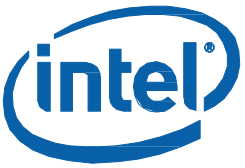

インテル® MPI ライブラリー for Windows*

ユーザーズガイド (5.1 Update 3)

目次

著作権と商標について

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も許諾するものではありません。

インテルは、明示されているか否かにかかわらず、いかなる保証もいたしません。ここにいう保証には、商品適格性、特定目的への適合性、知的財産権の非侵害性への保証、およびインテル製品の性能、取引、使用から生じるいかなる保証を含みますが、これらに限定されるものではありません。

本資料には、開発の設計段階にある製品についての情報が含まれています。この情報は予告なく変更されることがあります。最新の予測、スケジュール、仕様、ロードマップについては、インテルの担当者までお問い合わせください。

本資料で説明されている製品およびサービスには、不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。

MPEG-1、MPEG-2、MPEG-4、H.261、H.263、H.264、MP3、DV、VC-1、MJPEG、AC3、AAC、G.711、G.722、G.722.1、G.722.2、AMRWB、Extended AMRWB (AMRWB+)、G.167、G.168、G.169、G.723.1、G.726、G.728、G.729、G.729.1、GSM AMR、GSM FR は、ISO、IEC、ITU、ETSI、3GPP およびその他の機関によって制定されている国際規格です。これらの規格の実装、または規格が有効になっているプラットフォームの利用には、Intel Corporation を含む、さまざまな機関からのライセンスが必要になる場合があります。

性能に関するテストに使用されるソフトウェアとワークロードは、性能がインテル® マイクロプロセッサー用に最適化されていることがあります。SYSmark* や MobileMark* などの性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。製品の購入を検討される場合は、他の製品と組み合わせた場合の本製品の性能など、ほかの情報や性能テストも参考にして、パフォーマンスを総合的に評価することをお勧めします。

Intel、インテル、Intel ロゴは、アメリカ合衆国および / または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。

Microsoft、Windows、Windows ロゴは、米国 Microsoft Corporation の、米国およびその他の国における登録商標または商標です。

Java は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。

Bluetooth は商標であり、インテルは権利者から許諾を得て使用しています。

インテルは、Palm, Inc. の許諾を得て Palm OS ready マークを使用しています。

OpenCL および OpenCL ロゴは、Apple Inc. の商標であり、Khronos の使用許諾を受けて使用しています。

最適化に関する注意事項

インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。これには、インテル® ストリーミング SIMD 拡張命令 2、インテル® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサー用のものがあります。この注意事項で言及した命令セットの詳細については、該当する製品のユーザー・リファレンス・ガイドを参照してください。

注意事項の改訂 #20110804

1. はじめに

『インテル® MPI ライブラリー for Windows* ユーザーズガイド』は、いくつかの一般的なシナリオでインテル® MPI ライブラリーを使用する方法を説明しています。MPI アプリケーションのコンパイル、リンク、実行、およびデバッグに関する情報とともに、クラスター環境への統合に関する情報を提供します。

このユーザーズガイドでは、以下の情報を提供します。

ドキュメント構成

セクション	説明
第 1 章 – はじめに	このドキュメントについて
第 2 章 - 利用モデル	インテル® MPI ライブラリーを導入するための利用モデル
第 3 章 - インストールとライセンス	インストールの手順を説明し、ライセンスに関する情報を提示
第 4 章 - コンパイルとリンク	MPI アプリケーションのコンパイルとリンクの方法を説明
第 5 章 - アプリケーションの実行	アプリケーションを実行する手順を説明
第 6 章 - デバッグとテスト	デバッガーからアプリケーションを実行する方法を説明
第 7 章 - プロセス管理	プロセス管理に関する情報を提示
第 8 章 - ユーティリティーによるチューニング	インテル® MPI ライブラリー向けの最適な設定を見つけるため mpitune ユーティリティーを使用する方法を説明
第 9 章 - サポートされるジョブ・スケジューラー	ジョブ・スケジューラーとの統合方法を説明
第 10 章 - 一般的なクラスターに関する考察	MPI を利用する上で考慮すべき一般的なクラスターに関す情報を説明
第 11 章 – トラブルシューティング	一般的なトラブルシューティングの手順と例

1.1. インテル® MPI ライブラリーの紹介

インテル® MPI ライブラリーは、メッセージ・パッシング・インターフェイス 3.0 (MPI-3.0) 仕様を実装する、マルチファブリックをサポートするメッセージ・パッシング・ライブラリーです。インテル® プラットフォームに標準ライブラリーを提供します。

- 企業、事業部、部門および作業グループにおけるハイパフォーマンス・コンピューティング向けに業界で最高の性能を提供します。インテル® MPI ライブラリーは、インテル® アーキテクチャー・ベースのクラスター上でアプリケーションのパフォーマンスを向上することに注目します。
- 開発者の要求に応じて、MPI-3.0 の機能を導入できるようにします。

この製品には、次のコンポーネントが含まれています。

- インテル® MPI ライブラリー・ランタイム環境 (RTO) には、スケーラブルなプロセス管理システム (Hydra) やサポート・ユーティリティーなどプログラムの実行に必要なツール、ダイナミック (`.dll`) ライブラリー、ドキュメントなどが含まれています。
- インテル® MPI ライブラリー開発キット (SDK) には、すべてのランタイム環境コンポーネントに加え、`mpiicc` などのコンパイラー・コマンド、インクルード・ファイルとモジュール、デバッグ・ライブラリー、プログラム・データベース (`.pdb`) ファイル、およびテストコードなどが含まれます。

1.2. この資料の対象

このユーザーズガイドには、経験豊富な開発者がインテル® MPI ライブラリーを導入するのを支援する使い方の手順と例題による主要な機能の説明が含まれています。完全な説明は、『インテル® MPI ライブラリー for Windows* リファレンス・マニュアル』をご覧ください。

1.3. このドキュメントの表記について

このドキュメントでは、以下のフォント規則を使用しています。

表 1.3-1 このドキュメントの表記

`This type style`	コマンド、引数、オプション、ファイル名を示します。
`THIS_TYPE_STYLE`	環境変数を示します。
<この書式>	実際の値を挿入します。
`[ 項目 ]`	オプションの項目です。
`{ 項目 | 項目 }`	縦線で区切られた選択可能な項目です。
(SDK のみ)	ソフトウェア開発キット (SDK) 利用者向け

1.4. 関連情報

インテル® MPI ライブラリーに関する詳しい情報については、次のリソースをご覧ください。

- 『インテル® MPI ライブラリー・リリースノート』には、要件、技術サポート、および既知の問題に関する最新情報が含まれます。
- 『インテル® MPI ライブラリー・リファレンス・マニュアル』には、製品の機能、コマンド、オプションおよび環境変数に関する詳しい情報が含まれます。
- インテル® MPI ライブラリー for Windows* ナレッジベース (英語) では、トラブルシューティングのヒントや秘訣、互換性ノート、既知の問題、技術ノートが提供されます。

次のリソースもご覧ください。

インテル® MPI ライブラリー製品ウェブサイト

インテル® MPI ライブラリー製品サポート

インテル® クラスターツール製品ウェブサイト

インテル® ソフトウェア開発製品ウェブサイト

2. 使用モデル

インテル® MPI ライブラリーは、次の手順で導入します。

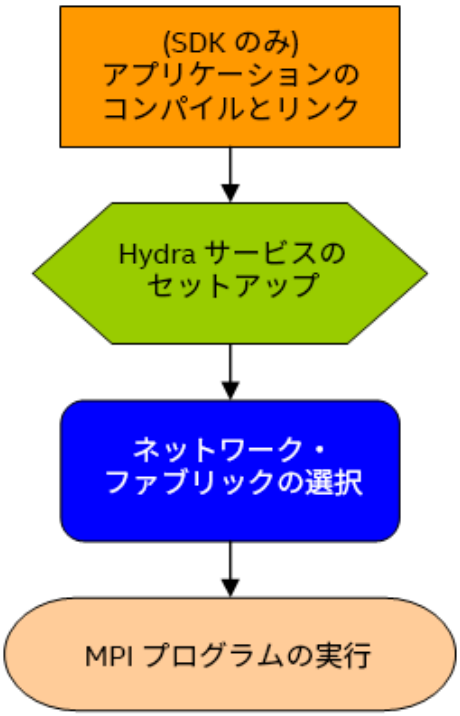

図 1: インテル® MPI ライブラリーを導入するための利用モデル

3. インストールとライセンス

このセクションでは、インテル® MPI ライブラリー・ランタイム環境 (RTO) とインテル® MPI ライブラリー・ソフトウェア開発キット (SDK)のインストール手順とライセンスに関する情報を提供します。

3.1. インテル® MPI ライブラリーのインストール

最新のインテル® MPI ライブラリー for Windows* をインストールする際に、以前のバージョンがインストールされている場合、それをアンインストールする必要はありません。

インテル® MPI ライブラリーをインストールするには、提供されたインストール・パッケージ `w_mpi_p_<version>.<package_num>.exe` (SDK) と `w_mpi- rt_p_<version>.<package_num>.exe` (RTO) をダブルクリックします。

インストールを開始する前に、インストール・ファイルを展開するディレクトリーを選択するように求められます。インストール終了後、ファイルはこのディレクトリーに残されます。インテル® 64 アーキテクチャーでは、デフォルトで `C:\Program Files (x86)\Intel\Download` が適用されます。

インストールを完了するには、インストール・ウィザードの指示に従います。インストール手順と要件の詳細については、『インテル® MPI ライブラリー for Windows* インストール・ガイド』をご覧ください。また、インストール・ガイドには、サイレント・インストールとクラスター・インストール関する情報も記載されています。

注意

Windows* でインテル® MPI ライブラリーをインストールするには、管理者権限が必要です。管理者権限のないアカウントでインストールを行った場合、Windows* 上で Active Directory* のセットアップを続行できません。Active Directory* の設定に関する詳細は、『インテル® MPI ライブラリー・リファレンス・マニュアル』をご覧ください。

3.2. インテル® MPI ライブラリー・ランタイム環境 (RTO) とインテル® MPI ライブラリー・ソフトウェア開発キット (SDK) のライセンス

インテル® MPI ライブラリーには、2 つのライセンスがあります。

- インテル® MPI ライブラリー・ランタイム環境 (RTO) ライセンス。このライセンスは、ランタイム・プロセス管理、ユーティリティーのサポート、共有ライブラリー (`.dll`) およびドキュメントを含む MPI プログラムの実行に必要なツールに適用されます。ライセンスには、インテル® MPI ライブラリー・ベースのアプリケーションを実行するために必要なすべてが含まれており、無料で永続的に利用できます。
- インテル® MPI ライブラリー・ソフトウェア開発キット (SDK) ライセンス。このライセンスには、コンパイルツールとともにランタイム環境のすべてのコンポーネントが含まれます: コンパイラー・コマンド (`mpiicc`、`mpicc` など)、ファイルとモジュール、スタティック・ライブラリー (`.lib`)、デバッグ・ライブラリー、トレース・ライブラリー、およびテスト用ソース。このライセンスは有料で、ソフトウェア利用許諾契約 (EULA) に記載されているようにいくつかの種類があります。

ライセンスの詳細は、EULA またはインテル® MPI ライブラリー製品ウェブサイトをご覧ください。

4. コンパイルとリンク

このセクションでは、インテル® MPI ライブラリーを使用したアプリケーションのコンパイルとリンクの手順を説明し、異なるデバッガーとコンパイラーのサポートに関する詳細を説明します。次の用語がこのセクションで使用されています。

- MPI プログラムのコンパイル
- デバッグ情報の追加
- その他のコンパイラー・サポート

4.1. MPI プログラムのコンパイル

(SDK のみ)

インテル® MPI ライブラリーを使用して MPI プログラムをコンパイル/リンクするには、次の手順に従ってください。

1. Microsoft* Visual Studio* で、WinXX コンソール・プロジェクトを作成します。
2. X64 ソリューション・プラットフォームを選択します。
3. インクルード・パスに `<installdir>\intel64\include` を追加します。
4. ライブラリー・パスに `<installdir>\intel64\lib\<configuration>` を追加します。`<configuration>` に次を設定します。
   - `Debug`: シングルスレッド版のデバッグ向けライブラリー。
   - `Release`: シングルスレッド版の最適化されたライブラリー。
   - `Debug_mt`: マルチスレッド版のデバッグ向けライブラリー。
   - `Release_mt`: マルチスレッド版の最適化されたライブラリー。
5. ターゲットリンク・コマンドに適切なインテル® MPI ライブラリーを追加します。
   - C アプリケーションには `impi.lib` を追加します。
   - C++ アプリケーションには、`impi.lib` と `impicxx.lib` (`Release`) または、`impid.lib` と `impicxxd.lib` (`Debug`) を追加します。
6. プログラムをコンパイルおよびリンクします。
7. アプリケーションと関連するダイナミック・ライブラリーを共有ストレージに配置するか、それらをすべてのノードにコピーします。
8. `mpiexec.exe` コマンドを使用してアプリケーションを実行します。

4.2. デバッグ情報の追加

アプリケーションをデバッグする場合、`/Zi` オプションを追加します。これにより、バイナリーにデバッグ情報が追加されます。アプリケーションのデバッグには、任意のデバッガーを利用できます。

```
> mpiicc /Zi test.c -o testc.exe
```

4.3. その他のコンパイラー・サポート

インテル® MPI ライブラリーは、最新のインテル® コンパイラーに加えて Microsoft* Visual Studio* コンパイラーをサポートしています。サポートされる Micsoroft Visual Studio* のバージョンについては、『インテル® MPI ライブラリー・リリースノート』をご覧ください。サードパーティーのライブラリーとのリンク方法については、Microsoft* Visual Studio* のドキュメントを参照してください。

5. アプリケーションの実行

アプリケーションのコンパイルとリンクが完了したら、インテル® MPI アプリケーションを実行する準備ができました。ここでは、アプリケーションを実行する手順を説明します。

5.1. インテル® MPI プログラムの実行

インテル® MPI ライブラリーとリンクしたアプリケーションを実行するには、mpiexec コマンドを使用します。

```
> mpiexec.exe -n <プロセス数> myprog.exe
```

wmpiexec ユーティリティーは、mpiexec.exe への GUI ラッパーです。このコマンドに関しては、『インテル® MPI ライブラリー・リファレンス・マニュアル』をご覧ください。

ローカルノード上のプロセス数を指定するには、mpiexec -n オプションを使用します。

ホスト名とプロセス数を指定するには、-hosts オプションを使用します。

```
> mpiexec.exe -hosts 2 host1 2 host2 2 myprog.exe
```

デフォルト以外のネットワーク・ファブリックを使用する場合、-genv オプションを使用して I_MPI_FABRICS 変数を設定します。

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、次のコマンド行を入力します。

```
> mpiexec.exe -genv I_MPI_FABRICS shm -n <プロセス数> myprog.exe
```

-configfile オプションを使用して、構成オプションが定義されたファイルを指定することで、複数の構成を容易に切り替えることもできます。

```
> mpiexec.exe -configfile config_file
```

この例では、次のオプションがファイルに含まれます。

```
-genv I_MPI_FABRICS shm:dapl
-host host1 -n 1 myprog.exe
-host host2 -n 1 myprog.exe
```

これは、異なるホスト上で高速な dapl ファブリックを使用して 2 つのバージョンの実行形式ファイル (myprog.exe) を実行することを可能にします。

RDMA ネットワークが利用可能な場合、次のコマンドを使用します。

```
> mpiexec.exe -hosts 2 host1 1 host2 1 -genv I_MPI_FABRICS dapl myprog.exe
```

詳細は、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たアプリケーションが正しく実行できたら、そのアプリケーションは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するほかのクラスターへ移行できます。問題が生じた場合、「デバッグとテスト」で解決策を探してください。

5.2. マルチスレッド・アプリケーション

新しいスレッドを生成するアプリケーションを実行するには、インテル® MPI ライブラリーのスレッドセーフ版をリンクする必要があります。しかし、コンパイル時にオプションを追加する必要はありません。mpiicc などのコンパイル・スクリプトは、デフォルトでスレッドセーフ版のライブラリーをリンクします。

OpenMP* アプリケーションを実行し、ドメイン内でスレッドを固定するには、KMP_AFFINITY 環境変数が、対応する OpenMP* の機能を使用するように設定されている必要があります。

アプリケーションを実行するには次のコマンドを入力します。

```
> mpiexec.exe –genv OMP_NUM_THREADS 4 -n <プロセス数> .\myprog
```

OpenMP* との相互利用に関する詳しい情報は、『インテル® MPI ライブラリーfor Windows* リファレンス・マニュアル』をご覧ください。

マルチスレッド版のデバッグ向けライブラリーに切り替えるには、次のコマンドを使用します。

```
> call mpivars.bat debut_mt
```

5.3. ファブリックの選択

インテル® MPI ライブラリーは、デフォルトで I_MPI_FABRICS_LIST に指定されるファブリックのリストをベースにネットワーク・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、-genv オプションを使用して I_MPI_FABRICS 環境変数に割り当てます。また、set コマンドを使用して値を割り当てることもできます。

指定されたファブリックが利用できない場合、インテル® MPI ライブラリーは、I_MPI_FABRICS_LIST に指定されるリストから次に利用可能なファブリックを選択します。

このフォールバックの振る舞いは、I_MPI_FALLBACK 変数を使用して無効にできます。

```
-genv I_MPI_FALLBACK 0
```

デフォルトでは、フォールバックは有効です。I_MPI_FABRICS が設定されると、フォールバックは無効になります。

5.3.1. TCP ソケット接続

クラスターで利用可能なイーサーネット接続を使用して TCP ソケットを介して 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。プログラムは、ノード内で共有メモリーを使用しません。

```
> mpiexec.exe -genv I_MPI_FABRICS tcp -n <プロセス数> myprog.exe
```

5.3.2. 共有メモリー

共有メモリーファブリック (shm) のみを介して 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.exe -genv I_MPI_FABRICS shm -n <プロセス数> myprog.exe
```

5.3.3. 共有メモリーと DAPL* 接続

ノード内の通信に共有メモリーを使用し、ノード間の通信に Direct Access Programming Library* (DAPL*) を使用するには、次のコマンドを使用します。

```
> mpiexec.exe -genv I_MPI_FABRICS shm:dapl -n <プロセス数> myprog.exe
```

これは、ファブリック・オプションが指定されていない場合のデフォルトです。

ファブリック制御に関する詳しい情報は、『インテル® MPI ライブラリーfor Windows* リファレンス・マニュアル』をご覧ください。

インテル® MPI ライブラリーを使用して任意のファブリックでアプリケーションが正しく実行できたら、そのアプリケーションは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するほかのクラスターへ移行できます。問題が生じた場合、「デバッグとテスト」で解決策を探してください。

5.3.4. I_MPI_FABRICS

このトピックは、『インテル® MPI ライブラリー for Windows* リファレンス・マニュアル』からの抜粋であり、I_MPI_FABRICS 環境変数について詳しく説明します。

通信に使用する特定のファブリックを選択します。

構文

`I_MPI_FABRICS=<fabric>|<intra-node fabric>:<inter-nodes fabric>`

ここで各引数には、以下を設定できます。

- `<fabric> := {shm, dapl, tcp}`
- `<intra-node fabric> := {shm, dapl, tcp}`
- `<inter-nodes fabric> := {dapl, tcp}`

引数

引数	定義
<fabric>	ネットワーク・ファブリックを定義
`shm`	共有メモリー
`dapl`	InfiniBand*、iWarp*、Dolphin* や XPMEM* (DAPL* を介して) などの DAPL ケーブル・ネットワーク・ファブリック
`tcp`	イーサーネットや InfiniBand* (IPoIB* を介して) のような TCP/IP ケーブル・ネットワーク・ファブリック

例えば、winOFED* InfiniBand* デバイスを選択するには、次のコマンドを使用します。

`>mpiexec.exe -n <プロセス数> -env I_MPI_FABRICS shm:dapl <実行形式>`

これらのデバイスは、*<provider>* が指定されていない場合、`dat.conf` ファイルの最初の DAPL* プロバイダーが使用されます。`shm` ファブリックは、インテル製マイクロプロセッサーおよび互換マイクロプロセッサーで利用可能ですが、インテル製マイクロプロセッサーにおいてより多くの最適化が行われる場合があります。

注意

選択されているファブリックが利用可能であることを確認してください。例えば、すべてのプロセスが共有メモリーを介してのみ通信可能な場合、`shm` を使用します。すべてのプロセスが単一の DAPL プロバイダーを介してのみ通信可能な場合、`dapl` を使用します。`dat.dll` ライブラリーへのパスが、%PATH% に設定されていることを確認してください。または、`I_MPI_DAT_LIBRARY` 環境変数に `dat.dll` ライブラリーへの完全なパスを設定するため、`mpiexec.exe` に `-genv` オプションを使用します。

6. デバッグとテスト

インテル® MPI ライブラリーは、Microsoft* Visual Studio* デバッガーをサポートします。サポートされる Micsoroft* Visual Studio* のバージョンについては、『インテル® MPI ライブラリー・リリースノート』をご覧ください。サードパーティーのライブラリーとのリンク方法については、Microsoft* Visual Studio* のドキュメントを参照してください。

また、インテル® MPI ライブラリーは、Microsoft* MPI Windows* デバッガー拡張 (MS-MPI v6 WinDbg 拡張以降) で利用可能なメッセージ・キュー・デバッガー機能をサポートしています。この機能は、Windows* プラットフォーム上の任意の MPI ツールやデバッガーが、標準化されたメッセージ・キュー・インターフェイスを使用してインテル® MPI ライブラリーと相互運用することを可能にします。

このセクションでは、デバッグツールの使い方とインテル® MPI ライブラリーのデバッグ拡張機能について説明します。

6.1. ログの取得

時には、アプリケーションのデバッグよりも、ログを利用することが有効なことがあります。実行中のアプリケーションのログを取得するには、いくつかの方法があります。

6.1.1. デバッグ情報の取得

環境変数 `I_MPI_DEBUG` は、実行中の MPI アプリケーションの情報を取得する非常に便利な機能を提供します。この変数には、`0` (デフォルト) から `1000` の値を設定できます。値が大きくなるほど詳しいデバッグ情報を取得できます。

```
> mpiexec.exe -genv I_MPI_DEBUG 5 -n 8 ./my_application
```

注意

`I_MPI_DEBUG` に大きな値を設定すると、多くの情報を取得できますが、アプリケーションの実行パフォーマンスを大幅に低下させます。始めに一般的なエラーを検出するには、`I_MPI_DEBUG=5` で開始することを推奨します。`I_MPI_DEBUG` に関する詳しい情報は、『インテル® MPI ライブラリー for Windows* リファレンス・マニュアル』をご覧ください。

6.1.2. アプリケーションのトレース

`-t` または `-trace` オプションを使用してインテル® Trace Collector ライブラリーとのリンクを行います。これは、`mpiicc` やほかのコンパイラー・スクリプトに `-profile=vt` オプションを指定するのと同じ効果があります。

```
> mpiicc -trace test.c -o testc
```

このオプションを使用するには、次の操作を行う必要があります。

- インテル® Trace Analyzer & Collector を最初にインストールします。このツールは、インテル® Parallel Studio XE Cluster Edition スイートの一部としてのみ提供されます。
- `VT_ROOT` 環境変数に、インテル® Trace Collector のインストール先のパスが含まれます。ほかのプロファイル・ライブラリーを指定するには、`I_MPI_TRACE_PROFILE` 環境変数に <プロファイル名> を設定します。例えば、`I_MPI_TRACE_PROFILE` に `vtfs` を設定すると、フェイルセーフ版のインテル® Trace Collector とリンクを行います。

6.1.3. 正当性のチェック

`-check_mpi` オプションを使用してインテル® Trace Collector の正当性チェック・ライブラリーとのリンクを行います。これは、`mpiicc` やほかのコンパイラー・スクリプトに `-profile=vtmc` オプションを指定するのと同じ効果があります。

```
> mpiicc -profile=vtmc test.c -o testc
```

もしくは

```
> mpiicc -check_mpi test.c -o testc
```

このオプションを使用するには、次の操作を行う必要があります。

- インテル® Trace Analyzer & Collector を最初にインストールします。このツールは、インテル® Parallel Studio XE Cluster Edition スイートの一部としてのみ提供されます。
- VT_ROOT 環境変数に、インテル® Trace Collector のインストール先のパスが含まれます。ほかのプロファイル・ライブラリーを指定するには、`I_MPI_CHECK_PROFILE` 環境変数に <プロファイル名> を設定します。

インテル® Trace Analyzer & Collector の詳しい情報は、製品に含まれるドキュメントをご覧ください。

6.1.4. 統計収集

アプリケーションが使用する MPI 関数の統計情報を収集するには、`I_MPI_STATS` 環境変数に `1` から `10` の値を設定します。この環境変数は、収集する統計情報の量を制御し、ログをファイルに出力します。デフォルトでは、統計情報は収取されません。

統計収集に関する詳しい情報は、『インテル® MPI ライブラリーfor Windows* リファレンス・マニュアル』をご覧ください。

6.2. インストールのテスト

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、テストプログラムのコンパイルと実行に加え、次の手順で一般的なテストを行います。

次の手順でインストールをテストします。

1. [コンピューターの管理] > [サービス] コンソールで、Hydra サービス (Intel® MPI Library Hydra Process Manager) が開始されていることを確認します。これは、インテル® MPI プロセス管理の初期化に呼び出されます。
2. インテル® 64 アーキテクチャーでは、PATH に `<installdir>\intel64\bin` が含まれていることを確認します。

   ```
   > echo %PATH%
   ```

   各ノードでパスが正しいことを確認してください。
3. **(SDK のみ)** インテル® コンパイラーを使用している場合、`PATH` と `LIB` 環境変数に適切なディレクトリーが設定されていることを確認してください。

   ```
   > mpiexec.exe -hosts 2 host1 1 host2 1 a.bat
   ```

   `a.bat` には以下が含まれます。

   ```
   echo %PATH%
   ```

各ノードでパスが正しいことを確認してください。正しくない場合、`compilervars.bat` スクリプトを呼び出します。例えば、インテル® 64 アーキテクチャー向けのインテル® C++ コンパイラー for Windows* を 64 ビット環境で使用する場合、Windows* の [プログラム メニュー] から [Intel(R) Parallel Studio XE 2016] > [コマンドプロンプト] > [インテル(R) 64 Visual Studio* 2012 mode] を選択して、コマンドラインで次のスクリプトを実行します。

```
%ProgramFiles%\Intel\Composer XE\bin\iclvars.bat
```

6.3. テストプログラムのコンパイルと実行

インストール・ディレクトリー <installdir>\samples_2016\en\mpi には、テスト用のプログラムが含まれます。テストプログラムをコンパイルするには、次の手順に従ってください。

1. **(SDK のみ)**「コンパイルとリンク」で示される手順でテストプログラムをコンパイルします。
2. InfiniBand* またはその他の RDMA-ケーブル・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、すべてが機能していることを確認してください。
3. クラスターのすべての利用可能な構成でテストプログラムを実行します。
4. TCP/IP ネットワーク・ファブリックをテストするには、次のコマンドを使用します。

   ```
   > mpiexec.exe -n 2 -env I_MPI_DEBUG 2 -env I_MPI_FABRICS tcp .\myprog.exe
   ```

5. DAPL ネットワーク・ファブリックをテストするには、次のコマンドを使用します。

   ```
   > mpiexec.exe -n 2 -env I_MPI_DEBUG 2 -env I_MPI_FABRICS
   shm:dapl .\myprog.exe
   ```

mpiexec コマンドを使用すると、ランクごとに 1 行のメッセージと使用しているファブリックのデバッグ情報が表示されるはずです。デバイスは、I_MPI_FABRICS の設定と一致している必要があります。

7. プロセス管理

ここでは、インテル® MPI ライブラリーに含まれるプロセス管理について説明します。

- スケーラブルなプロセス管理システム (Hydra)
- MPI プロセスの配置を制御する

7.1. スケーラブルなプロセス管理システム (Hydra)

Hydra は、シンプルでスケーラブルなプロセス管理です。Hydra は、既存のリソース管理がプロセスを実行する場所を決め、各ホストのプロキシーを使用してターゲット間に分散するのをチェックします。プロキシーは、プロセスの起動、クリーンアップ、I/O 転送、信号転送、およびほかのタスクで使用されます。

`mpiexec.exe` を使用して Hydra を起動できます。スケーラブル・プロセス管理システム (Hydra) コマンドの詳細は、『インテル® MPI ライブラリー・リファレンス・マニュアル』をご覧ください。

注意

マルチプロセス・デーモン (SPMD) は、インテル® MPI ライブラリー 5.0 リリースでは使用されなくなりました。並列ジョブを開始するには、スケーラブル・プロセス管理システム (Hydra) を使用します。

7.1.1. Hydra サービスの設定

Microsoft* Visual Studio* のコンパイラーでコンパイルされたプログラムを実行するには、最初に HYDRA サービスを設定します。

注意

Hydra サービスを開始するには管理者権限が必要です。一度 Hydra サービスを開始すると、すべてのユーザーは `mpiexec.exe` でプロセスを起動できます。

Hydra サービスは、インテル® MPI ライブラリーのインストール中に開始されます。インストール時に Hydra サービスの開始をキャンセルすることもできます。

インテル® MPI ライブラリーがインストールされると、Hydra サービスを手動で開始、停止、または削除できます。これらの操作は、`<installdir>\intel64\bin` に配置される `hydra_service.exe` で行います。

`<installdir>\intel64\bin`

HYDRA サービスを設定するには次の操作を行います。

1. クラスターの各ノードで次のコマンドを実行して古い Hydra サービスを削除します。

   ```
   hydra_service.exe -remove
   ```

2. クラスターの各ノードで次のコマンドを実行して手動で Hydra サービスをインストールします。

   ```
   hydra_service.exe -install
   ```

7.2. MPI プロセスの配置を制御する

`mpiexec` コマンドは、プロセスのランクがクラスターのノードにどのように割り当てられるかを制御します。デフォルトでは、`mpiexec` コマンドはグループ・ラウンドロビン割り当てにより、ノードのすべてのプロセッサー・ランクに連続する MPI プロセスを投入します。この配置アルゴリズムは、対称型マルチプロセッサー (SMP) ノードを持つクラスターでは、アプリケーションにとって最良の選択ではないかもしれません。

各ノードに割り当てられる隣接するランクのペアのジオメトリーは、<ランク数> = 4 および <ノード数> = 2 と仮定します (例えば、2 ウェイ SMP ノードの場合)。クラスターノードを参照するには、次のコマンドを入力します。

```
> type machines.Windows
```

結果は以下のようになります。

```
clusternode1
clusternode2
```

アプリケーションの 4 つのプロセスを 2 ウェイ SMP クラスターに均等に配置するには、次のコマンドを入力します。

```
> mpiexec.exe -n 2 -host clusternode1 .\myprog.exe :-n 2 –host
clusternode2 .\myprog.exe
```

myprog.exe の実行結果は、次のように出力されます。

```
Hello world: rank 0 of 4 running on clusternode1
Hello world: rank 1 of 4 running on clusternode1
Hello world: rank 2 of 4 running on clusternode2
Hello world: rank 3 of 4 running on clusternode2
```

一般的に、クラスターが i 個のノードを持ち、各ノードが j-ウェイの SMP システムである場合、クラスター内の i*j 個のプロセッサーで i*j 個のプロセスを分配するには、次の mpiexec コマンドを使用します。

```
> mpiexec.exe -n j -host <nodename-1> .\myprog.exe :\
-n j -host <nodename-2> .\myprog.exe :\
-n j -host <nodename-3> .\myprog.exe :\
...
-n j -host <nodename-i> .\myprog.exe
```

注意

ノード名 <nodename-1> から <Nodename-i> には、実際のクラスターシステムのホスト名を入力する必要があります。

関連情報

ローカルのオプションに関する詳細は、『インテル® MPI ライブラリー・リファレンス・マニュアル』をご覧ください。

MPI プロセスの配置の制御に関する詳しい情報は、オンラインの「インテル® MPI ライブラリーのプロセス配置制御」(英語) もご覧ください。

8. mpitune ユーティリティーによるチューニング

この章では、インテル® MPI ライブラリー向けの最適な設定を見つけるため mpitune ユーティリティーを使用する方法を説明します。

8.1. クラスター固有のチューニング

インテル® MPI ライブラリーには 100 個以上のパラメーターがあります。デフォルトは、一般的な利用向けの設定が行われ、ほとんどのクラスターとアプリケーションで優れたパフォーマンスを提供します。しかし、より高いパフォーマンスを求める場合、mpitune ユーティリティーを利用できます。このユーティリティーは、ベンチマーク・プログラムとしてインテル® MPI Benchmarks (IMB) を使用し、異なるパラメーターでテストを数回実行してクラスターシステムに最適な設定を探します。mpitune ユーティリティーは次のコマンドで起動します。

```
> mpitune
```

その後、チューニングされた設定を有効にするには、アプリケーション起動時に -tune オプションを指定します。

```
> mpiexec.exe -tune -perhost 8 –n 64 ./myprog.exe
```

最良の結果を得るには、チューニングされたパラメーターのデフォルトの格納先である <installdir>\<arch>\etc への書き込み権限を持つアカウントで mpitune を実行します。このディレクトリーへの書き込み権限が無い場合、カレント・ディレクトリーに新しい設定ファイルが保存されます。

デフォルトで、mpitune はベンチマークとしてインテル® MPI Benchmarks (IMB) を使用します。また、次のようなコマンドでベンチマークを置き換えることができます。

```
> mpitune -test \”your_benchmark –param1 –param2\”
```

新しい設定を適用するには、クラスター固有のチューニングをご覧ください。

インテル® MPI Benchmarks の実行可能ファイルは、デフォルトで非インテル互換プロセッサーよりもインテル® マイクロプロセッサーに最適化されています。そのため、インテル® マイクロプロセッサーと非インテル互換プロセッサーでは、チューニングの設定が異なることがあります。

8.2. アプリケーション固有のチューニング

アプリケーション固有の最適化設定を見つけるには、mpitune ユーティリティーを使用します。

```
> mpitune --application \”your_app\” –-output-file yourapp.conf
```

“your_app” には、実際に起動するアプリケーションを指定します。次に例を示します。

```
> mpitune –-application \".\myprog.exe\" –-output-file $PWD\myprog.conf
```

チューニングされた設定は、myprog.conf ファイルに保存されます。それを適用するには、mpiexec を次のように起動します。

```
> mpiexec.exe -tune $PWD\myprog.conf -perhost 8 –n 64 .\myprog.exe
```

注意

“myprog.exe” には、実行可能ファイルだけでなく異なるプロセスで開始するスクリプトを指定することもできます。

注意

スクリプトでは、I_MPI_* 環境変数を変更してはいけません。

8.3. 時間制限の設定

チューニングの過程には時間がかかります。各クラスターとアプリケーションの設定にはさまざまな要因があるため、チューニングの時間は予測できないことがあります。

チューニングの時間を制限するには、-time-limit オプションを使用します。例えば、チューニングを 8 時間 (480 分) に制限するには、次のコマンドを実行します。

```
> mpitune --time-limit 480 (時間は分単位で指定します)
```

8.4. ファブリック・リストの設定

テストするファブリックを定義するには、-fabrics-list オプションを使用します。

```
> mpitune --fabrics-list shm,dapl
```

利用可能なファブリックには以下があります: shm:dapl, shm:tcp, shm, tcp

8.5. プロセス数の範囲を設定

1 つのノード上で実行するプロセスを制限するには、perhost-range min:max オプションを使用します。例えば、次のコマンドは各ノード上の MPI ランク数を 4 から 8 に設定します。

```
> mpitune --perhost-range 4:8
```

8.6. ホストを使用する制限を設定

チューニングを実行するノードを制限するには、host-range min:max オプションを使用します。例えば、次のコマンドは 8 から 16 ノードでのみ実行するように制限します:

```
> mpitune --host-range 8:16
```

8.7. 最後に保存したセッションから mpitune をリストア

mpitune を実行中に予測しないイベントが発生することがあります。このケースでは、mpituner_session_<session-id>.mts ファイルに保存された途中結果を使用します。最後に保存したセッションから mpitune を開始します。

```
> mpitune --session-file .\mpituner_session_<session-id>.mts
```

<session-id> には、チューニングを開始するタイムスタンプを指定します。

8.8. 手動でアプリケーションをチューニング

一連の I_MPI_ADJUST_* 環境変数を使用して、インテル® MPI ライブラリーの集合操作を手動でチューニングすることができます。メッセージサイズの範囲を設定し、異なるアルゴリズムを選択するとアプリケーションのパフォーマンスを向上することができます。I_MPI_ADJUST に関する詳しい情報は、『インテル® MPI ライブラリーfor Windows* リファレンス・マニュアル』をご覧ください。

9. サポートされるジョブ・スケジューラー

インテル® MPI ライブラリーは、HPC 市場で一般的に利用されているジョブ・スケジューラーの大部分をサポートしています。Windows* では、次のジョブ・スケジューラーがサポートされます。

- Microsoft* HPC Pack*
- Altair* PBS Pro*

9.1. Microsoft* HPC Pack*

インテル® MPI ライブラリーのジョブ起動コマンド `mpiexec` は、MPI アプリケーションを実行するため Microsoft* HPC ジョブ・スケジューラーから呼び出すことができます。この場合、`mpiexec` コマンドは、ホストのリスト、プロセス数、ジョブに割り当てられた作業ディレクトリーを自動的に継承します。

次のコマンドを使用して MPI ジョブを送信します。

```
job submit /numprocessors:4 /stdout:test.out mpiexec -delegate test.exe
```

`mpiexec` とダイナミック・ライブラリーが、`PATH` からアクセスできることを確認してください。インテル® MPI ライブラリーの環境変数は、インストール中に登録できます。

9.2. Altair* PBS Pro*

インテル® MPI ライブラリーのジョブ起動コマンド `mpiexec` は、MPI アプリケーションを実行するため PBS Pro* ジョブ・スケジューラーから呼び出すことができます。この場合、`mpiexec` コマンドは、ユーザーが手動で指定していなければ、ホストのリスト、ジョブに割り当てられたプロセス数を自動的に継承します。`mpiexec` は、プロセス数をカウントし `machinefile` として使用するため `%PBS_NODEFILE%` 環境変数を読み込みます。

例:

ジョブスクリプトの内容:

```
REM PBS -l nodes=4:ppn=2
REM PBS -l walltime=1:00:00
cd %PBS_O_WORKDIR%
mpiexec test.exe
```

次のコマンドを使用してジョブを送信します。

```
qsub -C "REM PBS" job
```

`mpiexec` は、このジョブを 4 つのノードのそれぞれで 2 つのプロセスを実行します。

10. 一般的なクラスターに関する考察

この章では、MPI を利用する上で考慮すべき一般的なクラスターに関す情報を説明します。

- 使用するノードの定義
- ヘテロジニアス・システムとジョブ
- ユーザー認証

10.1. 使用するノードの定義

インテル® MPI ライブラリーは、デフォルトで `mpd.hosts` ファイルを検索します。このファイルには、アプリケーションが利用できるクラスター上のすべてのノードが含まれている必要があります。`mpd.hosts` ファイルの形式は、ノード名のリストを 1 行に 1 つ定義します。空白行と # に続く行は無視されます。

`–f` オプションを使用して、このファイルへのパスを指定できます。

ジョブ・スケジューラーの下で実行している場合、ホストはスケジューラーで決定されるため `–f` オプションは必要ありません。

10.2. ヘテロジニアス・システムとジョブ

すべてのクラスターは、ホモジニアス (同種) ではありません。すべてのジョブは、ホモジニアス (同種) ではありません。インテル® MPI ライブラリーは、複数のコマンドと引数を 1 つのコマンドで実行することができます。これには 2 つの方法があります。

最も簡単な方法は、設定ファイルを作成して `-configfile` オプションを使用することです。設定ファイルは、`mpiexec` への引数を行ごとに 1 つのグループとして記述します。

```
-n 1 –host node1 ./io <io_args>
-n 4 –host node2 ./compute <compute_args_1>
-n 4 –host node3 ./compute <compute_args_2>
```

また、オプションのセットをコマンド行でグループごとに ':' で区切って渡すことができます。

```
> mpiexec.exe –n 1 –host node1 ./io <io_args> :-n 4 –host node2 \
./compute <Compute_args_1> :-n 4 –host node3 ./compute <compute_args_2>
```

プロセスが起動されると、作業ディレクトリーは、ジョブが起動されたマシンの作業ディレクトリーに設定されます。変更するには、`-wdir <パス>` オプションを使用します。

1 つのプロセスグループのみに環境変数を適用するには、`–env <変数> <値>` オプションを使用します。すべてのプロセスグループに環境変数を適用するには、`–genv` を使用します。デフォルトでは、すべての環境変数が起動時に実行環境から継承されます。

10.3. ユーザー認証

インテル® MPI ライブラリーは、Windows* クラスター全体でユーザー認証を行う 2 つの方法をサポートしています。

- パスワードベースの認証
- ドメインベースの認証

パスワードベースの認証は、既存のユーザー・アカウントとパスワードを使用してリモートノードへアクセスする最も一般的な方法です。インテル® MPI ライブラリーは、`mpiexec – register` または `wmpiregister` ツールを使用して、レジストリーへユーザーのログイン情報を暗号化することができます。これは、ユーザーがアプリケーションを最初に実行するときに一度だけ行います。

ドメインベースの認証では、ユーザーがリモートマシンへのアクセスを許可するため Microsoft* Security Service Provided Interface* (SSPI*) を使用します。これはクラスター管理者によって定義されるドメインポリシーに基づいて管理されます。マシン上にユーザー情報 (ユーザー名とパスワード) を保存する必要はありません。

Windows* クラスターでドメインベースの任法方式を有効にするには、管理者が計算ノードとクラスターユーザーを Active Directory* に委任するよう設定する必要があります。

Active Directory* の設定に関する詳細は、『インテル® MPI ライブラリーfor Windows* リファレンス・マニュアル』をご覧ください。

11. トラブルシューティング

このセクションでは、次のようなトラブルシューティングに関する情報を提供します。

- インテル® MPI ライブラリーの一般的なトラブルシューティングの手順
- 障害が発生した際の典型的な MPI の失敗に関する出力メッセージと対処法
- 潜在的な原因と解決策に関する提言

11.1. 一般的なトラブルシューティングの手順

インテル® MPI ライブラリーの使用中にエラーや障害が発生した際の、一般的なトラブルシューティングの手順を以下に示します。

1. システム要件のセクションとインテル® MPI ライブラリーのリリースにある既知の問題をチェックします。
2. ホストが利用できるかチェックします。`mpiexec` を使用して、ホスト・プラットフォーム上で簡単な非 MPI アプリケーション (例えば、`hostname` ユーティリティーなど) を実行します。

   例:

   ```
   > mpiexec.exe -ppn 1 -n 2 -hosts node01,node02 hostname
   node01
   node02
   ```

   これは、環境の問題や接続性の問題 (ホストに接続できないなど) を明らかにするのに役立ちます。
3. 環境変数 `I_MPI_DEBUG=6` および (または) `I_MPI_HYDRA_DEBUG=on` に設定し、デバッグ情報を有効にして MPI アプリケーションを実行します。より詳しいデバッグ情報を取得するには、設定する整数値の増やしてください。これは、問題のあるコンポーネントを特定するのに役立ちます。
4. 製品ページからインテル® MPI ライブラリーの最新バージョンをダウンロードしてインストールし、問題が解決しないか確認してください。
5. 問題が解決しない場合、インテル® プレミアサポート (英語) から問題を報告してください。

11.2. MPI の失敗の例

このセクションでは、エラーの説明、エラーメッセージ、および関連する推奨事項などを含む典型的な MPI のエラーの例を紹介します。

11.2.1. 例 1

症状 / エラーメッセージ

```
Error connecting to the Service
[mpiexec@node01] ..\hydra\utils\sock\sock.c (270): unable to connect from "node01"
to "node02" (No error)
read from stdin failed, error 9.
[mpiexec@node01] ..\hydra\tools\demux\demux_select.c (78): select error (No such
file or directory)
[mpiexec@node01] ..\hydra\pm\pmiserv\pmiserv_pmci.c (501): error waiting for event
[mpiexec@node01] ..\hydra\ui\mpich\mpiexec.c (1063): process manager error waiting
for completion
```

原因

ノードのいずれかで Hydra サービスが開始されていません。

ソリューション

すべてのノード上で Hydra サービスの状態をチェックします。

11.2.2. 例 2

症状 / エラーメッセージ

```
Error connecting to the Service
[mpiexec@node01] ..\hydra\utils\sock\sock.c (224): unable to get host address for
node02 (11001)
read from stdin failed, error 9.
[mpiexec@node01] ..\hydra\tools\demux\demux_select.c (78): select error (No such
file or directory)
[mpiexec@node01] ..\hydra\pm\pmiserv\pmiserv_pmci.c (501): error waiting for event
[mpiexec@node01] ..\hydra\ui\mpich\mpiexec.c (1063): process manager error waiting
for completion
```

原因

いずれかのノードにアクセスできません。

解決方法

すべてのノードが操作可能であるかチェックしてください。

11.2.3. 例 3

症状 / エラーメッセージ

```
pmi_proxy not found on node02.Set Intel MPI environment variables read from stdin
failed, error 9.
[mpiexec@node01] ..\hydra\tools\demux\demux_select.c (78): select error (No such
file or directory)
[mpiexec@node01] ..\hydra\pm\pmiserv\pmiserv_pmci.c (501): error waiting for event
[mpiexec@node01] ..\hydra\ui\mpich\mpiexec.c (1063): process manager error waiting
for completion
```

原因

インテル® MPI ライブラリーのランタイム・スクリプトが利用できません。共有ストレージにアクセスできない可能性があります。

解決方法

すべてのノード上で MPI が利用できるかチェックします。おそらく、ネットワーク共有ドライブに関連するいくつかの問題があります。

11.2.4. 例 4

症状 / エラーメッセージ

```
[-1:5612]ERROR:execvp error on file: <mpi_application_path>, The filename,
directory name, or volume label syntax is incorrect.
```

原因

いずれかのノードに MPI アプリケーションがありません。

解決方法

すべてのノードに MPI アプリケーションがあるかチェックしてください。おそらく、ネットワーク共有ドライブに関連するいくつかの問題があります。

11.2.5. 例 5

症状 / エラーメッセージ

```
[3:node01] unexpected disconnect completion event from [8:node02]
```

原因

MPI プロセスの 1 つが終了しました。MPI アプリケーションは `dapl` ファブリックを介して実行されました。

解決方法

MPI プロセスが終了した原因を特定してください。

11.2.6. 例 6

症状 / エラーメッセージ

```
Attempting to use an MPI routine before initializing MPI
```

原因

異なるインテル® MPI ライブラリーの構成と依存関係を持つクロスリンクされたアプリケーションです。

解決方法

1. `dumpbin` Microsoft* Visual Studio* ユーティリティーを使用して、アプリケーションの依存関係をチェックしてください。

   ```
   > dumpbin /DEPENDENTS app.exe

   Microsoft (R) Program Maintenance Utility Version 11.00.61030.0 Copyright
   (C) Microsoft Corporation.All rights reserved.Dump of file app.exe

   File Type:EXECUTABLE IMAGE
     Image has the following dependencies:
     impi.dll
     foo.dll
   ```

2. また、サードパーティーのライブラリーの依存関係をチェックします。

   ```
   > dumpbin /DEPENDENTS foo.dll

   Microsoft (R) Program Maintenance Utility Version 11.00.61030.0 Copyright
   (C) Microsoft Corporation.All rights reserved. Dump of file foo.dll

   File Type: DLL
     Image has the following dependencies:
     impimt.dll
   ```

3. `foo.dll` は `impimt.dll` に、`app.exe` は `impi.dll` に依存します。
4. アプリケーションを下位互換モードで `impimt.dll` と再リンクします。コンパイラー・ドライバーの `-link_mpi` オプションを使用するか、`I_MPI_LINK` 環境変数を使用します。

   ```
   > mpiicc -link_mpi opt_mt_compat app.c -o app.exe foo.lib
   ```

   または

   ```
   > set I_MPI_LINK=opt_mt_compat
   > mpiicc app.c -o app.exe foo.lib
   ```

11.2.7. 例 7

症状 / エラーメッセージ

```
> hydra_service -start
OpenSCManager failed:
Access is denied.(error 5)
```

原因

Hydra サービスを開始するには管理者権限が必要です。一度 Hydra サービスを開始すると、すべてのユーザーは `mpiexec.exe` でプロセスを起動できます。

解決方法

管理者権限を持つコマンド・プロンプトから次のコマンドを実行します。

```
> hydra_service –start
Intel(R) MPI Library Hydra Process Manager started.
```