンク・実行方法と、MPICH に付属 するツールの使用方法を説明します。このマニュアルは暫定版であり、まだ完成していない部分があります。しかし、MPICH を使い始めるには十分な内容になっていると思います。

2 Getting Started with MPICH

MPICH は、MPI 規格の高性能で移植性の高い実装です。MPI-1, MPI-2, MPI-3 (動的プロセス管理、片側操作、並列入出力、その他の拡張機能を含む) のすべてを実装するように設計されています。MPICH インストーラガイド には、MPICH の設定とインストールに関するいくつかの情報があります。 MPI プログラムのコンパイル、リンク、実行に関する詳細は以下の通りです。

2.1 デフォルトの実行環境

MPICH はプロセス管理と通信の分離を実現します。MPICH のデフォルトの実行環境は Hydra と呼ばれます。他のプロセスマネージャーも利用可能です。

2.2 並列処理の実行

MPICH は `mpiexec` とその標準的な引数をすべて実装しており、さらにいくつかの拡張機能を備えています。`mpiexec` の標準的な引数についてはセクション 5.1 を、 様々なプロセス管理システムに特有の拡張についてはセクション 5 のサブセクションを参照してください。

2.3 Fortran でのコマンドライン引数

`MPICH1`(正確には `MPICH1` の `mpirun`)では、`Fortran` を含むすべてのアプリケーションプログラムでコマンドライン引数にアクセスする必要とし、MPICH1 の `configure` は、`iargc` と `getarg` の正しいバージョンを含むライブラリを見つけ、それらのライブラリを `mpifort` スクリプトが MPI プログラムをリンクするようにしています。MPICH はアプリケーションのコマンドライン引数へのアクセスを必要としないので、これらの関数はオプションであり、`configure` はこれらに対して特別なことはしません。もしアプリケーションでこれらが必要な場合は、使用する Fortran 環境でそれらが利用可能であることを確認する必要があります。使用する Fortran 環境で利用可能であることを確認する必要があります。

3 Quick Start

MPICH を使用するには、MPICH がインストールされているディレクトリを知る必要があります(あなた自身がインストールしたか、システム管理者がインストールしたかのどちらかです。この場合、`/usr/local` が一つの場所となります。MPICH がまだインストールされていない場合は、MPICH イ

ンストーラのガイド をご覧ください)。そのディレクトリの **bin** サブディレクトリをパスに入れます。（訳注：ここでのパスは Windows でいう環境変数 Path のこと）これにより、コンパイルやリンク、プログラムの実行のための様々な MPICH コマンドにアクセスできるようになります。**bin** ディレクトリにある他のコマンドは、ランタイム環境の一部を管理し、ツールを実行します。

最初に実行するコマンドのひとつは **mpichversion** で、MPICH の正確なバージョンと設定を知ることができます。このマニュアルに書かれていることのいくつかは、あなたが使っている MPICH のバージョンと、それがどのように設定されているかに依存します。

これで、MPI プログラムが実行できるようになったはずです。ここでは、MPICH をインストールしたディレクトリが **/home/you/mpich-installed** であるとします。そして、そのディレクトリをパスに追加したとします。そして、次のコマンドを実行します。

```
export PATH=/home/you/mpich-installed/bin:$PATH
```

そうしたら、**1** 台のマシン上ではありますが、次のコマンドで **MPI** プログラムを実行することができます。(訳注：要するに、**example** ディレクトリに公式によるサンプルプログラムが載っているということです)

```
cd /home/you/mpich-installed/examples
mpiexec -n 3 ./cpi
```

4 Compiling and Linking

これらのコマンドの詳細は後述しますが、ここで正常に実行できれば、**MPICH** が正しくインストールされ、**MPI** プログラムが実行できたことになります。

プログラムのコンパイルとリンクに便利な方法は、**MPICH** がビルドされたのと 同じコンパイラを使用したスクリプトを使用することです。そのコンパイラには、**mpicc, mpicxx, mpifort** があり、それぞれ **C, C++, Fortran** プログラム用です。もしこれらのコマンドのどれかが欠けている場合、それは **MPICH** がその言語をサポートせずに設定されたことを意味します。

4.1 C++での問題点

ユーザーによっては、次のようなエラーメッセージが表示されることがあります。

```
SEEK_SET is #defined but must not be for the C++ binding of MPI
```

問題は、**stdio.h** （訳注：**printf** や **scanf** などの標準的な関数を提供する **C** ヘッダー）と **MPI C++** インターフェイスの両方が **SEEK SET, SEEK CUR, SEEK END** を使っていることです。これは本当に **MPI** 標準のバグです。そこでこれらをプログラムにで **mpi.h** の **include** の前に追加してみてください。

（訳注：おそらく MPI と stdio.h で宣言されている定数の名前がかぶっている。これはバグというより仕様というよりただの開発者のミス）

```
#undef SEEK_SET
#undef SEEK_END
#undef SEEK_CUR
```

または以下の定義をコマンドラインに追加してください。

```
-DMPICH_IGNORE_CXX_SEEK
```

これにより mpi.h 内の SEEK SET 等がスキップされます。

4.2 Fortran での問題点

MPICH は、Fortran プログラムに対して 2 種類のサポートを提供しています。Fortran 77 プログラマは mpif.h に MPI COMM WORLD などの MPI 定数の定義があります。Fortran 90 プログラマは、代わりに MPI モジュールがあり、このモジュールは、多くの MPI 関数のインターフェイスを定義しています。しかし、この MPI モジュールは、Fortran 90 を完全にサポートしているわけではありません。たとえば、MPI Send のような "choice" 引数を取るルーチンのインターフェースは提供されていません。

5 Running Programs with “mpiexec”

MPI 標準規格では、MPI プログラムを実行するための推奨される方法として mpiexec が記述されています。MPICH は mpiexec 標準を実装しており、さらにいくつかの拡張機能を提供します。

5.1 Standard “mpiexec”

ここでは MPI Standard[1]にある標準的な mpiexec の引数を説明します。ローカルマシン上（単一マシン）で n 個のプロセスを持つプログラムを実行するには、次のコマンドを使用します。

```
mpiexec -n <number> ./a.out
```
（訳注：a.out は実行ファイル）

複数ノード（ネットワークで接続された複数のマシン）で実行するには、次のコマンドを使用します。

```
mpiexec -f machinefile -n <number> ./a.out
```

“machinefile”の書式は、次のようになります。

```
host1
host2:2
host3:4    # Random comments
host4:1
```

'host1'、'host2'、'host3'、'host4'は、ジョブを実行するマシンのホスト名です。'2'、'4'、'1'のコロンの後の各数値は、各ノードで実行したいプロセスの数を表します。何も指定しない場合、':1'が指定されたものとみなします。

Hydra についてのさらに詳しい情報については、
http://wiki.mpich.org/mpich/index.php/Using_the_Hydra_Process_Manager
を参照してください。

5.2 すべてのプロセス管理環境に対応する拡張機能

いくつかの mpiexec の引数は、特定の通信サブシステム（「デバイス」）やプロセス管理環境（「プロセスマネージャ」）に特有のものです。私たちの意図は、すべての引数をデバイスとプロセスマネージャの間で可能な限り統一することです。以下、これらを別々に記述します。

5.3 Hydra 用の mpiexec 拡張機能

MPICH は、数多くのプロセスマネジメントシステムを提供しています。Hydra は MPICH のデフォルトのプロセスマネージャです。Hydra の詳細および mpiexec の拡張については、以下を参照してください。
http://wiki.mpich.org/mpich/index.php/Using_the_Hydra_Process_Manager

5.4 gforker 用の拡張機能

gforker は一つのマシン上でプロセスを起動するためのプロセス管理システムで、 MPI プロセスが単に mpiexec プロセスから fork されることからこのように呼ばれています。このプロセスマネージャは MPI Commawn と他の動的プロセスルーチンを使用するプログラムはサポートしますが、 mpiexec で起動されないプログラムからの動的プロセスルーチンの使用はサポートしません。gforker プロセスマネージャは主に MPI プログラムの開発とテストを簡素化するため、デバッグの補助としてしようされます。

5.4.1 gforker 用の mpiexec コマンドライン引数

mpiexec の標準的なコマンドライン引数に加え、gforker の mpiexec は以下のオプションをサポートしています。

-np <num>　標準の -n 引数と同義です。

-env <name> <value>　環境変数 <name> に <value> を設定します。

-envnone 環境変数(他の -env または -genv 引数で指定されたもの以外)を渡さない。
デフォルトでは、すべての環境変数が各 MPI プロセスに提供されます。(根拠: ユーザが最も驚かない原則)

-envlist <list> リストされた環境変数(カンマで区切られた名前) を現在の値と共に、
mpiexec によって実行されるプロセスに渡します。

-genv <名前> <値> -genv オプションは対応する -env バージョンと同じ意味です。
ただし、現在の実行ファイルだけでなく、すべての実行ファイルに適用されます (複数の実行ファイルを指定するためにコロン構文が使用されている場合)。

-genvnone -envnone と同様ですが、すべての実行ファイルに適用されます。

-genvlist <list> -envlist と同様ですが、すべての実行ファイルに適用されます。

-usize <n> 属性 MPI UNIVERSE SIZE の値として返される値を指定します。

-l 標準出力と標準エラー(stdout と stderr)に、プロセスのランクをラベル付けします。

-maxtime <n> タイムリミットを <n> 秒に設定します。

-exitinfo 異常終了した場合、各プロセスが終了した理由をより詳しく説明します。

コマンドライン引数に加え、gforker mpiexec は mpiexec の動作を制御するために、いくつかの環境変数を提供しています。

・MPIEXEC_TIMEOUT
最大実行時間(秒)です。Mpiexec は、指定された時間以上かかる MPI プログラムを終了させます。

・MPIEXEC_UNIVERSE_SIZE
ユニバースの大きさを設定します。

・MPIEXEC_PORT_RANGE
mpiexec が起動するプロセスとの通信に使用するポートの範囲を設定します。この形式は <low>:<high> です。例えば、10000 から 10100 までの任意のポートを指定するには、10000:10100 とします。MPICH_PORT_RANGE は、MPIEXEC PORT RANGE と同じ意味をもち、MPIEXEC_PORT_RANGE が設定されていない場合に使用されます。

・MPIEXEC_PREFIX_DEFAULT
この環境変数が設定されている場合、標準出力にはプロセスの MPI_COMM_WORLD におけるランクが先頭にきます。また、標準エラー出力には、そのランクとテキスト(err)が出力され、その後に角括弧(>)が続きます。この変数が設定されていない場合、接頭辞はありません。

・MPIEXEC_PREFIX_STDOUT
標準出力に送られる行に使われるプレフィックスを設定します。%d は MPI_COMM_WORLD のランクに置き換えられ、%w は複数の MPI_COMM_WORLD を使用する MPI ジョブ(例: MPI Comm spawn や MPI Comm connect を使用するジョブ)において、どの MPI_COMM_WORLD かを示すものに置き換えられます。

・MPIEXEC_PREFIX_STDERR
MPIEXEC_PREFIX_STDOUT に似ていますが、標準エラー用です。

・MPIEXEC_STDOUTBUF 標準出力のバッファリングモードを設定します。有効な値は、NONE (バッファリングなし)、LINE (行単位でバッファリング)、BLOCK (文字のブロック単位でバッファリングします。ブロックのサイズは実装で定義されます)。デフォルトは NONE です。

・MPIEXEC_STDERRBUF
MPIEXEC_STDOUTBUF と似ていますが、標準エラー用です。

5.5 remshell プロセスマネジメント環境の制限事項

remshell プロセスマネージャは、マシンの集まりでプロセスを開始するために secure shell コマンド (ssh)を利用する非常にシンプルなバージョンの mpiexec を提供します。これは主に他のプロセスマネージャで動作するバージョンの mpiexec を構築する方法の説明として意図されているので、このドキュメントで説明されている他の mpiexec プログラムのすべての機能を実装しているわけではありません。特に、MPI プログラムに与えられる環境変数を制御するコマンドラインオプションは無視します。mpiexec の gforker 版で提供されているものと同じ出力ラベリング機能をサポートしています。しかし、このバージョンの mpiexec は MPICH-1 の ch p4 デバイス用の mpirun と同じように、リモートシェルを許可しているマシンの集まりでプログラムを実行するために 使用することができます。machines という名前のファイルには、プロセスを実行できるマシンの名前が 1 行に 1 つずつ書かれている必要があります。要求されたプロセス数を満たすだけのマシンがリストアップされていなければなりません。必要であれば、同じマシン名を複数回リストアップすることができます。

5.6 Slurm と PBS によって MPICH を使う

MPICH を Slurm や PBS と一緒に使う方法は複数あります。Hydra は Slurm と PBS の両方をネイティブにサポートしており、これらのシステムで MPICH を使う最も簡単な方法でしょう。(詳しくは上

記の Hydra のドキュメントを参照してください)。

あるいは Slurm も MPICH の内部プロセス管理インタフェースと互換性があります。これを使うには、MPICH を Slurm サポート付きで設定し、Slurm が提供する srun ジョブ起動ユーティリティを使用する必要があります。

PBS では、MPICH のジョブは次の二つの方法で起動することができます。 ①Hydra の mpiexec を PBS に対応した適切なオプションで使用する方法と、 ②OSC mpiexec を使用する方法です。

5.6.1 OSC mpiexec

オハイオ州にあるスーパーコンピュータセンターの Pete Wyckoff は、PBS システム上で MPICH ジョブを起動するための OSC mpiexec という代替ユーティリティを提供しています。これに関するより詳しい情報は http://www.osc.edu/~pw/mpiexec をご覧ください。

6 Specification of Implementation Details

MPI 標準規格では、ライブラリが独自の動作を自由に定義できる領域が数多く定義されていますが、そのような動作が適切に文書化されていることが必要です。このセクションでは、MPICH のために必要なドキュメントを提供します。

6.1 コミュニケーターのための MPI エラーハンドラー

MPI3.0 のドキュメントのセクション 8.3.1 では、MPI はエラーハンドラーのコールバック関数を次のように定義しているとあります。

```
typedef void MPI_Comm_errhandler_function(MPI_Comm *, Int *, ……);
```

第一引数は使用中のコミュニケータ、第二引数はエラーを発生させた MPI ルーチンが返すべきエラーコード、残りの引数は実装に依存する "varargs "です。MPICH はこのリストの一部として引数を提供しません。そのため、MPICH に提供されるコールバック関数は、ヘッダが

```
typedef void MPI_Comm_errhandler_function(MPI_Comm *, Int *);
```

であれば十分です。

7 Debugging

並列プログラムのデバックはそれなりに難しいです。ここでは、いくつかのアプローチについて説明します。いくつかのアプローチは MPICH の正確なバージョンに依存します。

7.1 TotalView

MPICH は Etnus 社の TotalView デバッガの使用をサポートしています。MPICH が TotalView によ

るデバッグを有効にするように設定されている場合、次のコマンドを使って MPI プログラムをデバッグすることができます。

```
Totalview -a mpiexec -a -n 3 cpi
```

すると、TotalView からのポップアップウィンドウが表示され、停止した状態でジョブを開始するかどうか尋ねられます。その場合、TotalView のウィンドウが表示されると、ソースウィンドウにアセンブリコードが表示されることがあります。スタックウィンドウ（左上）の main をクリックすると、main 関数のソースが表示されます。TotalView は、MPI Init の呼び出しでプログラム（全プロセス）が停止していることを表示します。

TotalView8.1.0 以降であれば、TotalView の機能である間接起動を MPICH で使用することができます。これには、次のコマンドを使います。

```
Totalview <program> -a <program args>
```

次に、Process/Startup Parameters コマンドを選択します。表示されたダイアログ・ボックスで Parallel タブを選択し、並列システムとして MPICH を選択します。次に Tasks フィールドでタスクの数を設定し、Additional Starter Arguments フィールドにその他の必要な mpiexec の引数を入力します。

8 Checkpointing

MPICH は、Hydra プロセスマネージャと併用することで、チェックポイント/ロールバックのフォールトトレランス（耐障害性）をサポートします。現在のところ、BLCR(Berkeley Lab Checkpoint/Restart) チェックポイントライブラリのみがサポートされています。BLCR は別途インストールする必要があります。以下では、MPICH でこの機能を有効にする方法について説明します。この情報は MPICH Wiki でも見ることができます: http://wiki.mpich.org/mpich/index.php/Checkpointing

8.1 Checkpointing の設定

まず、お使いのマシンに BLCR バージョン 0.8.2 がインストールされていることが必要です。もしデフォルトの場所にインストールされている場合は、configure コマンドに次の 2 つのオプションを追加してください。

```
--enable-checkpointing -with-hydra-ckpointlib=blcr
```

BLCR がシステムのデフォルトの場所にインストールされていない場合、MPICH の configure にその場所を教える必要があります。MPICH の configure にその場所を設定してください。また、BLCR の共有パスを設定するために LD_LIBRARY_PATH 環境変数を設定し、BLCR の共有ライブラリが見つかるよう

にする必要があるかもしれません。この場合、configure コマンドに以下のオプションを追加してください。

```
--enable-checkpointing -with-hydra-ckpointlib=blcr
--with-blcr=BLCR_INSTALL_DIR LD_LIBRARY_PATH=BLCR_INSTALL_DIR/lib
```

ここで、BLCR INSTALL DIR は BLCR がインストールされたディレクトリです（BLCR の設定時に --prefix で指定されたもの)。注意：checkpointing は Hydra プロセスマネージャでのみサポートされています。デフォルトでは Hyrda が使用されますが、"--with-pm=" configure オプションで他のものを選択した場合はその限りではありません。なお、configure オプションで他のものを選ばない限り、Hyrda がデフォルトで使われます。

設定後、通常通りコンパイル（例：make; make install）します。

8.2 Taking checkpoints

チェックポイントを使用するには、mpiexec の-ckpointlib オプションで使用するチェックポイントライブラリを指定し、-ckpoint-prefix オプションでチェックポイントイメージを書き込むディレクトリを指定します。

```
shell$ mpiexec -ckpointlib blcr ¥
       -ckpoint-prefix /home/buntinas/ckpts/app.ckpoint ¥
       -f hosts -n 4 ./app
```

アプリケーションの実行中、ユーザはいつでもチェックポイントを要求することができます。SIGUSR1 シグナルを mpiexec に送ることにより、いつでもチェックポイントを要求できます。mpiexec のオプション-ckpoint-interval を使ってチェックポイントの間隔を秒単位で指定することができます。

```
shell$ mpiexec -ckpointlib blcr ¥
       -ckpoint-prefix /home/buntinas/ckpts/app.ckpoint ¥
       -ckpoint-interval 3600 -f hosts -n 4 ./app
```

チェックポイント/リスタートパラメータは、環境変数 HYDRA_CKPOINTLIB、HYDRA_CKPOINT_PREFIX、HYDRA_CKPOINT_INTERVAL で制御できます。

各チェックポイントはノードあたり 1 つのファイルを生成します。ノード上の全プロセスのチェックポイントは同じファイルに保存されます。新しいチェックポイントが行なわれる度に、追加のファイル群が生成されます。このファイルにはチェックポイント番号が振られています。これにより、アプリケーションを再起動することができます。チェックポイント番号は-ckpoint-num パラメータで指定します。プロセスを再起動するには次のコマンドを実行します。

```
shell$ mpiexec -ckpointlib blcr ¥
       -ckpoint-prefix /home/buntinas/ckpts/app.ckpoint ¥
       -ckpoint-num 5 -f hosts -n 4
```

デフォルトでは、プロセスは最初のチェックポイントから再開されるため、ほとんどの場合、チェックポイント番号を指定する必要があることに注意してください。

9 Other Tools Provided with MPICH

MPICH は MPI 機能のテストスイートも含んでいます。このテストスイートは mpich/test/mpi のソースディレクトリにあり、実行するには make tasting コマンドを使用します。このテストスイートは MPICH だけではなく、どのような MPI プログラムでも動作するはずです。