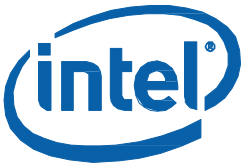


インテル® MPI ライブラリー for Windows*

リファレンス・マニュアル (5.1 Update 3)

目次

著作権と商標について 4

1. 概要 5

1.1. インテル® MPI ライブラリーの紹介 5

1.2. 対象となる開発者 5

1.3. 新機能 6

1.4. 表記規則 6

1.5. 関連情報 6

2. コマンド・リファレンス 7

2.1. コンパイラーのコマンド 7

2.1.1. コンパイラーのコマンドオプション 8

2.1.2. 設定ファイル 10

2.1.3. プロファイル 10

2.1.4. 環境変数 10

2.2. ジョブ開始コマンド 13

2.2.1. グローバルオプション 13

2.2.2. ローカルオプション 16

2.2.3. 環境変数 17

2.3. SMPD (Simple Multi-Porpose Daemon*) 25

2.4. スケーラブルなプロセス管理システム (Hydra) 27

2.4.1. Hydra サービス 27

2.4.2. ジョブ開始コマンド 28

2.4.3. グローバルオプション 28

2.4.4. ローカルオプション 36

2.4.5. 拡張デバイス制御オプション 37

2.4.6. 環境変数 38

2.5. Microsoft* HPC ジョブ・スケジューラーと統合 44

2.6. PBS Pro* ジョブ・スケジューラーと統合 44

2.7. 異種オペレーティング・システムのクラスターをサポート 45

2.8. プロセッサー情報ユーティリティー 45

3. ユーザー認証 49

3.1. 概要 49

3.2. インストール 49

3.2.1. Active Directry* の設定 49

3.3. 環境変数 50

4. チューニング・リファレンス 52

4.1. mpitune ユーティリティーを使用 52

4.1.1. クラスター固有のチューニング 56

4.1.2. チューニング・ユーティリティーの出力 58

4.2. プロセスのピニング (固定) 58

4.2.1. プロセスピニングのデフォルト設定 58

4.2.2. プロセッサーの識別 59
4.2.3. 環境変数 59
4.2.4. OpenMP* API との相互利用 66
4.3. ファブリック制御 78
4.3.1. 通信ファブリック制御 78
4.3.2. 共有メモリー制御 84
4.3.3. DAPL ネットワーク・ファブリック制御 90
4.3.4. TCP ネットワーク・ファブリック制御 99
4.4. 集団操作制御 101
4.4.1. I_MPI_ADJUST ファミリー 101
4.4.2. I_MPI_MSG ファミリー 110
4.5. その他 115
4.5.1. 互換性制御 115
4.5.2. ダイナミック・プロセスのサポート 115
4.5.3. 統計収集モード 116
4.5.4. ILP64 サポート 131
4.5.5. ユニファイド・メモリー管理 132
4.6. ダイナミック・リンク・ライブラリーの安全なロード 133
5. グラフィカル・ユーティリティー 135
6. 用語集 138
7. 索引 139

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。

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、Intel Xeon Phi、Xeon 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。

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. 概要

このリファレンス・マニュアルは、インテル® MPI ライブラリーのコマンドとチューニング向けのリファレンスを提供します。本書には、次の章が含まれます。

ドキュメント構成

章	説明
第 1 章 - はじめに	このドキュメントについて
第 2 章 - コマンド・リファレンス	コンパイラー・コマンドのオプションと変数、ジョブ開始コマンドと SMPD のコマンドについて
第 3 章 - ユーザー認証	ユーザー認証方式の違い
第 4 章 - チューニング・リファレンス	実行時にプログラムの動作やパフォーマンスに影響を与える環境変数について
第 5 章 - グラフィカル・ユーティリティー	インテル® MPI ライブラリーとともに配布されるグラフィカル・ユーザー・インターフェイス (GUI) ユーティリティーについて
第 6 章 - 用語集	このドキュメントで使用される基本用語
第 7 章 - 索引	オプションと環境変数名のリファレンス

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。開発者のニーズに応じて、MPI-3.0 の機能を使用することを可能にする標準ライブラリーをインテル® プラットフォーム向けに提供します。

インテル® MPI ライブラリーは、開発者がソフトウェアや動作環境を変更することなく、プロセッサーを変更もしくはアップグレードしたり、新たな技術のインターコネクトが利用可能なった時点で導入することを可能にします。

このライブラリーは以下を含みます。

- 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・ソフトウェア開発キット (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・ドライバー、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. 対象となる開発者

このリファレンス・マニュアルは、経験のある開発者がインテル® MPI ライブラリーのすべての機能を理解するのに役立ちます。

1.3. 新機能

このドキュメントは、インテル® MPI ライブラリー 5.1 Update 3 for Windows* 向けのアップデートを反映しています。

このドキュメントでは、次の項目に関する変更が行われています。

- I_MPI_ADJUST ファミリーの新しい環境変数 I_MPI_ADJUST_BCAST_SEGMENT の説明。
- I_MPI_ADJUST ファミリーの新しい非ブロッキング集団アルゴリズムの説明。
- 非ブロッキング操作による I_MPI_STATS_SCOPE のターゲット操作の拡張。「ネイティブ統計形式」をご覧ください。

1.4. 表記規則

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。

この書式	ハイパーリンクです。
この書式	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。

1.5. 関連情報

次の関連ドキュメントもご覧ください。

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. コマンド・リファレンス

この章では、それぞれのコマンドとその使い方に関する情報を提供します。

- コンパイラーのコマンド
- ジョブ開始コマンド
- SMPD (Simple Multi-Porpose Daemon*)
- スケーラブルなプロセス管理システム (Hydra)
- Microsoft* HPC ジョブ・スケジューラーと統合
- PBS Pro* ジョブ・スケジューラーと統合
- 異種オペレーティング・システムのクラスターをサポート
- プロセッサー情報ユーティリティー

2.1. コンパイラーのコマンド

(SDK のみ)

次の表は、MPI コンパイラー・コマンドと利用可能なコンパイラー、コンパイラー・ファミリー、言語、およびアプリケーション・バイナリー・インターフェイス (ABI) を示します。

表 2.1-1 インテル® MPI ライブラリーのコンパイラー・ドライバー

コンパイラーのコマンド	利用可能なコンパイラー	サポートされる言語	サポートされる ABI
一般的なコンパイラー			
`mpicc.bat`	`cl.exe`	C	64 ビット
`mpicxx.bat`	`cl.exe`	C++	64 ビット
`mpifc.bat`	`ifort.exe`	Fortran 77/Fortran 95	64 ビット
Microsoft* Visual C++* コンパイラー			
`mpicl.bat`	`cl.exe`	C/C++	64 ビット
インテル® Fortran および C++ コンパイラー 14.0 から 16.0 およびそれ以降			
`mpiicc.bat`	`icl.exe`	C	64 ビット
`mpiicpc.bat`	`icl.exe`	C++	64 ビット
`mpiifort.bat`	`ifort.exe`	Fortran 77/Fortran 95	64 ビット

- コンパイラー・コマンドは、インテル® MPI ライブラリー開発キットでのみ利用できます。
- コンパイラー・コマンドは、<インストール・ディレクトリー>\<アーキテクチャー>\bin ディレクトリーに配置されます。64 ビット対応コンパイラーでインテル® 64 アーキテクチャー向けの場合、<インストール・ディレクトリー>\intel64em64t\bin ディレクトリーに配置されます。

- 環境設定は、<インストール・ディレクトリー>\intel64\bin\mpivars.bat を実行することで設定できます。異なるライブラリー構成の設定が必要な場合、mpivars.bat スクリプトに次のいずれかの引数を指定して対応する環境に切り替えることができます。
  - debug
  - release
  - debug_mt
  - release_mt

  マルチスレッド版の最適化されたライブラリーが、デフォルトで選択されます。
- PATH に対応するコンパイラー (64 ビットなど適切な) へのパスが設定されていることを確認してください。
- 既存の MPI アプリケーションをインテル® MPI ライブラリーへ移行する場合、すべてのソースを再コンパイルします。
- コンパイラー・コマンドのヘルプを表示するには、引数なしでコマンドを実行します。

2.1.1. コンパイラーのコマンドオプション

-profile=<プロファイル名>

MPI プロファイル・ライブラリーを使用するには、このオプションを指定します。プロファイル・ライブラリーは、次のいずれかの方法で選択します。

- <インストール・ディレクトリー>\<アーキテクチャー>\etc に配置される、設定ファイル <プロファイル名>.con を介して。詳細については、「プロファイル」をご覧ください。
- 設定ファイルが存在しない場合、インテル® MPI ライブラリーと同じディレクトリーにある、ライブラリー lib<プロファイル名>.lib とリンクします。

-t または -trace

-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。

VT_ROOT 環境変数に、インテル® Trace Collector のインストール先のパスを含める必要があります。

-check_mpi

この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。VT_ROOT 環境変数に、インテル® Trace Collector のインストール先のパスを含める必要があります。

-ilp64

ILP64 をサポートする場合、このオプションを指定します。インテル® MPI ライブラリーのすべての整数引数が、64 ビットとして扱われます。

-no_ilp64

ILP64 サポートを無効にする場合、このオプションを指定します。このオプションは、インテル® Fortran コンパイラーの -i8 オプションと併用する必要があります。

注意

インテル® Fortran コンパイラーで -i8 オプションを指定した場合、リンク時に -ilp64 オプションを指定する必要があります。詳細ついては、「ILP64 のサポート」をご覧ください。

-link_mpi=<引数>

常にインテル® MPI ライブラリーの指定するバージョンとリンクする場合、このオプションを指定します。引数の詳しい説明は、「I_MPI_LINK」環境変数をご覧ください。このオプションは、特定のライブラリーを選択するほかのオプションをすべてオーバーライドします。

/Zi、/Z7 または /ZI

デバッグモードでプログラムをコンパイルし、デバッグバージョンのインテル® MPI ライブラリーとリンクするため、これらのオプションを指定します。/Zi、/Z7、/ZI またはデバッグビルドで追加されるデバッグ情報を使用する方法については、環境変数「I_MPI_DEBUG」をご覧ください。

注意

/ZI オプションは、C/C++ コンパイラーのみでサポートされます。

-O

コンパイラーの最適化を有効にする場合、このオプションを指定します。

このオプションが指定されると、libirc ライブラリーが呼び出されます。これらのライブラリーの多くのルーチンは、互換マイクロプロセッサーよりもインテル製マイクロプロセッサーでより高度に最適化されます。

-echo

コマンドスクリプトの動作をすべて表示するには、このオプションを指定します。

-show

実際にコンパイルすることなく、コンパイラーが呼び出される時のオプションを表示します。コンパイラー・フラグとオプションを確認するには、次のコマンドを指定します。

```
> mpiicc.bat -show -c test.c
```

リンクフラグ、オプションおよびライブラリーを確認するには、次のコマンドを指定します。

```
> mpiicc.bat -show -o a.exe test.obj
```

このオプションは、コンパイラーを使用して、複雑なビルドを行う際にコマンドラインを確認するのに役立ちます。

-show_env

コンパイラーの起動時に設定されている環境変数を確認するには、このオプションを使用します。

-{cc cxx fc}=<コンパイラー>

コンパイラーを選択します。

インテル® C++ コンパイラーを選択する場合、次のコマンドを使用します。

```
> mpiicc.bat -cc=icl.exe -c test.c
```

この際、icl.exe へのパスが PATH に設定済みである必要があります。別の方法として、フルパスでコンパイラーを指定することができます。

注意

このオプションは、mpiicc.bat と mpifc.bat オプションでのみ利用できます。

-v

コンパイラー・ドライバーのバージョンを表示します。

2.1.2. 設定ファイル

次の命名規則に従って、インテル® MPI ライブラリーのコンパイラー設定ファイルを作成できます。

<インストール・ディレクトリー>\<アーキテクチャー>\etc\mpi<コンパイラー>-<名前>.conf

説明:

<コンパイラー>={cc,fc} は、コンパイルする言語に依存します。

<名前> は、サポートされるコンパイラーの名称を指定します。

例えば、cc -64 向けの <名前> は、cc--64 です。

コンパイラー・コマンドで環境の変更を有効にするには、コンパイルとリンク前に環境設定スクリプトを実行する必要があります。

2.1.3. プロファイル

インテル® MPI ライブラリーのコンパイラー・ドライバーの -profile オプションを使用して、プロファイル・ライブラリーを選択することができます。また、独自のプロファイル設定を <インストール・ディレクトリー>\<アーキテクチャー>\etc\<プロファイル名>.conf に作成できます。次の環境設定を設定できます。

PROFILE_PRELIB - インテル® MPI ライブラリーの前にインクルードするライブラリー (とパス)
PROFILE_POSTLIB - インテル® MPI ライブラリーの後にインクルードするライブラリー (とパス)
PROFILE_INCPATHS - 任意のインクルードファイル向けの C プリプロセッサーの引数

例えば、次の内容の <インストール・ディレクトリー>\<アーキテクチャー>\etc\myprof.conf を作成します。

```
SET PROFILE_PRELIB=<myprof のパス>\lib\myprof.lib
SET PROFILE_INCPATHS=-I"<myprof のパス>\include"
```

この新しいプロファイルを選択するには、関連するコンパイラー・ドライバーのコマンドライン引数に -profile=myprof を指定します。

2.1.4. 環境変数

I_MPI_{CC,CXX,FC,F77,F90}_PROFILE

デフォルトのプロファイル・ライブラリーを指定します。

構文

I_MPI_{CC,CXX,FC,F77,F90}_PROFILE=<プロファイル名>

引数

<プロファイル名>	デフォルトのプロファイル・ライブラリーを指定します。

説明

デフォルトで使用する特定の MPI プロファイル・ライブラリーを選択するため、この環境変数を設定します。これは、mpiicc.bat やほかのインテル® MPI ライブラリーのコンパイラー・ドライバーの引数として、-profile=<プロファイル名> を指定するのと同じです。

I_MPI_{CC,CXX,FC,F77,F90} (MPICH_{CC,CXX,FC,F77,F90})

使用するコンパイラーのパス/名前を設定します。

構文

I_MPI_{CC,CXX,FC,F77,F90}=<コンパイラー>

引数

<コンパイラー>	使用するコンパイラーのフルパス/名前を設定します。

説明

デフォルトで使用する特定のコンパイラーを選択するため、この環境変数を設定します。検索パスに含まれていない場合、フルパスでコンパイラーを指定します。

注意

一部のコンパイラーは、追加のコマンドライン・オプションを必要とします。

I_MPI_ROOT

インテル® MPI ライブラリーのインストール先のディレクトリーを設定します。

構文

I_MPI_ROOT=<パス>

引数

<パス>	インテル® MPI ライブラリーのインストール先のディレクトリーを指定します。

説明

インテル® MPI ライブラリーのインストール先のディレクトリーを指定するには、この環境変数を設定します。

VT_ROOT

インテル® Trace Collector のインストール先のディレクトリーを設定します。

構文

VT_ROOT=<パス>

引数

<パス>	インテル® Trace Collector のインストール先のディレクトリーを指定します。

説明

インテル® Trace Collector のインストール先のディレクトリーを指定するには、この環境変数を設定します。

I_MPI_COMPILER_CONFIG_DIR

コンパイラーの設定ファイルの場所を設定します。

構文

I_MPI_COMPILER_CONFIG_DIR=<パス>

引数

<パス>	コンパイラーの設定ファイルの場所を指定します。デフォルトは、<インストール・ディレクトリー>\<アーキテクチャー>\etc です。

説明

コンパイラーの設定ファイルのデフォルトの場所を変更するには、この環境変数を設定します。

I_MPI_LINK

リンクするインテル® MPI ライブラリーの特定のバージョンを選択します。

構文

```
I_MPI_LINK=<引数>
```

引数

<引数>	ライブラリーのバージョン。
opt	シングルスレッド版の最適化されたインテル® MPI ライブラリー。
opt_mt	マルチスレッド版の最適化されたインテル® MPI ライブラリー。
dbg	シングルスレッド版のデバッグ向けインテル® MPI ライブラリー。
dbg_mt	マルチスレッド版のデバッグ向けインテル® MPI ライブラリー。
opt_mt_compat	マルチスレッド版の最適化されたインテル® MPI ライブラリー (下位互換モード)。
dbg_compat	シングルスレッド版のデバッグ向けインテル® MPI ライブラリー (下位互換モード)。
dbg_mt_compat	マルチスレッド版のデバッグ向けインテル® MPI ライブラリー (下位互換モード)。
log	シングルスレッド版のログ用インテル® MPI ライブラリー。
log_mt	マルチスレッド版のログ用インテル® MPI ライブラリー。

説明

指定するインテル® MPI ライブラリーのバージョンと常にリンクする場合、このオプションを指定します。

注意

下位互換モードは、古いインテル® MPI ライブラリー (impimt.dll、impid.dll および impidmt.dll) とリンクする際に使用します。

2.2. ジョブ開始コマンド

mpiexec.smpd

構文

```
mpiexec.smpd <g-オプション> <l-オプション> <実行形式>
```

または

```
mpiexec.smpd <g-オプション> <l-オプション> <実行形式> : \
<l-オプション> <実行形式>
```

または

```
mpiexec.smpd -configfile <ファイル>
```

引数

<g-オプション>	すべての MPI プロセスに適用するグローバルオプション。
<l-オプション>	単一の引数セットに適用するローカルオプション。
<実行形式>	.\a.exe または path\実行形式ファイル名。
<ファイル>	コマンドライン・オプションを持つファイル。

説明

最初のコマンドラインの構文を使用して、単一の引数セットで <実行形式> のすべての MPI プロセスを開始できます。例えば、次のコマンドは指定した <プロセス数> で a.exe を実行します。

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

2 番目のコマンドライン構文は、複数の MPI プログラムを開始したり、同じ MPI プログラムを異なる引数セットで開始できます。例えば、次のコマンドは指定された実行形式ファイルを異なるホスト上で実行します。

```
> mpiexec.smpd -n 2 -host host1 a.exe :\
-n 2 -host host2 b.exe
```

3 番目のコマンドライン構文は、コマンドラインを指定された <ファイル> から読み込みます。単一の引数セットを持つコマンドの場合、コマンド全体は <ファイル> 内の単一行に指定する必要があります。複数の引数セットを持つコマンドの場合、各コマンドはそれぞれ <ファイル> 内の単一行に指定する必要があります。グローバルオプションは、常に <ファイル> の先頭行になければいけません。

SMPD (Simple Multi-Porpose Daemon*) サービスは、mpiexec.smpd を実行する前に起動されている必要があります。

注意

クラスターのすべてのノードで PATH に実行形式へのパスが設定されていない場合、<実行形式> に a.exe の代わりに <パス>\a.exe を指定します。

2.2.1. グローバルオプション

-machinefile <マシンファイル>

このオプションは、<マシンファイル> を介してプロセスの配置を制御する際に使用します。開始時の総プロセス数は、-n オプションで制御されます。

マシンファイルは、完全に修飾された、もしくは短いホスト名のリストを 1 行に 1 つ持ちます。空白行と先頭文字が '#' の行は無視されます。

ホスト名を繰り返すことで、ホスト上に追加のプロセスを配置します。同じホスト名の重複を避けるため、次の形式で記述できます: <ホスト名>:<プロセス数>。以下に例を示します。

```
host1
host1
host2
host2
host3
```

上記のマシンファイルは次と等価です。

```
host1:2
host2:2
host3
```

また、各ノードで使用するネットワーク・インターフェイスを指定することもできます。

<ホスト名>:<プロセス数>[ifhn=<インターフェイス_ホスト名>]

-configfile <ファイル名>

このオプションは、コマンドライン・オプションを含むファイルを <ファイル名> に指定します。空白行と先頭文字が '#' の行は無視されます。例えば、実行形式 a.exe と b.exe を dapl ファブリックを使用して host1 と host2 で実行するには、次のコマンドラインを含む設定ファイルを作成します。

```
-host host1 -env I_MPI_DEBUG 2 -env I_MPI_FABRICS shm:dapl -n 2 ./a.exe
-host host2 -env I_MPI_DEBUG 2 -env I_MPI_FABRICS shm:dapl -n 2 ./b.exe
```

上記の設定ファイルを使用して MPI アプリケーションを起動するには、次のコマンドを使用します。

```
> mpiexec.smpd -configfile <ファイル名>
```

注意

このオプションは、mpiexec.smpd コマンドラインの解析を中断させてしまうため、単独で使用します。

-g<l-オプション>

ローカルオプション <l-オプション> をグローバルに適用するには、このオプションを使用します。すべてのローカルオプションについては、「ローカルオプション」をご覧ください。アプリケーション起動時のデフォルト値は、-genvuser オプションです。

注意

ローカルオプションは、グローバルオプションよりも優先順位が上です。

- -genv オプションは最も優先順位が高い
- -genvlist と -genvexcl は -genv よりも優先順位が低い
- そして -genvnone、-genvuser、および -genvall は、最も優先順位が低い

-l

このオプションは、標準出力に書き込まれたすべての行の先頭に、MPI プロセスのランクを挿入します。

-tune

mpitune ユーティリティーで収集されたデータを使用して、インテル® MPI ライブラリーのパフォーマンスを最適化するには、このオプションを使用します。<設定ファイル> が指定されていない場合、最適なチューニング・オプションが適用されます。それ以外は、指定された設定ファイルが適用されます。

64 ビット・モードのインテル® 64 アーキテクチャー向けの設定ファイルのデフォルト位置は<インストール・ディレクトリー>\intel64 です。デフォルトの位置を変更するには、I_MPI_TUNER_DATA_DIR 環境変数を設定します。

詳細は、自動チューニング・ユーティリティーをご覧ください。

-p <ポート> または -port <ポート>

このオプションは、mpiexec.smpd が接続に使用する SPMD ポートを指定します。SMPD がデフォルト以外のポートを使用している場合、このオプションが役に立ちます。

-hosts n <ホスト 1> <プロセス数 1> <ホスト 2> <プロセス数 2> ... <ホスト n> <プロセス数 n>

現在の引数セットで実行する各 MPI プロセス上の特定のホストリストとプロセス数を指定します。例えば、次のコマンドラインは、host1 と host2 上で実行形式 a.exe を実行します。2 つのプロセスが、host1 で実行され、1 つのプロセスが host2 で実行されます。

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

-logon

このオプションを使用すると、アカウント名とパスワードが求められます。

-delegate

委任機能を持つドメインベースの認証を有効にします。

-impersonate

制限付きドメインベースの認証を有効にします。これは、リモートマシン上のファイルを開いたり、割り当てられたネットワーク・ドライブにアクセスできなくなります。

-pwdfile <ファイル名>

指定したファイルからアカウント名とパスワードを読み込みます。ファイルには、最初の行にアカウントを、次の行にパスワードを記述します。

-nopopup_debug

プロセスがクラッシュした際に、システムが表示するポップアップ・ダイアログを無効にします。

-exitcodes

各プロセスが終了した際に、プロセスの終了コードを表示します。

-verbose

smpd の出力を標準出力へリダイレクトします。

-localroot

ホストがローカルである場合、mpiexec.smpd から直接 root プロセスを呼び出します。GUI アプリケーションを起動する際にこのオプションを使用できます。対話型のプロセスは、ジョブのほかのプロセスの前に起動されている必要があります。

例:

```
> mpiexec.smpd -n 1 -host <ホスト 2> -localroot interactive.exe :-n 1 -host
<ホスト 1> background.exe
```

-localonly

アプリケーションをローカルノードのみで実行します。ローカルノードのみでこのオプションを使用する場合、smpd サービスは必要ありません。

-register [-user n]

レジストリーに登録するユーザー名とパスワードを暗号化します。

-remove [-user n]

レジストリーから、暗号化された資格情報を削除します。ユーザー・インデックス n が省略された場合、すべてのエントリーが削除されます。

-validate [-user n] [-host ホスト名]

現在のまたは特定のホストの暗号化された資格情報を確認します。このオプションを使用する場合、特定のユーザー・インデックスを設定する必要があります。ユーザー・インデックスが省略されると、デフォルト値 0 となります。

-timeout <秒>

ジョブのタイムアウト時間を指定します。

-whoami

現在のユーザー名を表示します。

-h、-help または --help

mpiexec.smpd のヘルプメッセージを表示します。

2.2.2. ローカルオプション

-n <プロセス数> または -np <プロセス数>

現在の引数セットで実行する MPI プロセス数を指定します。

-env <環境変数> <値>

現在の引数セットですべての MPI プロセスに、指定された <値> の <環境変数> を設定します。

-envnone

現在の引数セットで MPI プロセスへの任意の環境変数の伝搬を抑制します。

-envlist <環境変数名のリスト>

引数リストと現在の値を渡します。

-envuser

次の環境変数を除き、すべてのユーザー環境変数の値をすべての MPI プロセスに渡します。

%ALLUSERSPROFILE%、%APPDATA%、%CommonProgramFiles%、%CommonProgramFiles (x86)%、

%COMPUTERNAME%、%HOMEDRIVE%、%HOMEPATH%、%NUMBER_OF_PROCESSORS%、%OS%、

%PROCESSOR_ARCHITECTURE%、%PROCESSOR_IDENTIFIER%、%PROCESSOR_LEVEL%、

%PROCESSOR_REVISION%、%ProfilePath%、%ProgramFiles%、%ProgramFiles (x86)%、

%SystemDrive%、%SystemRoot%、%TEMP%、%TMP%、%USERDNSDOMAIN%、%USERDOMAIN%、

%USERNAME%、%USERPROFILE%

これは、デフォルトの設定です。

-envexcl <環境変数名のリスト>

現在の引数セットで MPI プロセスへの指定された環境変数の伝搬を抑制します。

-host <ノード名>

現在の引数セットで MPI プロセスを実行する特定の <ノード名> を指定します。例えば、次のコマンドラインは、host1 上でのみ実行形式 a.exe を実行します。

```
> mpiexec.smpd -n 2 -host host1 ./a.exe
```

-path <ディレクトリー>

実行する <実行形式> へのパスを指定します。セパレーターは「;」です。

-dir <ディレクトリー> または -wdir <ディレクトリー>

現在の引数セットで実行する <実行形式> が使用するワーキング・ディレクトリーを指定します。

-map <ドライブ:\\ホスト名\共有名>

<実行形式> を開始する前に、ネットワーク割り当てドライブを作成します。割り当てられたドライブは、ジョブが終了すると自動的に解除されます。

-mapall

すべてのノードで <実行形式> を開始する前に、ユーザーが定義したネットワーク割り当てドライブの作成を要求します。割り当てられたドライブは、ジョブが終了すると自動的に解除されます。

2.2.3. 環境変数

I_MPI_DEBUG

MPI プログラムが実行を開始するとデバッグ情報を出力します。

構文

```
I_MPI_DEBUG=<レベル>[,<フラグ>]
```

引数

<レベル>	デバッグ情報のレベルを指定します。
0	デバッグ情報を出力しません。これは、デフォルト値です。
1	詳細なエラー診断を出力します。
2	どの I_MPI_FABRICS (I_MPI_DEVICE) とインテル® MPI ライブラリーの設定が使用されているかを確認します。
3	有効な MPI ランク、プロセス ID およびノード割り当てテーブルを出力します。
4	プロセスのピニング情報を出力します。
5	インテル® MPI ライブラリー固有の環境変数の値を出力します。
6	集団操作アルゴリズムの設定を出力します。
> 6	追加のデバッグ情報を出力します。

<フラグ>	カンマで区切られたデバッグフラグのリスト。
pid	各デバッグメッセージにプロセス ID を表示します。
tid	マルチスレッド・ライブラリー向けのデバッグメッセージにスレッド ID を表示します。
time	各デバッグメッセージに時間を表示します。
datetime	各デバッグメッセージに日付と時間を表示します。
host	各デバッグメッセージにホスト名を表示します。
level	各デバッグメッセージにレベルを表示します。
scope	各デバッグメッセージに範囲を表示します。
line	各デバッグメッセージに行番号を表示します。
file	各デバッグメッセージにソースファイル名を表示します。
nofunc	ルーチン名を表示しません。
norank	ランクを表示しません。
flock	異なるプロセスやスレッドからのデバッグ出力を同期します。
nobuf	デバッグ出力にバッファー I/O を使用しません。

説明

デバッグ情報の出力を制御するには、この環境変数を設定します。

注意

すべてのランクに同じ <レベル> 値を設定します。

I_MPI_DEBUG_OUTPUT 環境変数に値を設定することで、デバッグ情報の出力ファイル名を指定できます。

簡単にプロセスを特定するには、I_MPI_DEBUG に設定する数値の前に + や - 記号を追加します。この設定は、デバッグ出力の行に MPI プロセスのランク、Windows* のプロセス ID、およびプロセス起動時に定義されたホスト名をプリフィクスとして追加します。次に例を示します。

```
> mpiexec.smpd -n <プロセス数> -env I_MPI_DEBUG +2 a.exe
```

または

```
> mpiexec.smpd -n <プロセス数> -env I_MPI_DEBUG +2,pid,host a.exe
```

出力は次のようになります。

```
[ランク#プロセス Id@ホスト名] デバッグメッセージ
```

注意

mpiicc.bat とともに /Zi、/Z7 または /Z7 を指定してコンパイルすると、かなりの量の追加メッセージが出力されます。

I_MPI_DEBUG_OUTPUT

デバッグ情報の出力先のファイル名を設定します。

構文

```
I_MPI_DEBUG_OUTPUT=<引数>
```

引数

<引数>	文字列
stdout	標準出力に表示します (デフォルト値)。
stderr	標準エラー出力に表示します。
<ファイル名>	デバッグ情報を出力するファイル名を指定します (最大 256 文字)。

説明

アプリケーションが生成する出力から、デバッグ情報を分離したい場合にこの環境変数を設定します。%r、%p または %h フォーマットを使用して、ランク、プロセス ID または、ホスト名をファイル名に追加できます。

I_MPI_PRINT_VERSION

ライブラリーのバージョンを表示します。

構文

```
I_MPI_PRINT_VERSION=<引数>
```

引数

<引数>	バイナリー・インジケーター。
`enable | yes | on | 1`	ライブラリーのバージョンを表示します。
`disable | no | off | 0`	何もしません。これは、デフォルト値です。

説明

MPI アプリケーションが実行を開始するときに、インテル® MPI ライブラリーのバージョン情報の表示を `enable` (有効)/ `disable` (無効) にするには、この環境変数を設定します。

I_MPI_NETMASK

ソケット経由の MPI 通信のインターフェイスを選択します。

構文

`I_MPI_NETMASK=<引数>`

引数

<引数>	ネットワーク・インターフェイス(文字列) を定義します。
<インターフェイスの略称>	ネットワーク・インターフェイスの略: `ib` もしくは `eth`。
`ib`	IPoIB* を選択します。
`eth`	イーサーネットを選択します。これは、デフォルト値です。
<ネットワーク・アドレス>	ネットワーク・アドレス。末尾ゼロはネットマスクを示します。
<ネットワーク・アドレス/ネットマスク>	ネットワーク・アドレス。<ネットマスク> 値には、ネットマスクのレングスを指定します。
<インターフェイスのリスト>	コロンで区切られたネットワーク・アドレスまたは、インターフェイスの略称のリスト。

説明

この環境変数を設定して、`sock` と `ssm` 通信方式のソケット経由の MPI 通信のためのネットワーク・インターフェイスを選択します。インターフェイスのリストを指定した場合、ノード上で最初に検出されたインターフェイスが通信に使用されます。

例

1. InfiniBand* (IPoIB) ファブリック経由の IP を選択するには、次のように設定します。

   ```
   I_MPI_NETMASK=ib I_MPI_NETMASK=eth
   ```

2. ソケット通信に特定のネットワークを選択するには、次のように設定します。この設定は、`255.255.0.0` ネットマスクを意味します。

   ```
   I_MPI_NETMASK=192.169.0.0
   ```

3. ネットマスクが設定されたソケット通信に特定のネットワークを選択するには、次のように設定します。

   ```
   I_MPI_NETMASK=192.169.0.0/24
   ```

4. ソケット通信に特定のネットワーク・インターコネクトを選択するには、次のように設定します。

   I_MPI_NETMASK=192.169.0.5/24:ib0:192.169.0.0

注意

ライブラリーが、I_MPI_NETMASK の値で指定された適切なインターフェイスを検出できない場合、値はネットワーク・アダプターのディスクリプション・フィールドを検索する文字列として使用されます。ディスクリプション・フィールドに文字列が見つかった場合、そのネットワーク・インターフェイスがソケット通信に使用されます。例えば、I_MPI_NETMASK=myri が設定され、ディスクリプション・フィールドが Myri-10G adapter のような文字列を含んでいれば、このインターフェイスが選択されます。

I_MPI_JOB_TIMEOUT (MPIEXEC_TIMEOUT)

mpiexec.smpd のタイムアウトを設定します。

構文

I_MPI_JOB_TIMEOUT=<タイムアウト>

廃止された構文

MPIEXEC_TIMEOUT=<タイムアウト>

引数

<タイムアウト>	mpiexec.smpd のタイムアウト時間を秒単位で指定します。
<タイムアウト> >= 0	デフォルト値は 0 で、タイムアウトしません。

説明

この環境変数は、mpiexec.smpd がジョブの起動後 <タイムアウト> 秒でジョブを強制終了する時間を設定します。<タイムアウト> 値は、ゼロよりも大きくなければいけません。不正な値は無視されます。

注意

mpiexec.smpd コマンドを実行する前に、シェル環境で I_MPI_JOB_TIMEOUT 環境変数を設定します。<タイムアウト> 値を設定するのに、-genv や -env オプションを使ってはいけません。これらのオプションは、MPI プロセス環境に環境変数の値を渡すときにのみ使用します。

I_MPI_SMPD_VERSION_CHECK

強力な SMPD バージョンチェックを行うかどうか設定します。これは、Windows* でのみ利用できます。

構文

I_MPI_SMPD_VERSION_CHECK=<引数>

引数

<引数>	バイナリー・インジケーター。
`enable | yes | on | 1`	バージョンをチェックし、一致しない場合失敗します。これは、デフォルト値です。
`disable | no | off | 0`	バージョンが一致しない場合、警告メッセージを出力し、処理を続行します。

説明

強力な SMPD バージョンチェックを行うかどうか設定します。バージョンの不一致が検出された場合、インテル® MPI ライブラリーはアプリケーションを強制終了します。SMPD のバージョンチェックを無効にするには、`I_MPI_SMPD_VERSION_CHECK` 環境変数に `disable` (無効) を設定します。

I_MPI_DAT_LIBRARY

特定の DAT ライブラリーを選択します。

構文

`I_MPI_DAT_LIBRARY=<ライブラリー>`

引数

<ライブラリー>	デフォルトの `dat.dll` の代わりに使用するライブラリーを指定します。

説明

使用する特定の DAT ライブラリーを選択するため、この環境変数を設定します。検索パスに含まれていない場合、フルパスで DAT ライブラリーを指定します。

注意

この環境変数は、DAPL* プロバイダーを利用するときにのみ使用します。

I_MPI_TUNER_DATA_DIR

チューニング設定ファイルがあるディレクトリーへの代替パスを設定します。

構文

`I_MPI_TUNER_DATA_DIR=<パス>`

引数

<パス>	自動チューニング・ユーティリティーの出力ディレクトリーを指定します。 デフォルトは、<インストール・ディレクトリー>`\intel64\etc` です。

説明

チューニング設定ファイルの代替の場所を変更するには、この環境変数を設定します。

I_MPI_PLATFORM

最適化するプラットフォームを選択します。

構文

I_MPI_PLATFORM=<プラットフォーム>

引数

<プラットフォーム>	最適化するプラットフォーム (文字列)。
auto[:min]	すべてのノードで最も古いインテル® アーキテクチャー・プロセッサー向けの最適化を行います。これは、デフォルト値です。
auto:max	すべてのノードで最も新しいインテル® アーキテクチャー・プロセッサー向けの最適化を行います。
auto:most	すべてのノードで最も多いインテル® アーキテクチャー・プロセッサー向けの最適化を行います。同数の場合は、新しいプラットフォームが選択されます。
uniform	ローカルに最適化。ノード間の選択が異なる場合、動作は予測できません。
none	特定の最適化を行いません。
htn | generic	インテル® Xeon® プロセッサー 5400 番台とその他のインテル® アーキテクチャー (開発コード名 Harpertown) 向けに最適化。
nhm	インテル® Xeon® プロセッサー 5500、6500、7500 番台とその他のインテル® アーキテクチャー (開発コード名 Nehalem) 向けに最適化。
wsm	インテル® Xeon® プロセッサー 5600、3600 番台とその他のインテル® アーキテクチャー (開発コード名 Westmere) 向けに最適化。
snb	インテル® Xeon® プロセッサー E3、E5、E7 ファミリーとその他のインテル® アーキテクチャー (開発コード名 Sandy Bridge) 向けに最適化。
ivb	インテル® Xeon® プロセッサー E3、E5、E7 v2 製品ファミリーとその他のインテル® アーキテクチャー (開発コード名 Ivy Bridge) 向けに最適化。
knc	インテル® Xeon Phi™ コプロセッサー (開発コード名 Knights Corner) 向けに最適化。インテル® Xeon Phi™ コプロセッサーがクラスター上に存在する場合、この値がデフォルトになります。
hsw	インテル® Xeon® プロセッサー E3、E5、E7 v3 製品ファミリーとその他のインテル® アーキテクチャー (開発コード名 Haswell) 向けに最適化。

説明

事前定義されたプラットフォーム設定を使用するには、この環境変数を設定します。この環境変数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

auto:min、auto:max および auto:most を設定すると、MPI ジョブ開始時の時間が長くなることがあります。

I_MPI_PLATFORM_CHECK

類似性チェックの最適化を on/off にします。

構文

I_MPI_PLATFORM_CHECK=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	プラットフォームの類似性チェックの最適化を on にします。これは、デフォルト値です。
disable | no | off | 0	プラットフォームの類似性チェックの最適化を off にします。

説明

すべての最適化プラットフォームの設定の類似性を確認する際に、この環境変数を設定します。すべてのランク上の設定が同一でない場合、インテル® MPI ライブラリーはプログラムを強制終了します。この設定を disable (無効) にすることで、MPI プログラムの起動時間を短縮できます。

I_MPI_THREAD_LEVEL_DEFAULT

MPI_Init() を初期化に使用する場合、マルチスレッド・ライブラリーの MPI スレッド環境を初期化するためこの環境変数を設定します。

構文

I_MPI_THREAD_LEVEL_DEFAULT=<スレッドレベル>

引数

<スレッドレベル>	スレッドサポートのデフォルトレベルを定義します。
SINGLE | single	スレッドサポートのデフォルトとレベルを MPI_THREAD_SINGLE に設定します。
FUNNELED | funneled	スレッドサポートのデフォルトとレベルを MPI_THREAD_FUNNELEDE に設定します。初期化に MPI_Init() を使用する際のデフォルトです。
SERIALIZED | serialized	スレッドサポートのデフォルトとレベルを MPI_THREAD_SERIALIZED に設定します。
MULTIPLE | multiple	スレッドサポートのデフォルトとレベルを MPI_THREAD_MULTIPLE に設定します。

説明

初期化に MPI_Init() を使用している場合、マルチスレッド・ライブラリーのスレッドサポートのデフォルトレベルを定義するため、I_MPI_THREAD_LEVEL_DEFAULT を設定します。

2.3. SMPD (Simple Multi-Porpose Daemon*)

smpd

簡単な複数用途デーモン

構文

```
smpd.exe [ -h] [ --help] [ -port <ポート> ] [ -d ] \
[ -install | -regserever ] [ -start ] [ -stop ] \
[ -shutdown <ホスト名> ] [ -status <ホスト名> ] \
[ -restart <ホスト名> ] [ -anyport ] [ -hosts] [ -sethosts ] \
[ -set <オプション名> <オプション値> ] [ -get <オプション名> ] \
[ -traceon <ログファイル名> [ <ホスト A> <ホスト B> ...]] \
[ -traceoff [ <ホスト A> <ホスト B> ...]]\
[ -remove | -unregister | -uninstall ] [ -register_spn]\
[ -remove_spn ] [ -V ] [ -version ]
```

引数

-h| --help	ヘルプメッセージを表示します。
-p <ポート> | -port <ポート>	smpd が監視するポートを指定します。
-d | -debug	smpd をデバッグモードで開始します。
-install -regserver	smpd サービスをインストールします。
-start	smpd サービスを開始します。
-stop	smpd サービスを停止します。
-shutdown <ホスト名>	指定した <ホスト名> の smpd をシャットダウンします。
-status <ホスト名>	指定した <ホスト名> の smpd のステータスを取得します。
-restart <ホスト名>	指定した <ホスト名> の smpd を再起動します。
-anyport	管理のため任意のポートを使用します。
-hosts	smpd リングリストを取得します。
-sethosts	指定したホストから smpd リングを設定します。この設定はすべてのユーザーに影響します。
-set <オプション名> <オプション値>	<オプション値> キーを HKEY_LOCAL_MACHINE レジストリー・キーに登録します。
-get <オプション名>	HKEY_LOCAL_MACHINE レジストリー・キーから、登録されている <オプション値> の値を取得します。

`-traceon <ログファイル名> <ホスト A> <ホスト B> ...`	smpd を再起動して、<ログファイル名> に出力を保存します。
`-traceoff <ホスト A> <ホスト B>`	ログを生成せずに smpd を再起動します。
`-remove | -unregserver | -uninstall`	smpd サービスを削除します。
`-register_spn`	このコマンドが実行されるクラスターノードの Windows* ドメインに、サービス・プリンシパル名 (SPN) を登録します。
`-remove_spn`	このコマンドが実行されるクラスターノードの Windows* ドメインの、サービス・プリンシパル名 (SPN) を削除します。
`-V`	インテル® MPI ライブラリーのバージョン情報を取得します。
`-version`	smpd のバージョン情報を表示します。

説明

SMPD (Simple Multipurpose Daemon*) は、インテル® MPI ライブラリーの並列ジョブを開始するプロセス管理システムです。ジョブを実行する前に、各ホスト上で smpd サービスを開始しリングに接続します。

SMPD サービスをインストール、アンインストール、開始および停止するには、`smpd.exe` コマンドを使用します。

例

1. 次のコマンドを使用して SMPD サービスをインストールします。

   ```
   > smpd.exe -install
   ```

注意

このコマンドは、管理者権限を持つユーザーで実行する必要があります。その後、すべてのユーザーが `mpiexec` を使用して MPI ジョブを起動できます。

2. デバッグモードで SMPD サービスを開始するには、次のコマンドを使用します。

   ```
   > smpd.exe -d
   ```

注意

SMPD (Simple Multi-Porpose Daemon*) 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。並列ジョブを起動する代わりに、スケーラブルなプロセス管理システム (Hydra) を使用します。

2.4. スケーラブルなプロセス管理システム (Hydra)

Windows* 上の Hydra は、Linux* 上の Hydra と同じ環境変数とオプションを使用できます。この章では、固有のオプションについて説明します。

2.4.1. Hydra サービス

hydra_service

HYDRA サービス・エージェント

構文

```
hydra_service.exe [ -install | -regserver ] [ -start ] [ -stop ] \
[ -remove | -unregister | -uninstall ] [ -register_spn ]
```

引数

-install -regserver	hydra サービスをインストールします。
-start	hydra サービスを開始します。
-stop	hydra サービスを停止します。
-shutdown <ホスト名>	<ホスト名> で指定されたノードの hydra サービスをシャットダウンします。
-status <ホスト名>	<ホスト名> で指定されたノードの hydra サービスのステータスを取得します。
-restart <ホスト名>	<ホスト名> で指定されたノードの hydra サービスを再起動します。
-remove | -unregserver | -uninstall	hydra サービスを削除します。
-register_spn	このコマンドが実行されるクラスターノードの Windows* ドメインに、サービス・プリンシパル名 (SPN) を登録します。
-remove_spn	このコマンドが実行されるクラスターノードの Windows* ドメインの、サービス・プリンシパル名 (SPN) を削除します。

説明

Hydra サービスは、インテル® MPI ライブラリーの並列ジョブを開始するプロセス管理システムです。ジョブを実行する前に、各ホスト上でサービスを開始してください。

例

1. サービスをインストール、アンインストール、開始および停止するには、hydra_service.exe コマンドを使用します。

   ```
   > hydra_service.exe -install
   ```

注意

このコマンドは、管理者権限を持つユーザーで実行する必要があります。その後、すべてのユーザーが mpiexec を使用して MPI ジョブを起動できます。

2. 次のコマンドを使用してサービスを削除します。

```
> hydra_service.exe -remove
```

2.4.2. ジョブ開始コマンド

mpiexec

mpiexec は、SMPD プロセス管理へのよりスケーラブルな代替手段です。

構文

mpiexec <g-オプション> <l-オプション> <実行形式>

または

mpiexec <g-オプション> <l-オプション> <実行形式 1> : \

<l-オプション> <実行形式 2>

引数

<g-オプション>	すべての MPI プロセスに適用されるグローバルオプション。
<l-オプション>	単一の引数セットに適用されるローカルオプション。
<実行形式>	.\a.exe または path\実行形式ファイル名。

2.4.3. グローバルオプション

-hostfile <ホストファイル> または -f <ホストファイル>

アプリケーションを実行するホスト名を指定します。ホスト名が重複する場合 1 つだけ適用されます。

詳細は、「I_MPI_HYDRA_HOST_FILE」環境変数をご覧ください。

注意

クラスターノード上のプロセスの配置を変更するには、-perhost、-ppn、-grr *および* -rr オプションを使用します。

-machinefile <マシンファイル> または -machine <マシンファイル>

このオプションは、<マシンファイル> を介してプロセスの配置を制御する際に使用します。開始する総プロセス数は、-n オプションで制御されます。

マシン内にピニングする場合、マシンファイルの各行で -binding=map オプションを使用できます。次に例を示します。

```
> type .\machinefile
node0:2 binding=map=0,3
node1:2 binding=map=[2,8]
```

```
node0:1 binding=map=8
> mpiexec.hydra -machinefile
```

-genv <環境変数> <値>

すべての MPI プロセスに、指定された <値> の <環境変数> を設定します。

-genvall

すべての環境変数をすべての MPI プロセスに伝搬するのを有効にします。

-genvnone

任意の環境変数を任意の MPI プロセスに伝搬するのを抑制します。

-genvlist <環境変数名のリスト>

引数リストと現在の値を渡します。<環境変数名のリスト> は、すべての MPI プロセスに送るカンマで区切られた環境変数のリストです。

-pmi-connect <モード>

プロセス管理インターフェイス (PMI) のメッセージキャッシュのモードを選択します。利用可能な <モード> は以下です。

- nocache - PMI メッセージをキャッシュしません。
- cache - PMI への要求を最小限に抑えるため、ローカル pmi_proxy 管理プロセスで PMI メッセージをキャッシュします。キャッシュされた情報は、子の管理プロセスへ伝搬されます。
- lazy-cache - PMI 情報の伝搬要求でのキャッシュモード。lazy-cache モードはデフォルトモードです。

詳細は、「I_MPI_HYDRA_PMI_CONNECT」環境変数をご覧ください。

-perhost <プロセス数>、-ppn <プロセス数> または -grr <プロセス数>

グループ内のすべてのホスト上で、ラウンドロビン・スケジューリングにより連続した数の MPI プロセスを配置します。詳細は、「I_MPI_PERHOST」環境変数をご覧ください。

-rr

ラウンドロビン・スケジューリングにより、異なるホスト上で連続した MPI プロセスを配置します。このオプションは、-perhost 1 と等価です。詳細は、「I_MPI_PERHOST」環境変数をご覧ください。

(SDK のみ) -trace-pt2pt

ポイントツーポイント操作に関する情報を収集します。

(SDK のみ) -trace-collectives

集合操作に関する情報を収集します。

注意

トレースファイルのサイズやメッセージチェッカーのレポート数を減らすには、-trace-pt2pt と -trace-collectives オプションを使用します。このオプションは、スタティックおよびダイナミック・リンクされたアプリケーションの両方で利用できます。

-configfile <ファイル名>

このオプションは、コマンドライン・オプションを含むファイルの <ファイル名> に指定します。空白行と先頭文字が '#' の行は無視されます。

-branch-count <数値>

mpiexec コマンドまたは、pmi_proxy 管理プロセスで起動される子管理プロセスの数を制限します。

詳細は、「I_MPI_HYDRA_BRANCH_COUNT」環境変数をご覧ください。

-pmi-aggregate または -pmi-noaggregate

PMI リクエストの集約を On または Off に切り替えます。デフォルトは、集約が有効となる -pmi-aggregate です。

詳細は、「I_MPI_HYDRA_PMI_AGGREGATE」環境変数をご覧ください。

-hosts <ノードリスト>

MPI プロセスを実行する特定の <ノードリスト> を指定します。例えば、次のコマンドラインは、host1 と host2 で実行形式 a.exe を実行します。

```
> mpiexec -n 2 -hosts host1,host2 ./a.exe
```

注意

<ノードリスト> が 1 つのノードのみを含む場合、このオプションはローカルオプションとして解釈されます。詳細は、「ローカルオプション」をご覧ください。

-iface <インターフェイス>

適切なネットワーク・インターフェイスを選択します。例えば、InfiniBand* ネットワークの IP エミュレーションが ib0 に設定されている場合、次のコマンドを使用できます。

```
> mpiexec -n 2 -iface ib0 ./a.exe
```

詳細は、「I_MPI_HYDRA_IFACE」環境変数をご覧ください。

-l

このオプションは、標準出力に書き込まれたすべての行の先頭に、MPI プロセスのランクを挿入します。

-tune [<引数>]

引数には以下を指定します。

<引数>= {<ディレクトリー名>, <設定ファイル>}

mpitune ユーティリティーで収集されたデータを使用して、インテル® MPI ライブラリーのパフォーマンスを最適化するには、このオプションを使用します。

注意

このオプションを使用する前に、パフォーマンス・チューニング・データを収集するため mpitune ユーティリティーを使用します。

<引数> が指定されていない場合、指定された設定向けに最適なチューニング・オプションが適用されます。設定ファイルのデフォルトの場所は、<インストール・ディレクトリー>/<アーキテクチャー>/etc ディレクトリーです。

異なる場所にある設定ファイルを指定するには、<引数>=<ディレクトリー名> を設定します。異なる設定ファイルを指定するには、<引数>=<設定ファイル> を設定します。

-s <spec>

指定された MPI プロセスへの標準入力をリダイレクトします。

引数

<spec>	MPI プロセスのランクを定義します。
all	すべてのプロセスを使用します。
<l>,<m>,<n>	使用するプロセスのリストを指定します。この場合 <l>、<m> および <n> のみを使用します。デフォルト値は 0 です。
<k>,<l>-<m>,<n>	使用するプロセスの範囲を指定します。この場合 <k>、<l> から <m>、および <n> を使用します。

-noconf

設定ファイルに記載される mpiexec 設定ファイルの処理を無効にします。

-ordered-output

MPI プロセスから出力されるデータの混在を避けるには、このオプションを使用します。このオプションは、標準出力と標準エラー出力に影響します。

注意

このオプションを使用する場合、各プロセスの最後の行の出力を改行 (\n) で終了します。そうしないと、アプリケーションが応答を停止することがあります。

-path <ディレクトリー>

実行する <実行形式> ファイルへのパスを指定します。

-tmpdir

一時ファイルのディレクトリーを設定します。

詳細は、「I_MPI_TMPDIR」環境変数をご覧ください。

-version または -V

インテル® MPI ライブラリーのバージョンを表示します。

-info

インテル® MPI ライブラリーのビルド情報を表示します。このオプションが指定されると、その他のコマンドライン引数は無視されます。

-delegate

委任機能を持つドメインベースの認証を有効にします。

-impersonate

制限付きドメインベースの認証を有効にします。これにより、リモートマシン上のファイルを開いたり、割り当てられたネットワーク・ドライブにアクセスできなくなります。

-localhost

起動ノードのローカルホスト名を明示的に指定します。

例:

```
> mpiexec -localhost <ローカルホスト名> -machinefile <ファイル> -n 2 test.exe
```

-localroot

ホストがローカルである場合、mpiexec から直接 root プロセスを呼び出します。GUI アプリケーションを起動する際にこのオプションを使用できます。対話型のプロセスは、ジョブのほかのプロセスの前に起動されている必要があります。

例:

```
> mpiexec -n 1 -host <ホスト 2> -localroot interactive.exe :-n 1 -host <ホスト 1>
background.exe
```

-localonly

アプリケーションをローカルノードのみで実行します。ローカルノードのみでこのオプションを使用する場合、Hydra サービスは必要ありません。

-register

レジストリーに登録するユーザー名とパスワードを暗号化します。

-remove

レジストリーから、暗号化された資格情報を削除します。

-validate

現在のホストの暗号化された資格証明書を検証します。

-whoami

現在のユーザー名を表示します。

-map <ドライブ:\\ホスト名\共有名>

<実行形式> を開始する前に、ネットワーク割り当てドライブを作成します。割り当てられたドライブは、ジョブが終了すると自動的に解除されます。

-mapall

ノードで <実行形式> を開始する前に、ユーザーが定義したネットワーク割り当てドライブの作成を要求します。割り当てられたドライブは、ジョブが終了すると自動的に解除されます。

バインディング・オプション

-binding

MPI プロセスを特定のプロセッサーにピニングまたはバインドし、望ましくないプロセスのマイグレーションを避けるため、このオプションを使用します。次の構文で記述します。引用符で 1 つのメンバーリストを囲みます。各パラメーターは、単一のピニング・プロパティーに対応します。

このオプションは、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。

構文

`-binding"<パラメーター>=<値>[;<パラメーター>=<値> ...]"`

パラメーター

`pin`	ピニング (固定) スイッチ。
`enable | yes | on | 1`	ピニング (固定) プロパティーを on にします。これは、デフォルト値です。
`disable | no | off | 0`	ピニング (固定) プロパティーを `off` にします。

`cell`	ピニング (固定) の解像度。
`unit`	基本プロセッサー・ユニット (論理 CPU)。
`core`	マルチコアシステムのプロセッサー・コア。

`map`	プロセスマッピング。
`spread`	プロセスは、ほかのプロセッサーのセルに連続的にマッピングされます。そのため、プロセスは隣接するセルとリソースを共有しません。
`scatter`	プロセスは、ほかのプロセッサーのセルに離れてマッピングされます。隣接するプロセスは、マルチコアトポロジーで最も離れているセルにマッピングされます。
`bunch`	プロセスは、ソケットごとのプロセッサー数/ソケット数によって別々のプロセッサー・セルにマッピングされます。 各ソケットのプロセッサーは、マルチコアトポロジーに最も近いセルの集合です。
`p0,p1,...,pn`	プロセスは、`p0`、`p1`、`... pn` リスト上のプロセッサーの使用に応じて別々のプロセッサーにマッピングされます: `i` 番目のプロセスは `pi` プロセッサーにマッピングされます。 ここで `pi` は次のいずれかの値となります。 • プロセッサー番号 `n` • プロセッサー番号の範囲 `n-m` • 対応するプロセスのピニングが必要ない場合は `-1`
`[m0,m1,...,mn]`	`i` 番目のプロセスは、次の規則による 16 進マスクの m_i で定義されるプロセッサーのサブセット上に割り当てられます。 m_i の `j` 番目のビットが `1` であれば、`j` 番目のプロセッサーはサブセット m_i に含まれます。

domain	ノード上のプロセッサー・ドメインのセット。
cell	セットの各ドメインは、単一のプロセッサー・セル (unit もしくは core)。
core	セットの各ドメインは、特定のコアを共有するプロセッサーのセルで構成されます。
cache1	セットの各ドメインは、特定のレベル 1 キャッシュを共有するプロセッサーのセルで構成されます。
cache2	セットの各ドメインは、特定のレベル 2 キャッシュを共有するプロセッサーのセルで構成されます。
cache3	セットの各ドメインは、特定のレベル 3 キャッシュを共有するプロセッサーのセルで構成されます。

cache	セットの要素は、キャッシュ 1、キャッシュ 2、キャッシュ 3 の中で最も大きいドメインです。
socket	セットの各ドメインは、特定のソケットに配置されるプロセッサーのセルで構成されます。
node	ノード上のすべてのプロセッサー・セルは、単一のドメインに配置されます。
<サイズ>[:<レイアウト>]	セットの各ドメインは、<サイズ> のプロセッサー・セルで構成されます。<サイズ> は次の値です。 • auto - ドメインサイズ = セル数/プロセス数 • omp - ドメインサイズ = OMP_NUM_THREAD 環境変数の値 • 正の整数 - 実際のドメインサイズ **注意** ドメインのサイズは、ノード上のプロセッサー・コア数で制限されます。 ドメイン内の各メンバーの位置は、オプションの <レイアウト> パラメーター値で定義されます。 • compact - マルチコアトポロジー内で可能な限りほかと近くに配置 • scatter - マルチコアトポロジー内で可能な限りほかと遠くに配置 • range - プロセッサーの BIOS による番号で配置 <レイアウト> パラメーターを省略すると、compact が想定されます。

order	ドメインをリニアに順序付けします。
compact	隣接するドメインがマルチコアトポロジーで最も近くなるようドメインの順番を設定します。
scatter	隣接するドメインがマルチコアトポロジーで最も遠くなるようドメインの順番を設定します。
range	BIOS のプロセッサーの番号付けに従ってドメインの順番を設定します。

offset	ドメインリストのオフセット。
<n>	リニアな順番のドメインで開始するドメインの整数番号。このドメインは番号 0 を取得。ほかのドメイン番号は、巡回してシフトします。

ブートオプション

-bootstrap <ブートストラップ・サーバー>

使用するブートストラップ・サーバーを選択します。ブートストラップ・サーバーは、システムで提供される基本的なリモートノードへのアクセスメカニズムです。デフォルトのブートストラップ・サーバーはサービスです。

引数

<引数>	すべての MPI プロセスに適用されるグローバルオプション。
service	hydra サービス・エージェントを使用します。これは、デフォルト値です。
ssh	セキュアシェルを使用します。
fork	アプリケーションをローカルノードのみで実行する際に使用します。

インテル® MPI ライブラリーで、-bootstrap ssh オプションを使用するには、ノード間の ssh 接続が確立されている必要があります。対応する ssh クライアントの場所が、PATH 環境変数に設定されていることを確認してください。

-bootstrap-exec <ブートストラップ・サーバー>

ブートストラップ・サーバーとして使用する実行ファイルを設定します。次に例を示します。

```
$ mpiexec -bootstrap-exec <ブートストラップ・サーバーの実行形式> \
-f hosts.file -env <変数 1> <値 1> -n 2 ./a.out
```

詳細は、「I_MPI_HYDRA_BOOTSTRAP」環境変数をご覧ください。

その他のオプション

-verbose または -v

mpiexec から提供される次のようなデバッグ情報を表示します。

- サービスプロセスの引数
- 開始時にアプリケーションに渡される環境変数と引数
- ジョブが起動中の PMI リクエストとレスポンス

詳細は、「I_MPI_HYDRA_DEBUG」環境変数をご覧ください。

-print-rank-map

MPI ランクのマッピングを表示します。

-print-all-exitcodes

すべてのプロセスが終了した際に終了コードを表示します。

2.4.4. ローカルオプション

-n <プロセス数> または -np <プロセス数>

現在の引数セットで実行する MPI プロセス数を指定します。

-env <環境変数> <値>

現在の引数セットですべての MPI プロセスに、指定された <値> の <環境変数> を設定します。

-envall

現在の引数セットですべての環境変数を伝搬します。詳細は、「I_MPI_HYDRA_ENV」環境変数をご覧ください。

-envnone

現在の引数セットで MPI プロセスに任意の環境変数の伝搬を抑制します。

-envlist <環境変数名のリスト>

引数リストと現在の値を渡します。<環境変数名のリスト> は、MPI プロセスに送るカンマで区切られた環境変数のリストです。

-host <ノード名>

MPI プロセスを実行する特定の *<ノード名>* を指定します。例えば、次のコマンドラインは、host1 と host2 で実行形式 a.exe を実行します。

```
> mpiexec -n 2 -host host1 ./a.exe :-n 2 -host host2 ./a.exe
```

-path <ディレクトリー>

現在の引数セットで実行する <実行形式> ファイルへのパスを指定します。

-wdir <ディレクトリー>

現在の引数セットで実行する <実行形式> ファイルが使用するワーキング・ディレクトリーを指定します。

-hostos <ホスト OS>

特定のホストにインストールされているオペレーティング・システムを指定します。MPI プロセスは、このオプションの指示に従って各ホスト上で起動されます。デフォルト値は windows です。

引数

<引数>	文字列パラメーター。
linux	ホストには Linux* がインストールされています。
windows	ホストには Windows* がインストールされています。これは、デフォルト値です。

注意

このオプションは、-host オプションと組み合わせて使用されます。例えば、次のコマンドラインは、host1 で a.exe を実行し、host2 で a.out を実行します。

```
> mpiexec -n 1 -host host1 -hostos windows a.exe :-n 1 -host host2 \ -hostos
linux ./a.out
```

2.4.5. 拡張デバイス制御オプション

-rdma

RDMA ネットワーク・ファブリックを選択します。アプリケーションは、最初に dapl,ofa リストから利用可能な RDMA ネットワーク・ファブリックの使用を試みます。利用できない場合、tcp が使用されます。このオプションは、-genv I_MPI_FABRICS_LIST dapl,ofa,tcp -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-RDMA

RDMA ネットワーク・ファブリックを選択します。アプリケーションは、最初に dapl,ofa リストから利用可能な RDMA ネットワーク・ファブリックの使用を試みます。指定されたファブリックが存在しない場合、アプリケーションは失敗します。このオプションは、-genv I_MPI_FABRICS_LIST dapl,ofa -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-dapl

DAPL ネットワーク・ファブリックを選択します。アプリケーションは、DAPL ネットワーク・ファブリックの使用を試みます。利用できない場合、tcp,ofa リストのほかのファブリックが使用されます。が使用されます。このオプションは、-genv I_MPI_FABRICS_LIST dapl,tcp,ofa -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-DAPL

DAPL ネットワーク・ファブリックを選択します。指定されたファブリックが存在しない場合、アプリケーションは失敗します。このオプションは、-genv I_MPI_FABRICS_LIST dapl -genv I_MPI_FALLBACK 0 オプションを指定するのと等価です。

-ib

OFA ネットワーク・ファブリックを選択します。アプリケーションは、OFA ネットワーク・ファブリックの使用を試みます。利用できない場合、dapl,tcp リストのほかのファブリックが使用されます。このオプションは、-genv I_MPI_FABRICS_LIST ofa,dapl,tcp -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-IB

OFA ネットワーク・ファブリックを選択します。指定されたファブリックが存在しない場合、アプリケーションは失敗します。このオプションは、-genv I_MPI_FABRICS_LIST ofa -genv I_MPI_FALLBACK 0 オプションを指定するのと等価です。

2.4.6. 環境変数

I_MPI_HYDRA_HOST_FILE

アプリケーションが実行するホストファイルを設定します。

構文

I_MPI_HYDRA_HOST_FILE=<引数>

廃止された構文

HYDRA_HOST_FILE=<引数>

引数

<引数>	文字列パラメーター。
<ホストファイル>	ホストファイルへの絶対もしくは相対パス。

説明

この環境変数にはホストファイルを設定します。

I_MPI_HYDRA_DEBUG

デバッグ情報を表示します。

構文

I_MPI_HYDRA_DEBUG=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	デバッグ出力を on にします。
disable | no | off | 0	デバッグ出力を off にします。これは、デフォルト値です。

説明

デバッグモードを有効にするには、この環境変数を設定します。

I_MPI_HYDRA_ENV

環境変数の伝搬を制御します。

構文

I_MPI_HYDRA_ENV=<引数>

引数

<引数>	文字列パラメーター。
all	すべての MPI プロセスにすべての環境変数を渡します。

説明

MPI プロセスへの環境の伝搬を制御するには、この環境変数を設定します。デフォルトでは、起動ノードの環境全体が MPI プロセスへ渡されます。この環境変数を設定すると、リモートシェルによって環境変数をオーバーライドします。

I_MPI_JOB_TIMEOUT、I_MPI_MPIEXEC_TIMEOUT (MPIEXEC_TIMEOUT)

mpiexec のタイムアウト時間を設定します。

構文

```
I_MPI_JOB_TIMEOUT=<タイムアウト>
I_MPI_MPIEXEC_TIMEOUT=<タイムアウト>
```

廃止された構文

```
MPIEXEC_TIMEOUT=<タイムアウト>
```

引数

<タイムアウト>	mpiexec のタイムアウト時間を秒単位で指定します。
<n> >=0	デフォルト値は 0 で、タイムアウトしません。

説明

この環境変数は、mpiexec がジョブの起動後 <タイムアウト> 秒でジョブを強制終了する時間を設定します。<タイムアウト> 値は、ゼロよりも大きくなければいけません。不正な値は無視されます。

注意

mpiexec コマンドを実行する前に、シェル環境で I_MPI_JOB_TIMEOUT 環境変数を設定します。<タイムアウト> 値を設定するのに、-genv や -env オプションを使ってはいけません。これらのオプションは、MPI プロセス環境に環境変数の値を渡すときにのみ使用します。

I_MPI_JOB_TIMEOUT_SIGNAL (MPIEXEC_TIMEOUT_SIGNAL)

タイムアウトでジョブが終了した際に送信するシグナルを定義します。

構文

```
I_MPI_JOB_TIMEOUT_SIGNAL=<番号>
```

廃止された構文

```
MPIEXEC_TIMEOUT_SIGNAL=<番号>
```

引数

<番号>	シグナル番号を定義します。
<n> > 0	デフォルト値は 9 (SIGKILL) です。

説明

I_MPI_JOB_TIMEOUT 環境変数で指定されたタイムアウト時間が経過した場合、MPI ジョブを停止するため送信するシグナル番号を定義します。システムでサポートされないシグナル番号を設定した場合、mpiexec は警告メッセージを表示し、デフォルトのシグナル番号 9 (SIGKILL) でタスクを終了します。

I_MPI_JOB_ABORT_SIGNAL

ジョブが予期せずに終了した場合に、すべてのプロセスに送信するシグナルを定義します。

構文

I_MPI_JOB_ABORT_SIGNAL=<番号>

引数

<番号>	シグナル番号を定義します。
<n> > 0	デフォルト値は 9 (SIGKILL) です。

説明

この環境変数を設定して、タスクを強制終了するシグナルを定義します。サポートされないシグナル番号を設定した場合、mpiexec は警告メッセージを表示し、デフォルトのシグナル番号 9 (SIGKILL) でタスクを終了します。

I_MPI_JOB_SIGNAL_PROPAGATION (MPIEXEC_SIGNAL_PROPAGATION)

シグナルの伝搬を制御します。

構文

I_MPI_JOB_SIGNAL_PROPAGATION=<引数>

廃止された構文

MPIEXEC_SIGNAL_PROPAGATION=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	伝搬をオンにします。
disable | no | off | 0	伝搬をオフにします。これは、デフォルト値です。

説明

この環境変数を設定して、シグナル (SIGINT、SIGALRM、SIGTERM) の伝搬を制御します。シグナルの伝搬を有効にすると、受信したシグナルはすべての MPI ジョブを実行するプロセスへ送信されます。シグナルの伝搬を無効にすると、MPI ジョブを実行するすべてのプロセスは、デフォルトのシグナル 9 (SIGKILL) で停止されます。

I_MPI_HYDRA_BOOTSTRAP

ブートストラップ・サーバーを設定します。

構文

I_MPI_HYDRA_BOOTSTRAP=<引数>

引数

<引数>	文字列パラメーター。
service	hydra サービス・エージェントを使用します。
ssh	セキュアシェルを使用します。これは、デフォルト値です。
fork	fork 呼び出しを使用します。

説明

この環境変数は、ブートストラップ・サーバーを設定します。

注意

mpiexec コマンドを実行する前に、シェル環境で I_MPI_HYDRA_BOOTSTRAP 環境変数を設定します。<引数> 値を設定するのに、-env オプションを使ってはいけません。これらのオプションは、MPI プロセス環境に環境変数の値を渡すときに使用します。

I_MPI_HYDRA_BOOTSTRAP_EXEC

ブートストラップ・サーバーとして使用する実行ファイルを設定します。

構文

I_MPI_HYDRA_BOOTSTRAP_EXEC=<引数>

引数

<引数>	文字列パラメーター。
<実行形式>	実行ファイル名。

説明

この環境変数は、ブートストラップ・サーバーとして使用する実行ファイルを設定します。

I_MPI_HYDRA_PMI_CONNECT

PMI メッセージの処理方式を定義します。

構文

I_MPI_HYDRA_PMI_CONNECT=<値>

引数

<値>	使用するアルゴリズム。
nocache	PMI メッセージをキャッシュしません。
cache	PMI への要求を最小限に抑えるため、ローカル pmi_proxy 管理プロセスで PMI メッセージをキャッシュします。キャッシュされた情報は、自動的に子の管理プロセスへ伝搬されます。
lazy-cache	オンデマンドの cache モードの伝搬。これは、デフォルト値です。

説明

この環境変数を設定して、PMI メッセージの処理方式を選択します。

I_MPI_PERHOST

mpiexec コマンドの -perhost オプションのデフォルトを設定します。

構文

I_MPI_PERHOST=<値>

引数

<値>	-perhost オプションで使用されるデフォルトの値を定義します。
整数値 > 0	オプションの正確な値。
all	ノード上のすべての論理 CPU。
allcores	ノード上のすべてのコア (物理 CPU)。これは、デフォルト値です。

説明

この環境変数を設定して、-perhost オプションに適用されるデフォルトの値を定義します。
I_MPI_PERHOST 環境変数が定義されている場合、-perhost オプションは指定されている値を意味します。

I_MPI_HYDRA_BRANCH_COUNT

階層的な分岐数を設定します。

構文

I_MPI_HYDRA_BRANCH_COUNT =<数値>

引数

<数値>	番号。
<n> >= 0	• ノードが 128 未満の場合、デフォルト値は -1 です。これは階層構造が無いことを意味します。 • ノードが 128 以上の場合、デフォルト値は 32 です。

説明

この環境変数を設定して、mpiexec コマンドまたは、pmi_proxy 管理プロセスで起動される子管理プロセスの数を制限します。

I_MPI_HYDRA_PMI_AGGREGATE

PMI メッセージの集約を on/off にします。

構文

I_MPI_HYDRA_PMI_AGGREGATE=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	PMI メッセージの集約を有効にします。これは、デフォルト値です。
disable | no | off | 0	PMI メッセージの集約を無効にします。

説明

この環境変数を設定して、PMI メッセージの集約を有効/無効にします。

I_MPI_HYDRA_IFACE

ネットワーク・インターフェイスを設定します。

構文

I_MPI_HYDRA_IFACE=<引数>

引数

<引数>	文字列パラメーター。
<ネットワーク・インターフェイス>	システムで設定されたネットワーク・インターフェイス。

説明

この環境変数は、使用するネットワーク・インターフェイスを設定します。例えば、InfiniBand* ネットワークの IP エミュレーションが ib0 に設定されている場合、-iface ib0 を使用します。

I_MPI_TMPDIR (TMPDIR)

一時ディレクトリーを設定します。

構文

I_MPI_TMPDIR=<引数>

引数

<引数>	文字列パラメーター。
<パス>	一時ディレクトリーを設定します。デフォルト値は /tmp です。

説明

この環境変数を設定して、mpicleanup の入力ファイルを保存する一時ディレクトリーを指定します。

I_MPI_JOB_RESPECT_PROCESS_PLACEMENT

ジョブ・スケジューラーが提供するプロセスノードごとのパラメーターを使用するかどうか指定します。

構文

`I_MPI_JOB_RESPECT_PROCESS_PLACEMENT=<引数>`

引数

<値>	バイナリー・インジケーター。
`enable | yes | on | 1`	ジョブ・スケジューラーで提供されるプロセス配置を使用します。これは、デフォルト値です。
`disable | no | off | 0`	ジョブ・スケジューラーで提供されるプロセス配置を使用しません。

説明

`I_MPI_JOB_RESPECT_PROCESS_PLACEMENT=enable` に設定すると、Hydra プロセス管理はジョブ・スケジューラーで提供される PPN を使用します。

`I_MPI_JOB_RESPECT_PROCESS_PLACEMENT=disable` に設定すると、Hydra プロセス管理はコマンドライン・オプション、または `I_MPI_PERHOST` 環境変数で指定される PPN を使用します。

2.5. Microsoft* HPC ジョブ・スケジューラーと統合

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。

2.6. PBS Pro* ジョブ・スケジューラーと統合

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。

2.7. 異種オペレーティング・システムのクラスターをサポート

インテル® MPI ライブラリーは、Windows* と Linux* の異種環境をサポートします。Windows* と Linux* 上で利用可能な hydra プロセス管理は、インテル® MPI ライブラリーが 1 つのジョブを Linux* と Windows* 上で協調して処理すること可能にします。hydra プロセス管理の詳細については、スケーラブルなプロセス管理システム (Hydra) をご覧ください。

Linux* と Windows* オペレーティング・システム上でジョブを混在して実行するには、次の操作を行います。

- ジョブを実行するそれぞれのノードにインテル® MPI ライブラリーがインストールされ、利用可能であることを確認します。
- `-demux=select` と `I_MPI_FABRICS=shm:tcp` が、オペレーティング・システムが混在した実行をサポートします。
- `-bootstrap` オプションを設定します。オペレーティング・システム混在実行モードのデフォルトは、`bootstrap service` です。この設定を有効にするには、MPI ジョブを実行する各ノードの Windows* 上で hydra サービスが起動され、Linux* 上で Hydra persist server が起動されている必要があります。`-bootstrap ssh` を使用するため、Linux* と Windows* 間の ssh 接続を確立します。
- `-hostos` オプションを使用して、特定のホストにインストールされているオペレーティング・システムを指定します。
- 隣接するオペレーティング・システム環境の継承により不適切な設定が行われるのを避けるため、`I_MPI_ROOT` と `LD_LIBRARY_PATH` ローカル環境変数を使用します。

例えば、Windows* と Linux* の異種環境で `IMB-MPI1` ジョブを実行するには次のコマンドを使用します。

```
> mpiexec -demux select -genv I_MPI_FABRICS shm:tcp -env I_MPI_ROOT \
<Linux* のインストール・ディレクトリー> -env LD_LIBRARY_PATH \
<Linux* のインストール・ディレクトリー>/<アーキテクチャー>/lib \
-hostos linux -host <Linux* のホスト> -n 2 \
<Linux* のインストール・ディレクトリー>/<アーキテクチャー>/bin/IMB-MPI1 :-host \
<Windows* のホスト> -n 3 \
<Windows* のインストール・ディレクトリー>\<アーキテクチャー>\bin\IMB-MPI1
```

2.8. プロセッサー情報ユーティリティー

cpuinfo

`cpuinfo` ユーティリティーは、プロセッサーのアーキテクチャー情報を表示します。

構文

```
cpuinfo [[-]<オプション>]]
```

引数

<オプション>	1 文字のオプションシーケンス。それぞれのオプションは、出力データの特定の情報を制御します。
g	単一クラスターノードの一般的な情報を表示: • プロセッサーの製品名 • ノード上のパッケージ/ソケット数 • ノードと各パッケージ内のコアとスレッド数 • SMT モードの有効化

i	論理プロセッサー特定テーブルは、各論理プロセッサーのスレッド、コア、およびパッケージに応じて識別されます。 • Processor - 論理プロセッサー番号 • Thread Id - コア内の一意なプロセッサー識別子 • Core Id - パッケージ内の一意なコア識別子 • Package Id - ノード内の一意なパッケージ識別子
d	ノード分解テーブルは、ノードの内容を示します。各エントリーは、パッケージ、コア、および論理プロセッサーに関する情報を含みます。 • Package Id - 物理パッケージの識別子 • Cores Id - このパッケージ内のコア識別子のリスト • Processors Id - このパッケージ内のプロセッサー識別子のリスト。このリストの順番は、コアリストに対応します。括弧で囲まれたプロセッサー・グループは、1 つのコアに属します。
c	論理プロセッサーのキャッシュ共有は、特定のキャッシュレベルで共有されるサイズとプロセッサー・グループの情報を表示します。 • Size - キャッシュサイズ (バイト) • Processors - 括弧で囲まれたプロセッサー・リストは、このキャッシュを共有するか、共有しないかを示します。
s	マイクロプロセッサーの署名 16 進フィールド (インテルのプラットフォーム表記) は、署名値を示します。 • extended family (拡張ファミリー) • extended model (拡張モデル) • family (ファミリー) • model (モデル) • type (タイプ) • stepping (ステッピング)
f	マイクロプロセッサー機能フラグは、マイクロプロセッサーでサポートされる機能を示します。インテルのプラットフォーム表記が使用されます。
A	gidcsf に相当します。
gidc	デフォルトシーケンス。
?	ユーティリティーの使い方情報。

説明

cpuinfo ユーティリティーは、適切なプロセスのピニング設定を定義する際に使用する、プロセッサー・アーキテクチャーの情報を表示します。出力はいくつかのテーブルで構成されます。各テーブルは、引数テーブルにリストされる 1 つのオプションに対応します。

注意

アーキテクチャー情報は、インテル® 64 アーキテクチャー・ベースのシステムで利用できます。

cpuinfo ユーティリティーは、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、非インテル製マイクロプロセッサーでは一部の情報のみを取得できます。

例

インテル® Xeon® プロセッサーE5-2697 v2 製品ファミリー上で cpuinfo を実行した例:

```
$ cpuinfo A
```

```
Intel(R) processor family information utility, Version 5.1 Build 20150225 (build id: 11316)
Copyright (C) 2005-2015 Intel Corporation.  All rights reserved.

=====  Processor composition  =====
Processor name    : Intel(R) Xeon(R)  E5-2697 v2
Packages(sockets) : 2
Cores             : 24
Processors(CPUs)  : 48
Cores per package : 12
Threads per core  : 2


=====  Processor identification  =====
Processor     Thread Id.    Core Id.      Package Id.
0             0             0             0
1             0             1             0
2             0             2             0
3             0             3             0
4             0             4             0
5             0             5             0
6             0             8             0
7             0             9             0
8             0             10            0
9             0             11            0
10            0             12            0
11            0             13            0
12            0             0             1
13            0             1             1
14            0             2             1
15            0             3             1
16            0             4             1
17            0             5             1
18            0             8             1
19            0             9             1
20            0             10            1
21            0             11            1
22            0             12            1
23            0             13            1
24            1             0             0
25            1             1             0
26            1             2             0
27            1             3             0
28            1             4             0
29            1             5             0
30            1             8             0
31            1             9             0
32            1             10            0
33            1             11            0
34            1             12            0
35            1             13            0
36            1             0             1
37            1             1             1
38            1             2             1
39            1             3             1
40            1             4             1
41            1             5             1
42            1             8             1
43            1             9             1
44            1             10            1
45            1             11            1
46            1             12            1
47            1             13            1


=====  Placement on packages  =====
Package Id.   Core Id.      Processors
0             0,1,2,3,4,5,8,9,10,11,12,13   (0,24) (1,25) (2,26) (3,27) (4,28) (5,29) (6,30) (7,31) (8,32) (9,33) (10,34) (11,35)
1             0,1,2,3,4,5,8,9,10,11,12,13   (12,36) (13,37) (14,38) (15,39) (16,40) (17,41) (18,42) (19,43) (20,44) (21,45) (22,46) (23,47)


=====  Cache sharing  =====
Cache  Size    Processors
L1      32  KB
        (0,24) (1,25) (2,26) (3,27) (4,28) (5,29) (6,30) (7,31) (8,32) (9,33) (10,34) (11,35) (12,36) (13,37) (14,38) (15,39) (16,40) (17,41) (18,42) (19,43) (20,44) (21,45) (22,46) (23,47)
L2      256 KB
        (0,24) (1,25) (2,26) (3,27) (4,28) (5,29) (6,30) (7,31) (8,32) (9,33) (10,34) (11,35) (12,36) (13,37) (14,38) (15,39) (16,40) (17,41) (18,42) (19,43) (20,44) (21,45) (22,46) (23,47)
L3      30  MB
                (0,1,2,3,4,5,6,7,8,9,10,11,24,25,26,27,28,29,30,31,32,33,34,35) (12,13,14,15,16,17,18,19,20,21,22,23,36,37,38,39,40,41,42,43,44,45,46,47)


======  Processor Signature  ======
 _________ _________ ______ ________ _______ ____________
| xFamily | xModel  | Type | Family | Model | Stepping |
|_________|_________|______|________|_______|__________|
| 00      | 3       | 0    | 6      | e     | 4        |
|_________|_________|______|________|_______|__________|

======  Processor Feature Flags  ======
 _____ ___________ ________ _________ ________ _____ _____ _____ _____ _____ _________ _____ _____ _____
| SSE3 | PCLMULDQ | DTES64 | MONITOR | DS-CPL | VMX | SMX | EIST | TM2 | SSSE3 | CNXT-ID | FMA | CX16 | xTPR |
|______|__________|________|_________|________|_____|_____|______|_____|_____|_________|_____|______|______|
| 1    | 1        | 1      | 1       | 1      | 1   | 1   | 1    | 1   | 1     | 0       | 0   | 1    | 1    |
|______|__________|________|_________|________|_____|_____|______|_____|_____|_________|_____|______|______|
```

3. ユーザー認証

この章では、Windows* における異なる認証方式を説明し、それぞれの認証方式を使用する方法を紹介します。

3.1. 概要

インテル® MPI ライブラリーは、Windows* 環境でいくつかの認証方式をサポートします。

- パスワードベースの認証
- 委任機能を持つドメインベースの認証
- 制限付きドメインベースの認証を使用します

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。

ドメインベースの認証では、Windows* 環境で Microsoft* 社から提供されるセキュリティー・サービス・プロバイダー・インターフェイス (SSPI) を使用します。SSPI は、ドメインのポリシーに応じてリモートマシン上でユーザーの認証を可能にします。このような方式を使用する場合、アカウント名とパスワードを入力または保存する必要がありません。

注意

これらドメインベースの認証方式は、パスワードベースの認証と比べ、MPI タスクの起動時間が長くなります。これは、ドメインの構成に依存します。

注意

制限付きドメインベースの認証は、ネットワークへのアクセスを制限します。これは、リモートマシン上のファイルを開いたり、割り当てられたネットワーク・ドライブにアクセスできなくなることを意味します。

3.2. インストール

この機能は Windows* クラスター上でサポートされ、次の Microsoft* Windows* オペレーティング・システムで使用できます: Windows* HPC Server 2012 と Windows* HPC Pack 2012。

ドメイン・コントローラーで Microsoft* の Kerberos Distribution Center (KDC) を有効にしなければいけません。これは、デフォルトの動作です。

委任機能を持つドメインベースの認証方式を使用するには、ドメインの特定のインストールが必要です。これは次の手順で行います。

- ドメインの管理者権限を持っている場合、 IMPI インストーラーを使用します。
- 以降に示す「Active Directory* の設定」に従ってください。

3.2.1. Active Directry* の設定

Active Directory* の委任機能を有効にするには、次の操作を行います。

1. ドメイン・コントローラーに管理者アカウントでログインします。
2. クラスターノードの委任を有効にします。
3. ユーザーの委任を有効にします。

a. **[Active Directory ユーザーとコンピューター]** 管理者ユーティリティーの **[ユーザー]** リストを開きます。

b. 特定のユーザー・オブジェクトを右クリックし、**[プロパティ]** を選択します。

c. **[アカウント]** タブを選択し、**[アカウントは重要なので委任できない]** オプションを無効にします。

4. クラスターノード向けのサービス・プリンシパル名 (SPN) を登録します。SPN を登録するには、次のいずれかの方式を使用します。

a. Microsoft* から提供される `setspn.exe` ユーティリティーを使用する。例えば、ドメイン・コントローラー上で次のコマンドを実行します。

```
setspn.exe -A impi_hydra/<ホスト>:<ポート>/impi_hydra <ホスト>
```

説明:

<ホスト> はクラスターノード名。

<ポート> は Hydra ポートです。デフォルト値は 8679 です。これは、hydra サービスがデフォルト以外のポートを使用する場合にのみ変更します。

b. ノードに管理者アカウントでログインして、`hydra_service - register_spn` コマンドを実行します。

注意

MPI タスクの起動で問題が生じた場合、MPI タスクを起動したマシンを再起動します。代替手段として、Microsoft* 社が提供する `klist.exe` ユーティリティーが使用可能であれば、`klist purge` コマンドを実行します。

3.3. 環境変数

I_MPI_AUTH_METHOD

ユーザー認証方式を選択します。

構文

`I_MPI_AUTH_METHOD=<方式>`

引数

<方式>	認証方式を定義します。
`password`	パスワードベースの認証を使用します。これは、デフォルト値です。
`Delegate`	委任機能を持つドメインベースの認証を使用します。
`impersonate`	制限付きドメインベースの認証を使用します。これにより、リモートマシン上のファイルを開いたり、割り当てられたネットワーク・ドライブにアクセスできなくなります。

説明

この環境変数を使用して、目的とする認証方式を選択します。この環境変数が定義されていない場合、`mpiexec` はデフォルトでパスワードベースの認証方式を使用します。

注意

代替手段として、`mpiexec -delegate` または `mpiexec -impersonate` オプションを使用してデフォルトの認証方式を変更できます。

4. チューニング・リファレンス

インテル® MPI ライブラリーは、実行時のプログラムの振る舞いやパフォーマンスを調整する自動チューニング・ユーティリティーと多くの環境変数を提供しています。

4.1. mpitune ユーティリティーを使用

mpitune

クラスター構成やアプリケーションに関連するインテル® MPI ライブラリー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します。

構文

```
mpitune [ -a \"<アプリケーションのコマンドライン>\" ] [ -of <ファイル名> ] \
[ -t \"<テストコマンドライン>\" ] [-cm ] [ -d ] [ -D] \
[ -dl [d1[,d2...[,dN]]] ] [ -fl [f1[,f2…[,fN]]] ] \
[ -hf <ホストファイル> ] [ -h ] [ -hr {min:max|min:|:max} ] \
[ -i <カウント> ] [ -mr {min:max|min:|:max}] [ -od <出力ディレクトリー> ] \
[ -odr <出力ディレクトリー> ] [ -pr {min:max|min:|:max}] \
[ -sf [ファイルパス] ] [ -ss ] [ -s ] [ -td <ディレクトリーパス> ] \
[ -tl <時間>] [ -mh ] [ -os <opt1,...,optN> ] \
[ -oe <opt1,...,optN> ] [ -V ] [ -vi {パーセント} | -vix {X ファクター} ]\
[ - zb ] [ -t ] [ -so ] [ -ar \正規表現\" ] [ -trf <アプリケーションの出力ファイル> ]\
[ -m {base|optimized} ] [ -avd {min|max} ][ -pm {mpd|hydra} ]\
[ -co ] [ -sd ] [ -soc ]
```

または

```
mpitune [ --application\"<アプリケーションのコマンドライン>\" ] \
[ --output-file <ファイル名> ]\
[ --test \"<テストコマンドライン>\" ] [ --cluster-mode ] [ --debug ] \
[ --distinct ] [ --device-list [d1[,d2,… [,dN]]] ] \
[ --fabric-list [f1[,f2...[,fN]]] ]\
[ --host-file <ホストファイル> ] [ --help ] \
[ --host-range {min:max|min:|:max} ] [ --iterations<カウント> ] \
[ --message-range {min:max|min:|:max} ] \
[ --output-directory <出力ディレクトリー> ] \
[ --output-directory-results <出力ディレクトリー> ] \
[ --ppn-range {min:max|min:|:max} | --perhost-range {min:max|min:|:max} ] \
[ --session-file [ファイルパス] ] [ --show-session ] [ --silent ] \
[--temp-directory <ディレクトリーパス> ] [ --time-limit <時間> ] \
[ --master-host ] [ --options-set <opt1,...,optN> ] \
[ --options-exclude <opt1,...,optN> ] [ --version ] \
[ --valuable-improvement | --valuable-improvement-x {X ファクター} ]\
[ --zero-based ] [ --trace] [ --scheduler-only ] \
[ --application-regexp \」正規表現\" ] \
[ --test-regexp-file <アプリケーションの出力ファイル> ] [ --model {base|optimized} ] \
[ --application-value-direction {min|max} ] \
[ --process-manager {mpd|hydra} ] [ -co ] [ -sd ] [ -soc ]
```

引数

-a \"<アプリケーションのコマンドライン>\"| --application \"<アプリケーションのコマンドライン>\"	アプリケーション固有モードに切り替えます。バックスラッシュを含む完全なコマンドラインを入力します。
-of <ファイル名> | --output-file <ファイル名>	アプリケーション固有モードで生成される、アプリケーション構成ファイル名を指定します。デフォルトのファイル名は、app.conf です。
-t \"<テスト・コマンドライン>\" | --test \"<テスト・コマンドライン>\"	クラスター固有モードで指定するベンチマーク・プログラムを、インテル® MPI ベンチマークと入れ替えます。バックスラッシュを含む完全なコマンドラインを入力します。
-cm {exclusive|full} | --cluster-mode {exclusive|full}	クラスター利用モードを設定します。 • full - 最大数のタスクが実行されます。これはデフォルトのモードです。 • exclusive - クラスター上で 1 つのタスクのみが同時実行されます。
-d | --debug	デバッグ情報を表示します。
-D | --distinct	すべてのオプションを個別にチューニングします。この引数はクラスター固有モードでのみ有効です。
-dl [d1[,d2...[,dN]]] | --device-list[d1[,d2,… [,dN]]]	チューニングするデバイスを選択します。以前に設定したデバイスは無視されます。デフォルトでは、<インストール・ディレクトリー>\<アーキテクチャー>\etc\devices.xml ファイルにリストされるすべてのデバイスを使用します。
-fl [f1[,f2...[,fN]]] | --fabric-list [f1[,f2…[,fN]]]	チューニングするファブリックを選択します。以前に設定したファブリックは無視されます。デフォルトでは、<インストール・ディレクトリー>\<アーキテクチャー>\etc\fabrics.xml ファイルにリストされるすべてのファブリックを使用します。
-hf <ホストファイル> | --host-file <ホストファイル>	代替のホストファイル名を指定します。デフォルトは、mpd.hosts です。
-h| --help	ヘルプメッセージを表示します。
-hr {min:max|min:|:max} | --host-range {min:max|min:|:max}	テストに使用するホストの範囲を設定します。デフォルト値は 1 です。デフォルトの max 値は、mpd.hosts に定義されるホスト数か既存の MPD リング数です。min: または :max 形式は、必要に応じてデフォルト値を取ります。
-i <カウント> | --iterations <カウント>	各チューニング過程での実行回数を定義します。<カウント> 数を大きくすると、チューニングにかかる時間が増えますが、結果の精度は高まります。デフォルト値は 3 です。

-mr {min:max|min:|:max} | --message-range {min:max|min:|:max}	メッセージサイズの範囲を設定します。デフォルトの min 値は 0 です。デフォルトの max 値は 4194304 です (4MB)。デフォルトの値の単位はバイトです。次の形式で変更できます: 16kb、8mb または 2gb。min: または :max 形式は、必要に応じてデフォルト値を取ります。
-od <出力ディレクトリー> | --output-directory <出力ディレクトリー>	すべての出力ファイルへのディレクトリー名を指定します: ログファイル、セッションファイル、ローカルホスト・ファイル、レポートファイル。デフォルトは、カレント・ディレクトリーです。このディレクトリーは、すべてのホストがアクセスできる必要があります。
-odr <出力ディレクトリー> | --output-directory-results <出力ディレクトリー>	結果として生成される構成ファイルのディレクトリー名を指定します。デフォルトは、アプリケーション固有モードではカレント・ディレクトリーで、クラスター固有モードでは <インストール・ディレクトリー>\<アーキテクチャー>\etc です。もし、クラスター固有モードで、<インストール・ディレクトリー>\<アーキテクチャー>\etc が利用できない場合、カレント・ディレクトリーが選択されます。
-pr {min:max|min:|:max} | --ppn-range {min:max|min:|:max} | --perhost-range {min:max|min:|:max}	ホストごとの最大プロセッサー数を設定します。デフォルトの min 値は 1 です。デフォルトの max 値は、プロセッサーのコア数です。min: または :max 形式は、必要に応じてデフォルト値を取ります。
-sf [ファイルパス] | --session-file [ファイルパス]	[ファイルパス] セッションファイルに保存されている状態から、チューニングを再開します。
-ss | --show-session	セッションファイルと終了に関する情報を表示します。このオプションは、-sf オプションと併用する場合にのみ効果があります。
-s | --silent	すべての診断を抑制します。
-td <ディレクトリー・パス> | --temp-directory <ディレクトリー・パス>	一時データが使用するディレクトリー名を指定します。デフォルトで、インテル® MPI ライブラリーは、カレント・ディレクトリーの mpitunertemp フォルダーを使用します。このディレクトリーは、すべてのホストがアクセスできる必要があります。
-tl <時間> | --time-limit <時間>	mpitune を実行する制限時間を分単位で指定します。デフォルト値は 0 で、制限はありません。
-mh | --master-host	mpitune を単一ホストで実行します。
-os <opt1,...,optN>| --options-set <opt1,...,optN>	指定されたオプション値のみをチューニングします。

-oe <opt1,...,optN>| --options-exclude <opt1,...,optN>	チューニング・プロセスから指定されたインテル® MPI ライブラリーのオプション設定を除外します。
-V | --version	バージョン情報を表示します。
-vi {パーセント} |> --valuable-improvement {パーセント} -vix{X ファクター}| --valuable-improvement-x {X ファクター}	パフォーマンス向上の閾値を制御します。デフォルトのしきい値は 3% です。
-zb | --zero-based	チューニングの前に、すべてのオプションの基準として 0 を設定します。この引数はクラスター固有モードでのみ有効です。
-t | --trace	エラー番号やチューナーのトラックバックなどのエラー情報を表示します。
-so | --scheduler-only	実行すべきタスクのリストを作成し、タスクを表示して、実行を終了します。タスクを実行せず、スケジュールのみを行います。
-ar \正規表現\" | --application-regexp \正規表現\"	アプリケーションのパフォーマンス期待値を決定するため正規表現を使用します。この引数はクラスター固有モードでのみ有効です。正規表現の文字列は、mpitune が解析に使用する 1 つの数値グループのみを含めることができます。オペレーティング・システムの要求に応じて、引数の値を設定する際、シンボルにバックスラッシュを使用してください。
-trf <アプリケーションの出力ファイル> | --test-regexp-file <アプリケーションの出力ファイル>	正規表現の正当性を確認するため、テスト出力ファイルを使用します。-ar オプションを使用する場合、引数はクラスター固有モードにのみ有効です。
-m {base|optimized} | --model {base|optimized}	検索モデルを指定します。 • 古いモデルを使用するには base に設定します。 • 新しい高速な検索モデルを使用するには、optimized に設定します。これは、デフォルト値です。
-avd {min|max} | --application-value-direction {min|max}	値に最適化の方向性を指定します。 • 下位が良好である場合、min に設定します。例えば、実行時間を最適化する場合この値を使用します。 • 上位が良好である場合、max に設定します。例えば、解決率を最適化する場合この値を使用します。

-pm {mpd|hydra} | --process-manager {mpd|hydra}	ベンチマークの実行に使用するプロセス管理を指定します。デフォルトは hydra です。
-co | --collectives-only	集合操作のみをチューニングします。
-sd | --save-defaults	インテル® MPI ライブラリーのデフォルト値を保存するため mpitune を使用します。
-soc | --skip-options- check	コマンドライン・オプションを確認するかどうか指定します。

廃止されたオプション

廃止されたオプション	新しいオプション
--outdir	-od |--output-directory
--verbose	-d | --debug
--file	-hf |--host-file
--logs	-lf |--log-file
--app	-a |--application

説明

特定のクラスターやアプリケーション向けの最適な設定が含まれる、インテル® MPI ライブラリーの設定ファイルを作成するため、mpitune ユーティリティーを使用します。mpiexec でジョブを起動する際に、-tune オプションを使用してこのファイルを再利用することができます。以前の mpitune セッションの設定ファイルが存在する場合、mpitune は実行を開始する前に既存のファイルのコピーを作成します。

mpitune ユーティリティーは、次の 2 つのモードで操作します。

- クラスター固有。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、インテル® MPI Benchmarks やユーザーから提供されるベンチマーク・プログラムを使用して、クラスター環境を評価します。このオプションはデフォルトで使用されます。
- アプリケーション固有。特定のアプリケーション向けにインテル® MPI ライブラリーの最適な設定を見つけるため、MPI アプリケーションの性能を評価します。アプリケーションのチューニングには、--application コマンドライン・オプションを指定します。

4.1.1. クラスター固有のチューニング

クラスターをチューニングして最適な設定を見つけるため、インテル® MPI ライブラリーのインストール後 mpitune ユーティリティーを実行し、すべてのクラスターを再構成します (プロセッサーやメモリーのアップグレード、ネットワークの再構成、など)。設定リストを取得するには、インテル® MPI ライブラリーをインストールしたアカウントでユーティリティーを実行するか、--output-directory オプションでチューナーのデータ・ディレクトリーと --output-directory-results オプションで結果の出力ディレクトリーを指定してユーティリティーを実行します。

<インストール・ディレクトリー>\<アーキテクチャー>\etc ディレクトリーに設定ファイルが存在する場合、mpiexec に -tune オプションを指定すると記録されているインテル® MPI ライブラリーの構成設定が自動的に使用されます。

次に例を示します。

- インテル® MPI Benchmarks によって使用される .\mpd.hosts ファイルに含まれるクラスターホスト向けに構成の設定を収集します。

  ```
  >mpitune
  ```

- クラスター上で実行する場合、記録された設定ファイルを使用します。

  ```
  >mpiexec-tune -n 32 .\myprog
  ```

ジョブランチャーは、通信ファブリック、ホストとプロセス数などの実行条件に基づいて、適切な構成のセットを検出します。<インストール・ディレクトリー>\<アーキテクチャー>\etc への書き込み権限がある場合、生成されたすべてのファイルはこのディレクトリーに保存されます。書き込み権限がない場合、カレント・ワーキング・ディレクトリーが使用されます。

注意

クラスター固有モードで -tune オプションを使用する場合 (チューニング設定ファイル名を指定せず)、明示的に通信デバイスやファブリック、ノードごとのプロセス数、およびプロセス数の合計を指定する必要があります。次に例を示します。

```
> mpiexec -tune -genv I_MPI_FABRICS shm:dapl -ppn 8 -n 32
```

デフォルトのベンチマークを置き換え

このチューニング機能は、クラスター固有モードの拡張であり、チューニングに使用するベンチマーク・アプリケーションを指定することができます。

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。

次に例を示します。

1. 要求されるベンチマーク・プログラムによって使用される .\mpd.hosts ファイルに含まれるクラスターホスト向けに構成の設定を収集します。

   ```
   >mpitune --test \"benchmark -param1 -param2\"
   ```

2. クラスター上で実行する場合、記録された設定ファイルを使用します。

   ```
   >mpiexec -tune -n 32 .\myprog
   ```

コマンドラインで指定することで、任意のアプリケーションのチューニングを実行できます。パフォーマンスは、指定されたアプリケーションの逆実行時間として計測されます。全体のチューニング時間を短縮するため、設定 (ファブリック、ランクの配置など) を適用可能な、最も典型的なアプリケーションのワークロードを使用します。

注意

アプリケーション固有モードでは、同様のコマンドラインと環境を使用して最も適切なチューニング結果を得ることができます。

次に例を示します。

指定されたアプリケーションの構成設定を収集します。

```
>mpitune --application \"mpiexec-n 32 .\myprog\" -of .\myprog.conf
```

アプリケーションを実行する場合、記録された設定ファイルを使用します。

```
>mpiexec-tune .\myprog.conf -n 32 .\myprog
```

デフォルトのチューニング規則に基づき、自動化されたチューニング・ユーティリティーは、アプリケーションの実行時間を最小化するため、すべてのライブラリーを構成するパラメーターを評価します。デフォルトでは、生成されたファイルはすべてカレント・ワーキング・ディレクトリーに保存されます。

アプリケーションの設定ファイルには、そのアプリケーションと構成のみに最適なインテル® MPI ライブラリーのパラメーターが含まれます。インテル® MPI ライブラリーを同じアプリケーションの異なる構成 (ホスト数、ワークロードなど) 向けにチューニングする場合、対象の構成で自動チューニング・ユーティリティーを再実行してください。

注意

デフォルトでは、自動チューニング・ユーティリティーは既存のアプリケーション向けの設定ファイルを上書きします。アプリケーションの設定ファイルを保持したい場合は、異なる名前で保存し、必要なときにすぐに選択できるように、名前を付ける必要があります。

4.1.2. チューニング・ユーティリティーの出力

チューニング・プロセスが完了すると、インテル® MPI ライブラリーのチューニング・ユーティリティーは、次の形式で選択された値を記録します。

```
-genv I_MPI_DYNAMIC_CONNECTION 1
-genv I_MPI_ADJUST_REDUCE 1:0-8
```

インテル® MPI ライブラリーのチューニング・ユーティリティーは、調査した差がノイズレベル (1%) である場合、アプリケーションに影響しない環境変数を無視します。この場合、ユーティリティーは、環境変数を設定せずデフォルトのライブラリーのヒューリスティックを保持します。

実行するたびにアプリケーションのパフォーマンスが変動する場合、インテル® MPI ライブラリーのチューニング・ユーティリティーは、同じ条件下で同じ環境変数に異なる値を選択することがあります。決定精度を向上するため、`--iterations` コマンドライン・オプションを使用してそれぞれのテスト実行の反復回数を増やします。デフォルトの反復回数は 3 です。

4.2. プロセスのピニング (固定)

MPI プロセスをノード内のプロセッサーにピニング (固定) し、望ましくないプロセスのマイグレーションを避けるため、このオプションを使用します。この機能は、オペレーティング・システムがカーネル・インターフェイスを提供する場合に利用できます。

4.2.1. プロセスピニングのデフォルト設定

環境変数にプロセスピニングが指定されていない場合、次のデフォルト設定が使用されます。この設定の詳細は、「環境変数」と「OpenMP* API との相互利用」をご覧ください。

- `I_MPI_PIN=on`
- `I_MPI_PIN_MODE=pm`
- `I_MPI_PIN_RESPECT_CPUSET=on`
- `I_MPI_PIN_RESPECT_HCA=on`
- `I_MPI_PIN_CELL=unit`
- `I_MPI_PIN_DOMAIN=auto:compact`
- `I_MPI_PIN_ORDER=compact`

4.2.2. プロセッサーの識別

システムの論理プロセッサーを特定するため次のスキームが適用されます。

- システム定義の論理列挙値
- トリプレット (パッケージ/ソケット、コア、スレッド) を介した 3 レベルの階層型識別に基づくトポロジーの列挙

論理 CPU 番号は、カーネルのアフィニティー・ビットマスクでその CPU ビットに対応する位置として定義されます。論理 CPU 番号を特定するため、インテル® MPI ライブラリーで提供される cpuinfo ユーティリティーを使用してください。

3 レベルの階層構造による識別は、プロセッサーの場所とその並びに関連する情報を提供するトリプレットを採用しています。トリプレットは階層構造です (パッケージ、コア、スレッド)。

2 ソケット、4 コア (ソケットあたり 2 コア)、8 論理プロセッサー (コアあたり 2 プロセッサー)におけるプロセッサー番号の例をご覧ください。

注意

論理とトポロジーの列挙によるプロセッサーは、同一ではありません。

表 3.2-1 論理一覧

0	4	1	5	2	6	3	7

表 3.2-2 階層レベル

ソケット	0	0	0	0	1	1	1	1
コア	0	0	1	1	0	0	1	1
スレッド	0	1	0	1	0	1	0	1

表 3.2-3 トポロジー一覧

0	1	2	3	4	5	6	7

cpuinfo ユーティリティーを使用して、論理とトポロジー列挙の間の対応関係を特定します。詳細は、「プロセッサー情報ユーティリティー」をご覧ください。

4.2.3. 環境変数

I_MPI_PIN

プロセスのピニングを on/off にします。

構文

```
I_MPI_PIN=<引数>
```

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	プロセスのピニングを有効にします。これは、デフォルト値です。
disable | no | off | 0	プロセスのピニングを無効にします。

説明

インテル® MPI ライブラリーのプロセスピニング機能を制御するには、この環境変数を設定します。

I_MPI_PIN_PROCESSOR_LIST (I_MPI_PIN_PROCS)

プロセッサー・サブセットとこのサブセット内の MPI プロセスのマッピング規則を定義します。

構文

`I_MPI_PIN_PROCESSOR_LIST=<値>`

<値> には以下の構文があります。

1. <プロセッサー・リスト>
2. `[<プロセッサー・リスト>][:[grain=<グレイン>][,shift=<シフト>][,preoffset=<前オフセット>][,postoffset=<後オフセット>]`
3. `[<プロセッサー・リスト>][:map=<マップ>]`

次の段落でこれらの構文の詳しい値を説明します。

廃止された構文

`I_MPI_PIN_PROCS=<プロセッサー・リスト>`

注意

`postoffset` キーワードは `offset` をエリアスします。

注意

ピニング手順の 2 番目の形式には、次の 3 つの手順があります。

1. `preoffset*grain` 値で、ソース・プロセッサー・リストを循環シフトします。
2. `shift*grain` 値 で最初のステップから派生したリストをラウンドロビンでシフトします。
3. `postoffset*grain` 値 で 2 番目のステップから派生したリストを循環シフトします。

注意

`grain`、`shift`、`preoffset` および `postoffset` パラメーターは、統一された定義スタイルを持ちます。

この環境変数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。

構文

`I_MPI_PIN_PROCESSOR_LIST=<プロセッサー・リスト>`

引数

<プロセッサー・リスト>	論理プロセッサー番号および (または)、プロセッサーの範囲をカンマで区切ったリスト。i 番目のランクのプロセスは、リスト内の i 番目のプロセッサーにピニング (固定) されます。番号は、ノード内のプロセッサー数を越えてはいけません。
<l>	論理番号 <l> のプロセッサー。
<l>-<m>	論理番号 <l> から <m> の範囲のプロセッサー。
<k>,<l>-<m>	論理番号 <k> と <l> から <m> までのプロセッサー。

構文

`I_MPI_PIN_PROCESSOR_LIST=[<プロセッサー・セット>][:[grain=<粒度>][,shift=<シフト>][,preoffset=<前オフセット>][,postoffset=<後オフセット>]`

引数

<プロセッサー・セット>	トポロジーによる算出法に基づいて、プロセッサー・サブセットを指定します。デフォルト値は `allcores` です。
`all`	すべての論理プロセッサー。ノード上の CPU 番号を定義するためにこのサブセットを指定します。
`allcores`	すべてのコア (物理 CPU)ノード上のコア番号を定義するためにこのサブセットを指定します。これは、デフォルト値です。 インテル® ハイパースレッディング・テクノロジーが無効の場合、`allcores` は `all` と等価です。
`allsocks`	すべてのパッケージ/ソケット。ノード上のソケット番号を定義するためにこのサブセットを指定します。

<粒度>	定義された <プロセッサー・セット> に、セルをピニング (固定)する粒度を指定します。最小 <粒度> 値は、<プロセッサー・セット> の単一要素です。最大 <粒度> 値は、ソケットの <プロセッサー・セット> 要素の数です。<粒度> 値は、<プロセッサー・セット> 値の倍数でなければいけません。そうでない場合、最小 <粒度> 値が想定されます。デフォルトは、最小 <粒度> 値です。
<シフト>	<プロセッサー・セット> のセルをラウンドロビン・スケジューリングする際のシフトの粒度を指定します。<シフト> は、定義された <粒度> ユニットを基準とします。<シフト> 値は、正の整数でなければなりません。そうでない場合、シフトは行われません。デフォルトはシフトなしで、1 つインクリメントするのに相当します。

<前オフセット>	<前オフセット> 値をラウンドロビン・シフトする前に定義された、プロセッサー・サブセット <プロセッサー・セット> の巡回シフトを指定します。値は、定義された <粒度> ユニットを基準とします。<前オフセット> 値は、正の整数でなければなりません。そうでない場合、シフトは行われません。デフォルトは "no shift" です。
<後オフセット>	<後オフセット> 値をラウンドロビン・シフトした後に誘導された、プロセッサー・サブセット <プロセッサー・セット> の巡回シフトを指定します。値は、定義された <粒度> ユニットを基準とします。<後オフセット> 値は、正の整数でなければなりません。そうでない場合、シフトは行われません。デフォルトは "no shift" です。

次の表は、<粒度>、<シフト>、<前オフセット> および <後オフセット>の値を示します。

<n>	対応するパラメーターの明示的な値を指定します。<n> は、正の整数値です。
fine	対応するパラメーターの最小値を指定します。
core	1 つのコアに含まれるパラメーター・ユニットと同じ数のパラメーター値を指定します。
cache1	L1 キャッシュを共有するパラメーター・ユニットと同じ数のパラメーター値を指定します。
cache2	L2 キャッシュを共有するパラメーター・ユニットと同じ数のパラメーター値を指定します。
cache3	L3 キャッシュを共有するパラメーター・ユニットと同じ数のパラメーター値を指定します。
cache	cache1、cache2 および cache3 中の最大値。
socket | sock	1 つの物理パッケージ/ソケットに含まれるパラメーター・ユニットと同じ数のパラメーター値を指定します。
half | mid	socket/2 と等しいパラメーター値を指定します。
third	socket/3 と等しいパラメーター値を指定します。
quarter	socket/4 と等しいパラメーター値を指定します。
octavo	socket/8 と等しいパラメーター値を指定します。

構文

I_MPI_PIN_PROCESSOR_LIST=[<プロセッサー・セット>][:map=<マップ>]

引数

<マップ>	プロセスの配置に使用するマッピングのパターン。
bunch	プロセスをソケット上で可能な限り隣接してマップします。
scatter	リソース (FSB、キャッシュおよびコア) を共有しないように、プロセスは可能な限り離れてマップされます。
spread	共有リソースを共有しないように、プロセスは可能な限り連続してマップされます。

説明

プロセッサー配置を設定するには、I_MPI_PIN_PROCESSOR_LIST 環境変数を設定します。別シェルとの競合を避けるため、環境変数の値は引用符で囲む必要があります。

注意

この環境変数は、I_MPI_PIN が有効なときにのみ効果があります。

I_MPI_PIN_PROCESSOR_LIST 環境変数には次の異なる構文があります。

- 明示的なプロセッサー・リスト。論理プロセッサー番号が定義されるカンマで区切られたリスト。プロセスの相対ノードランクは、i 番目のプロセスは i 番目のリスト番号へピニングするなど、プロセッサー・リストへのインデックスとなります。CPU 上で任意のプロセス配置を定義することを許可します。

  例えば、I_MPI_PIN_PROCESSOR_LIST=p0,p1,p2,..., pn というプロセスマッピングは、次のように展開されます。

ノードのランク	0	1	2	...	n-1	N
論理 CPU	p0	p1	p2	...	pn-1	Pn

- ゲイン/シフト/オフセットのマッピング。この方式は、<シフト>*<粒度> に等しいステップと、末端が <オフセット>*<粒度> の単一シフトによる、プロセッサー・リストに沿って定義された粒度の巡回シフトを行います。このシフト動作は、<シフト> 回繰り返されます。

  例: 粒度 = 2 論理プロセッサー、シフト = 3 粒度、オフセット = 0

凡例:

灰色 - MPI プロセスの粒度

A) 赤色 - 最初のパスで選択されたプロセッサー粒度

B) 水色 - 2 番目のパスで選択されたプロセッサー粒度

C) 緑色 - 最後の 3 番目のパスで選択されたプロセッサー粒度

D) MPI ランクによる並びの最終的なマップテーブル

A)

0 1			2 3			...	2n-2 2n-1		
0 1	2 3	4 5	6 7	8 9	10 11	...	6n-6 6n-5	6n-4 6n-3	6n-2 6n-1

B)

0 1	2n 2n+1		2 3	2n+2 2n+3		…	2n-2 2n-1	4n-2 4n-1	
0 1	2 3	4 5	6 7	8 9	10 11	…	6n-6 6n-5	6n-4 6n-3	6n-2 6n-1

C)

0 1	2n 2n+1	4n 4n+1	2 3	2n+2 2n+3	4n+2 4n+3	…	2n-2 2n-1	4n-2 4n-1	6n-2 6n-1
0 1	2 3	4 5	6 7	8 9	10 11	…	6n-6 6n-5	6n-4 6n-3	6n-2 6n-1

D)

0 1	2 3	…	2n-2 2n-1	2n 2n+1	2n+2 2n+3	…	4n-2 4n-1	4n 4n+1	4n+2 4n+3	…	6n-2 6n-1
0 1	6 7	…	6n-6 6n-5	2 3	8 9	…	6n-4 6n-3	4 5	10 11	…	6n-2 6n-1

- 事前定義マッピング。この場合、大部分のプロセスのピニングは、実行時に選択できるキーワードとして定義されます。bunch と scatter の 2 つのシナリオがあります。

bunch シナリオでは、プロセスは可能な限り近いソケットにマッピングされます。このマッピングは、部分的なプロセッサー負荷に適しています。この場合、プロセス数はプロセッサー数よりも少なくなります。

scatter シナリオでは、プロセスはリソース (FSB、キャッシュおよびコア) を共有しないように可能な限り離れてにマッピングされます。

例えば、2 ソケット、ソケットごとに 4 コア、コアあたり 1 論理 CPU では、2 つのコアごとにキャッシュを共有します。

凡例:

灰色 - MPI プロセス

水色 - 最初のソケットのプロセッサー

緑色 - 2 番目のソケットのプロセッサー

同じ色は、キャッシュを共有するプロセッサーのペアを示します。

0	1	2	
0	1	2	3

3	4		
4	5	6	7

5 プロセスでの bunch シナリオ

0	4	2	6
0	1	2	3

1	5	3	7
4	5	6	7

すべてを使用する scatter シナリオ

例

1. ノード全体でプロセスを CPU0 と CPU3 にピニングするには、次のコマンドを使用します。

```
> mpiexec.exe -genv I_MPI_PIN_PROCESSOR_LIST 0,3 \
-n <プロセス数> <実行形式>
```

2. 各ノードで個別に異なる CPU にプロセスをピニング (host1 で CPU0 と CPU3、host2 で CPU0、CPU1 および CPU3) するには、次のコマンドを使用します。

```
> mpiexec.exe -host host1 -env I_MPI_PIN_PROCESSOR_LIST 0,3 \
-n <プロセス数> <実行形式> :\
-host host2 -env I_MPI_PIN_PROCESSOR_LIST 1,2,3 \
-n <プロセス数> <実行形式>
```

3. プロセスのピニングに関する拡張デバッグ情報を表示するには、次のコマンドを使用します。

```
> mpiexec.exe -genv I_MPI_DEBUG 4 -m -host host1 \
-env I_MPI_PIN_PROCESSOR_LIST 0,3 -n <プロセス数> <実行形式> :\
-host host2 -env I_MPI_PIN_PROCESSOR_LIST 1,2,3 \
-n <プロセス数> <実行形式>
```

注意

プロセス数がピニングする CPU 数よりも大きい場合、プロセスリストはプロセッサー・リストの先頭にラップアラウンドします。

I_MPI_PIN_CELL

ピニングの解像度を定義します。I_MPI_PIN_CELL は、MPI プロセスを実行する際に、最小のプロセッサー・セルを指定します。

構文

I_MPI_PIN_CELL=<セル>

引数

<セル>	粒度の解像度を指定します。
unit	基本プロセッサー・ユニット (論理 CPU)。
core	物理プロセッサー・コア。

説明

この環境変数を設定して、プロセスが実行される際に使用するプロセッサー・サブセットを定義します。2 つのシナリオを選択できます。

- ノード内のすべての利用可能な CPU (unit)
- ノード内のすべての利用可能なコア (core)

この環境変数は、どちらのピニングにも影響します。

- I_MPI_PIN_PROCESSOR_LIST 環境変数を介した 1 対 1 のピニング
- I_MPI_PIN_DOMAIN 環境変数を介した 1 対多数のピニング

デフォルト値は以下のようになります。

- I_MPI_PIN_DOMAIN を使用する場合、セルの粒度は unit です。
- I_MPI_PIN_PROCESSOR_LIST を使用する場合、次の規則が適用されます。
  - プロセス数がコア数よりも多い場合、セルの粒度は unit です。
  - プロセス数がコア数以下の場合、セルの粒度は core です。

注意

システムでインテル® ハイパースレッディング・テクノロジーの有効/無効を切り替えても core 値は影響を受けません。

4.2.4. OpenMP* API との相互利用

I_MPI_PIN_DOMAIN

インテル® MPI ライブラリーは、MPI/OpenMP* ハイブリッド・アプリケーションのプロセスピニングを制御する追加の環境変数を提供します。この環境変数は、ノード上の論理プロセッサーがオーバーラップしないサブセット (ドメイン) を定義し、ドメインあたり 1 つの MPI プロセスにすることで、ドメインへ MPI プロセスをバインドするルールを設定することができます。次の図を参照してください。

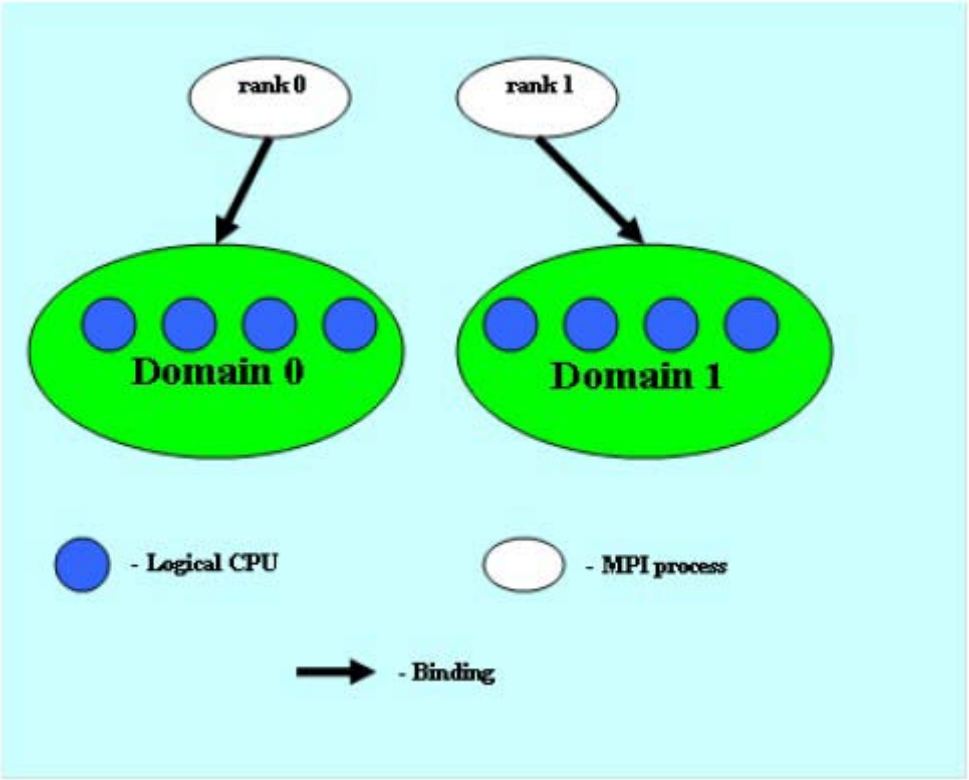


図 4.2-1 ドメインの例

各 MPI プロセスは、対応するドメイン内で実行する子スレッドを作成できます。プロセススレッドは、ドメイン内の論理プロセッサーからほかの論理プロセッサーへ自由に移行できます。

I_MPI_PIN_DOMAIN 環境変数が定義されている場合、I_MPI_PIN_PROCESSOR_LIST 環境変数の設定は無視されます。

I_MPI_PIN_DOMAIN 環境変数が定義されない場合、MPI プロセスは I_MPI_PIN_PROCESSOR_LIST 環境変数の値に従ってピニングされます。

I_MPI_PIN_DOMAIN 環境変数には、次の構文があります。

- マルチコア用語 <マルチコアの形態> を介したドメイン定義
- ドメインサイズとドメイン・メンバーリスト <サイズ>[:<レイアウト>] を介したドメイン定義
- ビットマスク <マスクリスト> を介したドメイン定義

次の表で構文形式を説明します。

マルチコアの形態

I_MPI_PIN_DOMAIN=<マルチコアの形態>

<マルチコアの形態>	マルチコア用語を介してドメインを定義します。
core	各ドメインは、特定のコアを共有する論理プロセッサーで構成されます。ノードのドメイン数はノードのコア数と等しくなります。
socket | sock	各ドメインは、特定のソケットを共有する論理プロセッサーで構成されます。ノードのドメイン数はノードのソケット数と等しくなります。これは、推奨値です。
node	ノード上のすべての論理プロセッサーは、単一のドメインに配置されます。
cache1	特定のレベル 1 キャッシュを共有する論理プロセッサーは、単一ドメインに配置されます。
cache2	特定のレベル 2 キャッシュを共有する論理プロセッサーは、単一ドメインに配置されます。
cache3	特定のレベル 3 キャッシュを共有する論理プロセッサーは、単一ドメインに配置されます。
cache	cache1、cache2 および cache3 中の最大のドメインが選択されます。

明示的な形態

I_MPI_PIN_DOMAIN=<サイズ>[:<レイアウト>]

<サイズ>	各ドメインの論理プロセッサー数を定義します (ドメインサイズ)。
omp	ドメインサイズは、OMP_NUM_THREADS 環境変数の値を同じです。OMP_NUM_THREADS 環境変数が定義されていない場合、各ノードは個別のドメインとして扱われます。
auto	ドメインサイズは、サイズ=#cpu/#proc の式で定義されます。 ここで、#cpu は、ノード上の論理プロセッサー数で、#proc は、ノード上の MPI プロセス数です。
<n>	正の十進数 <n> でドメインサイズを指定します。

<レイアウト>	ドメインメンバーの順番。デフォルト値は compact です。
platform	ドメインのメンバーは、BIOS で定義される番号付け (プラットフォーム固有の番号) に従って並べられます。
compact	ドメインのメンバーは、リソース (コア、キャッシュ、ソケットなど) を共有するように可能な限り近く並べられますこれは、デフォルト値です。
scatter	ドメインのメンバーは、リソース (コア、キャッシュ、ソケットなど) を共有しないように可能な限り離れて並べられます。

明示的なドメインマスク

`I_MPI_PIN_DOMAIN=<マスクリスト>`

<マスクリスト>	カンマで区切られた 16 進数でドメインを定義します (ドメインマスク)。
`[m1,...,mn]`	<マスクリスト> の各 m_i は個別のドメインを定義する 16 進数のビットマスクです。次の規則が適用されます: 対応する m_i ビットが `1` であれば、`i` 番目の論理プロセッサーは、ドメインに含まれます。 その他のプロセッサーは、異なるドメインに配置されます。BIOS のナンバリングが使用されます。 **注意** <マスクリスト> の設定が正しく解釈されることを確実にするため、<マスクリスト> で指定するドメインを括弧で囲みます。次に例を示します。 `I_MPI_PIN_DOMAIN=[0x55,0xaa]`

注意

これらのオプション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

ドメイン内で OpenMP* プロセスやスレッドをピニングするには、OpenMP* でサポートされる機能 (インテル® コンパイラーの `KMP_AFFINITY` 環境変数など) を使用します。

SMP ノードのモデルを以下に示します。

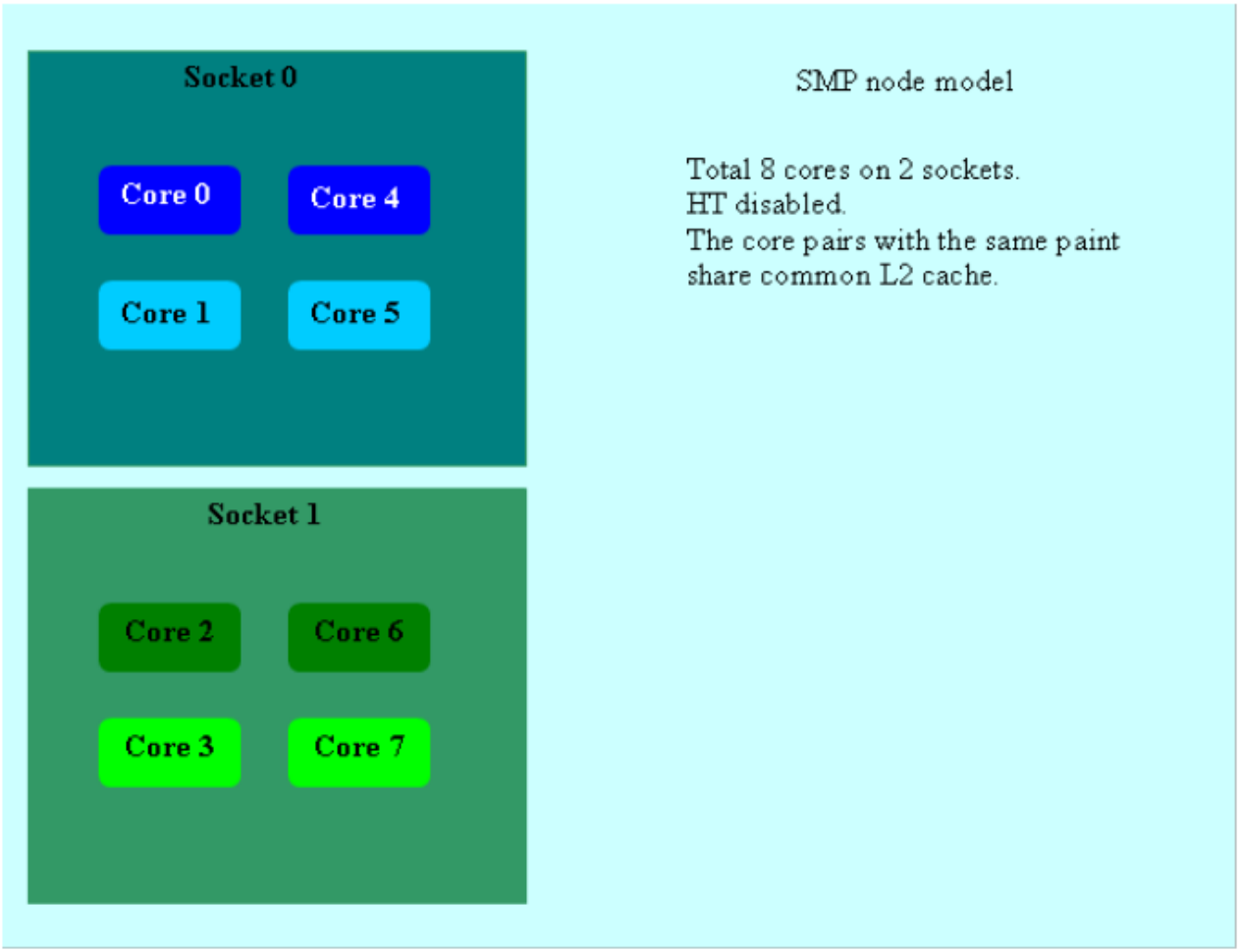


図 4.2-2 ノードのモデル

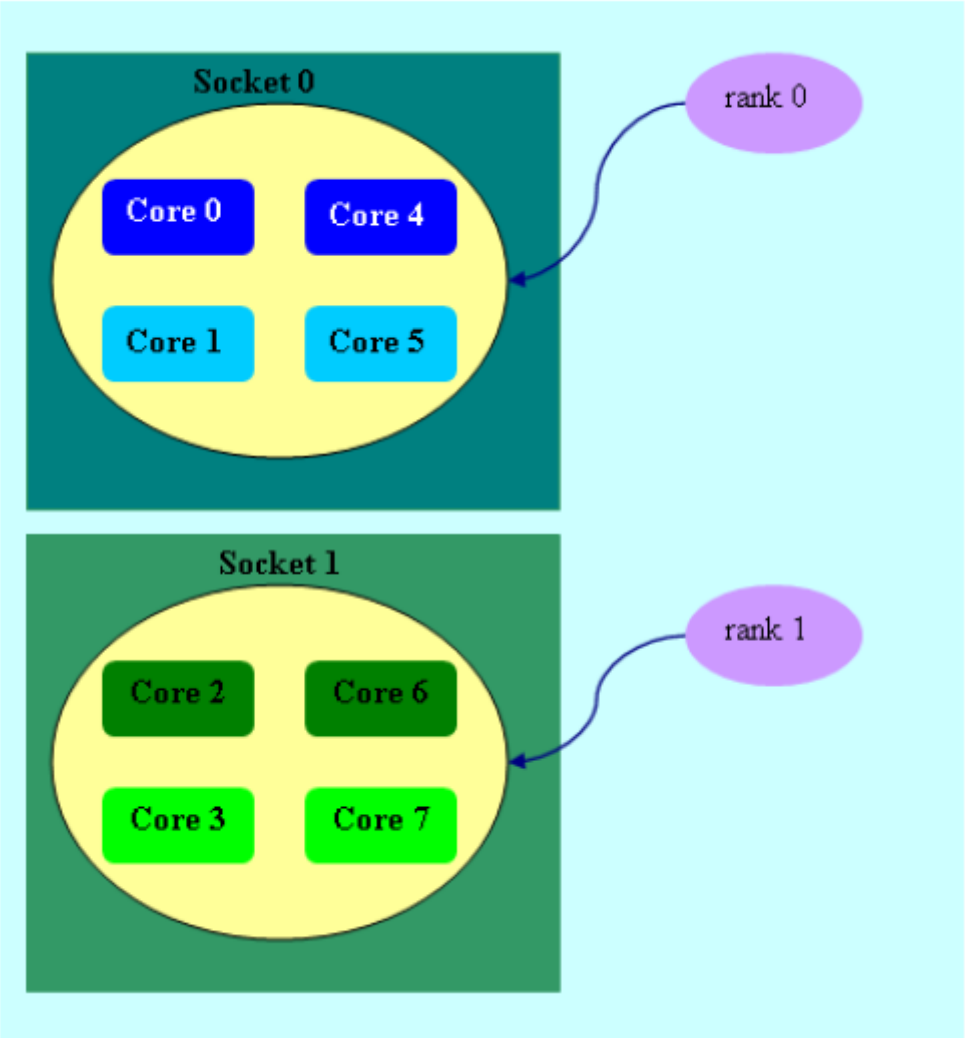


図 4.2-3 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN socket test.exe`

図 4.2-3 では、ソケット数に応じて 2 つのドメインが定義されます。プロセスランク 0 は、0 番目のソケットのすべてのコアに移行できます。プロセスランク 1 は、1 番目のソケットのすべてのコアに移行できます。

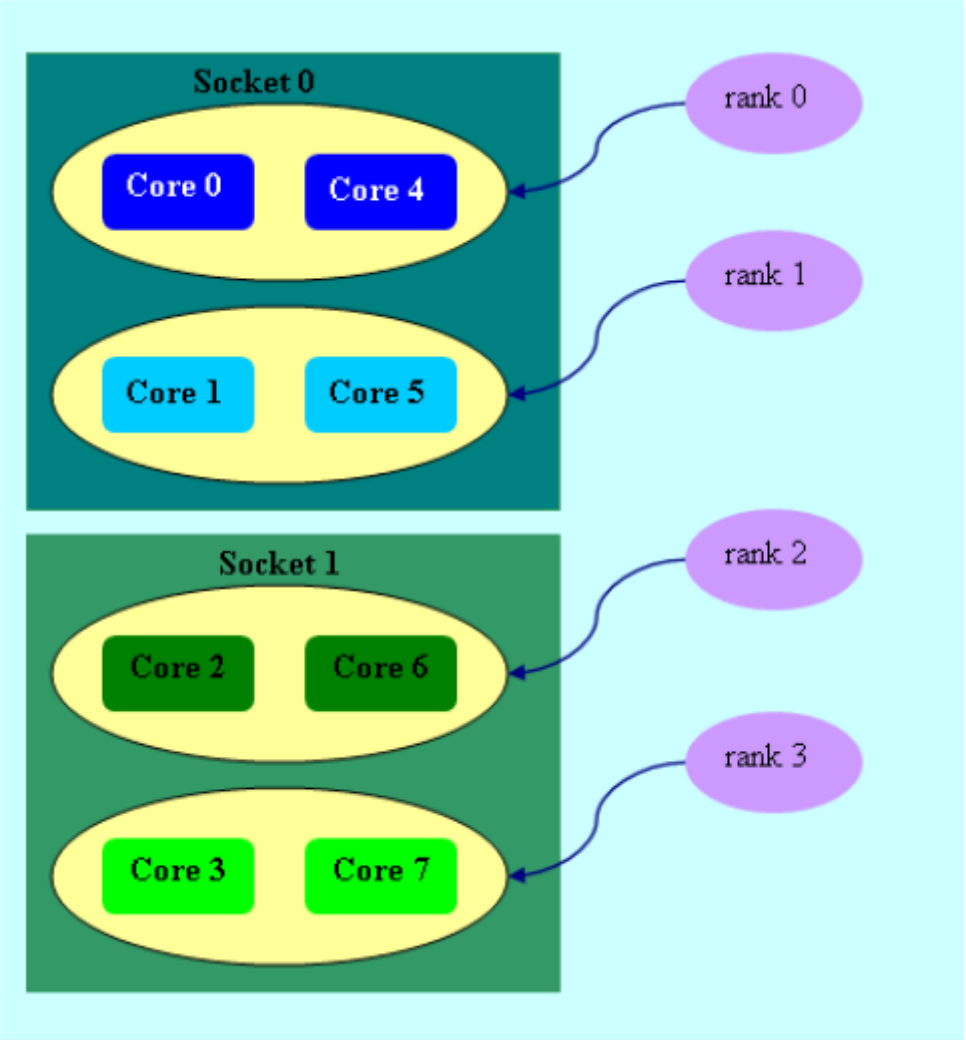


図 4.2-4 `mpiexec -n 4 -env I_MPI_PIN_DOMAIN cache2 test.exe`

図 4.2-4 では、共有 L2 キャッシュの量に応じて 4 つのドメインが定義されます。プロセスランク 0 は、L2 キャッシュを共有するコア {0,4} で実行されます。プロセスランク 1 は、同様に L2 キャッシュを共有するコア {1,5} で実行されます。

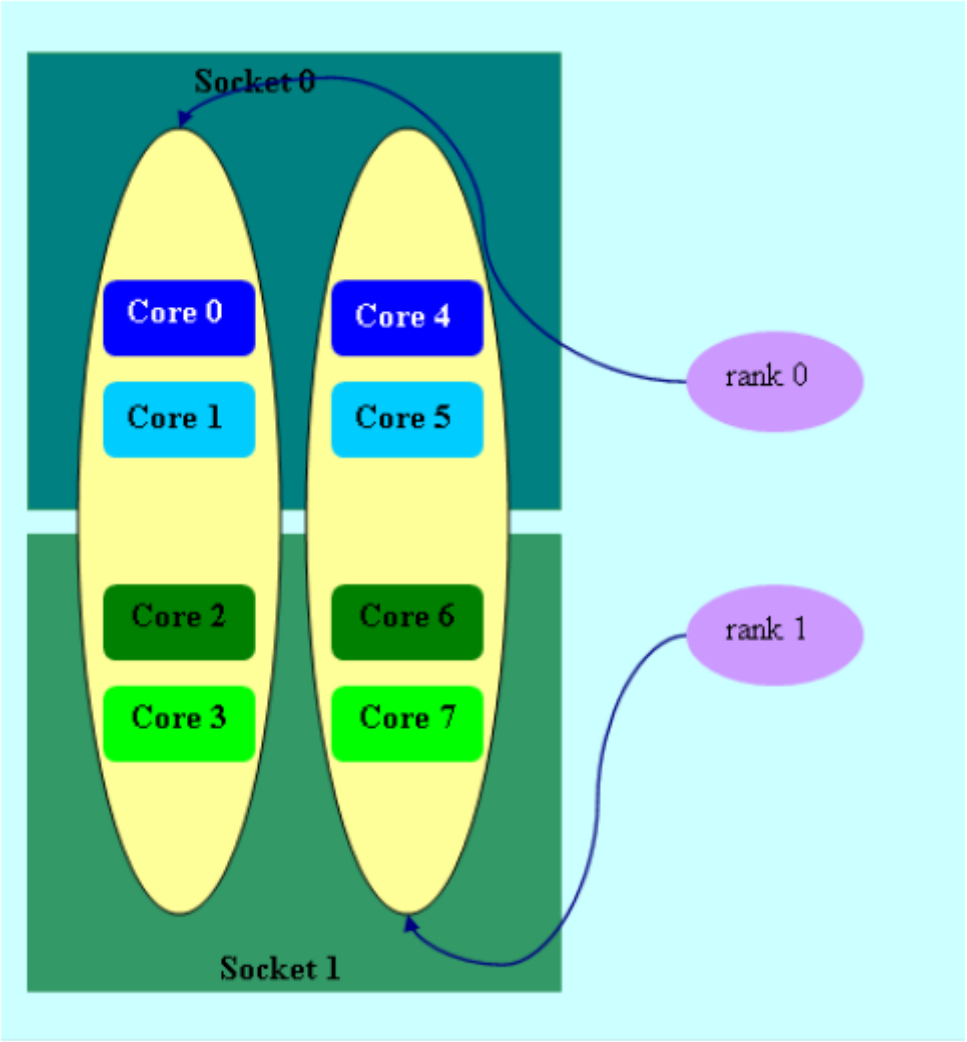


図 4.2-5 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN 4:platform test.exe`

図 4.2-5 では、サイズ=4 の 2 つのドメインが定義されます。最初のドメインはコア {0,1,2,3} を含み、2 番目のドメインはコア {4,5,6,7} を含みます。`platform` オプションで定義されるドメインメンバー (コア) は、連続する番号になります。

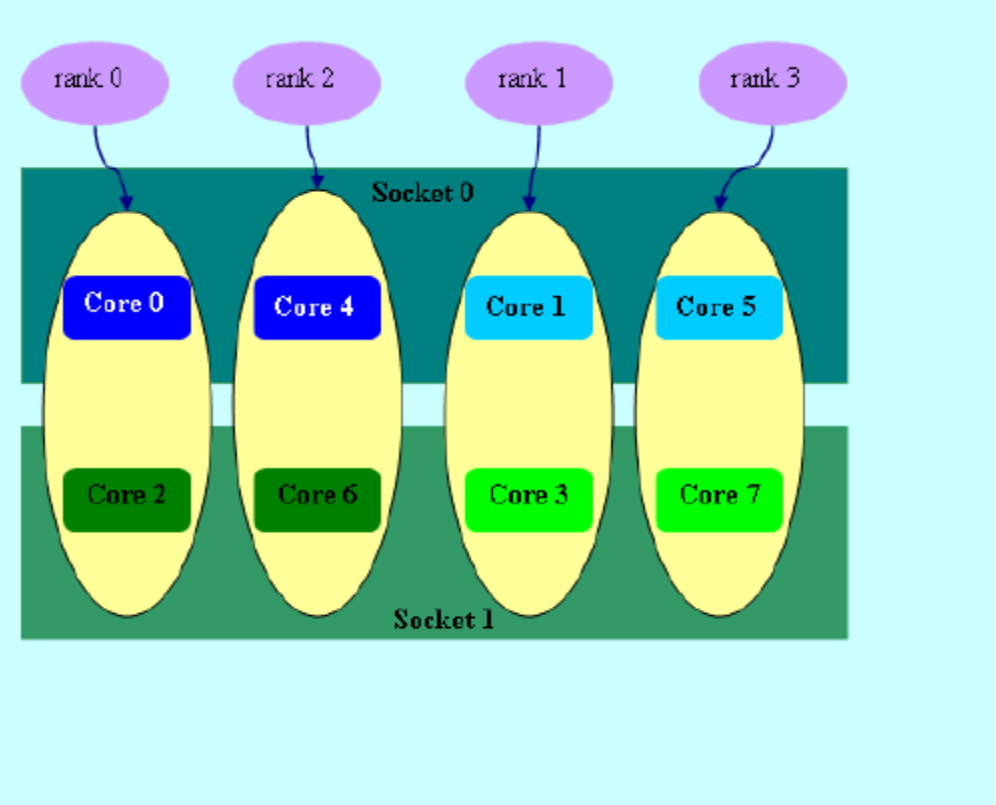


図 4.2-6 `mpiexec -n 4 -env I_MPI_PIN_DOMAIN auto:scatter test.exe`

図 4.2-6 では、ドメイン サイズ=2 (CPU 数 = 8 / プロセス数 = 4 で定義される)、`scatter` レイアウト。4 つのドメイン {0,2}、{1,3}、{4,6}、{5,7} が定義されます。ドメインのメンバーは、いかなるリソースも共有しません。

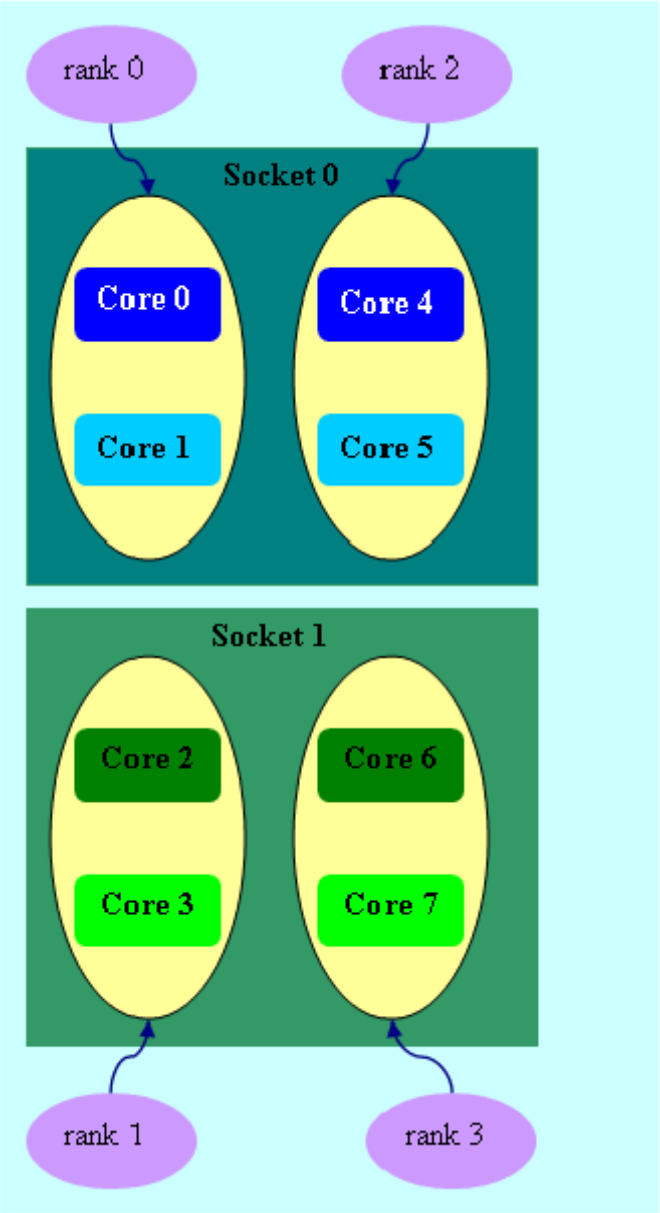


図 4.2-7 setenv OMP_NUM_THREADS=2

```
mpiexec -n 4 -env I_MPI_PIN_DOMAIN omp:platform test.exe
```

図 4.2-7 では、ドメイン サイズ=2 (OMP_NUM_THREADS=2 で定義される)、platform レイアウト。4 つのドメイン {0,1}、{2,3}、{4,5}、{6,7} が定義されます。ドメインメンバー (コア) は、連続する番号になります。

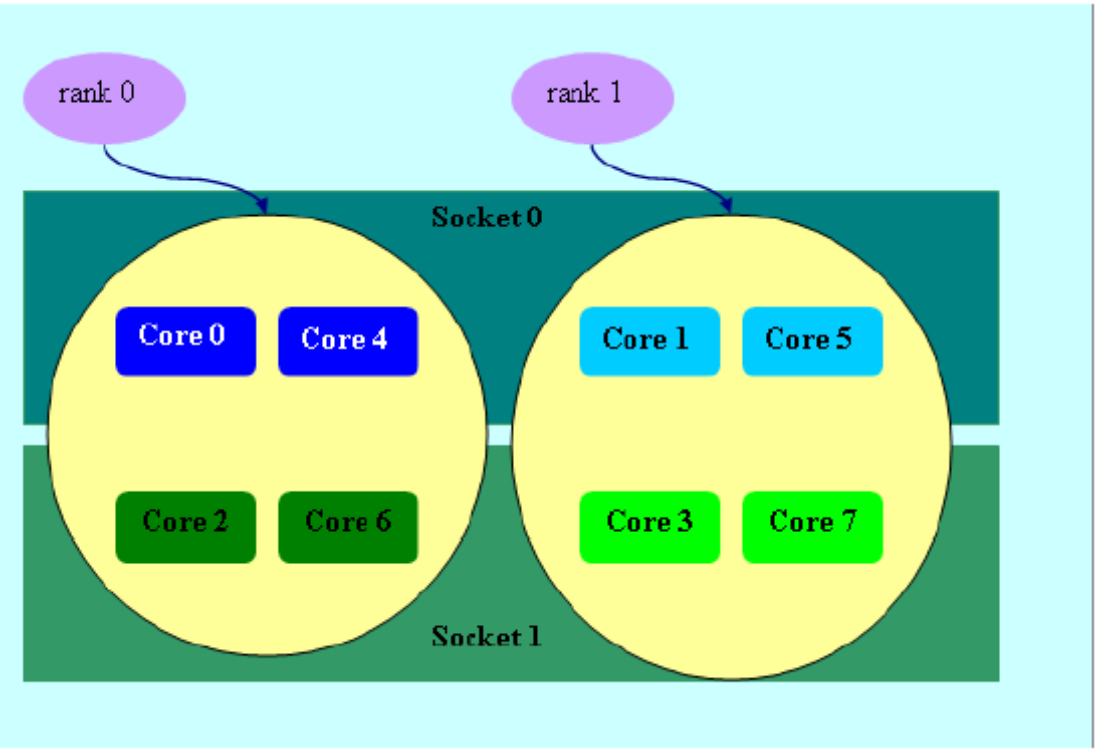


図 4.2-8 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN [0x55,0xaa] test.exe`

図 4.2-8 (I_MPI_PIN_DOMAIN=<マスクリスト> の例) では、最初のドメインは 0x55 マスクで定義されます。偶数番号 {0,2,4,6} を持つすべてのコアが含まれます。2 番目のドメインは 0xAA マスクで定義されます。奇数番号 {1,3,5,7} を持つすべてのコアが含まれます。

I_MPI_PIN_ORDER

I_MPI_PIN_DOMAIN 環境変数の値で指定されたドメインへの MPI プロセスの順番割り当てを定義します。

構文

I_MPI_PIN_ORDER=<順番>

引数

<順番>	ランクの順番を指定します。
range	ドメインは、プロセッサーの BIOS 番号付けに従って配置されます。これはプラットフォーム固有の番号付けです。
scatter	隣接するドメインが共有リソースを最小限に共有するようにドメインが配置されます。
compact	隣接するドメインが共有リソースを最大限に共有するようにドメインが配置されますこれは、デフォルト値です。
spread	共有リソースを共有しないように、ドメインは可能な限り連続配置されます。
bunch	プロセスはソケットに応じてマッピングされ、ドメインはソケット上で可能な限り隣接して配置されます。

説明

この環境変数はオプションで、アプリケーション固有です。隣接するプロセスが、コア、キャッシュ、ソケット、FSB などのリソースを共有する場合、compact または bunch に設定します。そうでない場合は、scatter または spread にします。必要に応じて range 値を使用します。これらの値に関する詳しい説明と例は、この章の I_MPI_PIN_ORDER の引数テーブルと例をご覧ください。

scatter、compact、spread および bunch オプションは、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。

例

次の構成の場合:

- 4 コアと対応するコアのペアが L2 キャッシュを共有する 2 つのソケットノード。
- 4 つの MPI プロセスを次の設定で、ノードで実行するとします。
  - compact の場合:
    I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=compact

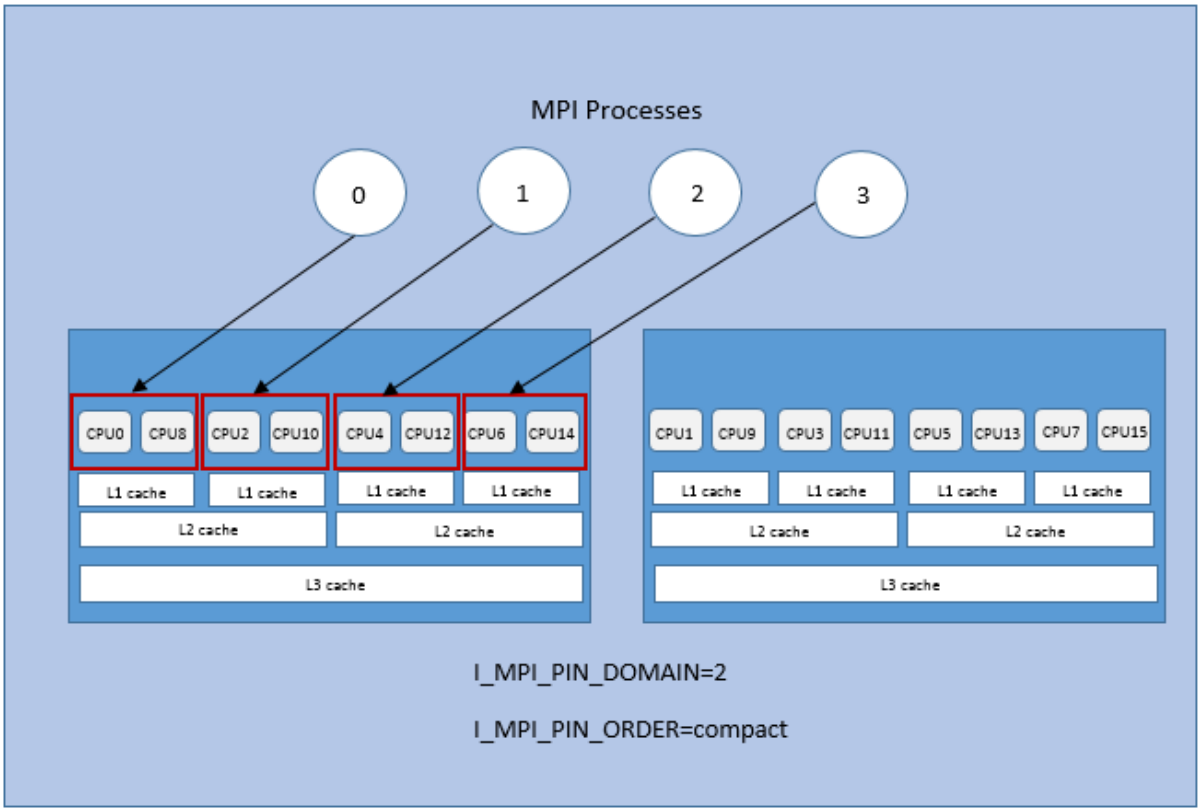


図 4.2-9 Compact オーダーの例

- scatter の場合:
  I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=scatter

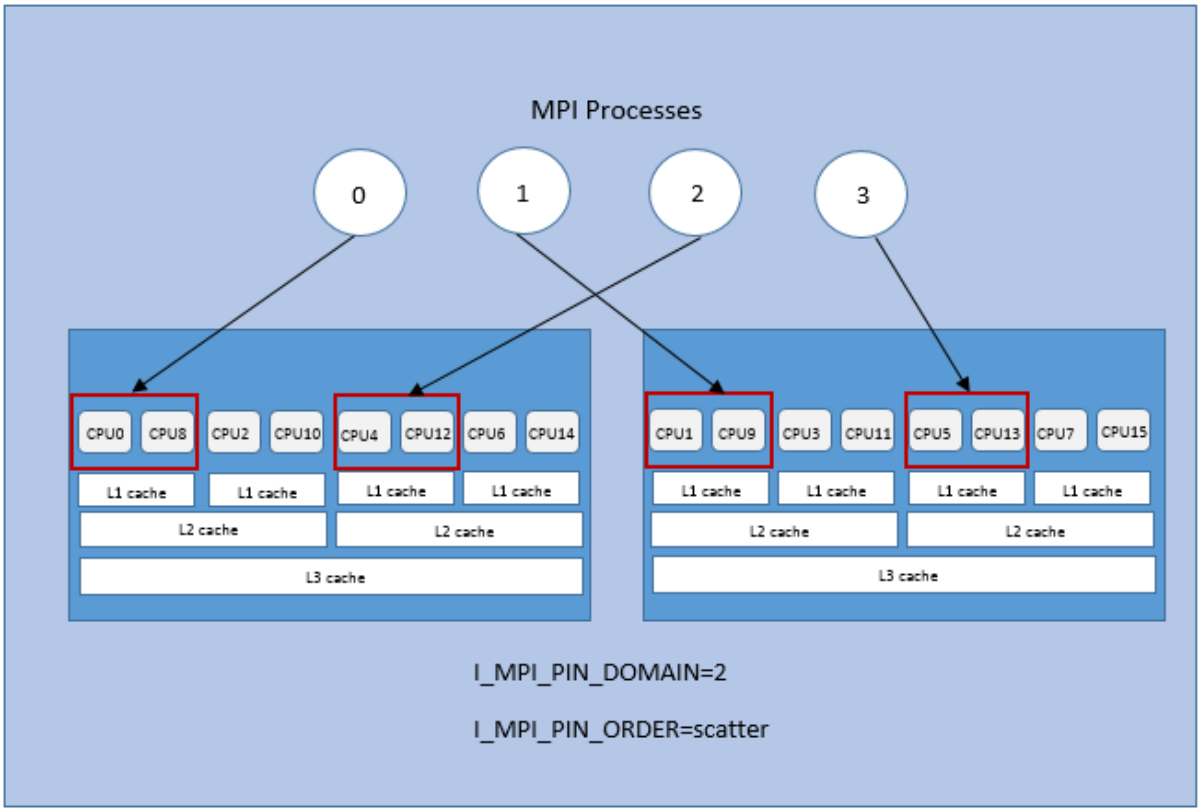


図 4.2-10 Scatter オーダーの例

- spread の場合:
  I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=spread

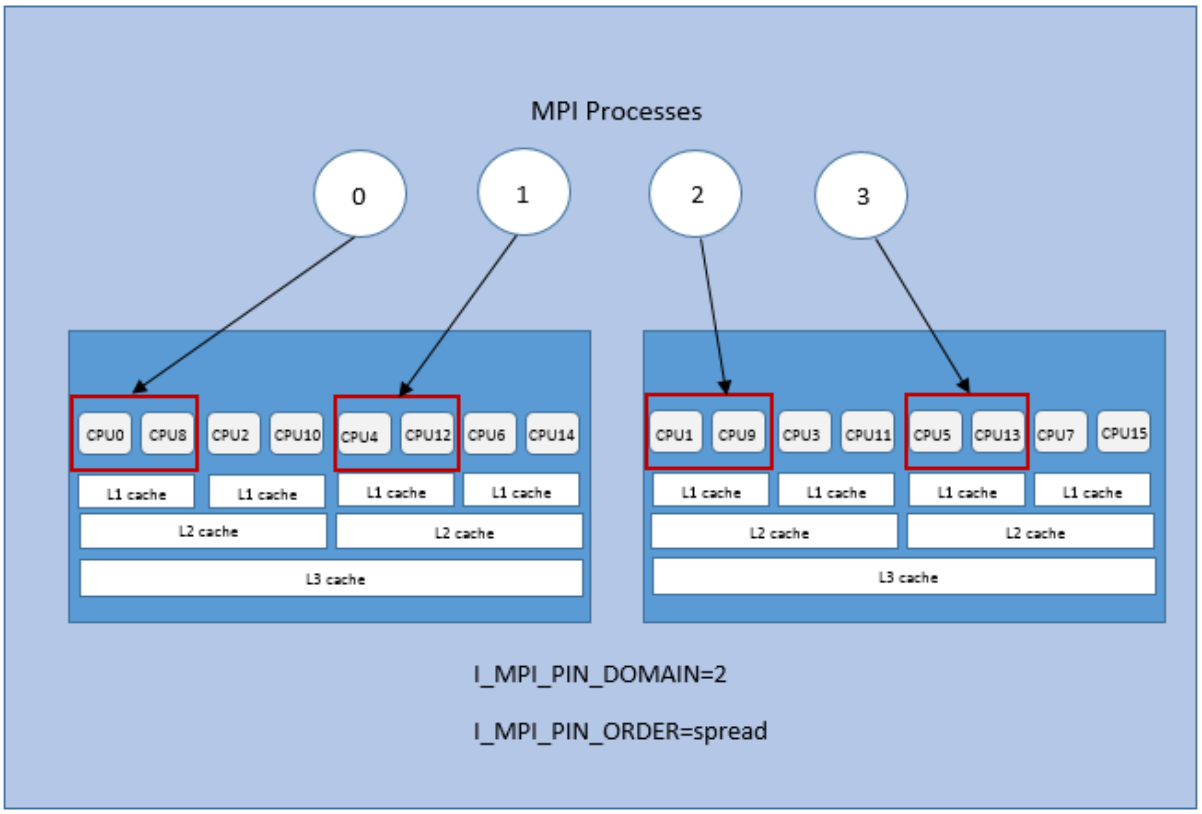


図 4.2-11 Spread オーダーの例

- bunch の場合:
  `I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=bunch`

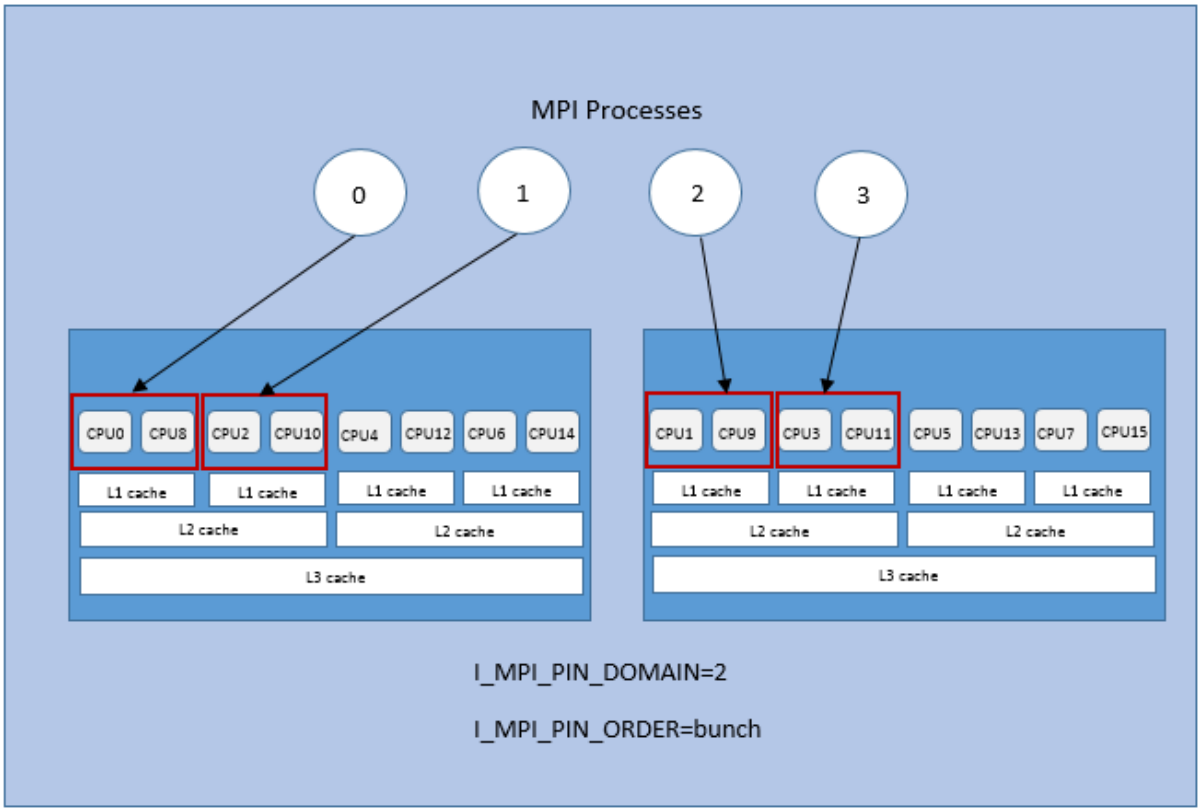


図 4.2-12 Bunch オーダーの例

4.3. ファブリック制御

ここでは、以下のファブリックを制御するため環境変数をどのように使用するか説明します。

- 通信ファブリック
- 共有メモリー・ファブリック
- DAPL ネットワーク・ファブリック
- TCP ネットワーク・ファブリック

4.3.1. 通信ファブリック制御

I_MPI_FABRIC (I_MPI_DEVICE)

特定のファブリックを選択します。

構文

`I_MPI_FABRICS=<ファブリック>|<ノード内のファブリック>:<ノード間のファブリック>`

引数には次を指定します。

`<ファブリック> := {shm, dapl, tcp}`

`<ノード内のファブリック> := {shm, dapl, tcp}`

`<ノード間のファブリック>:= {dapl, tcp}`

廃止された構文

`I_MPI_DEVICE=<デバイス>[:<プロバイダー>]`

引数

<ファブリック>	ネットワーク・ファブリックを定義します。
`shm`	共有メモリー。
`dapl`	InfiniBand*、iWarp*、Dolphin* や XPMEM* (DAPL* を介して) などの DAPL 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 ネットワーク・ファブリック。

I_MPI_DEVICE に対応

<デバイス>	<ファブリック>
`sock`	`tcp`
`shm`	`shm`
`ssm`	`shm:tcp`
`rdma`	`dapl`
`rdssm`	`shm:dapl`
<プロバイダー>	オプションの DAPL* プロバイダー名 (`rdma` と `rdssm` デバイスのみ) `I_MPI_DAPL_PROVIDER=<プロバイダー>`

`{rdma,rdssm}` デバイス向けにのみ <プロバイダー> 指定を使用します。

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 -n <プロセス数> \
-env I_MPI_DEVICE rdssm:OpenIB-cma <実行形式>
```

これらのデバイスは、<プロバイダー> が指定されていない場合、`\etc\dat.conf` ファイルの最初の DAPL* プロバイダーが使用されます。

説明

特定のファブリックを選択するため、この環境変数を設定します。要求するファブリックが利用できない場合、インテル® MPI ライブラリーはほかのファブリックにフォールバックします。詳細については、「I_MPI_FALLBACK」を参照してください。`I_MPI_FABRICS` 環境変数が定義されていない場合、インテル® MPI ライブラリーは、最も適切なファブリックの組み合わせを自動的に選択します。

ファブリックの実際の組み合わせは、ノードごとに開始されたプロセス数によって異なります。

- 1 つのノードですべてのプロセスが開始された場合、ライブラリーは `shm` ノード内通信を使用します。
- 開始されたプロセス数が利用可能なノード数以下の場合、ライブラリーは、ノード間通信にファブリック・リストから利用可能な最初のファブリックを選択します。
- 例えば、ライブラリーは、ノード内通信に `shm` を使用し、ノード間通信にファブリック・リストの最初の利用可能なファブリックを使用します。詳細については、「I_MPI_FABRICS_LIST」をご覧ください。

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

選択されたファブリックの組み合わせでジョブが実行されることは保証されますが、その組み合わせがクラスター構成の最高のパフォーマンスを提供するとは限りません。

例えば、ファブリックに共有メモリーを選択するには、次のコマンドを使用します。

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

ファブリック通信に共有メモリーと DAPL ネットワーク・ファブリックを使用するには、次のコマンドを使用します。

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

インテル® MPI ライブラリーが適切なファブリック通信を自動選択するようにするには、次のコマンドを使用します。

```
> mpiexec -n <プロセス数> -machinefile smpd.hosts <実行形式>
```

デバッグ情報のレベルに 2 以上を設定すると、初期化されたファブリックをチェックできます。詳細は、「I_MPI_DEBUG」をご覧ください。次に例を示します。

```
[0] MPI startup(): shm and dapl data transfer modes
```

または

```
[0] MPI startup(): tcp data transfer mode
```

I_MPI_FABRICS_LIST

ファブリック・リストを定義します。

構文

```
I_MPI_FABRICS_LIST=<ファブリック・リスト>
```

```
<ファブリック・リスト>:= <ファブリック>,...,<ファブリック>
```

```
<ファブリック> := {dapl, tcp}
```

引数

<ファブリック・リスト>	ファブリックのリストを指定します。デフォルトは dapl, tcp です。

説明

この環境変数を設定して、ファブリックのリストを定義します。ライブラリーは、自動的に適切なファブリックの組み合わせを選択するため、ファブリック・リストを使用します。ファブリックの組み合わせに関する詳細は、「I_MPI_FABRICS」をご覧ください。

例えば、I_MPI_FABRICS_LIST=dapl, tcp が設定され、I_MPI_FABRICS が定義されていない場合、DAPL ネットワーク・ファブリックの初期化に失敗すると、ライブラリーは TCP ネットワーク・ファブリックにフォールバックします。フォールバックに関する詳細は、「I_MPI_FALLBACK」をご覧ください。

I_MPI_FALLBACK (I_MPI_FALLBACK_DEVICE)

最初に利用可能なファブリックへのフォールバックドを有効にするには、この環境変数を設定します。

構文

```
I_MPI_FALLBACK=<引数>
```

廃止された構文

`I_MPI_FALLBACK_DEVICE=<引数>`

引数

<引数>	バイナリー・インジケーター。
`enable | yes | on | 1`	最初に利用可能なファブリックにフォールバックします。これは、I_MPI_FABRICS (I_MPI_DEVICE) 環境変数が定義されていない場合のデフォルトです。
`disable | no | off | 0`	MPI は、I_MPI_FABRICS 環境変数で選択されているファブリックの 1 つの初期化に失敗すると、ジョブを強制終了します。これは、I_MPI_FABRICS (I_MPI_DEVICE) 環境変数が定義されていない場合のデフォルトです。

説明

最初に利用可能なファブリックへのフォールバックドを制御するには、この環境変数を設定します。

I_MPI_FALLBACK 環境変数が `enable` に設定され、指定されたファブリックの初期化に失敗すると、ライブラリーはファブリック・リストを参照し、最初に利用可能なファブリックを使用します。詳細については、「I_MPI_FABRICS_LIST」をご覧ください。

I_MPI_FALLBACK 環境変数が `disable` に設定され、指定されたファブリックの初期化に失敗すると、ライブラリーは MPI ジョブを強制終了します。

注意

I_MPI_FABRICS を設定し `I_MPI_FALLBACK=enable` にすると、ライブラリーはファブリック・リストの最上位の番号のファブリックへフォールバックします。例えば、`I_MPI_FABRICS=dapl`, `I_MPI_FABRICS_LIST=dapl,tcp`、`I_MPI_FALLBACK=enable` が設定されると、DAPL ネットワーク・ファブリックの初期化に失敗すると、ライブラリーは TCP ネットワーク・ファブリックにフォールバックします。

I_MPI_EAGER_THRESHOLD

すべてのデバイスの eager/rendezvous メッセージサイズのしきい値を変更します。

構文

`I_MPI_EAGER_THRESHOLD=<バイト数>`

引数

<バイト数>	eager/rendezvous メッセージサイズのしきい値を設定します。
`> 0`	デフォルトの <バイト数> 値は、262144 バイトです。

説明

この環境変数は、ポイントツーポイント通信に使用されるプロトコルを制御します。

- メッセージが、<バイト数> 以下の場合、eager プロトコルが使用されます。
- メッセージが、<バイト数> より長い場合、rendezvous プロトコルが使用されます。rendezvous プロトコルは、メモリーを効率良く使用します。

I_MPI_INTRANODE_EAGER_THRESHOLD

ノード内通信の eager/rendezvous メッセージサイズのしきい値を変更します。

構文

I_MPI_INTRANODE_EAGER_THRESHOLD=<バイト数>

引数

<バイト数>	ノード内通信の eager/rendezvous メッセージサイズのしきい値を設定します。
> 0	すべてのファブリックのデフォルトの <バイト数> 値は、262144 バイトです (shm を除く)。shm の場合、しきい値は、I_MPI_SHM_CELL_SIZE 環境変数の値に等しくなります。

説明

この環境変数は、ノード内での通信に使用されるプロトコルを変更します。

- メッセージが、<バイト数> 以下の場合、eager プロトコルが使用されます。
- メッセージが、<バイト数> より長い場合、rendezvous プロトコルが使用されます。rendezvous プロトコルは、メモリーを効率良く使用します。

I_MPI_INTRANODE_EAGER_THRESHOLD が設定されていない場合、I_MPI_EAGER_THRESHOLD の値が使用されます。

I_MPI_SPIN_COUNT

スピンカウント値を制御します。

構文

I_MPI_SPIN_COUNT=<スピンカウント>

引数

<スピンカウント>	ファブリックをポーリングする際のループのスピンカウントを定義します。
> 0	プロセッサー/コアごとに複数のプロセスが実行されている場合、デフォルトの <スピンカウント> は 1 です。それ以外のデフォルトは、250 になります。最大値は、2147483647 です。

説明

スピンカウントの上限を設定します。処理するメッセージを受信していない場合、ライブラリーがプロセスを開放する前にファブリックのポーリングに、<スピンカウント> で指定される回数だけループします。それぞれのスピンループ内で、shm ファブリック (有効であれば) は、I_MPI_SHM_SPIN_COUNT 回だけ余分にポーリングします。<スピンカウント> に小さな値を設定すると、インテル® MPI ライブラリーは頻繁にプロセッサーを開放します。

アプリケーションのパフォーマンスをチューニングするには、I_MPI_SPIN_COUNT 環境変数を使用します。<スピンカウント> に設定する最良の値は、経験則に従います。それは、計算環境やアプリケーションに依存します。

I_MPI_SCALABLE_OPTIMIZATION

ネットワーク・ファブリック通信のスケーラブルな最適化を on/off にします。

構文

I_MPI_SCALABLE_OPTIMIZATION=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	ネットワーク・ファブリック通信のスケーラブルな最適化を on にします。これは、16 プロセッサー以上のデフォルトです。
disable | no | off | 0	ネットワーク・ファブリック通信のスケーラブルな最適化を off にします。これは、16 プロセッサー未満のデフォルトです。

説明

ネットワーク・ファブリック通信のスケーラブルな最適化を有効にするには、この環境変数を設定します。多くの場合、最適化を使用するとレイテンシーが増え、大規模なプロセスではバンド幅が増加します。

I_MPI_WAIT_MODE

待機モードを on/off にします。

構文

I_MPI_WAIT_MODE=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	待機モードを on にします。
disable | no | off | 0	待機モードを off にします。これはデフォルトです。

説明

待機モードを制御するには、この環境変数を設定します。このモードを有効にすると、プロセスはファブリックをポーリングすることなくメッセージの受信を待ちます。このモードは、ほかのタスクに CPU 時間を温存できます。

shm 通信には、ネイティブ POSIX* スレッド・ライブラリーを待機モードで使用します。

注意

次のコマンドを使用して、インストールされているスレッド・ライブラリーのバージョンを確認できます。

```
$ getconf GNU_LIBPTHREAD_VERSION
```

I_MPI_DYNAMIC_CONNECTION (I_MPI_USE_DYNAMIC_CONNECTIONS)

ダイナミック接続確立を制御します。

構文

I_MPI_DYNAMIC_CONNECTION=<引数>

廃止された構文

I_MPI_USE_DYNAMIC_CONNECTIONS=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	ダイナミック接続確立を on にします。これは、64 プロセッサー以上のデフォルトです。
disable | no | off | 0	ダイナミック接続確立を off にします。これは、64 プロセッサー未満のデフォルトです。

説明

ダイナミック接続確立を制御するには、この環境変数を設定します。

- このモードが有効な場合、すべての接続は各プロセスのペア間で最初の通信時に確立されます。
- このモードが無効な場合、すべての接続は事前に確立されます。

デフォルトの値は、MPI ジョブのプロセス数に依存します。ダイナミック接続確立は、プロセス数が 64 未満の場合 off です。

4.3.2. 共有メモリー制御

I_MPI_SHM_CACHE_BYPASS (I_MPI_CACHE_BYPASS)

共有メモリー向けのメッセージ転送のアルゴリズムを制御します。

構文

I_MPI_SHM_CACHE_BYPASS=<引数>

廃止された構文

I_MPI_CACHE_BYPASS=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	メッセージ転送バイパスキャッシュを有効にします。これは、デフォルト値です。
disable | no | off | 0	メッセージ転送バイパスキャッシュを無効にします。

説明

共有メモリー向けのメッセージ転送のバイパスキャッシュを有効/無効にするには、この環境変数を設定します。この機能を有効にすると、MPI はバイパスキャッシュを介して、I_MPI_SHM_CACHE_BYPASS_THRESHOLD 環境変数に設定される値と同じか、大きなサイズのメッセージを送信します。この機能は、デフォルトで有効になっています。

I_MPI_SHM_CACHE_BYPASS_THRESHOLDS (I_MPI_CACHE_BYPASS_THRESHOLDS)

メッセージコピーのアルゴリズムのしきい値を設定します。

構文

I_MPI_SHM_CACHE_BYPASS_THRESHOLDS=<nb_send>,<nb_recv>[,<nb_send_pk>,<nb_recv_pk>]

廃止された構文

`I_MPI_CACHE_BYPASS_THRESHOLDS=<nb_send>,<nb_recv>[,<nb_send_pk>,<nb_recv_pk>]`

引数

`<nb_send>`	次の状況で送信するメッセージのしきい値を設定します。 • プロセスは、同じ物理コアのプロセッサー・パッケージ内にないコアにピニングされている • プロセスはピニングされていない
`<nb_recv>`	次の状況で受信するメッセージのしきい値を設定します。 • プロセスは、同じ物理プロセッサー・パッケージ内にないコアにピニングされている • プロセスはピニングされていない
`<nb_send_pk>`	プロセスが、同じ物理プロセッサー・パッケージ内のコアにピニングされている場合に、送信するメッセージのしきい値を設定します。
`<nb_recv_pk>`	プロセスが、同じ物理プロセッサー・パッケージ内のコアにピニングされている場合に、受信するメッセージのしきい値を設定します。

説明

メッセージコピーのアルゴリズムのしきい値を制御するには、この環境変数を設定します。インテル® MPI ライブラリーは、異なるメモリー階層レベルで動作するように最適化されたメッセージコピーの実装を使用します。インテル® MPI ライブラリーは、離れたメモリーアクセスに最適化されたコピー・アルゴリズムを使用して、定義されたしきい値以上のメッセージをコピーします。`-1` を設定すると、これらのアルゴリズムの使用を無効にします。デフォルトの値は、アーキテクチャーとインテル® MPI ライブラリーのバージョンに依存します。この環境変数は、`I_MPI_SHM_CACHE_BYPASS` が有効なときにのみ効果があります。

この環境変数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。

I_MPI_SHM_FBOX

共有メモリーのファストボックスを制御します。

構文

`I_MPI_SHM_FBOX=<引数>`

引数

`<引数>`	バイナリー・インジケーター。
`enable | yes | on | 1`	ファストボックスの利用を on にします。これは、デフォルト値です。
`disable | no | off | 0`	ファストボックスの利用を `off` にします。

説明

ファストボックスを制御するには、この環境変数を設定します。同一ノード上の MPI プロセスのペアは、eager メッセージを送受信するため 2 つの共有メモリー・ファストボックスを持っています。

アプリケーションが、ブロック化されていない短いメッセージを大量に転送する場合、メッセージ同期のオーバーヘッドを避けるためファストボックスの利用を off にします。

I_MPI_SHM_FBOX_SIZE

共有メモリーのファストボックスのサイズを設定します。

構文

I_MPI_SHM_FBOX_SIZE=<バイト数>

引数

<バイト数>	共有メモリーのファストボックスのサイズをバイト単位で指定します。
> 0	デフォルトの <バイト数> は、プラットフォームに依存します。値の範囲は、一般的に 8K から 64K です。

説明

共有メモリー・ファストボックスのサイズを定義するには、この環境変数を設定します。

I_MPI_SHM_CELL_NUM

共有メモリー受信キューのセル数を変更するには、この環境変数を設定します。

構文

I_MPI_SHM_CELL_NUM=<数値>

引数

<数値>	共有メモリーセルの数。
> 0	デフォルト値は 128 です。

説明

共有メモリー受信キューのセル数を定義するには、この環境変数を設定します。各 MPI プロセスは、ほかのプロセスが eager メッセージを送信できる独自の共有メモリー受信キューを持っています。共有メモリー・ファストボックスがほかの MPI 要求でブロックされると、このキューが使用されます。

I_MPI_SHM_CELL_SIZE

共有メモリーセルのサイズを変更します。

構文

I_MPI_SHM_CELL_SIZE=<バイト数>

引数

<バイト数>	共有メモリーセルのサイズをバイト単位で指定します。
> 0	デフォルトの <バイト数> は、プラットフォームに依存します。値の範囲は、一般的に 8K から 64K です。

説明

共有メモリーセルのサイズを定義するにはこの環境変数を設定します。

この環境変数を設定すると、I_MPI_INTRANODE_EAGER_THRESHOLD も同時に変更され、設定された値に等しくなります。

I_MPI_SHM_LMT

共有メモリー向けのラージメッセージ転送 (LMT) のメカニズムを制御します。

構文

I_MPI_SHM_LMT=<引数>

引数

<引数>	バイナリー・インジケーター。
direct	直接コピー LMT メカニズムを on にします。これは、デフォルト値です。
disable | no | off | 0	LMT メカニズムを off にします。

説明

ラージメッセージ転送 (LMT) の使用法を制御するには、この環境変数を設定します。rendezvous メッセージを転送するには、次のどちらかの方法で LMT メカニズムを使用します。

- メッセージを送信するため、中間共有メモリーキューを使用する。
- 中間バッファーなしでメッセージを転送する直接コピーメカニズムを使用する。

I_MPI_SHM_LMT_BUFFER_NUM (I_MPI_SHM_NUM_BUFFERS)

ラージメッセージ転送 (LMT) メカニズム向けの共有メモリーバッファーの数を変更します。

構文

I_MPI_SHM_LMT_BUFFER_NUM=<数値>

廃止された構文

I_MPI_SHM_NUM_BUFFERS=<数値>

引数

<数値>	プロセスの各ペア向けの共有メモリーバッファーの数。
> 0	デフォルト値は 8 です。

説明

ペアの各プロセッサー間の共有メモリーバッファー数を定義するには、この環境変数を設定します。

I_MPI_SHM_LMT_BUFFER_SIZE (I_MPI_SHM_BUFFER_SIZE)

LMT メカニズム向けの共有メモリーバッファーのサイズを制御します。

構文

I_MPI_SHM_LMT_BUFFER_SIZE=<バイト数>

廃止された構文

I_MPI_SHM_BUFFER_SIZE=<バイト数>

引数

<バイト数>	共有メモリーバッファーのサイズをバイト単位で指定します。
> 0	デフォルトの <バイト数> 値は、32768 バイトです。

説明

ペアの各プロセッサー間の共有メモリーバッファーのサイズを定義するには、この環境変数を設定します。

I_MPI_SSHM

スケーラブルな共有メモリーメカニズムを制御します。

構文

```
I_MPI_SSHM =<引数>
```

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	このメカニズムを on にします。
disable | no | off | 0	このメカニズムを off にします。これは、デフォルト値です。

説明

代替共有メモリーメカニズムの使用法を制御するには、この環境変数を設定します。このメカニズムは、共有メモリー・ファストボックス、受信キュー、および LMT メカニズムを置き換えます。

この環境変数を設定すると、I_MPI_INTRANODE_EAGER_THRESHOLD 環境変数も変更され、262,144 バイトに等しくなります。

I_MPI_SSHM_BUFFER_NUM

代替共有メモリー向けの共有メモリーバッファー数を制御します。

構文

```
I_MPI_SSHM_BUFFER_NUM=<数値>
```

引数

<数値>	プロセスの各ペア向けの共有メモリーバッファーの数。
> 0	デフォルト値は 4 です。

説明

ペアの各プロセッサー間の共有メモリーバッファー数を定義するには、この環境変数を設定します。

I_MPI_SSHM_BUFFER_SIZE

代替共有メモリー向けの共有メモリーバッファーのサイズを制御します。

構文

```
I_MPI_SSHM_BUFFER_SIZE=<バイト数>
```

引数

<バイト数>	共有メモリーバッファーのサイズをバイト単位で指定します。
> 0	デフォルトの <バイト数> は、プラットフォームに依存します。 値の範囲は、一般的に 8K から 64K です。

説明

ペアの各プロセッサー間の共有メモリーバッファーのサイズを定義するには、この環境変数を設定します。

I_MPI_SSHM_DYNAMIC_CONNECTION

代替共有メモリーメカニズム向けのダイナミック接続確立を制御します。

構文

```
I_MPI_SSHM_DYNAMIC_CONNECTION=<引数>
```

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	ダイナミック接続確立を on にします。
disable | no | off | 0	ダイナミック接続確立を off にします。これは、デフォルト値です。

説明

ダイナミック接続確立を制御するには、この環境変数を設定します。

- このモードが有効な場合、すべての接続は各プロセスのペア間で最初の通信時に確立されます。
- このモードが無効な場合、すべての接続は事前に確立されます。

I_MPI_SHM_BYPASS (I_MPI_INTRANODE_SHMEM_BYPASS, I_MPI_USE_DAPL_INTRANODE)

shm によるネットワーク・ファブリックを介したノード内通信モードを on/off にします。

構文

```
I_MPI_SHM_BYPASS=<引数>
```

廃止された構文

```
I_MPI_INTRANODE_SHMEM_BYPASS=<引数>
I_MPI_USE_DAPL_INTRANODE=<引数>
```

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	ネットワーク・ファブリックを介したノード内通信モードを on にします。
disable | no | off | 0	ネットワーク・ファブリックを介したノード内通信モードを off にします。これはデフォルトです。

説明

この環境変数は、ノード内での通信モードを設定します。ネットワーク・ファブリックを介してノード内の通信モードが有効にされている場合、データ転送メカニズムは次のスキームに従って選択されます。

- メッセージのサイズが、I_MPI_INTRANODE_EAGER_THRESHOLD 環境変数に設定されるしきい値以下の場合、共有メモリーを使用して転送されます。
- メッセージのサイズが、I_MPI_INTRANODE_EAGER_THRESHOLD 環境変数に設定されるしきい値より大きい場合は、ネットワーク・ファブリック・レイヤー経由で転送されます。

注意

この環境変数は、共有メモリーを有効にし、I_MPI_FABRICS 環境変数をデフォルトもしくは shm:<ファブリック> に設定するか、I_MPI_DEVICE 環境変数を同様に設定することでネットワーク・ファブリックが指定される場合にのみ有効になります。このモードは、dapl と tcp ファブリックでのみ利用できます。

I_MPI_SHM_SPIN_COUNT

共有メモリー・ファブリック向けのスピンカウントを制御します。

構文

I_MPI_SHM_SPIN_COUNT=<共有メモリー・スピンカウント>

引数

<スピンカウント>	shm ファブリックをポーリングする際の、ループのスピンカウントを定義します。
> 0	デフォルトの <共有メモリー・スピンカウント> 値は、100 回です。

説明

ポーリングの頻度を高めるため、共有メモリー・ファブリックのスピンカウントの上限を設定します。この構成は、制御が全体のネットワーク・ファブリックのポーリングメカニズムに渡される前に、shm ファブリックのポーリングを <shm スピンカウント> 回許可します。

アプリケーションのパフォーマンスをチューニングするには、I_MPI_SHM_SPIN_COUNT 環境変数を使用します。<shm スピンカウント> の最適な値の選択は、経験に依存します。これは、アプリケーションと計算環境に大きく依存します。アプリケーションがメッセージパッシングにトポロジー的なアルゴリズムを使用する場合、<shm スピンカウント> の値を大きくすることでマルチコア・プラットフォームに利点があります。

4.3.3. DAPL ネットワーク・ファブリック制御

I_MPI_DAPL_PROVIDER

ロードする DAPL プロバイダーを定義します。

構文

I_MPI_DAPL_PROVIDER=<名前>

引数

<名前>	ロードする DAPL プロバイダーの名前を定義します。

説明

この環境変数は、共有メモリーとネットワーク・ファブリックを有効にするか、デフォルトもしくは I_MPI_FABRICS 環境変数を shm:<ファブリック> に設定するか、I_MPI_DEVICE 環境変数を設定する場合にのみ有効になります。このモードは、dapl と tcp ファブリックでのみ利用できます。

I_MPI_DAT_LIBRARY

DAPL* プロバイダーで使用する DAT ライブラリーを選択します。

構文

I_MPI_DAT_LIBRARY=<ライブラリー>

引数

<ライブラリー>	DAPL* プロバイダーで使用する DAT ライブラリーを指定します。デフォルト値は、DAPL* 1.2 プロバイダーには dat.dll、DAPL* 2.0 プロバイダーには dat2.dll です。

説明

DAPL プロバイダーで使用する特定の DAT ライブラリーを選択するには、この環境変数を設定します。ライブラリーが、ダイナミック・ローダーの検索パスに配置されていない場合、DAT ライブラリーへのフルパスを指定します。この環境変数は、DAPL ファブリックにのみ有効です。

I_MPI_DAPL_TRANSLATION_CACHE (I_MPI_RDMA_TRANSLATION_CACHE)

DAPL パスのメモリー登録キャッシュを on/off にします。

構文

I_MPI_DAPL_TRANSLATION_CACHE=<引数>

廃止された構文

I_MPI_RDMA_TRANSLATION_CACHE=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	メモリー登録キャッシュを on にします。これはデフォルトです。
disable | no | off | 0	メモリー登録キャッシュを off にします。

説明

DAPL パスのメモリー登録キャッシュを on/off にするため、この環境変数を使用します。

キャッシュはパフォーマンスを大幅に向上しますが、特定の状況で正当性の問題を引き起こす可能性があります。詳細については、製品のリリースノートをご覧ください。

I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE

DAPL パスの RDMA 変換キャッシュの AVL tree* ベースの実装を有効/無効にします。

構文

I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE=<引数>

引数

<引数>	バイナリー・インジケーター。
`enable | yes | on | 1`	AVL tree ベースの RDMA 変換キャッシュを on にします。
`disable | no | off | 0`	AVL tree ベースの RDMA 変換キャッシュを `off` にします。これは、デフォルト値です。

説明

この環境変数を設定して、DAPL パスの RDMA 変換キャッシュの AVL tree* ベースの実装を有効にします。RDMA 変換キャッシュが 10,000 を超える要素を処理する場合、AVL tree ベースの RDMA 変換キャッシュの方がデフォルト実装より高速です。

I_MPI_DAPL_DIRECT_COPY_THRESHOLD (I_MPI_RDMA_EAGER_THRESHOLD、RDMA_IBA_EAGER_THRESHOLD)

DAPL 直接コピープロトコルのしきい値を変更します。

構文

`I_MPI_DAPL_DIRECT_COPY_THRESHOLD=<バイト数>`

廃止された構文

`I_MPI_RDMA_EAGER_THRESHOLD=<バイト数>`
`RDMA_IBA_EAGER_THRESHOLD=<バイト数>`

引数

<バイト数>	DAPL 直接コピープロトコルのしきい値を定義します。
`> 0`	デフォルトの <バイト数> は、プラットフォームに依存します。

説明

DAPL 直接コピープロトコルのしきい値を制御するため、この環境変数を設定します。DAPL ネットワーク・ファブリック向けのデータ転送アルゴリズムは、次のスキームに従って選択されます。

- メッセージが <バイト数> 以下の場合、内部事前登録バッファーを介して eager プロトコルを使用して送信します。このアプローチは、ショートメッセージでは高速です。
- メッセージが、<バイト数> より長い場合、直接コピープロトコルが使用されます。これはバッファーを使用しませんが、送信側と受信側でメモリーの登録が必要です。このアプローチは、ラージメッセージでは高速です。

この環境変数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

インテル® Xeon Phi™ コプロセッサー向けの等価な値は、`I_MIC_MPI_DAPL_DIRECT_COPY_THRESHOLD` です。

I_MPI_DAPL_EAGER_MESSAGE_AGGREGATION

MPI の送信要求を延期するため、連結を使用して制御します。延期された MPI 送信要求は、すぐに送信できません。

構文

I_MPI_DAPL_EAGER_MESSAGE_AGGREGATION=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	MPI の送信要求を延期するため連結を有効にします。
disable | no | off | 0	MPI の送信要求を延期するため連結を無効にします。これは、デフォルト値です。

同じ MPI ランクへの MPI 送信要求を延期するため連結を使用するには、この環境変数を設定します。このモードは、特定の状況でアプリケーションのパフォーマンスを改善します。例えば、次のように MPI_Isend() が短いメッセージを同じランクに送信する場合にパフォーマンスが向上します。

```
for( i = 0; i < NMSG; i++){
ret = MPI_Isend( sbuf[i], MSG_SIZE, datatype, dest , tag,\ comm, &req_send[i]);
}
```

I_MPI_DAPL_DYNAMIC_CONNECTION_MODE (I_MPI_DYNAMIC_CONNECTION_MODE、I_MPI_DYNAMIC_CONNECTIONS_MODE)

DAPL* 接続を確立するアルゴリズムを選択します。

構文

I_MPI_DAPL_DYNAMIC_CONNECTION_MODE=<引数>

廃止された構文

I_MPI_DYNAMIC_CONNECTION_MODE=<引数>
I_MPI_DYNAMIC_CONNECTIONS_MODE=<引数>

引数

<引数>	モードセレクター。
reject	2 つの同時接続要求の一方を拒否します。これはデフォルトです。
disconnect	2 つの接続が確立された後、一方の同時接続要求を拒否します。

説明

次のスキームに従って、DAPL 対応ファブリックの動的に確立された接続を制御するには、この環境変数を設定します。

- reject モードでは、2 つのプロセスが同時に接続を開始すると、一方の要求が拒否されます。
- disconnect モードでは、両方の接続が確立されますが、その後一方が切断されます。
  disconnect モードは、特定の DAPL* プロバイダーのバグを回避するため設けられています。

I_MPI_DAPL_SCALABLE_PROGRESS (I_MPI_RDMA_SCALABLE_PROGRESS)

DAPL 読み込み向けのスケーラブルなアルゴリズムを on/off にします。

構文

I_MPI_DAPL_SCALABLE_PROGRESS=<引数>

廃止された構文

I_MPI_RDMA_SCALABLE_PROGRESS=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	スケーラブルなアルゴリズムを on にします。プロセス数が 128 より多い場合、これがデフォルトになります。
disable | no | off | 0	スケーラブルなアルゴリズムを off にします。プロセス数が 128 以下の場合、これがデフォルトになります。

説明

DAPL 読み込みのスケーラブルな最適化を有効にするには、この環境変数を設定します。特定の状況では、これはシステムと多くのプロセスの利点を生かすことが可能です。

I_MPI_DAPL_BUFFER_NUM (I_MPI_RDMA_BUFFER_NUM, NUM_RDMA_BUFFER)

DAPL パスの各プロセスのペア向けに、内部的に事前登録バッファーの数を変更します。

構文

I_MPI_DAPL_BUFFER_NUM=<バッファー数>

廃止された構文

I_MPI_RDMA_BUFFER_NUM=<バッファー数>
NUM_RDMA_BUFFER=<バッファー数>

引数

<バッファー数>	プロセスグループの各ペア向けにバッファー数を定義します。
> 0	デフォルト値は、プラットフォームに依存します。

説明

DAPL パスの各プロセスのペア向けに、内部的に事前登録バッファーの数を変更するには、この環境変数を設定します。

注意

事前登録バッファーの数が増えると、それぞれ確立された接続ごとのメモリー使用量は増加します。

I_MPI_DAPL_BUFFER_SIZE (I_MPI_RDMA_BUFFER_SIZE、I_MPI_RDMA_VBUF_TOTAL_SIZE)

DAPL パスの各プロセスのペア向けに、内部的に事前登録バッファーのサイズを変更します。

構文

I_MPI_DAPL_BUFFER_SIZE=<バイト数>

廃止された構文

I_MPI_RDMA_BUFFER_SIZE=<バイト数>
I_MPI_RDMA_VBUF_TOTAL_SIZE=<バイト数>

引数

<バイト数>	事前定義バッファーのサイズを定義します。
`> 0`	デフォルト値は、プラットフォームに依存します。

説明

DAPL パスの各プロセスのペア向けに、内部的に事前登録バッファーのサイズを変更するには、この環境変数を設定します。実際のサイズは、<バイト数> をバッファーをアライメントする最適値にすることによって求められます。

I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT (I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT、I_MPI_RDMA_RNDV_BUF_ALIGN)

DAPL 直接コピー転送向けのバッファー送信アルゴリズムを定義します。

構文

```
I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT=<引数>
```

廃止された構文

```
I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT=<引数>
I_MPI_RDMA_RNDV_BUF_ALIGN=<引数>
```

引数

<引数>	バッファーを送信するアルゴリズムを定義します。
`> 0` かつ `2` の累乗	デフォルト値は 64 です。

DAPL 直接コピー転送向けのバッファー送信アルゴリズムを定義するため、この環境変数を設定します。DAPL 操作で指定されるバッファーが適切にアライメントされている場合、データ転送のバンド幅は高まります。

I_MPI_DAPL_RDMA_RNDV_WRITE (I_MPI_RDMA_RNDV_WRITE、I_MPI_USE_RENDEZVOUS_RDMA_WRITE)

DAPL パスで RDMA 書き込みベースの rendezvous 直接コピープロトコルを on/off にします。

構文

```
I_MPI_DAPL_RDMA_RNDV_WRITE=<引数>
```

廃止された構文

```
I_MPI_RDMA_RNDV_WRITE=<引数>
I_MPI_USE_RENDEZVOUS_RDMA_WRITE=<引数>
```

引数

<引数>	バイナリー・インジケーター。
`enable | yes | on | 1`	RDMA 書き込みベースンの rendezvous 直接コピープロトコルを on にします。
`disable | no | off | 0`	RDMA 書き込みベースの rendezvous 直接コピープロトコルを `off` にします。

説明

DAPL パスで RDMA 書き込みベースの rendezvous 直接コピープロトコルを選択するため、この環境変数を設定します。DAPL* プロバイダーによっては、特定のプラットフォーム向けに低速の RDMA 読み込みが実装されます。

その場合、RDMA 書き込み操作で rendezvous 直接コピープロトコルに切り替えると、パフォーマンスを向上できます。デフォルト値は、DAPL プロバイダーの属性に依存します。

I_MPI_DAPL_CHECK_MAX_RDMA_SIZE (I_MPI_RDMA_CHECK_MAX_RDMA_SIZE)

DAPL 属性、max_rdma_size の値をチェックします。

構文

`I_MPI_DAPL_CHECK_MAX_RDMA_SIZE=<引数>`

廃止された構文

`I_MPI_RDMA_CHECK_MAX_RDMA_SIZE=<引数>`

引数

<引数>	バイナリー・インジケーター
`enable | yes | on | 1`	DAPL 属性、`max_rdma_size` の値をチェックします。
`disable | no | off | 0`	DAPL 属性、`max_rdma_size` の値をチェックしません。これは、デフォルト値です。

説明

以下のスキームに従ってメッセージの断片化を制御するには、この環境変数を設定します。

- このモードが有効な場合、インテル® ライブラリーは DAPL 属性 `max_rdma_size` の値よりも大きなメッセージを断片化します。
- このモードが無効な場合、インテル® ライブラリーはメッセージの断片化のため DAPL 属性 `max_rdma_size` の値を考慮しません。

I_MPI_DAPL_MAX_MSG_SIZE (I_MPI_RDMA_MAX_MSG_SIZE)

メッセージの断片化のしきい値を制御します。

構文

`I_MPI_DAPL_MAX_MSG_SIZE=<バイト数>`

廃止された構文

`I_MPI_RDMA_MAX_MSG_SIZE=<バイト数>`

引数

<バイト数>	断片化なしで DAPL を介して送信できる最大メッセージサイズを定義します。
`> 0`	`I_MPI_DAPL_CHECK_MAX_RDMA_SIZE` 環境変数が有効に設定されていると、デフォルトの <バイト数> の値は DAPL 属性 `max_rdma_size` と等しくなります。それ以外のデフォルト値は `MAX_INT` です。

説明

以下のスキームに従ってメッセージの断片化サイズを制御するには、この環境変数を設定します。

- I_MPI_DAPL_CHECK_MAX_RDMA_SIZE 環境変数が無効に設定されていると、インテル® MPI ライブラリーはサイズが <バイト数> より大きいメッセージを断片化します。
- I_MPI_DAPL_CHECK_MAX_RDMA_SIZE 環境変数が有効に設定されていると、インテル® MPI ライブラリーはサイズが <バイト数> の最小値と DAPL 属性 max_rdma_size より大きいメッセージを断片化します。

I_MPI_DAPL_CONN_EVD_SIZE (I_MPI_RDMA_CONN_EVD_SIZE、I_MPI_CONN_EVD_QLEN)

接続のための DAPL イベント・ディスパッチャーのイベントキューのサイズを定義します。

構文

I_MPI_DAPL_CONN_EVD_SIZE=<サイズ>

廃止された構文

I_MPI_RDMA_CONN_EVD_SIZE=<サイズ>
I_MPI_CONN_EVD_QLEN=<サイズ>

引数

<サイズ>	イベントキューの長さを定義します。
> 0	デフォルトの値は、MPI ジョブの 2 * プロセス数 + 32 です。

説明

接続に関連するイベントを処理する DAPL イベント・ディスパッチャーのイベントキューのサイズを定義するため、この環境変数を設定します。この環境変数が設定されると、<サイズ> とプロバイダーから取得した値の最小値がイベントキューのサイズとして使用されます。プロバイダーは、計算された値以上のキューサイズを供給する必要があります。

I_MPI_DAPL_SR_THRESHOLD

DAPL 待機モード用の RDMA パスに送受信を切り替えるしきい値を変更します。

構文

I_MPI_DAPL_SR_THRESHOLD=<引数>

引数

<バイト数>	rdma へ送受信を切り替えるメッセーサイズのしきい値を定義します。
>= 0	デフォルトの <バイト数> 値は、256 バイトです。

説明

DAPL 待機モードでのポイントツーポイント通信に使用されるプロトコルを制御するため、この環境変数を設定します。

- DAPL 送信/受信データ転送操作で使用して送信される <バイト数> 以下のメッセージ。
- <バイト数> より大きいメッセージサイズは、DAPL RDMA WRITE または RDMA WRITE 即時データ転送操作を使用して送信されます。

I_MPI_DAPL_SR_BUF_NUM

送受信パスで DAPL 待機モードの各プロセスのペア向が使用する、内部事前登録バッファーの数を変更します。

構文

I_MPI_DAPL_SR_BUF_NUM=<バッファー数>

引数

<バッファー数>	プロセスグループの各ペア向けに送受信バッファーの数を定義します。
> 0	デフォルト値は 32 です。

説明

各プロセスのペア向けの内部事前登録バッファーの数を変更するには、この環境変数を設定します。

I_MPI_DAPL_RDMA_WRITE_IMM (I_MPI_RDMA_WRITE_IMM)

DAPL 待機モードで即時データ InfiniBand (IB) 拡張子を持つ RDMA 書き込みを有効/無効にします。

構文

I_MPI_DAPL_RDMA_WRITE_IMM=<引数>

廃止された構文

I_MPI_RDMA_WRITE_IMM=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	即時データ IB 拡張子を持つ RDMA 書き込みを有効にします。
disable | no | off | 0	即時データ IB 拡張子を持つ RDMA 書き込みを無効にします。

説明

即時データ IB 拡張子を持つ RDMA 書き込みを利用するには、この環境変数を設定します。この環境変数が設定され、特定の DAPL プロバイダーの属性が即時データ IB 拡張を持つ RDMA 書き込みをサポートしていることを示す場合、アルゴリズムが有効になります。

I_MPI_DAPL_DESIRED_STATIC_CONNECTIONS_NUM

同時に DAPL スタティック接続を確立するプロセス数を定義します。

構文

I_MPI_DAPL_DESIRED_STATIC_CONNECTIONS_NUM=<プロセス数>

引数

<プロセス数>	同時に DAPL スタティック接続を確立するプロセス数を定義します。
> 0	デフォルトの <プロセス数> 値は 256 です。

説明

DAPL スタティック接続確立のアルゴリズムを制御するには、この環境変数を設定します。

MPI ジョブのプロセス数が <プロセス数> 以下の場合、すべての MPI プロセスは同時にスタティック接続を確立します。それ以外の場合、プロセスはいくつかのグループに分散されます。各グループのプロセス数は、<プロセス数> に近くなるように計算されます。その後、スタティック接続は (グループ間の接続設定を含む) いくつかの反復で確立されます。

I_MPI_CHECK_DAPL_PROVIDER_COMPATIBILITY

すべてのランクで同じ DAPL プロバイダーが選択されているかチェックするのを有効/無効にします。

構文

`I_MPI_CHECK_DAPL_PROVIDER_COMPATIBILITY=<引数>`

引数

<引数>	バイナリー・インジケーター。
`enable | yes | on | 1`	すべてのランクで同じ DAPL プロバイダーが選択されているかチェックします。これは、デフォルト値です。
`disable | no | off | 0`	すべてのランクで同じ DAPL プロバイダーが選択されているかチェックしません。

説明

すべての MPI ランクで DAPL プロバイダーが選択されているかどうかチェックするには、この環境変数を設定します。このチェックが有効になっていると、インテル® MPI ライブラリーは DAPL プロバイダーの名前とバージョンをチェックします。すべてのランクでこの設定が同一でない場合、インテル® MPI ライブラリーは RDMA パスを選択せずにソケットにフォールバックすることがあります。チェックを無効にすると、`MPI_Init()` の実行時間を短縮できます。これは、多数のプロセスによる MPI ジョブで顕著です。

4.3.4. TCP ネットワーク・ファブリック制御

I_MPI_TCP_NETMASK (I_MPI_NETMASK)

TCP ネットワーク・ファブリック経由の MPI 通信のネットワーク・インターフェイスを選択します。

構文

`I_MPI_TCP_NETMASK=<引数>`

引数

<引数>	ネットワーク・インターフェイスを定義 (文字列)。
`<interface_mnemonic>`	ネットワーク・インターフェイスの略: `ib` または `eth`。
`ib`	IPoIB* ネットワーク・インターフェイスを使用します。
`eth`	Ethernet ネットワーク・インターフェイスを使用します。これは、デフォルト値です。
<インターコネクト名>	ネットワーク・インターフェイス名。 通常は、UNIX* ドライバー名にユニット番号が続きます。
<ネットワーク・アドレス>	ネットワーク・アドレス。末尾ゼロはネットマスクを示します。

<ネットワーク・アドレス/ <ネットマスク>	ネットワーク・アドレス。<ネットマスク> 値には、ネットマスクのレングスを指定します。
<インターフェイスのリスト>	コロンで区切られたネットワーク・アドレスまたは、インターフェイス名のリスト。

説明

TCP ネットワーク・ファブリック経由の MPI 通信のネットワーク・インターフェイスを選択するため、この環境変数を使用します。インターフェイスのリストを指定した場合、ノード上で最初に検出されたインターフェイスが通信に使用されます。

例

- InfiniBand* (IPoIB) ファブリック経由の IP を選択するには、次のように設定します。

  ```
  I_MPI_TCP_NETMASK=ib
  ```

- ソケット通信に特定のネットワーク・インターフェイスを選択するには、次のように設定します。

  ```
  I_MPI_TCP_NETMASK=ib0
  ```

- ソケット通信に特定のネットワークを選択するには、次のように設定します。この設定は、255.255.0.0 ネットマスクを意味します。

  ```
  I_MPI_TCP_NETMASK=192.169.0.0
  ```

- ネットマスクが設定されたソケット通信に特定のネットワークを選択するには、次のように設定します。

  ```
  I_MPI_TCP_NETMASK=192.169.0.0/24
  ```

- ソケット通信に特定のネットワーク・インターコネクトを選択するには、次のように設定します。

  ```
  I_MPI_TCP_NETMASK=192.169.0.5/24:ib0:192.169.0.0
  ```

I_MPI_TCP_BUFFER_SIZE

TCP ソケットバッファーのサイズを変更します。

構文

```
I_MPI_TCP_BUFFER_SIZE=<バイト数>
```

引数

<バイト数>	TCP ソケットバッファーのサイズを定義します。
> 0	デフォルトの <バイト数> 値は、128KB です。

説明

TCP ソケットバッファーのサイズを定義するには、この環境変数を設定します。

指定するプロセス数にアプリケーションのパフォーマンスをチューニングするには、I_MPI_TCP_BUFFER_SIZE 環境変数を使用します。

注意

TCP ソケットのバッファーサイズが大きい場合、プロセス数が多いとアプリケーションはより多くのメモリーを必要とします。小さなサイズの TCP ソケットバッファーは、特に 10GB イーサーネットや IProIB では帯域幅を大幅に軽減できます (詳細は、「I_MPI_TCP_NETMASK」をご覧ください)。

4.4. 集団操作制御

インテル® MPI ライブラリーの集団操作では、いくつかの通信アルゴリズムがサポートされます。高度に最適化されたデフォルト設定に加え、ライブラリーは明示的にアルゴリズムを選択する機能 (次の章で説明する I_MPI_ADJUST 環境変数ファミリー) を提供します。

これら環境変数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。

インテル® MPI ライブラリーの集団操作では、いくつかの通信アルゴリズムがサポートされます。高度に最適化されたデフォルト設定に加え、ライブラリーはアルゴリズムの選択を明示的に制御する方法 (I_MPI_ADJUST 環境変数ファミリーと廃止された I_MPI_MSG 環境変数ファミリー) を提供します。これらについては、次の章で説明します。

この環境変数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。

4.4.1. I_MPI_ADJUST ファミリー

I_MPI_ADJUST_<opname>

集団操作のアルゴリズムを設定します。

構文

```
I_MPI_ADJUST_<opname>="<アルゴリズムの ID>[:<条件>][;<アルゴリズムの ID>:<条件>[...]]"
```

引数

<アルゴリズムの ID>	アルゴリズムの識別子。
>= 0	デフォルトの 0 は、最適なデフォルト設定を選択します。
<条件>	カンマで区切った条件。空のリストは、すべてのメッセージとプロセスの組み合わせを選択します。
<l>	サイズ <l> のメッセージ。
<l>-<m>	<l> 以上 <m> 以下のメッセージサイズ。
<l>@<p>	<l> のメッセージサイズと <p> のプロセス数。
<l>-<m>@<p>-<q>	<l> 以上 <m> 以下のメッセージサイズと <p> から <q> のプロセス数。

説明

特定の条件下で、集団操作 <opname> で必要とするアルゴリズムを選択するには、この環境変数を設定します。それぞれの集団操作は、個別の環境変数とアルゴリズムを持ちます。

表 4.4-1 環境変数、集団操作、およびアルゴリズム

環境変数	集団操作	アルゴリズム
I_MPI_ADJUST_ALLGATHER	MPI_Allgather	1. 二重再帰 2. Bruck 3. リング 4. トポロジーを意識した Gatherv + Bcast 5. Knomial
I_MPI_ADJUST_ALLGATHERV	MPI_Allgatherv	1. 二重再帰 2. Bruck 3. リング 4. トポロジーを意識した Gatherv + Bcast
I_MPI_ADJUST_ALLREDUCE	MPI_Allreduce	1. 二重再帰 2. Rabenseifner 3. Reduce + Bcast 4. トポロジーを意識した Reduce + Bcast 5. 二項 gather + scatter 6. トポロジーを意識した二項 gather + scatter 7. Shumilin のリング 8. リング 9. Knomial
I_MPI_ADJUST_ALLTOALL	MPI_Alltoall	1. Bruck 2. Isend/Irecv + waitall 3. ペアごとの交換 4. Plum
I_MPI_ADJUST_ALLTOALLV	MPI_Alltoallv	1. Isend/Irecv + waitall 2. Plum
I_MPI_ADJUST_ALLTOALLW	MPI_Alltoallw	Isend/Irecv + waitall

I_MPI_ADJUST_BARRIER	MPI_Barrier	1. 普及 2. 二重再帰 3. トポロジーを意識した普及 4. トポロジーを意識した二重再帰 5. 二項 gather + scatter 6. トポロジーを意識した二項 gather + scatter
I_MPI_ADJUST_BCAST	MPI_Bcast	1. 二項 2. 二重再帰 3. リング 4. トポロジーを意識した二項 5. トポロジーを意識した二重再帰 6. トポロジーを意識したリング 7. Shumilin 8. Knomial
I_MPI_ADJUST_EXSCAN	MPI_Exscan	1. 部分的な結果収集 2. プロセスのレイアウトに関連する部分的な結果収集
I_MPI_ADJUST_GATHER	MPI_Gather	1. 二項 2. トポロジーを意識した二項 3. Shumilin
I_MPI_ADJUST_GATHERV	MPI_Gatherv	1. 線形 2. トポロジーを意識した線形 3. Knomial
I_MPI_ADJUST_REDUCE_SCATTER	MPI_Reduce_scatter	1. 二分再帰 2. ペアごとの交換 3. 二重再帰 4. Reduce + Scatterv 5. トポロジーを意識した Reduce + Scatterv

I_MPI_ADJUST_REDUCE	MPI_Reduce	1. Shumilin 2. 二項 3. トポロジーを意識した Shumilin 4. トポロジーを意識した二項 5. Rabenseifner 6. トポロジーを意識した Rabenseifner 7. Knomial
I_MPI_ADJUST_SCAN	MPI_Scan	1. 部分的な結果収集 2. トポロジーを意識した部分的な結果収集
I_MPI_ADJUST_SCATTER	MPI_Scatter	1. 二項 2. トポロジーを意識した二項 3. Shumilin
I_MPI_ADJUST_SCATTERV	MPI_Scatterv	1. 線形 2. トポロジーを意識した線形
I_MPI_ADJUST_IALLGATHER	MPI_Iallgather	1. 二重再帰 2. Bruck 3. リング
I_MPI_ADJUST_IALLGATHERV	MPI_Iallgatherv	1. 二重再帰 2. Bruck 3. リング
I_MPI_ADJUST_IALLREDUCE	MPI_Iallreduce	1. 二重再帰 2. Rabenseifner 3. Reduce + Bcast 4. リング (Patarasuk) 5. Knomial 6. 二項
I_MPI_ADJUST_IALLTOALL	MPI_Ialltoall	1. Bruck 2. Isend/Irecv + waitall 3. ペアごとの交換
I_MPI_ADJUST_IALLTOALLV	MPI_Ialltoallv	Isend/Irecv + waitall
I_MPI_ADJUST_IALLTOALLW	MPI_Ialltoallw	Isend/Irecv + waitall

I_MPI_ADJUST_IBARRIER	MPI_Ibarrier	普及
I_MPI_ADJUST_IBCAST	MPI_Ibcast	1. 二項 2. 二重再帰 3. リング 4. Knomial
I_MPI_ADJUST_IEXSCAN	MPI_Iexscan	二重再帰
I_MPI_ADJUST_IGATHER	MPI_Igather	1. 二項 2. Knomial
I_MPI_ADJUST_IGATHERV	MPI_Igatherv	線形
I_MPI_ADJUST_IREDUCE_SCATTER	MPI_Ireduce_scatter	1. 二分再帰 2. ペアごと 3. 二重再帰
I_MPI_ADJUST_IREDUCE	MPI_Ireduce	1. Rabenseifner 2. 二項 3. Knomial
I_MPI_ADJUST_ISCAN	MPI_Iscan	二重再帰
I_MPI_ADJUST_ISCATTER	MPI_Iscatter	1. 二項 2. Knomial
I_MPI_ADJUST_ISCATTERV	MPI_Iscatterv	線形

集団操作のメッセージサイズを算出する規則は、表に記載されています。次の表で、「n/a」は、対応する間隔 <l>-<m> は省略されることを意味します。

表 4.4-2 メッセージ集団関数

集団操作	メッセージサイズ式
MPI_Allgather	recv_count*recv_type_size
MPI_Allgatherv	total_recv_count*recv_type_size
MPI_Allreduce	count*type_size
MPI_Alltoall	send_count*send_type_size
MPI_Alltoallv	n/a
MPI_Alltoallw	n/a
MPI_Barrier	n/a

MPI_Bcast	count*type_size
MPI_Exscan	count*type_size
MPI_Gather	recv_count*recv_type_size　MPI_IN_PLACE が使用される場合。それ以外は send_count*send_type_size。
MPI_Gatherv	n/a
MPI_Reduce_scatter	total_recv_count*type_size
MPI_Reduce	count*type_size
MPI_Scan	count*type_size
MPI_Scatter	send_count*send_type_size　MPI_IN_PLACE が使用される場合。それ以外は recv_count*recv_type_size。
MPI_Scatterv	n/a

例

MPI_Reduce 操作向けの第 2 のアルゴリズムを選択するには、次のように設定します。

```
I_MPI_ADJUST_REDUCE=2
```

MPI_scatter 操作向けのアルゴリズムを定義するには、次のように設定します。

```
I_MPI_ADJUST_REDUCE_SCATTER="4:0-100,5001-10000;1:101-3200,2:3201-5000;3"
```

この場合、アルゴリズム 4 はメッセージサイズ 0 から 100 バイトおよび 5001 から 10000 バイトを使用し、アルゴリズム 1 はメッセージサイズ 101 から 3200 バイトを使用し、アルゴリズム 2 はメッセージサイズ 3201 から 5000 バイトを使用し、アルゴリズム 3 がそれ以外のメッセージを処理します。

I_MPI_ADJUST_REDUCE_SEGMENT

構文

I_MPI_ADJUST_REDUCE_SEGMENT=<ブロックサイズ>|<アルゴリズムの ID>:<ブロックサイズ>[,<アルゴリズムの ID>:<ブロックサイズ>[...]]

引数

<アルゴリズムの ID>	アルゴリズムの識別子。
1	Shumilin アルゴリズム。
3	トポロジーを意識した Shumilin アルゴリズム。
<ブロックサイズ>	メッセージセグメントのサイズをバイト単位で指定します。
> 0	デフォルト値は 14000 です。

説明

指定されたアルゴリズム向けの MPI_Reduce メッセージのセグメント化を制御するため、内部ブロックサイズを設定します。もし、<アルゴリズムの ID> 値が設定されていなければ、<ブロックサイズ> 値は関連するすべてのアルゴリズムに適用されます。

注意

この環境変数は、Shumilin とトポロジーを意識した Shumilin アルゴリズムにのみ関連します (アルゴリズム N1 とアルゴリズム N3 相当)。

I_MPI_ADJUST_BCAST_SEGMENT

構文

`I_MPI_ADJUST_BCAST_SEGMENT=<ブロックサイズ>|<アルゴリズムの ID>:<ブロックサイズ>[,<アルゴリズムの ID>:<ブロックサイズ>[...]]`

引数

<アルゴリズムの ID>	アルゴリズムの識別子。
1	二項アルゴリズム。
4	トポロジーを意識した二項アルゴリズム。
7	Shumilin アルゴリズム。
8	Knomial アルゴリズム。
<ブロックサイズ>	メッセージセグメントのサイズをバイト単位で指定します。
> 0	デフォルト値は 12288 です。

説明

指定されたアルゴリズム向けの `MPI_Bcast` メッセージのセグメント化を制御するため、内部ブロックサイズを設定します。もし、<アルゴリズムの ID> 値が設定されていなければ、<ブロックサイズ> 値は関連するすべてのアルゴリズムに適用されます。

注意

この環境変数は、二項、トポロジーを意識した二項、Shumilin、および Knomial アルゴリズムにのみ関連します。

I_MPI_ADJUST_ALLGATHER_KN_RADIX

構文

`I_MPI_ADJUST_ALLGATHER_KN_RADIX=<基数>`

引数

<基数>	Knomial `MPI_Allgather` アルゴリズムで Knomial 通信ツリーの構築に使用される基数を指定する整数値。
> 1	デフォルト値は 2 です。

説明

この環境変数を `I_MPI_ADJUST_ALLGATHER=5` とともに設定し、対応する `MPI_Allgather` アルゴリズム向けの Knomial ツリーの基数を選択します。

I_MPI_ADJUST_BCAST_KN_RADIX

構文

I_MPI_ADJUST_BCAST_KN_RADIX=<基数>

引数

<基数>	Knomial MPI_Bcast アルゴリズムで Knomial 通信ツリーの構築に使用される基数を指定する整数値。
> 1	デフォルト値は 4 です。

説明

この環境変数を I_MPI_ADJUST_BCAST=8 とともに設定し、対応する MPI_Bcast アルゴリズム向けの Knomial ツリーの基数を選択します。

I_MPI_ADJUST_ALLREDUCE_KN_RADIX

構文

I_MPI_ADJUST_ALLREDUCE_KN_RADIX=<基数>

引数

<基数>	Knomial MPI_Allreduce アルゴリズムで Knomial 通信ツリーの構築に使用される基数を指定する整数値。
> 1	デフォルト値は 4 です。

説明

この環境変数を I_MPI_ADJUST_ALLREDUCE=9 とともに設定し、対応する MPI_Allreduce アルゴリズム向けの Knomial ツリーの基数を選択します。

I_MPI_ADJUST_REDUCE_KN_RADIX

構文

I_MPI_ADJUST_REDUCE_KN_RADIX=<基数>

引数

<基数>	Knomial MPI_Reduce アルゴリズムで Knomial 通信ツリーの構築に使用される基数を指定する整数値。
> 1	デフォルト値は 4 です。

説明

この環境変数を I_MPI_ADJUST_REDUCE=7 とともに設定し、対応する MPI_Reduce アルゴリズム向けの Knomial ツリーの基数を選択します。

I_MPI_ADJUST_GATHERV_KN_RADIX

構文

I_MPI_ADJUST_GATHERV_KN_RADIX=<基数>

引数

<基数>	Knomial MPI_Gatherv アルゴリズムで Knomial 通信ツリーの構築に使用される基数を指定する整数値。
> 1	デフォルト値は 2 です。

説明

この環境変数を I_MPI_ADJUST_GATHERV=3 とともに設定し、対応する MPI_Gatherv アルゴリズム向けの Knomial ツリーの基数を選択します。

I_MPI_ADJUST_IALLREDUCE_KN_RADIX

構文

I_MPI_ADJUST_IALLREDUCE_KN_RADIX=<基数>

引数

<基数>	Knomial MPI_Iallreduce アルゴリズムで Knomial 通信ツリーの構築に使用される基数を指定する整数値。
> 1	デフォルト値は 4 です。

説明

この環境変数を I_MPI_ADJUST_IALLREDUCE=5 とともに設定し、対応する MPI_Iallreduce アルゴリズム向けの Knomial ツリーの基数を選択します。

I_MPI_ADJUST_IBCAST_KN_RADIX

構文

I_MPI_ADJUST_IBCAST_KN_RADIX=<基数>

引数

<基数>	Knomial MPI_Ibcast アルゴリズムで Knomial 通信ツリーの構築に使用される基数を指定する整数値。
> 1	デフォルト値は 4 です。

説明

この環境変数を I_MPI_ADJUST_IBCAST=4 とともに設定し、対応する MPI_Ibcast アルゴリズム向けの Knomial ツリーの基数を選択します。

I_MPI_ADJUST_IREDUCE_KN_RADIX

構文

I_MPI_ADJUST_IREDUCE_KN_RADIX=<基数>

引数

<基数>	Knomial MPI_Ireduce アルゴリズムで Knomial 通信ツリーの構築に使用される基数を指定する整数値。
> 1	デフォルト値は 4 です。

説明

この環境変数を I_MPI_ADJUST_IREDUCE=3 とともに設定し、対応する MPI_Ireduce アルゴリズム向けの Knomial ツリーの基数を選択します。

I_MPI_ADJUST_IGATHER_KN_RADIX

構文

I_MPI_ADJUST_IGATHER_KN_RADIX=<基数>

引数

<基数>	Knomial MPI_Igather アルゴリズムで Knomial 通信ツリーの構築に使用される基数を指定する整数値。
> 1	デフォルト値は 4 です。

説明

この環境変数を I_MPI_ADJUST_IGATHER=2 とともに設定し、対応する MPI_Igather アルゴリズム向けの Knomial ツリーの基数を選択します。

I_MPI_ADJUST_ISCATTER_KN_RADIX

構文

```
I_MPI_ADJUST_ISCATTER_KN_RADIX=<基数>
```

引数

<基数>	Knomial MPI_Iscatter アルゴリズムで Knomial 通信ツリーの構築に使用される基数を指定する整数値。
> 1	デフォルト値は 4 です。

説明

この環境変数を I_MPI_ADJUST_ISCATTER=2 とともに設定し、対応する MPI_Iscatter アルゴリズム向けの Knomial ツリーの基数を選択します。

4.4.2. I_MPI_MSG ファミリー

これらの環境変数は廃止されており、下位互換性を提供するためにのみサポートされます。可能な限り I_MPI_ADJUST 環境変数ファミリーを使用します。

I_MPI_FAST_COLLECTIVES

特定の状況に応じて、適切な集団操作アルゴリズムの選択時にライブラリーのデフォルト動作を制御します。

構文

I_MPI_FAST_COLLECTIVES=<引数>

引数

<引数>	バイナリー・インジケーター。
enable | yes | on | 1	高速集団アルゴリズムが使用されます。これは、デフォルト値です。
disable | no | off | 0	低速で安全な集団アルゴリズムが使用されます。

説明

インテル® MPI ライブラリーは、デフォルトでアプリケーションのパフォーマンス向けの設計された、高度な集団操作アルゴリズムを使用します。実装は次のことを前提とします。

- プロセスのレイアウトやほかの可能性を活用するため、集団操作の実行の順番に関して、MPI 標準の柔軟性を利用しても安全であること。
- 追加の内部バッファーを割り当てるため、十分なメモリーが利用可能なこと。

物理プロセスのレイアウトやその他の要因に依存しない結果を得るには、I_MPI_FAST_COLLECTIVE 環境変数を無効に設定します。

注意

この環境変数によって制御される最適化のいくつかは、試行錯誤的なものです。障害が発生した場合、集団操作の最適化を off にして再度実行してください。

I_MPI_BCAST_NUM_PROCS

MPI_Bcast アルゴリズムのしきい値を制御します。

構文

I_MPI_BCAST_NUM_PROCS=<プロセス数>

引数

<プロセス数>	MPI_Bcast アルゴリズムを選択するプロセス数の閾値を定義します。
> 0	デフォルト値は 8 です。

I_MPI_BCAST_MSG

MPI_Bcast アルゴリズムのしきい値を制御します。

構文

I_MPI_BCAST_MSG=<バイト数 1,バイト数 2>

引数

<バイト数 1,バイト数 2>	MPI_Bcast アルゴリズム を選択するため、メッセージサイズのしきい値の範囲をバイト単位で定義します。
> 0 バイト数 2 >= バイト数 1	デフォルト値は 12288,524288 です。

説明

以下のスキームに従って、利用可能な 3 つの MPI_Bcast アルゴリズムを選択するには、この環境変数を設定します (アルゴリズムの詳細は、表 4.4-1 をご覧ください)。

最初のアルゴリズムは、メッセージサイズが <バイト数 1> 未満か、操作するプロセス数が <プロセス数> 未満の場合に選択されます。

2 番目のアルゴリズムは、メッセージサイズが <バイト数 1> 以上 <バイト数 2> 未満で、操作するプロセス数が 2 の累乗である場合に選択されます。

上記の条件が満たされない場合、3 番目のアルゴリズムが選択されます。

I_MPI_ALLTOALL_NUM_PROCS

MPI_Alltoall アルゴリズムのしきい値を制御します。

構文

I_MPI_ALLTOALL_NUM_PROCS=<プロセス数>

引数

<プロセス数>	MPI_Alltoall アルゴリズムを選択するプロセス数のしきい値を定義します。
> 0	デフォルト値は 8 です。

I_MPI_ALLTOALL_MSG

MPI_Alltoall アルゴリズムのしきい値を制御します。

構文

I_MPI_ALLTOALL_MSG=<バイト数 1,バイト数 2>

引数

<バイト数 1,バイト数 2>	MPI_Alltoall アルゴリズム を選択するため、メッセージサイズのしきい値の範囲をバイト単位で定義します。
> 0 バイト数 2 >= バイト数 1	デフォルト値は 256,32768 です。

説明

以下のスキームに従って、利用可能な 3 つの MPI_Alltoall アルゴリズムを選択するには、この環境変数を設定します (アルゴリズムの詳細は、表 4.4-1 をご覧ください)。

最初のアルゴリズムは、メッセージサイズが <バイト数 1> 以上で、操作するプロセス数が <プロセス数> 未満でない場合に選択されます。

2 番目のアルゴリズムは、メッセージサイズが <バイト数 1> より大きく <バイト数 2> 以下の場合、またはメッセージサイズが <バイト数 2> 未満で操作するプロセス数が <プロセス数> 未満の場合に選択されます。

上記の条件が満たされない場合、3 番目のアルゴリズムが選択されます。

I_MPI_ALLGATHER_MSG

MPI_Allgather アルゴリズムのしきい値を制御します。

構文

I_MPI_ALLGATHER_MSG=<バイト数 1,バイト数 2>

引数

<バイト数 1,バイト数 2>	MPI_Allgather アルゴリズム を選択するため、メッセージサイズのしきい値の範囲をバイト単位で定義します。
> 0 バイト数 2 >= バイト数 1	デフォルト値は 81920,524288 です。

説明

以下のスキームに従って、利用可能な 3 つの MPI_Allgather アルゴリズムを選択するには、この環境変数を設定します (アルゴリズムの詳細は、表 4.4-1 をご覧ください)。

最初のアルゴリズムは、メッセージサイズが <バイト数 2> 未満で、操作するプロセス数が 2 の累乗である場合に選択されます。

2 番目のアルゴリズムは、メッセージサイズが <バイト数 1> 未満で、操作するプロセス数が 2 の累乗でない場合に選択されます。

上記の条件が満たされない場合、3 番目のアルゴリズムが選択されます。

I_MPI_ALLREDUCE_MSG

MPI_Allreduce アルゴリズムのしきい値を制御します。

構文

I_MPI_ALLREDUCE_MSG=<バイト数>

引数

<バイト数>	MPI_Allreduce アルゴリズム を選択するため、メッセージサイズのしきい値の範囲をバイト単位で定義します。
> 0	デフォルト値は 2048 です。

説明

以下のスキームに従って、利用可能な 2 つの MPI_Allreduce アルゴリズムを選択するには、この環境変数を設定します (アルゴリズムの詳細は、表 4.4-1 をご覧ください)。

最初のアルゴリズムは、メッセージサイズが <バイト数> 以下の場合、ユーザー定義のリダクション操作が使用されている場合、または <count> 引数がプロセス数以下の 2 の累乗の近似値より小さい場合に選択されます。

上記の条件が満たされない場合、2 番目のアルゴリズムが選択されます。

I_MPI_REDSCAT_MSG

MPI_Reduce_scatter アルゴリズムのしきい値を制御します。

構文

I_MPI_REDSCAT_MSG=<バイト数 1,バイト数 2>

引数

<バイト数>	MPI_Reduce_scatter アルゴリズム を選択するため、メッセージサイズのしきい値の範囲をバイト単位で定義します。
> 0	デフォルト値は 512,524288 です。

説明

3 つの MPI_Reduce_scatter アルゴリズムの選択を制御するには、この環境変数を設定します。アルゴリズムは、次のスキームに従います (アルゴリズムの詳細は、表 4.4-1 をご覧ください)。

リダクション操作が可換であり、メッセージサイズが <バイト数 2> 未満の場合、最初のアルゴリズムが選択されます。

2 番目のアルゴリズムは、リダクション操作が可換であり、メッセージサイズが <バイト数 2> 以上の場合、またはリダクション操作が可換でなく、メッセージサイズが <バイト数 1> 以上の場合に選択されます。

上記の条件が満たされない場合、3 番目のアルゴリズムが選択されます。

I_MPI_SCATTER_MSG

MPI_Scatter アルゴリズムのしきい値を制御します。

構文

I_MPI_SCATTER_MSG=<バイト数>

引数

<バイト数>	MPI_Scatter アルゴリズム を選択するため、バッファーサイズのしきい値の範囲をバイト単位で定義します。
> 0	デフォルト値は 2048 です。

説明

以下のスキームに従って、利用可能な 2 つの MPI_Scatter アルゴリズムを選択するには、この環境変数を設定します (アルゴリズムの詳細は、表 4.4-1 をご覧ください)。

メッセージサイズが <バイト数> より大きい場合、最初のアルゴリズムがコミュニケーター間で選択されます。上記の条件が満たされない場合、2 番目のアルゴリズムが選択されます。

I_MPI_GATHER_MSG

MPI_Gather アルゴリズムのしきい値を制御します。

構文

I_MPI_GATHER_MSG=<バイト数>

引数

<バイト数>	MPI_Gather アルゴリズム を選択するため、バッファーサイズのしきい値の範囲をバイト単位で定義します。
> 0	デフォルト値は 2048 です。

説明

以下のスキームに従って、利用可能な 2 つの MPI_Gather アルゴリズムを選択するには、この環境変数を設定します (アルゴリズムの詳細は、表 4.4-1 をご覧ください)。

メッセージサイズが <バイト数> より大きい場合、最初のアルゴリズムがコミュニケーター間で選択されます。上記の条件が満たされない場合、2 番目のアルゴリズムが選択されます。

4.5. その他

このセクションでは、次のような情報を提供します。

- 互換性制御
- ダイナミック・プロセスのサポート
- 統計収集モード
- ILP64 サポート
- ユニファイド・メモリー管理

4.5.1. 互換性制御

I_MPI_COMPATIBILITY

ランタイムの互換性モードを選択します。

構文

`I_MPI_COMPATIBILITY=<値>`

引数

<値>	互換性モードを定義します。
定義しない	MPI-3.0 標準互換。これは、デフォルト値です。
3	インテル® MPI ライブラリー 3.x 互換モード。
4	インテル® MPI ライブラリー 4.0.x 互換モード。

説明

インテル® MPI ライブラリー・ランタイムの互換性モードを選択するには、この環境変数を設定します。デフォルトでは、MPI-3.0 標準互換でコンパイルされます。MPI-2.1 の機能を使用している場合は、`I_MPI_COMPATIBILITY` 環境変数を 4 に設定します。MPI-2.1 以前の機能を使用している場合は、`I_MPI_COMPATIBILITY` 環境変数を 3 に設定します。

4.5.2. ダイナミック・プロセスのサポート

インテル® MPI ライブラリーは、MPI アプリケーションが起動された後に、プロセスの生成と強調終了を可能にする MPI-2 のプロセスモデルをサポートしています。以下を提供します。

- 新規に生成されたプロセスと既存の MPI アプリケーション間の通信を確立するメカニズム
- 2 つの既存の MPI アプリケーションで一方が他方をスポーンしなくても通信を確立する、プロセスをアタッチするメカニズム

<マシンファイル> 内で示されるホスト (詳細は、「mpiexec」をご覧ください) は、スポーンされたプロセスを配置する際に使用されます。スポーンされたプロセスは、ラウンドロビン方式またはホストごとに異なるホストに配置されます。最初にスポーンされたプロセスは、親グループの最後のプロセスの後に配置されます。通常のファブリック選択アルゴリズムを使用して、特定のネットワーク・ファブリックの組み合わせが選択されます (詳細は、「I_MPI_FABRICS」と「I_MPI_FABRICS_LIST」をご覧ください)。

例えば、ダイナミック・アプリケーションを実行するするには、次のコマンドを使用します。

```
> mpiexec -n 1 -machinefile smpd.hosts -gwdir <実行形式へのパス> -genv I_MPI_FABRICS shm:tcp <スポーンされたアプリケーション>
```

この例では、spawn_app は 4 つのダイナミック・プロセスをスポーンします。

smpd.hosts が以下を含んでいる場合:

```
host1
host2
host3
host4
```

次のようにダイナミックにプロセスが分散されている場合、元のスポーンを行うプロセスは host1 に配置されます。

1 - host2、2 - host3、3 - host4、そして 4 - 再び host1。

smpd.hosts が以下を含んでいる場合:

```
host1:2
host2:2
```

次のようにダイナミックにプロセスが分散されている場合、通常のプロセスは host1 に配置されます:
1 - host1、2 と 3 - host2、そして 4 - host1。

クライアント・サーバー型のアプリケーションを実行するには、サーバーホスト上で次のコマンドラインを実行します。

```
> mpiexec -n 1 -genv I_MPI_FABRICS shm:tcp <サーバー・アプリケーション> > <ポート名>
```

そして、対応するクライアントで次のコマンドを実行します。

```
> mpiexec -n 1 -genv I_MPI_FABRICS shm:tcp <クライアント・アプリケーション> < <ポート名>
```

単純な MPI_COMM_JOIN ベースのアプリケーションを実行するには、ホスト上で次のコマンドラインを実行します。

```
>  mpiexec -n 1 -genv I_MPI_FABRICS shm:tcp <join サーバー・アプリケーション> < <ポート名>
>  mpiexec -n 1 -genv I_MPI_FABRICS shm:tcp <join クライアント・アプリケーション> < <ポート名>
```

4.5.3. 統計収集モード

ここでは、インテル® MPI ライブラリーの統計収集モデルと、環境変数を介してどのように収集を行うか説明します。インテル® MPI ライブラリーは、次の統計形式をサポートします。

- ネイティブ統計形式
- IPM 統計形式

ネイティブ統計形式については「ネイティブ統計形式」を、IPM 統計形式については「IPM 統計形式」をご覧ください。両方の統計タイプを収集する可能性もあります。詳細は、「ネイティブと IPM 統計」をご覧ください。

ネイティブ統計形式

インテル® MPI ライブラリーは、アプリケーションの実行を妨げることなく、パフォーマンス・データを収集する組込み統計収集機能を持っています。収集された情報はテキストファイルに書き込まれます。ここでは、組込み統計収集機能を制御するために利用できる環境変数、およびプロバイダーの出力ファイル例について説明します。

I_MPI_STATS

統計収集を制御します。

構文

```
I_MPI_STATS=[native:][n-] m
```

引数

n, m	出力情報の統計レベル。
1	各プロセスが送信したデータ量を出力。
2	セル数と転送されたデータ量を出力。
3	統計出力は、実際の引数に応じて組み合わせられます。
4	バケットリストで定義された統計出力。
10	すべての通信コンテキストの集団操作の統計出力。
20	すべての MPI 関数の時間情報を追加して出力。

説明

この環境変数は、収集する統計情報の量とファイルへのログ出力を制御します。統計出力はデフォルトではありません。

注意

n,m は正の整数値です。出力する情報の範囲を定義します。レベル n から レベル m の統計 (n と m を含む) が出力されます。n が指定されない場合、デフォルト値は 1 です。

I_MPI_STATS_SCOPE

統計情報を収集するサブシステムを選択します。

構文

```
I_MPI_STATS_SCOPE="<サブシステム>[:<ops>][;<サブシステム>[:<ops>][...]]"
```

引数

<サブシステム>	ターゲットのサブシステムを定義します。
all	すべての操作の統計データを収集します。これは、デフォルト値です。
coll	すべての集団操作の統計データを収集します。
p2p	すべてのポイントツーポイント操作の統計データを収集します。

<ops>	カンマで区切られたターゲット操作のリストを定義します。
Allgather	MPI_Allgather
Iallgather	MPI_Iallgather

Allgatherv	MPI_Allgatherv
Iallgatherv	MPI_Iallgatherv
Allreduce	MPI_Allreduce
Iallreduce	MPI_Iallreduce
Alltoall	MPI_Alltoall
Ialltoall	MPI_Ialltoall
Alltoallv	MPI_Alltoallv
Ialltoallv	MPI_Ialltoallv
Alltoallw	MPI_Alltoallw
Ialltoallw	MPI_Ialltoallw
Barrier	MPI_Barrier
Ibarrier	MPI_Ibarrier
Bcast	MPI_Bcast
Ibcast	MPI_Ibcast
Exscan	MPI_Exscan
Iexscan	MPI_Iexscan
Gather	MPI_Gather
Igather	MPI_Igather
Gatherv	MPI_Gatherv
Igatherv	MPI_Igatherv
Reduce_scatter	MPI_Reduce_scatter
Ireduce_scatter	MPI_Ireduce_scatter
Reduce	MPI_Reduce
Ireduce	MPI_Ireduce
Scan	MPI_Scan
Iscan	MPI_Iscan
Scatter	MPI_Scatter

Iscatter	MPI_Iscatter
Scatterv	MPI_Scatterv
Iscatterv	MPI_Iscatterv
Send	標準転送 (MPI_Send、MPI_Isend、MPI_Send_init)。
Sendrecv	送信-受信転送 (MPI_Sendrecv、MPI_Sendrecv_replace)。
Bsend	バッファー転送 (MPI_Bsend、MPI_Ibsend、MPI_Bsend_init)。
Csend	集団内部のポイントツーポイント操作。この内部操作はすべての集団を扱います。
Csendrecv	集団内部のポイントツーポイントの送受信操作。この内部操作はすべての集団を扱います。
Rsend	レディ転送 (MPI_Rsend、MPI_Irsend、MPI_Rsend_init)。
Ssend	同時転送 (MPI_Ssend、MPI_Issend、MPI_Ssend_init)。

説明

統計情報を収集するためターゲットのサブシステムを選択するには、この環境変数を設定します。すべての集団とポイントツーポイント操作は内部的に収集を行うため、デフォルトでカバーされます。

例

デフォルト設定は次と等価です。

```
I_MPI_STATS_SCOPE="coll;p2p"
```

MPI_Bcast、MPI_Reduce、そしてすべてのポイントツーポイント操作の統計情報を収集するには、次の設定を行います。

```
I_MPI_STATS_SCOPE="p2p;coll:bcast,reduce"
```

ポイントツーポイント操作内部の統計情報を収集するには、次の設定を行います。

```
I_MPI_STATS_SCOPE=p2p:csend
```

I_MPI_STATS_BUCKETS

統計情報の収集に使用するメッセージのサイズとコミュニケーターのサイズの範囲を示すリストを特定します。

構文

```
I_MPI_STATS_BUCKETS=<メッセージ>[@<プロセス>][,<メッセージ>[@<プロセス>]]...
```

引数

<メッセージ>	メッセージサイズの範囲をバイト単位で指定します。
<l>	単一のメッセージサイズ。
<l>-<m>	範囲 <l> から <m>。

<プロセス>	集合操作のプロセス (ランク) 範囲を指定します。
<p>	単一のコミュニケーター・サイズ。
<p>-<q>	範囲 <p> から <q>。

説明

メッセージサイズとコミュニケーター・サイズの範囲を定義するため、I_MPI_STATS_BUCKETS 環境変数を設定します。

レベル 4 の統計は、これらの範囲のプロファイル情報を提供します。

I_MPI_STATS_BUCKETS 環境変数が設定されていない場合、レベル 4 の統計が収集されます。範囲が指定されていないと、可能な限り最大の範囲が想定されます。

例

短いメッセージ (0 から 1000 バイトまで) と長いメッセージ (50000 から 100000 バイトまで) を指定するには、次のように設定します。

```
-env I_MPI_STATS_BUCKETS 0-1000,50000-100000
```

サイズが 16 バイトで、4 つのプロセス内で循環するメッセージを指定するには、次の設定を行います。

```
-env I_MPI_STATS_BUCKETS "16@4"
```

注意

@ シンボルがある場合、環境変数の値は引用符で囲む必要があります。

I_MPI_STATS_FILE

統計出力ファイル名を定義します。

構文

I_MPI_STATS_FILE=<名前>

引数

<名前>	統計出力ファイル名を定義します。

説明

この環境変数には統計出力ファイルを設定します。デフォルトでは、stats.txt ファイルはカレント・ディレクトリーに作成されます。

この変数が設定されず、統計出力ファイルがすでに存在している場合、ファイル名にインデックスが追加されます。例えば、stats.txt が存在すると、stats(2).txt という統計出力ファイルが作成され、stats(2).txt が存在すれば stats(3).txt が作成されます。

統計データは、MPI_COMM_WORLD コミュニケーターのプロセスランクに応じてブロックおよび順序付けされます。

タイミングデータは、マイクロ秒単位で指定します。例えば、次のようなプログラムについて考えてみます。

```
I_MPI_STATS=4
I_MPI_STATS_SCOPE="p2p;coll:allreduce"
```

MPI_Allreduce 操作のみを行う簡単なプログラムの統計出力は、次のようになります。

```
Intel(R) MPI Library Version 5.1
_______ MPI Communication Statistics _____
Stats level: 4
P2P scope:< FULL >
Collectives scope:< Allreduce >
~~~~ Process 0 of 2 on node svlmpihead01 lifetime = 414.13
Data Transfers
Src Dst Amount(MB) Transfers
---------------------------------------
000 --> 000 0.000000e+00 0
000 --> 001 7.629395e-06 2
=======================================
Totals 7.629395e-06 2
Communication Activity
Operation Volume(MB) Calls
---------------------------------------
P2P
Csend 7.629395e-06 2
Csendrecv 0.000000e+00 0
Send 0.000000e+00 0
Sendrecv 0.000000e+00 0
Bsend 0.000000e+00 0
Rsend 0.000000e+00 0
Ssend 0.000000e+00 0
Collectives
Allreduce 7.629395e-06 2
=======================================
Communication Activity by actual args
P2P
Operation Dst Message size Calls
-------------------------------------------
Csend
1 1 4 2
Collectives
Operation Context Algo Comm size Message size Calls Cost(%)
-----------------------------------------------------------------------------
Allreduce
1 0 1 2 4 2 44.96
=============================================================================
~~~~ Process 1 of 2 on node svlmpihead01 lifetime = 306.13
Data Transfers
Src Dst Amount(MB) Transfers
---------------------------------------
001 --> 000 7.629395e-06 2
001 --> 001 0.000000e+00 0
=======================================
Totals 7.629395e-06 2
```

```
Communication Activity
Operation Volume(MB) Calls
----------------------------------------
P2P
Csend 7.629395e-06 2
Csendrecv 0.000000e+00 0
Send 0.000000e+00 0
Sendrecv 0.000000e+00 0
Bsend 0.000000e+00 0
Rsend 0.000000e+00 0
Ssend 0.000000e+00 0
Collectives
Allreduce 7.629395e-06 2
========================================
Communication Activity by actual args
P2P
Operation Dst Message size Calls
----------------------------------------------
Csend
1 0 4 2
Collectives
Operation Context Comm size Message size Calls Cost(%)
-----------------------------------------------------------------------
Allreduce
1 0 2 4 2 37.93
=======================================================================
____ End of stats.txt file ____
```

上の例では:

- すべての時間は、マイクロ秒で計測されています。
- メッセージサイズはバイトでカウントされます。**MB** は、メガバイトの略で、2^{20} もしくは 1 048 576 バイトと等価です。
- プロセスのライフタイムは、`MPI_Init` と `MPI_Finalize` 間の時間の連続として計算されます。
- **Algo** フィールドは、指定された引数の操作で使用されるアルゴリズムの数を表します。
- **Cost** フィールドは、特定の集合演算の実行時間をプロセスのライフタイムのパーセンテージで表します。

領域制御

インテル® MPI ライブラリーは、オプションの領域機能もサポートします。領域は、IPM 統計形式です。IPM に関する詳細は、「IPM 統計形式」をご覧ください。この機能を使用するには、ソースコードを修正する必要があります。`MPI_Pcontrol` 関数を使用します。

領域は、標準的な `MPI_Pcontrol` 関数呼び出しにより、開始と終了ポイントでマークされたソースコードの一部です。`MPI_Pcontrol` 関数は、次の特殊パーマネント領域には使用できません。

- `MPI_Init` から `MPI_Finalize` までの、すべての MPI 呼び出しの統計情報を含むメイン領域。IPM 統計出力向けに「*」で命名されたメイン領域。この領域のデフォルト出力ファイルは、ネイティブ統計形式の `stats.txt` です。
- 補足領域は、統計情報を含みますが名前付き領域は含まれません。領域は、IPM 統計形式向けの出力で「ipm_noregion」と命名されます。この領域のデフォルト出力ファイルは、ネイティブ統計形式の `stats_noregion.txt` です。

名前付き領域を使用しない場合、メイン領域と補足領域は同一となり、補足領域は無視されます。

各領域は、領域内の MPI 関数呼び出しに関する独立した統計情報を含んでいます。

インテル® MPI ライブラリーは、次の領域タイプをサポートします。

- Discontiguous (不連続) (いくつかのオープンとクローズ)。
- Intersected (交差)。
- MPI プロセスのサブセットをカバーします (MPI_COMM_WORLD 環境変数の一部)。

MPI_Pcontrol(1, <名前>) 呼び出しで開かれた領域は、MPI_Pcontrol(-1, <名前>) 呼び出しで閉じられます。<名前> は、NULL で終わる文字列の領域名です。<名前> は、IPM 統計形式の出力に使用されます。この領域のデフォルト出力ファイルは、ネイティブ統計形式の stats_<名前>.txt です。

開かれているすべての領域は、MPI_Finalize 関数内で自動的にクローズされます。

IPM 統計形式

インテル® MPI ライブラリーは、前述したように、組込み統計収集メカニズムの一部として、統合パフォーマンス・モニタリング (IPM) サマリー形式をサポートしています。この情報を収集するため、ソースコードを変更したり、アプリケーションを再リンクする必要はありません。

I_MPI_STATS_BUCKETS 環境変数は、IPM 形式には適用されません。I_MPI_STATS_ACCURACY 環境変数で、特殊機能を制御できます。

I_MPI_STATS

統計データの出力形式を制御します。

構文

I_MPI_STATS=<レベル>

引数

<レベル>	統計データのレベル。
ipm	すべての領域のサマリーデータ。
ipm:terse	基本サマリーデータ。

説明

領域のサマリーを含む統計情報を出力するには、この環境変数に ipm を設定します。簡単な統計出力を得るには、この環境変数に ipm:terse を設定します。

I_MPI_STATS_FILE

出力ファイル名を定義します。

構文

I_MPI_STATS_FILE=<名前>

引数

<名前>	統計データを収集するためのファイル名。

説明

統計情報出力ファイル名のデフォルト stats.ipm を変更するには、この環境変数を設定します。

この変数が設定されず、統計出力ファイルがすでに存在している場合、ファイル名にインデックスが追加されます。例えば、stats.ipm が存在すると、stats(2).ipm という統計出力ファイルが作成され、stats(2).ipm が存在すれば stats(3).ipm が作成されます。

I_MPI_STATS_SCOPE

統計収集のための MPI 関数のサブセットのリストをカンマで区切って定義します。

構文

I_MPI_STATS_SCOPE="<サブセット>[;<サブセット>[;…]]"

引数

<サブセット>	ターゲットのサブセット。
all2all	all to all 関数タイプの統計データを収集します。
all2one	all to one 関数タイプの統計データを収集します。
attr	属性制御関数の統計データを収集します。
comm	コミュニケーター制御関数の統計データを収集します。
err	エラー制御関数の統計データを収集します。
group	グループサポート関数の統計データを収集します。
init	イニシャライズとファイナライズ関数の統計データを収集します。
io	入力/出力サポート関数の統計データを収集します。
one2all	one to all 関数タイプの統計データを収集します。
recv	受信関数の統計データを収集します。
req	要求サポート関数の統計データを収集します。
rma	一方向通信関数の統計データを収集します。
scan	スキャン集団関数の統計データを収集します。
send	送信関数の統計データを収集します。
sendrecv	送受信関数の統計データを収集します。
serv	追加のサービス関数の統計データを収集します。
spawn	ダイナミック・プロセス関数の統計データを収集します。
status	ステータス制御関数の統計データを収集します。
sync	バリア同期の統計データを収集します。
time	タイミングサポート関数の統計データを収集します。

topo	トポロジーサポート関数の統計データを収集します。
type	データ型サポート関数の統計データを収集します。

説明

次の表で示される統計情報のサブセット、または MPI 関数のサブセットを定義するには、この環境変数を設定します。すべてのサブセットの統合がデフォルトです。

表 4.5-1 MPI 関数の統計サブセット

all2all MPI_Allgather MPI_Allgatherv MPI_Allreduce MPI_Alltoll MPI_Alltoallv MPI_Alltoallw MPI_Reduce_scatter MPI_Iallgather MPI_Iallgatherv MPI_Iallreduce MPI_Ialltoll MPI_Ialltoallv MPI_Ialltoallw MPI_Ireduce_scatter MPI_Ireduce_scatter_block **all2one** MPI_Gather MPI_Gatherv MPI_Reduce MPI_Igather MPI_Igatherv MPI_Ireduce **attr** MPI_Comm_create_keyval MPI_Comm_delete_attr MPI_Comm_free_keyval MPI_Comm_get_attr MPI_Comm_set_attr MPI_Comm_get_name MPI_Comm_set_name MPI_Type_create_keyval MPI_Type_delete_attr MPI_Type_free_keyval MPI_Type_get_attr MPI_Type_get_name MPI_Type_set_attr MPI_Type_set_name MPI_Win_create_keyval MPI_Win_delete_attr MPI_Win_free_keyval MPI_Win_get_attr MPI_Win_get_name MPI_Win_set_attr	**recv** MPI_Recv MPI_Irecv MPI_Recv_init MPI_Probe MPI_Iprobe **req** MPI_Start MPI_Startall MPI_Wait MPI_Waitall MPI_Waitany MPI_Waitsome MPI_Test MPI_Testall MPI_Testany MPI_Testsome MPI_Cancel MPI_Grequest_start MPI_Grequest_complete MPI_Request_get_status MPI_Request_free **rma** MPI_Accumulate MPI_Get MPI_Put MPI_Win_complete MPI_Win_create MPI_Win_fence MPI_Win_free MPI_Win_get_group MPI_Win_lock MPI_Win_post MPI_Win_start MPI_Win_test MPI_Win_unlock MPI_Win_wait MPI_Win_allocate MPI_Win_allocate_shared MPI_Win_create_dynamic MPI_Win_shared_query MPI_Win_attach MPI_Win_detach MPI_Win_set_info

MPI_Win_set_name
MPI_Get_processor_name

comm
MPI_Comm_compare
MPI_Comm_create
MPI_Comm_dup
MPI_Comm_free
MPI_Comm_get_name
MPI_Comm_group
MPI_Comm_rank
MPI_Comm_remote_group
MPI_Comm_remote_size
MPI_Comm_set_name
MPI_Comm_size
MPI_Comm_split
MPI_Comm_test_inter
MPI_Intercomm_create
MPI_Intercomm_merge

err
MPI_Add_error_class
MPI_Add_error_code
MPI_Add_error_string
MPI_Comm_call_errhandler
MPI_Comm_create_errhandler
MPI_Comm_get_errhandler
MPI_Comm_set_errhandler
MPI_Errhandler_free
MPI_Error_class
MPI_Error_string
MPI_File_call_errhandler
MPI_File_create_errhandler
MPI_File_get_errhandler
MPI_File_set_errhandler
MPI_Win_call_errhandler
MPI_Win_create_errhandler
MPI_Win_get_errhandler
MPI_Win_set_errhandler

group
MPI_Group_compare
MPI_Group_difference
MPI_Group_excl
MPI_Group_free
MPI_Group_incl
MPI_Group_intersection
MPI_Group_range_excl
MPI_Group_range_incl
MPI_Group_rank
MPI_Group_size
MPI_Group_translate_ranks
MPI_Group_union

init
MPI_Init
MPI_Init_thread

MPI_Win_get_info
MPI_Win_get_accumulate
MPI_Win_fetch_and_op
MPI_Win_compare_and_swap
MPI_Rput
MPI_Rget
MPI_Raccumulate
MPI_Rget_accumulate
MPI_Win_lock_all
MPI_Win_unlock_all
MPI_Win_flush
MPI_Win_flush_all
MPI_Win_flush_local
MPI_Win_flush_local_all
MPI_Win_sync

scan
MPI_Exscan
MPI_Scan
MPI_Iexscan
MPI_Iscan

send
MPI_Send
MPI_Bsend
MPI_Rsend
MPI_Ssend
MPI_Isend
MPI_Ibsend
MPI_Irsend
MPI_Issend
MPI_Send_init
MPI_Bsend_init
MPI_Rsend_init
MPI_Ssend_init

sendrecv
MPI_Sendrecv
MPI_Sendrecv_replace

serv
MPI_Alloc_mem
MPI_Free_mem
MPI_Buffer_attach
MPI_Buffer_detach
MPI_Op_create
MPI_Op_free

spawn
MPI_Close_port
MPI_Comm_accept
MPI_Comm_connect
MPI_Comm_disconnect
MPI_Comm_get_parent
MPI_Comm_join
MPI_Comm_spawn
MPI_Comm_spawn_multiple

MPI_Finalize

io
MPI_File_close
MPI_File_delete
MPI_File_get_amode
MPI_File_get_atomicity
MPI_File_get_byte_offset
MPI_File_get_group
MPI_File_get_info
MPI_File_get_position
MPI_File_get_position_shared
MPI_File_get_size
MPI_File_get_type_extent
MPI_File_get_view
MPI_File_iread_at
MPI_File_iread
MPI_File_iread_shared
MPI_File_iwrite_at
MPI_File_iwrite
MPI_File_iwrite_shared
MPI_File_open
MPI_File_preallocate
MPI_File_read_all_begin
MPI_File_read_all_end
MPI_File_read_all
MPI_File_read_at_all_begin
MPI_File_read_at_all_end
MPI_File_read_at_all
MPI_File_read_at
MPI_File_read
MPI_File_read_ordered_begin
MPI_File_read_ordered_end
MPI_File_read_ordered
MPI_File_read_shared
MPI_File_seek
MPI_File_seek_shared
MPI_File_set_atomicity
MPI_File_set_info
MPI_File_set_size
MPI_File_set_view
MPI_File_sync
MPI_File_write_all_begin
MPI_File_write_all_end
MPI_File_write_all
MPI_File_write_at_all_begin
MPI_File_write_at_all_end
MPI_File_write_at_all
MPI_File_write_at
MPI_File_write
MPI_File_write_ordered_begin
MPI_File_write_ordered_end
MPI_File_write_ordered
MPI_File_write_shared
MPI_Register_datarep
MPI_Lookup_name
MPI_Open_port
MPI_Publish_name
MPI_Unpublish_name

status
MPI_Get_count
MPI_Status_set_elements
MPI_Status_set_cancelled
MPI_Test_cancelled

sync

MPI_Barrier
MPI_Ibarrier

time
MPI_Wtick
MPI_Wtime

topo
MPI_Cart_coords
MPI_Cart_create
MPI_Cart_get
MPI_Cart_map
MPI_Cart_rank
MPI_Cart_shift
MPI_Cart_sub
MPI_Cartdim_get
MPI_Dims_create
MPI_Graph_create
MPI_Graph_get
MPI_Graph_map
MPI_Graph_neighbors
MPI_Graphdims_get
MPI_Graph_neighbors_count
MPI_Topo_test

type
MPI_Get_address
MPI_Get_elements
MPI_Pack
MPI_Pack_external
MPI_Pack_external_size
MPI_Pack_size
MPI_Type_commit
MPI_Type_contiguous
MPI_Type_create_darray
MPI_Type_create_hindexed
MPI_Type_create_hvector
MPI_Type_create_indexed_block
MPI_Type_create_resized
MPI_Type_create_struct
MPI_Type_create_subarray
MPI_Type_dup
MPI_Type_free
MPI_Type_get_contents
MPI_Type_get_envelope

one2all MPI_Bcast MPI_Scatter MPI_Scatterv MPI_Ibcast MPI_Iscatter MPI_Iscatterv	MPI_Type_get_extent MPI_Type_get_true_extent MPI_Type_indexed MPI_Type_size MPI_Type_vector MPI_Unpack_external MPI_Unpack

I_MPI_STATS_ACCURACY

統計出力を減らすには、I_MPI_STATS_ACCURACY 環境変数を使用します。

構文

I_MPI_STATS_ACCURACY=<パーセント>

引数

<パーセント>	float のしきい値。

説明

すべての MPI 呼び出し内部で費やされた総時間のパーセンテージで示される、大部分の時間を占める MPI 関数のデータを収集するには、この環境変数を設定します。

例

次の例は、簡単なアプリケーションのソースと IPM 統計形式のサマリーを示します。

```
int main (int argc, char *argv[]){
    int i, rank, size, nsend, nrecv;
    MPI_Init (&argc, &argv);
    MPI_Comm_rank (MPI_COMM_WORLD, &rank);
    nsend = rank;
    MPI_Wtime();
    for (i = 0; i < 200; i++){
        MPI_Barrier(MPI_COMM_WORLD);
    }
    /* open "reduce" region for all processes */
    MPI_Pcontrol(1, "reduce");
    for (i = 0; i < 1000; i++)
        MPI_Reduce(&nsend, &nrecv, 1, MPI_INT, MPI_MAX, 0, MPI_COMM_WORLD);
    /* close "reduce" region */
    MPI_Pcontrol(-1, "reduce");
    if (rank == 0){
        /* "send" region for 0-th process only */
        MPI_Pcontrol(1, "send");
        MPI_Send(&nsend, 1, MPI_INT, 1, 1, MPI_COMM_WORLD);
        MPI_Pcontrol(-1, "send");
    }
    if (rank == 1){
        MPI_Recv(&nrecv, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    }
    /* reopen "reduce" region */
    MPI_Pcontrol(1, "reduce");
    for (i = 0; i < 1000; i++)
        MPI_Reduce(&nsend, &nrecv, 1, MPI_INT, MPI_MAX, 0, MPI_COMM_WORLD);
    MPI_Wtime();
```

```
    MPI_Finalize ();
    return 0;
}
```

コマンド:

```
mpiexec -n 4 -env I_MPI_STATS ipm:terse test.exe
```

統計出力:

```
################################################################################
#
# command : unknown (completed)
# host :SVLMPICL704/Windows mpi_tasks :4 on 1 nodes
# start :06/17/11/14:10:40 wallclock :0.037681 sec
# stop :06/17/11/14:10:40 %comm :99.17
# gbytes :0.00000e+000 total gflop/sec :NA
#
################################################################################
```

コマンド:

```
mpiexec -n 4 -env I_MPI_STATS ipm test.exe
```

統計出力:

```
################################################################################
#
# command : unknown (completed)
# host :SVLMPICL704/Windows mpi_tasks :4 on 1 nodes
# start :06/17/11/14:10:40 wallclock :0.037681 sec
# stop :06/17/11/14:10:40 %comm :99.17
# gbytes :0.00000e+000 total gflop/sec :NA
#
################################################################################
# region :* [ntasks] = 4
#
# [total] <avg> min max
# entries 4 1 1 1
# wallclock 0.118763 0.0296908 0.0207312 0.0376814
# user 0.0156001 0.00390002 0 0.0156001
# system 0 0 0 0
# mpi 0.117782 0.0294454 0.0204467 0.0374543
# %comm 99.1735 98.6278 99.3973
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Init 0.0944392 4 80.18 79.52
# MPI_Reduce 0.0183164 8000 15.55 15.42
# MPI_Recv 0.00327056 1 2.78 2.75
# MPI_Barrier 0.00174499 800 1.48 1.47
# MPI_Send 4.23448e-006 1 0.00 0.00
# MPI_Finalize 3.07963e-006 4 0.00 0.00
# MPI_Wtime 1.53982e-006 8 0.00 0.00
# MPI_Comm_rank 1.5398e-006 4 0.00 0.00
# MPI_TOTAL 0.117782 8822 100.00 99.17
################################################################################
```

```
# region : reduce [ntasks] = 4
#
# [total] <avg> min max
# entries 8 2 2 2
# wallclock 0.0190786 0.00476966 0.00273201 0.00665929
# user 0 0 0 0
# system 0 0 0 0
# mpi 0.0183199 0.00457997 0.00255377 0.00643987
# %comm 96.0231 93.4761 97.0543
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Reduce 0.0183164 8000 99.98 96.00
# MPI_Finalize 3.07963e-006 4 0.02 0.02
# MPI_Wtime 3.84956e-007 4 0.00 0.00
# MPI_TOTAL 0.0183199 8008 100.00 96.02
################################################################################
# region : send [ntasks] = 4
#
# [total] <avg> min max
# entries 1 0 0 1
# wallclock 1.22389e-005 3.05971e-006 1e-006 9.23885e-006
# user 0 0 0 0
# system 0 0 0 0
# mpi 4.23448e-006 1.05862e-006 0 4.23448e-006
# %comm 34.5986 0 45.8334
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Send 4.23448e-006 1 100.00 34.60
################################################################################
# region : ipm_noregion [ntasks] = 4
#
# [total] <avg> min max
# entries 13 3 3 4
# wallclock 0.0996611 0.0249153 0.0140604 0.0349467
# user 0.0156001 0.00390002 0 0.0156001
# system 0 0 0 0
# mpi 0.0994574 0.0248644 0.0140026 0.0349006
# %comm 99.7957 99.5893 99.8678
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Init 0.0944392 4 94.95 94.76
# MPI_Recv 0.00327056 1 3.29 3.28
# MPI_Barrier 0.00174499 800 1.75 1.75
# MPI_Comm_rank 1.5398e-006 4 0.00 0.00
# MPI_Wtime 1.15486e-006 4 0.00 0.00
# MPI_TOTAL 0.0994574 813 100.00 99.80
```

ネイティブと IPM 統計

サポートされるぞれぞれの形式は、個別に収集できます。最も詳細なレベルですべての形式の統計情報を収集するには、I_MPI_STAT 環境変数を使用します。

I_MPI_STATS

構文

```
I_MPI_STATS=all
```

注意

両方の形式の統計が収集された場合、I_MPI_STATS_SCOPE 環境変数は適用されません。

統計情報の量を制御するには、I_MPI_STATS にカンマで区切られた値を設定します。

構文

```
I_MPI_STATS=[native:][n-]m,ipm[:terse]
```

注意

現在エリアス `all` は、`I_MPI_STATS=native:20,ipm` に相当しますが、これは変更できます。

4.5.4. ILP64 サポート

ILP64 は、int、long および ポインターが、すべて 8 バイトであることを意味します。これは、long とポインターが 8 バイトで、int が 4 バイトである LP64 モデルとは異なります。歴史的背景とプログラミング・モデルのフィロソフィーに関する詳細は、http://www.unix.org/version2/whatsnew/lp64_wp.html (英語) などをご覧ください。

ILP64 を使用する

ILP64 インターフェイスを使用するには以下のオプションを使用します。

- 分割コンパイルには Fortran コンパイラー・ドライバーのオプション -i8 を、分割リンクには -ilp64 オプションを指定します。

  例:
  ```
  >mpiifort -i8 -c test.f
  >mpiifort -ilp64 -o test test.o
  ```
- 簡単なプログラムでは、コンパイルおよびリンクを行う Fortran コンパイラー・ドライバーに -i8 オプションを指定します。-i8 を指定すると、自動的に ILP64 ライブラリーがリンクされます。

  例:
  ```
  > mpiifort -i8 test.f
  ```

既存の問題と制限事項

- データ型のカウントとその他の引数では、2^{31}-1 を越える値はサポートされません。
- 特殊な MPI 型、MPI_FLOAT_INT、MPI_DOUBLE_INT、MPI_LONG_INT、MPI_SHORT_INT、MPI_2INT、MPI_LONG_DOUBLE_INT、MPI_2INTEGER は変更されず、4 バイトの整数フィールドが維持されます。
- 事前定義されたコミュニケーター属性 MPI_APPNUM、MPI_HOST、MPI_IO、MPI_LASTUSEDCODE、MPI_TAG_UB、MPI_UNIVERSE_SIZE および MPI_WTIME_IS_GLOBAL は、4 バイトの整数として関

数 MPI_GET_ATTR と MPI_COMM_GET_ATTR から返されます。これは、ウィンドウやファイル・オブジェクトに結合できる定義済み属性も同様です。

- エラー処理関数やユーザー定義リダクション操作など MPI コールバック関数をコンパイルする場合、-i8 オプションを指定してはいけません。
- 4 バイトの属性 (MPI_ATTR_GET、MPI_ATTR_PUT など) を保存または取得する非推奨の関数のコンパイルに -i8 オプションを指定してはいけません。代わりに推奨される代替手段 (MPI_COMM_GET_ATTR、MPI_COMM_SET_ATTR など) を使用します。
- インテル® Trace Collector でインテル® MPI の ILP64 実行形式を扱う場合、特殊なインテル® Trace Collector ライブラリーをリンクする必要があります。必要に応じて、インテル® MPI ライブラリーの mpiifort コンパイラー・ドライバーは自動的に適切なインテル® Trace Collector ライブラリーを選択します。
- これは、現在 C と C++ アプリケーションをサポートしていません。

4.5.5. ユニファイド・メモリー管理

インテル® MPI ライブラリーは、ユーザー定義のパッケージでメモリー管理サブシステムを交換する機能をサポートしています。状況に応じて次の関数ポインターを設定することがあります。

- i_malloc
- i_calloc
- i_realloc
- i_free

これらのポインターは、C++ の new と delete 操作に影響します。それぞれ標準 C ライブラリー関数が使用されます。

統合ユニファイド・メモリー・サブシステムを利用するには、アプリケーションを libimalloc.dll とリンクする必要があります。以下に統合メモリー・サブシステムの使用法を示します。

```
#include <i_malloc.h>
#include <my_malloc.h>

int main( int argc, int argv ){
    // override normal pointers
    i_malloc = my_malloc;
    i_calloc = my_calloc;
    i_realloc = my_realloc; i_free = my_free;

    #ifdef _WIN32
    // also override pointers used by DLLs
    i_malloc_dll = my_malloc;
    i_calloc_dll = my_calloc;
    i_realloc_dll = my_realloc;
    i_free_dll = my_free;
    #endif

// now start using Intel(R) libraries
}
```

4.6. ダイナミック・リンク・ライブラリーの安全なロード

インテル® MPI ライブラリーは、ダイナミック・リンク・ライブラリーをロードするため、拡張セキュリティー・オプションを提供します。外部 DLL* を検索するディレクトリーを定義するだけでなく、ダイナミック・ライブラリーを読み込むためセキュリティー・モードを拡張することができます。

セキュリティー・オプションは、HKEY_LOCAL_MACHINE\Software\Intel\MPI の Windows* レジストリー・キーに配置されます。そのため、管理者権限がないアカウントによる変更を防止できます。

SecureDynamicLibraryLoading

安全な DLL ロードモードを選択します。

構文

SecureDynamicLibraryLoading=<値>

引数

<値>	バイナリー・インジケーター。
enable | yes | on | 1	安全な DLL ロードモードを有効にします。
disable | no | off | 0	安全な DLL ロードモードを無効にします。これは、デフォルトの値です。

説明

HKEY_LOCAL_MACHINE\Software\Intel\MPI レジストリー・キーを使用して、SecureDynamicLibraryLoading レジストリー・エントリーを定義します。安全な DLL ロードモードを有効にするには、このエントリーを設定します。

I_MPI_DAT_LIBRARY

拡張セキュリティー・モードの DLL で使用する DAT ライブラリーを選択します。

構文

I_MPI_DAT_LIBRARY=<ライブラリー>

引数

<ライブラリー>	ロードするライブラリーの名前を指定します。

説明

安全な DLL ロードモードでは、ライブラリーは、DLL を検索するデフォルトのディレクトリーのセットを変更します。これにより、現在のワーキング・ディレクトリーと PATH 環境変数に設定されているディレクトリーを無視できます。ロードする外部 DAT ライブラリーを選択するため、HKEY_LOCAL_MACHINE\Software\Intel\MPI レジストリー・キーの I_MPI_DAT_LIBRARY エントリーを定義します。DAT ライブラリーへのフルパスを指定します。

注意

I_MPI_DAT_LIBRARY 環境変数を指定しただけでは、ダイナミック・ライブラリーはセキュアなモードでロードされません。詳細は、「I_MPI_DAT_LIBRARY」をご覧ください。

SecurePath

外部 DLL が配置されるディレクトリーを指定します。

構文

`SecurePath=<パス>[;<パス>[...]]`

引数

<パス>	ディレクトリーへのパスを指定します。

説明

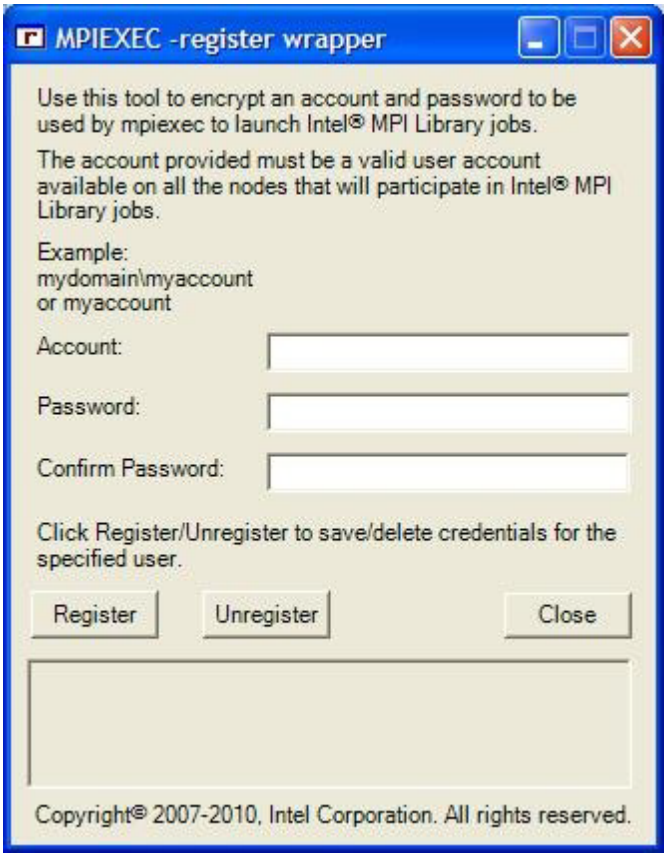
`HKEY_LOCAL_MACHINE\Software\Intel\MPI` レジストリー・キーで `SecurePath` エントリーを定義できます。安全な DLL 読み込みモードで外部 DLL の検索ディレクトリーを指定するには、このエントリーを設定します。安全でないライブラリーの読み込みを避けるため、書き込み可能なディレクトリーを公開する代わりに、安全なディレクトリーを設定してください。

注意

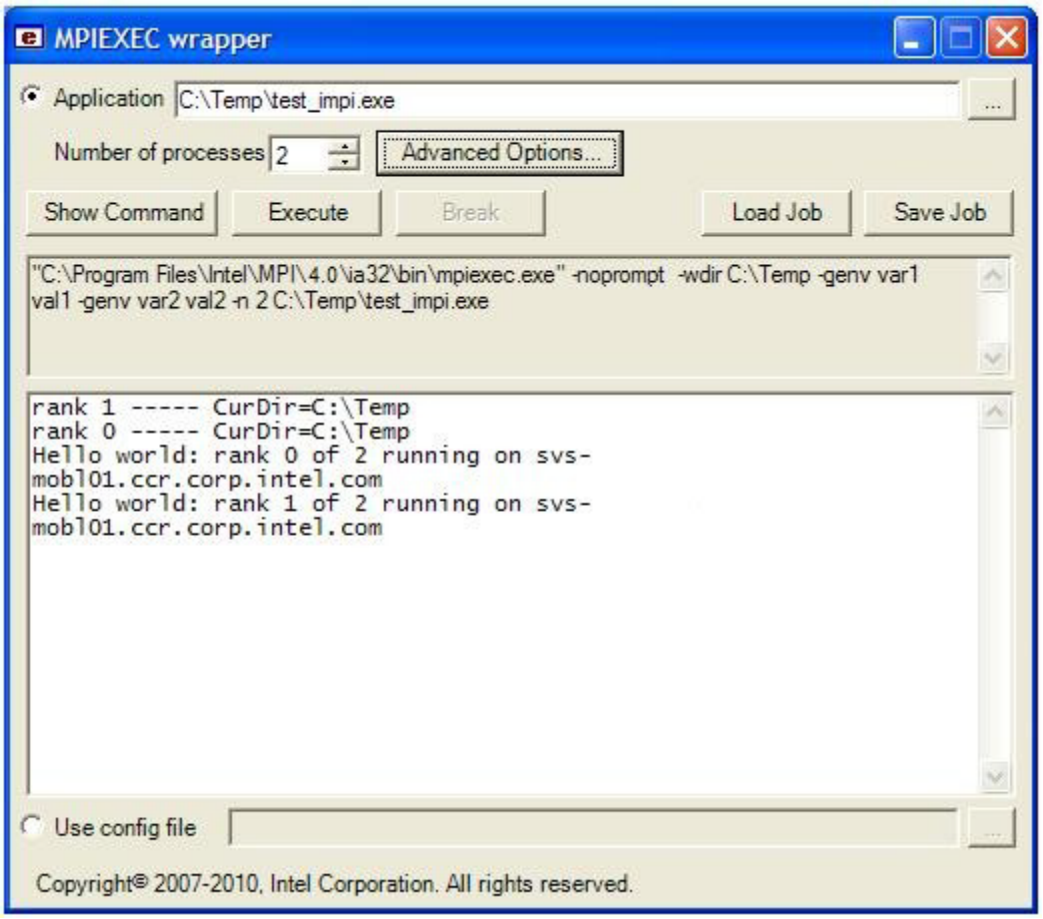
安全な DLL 読み込みモードで DLL をロードできない場合、このオプションを使用します。安全な DLL 読み込みモードが `off` になっている場合、このオプションは効果がありません。

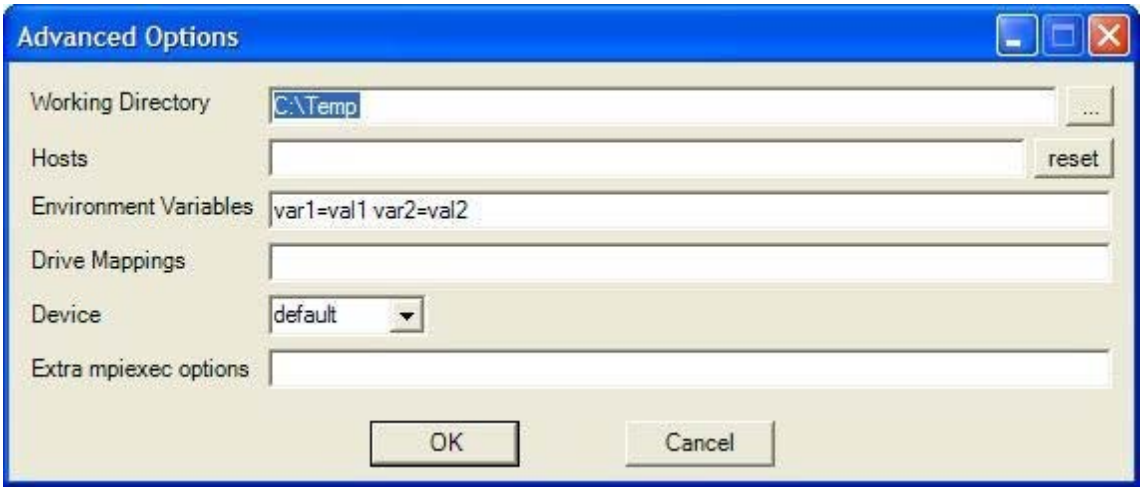
5. グラフィカル・ユーティリティー

インテル® MPI ライブラリーは、3 つのグラフィカル・ユーティリティーを提供します: `wmpiregister`、`wmpiexec` および `wmpiconfig`。これらのユーティリティーは、Windows* 環境でインテル® MPI ライブラリーの利用を簡素化します。



アカウント名とパスワードを暗号化して保存するため、`wmpiregister` ユーティリティーを使用します。指定されたアカウントとパスワードは、MPI ジョブの開始時に利用されます。このユーティリティーを始めて使用する場合、最初に `mpiexec` を起動する際にアカウント名とパスワードを入力する必要があります。



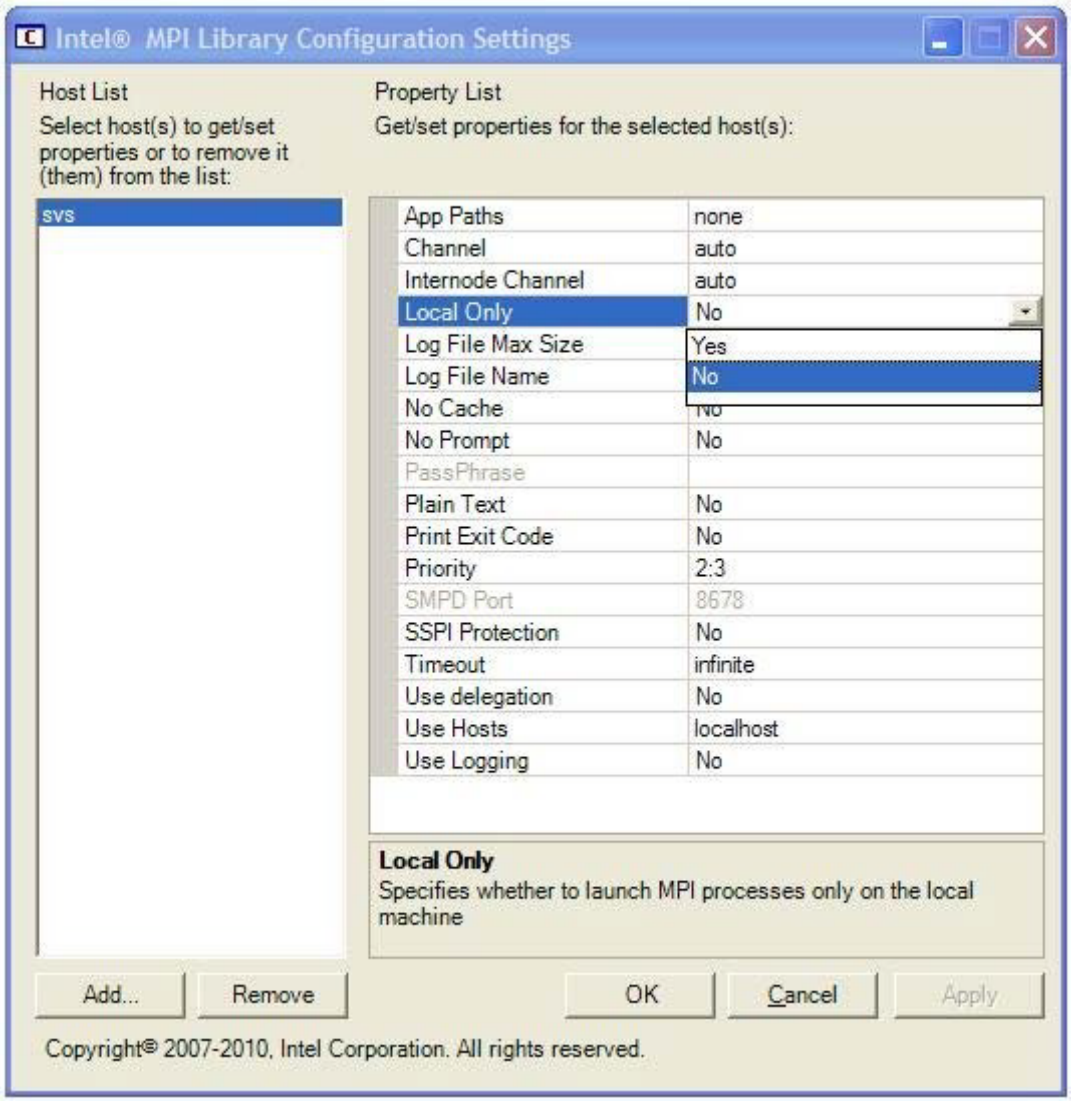


`mpiexec` コマンドのグラフィカル・インターフェイス版として、`wmpiexec` ユーティリティーを使用します。このユーティリティーは、以下を可能にします。

1. 以下を指定することでジョブを表現します。
    - 実行するアプリケーション
    - インスタンスの数
    - ホスト名
    - 使用される通信デバイス
    - MPI プロセス向けのワーキング・ディレクトリー
    - MPI プロセス向けに設定される環境変数

- 使用されるドライブ割り当て
- `wmpiexec` へのその他の MPI オプション

2. **[Save Job]** ボタンを使用してジョブの説明を保存します (オプション)。
3. **[Load Job]** ボタンを使用してジョブの説明を読み込みます (オプション)。
4. **[Show Command]** ボタンを使用して実際の `mpiexec` のコマンドラインを表示します。
5. **[Execute]** ボタンを使用してジョブを起動します。
6. **[Break]** ボタンを使用してジョブの実行をブレークします。



異なるホスト向けにインテル® MPI ライブラリーの設定を表示/変更するには、`wmpiconfig` ユーティリティーを使用します。これは、ホスト上で実行されるすべてのジョブに影響します。`wmpiconfig` ユーティリティーを使用する作業は、次の 3 つのステップに分けることができます

1. インテル® MPI ライブラリーの設定を変更するホストを選択し、**[Add]** をクリックしてホストリストに追加します。
2. ホストリストからホストを選択して、プロパティーを表示します。複数のホスト名を選択した場合、プロパティーは交互に表示されます。
3. 選択したホストのプロパティーを変更し、**[Apply]** ボタンを押して確定します。

6. 用語集

セル	ピニング・プロパティーで記述されるピニングの解像度。
ハイパースレッディング・テクノロジー	各プロセッサー・コアが複数の論理プロセッサーの機能を提供する、IA-64 とインテル® 64 プロセッサー・ファミリーに備わる機能。
論理プロセッサー	ソフトウェア実行 (OS) が、タスクをディスパッチまたはスレッド・コンテキストを実行することができる、プロセッサー・ハードウェア・リソースの基本モジュール。各論理プロセッサーは、同時に 1 つのスレッド・コンテキストを実行します。
マルチコア・プロセッサー	2 つ以上のプロセッサー・コアを集積した物理プロセッサー。
マルチプロセッサー・プラットフォーム	複数の物理パッケージを備えるコンピューター・システム。
プロセッサー・コア	命令をデコード、実行し、そして物理パッケージ内のサブシステム間でデータを転送するための専用の機能を備える回路。プロセッサー・コアは、1 つもしくは 2 つの論理プロセッサーを含みます。
物理パッケージ	マイクロプロセッサーの物理パッケージは、1 つ以上のソフトウェア・スレッドを同時に実行することができます。各物理パッケージは、物理ソケットに装着されます。各物理パッケージは、1 つ以上のプロセッサー・コアを搭載しています。
プロセッサー・トポロジー	1 つ以上のハードウェア・マルチスレッディングが可能なパッケージを使用する計算プラットフォーム内の「共有 vs. 専用」ハードウェア・リソースの階層関係。

7. 索引

/

/Zi 9

/ZI /debug 9

{

-{cc cxx fc}=<コンパイラー> 9

A

Active Directry* 49

B

-binding 32

-bootstrap 35

-bootstrap-exec 35

-branch-count 30

C

-configfile <ファイル名> 14, 30

cpuinfo 45

D

-dapl 37

-DAPL 37

-delegate 15, 31

-dir <ディレクトリー> 17

E

-echo 9

-env <環境変数> <値> 16, 36

-envall 36

-envexcl <環境変数名のリスト> 17

-envlist <環境変数名のリスト> 16, 36

-envnone 16, 36

-envuser 17

-exitcodes 15

F

-f <ホストファイル> 28

G

-g<l-オプション> 14

-genv <環境変数> <値> 29

-genvall 29

-genvlist 29

-genvnone 29

-grr 29

H

-h 16

-help 16

--help 16

-host <ノード名> 17, 36

-hostfile <ホストファイル> 28

-hostos <ホスト OS> 37

-hosts 15

-hosts <ノードリスト> 30

Hydra 27

hydra_service 27

I

I_MPI_{CC,CXX,FC,F77,F90} 11

I_MPI_{CC,CXX,FC,F77,F90}_PROFILE 10

I_MPI_ADJUST_<opname> 101

I_MPI_ADJUST_ALLGATHER_KN_RADIX 107

I_MPI_ADJUST_ALLREDUCE_KN_RADIX 108

I_MPI_ADJUST_BCAST_KN_RADIX 108

I_MPI_ADJUST_BCAST_SEGMENT 107

I_MPI_ADJUST_GATHERV_KN_RADIX 108

I_MPI_ADJUST_IALLREDUCE_KN_RADIX 109

I_MPI_ADJUST_IBCAST_KN_RADIX 109

I_MPI_ADJUST_IGATHER_KN_RADIX 110

I_MPI_ADJUST_IREDUCE_KN_RADIX 109

I_MPI_ADJUST_ISCATTER_KN_RADIX 110

I_MPI_ADJUST_REDUCE_KN_RADIX 108

I_MPI_ADJUST_REDUCE_SEGMENT 106

I_MPI_ALLGATHER_MSG....112
I_MPI_ALLREDUCE_MSG....113
I_MPI_ALLTOALL_MSG....112
I_MPI_ALLTOALL_NUM_PROCS....112
I_MPI_AUTH_METHOD....50
I_MPI_BCAST_MSG....111
I_MPI_BCAST_NUM_PROCS....111
I_MPI_CACHE_BYPASS....84
I_MPI_CACHE_BYPASS_THRESHOLDS....84
I_MPI_CHECK_DAPL_PROVIDER_COMPATIBILITY....99
I_MPI_COMPATIBILITY....115
I_MPI_COMPILER_CONFIG_DIR....11
I_MPI_CONN_EVD_QLEN....97
I_MPI_DAPL_BUFFER_NUM....94
I_MPI_DAPL_BUFFER_SIZE....94
I_MPI_DAPL_CHECK_MAX_RDMA_SIZE....96
I_MPI_DAPL_CONN_EVD_SIZE....97
I_MPI_DAPL_DESIRED_STATIC_CONNECTIONS_NUM 98
I_MPI_DAPL_DIRECT_COPY_THRESHOLD....92
I_MPI_DAPL_DYNAMIC_CONNECTION_MODE....93
I_MPI_DAPL_EAGER_MESSAGE_AGGREGATION....92
I_MPI_DAPL_MAX_MSG_SIZE....96
I_MPI_DAPL_PROVIDER....90
I_MPI_DAPL_RDMA_RNDV_WRITE....95
I_MPI_DAPL_RDMA_WRITE_IMM....98
I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT....95
I_MPI_DAPL_SCALABLE_PROGRESS....93
I_MPI_DAPL_SR_BUF_NUM....98
I_MPI_DAPL_SR_THRESHOLD....97
I_MPI_DAPL_TRANSLATION_CACHE....91
I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE....91
I_MPI_DAT_LIBRARY....22, 91, 133
I_MPI_DEBUG....17
I_MPI_DEBUG_OUTPUT....19
I_MPI_DEVICE....78
I_MPI_DYNAMIC_CONNECTION_MODE....93
I_MPI_DYNAMIC_CONNECTIONS_MODE....93
I_MPI_EAGER_THRESHOLD....81
I_MPI_FABRIC....78
I_MPI_FABRICS_LIST....80
I_MPI_FALLBACK....80
I_MPI_FALLBACK_DEVICE....80
I_MPI_FAST_COLLECTIVES....110
I_MPI_GATHER_MSG....114
I_MPI_HYDRA_BOOTSTRAP....41
I_MPI_HYDRA_BOOTSTRAP_EXEC....41
I_MPI_HYDRA_BRANCH_COUNT....42
I_MPI_HYDRA_DEBUG....38
I_MPI_HYDRA_ENV....38
I_MPI_HYDRA_HOST_FILE....38
I_MPI_HYDRA_IFACE....43
I_MPI_HYDRA_PMI_AGGREGATE....43
I_MPI_HYDRA_PMI_CONNECT....41
I_MPI_INTRANODE_EAGER_THRESHOLD....82
I_MPI_INTRANODE_SHMEM_BYPASS....89
I_MPI_JOB_ABORT_SIGNAL....40
I_MPI_JOB_RESPECT_PROCESS_PLACEMENT....43
I_MPI_JOB_SIGNAL_PROPAGATION....40
I_MPI_JOB_TIMEOUT....21, 39
I_MPI_JOB_TIMEOUT_SIGNAL....39
I_MPI_LINK....12
I_MPI_MPIEXEC_TIMEOUT....39
I_MPI_NETMASK....20, 99
I_MPI_PERHOST....42
I_MPI_PIN....59
I_MPI_PIN_CELL....65
I_MPI_PIN_DOMAIN....66
I_MPI_PIN_ORDER....75
I_MPI_PIN_PROCESSOR_LIST....60
I_MPI_PIN_PROCS....60
I_MPI_PLATFORM....23
I_MPI_PLATFORM_CHECK....24
I_MPI_PRINT_VERSION....19
I_MPI_RDMA_BUFFER_NUM....94
I_MPI_RDMA_BUFFER_SIZE....94
I_MPI_RDMA_CHECK_MAX_RDMA_SIZE....96

I_MPI_RDMA_CONN_EVD_SIZE ... 97
I_MPI_RDMA_EAGER_THRESHOLD ... 92
I_MPI_RDMA_MAX_MSG_SIZE ... 96
I_MPI_RDMA_RNDV_BUF_ALIGN ... 95
I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT ... 95
I_MPI_RDMA_RNDV_WRITE ... 95
I_MPI_RDMA_SCALABLE_PROGRESS ... 93
I_MPI_RDMA_TRANSLATION_CACHE ... 91
I_MPI_RDMA_VBUF_TOTAL_SIZE ... 94
I_MPI_RDMA_WRITE_IMM ... 98
I_MPI_REDSCAT_MSG ... 113
I_MPI_ROOT ... 11
I_MPI_SCALABLE_OPTIMIZATION ... 82
I_MPI_SCATTER_MSG ... 114
I_MPI_SHM_BUFFER_SIZE ... 87
I_MPI_SHM_BYPASS ... 89
I_MPI_SHM_CACHE_BYPASS ... 84
I_MPI_SHM_CACHE_BYPASS_THRESHOLDS ... 84
I_MPI_SHM_CELL_NUM ... 86
I_MPI_SHM_CELL_SIZE ... 86
I_MPI_SHM_FBOX ... 85
I_MPI_SHM_FBOX_SIZE ... 86
I_MPI_SHM_LMT ... 87
I_MPI_SHM_LMT_BUFFER_NUM ... 87
I_MPI_SHM_LMT_BUFFER_SIZE ... 87
I_MPI_SHM_NUM_BUFFERS ... 87
I_MPI_SHM_SPIN_COUNT ... 90
I_MPI_SMPD_VERSION_CHECK ... 21
I_MPI_SPIN_COUNT ... 82
I_MPI_SSHM ... 88
I_MPI_SSHM_BUFFER_NUM ... 88
I_MPI_SSHM_BUFFER_SIZE ... 88
I_MPI_SSHM_DYNAMIC_CONNECTION ... 89
I_MPI_STATS ... 116, 123, 131
I_MPI_STATS_ACCURACY ... 128
I_MPI_STATS_BUCKETS ... 119
I_MPI_STATS_FILE ... 120, 123
I_MPI_STATS_SCOPE ... 117, 124
I_MPI_TCP_BUFFER_SIZE ... 100
I_MPI_TCP_NETMASK ... 99
I_MPI_THREAD_LEVEL_DEFAULT ... 24
I_MPI_TMPDIR ... 43
I_MPI_TUNER_DATA_DIR ... 22
I_MPI_USE_DAPL_INTRANODE ... 89
I_MPI_USE_DYNAMIC_CONNECTIONS ... 83
I_MPI_USE_RENDEZVOUS_RDMA_WRITE ... 95
I_MPI_WAIT_MODE ... 83
-ib ... 37
-IB ... 38
-iface <インターフェイス> ... 30
-ilp64 ... 8
-impersonate ... 15, 32
-info ... 31
IPM ... 123

L

-l ... 15
LMT ... 87
-localhost ... 32
-localonly ... 16, 32
-localroot ... 16, 32
-logon ... 15

M

-machine <マシンファイル> ... 28
-machinefile <マシンファイル> ... 13, 28
-map <ドライブ:\\ホスト名\共有名> ... 17, 32
-mapall ... 17, 32
MAX_INT ... 96
max_rdma_size ... 97
MPI_Allgather ... 105
MPI_Allgatherv ... 105
MPI_Allreduce ... 105
MPI_Alltoall ... 105
MPI_Alltoallv ... 105
MPI_Alltoallw ... 105
MPI_Barrier ... 105

MPI_Bcast 106
MPI_COMM_WORLD 123
MPI_Exscan 106
MPI_Gather 106
MPI_Gatherv 106
MPI_Reduce 106
MPI_Reduce_scatter 106
MPI_Scan 106
MPI_Scatter 106
MPI_Scatterv 106
MPICH_{CC,CXX,FC,F77,F90} 11
mpiexec 28
mpiexec.smpd 13
MPIEXEC_SIGNAL_PROPAGATION 40
MPIEXEC_TIMEOUT 21, 39
MPIEXEC_TIMEOUT_SIGNAL 39
mpitune 52

N

-n <プロセス数> 16, 36
-nopopup_debug 15
NUM_RDMA_BUFFER 94

O

-O 9
-ordered-output 31

P

-p <ポート> 15
-path <ディレクトリー> 17, 31, 36
-perhost 29
-pmi-aggregate 30
-pmi-connect 29
-port <ポート> 15
-ppn 29
-print-all-exitcodes 36
-print-rank-map 36
-profile=<プロファイル名> 8
-pwdfile <ファイル名> 15

R

-rdma 37
-RDMA 37
RDMA_IBA_EAGER_THRESHOLD 92
-register 32
-register [-user n] 16
-remove 32
-remove [-user n] 16
-rr 29

S

-s <spec> 31
SecureDynamicLibraryLoading 133
SecurePath 133
-show 9
-show_env 9
smpd 25
SMPD 25
SPN 50
SSPI 49

T

-t 8
-timeout <秒> 16
-tmpdir 31
TMPDIR 43
-trace-collectives 29
-trace-pt2pt 29
-tune 15, 30

V

-v 10, 36
-V 31
-validate 16, 32
-verbose 15, 36
-version 31
VT_ROOT 11

W

-wdir <ディレクトリー> 17, 36

-whoami 16, 32