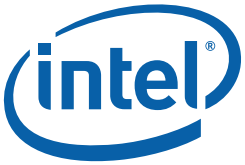

インテル® MPI ライブラリー Linux* 版

入門ガイド

インテル® MPI ライブラリーは、マルチファブリック対応のメッセージ・パッシング・ライブラリーで、MPI (Message Passing Interface) v2 (MPI-2) 仕様を実装しています。再リンクせずに、相互接続ファブリックを切り替えることができます。

この『入門ガイド』では、インテル® MPI ライブラリーを使用して、簡単な MPI プログラムのコンパイルと実行について説明します。また、基本的な使用例やトラブルシューティングのヒントも提供します。

本ガイドを印刷して、提供される例を参考にすすめてください。

目次

1 概要 4
1.1 対象ユーザー 4
1.2 Doc Type フィールドの使用 4
1.3 表記規則とシンボル 4
1.4 関連情報 5
2 インテル® MPI ライブラリーの使用 6
2.1 使用モデル 6
2.2 はじめに 6
2.3 クイックスタート 6
2.4 コンパイルとリンク 7
2.5 MPD デーモンのセットアップ 7
2.6 ネットワーク・ファブリックの選択 9
2.7 MPI プログラムの実行 9
3 トラブルシューティング 10
3.1 インストール・テスト 10
3.2 MPD セットアップのトラブルシューティング 11
3.3 テストプログラムのコンパイルと実行 12
4 次のステップ 13

著作権と商標について

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスを許諾するためのものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品性に関する保証、第三者の特許権、著作権、その他、知的所有権を侵害していないことへの保証を含む) にも一切応じないものとします。インテル製品は、医療、救命、延命措置、重要な制御または安全システム、核施設などの目的に使用することを前提としたものではありません。

インテル製品は、予告なく仕様が変更される場合があります。

本資料で説明されているソフトウェアには、不具合が含まれている可能性があり、公開されている仕様とは異なる動作をする場合があります。現在までに判明している不具合の情報については、インテルのサポートサイトをご覧ください。

本資料およびこれに記載されているソフトウェアはライセンス契約に基づいて提供されるものであり、その使用および複製はライセンス契約で定められた条件下でのみ許可されます。本資料で提供される情報は、情報供与のみを目的としたものであり、予告なく変更されることがあります。また、本資料で提供される情報は、インテルによる確約と解釈されるべきものではありません。インテルは本資料の内容およびこれに関連して提供されるソフトウェアにエラー、誤り、不正確な点が含まれていたとしても一切責任を負わないものとします。

ライセンス契約で許可されている場合を除き、インテルからの文書による承諾なく、本書のいかなる部分も複製したり、検索システムに保持したり、他の形式や媒体によって転送したりすることは禁じられています。

機能または命令の一覧で「留保」または「未定義」と記されているものがありますが、その「機能が存在しない」あるいは「性質が留保付である」という状態を開発の前提にしないでください。留保または未定義の機能を不適当な方法で使用すると、開発したソフトウェア・コードをインテル・プロセッサー上で実行する際に、予測不可能な動作や障害が発生するおそれがあります。これらの機能や命令は、インテルが将来のために留保しているものです。不正な使用により、衝突が生じたり互換性が失われたりしても、インテルは一切責任を負いません。

Intel、インテル、Intel ロゴは、アメリカ合衆国およびその他の国における Intel Corporation の商標です。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

1 概要

この『入門ガイド』は、インテル® MPI ライブラリーの使用方法とトラブルシューティングについて説明します。

1.1 対象ユーザー

この『入門ガイド』は、初めてインテル® MPI ライブラリーをインストールし、使用するユーザーを対象としています。

1.2 Doc Type フィールドの使用

この『入門ガイド』は次のセクションで構成されています。

ドキュメント構成

セクション	説明
セクション 1: 概要	このドキュメントの概要です。
セクション 2: インテル® MPI ライブラリーの使用	インテル® MPI ライブラリーの使用方法について説明します。
セクション 3: トラブルシューティング	トラブルシューティングの方法とヒントについて説明します。

1.3 表記規則とシンボル

このガイドでは、次の表記規則を使用しています。

本ドキュメントで使用されている表記規則とシンボル

This type style	ハイパーリンク
`This type style`	コマンド、引数、オプション、ファイルの名前
`THIS_TYPE_STYLE`	環境変数
`<this type style>`	実際の値のプレースホルダー
`[ items ]`	オプション項目
`{ item | item }`	| によって区切られた選択可能な項目
(SDK のみ)	ソフトウェア開発キット (SDK) ユーザー向けの情報

1.4 関連情報

インテル® MPI ライブラリーに関する情報は、次のリソースも参考にしてください。

- 製品 Web サイト
- インテル® MPI ライブラリーのサポート
- インテル® クラスターツール製品
- インテル® ソフトウェア開発製品

2 インテル® MPI ライブラリーの使用

2.1 使用モデル

インテル® MPI ライブラリーを使用するステップを次に示します。

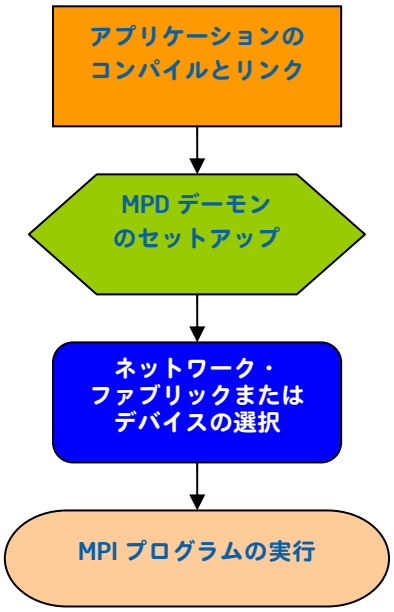

図 1: インテル® MPI ライブラリーで作業するための使用モデル・フローチャート

2.2 はじめに

インテル® MPI ライブラリーを使用する前に、ライブラリー、スクリプト、ユーティリティー・アプリケーションがインストールされていることを確認してください。インストール手順については、『インテル® MPI ライブラリー Linux* 版インストール・ガイド』を参照してください。

2.3 クイックスタート

1. 適切に環境変数を設定するには、インテル® MPI ライブラリーに含まれている `mpivars.[c]sh` スクリプトを source シェルコマンドで実行します。*`<installdir>/bin`* ディレクトリーまたは*`<installdir>`*`/bin64` ディレクトリー (インテル® 64 の 64 ビット・モードの場合) にあります。
2. 1 行に 1 つのホスト名が記されたクラスターノードのリスト、`mpd.hosts` テキストファイルを作成します。
3. **(SDK のみ)** `PATH` にコンパイラーが含まれていることを確認してください。

4. **(SDK のみ)** 適切なコンパイラー・ドライバーを使用して、テストプログラムをコンパイルします。次に例を示します。

   ```
   $ mpicc -o test <installdir>/test/test.c
   ```

5. `mpirun` コマンドを使用してテストを実行します。

   ```
   $ mpirun -n <# of processes> ./test
   ```

6. 詳細は、このドキュメントのほかの項目および『インテル® MPI ライブラリー・リファレンス・マニュアル』(英語) を参照してください。

2.4 コンパイルとリンク

(SDK のみ)

MPI プログラムをコンパイルしてインテル® MPI ライブラリーとリンクするには:

1. 使用するコンパイラーと関連するソフトウェアが `PATH` にあることを確認してください。

   インテル® コンパイラーを使用する場合は、コンパイラー・ライブラリーのディレクトリーが `LD_LIBRARY_PATH` 環境変数に含まれていることを確認してください。
   例えば、インテル® C++ コンパイラー 10.1 とインテル® Fortran コンパイラー 10.1 の場合、次のセットアップ・スクリプトを実行します。

   ```
   /opt/intel_cc_10.1/bin/iccvars.[c]sh
   ```

   および

   ```
   /opt/intel_fc_10.1/bin/ifortvars.[c]sh
   ```

2. MPI プログラムを適切な `mpi` コマンドでコンパイルします。
   例えば、GNU* C コンパイラーを使用して C コードをコンパイルするには、次のように `mpicc` コマンドを使用します。

   ```
   $ mpicc <installdir>/test/test.c
   ```

   <installdir> は、インストール・パッケージのフルパスです。

   サポートされるすべてのコンパイラー用に、標準的なコンパイラー・コマンドに接頭辞 `mpi` が付いた対応するコマンドが用意されています。例えば、インテル® Fortran コンパイラー 8.0 以上 (`ifort`) 用のインテル® MPI ライブラリー・コマンドは `mpiifort` です。

2.5 MPD デーモンのセットアップ

インテル® MPI ライブラリーでは、MPD (Multi-Purpose Daemon) ジョブ・スタートアップ・メカニズムが使用されます。`mpicc` (または関連する) コマンドを使用してコンパイルされたプログラムを実行するには、MPD デーモンをセットアップします。

システム上の全ユーザーにより使用される MPD デーモンを、システム・アドミニストレーターに 1 回起動させるのではなく、常に、専用の MPD デーモンのセットを起動し、管理してください。このセットアップにより、システムのセキュリティーが強化され、実行環境の制御を柔軟に行うことができます。

MPD デーモンをセットアップするには:

1. `.cshrc` ファイルまたは `.bashrc` ファイルなどで、適切な値とディレクトリーを使用して環境変数を設定します。

   - `PATH` 変数に、*<installdir>*`/bin` ディレクトリー (インテル® 64 の 64 ビット・モードの場合は、*<installdir>*`/bin64`) が含まれていることを確認してください。この変数を設定するには、インテル® MPI ライブラリーに含まれている `mpivars.[c]sh` スクリプトを使用します。

- PATH 変数に、Python* 2.2 以上のディレクトリーが含まれていることも確認してください。
- **(SDK のみ)** インテル® コンパイラーを使用する場合は、LD_LIBRARY_PATH 変数にコンパイラー・ライブラリーのディレクトリーが含まれていることを確認してください。この変数は、コンパイラーに含まれている *vars.[c]sh スクリプトを使用して設定します。
- アプリケーションが使用するその他の環境変数も設定します。

注: ステップ 2 から 4 はオプションです。

2. $HOME/.mpd.conf ファイルを作成します。MPD パスワードを設定するには、このファイルに次の行を入力します。

```
secretword=<mpd secret word>
```

Linux ログインパスワードは使用しないでください。任意の *<mpd secret word>* 文字列だけが、さまざまなクラスターユーザーによる MPD デーモンへのアクセスを制御します。

3. chmod コマンドを使用して $HOME/.mpd.conf ファイルを保護し、他のユーザーが読み取りおよび書き込み権限を持たないようにします。

```
$ chmod 600 $HOME/.mpd.conf
```

4. PATH 設定と .mpd.conf コンテンツをクラスターのすべてのノード上で、rsh を使用して参照できることを確認します。例えば、クラスターの各 *<node>* に対して次のコマンドを使用します。

```
$ rsh <node> env
$ rsh <node> cat $HOME/.mpd.conf
```

1 つのノードだけではなく、各ノードが他のノードに接続できることを確認します。

クラスターで rsh の代わりに ssh が使用される場合は、次の「注」 を参照してください。

5. 1 行に 1 つのホスト名が記されたクラスターノードのリスト、mpd.hosts テキストファイルを作成します。
6. mpdallexit コマンドを使用して MPD デーモンをシャットダウンします。

```
$ mpdallexit
```

7. mpdboot コマンドを使用して MPD デーモンを起動します。

```
$ mpdboot -n <#nodes>
```

$PWD/mpd.hosts ファイルがある場合は、このファイルがデフォルトとして使用されます。mpd.hosts がない場合は、mpdboot コマンドによりローカルマシンの MPD デーモンが起動されます。

8. mpdtrace コマンドを使用して MPD デーモンのステータスを確認します。

```
$ mpdtrace
```

出力は、現在 MPD デーモンを実行しているノードのリストです。このリストは、mpd.hosts ファイルの内容と一致します。

注: クラスターで rsh の代わりに ssh が使用される場合は、パスワードを入力しなくても各ノードが他のノードに接続できることを確認してください。詳細は、システムのマニュアルを参照してください。

注: クラスターで rsh の代わりに ssh が使用される場合は、-r ssh または -- rsh=ssh オプションを mpdboot 起動文字列に追加します。

2.6 ネットワーク・ファブリックの選択

インテル® MPI ライブラリーは、MPI プロセス間の通信に対して動的にファブリックを選択します。特定のファブリックの組み合わせを選択するには、I_MPI_DEVICE 環境変数を次のいずれかの値に設定します。

I_MPI_DEVICE 値	サポートされるファブリック
sock	TCP/Ethernet/ソケット
shm	共有メモリーのみ (ソケットなし)
ssm	TCP + 共有メモリー (Ethernet を介して接続される SMP クラスター用)
rdma[:<provider>]	InfiniBand*、Myrinet* (指定の DAPL* プロバイダー経由)
rdssm[:<provider>]	TCP + 共有メモリー + DAPL (RDMA 対応のファブリックを介して接続される SMP クラスター用)

選択されたファブリックが利用可能であることを確認します。例えば、shm は、すべてのプロセスが共有メモリーを介して互いに通信できるときのみに使用します。rdma は、すべてのプロセスが 1 つの DAPL プロバイダーを介して互いに通信できるときのみに使用します。

2.7 MPI プログラムの実行

インテル® MPI ライブラリーがリンクされたプログラムを起動するには、mpiexec コマンドを使用します。

```
$ mpiexec -n <# of processes> ./myprog
```

-n オプションで、プロセス数を設定します。このオプションは、mpiexec コマンドで唯一、必須のオプションです。

デフォルトのファブリックではなくネットワークのファブリックを使用する場合は、-genv オプションを使用して I_MPI_DEVICE 変数に割り当てられる値を指定します。

例えば、shm ファブリックを使用して MPI 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 -genv I_MPI_DEVICE shm -n <# of processes> ./a.out
```

rdma 対応ファブリックの場合は、次のコマンドを使用します。

```
$ mpiexec -genv I_MPI_DEVICE rdma -n <# of processes> ./a.out
```

任意のサポートデバイスを選択できます。詳細は、「ネットワーク・ファブリックの選択」を参照してください。

インテル® MPI ライブラリーを使用してアプリケーションが実行できれば、他の MPI ライブラリーでも実行できます。これで、再リンクすることなく、ノード間で異なるファブリックを使用してアプリケーションをあるクラスターから別のクラスターへ移動させることができます。問題が発生した場合は、「トラブルシューティング」を参照してください。

3 トラブルシューティング

次のセクションでは、インテル® MPI ライブラリーのインストール、セットアップ、アプリケーションの実行にかかわる問題のトラブルシューティングについて説明しています。

3.1 インストール・テスト

インテル® MPI ライブラリーがインストールされ、正しく機能していることを確認するには、一般的なテストを行い、テストプログラムをコンパイルして実行します。

インストール・テストを行うには:

1. 次のコマンドを使用して Python v2.2 以上が PATH に含まれていることを確認します。

   ```
   $ rsh <nodename> python -V
   ```

   エラーメッセージが返されたり、返された値が 2.2 よりも低い場合は、Python v2.2 以上をインストールし、インストールされた Python が PATH に含まれていることを確認します。

2. python-xml* または libxml2-python* などの Python XML モジュールがインストールされていることを確認します。

   ```
   $ rpm -qa | grep python-xml
   $ rpm -qa | grep libxml2-python
   ```

   出力に "python-xml" または "libxml2-python" という名前とバージョン番号が含まれていない場合は、必要なモジュールをインストールします。

3. expat* または pyxml* などの XML パーサーがインストールされていることを確認します。

   ```
   $ rpm -qa | grep expat
   $ rpm -qa | grep pyxml
   ```

   出力に "expat" または "pyxml" という名前とバージョン番号が含まれていない場合は、必要なモジュールをインストールします。

4. *<installdir>*/bin (インテル® 64 の 64 ビット・モードの場合は *<installdir>*/bin64) が PATH に含まれていることを確認します。

   ```
   $ rsh <nodename> which mpiexec
   ```

   テストする各ノードの適切なパスが表示されます。

   (SDK のみ) インテル® コンパイラーを使用する場合は、適切なディレクトリーが PATH 環境変数と LD_LIBRARY_PATH 環境変数に含まれていることを確認します。

   ```
   $ mpiexec -n <# of processes> env | grep PATH
   ```

   mpiexec コマンドを実行する前に mpd デーモンを開始してください。テストする各ノードのパス変数に対する適切なディレクトリーが表示されます。表示されない場合は、適切な *vars.[c]sh スクリプトを呼び出します。例えば、インテル® C++ コンパイラー 10.1 の場合、次のソースコマンドを使用します。

   ```
   $ . /opt/intel_cc_10.1/bin/iccvars.sh
   ```

5. ある特定の状況では、*<installdir>*/lib ディレクトリー (インテル® 64 の 64 ビット・モードの場合は *<installdir>*/lib64) を LD_LIBRARY_PATH に含める必要があります。次のコマンドを使用して、LD_LIBRARY_PATH 設定を確認します。

   ```
   $ mpiexec -n <# of processes> env | grep PATH
   ```

3.2 MPD セットアップのトラブルシューティング

ローカルマシンで mpd コマンドを実行できるか確認します。次のコマンドを実行します。

```
# mpd &
# mpdtrace
# mpdallexit
```

mpdtrace の出力で、実行しているマシンのホスト名が表示されます。ホスト名が表示されない場合や MPD を起動できない場合は、インストールが正しく行われているかどうか、また環境変数が正しく設定されているかどうかを確認してください。

次のトラブルシューティングのステップは、MPD デーモンがセットアップされ、実行されていることが前提で説明されています。診断を始める前に、次のコマンドを使用して、MPD デーモンがすべてのノードで実行されていることを確認してください。

```
mpdtrace
```

実行されているすべての MPD デーモンがリストされるか、またはエラーが表示されます。mpdtrace の出力リストに必要なノードが含まれていない場合は、次の操作を行います。

1. 次のコマンドを使用して MPD デーモンを再起動します。
   - 実行されているすべての MPD デーモンを終了します。

     ```
     $ mpdallexit
     ```
   - 各ノードで、すべてのデーモンが終了していることを確認します。

     ```
     $ rsh <nodename> ps -ael | grep python
     $ rsh <nodename> kill -9 <remaining python processes>
     ```
   - MPD デーモンを再起動します。適切な構成オプションとホストファイルが使用されていることを確認します。

     ```
     $ mpdboot [<options>]
     ```
   - すべての MPD デーモンが実行されていることを確認します。

     ```
     $ mpdtrace
     ```
2. mpdtrace コマンドによる出力結果で、実行されていない MPD デーモンがある場合は、次の操作を行います。

- 1 の説明にあるように、MPD デーモンを終了し、デバッグオプションと詳細オプションを、mpdboot コマンドに追加して再起動します。

  ```
  $ mpdboot -d -v [<options>]
  ```

  上記の出力結果に表示される rsh コマンドに注意してください。

  例:

  ```
  launch cmd= rsh  -n <nodename> '<installdir>/bin/mpd \
  -h <nodename> -p <port-number> --ncpus=<ncpus> -e -d'
  ```
- 出力結果に表示される rsh コマンドから最後までをコピーして貼り付けます。

  例:

  ```
  $ rsh -n <nodename> '<installdir>/bin/mpd \
  -h <nodename> -p <port-number> --ncpus=<ncpus> -e -d'
  ```

- 編集した rsh コマンドを実行します。出力結果を参考に、問題を診断し、修正します。例えば、最も一般的な問題には次のようなものがあります。
  - rsh コマンドで <nodename> にアクセスできない。
  - その他の rsh コマンドの失敗 (システム・セットアップの問題など)。
  - <installdir>/bin/mpd コマンドが見つからない、または実行できない。

3.3 テストプログラムのコンパイルと実行

テストプログラムをコンパイルして実行するには、次の操作を行います。

1. **(SDK のみ)** 製品リリースに含まれているテストプログラムを次のようにコンパイルします。

   ```
   $ cd <installdir>/test
   $ mpicc test.c
   ```

2. InfiniBand、Myrinet、またはその他の RDMA 対応のネットワーク・ハードウェアとソフトウェアを使用している場合は、すべてが適切に動作していることを確認します。
3. クラスター内のすべての利用可能な構成上でテストプログラムを実行します。

- 次のコマンドを使用して sock デバイスをテストします。

  ```
  $ mpiexec -n 2 -env I_MPI_DEBUG 2 –env I_MPI_DEVICE sock ./a.out
  ```

  各ランクに 1 行の出力と、sock デバイスが使用されていることを示すデバック情報が表示されます。
- 次のコマンドを使用して ssm デバイスをテストします。

  ```
  $ mpiexec -n 2 -env I_MPI_DEBUG 2 –env I_MPI_DEVICE ssm ./a.out
  ```

  各ランクに 1 行の出力と、ssm デバイスが使用されていることを示すデバック情報が表示されます。
- 次のコマンドを使用して、その他のファブリック・デバイスをテストします。

  ```
  $ mpiexec -n 2 -env I_MPI_DEBUG 2 -env I_MPI_DEVICE <device> ./a.out
  ```

  <device> は、shm、rdma、または rdssm です。

各 mpiexec コマンドで、各ランクごとに 1 行の出力結果と、使用されているデバイスを示すデバック情報が表示されます。デバイスが、I_MPI_DEVICE 設定と一致していることを確認してください。

注: インテル® MPI ライブラリー開発キットの <installdir>/test ディレクトリーには、test.c のほか、テストに使用できるプログラムが含まれています。

4 次のステップ

インテル® MPI ライブラリーに関する情報は、次のリソースも参考にしてください。

リリースノート: 製品の簡単な概要が含まれています。動作環境、テクニカルサポート、および既知の制限事項に関する更新情報は、リリースノートを参照してください。

詳細は、次の Web サイトを参照してください。

製品 Web サイト

インテル® MPI ライブラリーのサポート

インテル® クラスターツール製品

インテル® ソフトウェア開発製品