インテル® MPI ライブラリー for Linux*

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 11
2.1.3. プロファイル 11
2.1.4. 環境変数 12
2.2. 簡素化されたジョブ起動コマンド 15
2.3. スケーラブルなプロセス管理システム (Hydra) コマンド 19
2.3.1. グローバルオプション 20
2.3.2. ローカルオプション 35
2.3.3. 拡張デバイス制御オプション 36
2.3.4. 環境変数 38
2.3.5. Cleaning up ユーティリティー 50
2.3.6. チェックポイント・リスタートのサポート 51
2.4. 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 58
2.5. インテル® Xeon Phi™ コプロセッサーのサポート 58
2.5.1. 使用モデル 58
2.5.2. 環境変数 60
2.5.3. コンパイラーのコマンド 64
2.6. 多目的デーモン (MPD) のコマンド 65
2.6.1. ジョブ開始コマンド 72
2.6.2. 設定ファイル 91
2.6.3. 環境変数 92
2.7. プロセッサー情報ユーティリティー 95
3. チューニング・リファレンス 99
3.1. mpitune ユーティリティーを使用 99
3.1.1. クラスター固有のチューニング 103

3.1.2. アプリケーション固有のチューニング 104
3.1.3. チューニング・ユーティリティーの出力 107
3.2. プロセスのピニング (固定) 108
3.2.1. プロセスピニングのデフォルト設定 108
3.2.2. プロセッサーの識別 108
3.2.3. 環境変数 109
3.2.4. OpenMP* API との相互利用 118
3.3. ファブリック制御 131
3.3.1. 通信ファブリック制御 131
3.3.2. 共有メモリー制御 138
3.3.3. DAPL ネットワーク・ファブリック制御 145
3.3.4. DAPL UD ネットワーク・ファブリック制御 154
3.3.5. TCP ネットワーク・ファブリック制御 163
3.3.6. TMI ネットワーク・ファブリック制御 165
3.3.7. OFA ネットワーク・ファブリック制御 167
3.3.8. OFI* ネットワーク・ファブリック制御 172
3.4. 集団操作制御 173
3.4.1. I_MPI_ADJUST ファミリー 174
3.4.2. I_MPI_MSG ファミリー 183
3.5. その他 187
3.5.1. タイマー制御 187
3.5.2. 互換性制御 188
3.5.3. ダイナミック・プロセスのサポート 188
3.5.4. フォールトトレラント 189
3.5.5. 統計収集モード 190
3.5.6. ILP64 サポート 205
3.5.7. ユニファイド・メモリー管理 206
3.5.8. ファイルシステムのサポート 207
3.5.9. マルチスレッド化された memcpy のサポート 208
4. 用語集 210
5. 索引 211

著作権と商標について

本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスも許諾するものではありません。

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

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

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

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

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

Intel、インテル、Intel ロゴ、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。

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

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

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

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

最適化に関する注意事項

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

注意事項の改訂 #20110804

1. 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 – 概要	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 MPD デーモンのコマンドについて
第 3 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 4 章 – 用語集	このドキュメントで使用される基本用語
第 5 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)、多目的デーモン (MPD)、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、共有 (.so) ライブラリー、ドキュメントなどが含まれています。
- インテル® MPI ライブラリー開発キット (SDK) には、すべてのランタイム環境コンポーネントに加え、`mpiicc` などのコンパイラー・コマンド、インクルード・ファイルとモジュール、スタティック (.a) ライブラリー、デバッグ・ライブラリー、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 Linux* 向けのアップデートを反映しています。このドキュメントには、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。
- 拡張デバイス制御オプションに新しいオプション `-psm2` と `-PSM2` を追加。

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
- 簡素化されたジョブ起動コマンド
- スケーラブルなプロセス管理システム (Hydra) のコマンド
- 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® Xeon Phi™ コプロセッサーのサポート
- 多目的デーモン (MPD) のコマンド
- 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	デフォルト・コンパイラー	サポートされる言語	サポートされる ABI
汎用コンパイラー			
`mpicc`	`gcc, cc`	C	64 ビット
`mpicxx`	`g++`	C/C++	64 ビット
`mpifc`	`gfortran`	Fortran77/Fortran 95	64 ビット
GNU* コンパイラー・4.3 以降			
`mpigcc`	`gcc`	C	64 ビット
`mpigxx`	`g++`	C/C++	64 ビット
`mpif77`	`g77`	Fortran 77	64 ビット
`mpif90`	`gfortran`	Fortran 95	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`	`icc`	C	64 ビット
`mpiicpc`	`icpc`	C++	64 ビット
`mpiifort`	`ifort`	Fortran77/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、<installdir>/<arch>/bin ディレクトリーに配置されます。ここで、<installdir> はインテル® MPI ライブラリーのインストール・ディレクトリーで、<arch> は次のいずれかのアーキテクチャーです。
  - intel64 - インテル® 64 アーキテクチャー
  - mic - インテル® Xeon Phi™ コプロセッサーのアーキテクチャー
- 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

-static_mpi

インテル® MPI ライブラリーをスタティックにリンクするには、このオプションを使用します。このオプションは、ほかのライブラリーのデフォルトのリンク方法に影響しません。

-static

インテル® MPI ライブラリーをスタティックにリンクするには、このオプションを使用します。このオプションはコンパイラーへ渡されます。

-nostrip

このオプションは、インテル® MPI ライブラリーをスタティックにリンクする際に、デバッグ情報のストリップを off にするために使用します。

-config=<name>

このオプションで、設定ファイルを指定します。詳細は、「設定ファイル」をご覧ください。

-profile=<profile_name>

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。

- <installdir>/<arch>/etc にある設定ファイル <profile_name>.conf 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配置されるライブラリー lib<profile_name>.so や lib<profile_name>.a 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。これは、mpiicc やほかのコンパイラー・スクリプトに -profile=vt オプションを指定するのと同じ効果があります。

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。ほかのプロファイル・ライブラリーを指定するには、I_MPI_TRACE_PROFILE 環境変数に <profile_name> を設定します。例えば、I_MPI_TRACE_PROFILE に vtfs を設定すると、フェイルセーフ版のインテル® Trace Collector とリンクを行います。

-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。これは、mpiicc やほかのコンパイラー・スクリプトに -profile=vtmc オプションを指定するのと同じ効果があります。

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。ほかのプロファイル・ライブラリーを指定するには、I_MPI_CHECK_PROFILE 環境変数に <profile_name> を設定します。

-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 コンパイラーの個別コンパイルに -i8 オプションを指定した場合、リンク時に -i8 または -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 サポート」をご覧ください。

-dynamic_log

このオプションは、インテル® Trace Collector ライブラリーとダイナミックにリンクする際に、-t オプションと併用して使用します。このオプションは、ほかのライブラリーのデフォルトのリンク方法に影響しません。

リンクしたプログラムを実行するには、環境変数 LD_LIBRARY_PATH に $VT_ROOT/slib を含めます。

-g

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。-g のデバッグビルドで追加されるデバッグ情報の使い方については、環境変数「I_MPI_DEBUG」をご覧ください。

注意

最適化されたライブラリーは、デフォルトで -g オプションとリンクされます。

注意

実行時に特定の libmpi.so 設定をロードするには、mpivars,{sh | csh} [debug | debug_mt] コマンドを使用します。

-link_mpi=<arg>

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。

注意

実行時に特定の `libmpi.so` 設定をロードするには、`mpivars,{sh | csh} [debug | debug_mt]` コマンドを使用します。

-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。

-fast

このオプションは、プログラム全体の速度を最大限にします。このオプションは、スタティック・リンクを強制します。

インテル製プロセッサー以外でこのオプションを使用するには、「xHost」をご覧ください。

注意

このオプションは、`mpiicc`、`mpiicpc` および `mpiifort` のインテル® コンパイラー向けのドライバーでサポートされます。

-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 -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 -show -o a.out test.o
```

このオプションは、サポートされる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

サポートされるコンパイラーが起動されるときに影響す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,f77,f90}=<compiler>

サポートされる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。

```
$ mpicc -cc=icc -c test.c
```

icc へのパスが設定されていることを確認してください。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。

-gcc-version=<nnn>

特定の GNU* C++ 環境で実行できるように、mpicxx コンパイラー・ドライバーおよび mpiicpc コンパイラー・ドライバーにこのオプションを使用して、アプリケーションをリンクしてください。<nnn> に設定可能な値は以下のとおりです。

<nnn> 値	GNU* C++ のバージョン
430	4.3.x
440	4.4.x
450	4.5.x
460	4.6.x
470	4.7.x

認識された GNU* C++ コンパイラー・バージョンと互換性のあるライブラリーがデフォルトで使用されます。GNU* C++ のバージョンが 4.0.04.1.0 よりも古い場合は、このオプションを使用しないでください。

-compchk

コンパイラーの設定チェックを有効にする場合、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、各コンパイラーのドライバーは、基礎となるコンパイラーが適切に設定されているかチェックを行います。

-v

コンパイラー・ドライバーのバージョンとネイティブ・コンパイラ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。

<installdir>/<arch>/etc/mpi<compiler>-<name>.conf

ここで以下を指定します。

<arch>= {intel64[em64t],mic} は、インテル® 64 アーキテクチャーとインテル® Xeon Phi™ コプロセッサー・アーキテクチャーに相当します。

<compiler>={cc,cxx,f77,f90} は、コンパイルする言語に依存します。

<name> = ハイフン (-) でスペースを置き換えるベースとなるコンパイラーの名前を指定します。例えば、cc -64 の <name> 値は、cc--64。

コンパイラー・コマンドに影響する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が、コンパイルやリンクに –config オプションを使用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。プロファイル・ファイルは、<installdir>/<arch>/etc ディレクトリーに配置されます。インテル® MPI ライブラリーは、インテル® Trace Collector 向けの事前定義プロファイルを用意しています。

<installdir>/etc/vt.conf - 通常のインテル® Trace Collector ライブラリー
<installdir>/etc/vtfs.conf - フェイスセーフ版のインテル® Trace Collector ライブラリー
<installdir>/etc/vtmc.conf - 正当性をチェックするインテル® Trace Collector ライブラリー。

また、<profile_name>.conf という名称で独自のプロファイルを作成できます。

次の環境変数を定義できます。

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、次の内容で /myprof.conf ファイルを作成します。

```
PROFILE_PRELIB="-L<path_to_myprof>/lib -lmyprof"
PROFILE_INCPATHS="-I<paths_to_myprof>/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 (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=<profile_name>

廃止された構文

MPI{CC,CXX,FC,F77,F90}_PROFILE=<profile_name>

引数

<profile_name>	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 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=<profile_name> を指定するのと同じです。

I_MPI_TRACE_PROFILE

-trace オプションのデフォルト・プロファイルを指定します。

構文

I_MPI_TRACE_PROFILE=<profile_name>

引数

<profile_name>	トレース・プロファイル名を指定します。デフォルト値は vt です。

説明

mpicc やほかのインテル® MPI コンパイラー・ドライバーに -trace オプションを指定した時に使用する特定の 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。

I_MPI_{CC,CXX,F77,F90}_PROFILE 環境変数は I_MPI_TRACE_PROFILE をオーバーライドします。

I_MPI_CHECK_PROFILE

-check_mpi オプションのデフォルト・プロファイルを指定します。

構文

I_MPI_CHECK_PROFILE=<profile_name>

引数

<profile_name>	チェックするプロファイル名を指定します。デフォルト値は vtmc です。

説明

mpicc やほかのインテル® MPI コンパイラー・ドライバーに -check_mpi オプションを指定した時に使用する特定の MPI チェック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。

I_MPI_{CC,CXX,F77,F90}_PROFILE 環境変数は I_MPI_CHECK_PROFILE をオーバーライドします。

I_MPI_CHECK_COMPILER

コンパイラーの互換性チェックを on/off します。

構文

I_MPI_CHECK_COMPILER=<arg>

引数

<arg>	バイナリー・インジケーター。
enable | yes | on | 1	コンパイラーのチェックを有効にします。
disable | no | off | 0	コンパイラーのチェックを無効にします。これは、デフォルト値です。

説明

I_MPI_CHECK_COMPILER が enable に設定されている場合、インテル® MPI のコンパイラー・ドライバーはコンパイラーの互換性をチェックします。通常のコンパイルでは、ベースとなるコンパイラーの既知のバージョンを使用する必要があります。

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}=<compiler>

廃止された構文

MPICH_{CC,CXX,FC,F77,F90}=<compiler>

引数

<compiler>	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。

注意

指定するコンパイラーが存在する場合、設定ファイルが source されます。詳細は、「設定ファイル」をご覧ください。

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=<path>
```

引数

<path>	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=<path>
```

引数

<path>	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=<path>
```

引数

<path>	コンパイラーの設定ファイルの場所を指定します。デフォルト値は、<installdir>/<arch>/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=<arg>
```

引数

<arg>	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。

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。

I_MPI_DEBUG_INFO_STRIP

アプリケーションをスタティック・リンクする際にデバッグ情報のストリップを on/off します。

構文

```
I_MPI_DEBUG_INFO_STRIP=<arg>
```

引数

<arg>	バイナリー・インジケーター。
enable | yes | on | 1	有効にします。これは、デフォルト値です。
disable | no | off | 0	無効にします。

説明

このオプションは、インテル® MPI ライブラリーをスタティックにリンクする際に、デバッグ情報のストリップを on/off します。デフォルトでデバッグ情報はストリップされます。

2.2. 簡素化されたジョブ起動コマンド

mpirun

構文

```
mpirun <options>
```

ここで、<options>:= <mpiexec.hydra options> |[<mpdboot options>] <mpiexec options> です。

引数

<mpiexec.hydra options>	mpiexec.hydra の節で説明した mpiexec.hydra オプション。これはデフォルトの操作モードです。
<mpdboot options>	mpdboot コマンドの説明で記載される mpdboot オプション、-n オプションを除く。
<mpiexec options>	mpiexec の節で説明する mpiexec のオプション。

説明

次のコマンドを使用して MPI ジョブを送信します。mpirun コマンドは、基盤となるプロセス管理として Hydra や MPD を使用しています。Hydra は、デフォルトのプロセス管理です。デフォルトを変更するには、I_MPI_PROCESS_MANAGER 環境変数を設定します。

mpirun コマンドは、Torque*、PBS Pro*、LSF*、Parallelnavi* NQS*、SLURM*、Univa* Grid Engine*、LoadLeveler* などのジョブ・スケジューラーを使用して割り当てられたセッションから送信された MPI ジョブを検出します。mpirun コマンドは、それぞれの環境からホストリストを抽出し、上記の方式に従ってこれらのノードを使用します。

この場合、mpd.hosts ファイルを作成する必要はありません。システムにインストールされているジョブ・スケジューラーでセッションを配置し、MPI ジョブを実行するため内部で mpirun コマンドを使用します。

例

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

このコマンドは、デフォルトで Hydra プロセス管理を使用するため mpiexec.hydra コマンドを起動します。

Hydra の仕様

アクティブなプロセス管理として Hydra を選択した場合、mpirun コマンドは互換性を保持するため暗黙的に MPD 固有のオプションを無視します。次の表は、暗黙的に無視され、サポートされない MPD オプションのリストです。Hydra プロセス管理を使用する場合、これらサポートされないオプションの使用を避けてください。

無視される mpdboot オプション	無視される mpiexec オプション	サポートされない mpdboot オプション	サポートされない mpiexec オプション
--loccons	-[g]envuser	--user=<user> | -u <user>	-a
--remcons	-[g]envexcl	--mpd=<mpdcmd> | -m <mpdcmd>	
--ordered | -o	-m	--shell | -s	
--maxbranch=<maxbranch> | -b <maxbranch>	-ifhn <interface/hostname>	-1	
--parallel-startup | -p	-ecfn <filename> -tvsu	--ncpus=<ncpus>	

MPD の仕様

プロセス管理に MPD を選択した場合、mpirun コマンドは自動的に mpd デーモンの独立したリングを開始し、MPI ジョブを起動、そしてジョブの終了時に mpd リングをシャットダウンします。

最初の非 mpdboot オプション (-n や -np を含む) は、mpdboot と mpiexec オプションを区分します。この時点までの区切りオプションを除くすべてのオプションは、mpdboot コマンドに渡されます。これ以降は、区切りオプションを含むすべてのオプションが mpiexec コマンドに渡されます。

mpdboot と mpiexec コマンドに適用されるすべての設定ファイルと環境変数も、mpirun コマンドに適用されます。

次の規則で定義されるホストのセットは、この順番で実行されます。

1. mpdboot ホストファイルからのすべてのホスト名 (mpd.hosts もしくは –f オプションで指定されるどちらか一方)。
2. mpd リングが実行されている場合、mpdtrace コマンドから返されるすべてのホスト名。
3. ローカルホスト (この場合警告が発せられます)。

I_MPI_MPIRUN_CLEANUP

mpirun コマンド実行後の環境のクリーンアップを制御します。

構文

I_MPI_MPIRUN_CLEANUP=<value>

引数

<value>	オプションを定義します。
enable | yes | on | 1	環境のクリーンアップを有効にします。
disable | no | off | 0	環境のクリーンアップを無効にします。これは、デフォルト値です。

説明

mpirun の完了後に環境をクリーアップする場合、この環境変数を使用します。クリーンアップには、終了しないサービスプロセスや一時ファイルの削除などが含まれます。

I_MPI_PROCESS_MANAGER

mpirun コマンドで使用されるプロセス管理を選択します。

構文

I_MPI_PROCESS_MANAGER=<value>

引数

<value>	文字列。
hydra	Hydra プロセス管理を使用します。これは、デフォルト値です。
mpd	MPD プロセス管理を使用します。

説明

mpirun コマンドで使用されるプロセス管理を選択するため、この環境変数を設定します。

注意

mpiexec コマンド (MPD 向け) と mpiexec.hydra コマンド (Hydra 向け) を起動して、直接プロセス管理を実行することもできます。

I_MPI_YARN

YARN* で管理されたクラスター上で実行するには、この変数を設定します。

引数

<value>	バイナリー・インジケーター。
enable | yes | on | 1	YARN* サポートを有効にします。
disable | no | off | 0	YARN* サポートを無効にします。これは、デフォルト値です。

説明

MPI ジョブを実行する前に、YARN* クラスター管理から Hydra 要求リソースを作成するには、この環境変数を設定します。この機能は、Llama* がインストールされ YARN で管理されるクラスター上で MPI プロセスを起動する場合にのみ利用できます (例えば、Hadoop* 向けの Cloudera* ディストリビューションのクラスター)。

使用例

クラスター上で YARN* が Llama* と適切に動作するように設定され (設定の詳細は Llama* のドキュメントをご覧ください)、Apache* Thrift* がインストールされていることを確認します。

1. YARN* が実行されているホストで Llama* が起動されていることを確認します。起動していない場合、llama ユーザーで次のコマンドを実行して開始します。

   ```
   $ llama [--verbose &]
   ```

2. クラスターでパスワードなしの ssh が設定されていることを確認してください。
3. I_MPI_YARN 環境変数を設定します。

   ```
   $ export I_MPI_YARN=1
   ```

4. Thrift's Python* モジュールを指すように I_MPI_THRIFT_PYTHON_LIB 環境変数を設定するか、PYTHONPATH 環境変数にこれらのモジュールを追加します。
5. Llama* サーバーホスト/ポートを指すように I_MPI_LLAMA_HOST/I_MPI_LLAMA_PORT を設定します (Llama* サービスが実行されているホストから MPI を起動する場合、デフォルトで localhost:15000 に設定されており、この手順はスキップできます)。
6. 通常と同じように MPI を起動します (リソースは自動的に YARN* で割り当てられるため、ホストやマシンファイルを指定する必要はありません)。

   ```
   $ mpirun –n 16 –ppn 2 [other IMPI options] <application>
   ```

注意

この機能は、Hydra プロセス管理でのみ利用できます。

2.3. スケーラブルなプロセス管理システム (Hydra) コマンド

mpiexec.hydra

mpiexec.hydra は、MPD プロセス管理よりスケーラブルな代替手段です。

構文

```
mpiexec.hydra <g-options> <l-options> <executable>
```

または

```
mpiexec.hydra <g-options> <l-options> <executable1> : \
<l-options> <executable2>
```

引数

<g-options>	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-options>	単一の引数セットに適用するローカルオプション。
<executable>	./a.out または path/実行形式ファイル名。

説明

MPD リングで MPI アプリケーションを実行するには、mpiexec.hydra ユーティリティーを使用します。

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 <executable> 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とホストで a.out を実行します。

```
$ mpiexec.hydra -f <hostsfile> -n <# of processes> ./a.out
```

ここで以下を指定します。

- <# of processes> には、a.out を実行するプロセス数を指定します。
- <hostsfile> には、a.out を実行するホストのリストを指定します。

異なる MPI プログラムを異なる引数セットで実行するには、長いコマンドラインを使用します。例えば、次のコマンドは 2 つの異なる実行形式を異なる引数セットで実行します。

```
$ mpiexec.hydra -f <hostsfile> -env <VAR1> <VAL1> -n 2 ./a.out : \
-env <VAR2> <VAL2> -n 2 ./b.out
```

注意

クラスターのすべてのノード上で PATH 環境変数に「.」が設定されていない場合、a.out の代わりに ./a.out を指定してください。

注意

グローバルオプションとローカルオプションを区別する必要があります。コマンドラインで、ローカルオプションはグローバルオプションの後に指定してください。

2.3.1. グローバルオプション

-hostfile <hostfile> または -f <hostfile>

アプリケーションを実行するホスト名を指定します。ホスト名が重複されると 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。

- ラウンドロビン・スケジュールを使用して、各ホスト上で MPI プロセスを連続して配置するには、-perhost、-ppn、および -grr オプションを指定します。
- -rr オプションは、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。

-machinefile <machine file> または -machine <machine file>

このオプションは、<machine file> 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。

```
$ cat ./machinefile
node0:2 binding=map=0,3
node1:2 binding=map=[2,8]
node0:1 binding=map=8
$ mpiexec.hydra -machinefile ./machinefile -n 5 -l numactl --show
[4] policy: default
[4] preferred node: current [4] physcpubind: 8
[4] cpubind: 0
[4] nodebind: 0
[4] membind: 0 1
[0] policy: default
[0] preferred node: current [0] physcpubind: 0
[0] cpubind: 0
[0] nodebind: 0
[0] membind: 0 1
[1] policy: default
[1] preferred node: current [1] physcpubind: 3
[1] cpubind: 1
[1] nodebind: 1
[1] membind: 0 1
[3] policy: default
[3] preferred node: current [3] physcpubind: 3
[3] cpubind: 1
[3] nodebind: 1
[3] membind: 0 1
[2] policy: default
[2] preferred node: current [2] physcpubind: 1
[2] cpubind: 1
[2] nodebind: 1
[2] membind: 0 1
```

-genv <ENVVAR> <value>

すべての MPI プロセス向けに、<ENVVAR> に指定された <value> 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 <list of genv var names>

引数リストと現在の値を渡します。<list of genv var names> 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 <mode>

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 <mode> 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。
- alltoall - 任意の取得要求が完了する前に、情報はすべての pmi_proxy 間で自動的に交換されます。

lazy-cache が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 <# of processes>、-ppn <# of processes>、または -grr <# of processes>

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 [<profiling_library>] または -t [<profiling_library>]

指定された <profiling_library> を使用して MPI アプリケーションのプロファイルを行うには、このオプションを指定します。<profiling_library> が省略された場合、デフォルトのプロファイル・ライブラリーは、libVT.so です。

デフォルトのプロファイル・ライブラリーを変更するには、「I_MPI_JOB_TRACE_LIBS」環境変数を設定します。

(SDK のみ) -mps

内部的なインテル® MPI 統計情報と追加のコレクターを使用して、MPI アプリケーションのハードウェア・カウンター、メモリー消費量、MPI 内部のインバランス、および OpenMP* インバランス (アプリケーションが OpenMP* を使用する場合) などの統計情報を収集するには、このオプションを使用します。このオプションを使用すると、2 つのテキストファイルが作成されます: `stats.txt` と `app_stat.txt`。`stats.txt` ファイルはインテル® MPI ライブラリーのネイティブ統計情報を含み、`app_stat.txt` ファイルは MPI Performance Snapshot によるアプリケーションの統計情報を含みます。これらのファイルは、`mps` ユーティリティーで解析できます。`mps` ユーティリティーを使用してインテル® MPI 統計を簡単に解析できます。

例えば、統計を収集するには、次のコマンドを使用します。

```
$ mpirun -mps –n 2 ./myApp
```

詳細は、「ネイティブ統計形式」をご覧ください。

注意

1. このオプションを使用してアプリケーションを実行する前に、インテル® Trace Analyzer & Collector のインストール・ディレクトリーにある `mpsvars.sh[csh]` スクリプトを使用して環境を設定します。
2. 詳しい情報は、インテル® Parallel Studio XE Cluster Edition のインストール先にある、`<installdir>/itac_latest/doc/MPI_Perf_Snapshot_User_Guide.pdf` (MPI Performance Snapshot for Linux* Guide) をご覧ください。
3. コマンドラインに `-traceor` または `-check_mpi` オプションを指定すると、`–mps` オプションは無視されます。

(SDK のみ) -check_mpi [<checking_library>]

指定された <checking_library> を使用して MPI アプリケーションをチェックするには、このオプションを指定します。<checking_library> が省略された場合、デフォルトのチェック・ライブラリーは、`libVTmc.so` です。

デフォルトのチェック・ライブラリーを変更するには、「I_MPI_JOB_TRACE_LIBS」環境変数を設定します。

(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 <filename>

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 <filename> 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 <num>

mpiexec.hydra 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。

-tv

TotalView* デバッガー環境下で <executable> を実行するには、このオプションを使用します。次に例を示します。

```
$ mpiexec.hydra -tv -n <# of processes> <executable>
```

TotalView* 実行形式ファイルを選択する方法は、「環境変数」をご覧ください。

注意

TotalView* は、デフォルトで rsh を使用します。ssh を使用する場合、TVDSVRLAUNCHCMD 環境変数に ssh を設定します。

注意

TotalView* デバッガーは、MPI プログラムのメッセージキューの状態を表示できます。この機能を有効にするには、次の手順に従ってください。

1. <executable> を –tv オプションで実行します。

   ```
   $ mpiexec.hydra -tv -n <# of processes> <executable>
   ```

2. mpiexec.hydra ジョブの停止に関する問い合わせには、「YES」を選択します。

MPI ライブラリーの内部状態をテキストで表示するには、**[Tools] > [Message Queue]** コマンドを選択します。**[Process Window Tools] > [Message Queue Graph]** コマンドを選択すると、TotalView* 環境変数は現在のメッセージキューの状態をグラフ形式でウィンドウに表示します。詳細は、「TOTALVIEW」環境変数をご覧ください。

-tva <pid>

TotalView* デバッガーに実行中のインテル® MPI ライブラリーのジョブをアタッチするには、このオプションを使用します。

<pid> として、mpiexec.hydra プロセス ID を使用します。次のコマンドを使用できます。

```
$ mpiexec.hydra -tva <pid>
```

-gdb

GNU* デバッガー環境下で <executable> を実行するには、このオプションを使用します。次のコマンドを使用できます。

```
$ mpiexe.hydra -gdb -n <# of processes> <executable>
```

-gdba <pid>

GNU* デバッガーに実行中のインテル® MPI ライブラリーのジョブをアタッチするには、このオプションを使用します。次のコマンドを使用できます。

```
$ mpiexec.hydra -gdba <pid>
```

-gtool

mpiexec.hydra コマンドで指定されたランク上でインテル® VTune™ Amplifier XE、Valgrind*、GNU* デバッガーなどのツールを起動するには、このオプションを使用します。

注意

-gtool オプションでデバッガーを起動することを指定していない場合を除いて、-gdb オプションと -gtool オプションを同時に指定してはいけません。

構文

```
-gtool “<command line for a tool 1>:<ranks set 1>[=lanuch mode 1][@arch 1];
<command line for a tool 2>:<ranks set 2>[=exclusive][@arch 2]; … ;<command line
for a tool n>:<ranks set n>[=exclusive][@arch n]” <executable>
```

または

```
$ mpiexec.hydra -n <# of processes> -gtool “<command line for a tool 1>:<ranks set
1>[=launch mode 1][@arch 1]” -gtool “<command line for a tool 2>:<ranks set
2>[=launch mode 2][@arch 2]” … -gtool “<command line for a tool n>:<ranks set
n>[=launch mode 3][@arch n]” <executable>
```

注意

構文では、セパレーター「;」と -gtool オプションはどちらも使用できます。

引数

<arg>	パラメーター。
<rank set>	ツールの実行に関連するランクの範囲を指定します。カンマもしくは「-」で区切って、連続したランクを指定することができます。 **注意** 不正なランクのインデックスが指定された場合、警告が出力され、ツールは正しいランクを使用して処理を継続します。
[=launch mode]	起動モードを指定します。
[@arch]	ツールに適用するアーキテクチャーを指定します。<rank set> を指定すると、ツールは特定のアーキテクチャーを持つホストに割り当てられたランクのみに適用されます。このパラメーターはオプションです。[@arch] の値については、「I_MPI_PLATFORM」の引数テーブルの説明ご覧ください。 インテル® Xeon Phi™ コプロセッサー上でデバッガーを起動する場合、[@arch] 設定が必要です。詳細は、例を参照してください。

注意

ランクのセットは交差することはできません。例えば、-gtool オプションにパラメーターがない、または [@arch] と同じパラメーターがある。しかし、2 つのセットが交差しないように明確に異なる [@arch] パラメーターで同じランクを指定することはできます。単一の mpiexec.hydra 起動内で適用するツールを指定する必要があります。いくつかのツールは同時に動作しないか、誤った結果につながる可能性があります。

次の表に [=launch mode] に指定できるパラメーターの一覧を示します。

引数

exclusive	ホストごとに複数ランク向けのツールを起動しないようにするには、この値を指定します。これはオプションです。
attach	–gtool オプションから実行可能ファイルにツールをアタッチするには、この値を指定します。デバッガーやデバッガー方式でプロセスにアタッチできるほかのツールを使用する場合、この値を [=launch mode] に指定する必要があります。現在の実装では、デバッガー向けのテストのみが行われています。
node-wide	–gtool オプションから実行可能ファイルより高いレベル (pmi_proxy デーモンへ) にツールを適用するには、この値を指定します。このパラメーターは将来のリリースで実装される予定です。

各ツールに複数の値を指定することができます。この場合、カンマでツールを区切ります。次に例を示します。

```
$ mpiexec.hydra -gtool "gdb-ia:all=attach,exclusive; /usr/bin/gdb:all=exclusive,
attach@knc" -host <hostname> -n 2 <app> : -host <hostname-mic0> -n 2 <mic-app>
```

この場合、インテルバージョンの GNU* GDB (gdb-ia) とインテル® Xeon Phi™ コプロセッサー向けのネイティブ GNU* GDB が、ホスト上のランクとコプロセッサー上のランク向けにそれぞれ起動されます。

例

次のコマンドは、–gtool オプションを使用したもう 1 つの例です。

1. mpiexec.hydra コマンドを経由して、インテル® VTune™ Amplifier XE と Valgrind* を起動します。

   ```
   $ mpiexec.hydra -n 16 -gtool “amplxe-cl -collect advanced-hotspots
   -analyze-system -r result1:5,3,7-9=exclusive@nhm;valgrind -log-file=log_%p
   :0,1,10-12@wsm” a.out
   ```

   指定したランクセットからインテル® マイクロアーキテクチャー (開発コード名 Nehalem) のホスト上に割り当てられた最小のインデックスを持つランクで amplxe-cl を実行するには、このコマンドを使用します。同時に、指定したランクセットからインテル® マイクロアーキテクチャー (開発コード名 Westmere) のホスト上のすべてのランクで Valgrind* を実行します。Valgrind の結果は、ファイル名 log_<process ID> に出力されます。

2. mpiexec.hydra コマンドを経由して、GNU* デバッガー (GDB) を起動します。

   ```
   $ mpiexec.hydra -n 16 -gtool “gdb:3,5,7-9=attach” a.out
   ```

   このコマンドは、指定したランクセットに gdb を適用します。

3. 特定のランクセット向けに環境変数を設定します。

   ```
   $ mpiexec.hydra -n 16 -gtool “env VARIABLE1=value1 VARIABLE2=value2:3,5,7-9;
   env VARIABLE3=value3:0,11” a.out
   ```

   ランクセット 3、5、7、8、9 向けに VARIABLE1 と VARIABLE2 を設定し、ランク 0 と 11 に VARIABLE3 を設定するには、このコマンドを使用します。

4. mpiexec.hydra コマンドを経由して、指定したホストでアプリケーションをデバッグします。

   mpiexec.hydra コマンドを介してデバッガーを設定する場合、対応するデバッガーに次のオプションを使用できます。-gtool オプションを使用して、その他のデバッガーも起動できます。

   - 標準 GNU* デバッガー (GDB): gdb
   - インテルバージョンの GNU* デバッガー: gdb-ia
   - インテル® Xeon Phi™ コプロセッサー向けネイティブ GNU* デバッガー: gdb

   -gtool オプションは、ホストとインテル® Xeon Phi™ コプロセッサー向けデバッガーを同時にサポートできます。この場合、インテル® Xeon Phi™ コプロセッサー・マシンに割り当てられるランクに @arch パラメーターを指定します。インテル® Xeon Phi™ コプロセッサー (開発コード名 Knights Corner) アーキテクチャー向けにコマンドで設定するには、@arch=knc を使用します。

   注意

   インテル® Xeon Phi™ コプロセッサーを搭載するホストや異種クラスターでデバッグを行う際、I_MPI_MIC 環境変数を設定してインテル® Xeon Phi™ コプロセッサーの認識を有効にする必要があります。環境変数の指定方法は、「I_MPI_MIC」をご覧ください。

   @arch を指定すると、交差してランクセットを指定できます。

   バイナリー互換クラスター上でデバッグセッションが起動する場合、@arch パラメーターは省略できます。generic 値は、インテル® Xeon Phi™ コプロセッサーを搭載するプラットフォームを除くすべてのプラットフォームをカバーします。

   デバッガーが異種クラスター上で起動され、インテル® Xeon Phi™ コプロセッサー向けのデバッガーが指定されていない場合、デフォルトでコプロセッサー上の /usr/bin/gdb が起動されます。

   次に例を示します。

   a. `$ mpiexec.hydra -n 16 -gtool “gdb:3,5=attach;gdb-ia:7-9=attach” a.out`

      この場合、ランク番号 3、5、7、8、9 向けに標準 GNU* デバッガー (gdb が起動されます)。

   b. `$ mpiexec.hydra -gtool “gdb-ia:all=attach@generic; /tmp/gdb:all=attach@knc” -host <hostname> -n 8 <host-app> : -host <hostname-mic0> -n 8 <mic-app>`

      この場合、すべての hostname のランク向けにインテルバージョンの GNU* デバッガー(gdb-ia) が起動されます。インテル® Xeon Phi™ コプロセッサー向けのネイティブ GNU* GDB は、コプロセッサー上に割り当てられたすべてのランクに適用されます。次のコマンドは、ランクセットで異なるアーキテクチャーを指定した場合、ランクセットを交差できる例を示します。

   c. `$ mpiexec.hydra -gtool “gdb-ia:3,5,7-9” -host <hostname> -n 8 <host-app> : -host <hostname-mic0> -n 8 <mic-app>`

      コマンドを入力すると、hostname マシン上に割り当てられたランクセットのすべてのランク向けにインテルバージョンの GNU* GDB (gdb-ia) が起動されます。コプロセッサーに割り当てられているランクセットの任意のランクが、自動的にインテル® Xeon Phi™ コプロセッサー対応のネイティブ GNU* GDB (/usr/bin/gdb) を使用します。

5. <machine file> オプションを使用して、特定のランクにツールを適用します。この例では、m_file ファイルが次の内容を保持していると仮定します。

   ```
   hostname_1:2
   hostname_2:3
   hostname_3:1
   ```

次のコマンドラインは、–machinefile オプションを使用してツールに適用する例を示します。

```
$ mpiexec.hydra -n 6 -machinefile m_file -gtool “amplxe-cl -collect
advanced-hotspots -analyze-system -r
result1:5,3=exclusive@nhm;valgrind:0,1@wsm” a.out
```

ここで、–machinefie オプションは、hostname_1 マシンにランク・インデックス 0 と 1 が、hostname_2 マシンにランク・インデックス 3 が、そして hostname_3 マシンにランク・インデックス 5 が配置されることを意味します。その後、hostname_2 と hostname_3 マシンは、インテル® マイクロアーキテクチャー (開発コード名 Nehalem) であるため、ランク 3 と 5 のみで amplxe-cl が実行されます (これらのランクは異なるマシンに属しているため、排他的にオプションが適用されます)。同時に、Valgrind* ツールがインテル® マイクロアーキテクチャー (開発コード名 Westmere) である hostname_1 に割り当てられるランク (0 と 1) に適用されます。

6. ランクがクラスターノード間に分散される様子を表示します。-gtool オプションと z show map コマンドを使用します。コマンドの詳しい説明を見ることができます。

```
$ mpiexec.hydra -gtool "gdb-ia:0,2,3,9,10=attach;/tmp/gdb:5,6=attach@knc"
-host <hostname_1> -n 4 <host-app> : -host <hostname_1-mic0> -n 4 <mic-
app> : -host <hostname_2> -n 4 <host-app>
[0,2,3,5,6,9,10] (mpigdb) z show map
[0,2,3]: hostname_1
[5,6]: hostname_1-mic0
[9,10]: hostname_2
[0,2,3,5,6,9,10] (mpigdb) z help
z <positive number(s) up to 11 or all> - Sets ranks for debugging
z show map - Shows ranks distribution across the cluster nodes
z help - Shows help information
[0,2,3,5,6,9,10] (mpigdb)
```

-gtoolfile <gtool_config_file>

mpiexec.hydra コマンドで指定されたランク上でインテル® VTune™ Amplifier XE、Valgrind*、GNU* デバッガーなどのツールを起動するには、このオプションを使用します。

例

gtool_config_file に次の設定が含まれている場合:

```
env VARIABLE1=value1 VARIABLE2=value2:3,5,7-9; env VARIABLE3=value3:0,11 env
VARIABLE4=value4:1,12
```

次のコマンドで、ランクセット 3、5、7、8、9 向けに VARIABLE1 と VARIABLE2 を設定し、ランク 0 と 11 に VARIABLE3 を、VARIABLE4 を最初のランクと 12 番目のランクに設定します。

```
$ mpiexec.hydra -n 16 -gtoolfile gtool_config_file a.out
```

注意

オプション -gtool、-gtoolfile と環境変数 I_MPI_GTOOL は、互いに排他的です。オプション –gtool と –gtoolfile の優先順位は同じレベルです。コマンドラインでは最初に指定されたオプションが有効で、2 番目のオプションは無視されます。–gtool と -gtoolfile オプションは、I_MPI_GTOOL 環境変数よりも高い優先順位を持ちます。そのため、mpiexec.hydra のコマンドラインで –gtool や –gtoolfile オプションを指定していないときに、I_MPI_GTOOL 環境変数を使用します。

-nolocal

mpiexec.hydra が起動されたホスト上で <executable> の実行を避けるには、このオプションを使用します。MPI ジョブを開始する専用のマスターノードと、実際の MPI プロセスと実行する専用の計算ノードを配備するクラスター上でこのオプションを使用できます。

-hosts <nodelist>

MPI プロセスを実行する特定の <nodelist> 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.out を実行します。

```
$ mpiexec.hydra -n 2 -ppn 1 -hosts host1,host2 ./a.out
```

注意

<nodelist> 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 <interface>

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.hydra -n 2 -iface ib0 ./a.out
```

詳細は、「I_MPI_HYDRA_IFACE」環境変数をご覧ください。

-demux <mode>

複数の I/O 向けのポーリングモデルを設定するには、このオプションを使用します。デフォルト値は poll です。

引数

<spec>	複数の I/O 向けのポーリングモデルを定義します。
poll	ポーリングモデルとして poll を設定します。これは、デフォルト値です。
select	ポーリングモデルとして select を設定します。

詳細は、「I_MPI_HYDRA_DEMUX」環境変数をご覧ください。

-enable-x または -disable-x

Xlib* のトラフィック・フォワードを制御するには、このオプションを使用します。デフォルトは、Xlib* トラフィックはフォワードされない -disable-x です。

-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 [<arg >]

ここで以下を指定します。

```
<arg>= {<dir_name>, <configuration_file>}
```

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。

<arg> 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位置は、<installdir>/<arch>/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、<arg>=<dir_name> を設定します。異なる設定ファイルを指定するには、<arg>=<configuration_file> 設定します。

-ilp64

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。詳細ついては、「ILP64 サポート」をご覧ください。

-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.hydra 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 <directory>

実行する <executable> ファイルへのパスを指定します。

-cleanup

起動されたプロセスの情報を含む一時ファイルを作成するには、このオプションを使用します。ファイル名は、mpiexec_${username}_$PPID.log です。PPID は親プロセスの PID です。このファイルは、-tmpdir オプションで指定された一時ディレクトリーに作成されます。このファイルは、mpicleanup ユーティリティーで使用されます。ジョブが正常に終了すると、mpiexec.hydra コマンドは自動的にこのファイルを削除します。

詳細は、「I_MPI_HYDRA_CLEANUP」環境変数をご覧ください。

-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。

-use-app-topology <value>

統計ファイルまたはクラスタートポロジーから転送されたデータを基に、ランクを再配置するにはこのオプションを使用します。次のコマンドを使用できます。

```
$ mpiexec.hydra –use-app-topology ./stats.txt <…> ./my_app
```

引数

<value>	インテル® MPI ライブラリーのネイティブ統計ファイルのレベル 1 以上へのパス。

注意

Hydra PM は、スタティック・メソッドの mpitune_rank_placement と同様の方法で libmpitune.so の API を使用し、ランク割り当てにホストリストの結果を使用します。

詳細は、「I_MPI_HYDRA_USE_APP_TOPOLOGY」と「トポロジーを考慮したアプリケーションのチューニング」をご覧ください。

-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.hydra -localhost <localhost_ip> -machinefile <file> -n 2 ./a.out
```

ブートオプション

-bootstrap <bootstrap server>

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。Hydra は、MPI プロセスを起動するため、ssh、rsh、pdsh、fork、persist、slurm、ll、lsf、sge、または jmi などの複数のランタイム・ブートストラップ・サーバーをサポートします。デフォルトのブートストラップ・サーバーは ssh です。slurm、ll、lsf、または sge を選択すると、対応する srun、llspawn、stdio、blaunch、または qrsh ジョブ・スケジューラー・ユーティリティーを使用して、それぞれ選択されたジョブ・スケジューラーの下で (SLURM*、LoadLeveler*、LSF*、および SGE*) サービスプロセスを起動します。

引数

<arg>	文字列パラメーター。
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。
rsh	リモートシェルを使用します。
pdsh	並列分散シェルを使用します。
pbsdsh	Torque* と PBS* pbsdsh コマンドを使用します。
fork	fork 呼び出しを使用します。
persist	Hydra persist サーバーを使用します。
slurm	SLURM* srun コマンドを使用します。
ll	LoadLeveler* llspawn.stdi コマンドを使用します。
lsf	LSF* blaunch コマンドを使用します。
sge	Univa* Grid Engine* qrsh コマンドを使用します。
jmi	Job Manager Interface (より密接な統合) を使用します。

SLURM* または PBS Pro* ジョブ管理との密接な統合を有効にするには、jmi ブートストラップ・サーバーを使用します。密接な統合には、対応するジョブ管理によるプロセス識別子の登録も含まれます。この設定では、ジョブ終了時に対応するジョブ管理による効率良いリソース管理とノードのクリーンアップが可能になります。

注意

リモートプロセスの並列起動に使用する一部のブートストラップ・サーバー (slurm と pdsh) では、異種環境で動作しない可能性があります (例えば、I_MPI_MIC が有効に設定されている場合など)。

詳細は、「-bootstrap jmi」の説明と「I_MPI_HYDRA_BOOTSTRAP」環境変数をご覧ください。

-bootstrap-exec <bootstrap server>

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。デフォルトのブートストラップ・サーバーは ssh です。次に例を示します。

```
$ mpiexec.hydra -bootstrap-exec <bootstrap_server_executable> \
-f hosts.file -env <VAR1> <VAL1> -n 2 ./a.out
```

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

-bootstrap-exec-args <args>

ブートストラップ・サーバーの実行形式へ追加パラメーターを提供するには、このオプションを使用します。

```
$ mpiexec.hydra -bootstrap-exec-args <arguments> -n 2 ./a.out
```

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

-bootstrap persist

Hydra persist サーバーを使用して MPI プロセスを起動するには、このオプションを使用します。ジョブを実行する前に、各ホスト上でサーバーを開始してください。

```
$ hydra_persist&
```

注意

root アカウントでサービスを開始しないでください。サーバーは、Linux* シェルから kill コマンドを使用してシャットダウンできます。

-bootstrap jmi

SLURM* または PBS Pro* ジョブ管理との密接な統合を有効にするには、このオプションを使用します。密接な統合は、それぞれのジョブ・スケジューラーの API (アプリケーション・プログラミング・インターフェイス) やユーティリティーを使用して実装されます。このオプションを指定すると、デフォルトの `libjmi.so` ライブラリーがロードされます。デフォルトのライブラリー名は、`I_MPI_HYDRA_JMI_LIBRARY` 環境変数で変更できます。

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

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"<parameter>=<value>[;<parameter>=<value>  ...]"
```

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、ソケットごとにプロセッサー数/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をとなります。 • 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 `-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。
<size>[:<layout>]	セットの各ドメインは、<size> のプロセッサーセルで構成されます。<size> 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 <layout> 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 <layout> 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。

通信サブシステムのオプション

-rmk <RMK>

使用されるリソース管理カーネルを選択するには、このオプションを使用します。インテル® MPI ライブラリーは、pbs のみをサポートします。詳細は、「I_MPI_HYDRA_RMK」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.hydra 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。

2.3.2. ローカルオプション

-n <# of processes> または -np <# of processes>

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 <ENVVAR> <value>

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 <value> の環境変数 <ENVVAR> 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 <list of env var names>

引数リストと現在の値を渡します。<list of env var names> 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 <nodename>

MPI プロセスを実行する特定の <nodename> 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.out を実行します。

```
$ mpiexec.hydra -n 2 -host host1 ./a.out : -n 2 -host host2 ./a.out
```

-path <directory>

現在の引数セットで実行する <executable> ファイルへのパスを指定します。

-wdir <directory>

現在の引数セットで実行する <executable> 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。

-umask <umask>

リモートの <executable> ファイルに umask <umask> コマンドを実行します。

-hostos <host 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 linux です。

引数

<arg>	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。これは、デフォルト値です。
windows	ホストには Windows* がインストールされています。

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 b.out を実行します。

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

2.3.3. 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,tmi,ofi 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 dapl,ofa,tcp,tmi,ofi -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,tmi,ofa,ofi 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 dapl,tcp,tmi,ofa,ofi -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,tmi,ofi 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,tmi,ofi -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。

-tmi

TMI 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、TMI 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,ofa,ofi 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 tmi,dapl,tcp,ofa,ofi -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-TMI

TMI 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 tmi -genv I_MPI_FALLBACK 0 オプションを指定するのと等価です。

-mx

Myrinet MX* 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、Myrinet MX* 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,ofa,ofi 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 tmi,dapl,tcp,ofa,ofi -genv I_MPI_TMI_PROVIDER mx -genv I_MPI_DAPL_PROVIDER mx -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-MX

Myrinet MX* 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 tmi -genv I_MPI_TMI_PROVIDER mx - genv I_MPI_FALLBACK 0` オプションを指定するのと等価です。

-psm

PSM ネットワーク・ファブリックを選択します: PSM 互換モードのインテル® True Scale ファブリックまたはインテル® Omni-Path ファブリック。アプリケーションは、PSM 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,ofa,ofi` 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 tmi,dapl,tcp,ofa,ofi -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 1` オプションを指定するのと等価です。

-PSM

PSM ネットワーク・ファブリックを選択します: PSM 互換モードのインテル® True Scale ファブリックまたはインテル® Omni-Path ファブリック。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 tmi -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 0` オプションを指定するのと等価です。

-psm2

インテル® Omni-Path ファブリックを選択します。アプリケーションは、インテル® Omni-Path ファブリックの使用を試みます。利用できない場合、`dapl,tcp,ofa,ofi` 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 tmi,dapl,tcp,ofa,ofi -genv I_MPI_TMI_PROVIDER psm2 -genv I_MPI_FALLBACK 1` オプションを指定するのと等価です。

-PSM2

インテル® Omni-Path 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 tmi -genv I_MPI_TMI_PROVIDER psm2 -genv I_MPI_FALLBACK 0` オプションを指定するのと等価です。

-ofi

OpenFabrics Interfaces* (OFI*) 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、`tmi,dapl,tcp,ofa` 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 ofi,tmi,dapl,tcp,ofa -genv I_MPI_FALLBACK 1` オプションを指定するのと等価です。

-OFI

OFI 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 ofi -genv I_MPI_FALLBACK 0` オプションを指定するのと等価です。

2.3.4. 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=<arg>`

廃止された構文

`HYDRA_HOST_FILE=<arg>`

引数

`<arg>`	文字列パラメーター。
`<hostsfile>`	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=<arg>`

引数

`<arg>`	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=<arg>`

引数

`<arg>`	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.hydra` のタイムアウト時間を設定します。

構文

```
I_MPI_JOB_TIMEOUT=<timeout>
I_MPI_MPIEXEC_TIMEOUT=<timeout>
```

廃止された構文

MPIEXEC_TIMEOUT=<timeout>

引数

<timeout>	mpiexec.hydra 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.hydra がジョブの起動後 <timeout> 秒でジョブを強制終了する時間を設定します。<timeout> 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.hydra 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。<timeout> 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=<number>

廃止された構文

MPIEXEC_TIMEOUT_SIGNAL=<number>

引数

<number>	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際に、MPI ジョブを停止するために送信するシグナルを定義します。システムがサポートしないシグナル番号を設定した場合、mpiexec.hydra 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=<number>

引数

<number>	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.hydra 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=<arg>
```

廃止された構文

```
MPIEXEC_SIGNAL_PROPAGATION=<arg>
```

引数

<arg>	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=<arg>
```

引数

<arg>	文字列パラメーター。
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。
rsh	リモートシェルを使用します。
pdsh	並列分散シェルを使用します。
pbsdsh	Torque* と PBS* pbsdsh コマンドを使用します。
fork	fork 呼び出しを使用します。
slurm	SLURM* srun コマンドを使用します。
ll	LoadLeveler* llspawn.stdi コマンドを使用します。

lsf	LSF* blaunch コマンドを使用します。
sge	Univa* Grid Engine* qrsh コマンドを使用します。
jmi	Job Manager Interface (より親密な統合) 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.hydra 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。<arg> 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=<arg>

引数

<arg>	文字列パラメーター。
<executable>	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_BOOTSTRAP_EXEC_EXTRA_ARGS

ブートストラップ・サーバーへの追加の引数を設定します。

構文

I_MPI_HYDRA_BOOTSTRAP_EXEC_EXTRA_ARGS=<arg>

引数

<arg>	文字列パラメーター。
<args>	追加のブートストラップ・サーバーの引数。

説明

この環境変数は、ブートストラップ・サーバーに追加の引数を設定します。

I_MPI_HYDRA_BOOTSTRAP_AUTOFORK

ローカルプロセス向けの fork 呼び出しの使い方を制御します。

構文

I_MPI_HYDRA_BOOTSTRAP_AUTOFORK = <arg>

引数

<arg>	文字列パラメーター。
enable | yes | on | 1	ローカルプロセス向けに fork を使用します。これは、ssh、rsh、ll、lsf、pbsdsh ブートストラップ・サーバー向けのデフォルト値です。
disable | no | off | 0	ローカルプロセス向けに fork を使用しません。これは、sge ブートストラップ・サーバー向けのデフォルト値です。

説明

ローカルプロセス向けの fork 呼び出しの使い方を制御するには、この環境変数を設定します。

注意

このオプションは、slurm、pdsh、persist、jmi ブートストラップ・サーバーには適用されません。

I_MPI_HYDRA_RMK

リソース管理カーネルを使用します。

構文

```
I_MPI_HYDRA_RMK=<arg>
```

引数

<arg>	文字列パラメーター。
<rmk>	リソース管理カーネル。サポートされる値は、pbs のみです。

説明

pbs リソース管理カーネルを使用するには、この環境変数を設定します。インテル® MPI ライブラリーは、pbs のみをサポートします。

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=<value>
```

引数

<value>	使用するアルゴリズム。
ocache	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	オンデマンドのキャッシュモードの伝搬。これは、デフォルト値です。

alltoall	任意の取得要求が完了する前に、情報はすべての pmi_proxy 間で自動的に交換されます。

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.hydra 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=<value>

引数

<value>	-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。
integer > 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_JOB_TRACE_LIBS

–trace オプションを介して事前ロードするライブラリーを選択します。

構文

I_MPI_JOB_TRACE_LIBS=<arg>

廃止された構文

MPIEXEC_TRACE_LIBS=<arg>

引数

<arg>	文字列パラメーター。
<list>	スペース (空白で) 区切られた、事前ロードするライブラリー。デフォルト値は vt です。

説明

–trace オプションを介して事前ロードする代替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。

I_MPI_JOB_CHECK_LIBS

-check_mpi オプションを介して事前ロードするライブラリーを選択します。

構文

I_MPI_JOB_CHECK_LIBS=<arg>

引数

<arg>	文字列パラメーター。
<list>	スペース (空白で) 区切られた、事前ロードするライブラリー。デフォルト値は vtmc です。

説明

-check_mpi オプションを介して事前ロードする代替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。

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 =<num>

引数

<num>	番号。
<n> >=0	• ノードが 128 未満の場合、デフォルト値は -1 です。これは階層構造がな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

mpiexec.hydra 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するには、この環境変数を設定します。

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=<arg>

引数

<arg>	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_GDB_REMOTE_SHELL

GNU* デバッガーを実行するリモートシェル・コマンドを設定します。

構文

I_MPI_HYDRA_GDB_REMOTE_SHELL=<arg>

引数

<arg>	文字列パラメーター。
ssh	セキュアシェル (SSH)。これは、デフォルト値です。
rsh	リモートシェル (RSH)。

説明

リモートマシン上で GNU* デバッガーを実行するリモートシェル・コマンドを設定するには、この環境変数を設定します。SSH や RSH と同じ形式でリモートシェル・コマンドを指定するため、この環境変数を設定を使用できます。

I_MPI_ HYDRA_JMI_LIBRARY

JMI ライブラリーのデフォルト設定を定義します。

構文

I_MPI_HYDRA_JMI_LIBRARY=<value>

引数

<value>	JMI ダイナミック・ライブラリー名もしくはパスを文字列で定義します。
libjmi_slurm.so.1.1 | libjmi_pbs.so.1.0	ライブラリー名をフルパスで設定します。デフォルト値は libjmi.so です。

説明

Hydra プロセス管理によってロードされる JMI ライブラリーを定義するには、この環境変数を設定します。LD_LIBRARY_PATH 環境変数にパスが登録されていない場合、ライブラリーへのフルパスを設定します。mpirun コマンドを使用している場合、この環境変数を設定する必要はありません。JMI ライブラリーは、自動検知されて設定されます。

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=<arg>

引数

<arg>	文字列パラメーター。
<network interface>	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_HYDRA_DEMUX

demultiplexer (demux) モードを設定します。

構文

`I_MPI_HYDRA_DEMUX=<arg>`

引数

`<arg>`	文字列パラメーター。
`poll`	複数 I/O demultiplexer (demux) モードエンジンとして `poll` を設定します。これは、デフォルト値です。
`select`	複数 I/O demultiplexer (demux) モードエンジンとして `select` を設定します。

説明

複数 I/O demux モードエンジンを指定するには、この環境変数にを設定します。デフォルト値は `poll` です。

I_MPI_HYDRA_CLEANUP

デフォルトの `mpicleanup` 入力ファイルの作成を制御します。

構文

`I_MPI_HYDRA_CLEANUP=<value>`

引数

`<value>`	バイナリー・インジケーター。
`enable | yes | on | 1`	`mpicleanup` 入力ファイルの作成を有効にします。
`disable | no | off | 0`	`mpicleanup` 入力ファイルの作成を無効にします。これは、デフォルト値です。

説明

`mpicleanup` ユーティリティーの入力ファイルを作成するには、`I_MPI_HYDRA_CLEANUP` 環境変数を設定します。

I_MPI_TMPDIR (TMPDIR)

一時ディレクトリーを設定します。

構文

`I_MPI_TMPDIR=<arg>`

引数

`<arg>`	文字列パラメーター。
`<path>`	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=<arg>
```

引数

`<value>`	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` 環境変数に指定される PPN を使用します。

I_MPI_GTOOL

選択されたランク向けに起動されるツールを指定します。

構文

```
I_MPI_GTOOL="<command line for a tool 1>:<ranks set 1>[=exclusive][@arch 1];
<command line for a tool 2>:<ranks set 2>[=exclusive][@arch 2]; … ;<command line
for a tool n>:<ranks set n>[=exclusive][@arch n]"
```

引数

`<arg>`	パラメーター。
`<command line for a tool>`	パラメーターとともにツールを指定します。
`<rank set>`	ツールの実行に関連するランクの範囲を指定します。カンマもしくは「-」で区切って、連続したランクを指定することができます。 **注意** 不正なランクのインデックスが指定された場合、ツールは警告を出力し、正しいランクを使用して処理を継続します。
`[=exclusive]`	ホストごとに複数ランク向けのツールを起動しないようにするには、このパラメーターを指定します。このパラメーターはオプションです。
`[@arch]`	ツールに適用するアーキテクチャーを指定します。`<rank set>` を指定すると、ツールは特定のアーキテクチャーを持つホストに割り当てられたランクのみに適用されます。このパラメーターはオプションです。`[@arch]` の値については、`I_MPI_PLATFORM` の引数テーブルの説明ご覧ください。 インテル® Xeon Phi™ コプロセッサー上でデバッガーを起動する場合、`[@arch]` 設定が必要です。詳細は、例を参照してください。

説明

インテル® VTune™ Amplifier XE、Valgrind*、GNU* デバッガーなどのツールを特定のランク上で起動するには、このオプションを使用します。

例

次のコマンドは、異なる状況で I_MPI_GTOOL 環境変数をを使用する例です。

1. I_MPI_GTOOL 環境変数を設定して、インテル® VTune™ Amplifier XE と Valgrind* を起動します。

   ```
   $ export I_MPI_GTOOL="amplxe-cl -collect advanced-hotspots -analyze-system
   -r result1:5,3,7-9=exclusive@nhm;valgrind -log-file=log_%p :0,1,10-12@wsm"
   $ mpiexec.hydra -n 16 a.out
   ```

   指定したランクセットからインテル® マイクロアーキテクチャー (開発コード名 Nehalem) のホスト上に割り当てられた最小のインデックスを持つランクで amplxe-cl を実行するには、このコマンドを使用します。同時に、指定したランクセットからインテル® マイクロアーキテクチャー (開発コード名 Westmere) のホスト上のすべてのランクで Valgrind* を実行します。Valgrind* の結果は、ファイル名 log_<process ID> に出力されます。

2. I_MPI_GTOOL 環境変数を設定して、GNU* デバッガー (GDB) を起動します。

   ```
   $ mpiexec.hydra -n 16 -genv I_MPI_GTOOL="gdb:3,5,7-9" a.out
   ```

   このコマンドは、指定したランクセットに gdb を適用します。

注意

オプション -gtool、-gtoolfile と環境変数 I_MPI_GTOOL は、互いに排他的です。オプション –gtool と –gtoolfile の優先順位は同じレベルです。コマンドラインでは最初に指定されたオプションが有効で、2 番目のオプションは無視されます。–gtool と -gtoolfile オプションは、I_MPI_GTOOL 環境変数よりも高い優先順位を持ちます。そのため、mpiexec.hydra のコマンドラインで –gtool や –gtoolfile オプションを指定していないときに、I_MPI_GTOOL 環境変数を使用します。

I_MPI_HYDRA_USE_APP_TOPOLOGY

構文

I_MPI_HYDRA_USE_APP_TOPOLOGY=<value>

引数

<value>	インテル® MPI ライブラリーのネイティブ統計ファイルのレベル 1 以上へのパス。

説明

I_MPI_HYDRA_USE_APP_TOPOLOGY が設定されていると、Hydra プロセス管理 (PM) は、統計ファイルまたはクラスタートポロジーから転送されたデータを基に、ランクを再配置します。

```
$ mpiexec.hydra –use-app-topology ./stats.txt <…> ./my_app
```

注意

Hydra PM は、スタティック・メソッドの mpitune_rank_placement と同様の方法で libmpitune.so の API を使用し、ランク割り当てにホストリストの結果を使用します。

詳細は、「-use-app-topology」と「トポロジーを考慮したアプリケーションのチューニング」をご覧ください。

2.3.5. Cleaning up ユーティリティー

mpicleanup

mpiexec.hydra プロセス管理の下で、MPI 実行が異常終了した後、環境をクリーンアップします。

構文

```
mpicleanup [ -i <input_file> | -t -f <hostsfile> ] [ -r <rshcmd> ] \
[ -b <branch_count> ] [-p] [-s | -d] [-h] [-V]
```

または

```
mpicleanup [ --input <input_file> | --total --file <hostsfile> ] \
[ --rsh <rshcmd> ] [ --branch <branch_count> ] [ --parallel ] \
[ --silent | --verbose ] [ --help ] [ --version ]
```

引数

-i <input_file> | --input <input_file>	mpiexec.hydra で生成された入力ファイルを指定します。デフォルト値は mpiexec_${username}_$PPID.log で、環境変数 I_MPI_TMPDIR や TMPDIR で定義される一時ディレクトリー、または /tmp ディレクトリーに配置されます。
-t | --total	指定したマシン上のすべてのユーザープロセスを停止するには、total モードを使用します。このオプションは、root アカウントではサポートされません。
-f <hostsfile> | --file <hostsfile>	クリーンアップするマシンのリストを含むファイルを指定します。
-r <rshcmd> | --rsh <rshcmd>	使用するリモートシェルを指定します。デフォルトは ssh です。
-b <branch_count> | --branch <branch_count>	子プロセスの数を定義します。デフォルト値は 32 です。
-p | --parallel	並列起動モードを使用します。このオプションは、すべてのホストが利用可能な場合にのみ適用できます。さもないと、いくつかのマシンは未定義状態になります。
-s | --silent	出力を抑制します。
-d | --verbose	詳細なデバッグ情報を出力します。
-h | --help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します

説明

MPI ジョブが異常終了した後に、環境をクリーンアップするためこのコマンドを使用します。

例えば、事前の mpiexec.hydra の起動によって作成された入力ファイルに記載されたプロセスを停止するには、次のコマンドを使用します。

```
$ mpicleanup
```

または

```
$ mpicleanup --input /path/to/input.file
```

hostsfile ファイルで指定されているマシン上のすべてのプロセスを停止するには、次のコマンドを使用します。

```
$ mpicleanup --file hostsfile --total
```

2.3.6. チェックポイント・リスタートのサポート

インテル® MPI ライブラリーのチェックポイント・リスタート機能は、アプリケーションから透過に設定されています。MPI プロセス管理インターフェイスを経由して、チェックポイント・リスタート機能にアクセスできます。チェックポイント・リスタートのオプションと環境変数は、Hydra プロセス管理でのみ適用されます。デフォルトのプロセス管理を変更して Hydra プロセス管理を使用するには、`I_MPI_PROCESS_MANAGER=hydra` を設定します。

注意

チェックポイント・リスタート機能を使用するには、OFA* ネットワーク・モジュールが必要です。`I_MPI_FABRICS` 環境変数に `ofa` を設定するか、`-ib` オプションを指定することで、OFA ネットワーク・モジュールを選択できます。

注意

チェックポイント・リスタート機能を有効にするには、次の設定を行います。

- `I_MPI_OFA_DYNAMIC_QPS` 環境変数に `1` を設定
- `I_MPI_OFA_NUM_RDMA_CONNECTIONS` 環境変数に `0` を設定

注意

チェックポイント・リスタート機能を使用するため、Berkeley Lab Checkpoint/Restart* (BLCR) ソフトウェアをインストールします。

グローバルオプション

`-ckpoint <switch>`

引数

`<switch>`	チェックポイントのスイッチ。
`enable | yes | on | 1`	起動されたアプリケーション向けにチェックポイント機能を有効にします。
`disable | no | off | 0`	起動されたアプリケーション向けにチェックポイント機能を無効にします。これは、デフォルト値です。

チェックポイント機能を 有効/無効 にする場合、このオプションを指定します。無効に設定された場合、ほかのチェックポイントのオプションは無視されます。

-ckpoint-interval <sec>

引数

<sec>	秒単位のチェックポイントの間隔。

タイマー駆動型チェックポイントの機能を有効にするには、このオプションを使用します。「タイマー駆動型チェックポイント」をご覧ください。<sec> 秒ごとにチェックポイントが作成されます。このオプションが省略されると、シグナル駆動のチェックポイント機能が使用されます。詳細は、「明示的なシグナル駆動型チェックポイント」をご覧ください。

-ckpoint-preserve <N>

引数

<N>	保持するチェックポイント・イメージの最大数。デフォルト値は 1 です。

チェックポイントのイメージを減らすため、実行中のチェックポイントの最後の <N> チェックポイントを保持するには、このオプションを使用します。デフォルトでは、最後のチェックポイントのみが保持されます。

-restart

保存されたチェックポイントの 1 つからアプリケーションを再開するには、このオプションを使用します。-ckpointlib、-ckpoint-prefix および -ckpoint-num オプションが再開に影響します。実行形式のファイル名はプロセス管理から提供されますが、無視されます。チェックポイントを取得するには再起動したアプリケーションに許されているため、-restart オプションは -chkpoint やほかのチェックポイントオプションを伴うことはできません。

-ckpoint-num <N>

引数

<N>	アプリケーションを再開するチェックポイント・イメージの識別子。有効な値は、最後のチェックポイントの数と同じかそれ以下の数値です。デフォルトは、最後のチェックポイント番号です。

このオプションは、アプリケーションを再開中に使用します。チェックポイント番号 <N> (0 からカウント) が、再開ポイントとして使用されます。最適値を求めるには、-ckpoint-prefix オプションを指定してチェックポイント・ストレージ・ディレクトリーの設定を調査します。

注意

-ckpoint-preserve オプションで決定されるイメージ数が最大数として保持されます。

チェックポイントが存在しない場合、アプリケーションは起動時にエラーメッセージを出力して中断されます。デフォルトでは、最後のチェックポイントが選択されます。

ローカルオプション

-ckpointlib <lib>

引数

<lib>	チェックポイント・リスタートのシステム・ライブラリー。
blcr	Berkeley Lab Checkpoint/Restart* (BLCR) ライブラリー。これは、デフォルト値です。

チェックポイント・リスタート・システムのライブラリーを選択するには、このオプションを使用します。Berkeley Lab Checkpoint/Restart* (BLCR) ライブラリーのみがサポートされます。

注意

チェックポイント関数を使用したり、アプリケーションを再開するときに、同じオプションを指定する必要があります。

-ckpoint-prefix <dir>

引数

<dir>	チェックポイントを保存するディレクトリー。デフォルト値は /tmp です。

チェックポイントを保存するディレクトリーを設定します。デフォルトで、/tmp が使用されます。ディレクトリー <dir> は書き込み可能でなければいけません。さもないと、プロセス起動中にエラーが発生し、アプリケーションはアボートします。

注意

チェックポイント関数を使用したり、アプリケーションを再開するときに、同じオプションを指定する必要があります。

-ckpoint-tmp-prefix <dir>

引数

<dir>	チェックポイントを一時保存するディレクトリー。デフォルト値は /tmp です。

一時チェックポイントを保存するディレクトリーを設定します。チェックポイントは、-ckpoint-tmp-prefix から -ckpoint-prefix で指定されたディレクトリーへ移行します。ディレクトリー <dir> は書き込み可能でなければいけません。さもないと、アプリケーション起動中にアボートします。オプションが設定されていないと、一時ストレージは使用されません。

-ckpoint-logfile <file>

チェックポイントのアクティビティーをモニターするには、このオプションを使用します。トレース結果は、<file> へ出力されます。<file> は書き込み可能でなければいけません。さもないと、アプリケーション起動中にアボートします。これは、オプション機能です。

環境変数

I_MPI_CKPOINT

構文

`I_MPI_CKPOINT=<switch>`

引数

`<switch>`	チェックポイントのスイッチ。
`enable | yes | on | 1`	起動されたアプリケーション向けにチェックポイント機能を有効にします。
`disable | no | off | 0`	起動されたアプリケーション向けにチェックポイント機能を無効にします。これは、デフォルト値です。

説明

チェックポイントの機能を切り替えるには、この環境変数を使用します。このオプションは、`-ckpoint` オプションと同じ効果があります。`-ckpoint` オプションを指定すると、Hydra プロセス管理は `I_MPI_CKPOINT` 環境変数を設定します (もし設定されていなければ)。

I_MPI_CKPOINTLIB

構文

`I_MPI_CKPOINTLIB=<lib>`

引数

`<lib>`	チェックポイント・リスタートのシステム・ライブラリー。
`blcr`	Berkeley Lab Checkpoint/Restart* (BLCR) ライブラリー。これは、デフォルト値です。

説明

チェックポイント・リスタート・システムのライブラリーを選択するには、この環境変数を使用します。このオプションは、`-ckpointlib` オプションと同じ効果があります。

I_MPI_CKPOINT_PREFIX

構文

`I_MPI_CKPOINT_PREFIX=<dir>`

引数

`<dir>`	チェックポイントを保存するディレクトリー。デフォルト値は `/tmp` です。

説明

チェックポイントを保存するディレクトリーを設定します。このオプションは、`-ckpoint-prefix` オプションと同じ効果があります。

I_MPI_CKPOINT_TMP_PREFIX

構文

```
I_MPI_CKPOINT_TMP_PREFIX=<dir>
```

引数

<dir>	チェックポイントを一時保存するディレクトリー。

説明

-ckpoint-prefix が永続的なストレージを示し、この環境変数を使用して一時的なチェックポイントのストレージを示します。このオプションは、-ckpoint-tmp-prefix オプションと同じ効果があります。

I_MPI_CKPOINT_INTERVAL

構文

```
I_MPI_CKPOINT_INTERVAL=<sec>
```

引数

<sec>	秒単位のチェックポイントの間隔。

説明

タイマー駆動型チェックポイントの機能を切り替えるには、この環境変数を使用します。このオプションは、-ckpoint-interval オプションと同じ効果があります。

I_MPI_CKPOINT_PRESERVE

構文

```
I_MPI_CKPOINT_PRESERVE=<N>
```

引数

<N>	保持するチェックポイント・イメージの最大数。デフォルト値は 1 です。

説明

チェックポイントのイメージを減らすため、実行中のチェックポイントの最後の <N> チェックポイントを保持するには、このオプションを使用します。このオプションは、-ckpoint-preserve オプションと同じ効果があります。

I_MPI_CKPOINT_LOGFILE

構文

```
I_MPI_CKPOINT_LOGFILE=<file>
```

引数

<file>	チェックポイントのアクティビティーを保存するトレースファイル。

説明

チェックポイントのアクティビティーを監視するには、このオプションを使用します。トレース結果は、<file> へ出力されます。このオプションは、-ckpoint-logfile オプションと同じ効果があります。

I_MPI_CKPOINT_NUM

構文

```
I_MPI_CKPOINT_NUM=<N>
```

引数

<N>	アプリケーションを再開するチェックポイントのイメージ数。

説明

このオプションは、アプリケーションを再開中に使用します。このオプションは、-ckpoint-num オプションと同じ効果があります。

I_MPI_RESTART

構文

```
I_MPI_RESTART=<switch>
```

引数

<switch>	再開スイッチ。
enable | yes | on | 1	保存されているチェックポイントの 1 つからアプリケーションを再開する機能を有効にします。
disable | no | off | 0	アプリケーションの再開を無効にします。これは、デフォルト値です。

説明

保存されたチェックポイントの 1 つからアプリケーションを再開するには、この環境変数を使用します。このオプションは、–restart オプションと同じ効果があります。

MPI アプリケーションの実行

チェックポイント・リスタートの機能は、Hydra プロセスランチャー (mpiexec.hydra) で利用できます。ランチャーは、チェックポイントを取得するため 2 つの互いに排他的な方式を提供します。

- 時間で
- 明示的なシグナルで

チェックポイントを一時的または永続的に保存するディレクトリーのパスを指定できます。

タイマー駆動型チェックポイント

次の例では、3600 秒 (=1 時間) ごとにチェックポイントが取得されます。チェックポイントは、ckptdir と呼ばれるディレクトリーに保存されます。各ノードは、ノード番号とチェックポイント番号からなる 1 つのチェックポイントを生成します。

```
user@head $ mpiexec.hydra -ckpoint on -ckpoint-prefix /home/user/ckptdir -
ckpoint-interval 3600 -ckpointlib blcr -n 32 -f hosts /home/user/myapp
```

明示的なシグナル駆動型チェックポイント

次の例では、アプリケーションが起動されチェックポイントを取得するためアプリケーションに明示的なシグナル (SIGUSR1) が送信されます。チェックポイントは、ckptdir と呼ばれるディレクトリーに保存されます。

```
user@head $ mpiexec.hydra -ckpoint on -ckpoint-prefix /home/user/ckptdir -
ckpointlib blcr -n 32 -f hosts /home/user/myapp
```

```
...
user@head $ kill -s SIGUSR1 <PID of mpiexec.hydra>
```

ヘッドノードで mpiexec.hydra にシグナルを送るだけで済みます。

ローカル・ストレージを使用する

次の例は、チェックポイントを保存する 2 つの場所を示します。

- 一時的な場所: -ckpoint-tmp-prefix への引数で指定
- 永続的な場所: -ckpoint--prefix への引数で指定

```
user@head $ mpiexec.hydra -ckpoint on -ckpoint-tmp-prefix
/ssd/user/ckptdir -ckpoint-prefix /home/user/ckptdir -ckpointlib blcr –n
32 -f hosts /home/user/myapp
```

MPI アプリケーションの再起動

次のコマンドは、チェックポイント番号 <N> からアプリケーションを再開する例です。

```
user@head $ mpiexec.hydra -restart -ckpoint-prefix /home/user/ckptdir - ckpointlib
blcr -ckpoint-num <N> -n 32 -f hosts
```

再起動後、壊れたまたは利用できないノードを「hosts」ファイルから削除する必要があります。チェックポイント・イメージの中に保存されているため、再開時に実行ファイル名を指定する必要はありません。

ログファイルのチェックポイント・アクティビティーを表示する

次の例では、チェックポイントのアクティビティーを監視できるように MPI ジョブを起動して、チェックポイントのログファイルを指定します。

```
user@head $ mpiexec.hydra -ckpoint on -ckpoint-logfile /home/user/ckpt.log
-ckpoint-tmp-prefix /ssd/user/ckptdir -ckpoint-prefix /home/user/ckptdir
-ckpointlib blcr -n 32 -f hosts /home/user/myapp
```

次はサンプルのログです。

```
[Mon Dec 19 13:31:36 2011] cst-linux Checkpoint log initialized (master mpiexec
pid 10687, 48 processes, 6 nodes)
[Mon Dec 19 13:31:36 2011] cst-linux Permanent checkpoint storage:
/mnt/lustre/user
[Mon Dec 19 13:31:36 2011] cst-linux Temporary checkpoint storage: /tmp [Mon Dec
19 13:32:06 2011] cst-linux Started checkpoint number 0 ...
[Mon Dec 19 13:33:00 2011] cst-linux Finished checkpoint number 0.
[Mon Dec 19 13:33:00 2011] cst-linux Moving checkpoint 0 from /tmp to
/mnt/lustre/user ...
[Mon Dec 19 13:38:00 2011] cst-linux Moved checkpoint 0 from /tmp to
/mnt/lustre/user
```

以前のチェックポイントの自動クリーンアップ

チェックポイントのイメージは大容量です。インテル® MPI ライブラリーは、デフォルトでは最後に使用したチェックポイントのみを保持します。以前のチェックポイントを <N> 個保持するには次のオプションを使用します。-ckpoint-preserve <N>-ckpoint-preserve のデフォルト値は 1 です (最後のチェックポイントのみを保持)。

```
user@head $ mpiexec.hydra -ckpoint on -ckpoint-preserve <N> -ckpoint-tmp-prefix
/ssd/user/ckptdir -ckpoint-prefix /home/user/ckptdir -ckpointlib blcr -n 32 -f
hosts /home/user/myapp
```

2.4. 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のコマンド」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` は、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。

```
$ mpirun -demux select -genv I_MPI_FABRICS shm:tcp -env I_MPI_ROOT \
<windows_installdir> -env PATH <windows_installdir>\\<arch>\\bin -hostos \
windows -host <win_host> -n 2 <windows_installdir>\\<arch>\\bin\\IMB-MPI1 : \
-host <lin_host> -n 3 <linux_installdir>/<arch>/bin/IMB-MPI1
```

2.5. インテル® Xeon Phi™ コプロセッサーのサポート

ここでは、インテル® メニー・インテグレーテッド・コア (インテル® MIC アーキテクチャー) ベースのインテル® Xeon Phi™ コプロセッサー (開発コード名 Knights Corner) に関連するインテル® MPI ライブラリーのトピックについて説明します。

2.5.1. 使用モデル

インテル® Xeon Phi™ コプロセッサー (開発コード名 Knights Corner) ベースのシステムでインテル® MPI ライブラリーを使用するには、次の条件を満たす必要があります。

- それぞれのホストとインテル® Xeon Phi™ コプロセッサーは、ユニークな IP アドレスやシンボル名を持つ必要があります (通常のクラスターと同様)。
- ホストとインテル® Xeon Phi™ コプロセッサー間は、パスワードなしの SSH が確立します。

接続に失敗する場合、次の原因を調査してください。

可能性のある原因	解決方法
インテル® MPSS のバージョンが古い。	新しいバージョンをインストールします。
ホスト上で、`iptables` サービスが起動している。	サービスを停止します。
経路が不正である。	適切な経路を追加します。

IP 接続の構成を設定するには、システム管理者に尋ねるかインテル® MIC ソフトウェア・スタックの readme をご覧ください。

インテル® Xeon Phi™ コプロセッサー上で MPI ライブラリーを使用する場合、インテル® Xeon Phi™ コプロセッサー・カードは、異なるインテル® アーキテクチャーの別のクラスターであることに留意してください。インテル® Xeon Phi™ コプロセッサーで MPI の機能を使用する方法は、インテル® Xeon® プロセッサーの方法と類似しています。

例えば、MPI ライブラリーは、インテル® Xeon® プロセッサー・ホストとインテル® Xeon Phi™ コプロセッサーで同じパスを持つ NFS 共有を介して、インテル® Xeon® プロセッサーおよびインテル® Xeon Phi™ コプロセッサーの両方で利用できます。MPI タスクは、インテル® Xeon® プロセッサー・ホストまたはインテル® Xeon Phi™ コプロセッサーで実行を開始することができます。

インテル® Xeon Phi™ コプロセッサーとホストノードで実行するアプリケーションをビルドするには、次の手順に従ってください。

1. コンパイラーとインテル® MPI ライブラリー向けの環境設定を行います。
   ```
   (host)$ .<compiler_installdir>/bin/compilervars.sh intel64em64t
   (host)$ .<mpi_installdir>/intel64em64t/bin/mpivars.sh
   ```
2. インテル® MIC アーキテクチャー向けにアプリケーションをビルドします。
   ```
   (host)$ mpiicc -mmic test.c -o test_hello.mic
   ```
3. インテル® 64 アーキテクチャー向けにアプリケーションをビルドします。
   ```
   (host)$ mpiicc test.c -o test_hello
   ```

インテル® Xeon Phi™ コプロセッサーとホストノードでアプリケーションを実行するには、次の手順に従ってください。

1. ホストとインテル® Xeon Phi™ コプロセッサー間で NFS が正しく設定されていることを確認してください。これは、インテル® MIC アーキテクチャー上でインテル® MPI ライブラリーを利用する際の推奨方法です。

   インテル® Xeon Phi™ コプロセッサー上で NFS を設定する方法は、http://software.intel.com/en-us/articles/intel-mpi-library-for-linux-kb/all/ (英語) と http://www.isus.jp/article/idz/mic-developer/ をご覧ください。
2. インテル® MPI ライブラリー向けに環境設定を確立します。
   ```
   (host)$ .<mpi_installdir>/intel64em64t/bin/mpivars.sh
   ```
3. ホストから実行形式ファイルを起動します。
   ```
   (host)$ export I_MPI_MIC=1
   ```
   ```
   (host)$ mpirun -n 2 -host <host ID> ./test_hello : -n 2 -host <coprocessor ID> ./test_hello.mic
   ```

注意

`-configfile`、`-hostfile` および `–machinefile` オプションも使用できます。各オプションの使用方法は、それぞれの説明をご覧ください。

インテル® Xeon Phi™ コプロセッサー上でのみアプリケーションを実行するには、インテル® 64 アーキテクチャー向けにアプリケーションを構築する方法を除いて上記の手順に従ってください。また、ホストファイルがインテル® Xeon Phi™ コプロセッサー名のみを含むことを確認します。

詳細については、http://software.intel.com/en-us/articles/intel-mpi-library-for-linux-kb/all/ (英語) と http://www.isus.jp/article/idz/mic-developer/ をご覧ください。

2.5.2. 環境変数

I_MPI_MIC

構文

```
I_MPI_MIC=<value>
```

引数

`<value>`	インテル® Xeon Phi™ の認識。
`enable | yes | on | 1`	インテル® Xeon Phi™ コプロセッサーの認識を有効にします。
`disable | no | off | 0`	インテル® Xeon Phi™ コプロセッサーの認識を無効にします。これは、デフォルト値です。

説明

インテル® MPI ライブラリーがインテル® Xeon Phi™ コプロセッサーを検出し、インテル® MIC アーキテクチャーのコンポーネントを動作させるかどうかを制御するには、この環境変数を設定します。

環境変数 `I_MPI_MIC` が `enable` である場合、環境変数 `I_MPI_SSHM` のデフォルトは `enable` です。

注意

これは暫定的な変数であり、アーキテクチャー検出と関連事項が明確に定義されるまで、一時的に導入されます。

I_MPI_MIC_PREFIX

構文

```
I_MPI_MIC_PREFIX=<value>
```

引数

`<value>`	インテル® Xeon Phi™ コプロセッサー向けの実行ファイル名のプリフィクスを文字列で指定します。デフォルト値は、空文字です。

説明

ホストの実行ファイル名にプリフィクスを追加し、対応するインテル® Xeon Phi™ コプロセッサーの実行ファイル名にするには、この環境変数を設定します。

例えば、インテル® MIC アーキテクチャーとインテル® 64 アーキテクチャーの実行形式ファイルを区別するため、`I_MPI_MIC_PREFIX` 環境変数に異なる場所を設定します。

```
(host)$ mpiicc test.c -o test_hello
(host)$ mpiicc -mmic test.c -o ./MIC/test_hello (host)$ export I_MPI_MIC=1
(host)$ export I_MPI_MIC_PREFIX=./MIC/
(host)$ mpirun -n 4 -hostfile <hostfile> test_hello
```

この例では、`./test_hello` バイナリーがインテル® 64 アーキテクチャーのノードで起動され、`./MIC/test_hello` バイナリーがインテル® Xeon Phi™ コプロセッサー上で起動されます。

I_MPI_MIC_POSTFIX

構文

```
I_MPI_MIC_POSTFIX=<value>
```

引数

<value>	インテル® Xeon Phi™ コプロセッサー向けの実行ファイル名のポストフィクスを文字列で指定します。デフォルト値は、空文字です。

説明

ホストの実行ファイル名にポストフィクスを追加し、対応するインテル® Xeon Phi™ コプロセッサーの実行ファイル名にするには、この環境変数を設定します。

例えば、インテル® MIC アーキテクチャーとインテル® 64 アーキテクチャーの実行形式ファイルを区別するため、I_MPI_MIC_POSTFIX 環境変数に異なる名前を設定します。

```
(host)$ mpiicc test.c -o test_hello
(host)$ mpiicc -mmic test.c -o test_hello.mic (host)$ export I_MPI_MIC=1
(host)$ export I_MPI_MIC_POSTFIX=.mic
(host)$ mpirun -n 4 -hostfile <hostfile> test_hello
```

この例では、./test_hello バイナリーがインテル® 64 アーキテクチャーのノードで起動され、test_hello.mic バイナリーがインテル® Xeon Phi™ コプロセッサー上で起動されます。

I_MPI_DAPL_PROVIDER_LIST

構文

```
I_MPI_DAPL_PROVIDER_LIST=<primary provider>[,<local secondary provider> [,<remote secondary provider>]]
```

引数

<primary provider>	最良のレイテンシーと利用可能なすべてのネットワーク・セグメントを提供します (ノード間とノード内)。
<local secondary provider>	ローカル設定向けに最良のバンド幅を提供します。距離は、I_MPI_DAPL_LOCALITY_THRESHOLD 環境変数よりも小さな値です。
<remote secondary provider>	リモート設定向けに最良のバンド幅を提供します。距離は、I_MPI_DAPL_LOCALITY_THRESHOLD 環境変数よりも大きな値です。

説明

ロードする DAPL プロバイダーを定義するには、この環境変数を設定します。

インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (Intel® MPSS) で、I_MPI_DAPL_PROVIDER_LIST は次のように設定されます。

- <primary provider>-CCL-direct
- <local secondary provider>-IBSCIF
- <remote secondary provider>-CCL-proxy

I_MPI_DAPL_PROVIDER_LIST=<CCL-direct>[,<IBSCIF>[,<CCL-proxy>]] を設定します。以下は、インテル® MPSS で提供されるデフォルトの dat.conf の例です。

```
I_MPI_DAPL_PROVIDER_LIST=ofa-v2-mlx4_0-1u,ofa-v2-scif0,ofa-v2-mcm-1
```

2 次プロバイダーのしきい値を次のように調整できます。

I_MPI_DAPL_DIRECT_COPY_THRESHOLD 環境変数 (<2 次プロバイダーのしきい値>):
I_MPI_DAPL_DIRECT_COPY_THRESHOLD=<主なプロバイダーの直接コピーしきい値>[,<2 次プロバイダーのしきい値>]

<主なプロバイダーの直接コピーしきい値> は、<2 次プロバイダーのしきい値> よりも小さな値でなければいけません。

環境変数 I_MPI_DAPL_PROVIDER_LIST が値のリストを含んでいる場合、次の環境変数の構文は、対応するプロバイダーに関連する値で拡張されます。

- I_MPI_DAPL_DIRECT_COPY_THRESHOLD
- I_MPI_DAPL_TRANSLATION_CACHE
- I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE
- I_MPI_DAPL_CONN_EVD_SIZE
- I_MPI_DAPL_RDMA_RNDV_WRITE

単一の値を設定すると、すべてのプロバイダーに適用されます。また、一致しないもしくは誤った値を設定すると、すべてのプロバイダーにデフォルト値が適用されます。

次に例を示します。

```
export I_MPI_DAPL_PROVIDER_LIST=ofa-v2-mlx4_0-1,ofa-v2-scif0 export
I_MPI_DAPL_TRANSLATION_CACHE=enable,disable
```

この I_MPI_DAPL_TRANSLATION_CACHE の設定は、最初のプロバイダー向けにメモリー登録キャッシュを有効にしますが、2 次プロバイダーでは有効になりません。

I_MPI_DAPL_LOCALITY_THRESHOLD

ローカルの 2 次プロバイダーからリモートの 2 次プロバイダーへ切り替えるしきい値を定義します。

構文

```
I_MPI_DAPL_LOCALITY_THRESHOLD=<value>
```

引数

<value>	ローカルの 2 次プロバイダーからリモートの 2 次プロバイダーへ切り替えるしきい値を定義します。 オプションの詳細は、「I_MPI_DAPL_PROVIDER_LIST」をご覧ください。
次の説明を参照してください。	デフォルト値は、非均一メモリー・アーキテクチャー (NUMA) の構成と DAPL*/インテル® MPSS のバージョンに依存します。

説明

この値の範囲 [10: 255] は、ホスト内の NUMA と装着されるインテル® Xeon Phi™ コプロセッサーの距離に関連します。デフォルト値はクラスターとジョブに関連して、次のように決定されます。

DAPL* 2.1.3 から開始して、式を使った自動調整するロジックあります: $255 - d_{max} + d_{min}$。次の表は、I_MPI_DAPL_LOCALITY_THRESHOLD のデフォルト値を状況に応じて使用する場合の数式を示します。

デフォルト値を求める式	モード
d + (255 - 1 - d_{max})	同じホストで 2 つのインテル® Xeon Phi™ コプロセッサーでランクが実行される場合。
d + (255 - 1 - d_{max}) / 2	同じホストでインテル® Xeon Phi™ コプロセッサーで 1 つのランクが実行される場合。
d	両方のランクは同じホストで実行されます。
255	両方のランクは異なるホストで実行されます。

上記の式で、

- d は、ホストシステムの NUMA の距離を示します。
- d_{min} は、ホストシステムの NUMA の最小距離を示します。
- d_{max} は、ホストシステムの NUMA の最大距離を示します。

DAPL 2.1.3 以前に、I_MPI_DAPL_LOCALITY_THRESHOLD は、ノード内で実行されるすべてのランクが選択され、ローカルの 2 次プロバイダーに準拠し、デフォルトで 255 と等価です。そうでない場合、リモートの 2 次プロバイダーが選択されます。

I_MPI_ENV_PREFIX_LIST

特定のプラットフォーム向けの環境変数のプリフィクスを定義します。

構文

```
I_MPI_ENV_PREFIX_LIST=[platform:prefix][,...]
```

引数

platform	特定のプラットフォーム (文字列)。 オプション: htn,nhm,wsm,snb,ivb オプションの詳細は、「I_MPI_PLATFORM」をご覧ください。
prefix	特定のプラットフォーム向けに使用される環境変数名のプリフィクス (文字列)。

説明

特定のプラットフォーム向けに使用される環境変数名のプリフィクスを定義するには、この環境変数を設定します。

環境変数のプリフィクスを I_MPI_ENV_PREFIX_LIST で指定する場合、プリフィクスの対象となる環境変数は、特定のプラットフォーム上でそれぞれの非プリフィクス環境変数を上書きします。

I_MPI_ENV_PREFIX_LIST を指定しない場合、環境変数はすべてのプラットフォームに適用されます。

注意

プラットフォーム名を指定する場合、小文字を使用します。

例

1. `I_MPI_ENV_PREFIX_LIST=platform:prefix`

   `<NAME>=valueis` はすべてのシステムに適用されます。

   `<prefix>_<NAME>=value` は、すべての `<platform>` システム向けに、`<NAME>=value` を定義します。

2. 何台かのマシンが、インテル® マイクロアーキテクチャー (開発コード名 Sandy Brigge) ベースのプラットフォームで、残りのマシンはその他のアーキテクチャー・ベースのプラットフォームであると仮定します。環境変数 `OMP_NUM_THREADS` の値は、すべてのプラットフォーム上で 3 です。

   インテル® マイクロアーキテクチャー (開発コード名 Sandy Brigge) ベースのプラットフォーム上のランク向けに `OMP_NUM_THREADS=5` に設定するには、次のように `OMP_NUM_THREADS` 向けに `I_MPI_ENV_PREFIX_LIST` でプリフィクスを指定します。

   ```
   I_MPI_ENV_PREFIX_LIST=snb:<prefix>
   OMP_NUM_THREADS=3
   <prefix>_OMP_NUM_THREADS=5
   ```

2.5.3. コンパイラーのコマンド

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。

コンパイラーのコマンド	デフォルト・コンパイラー	サポートされる言語	サポートされる ABI
`mpiicc`	`icc`	C	64 ビット
`mpiicpc`	`icpc`	C++	64 ビット
`mpiifort`	`ifort`	Fortran77/Fortran 95	64 ビット

コンパイラー・コマンドは次のような共通の機能を持ちます。

- コンパイラー・コマンドは、`<installdir>/intel64em64t/bin` ディレクトリーに配置されます。
- 環境の設定は、`<installdir>/intel64em64t/bin/mpivars.sh` スクリプトを実行することで行うことができます。異なるライブラリー向けの環境設定が必要な場合、対応する環境に切り替えるため、次の引数を `mpivars.sh` スクリプトに渡すことができます。
  - `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。

- 異種 MPI アプリケーションをコンパイルするには、インテル® 64 アーキテクチャー向けとインテル® MIC アーキテクチャー向けに、2 回コンパイルを行います。
- ターゲット・アーキテクチャーを区別するため、スクリプトはコンパイラー・オプションを解析します。ターゲットをインテル® MIC アーキテクチャーとするオプションを (`-mmic` など) 検出すると、インテル® Parallel Studio XE 2016 for Linux* (インテル® MIC アーキテクチャー向け) のコンパイラーを使用して、インテル® MIC アーキテクチャー向けの実行形式ファイルが作成されます。そうでない場合、インテル® Xeon® プロセッサー向けの実行形式ファイルが生成されます。

- `-cc/-cxx/-fc/-f77/-f90` オプションか、リファレンス・マニュアルに記載される環境変数によって GNU* コンパイラーによるコンパイルを要求できます。

  次に例を示します。

  ```
  (host)$ mpicc -cc=/usr/linux-k1om-4.7/bin/x86_64-k1om-linux-gcc -mmic test.c
  -o test_hello.mic
  ```

注意

インテル® MIC アーキテクチャーとインテル® 64 アーキテクチャーの実行形式ファイルを区別するため、異なる格納場所やファイル名を使用してください。

2.6. 多目的デーモン (MPD) のコマンド

mpd

多目的デーモン (MPD)。

構文

```
mpd[ --help] [ -V ] [ --version ] [ --host=<host> --port=<portnum> ] \
[ --noconsole ] [ --trace ] [ --echo] [ --daemon ] [ --bulletproof ]\
[ --i fhn <interface/hostname>] [ --listenport <listenport> ]
```

引数

`-help`	ヘルプメッセージを表示します。
`-V | --version`	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。
`-h <host> -p <portnum> | --host=<host> -- port= <portnum>`	既存のリングに加わるためのホストとポート指定します。 `--host` と `–port` オプションは、どちらも指定する必要があります。
`-n | --noconsole`	開始時にコンソールを開きません。
`-t | --trace`	内部 MPD トレース情報を出力します。
`-e | --echo`	ほかの `mspd` 接続の開始時にポート番号を表示します。
`-d | --daemon`	デーモンモードで `mpd` を起動します。デフォルトでインタラクティブ・モードが有効です。
`--bulletproof`	MPD の `bulletproofing` を有効にします。
`--ifhn=<interface/ hostname>`	MPD 通信に使用する `<interface/hostname>` を指定します。
`-l <listenport> | --listenport= <listenport>`	`mpd` の監視ポートを指定します。

説明

多目的デーモン (MPD: 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 mpd デーモンを開始しリングに接続します。長いパラメーター名は、先頭文字にハイフン (等号なし) 記号を使用して省略することができます。

例:

```
$ mpd -h masterhost -p 4268 -n
```

上記のコードは次のコードと等価です。

```
$ mpd --host=masterhost --port=4268 -noconsole
```

ホーム・ディレクトリーに .mpd.conf という名前のファイルが存在する場合、ユーザーだけが読み書きできます。ファイルは、secretword=<secretword> を含んでいる必要があります。MPD を root 権限で実行する場合、mpd を root 権限で実行するアカウントのホーム・ディレクトリーの .mpd.conf ファイルの代わりに、/etc ディレクトリーに mod.conf ファイルを作成します。root アカウントでの MPD リングの開始を避けてください。

注意

多目的デーモン (MPD) は、インテル® MPI ライブラリー 5.0 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。

mpdboot

mpd リングを開始します。

構文

```
mpdboot[ -h] [ -V ] [ -n <#nodes> ] [ -f <hostsfile> ] [ -r <rshcmd> ] \
[ -u <user> ] [ -m<mpdcmd>] [ --loccons ] [ --remcons ] \
[ -s ] [ -d ] [ -v] [ -1 ] [ --ncpus=<ncpus> ] [ -o ] \  [ -b <maxbranch> ] [ -p ]
```

または

```
mpdboot[ --help] [ --version ] [ --totalnum=<#nodes> ] \
[ --file=<hostsfile> ] [ --rsh=<rshcmd> ] [ --user=<user> ] \
[ --mpd=<mpdcmd> ] [ --loccons ] [ --remcons ] [ --shell ] \
[ --debug ] [ --verbose] [ -1 ] [ --ncpus=<ncpus> ] [ --ordered ]
[ --maxbranch=<maxbranch> ] [ --parallel-startup ]
```

引数

-h | --help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します。
-d | --debug	デバッグ情報を表示します。
-v | --verbose	詳しい情報を表示します。<rshcmd> 試行を表示します。
-n <#nodes> | --totalnum=<#nodes>	デーモンが開始される mpd.hosts 内のノード数。
-r <rshcmd> | --rsh=<rshcmd>	デーモンとジョブを開始するリモートシェルを指定します。デフォルト値は ssh です。

-f <hostsfile> | --file=<hostsfile>	デーモンが起動しているマイン名のリストを格納するファイスのパスと名前。
-1	マシンごとに複数の mpd の起動を有効にします。
-m<mpdcmd> | --mpd=<mpdcms>	リモートホスト上の mpd のフルパス名を指定します。
-s | --shell	シェルを指定します。
-u <user> | -- user=<user>	ユーザーを指定します。
--loccons	ローカル MPD コンソールを開きません。
--remcons	リモート MPD コンソールを開きません。
--ncpus=<ncpus>	ローカルマシン上で使用するプロセス数を指定します (ほかのノードは hosts ファイルで示されます)。
-o | --ordered	mpd デーモンを mpd.hosts ファイルで指定されている順番で開始します。
-b <maxbranch> | --maxbranch=<maxbranch>	ほかの mpd リングに参加する mpd デーモンの最大数を指定するには、このオプションを使用します。これは、mpd デーモンの並列処理を制御するのに役立ちます。デフォルト値は 4 です。
-p |--parallel-startup	このオプションは、単独のローカル root における mpd デーモンの並列高速起動を可能にします。デーモンのチェックを行いません。このオプションはまた、リモートコマンドからの出力を転送しないシェルをサポートします。

説明

<mpd.hosts> にノード名のリストを記述することで、指定した数のノード上で mpd デーモンを起動します。

mpd デーモンは、デフォルトで ssh コマンドを使用して起動されます。ssh 接続が確立されていない場合、-r rsh オプションを使用して rsh に切り替えます。クラスター上のすべてのノードが、パスワードなしの ssh コマンドによる接続が互いに可能であるか、-r rsh オプションを使用してパスワードなしの rsh コマンドを使用します。

注意

mpdboot コマンドは、mpd.hosts ファイルにマシン名が記述されていない場合でも、ホストマシン上で MPD デーモンを起動します。

mpdexit

単一の mpd デーモンをシャットダウンします。

構文

```
mpdexit [ --help][ -V ] [--version ] <mpdid>
```

引数

-help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します
<mpdid>	終了する mpd デーモンを指定します。

説明

このコマンドは、シグナルを送信して mpd デーモンを終了させます。mpdtrace -l コマンドで、<hostname>_<port number> の形式で取得した <mpdid> を使用します。

mpdallexit

すべてのノード上のすべての mpd デーモンをシャットダウンします。

構文

```
mpdallexit [ --help] [ -V ] [ --version ]
```

引数

-help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します。

説明

すべての MPD リングをシャットダウンします。

mpdcleanup

mpd がクラッシュした場合に、環境をクリーンアップします。

構文

```
mpdcleanup [ -h] [ -V ] [ -f <hostsfile> ] [ -r <rshcmd> ] [ -u <user> ]\
[ -c <cleancmd> ] [ -a]
```

または

```
mpdcleanup [ --help] [ --version ] [ --file=<hostsfile> ] \
[ --rsh=<rshcmd> ] [ --user=<user> ] [ --clean=<cleancmd> ] \
[ --all]
```

引数

-h| --help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します
-f <hostsfile> | --file=<hostsfile>	クリーンアップするマシンのリストを含むファイルを指定します。
-r <rshcmd> | --rsh=<rshcmd>	使用するリモートシェルを指定します。
-u <user> | --user=<user>	ユーザーを指定します。

-c <cleancmd> | --clean=<cleancmd>	UNIX* ソケットの削除に使用するコマンドを指定します。デフォルトのコマンドは、/bin/rm -f です。
-a | --all	<hostsfile> に指定されるすべてのホスト上で、I_MPI_JOB_CONTEXT 環境変数の設定に関連するすべての mpd デーモンを終了します。

説明

mpd がクラッシュした場合に、環境をクリーンアップするには、このコマンドを使用します。これは、ローカルマシンとリモートマシン上の UNIX* ソケットを削除するか、I_MPI_JOB_CONTEXT 環境変数の設定に関連するすべての mpd デーモンを終了します。

例えば、次のコマンドは hostsfile ファイルに指定されるマシン上の UNIX* ソケットを削除します。

```
$ mpdcleanup --file=hostsfile
```

hostsfile ファイルで指定されているマシン上の mpd デーモンを kill するには、次のコマンドを使用します。

```
$ mpdcleanup --file=hostsfile --all
```

mpdtrace

mpd が起動しているか確認します。

構文

```
mpdallexit [ --help] [ -V ] [ --version ]
```

引数

-help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します。
-l	ホスト名の代わりに MPD 識別子を表示します。

説明

リング中のすべての mpd のホスト名もしくは識別子をリストするには、このコマンドを使用します。出力される識別子は、<hostname>_<port number> の形式です。

mpdcheck

ホスト上の設定の問題をチェックして、設定情報を表示します。

構文

```
mpdcheck [ -v] [ -l] [ -h ] [ --help ] [ -V ] [ --version ] mpdcheck -pc [ -v ] [ -l]
mpdcheck -f <host_file> [ -ssh ] [ -v ] [ -l] mpdcheck -s [ -v ] [ -l]
mpdcheck -c < server_host> <server_port> [ -v ] [ -l]
```

引数

-h| --help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します。
-pc	ローカルホストの設定情報を表示します。

-f <host_file>	<host_file> にリストされるホストの情報を表示します。
-ssh	それぞれのリモートホストで ssh の起動をテストします。-f オプションと同時に使用します。
-s	1 つのホスト上でサーバーとして mpdcheck を実行します。
-c <server_host> <server_port>	現在のホストまたは異なるホスト上でクライアントとして mpdcheck を実行します。<server_host> <server_port> へ接続します。
-l	拡張形式で診断メッセージを出力します。
-v	mpdcheck が行ったアクションを出力します。

説明

クラスターノード上の設定に関する問題をチェックするには、このコマンドを使用します。出力の *** で始まる行は、何らかの問題を明示します。

1 つ以上のホストで mpd を介して並列ジョブの実行に問題が生じた場合、それぞれのノードで一度だけスクリプトを実行してみてください。

mpdringtest

MPD リングをテストします。

構文

```
mpdringtest [ --help ] [ -V ] [ --version ] <number of loops>
```

引数

-help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します。
<number of loops>	ループの回数。

説明

このコマンドは、メッセージが mpd リングを一週するのにかかる時間をテストします。

mpdlistjobs

構文

```
mpdlistjobs [ -h] [ -V ] [-u <username> ] [ -a <jobalias> ] [ -j <jobid> ]
```

または

```
mpdlistjobs [ --help] [ --version ] [ --user=<username> ] \
[ --alias=<jobalias> ] [ --jobid=<jobid> ]
```

引数

-h | --help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します。

-u <username>| --user=<username>	特定のユーザーのジョブをリストします。
-a <jobalias> | -- alias=<jobalias>	<jobalias> で指定された特定のジョブの情報をリストします。
-j <jobid> | --jobid=<jobid>	<jobid> で指定された特定のジョブの情報をリストします。

説明

MPI ジョブを実行する実行中のプロセスをリストするには、このコマンドを使用します。現在のマシン上のすべてのジョブがデフォルトで表示されます。

mpdsigjob

アプリケーションを実行するプロセスへシグナルを送ります。

構文

```
mpdsigjob [ --help] [ -V ] [ --version ] <sigtype> \
[-j <jobid> | -a <jobalias> ] [-s | -g ]
```

引数

-help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します。
<sigtype>	送信するシグナルのタイプを指定します。有効なオプションは -j もしくは -a です。
-a <jobalias>	<jobalias> で指定されたジョブへシグナルを送信します。
-j <jobid>	<jobid> で指定されたジョブへシグナルを送信します。
-s	特定のユーザーのプロセスへシグナルを送信します。
-g	特定のグループのプロセスへシグナルを送信します。これは、デフォルトの動作です。

説明

ジョブを実行するプロセスへシグナルを送信するには、このコマンドを使用します。必要なシグナルは最初の引数です。1 つまたは 2 つのオプション (-j または -a) を指定します。

mpdkilljob

ジョブを強制終了します。

構文

```
mpdkilljob [ --help ] [ -V ] [ --version ] [ <jobnum> ] [ -a <jobalias> ]
```

引数

-help	ヘルプメッセージを表示します。
-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します。
<jobnum>	<jobnum> で指定されるジョブを Kill します。
-a <jobalias>	<jobalias> で指定されるジョブを Kill します。

説明

<jobnum> または <jobalias> で指定されたジョブを kill するには、このコマンドを使用します。

mpdlistjobs コマンドから <jobnum> と <jobalias> を取得します。<jobid> フィールドは、次の形式です: <jobnum>@<mpdid>。

mpdhelp

MPD コマンドに関する簡単なヘルプを表示します。

構文

```
mpdhelp [ -V ] [ --version ]
```

引数

-V | --version	インテル® MPI ライブラリーのバージョン情報を表示します。

説明

MPD コマンドに関する簡単なヘルプ・メッセージを表示するには、このコマンドを使用します。

2.6.1. ジョブ開始コマンド

mpiexec

構文

```
mpiexec <g-options> <l-options> <executable>
```

または

```
mpiexec <g-options> <l-options> <executable1> : \
<l-options> <executable2>
```

または

```
mpiexec.smpd -configfile <file>
```

引数

<g-options>	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-options>	単一の引数セットに適用するローカルオプション。
<executable>	./a.out または path/実行形式ファイル名。
<file>	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 <executable> 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 <# of processes> の a.out を実行します。

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

2 番目のコマンドラインの構文は、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 -n 2 -host host1 ./a.out : \
-n 2 -host host2 ./b.out
```

3 番目のコマンドライン構文では、コマンドラインを指定された <file> 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 <file> 内の単一行に指定される必要があります。

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 <file> 内の単一行に指定される必要があります。グローバルオプションは、常に <file> の先頭行になければいけません。

mpiexec が成功するには、事前に MPD デーモンが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上で PATH 環境変数に「.」が設定されていない場合、a.out の代わりに ./a.out を指定してください。

拡張デバイス制御オプション

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。

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。

単一のノード上ですべてのプロセスが開始されると、インテル® MPI ライブラリーはオプションに関係なく 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。詳細は、「I_MPI_FABRICS」と「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。

-rdma

ノード間通信に 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 もしくは 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 または tmi 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 dapl,ofa,tcp,tmi -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-RDMA

ノード間通信に 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 もしくは 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

ノード間通信に 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、tmi または ofa 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 dapl,ofa -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-DAPL

ノード間通信に 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

ノード間通信に 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 または tmi 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,tmi -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-IB

ノード間通信に 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。

-tmi

ノード間通信に TMI 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、TMI 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 または ofa 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 tmi,dapl,tcp,ofa -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-TMI

ノード間通信に TMI 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 tmi -genv I_MPI_FALLBACK 0 オプションを指定するのと等価です。

-mx

ノード間通信に Myrinet MX* 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、Myrinet MX* 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 または ofa 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 tmi,dapl,tcp,ofa -genv I_MPI_TMI_PROVIDER mx -genv I_MPI_DAPL_PROVIDER mx -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-MX

ノード間通信に Myrinet MX* 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 tmi -genv I_MPI_TMI_PROVIDER mx - genv I_MPI_FALLBACK 0 オプションを指定するのと等価です。

-psm

ノード間通信にインテル® True Scale 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、インテル® True Scale 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 または ofa 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 tmi,dapl,tcp,ofa -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 1 オプションを指定するのと等価です。

-PSM

ノード間通信にインテル® True Scale 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 tmi -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 0 オプションを指定するのと等価です。

グローバルオプション

-version または -V

インテル® MPI ライブラリーのバージョン情報を表示します。

-h、-help または --help

mpiexec のヘルプ・メッセージを表示します。

-tune [<arg >]

ここで以下を指定します。

<arg>= {<dir_name>, <configuration_file>}

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。

<arg> 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位置は、<installdir>/<arch>/etc ディレクトリーです。次のように指定することで、このデフォルトの位置を変更できます: <arg>=<dir_name>。<arg>=<configuration_file> を設定した場合に限り、提供される設定が適用されます。

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。

注意

<arg> が設定ファイルの位置を指していない場合、I_MPI_FABRICS 環境変数を設定します。

I_MPI_FABRICS が設定されていない場合、パフォーマンス・データは提供されず警告が表示されます。

-nolocal

mpiexec が起動されたホスト上で <executable> の実行を避けるには、このオプションを使用します。MPI ジョブを開始する専用のマスターノードと、実際の MPI プロセスと実行する専用の計算ノードを配備するクラスター上でこのオプションは有用です。

-perhost <# of processes>

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。開始時の総プロセス数は、-n オプションで制御されます。

mpiexec コマンドは、プロセスのランクがクラスターのノードにどのように割り当てられるかを制御します。デフォルトで、mpiexec はノードへラウンドロビン方式でランクを配置し、すべてのプロセッサーコア上で連続した MPI プロセスを実行します。

このデフォルトの動作を変更するには、-perhost オプションを使用してホストごとのプロセス数を設定し、-n オプションで総プロセス数を設定します。詳細は、「ローカルオプション」をご覧ください。-perhost オプションで指定された最初の <#of processes> は、最初のホストで実行され、次の <#of processes> は 2 番目のホストで実行されます。

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。

-grr <# of processes>

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。このオプションは、-perhost <#of processes> と等価です。

-ppn <# of processes>

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。このオプションは、-perhost <#of processes> と等価です。

-machinefile <machine file>

このオプションは、<machine file> 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: <host name>:<number of processes>。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。

```
<host name>:<number of processes>[ifhn=<interface_host_name>]
```

注意

-machinefile、-ppn、-rr、および –perhost オプションは、プロセスの分散を目的とします。同時に使用した場合、-machinefile が優先されます。

-configfile <filename>

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 <filename> 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.out と b.out を shm: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.out
-host host2 -env I_MPI_DEBUG 2 -env I_MPI_FABRICS shm:dapl -n 2 ./b.out
```

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 -configfile <filename>
```

注意

このオプションは、mpiexec 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-option>

ローカルオプション <l-option> 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

-genv <ENVVAR> <value>

すべての MPI プロセス向けに、<ENVVAR> に指定された <value> を設定します。

-genvuser

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: $HOSTNAME、$HOST、$HOSTTYPE、$MACHTYPE、$OSTYPE。これは、デフォルトの設定です。

-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。

(SDK のみ) -trace [<profiling_library>] または -t [<profiling_library>]

指定された <profiling_library> を使用して MPI アプリケーションのプロファイルを行うには、このオプションを指定します。

<profiling_library> が省略された場合、デフォルトのプロファイル・ライブラリーは libVT.so です。

デフォルトのプロファイル・ライブラリーを変更するには、I_MPI_JOB_TRACE_LIBS 環境変数を設定します。

注意

実行前にアプリケーションをプロファイル・ライブラリーとリンクする必要はありません。

(SDK のみ) -check_mpi [<checking_library>]

指定された <checking_library> を使用して MPI アプリケーションをチェックするには、このオプションを指定します。

<checking_library> が省略された場合、デフォルトのチェックライブラリーは libVTmc.so です。

デフォルトのチェック・ライブラリーを変更するには、I_MPI_JOB_CHECK_LIBS 環境変数を設定します。

注意

実行前にアプリケーションをチェックライブラリーとリンクする必要はありません。

-tv

TotalView* デバッガー環境下で <executable> を実行するには、このオプションを使用します。次に例を示します。

```
$ mpiexec -tv -n <# of processes> <executable>
```

TotalView* 実行形式ファイルを選択する方法は、「環境変数」をご覧ください。

注意

TotalView* がデフォルトで rsh を使用するため、環境変数 TVDSVRLAUNCHCMD=ssh に設定されていることを確認してください。

注意

TotalView* デバッガーは、MPI プログラムのメッセージキューの状態を表示できます。この機能を有効にするには、次の手順に従ってください。

1. <executable> を -tv オプションで実行します。

   ```
   $ mpiexec -tv -n <# of processes> <executable>
   ```

2. Python* ジョブの停止に関する問い合わせには、「**YES**」を選択します。

MPI ライブラリーの内部状態をテキストで表示するには、**[Tools] > [Message Queue]** コマンドを選択します。**[Process Window Tools] > [Message Queue Graph]** コマンドを選択すると、TotalView* は現在のメッセージキューの状態をグラフ形式でウィンドウに表示します。詳細は、「TOTALVIEW」をご覧ください。

-tva <jobid>

TotalView* デバッガーに実行中の <jobid> をアタッチするには、このオプションを使用します。次に例を示します。

```
$ mpiexec -tva <jobid>
```

-tvsu

TotalView* デバッガー環境下で <executable> を実行するには、このオプションを使用します。次に例を示します。

```
$ mpiexec -tvsu -n<# of processes> <executable>
```

注意

実行中のインテル® MPI ジョブをデバッグするには、mpiexec スクリプトを実行する Python* のインスタンスに TotalView* をアタッチします。

-gdb

GNU* デバッガー環境下で <executable> を実行するには、このオプションを使用します。次に例を示します。

```
$ mpiexec -gdb -n <# of processes> <executable>
```

-gdba <jobid>

GNU* デバッガーに実行中の <jobid> をアタッチするには、このオプションを使用します。次に例を示します。

```
$ mpiexec -gdba <jobid>
```

-a <alias>

ジョブに <alias> を割り当てます。

-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

このオプションを適切に動作させるには、各プロセスが出力する最後の行は \n 文字で終了する必要があります。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。

-m

出力情報の行をマージするにはこのオプションを使用します。

-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。

-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.hydra 設定ファイルの処理を無効にします。

-ifhn <interface/hostname>

ローカル MPD デーモンとの通信のネットワーク・インターフェイスを指定するには、このオプションを使用します。ここで、<interface/hostname> は、代替ネットワーク・インターフェイスに関連付けられた IP アドレスもしくはホスト名です。

-ecfn <filename>

このオプションは、<filename> に XML exit コードのリストを出力します。

ローカルオプション

-n <# of processes> または -np <# of processes>

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 <ENVVAR> <value>

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

-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: $HOSTNAME、$HOST、$HOSTTYPE、$MACHTYPE、$OSTYPE。これは、デフォルトの設定です。

-envall

現在の引数セットですべての環境変数を伝搬します。

-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 <list of env var names>

引数リストと現在の値を渡します。<list of env var names> 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。このオプションがコマンドラインで複数回使用された場合、引数で指定されるすべての変数は 1 つのリストにまとめられます。

-envexcl <list of env var names>

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

-host <nodename>

現在の引数セットで MPI プロセスを実行する特定の <nodename> 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.out を実行します。

```
$ mpiexec -n 2 -host host1 ./a.out
```

-path <directory>

実行する <executable> へのパスを指定します。

-wdir <directory>

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

-umask <umask>

リモートプロセス向け umask <umask> コマンドを実行するには、このオプションを使用します。

設定ファイル

mpiexec 設定ファイルでは、すべての mpiexec コマンドに適用するデフォルトのオプションを指定します。

これらのファイルのいずれかが存在する場合、その内容は次の順番で mpiexec コマンドラインのオプションの先頭に追加されます。

システム全体 <installdir>/etc/mpiexec.conf 設定ファイルのデフォルトの位置は、<installdir>/<arch>/etc ディレクトリーです。

ユーザー固有: $HOME/.mpiexec.conf

セッション固有: $PWD/mpiexec.conf

環境変数を定義するかコマンドライン・オプションを指定してこれらのファイルを上書きできます。mpiexec –noconf オプションで、これらの設定ファイルの使用をスキップできます。

これらのファイルを作成または変更できます。それらは、mpiexec コマンドラインのオプションを含みます。空白行と先頭文字が '#' の行は無視されます。例えば、デフォルトのファブリックを指定するには、mpiexec.con ファイルに次の行を追加します。

```
-genv I_MPI_FABRICS <fabric>
```

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=<level>[,<flags>]

引数

<level>	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 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。

<flags>	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ド 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 <level> 値を設定します。

I_MPI_DEBUG_OUTPUT 環境変数にデバッグ情報の出力ファイル名を指定できます。

出力は次の形式です。

```
[<identifier>] <message>
```

ここで、

- <identifier> は、メッセージを生成する MPI プロセスを識別します。<level> が符号なしの番号であれば、<identifier> は MPI プロセスのランクです。'+' 記号を <level> 番号の前に追加すると、<identifier> は rank#pid@hostname タプルを含みます。ここで、rank は MPI プロセスのランクで、pid は UNIX* プロセスの id で、hostname はプロセス起動時に定義されたホスト名です。
- <message> は、デバッグ出力を含みます。

次に例を示します。

```
$ mpiexec -n 1 -env I_MPI_DEBUG 2 ./a.out
```

出力は次のようになります。

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

コマンドは次のようになります。

```
$ mpiexec -n 1 -env I_MPI_DEBUG +2 ./a.out
```

または

```
$ mpiexec -n 1 -env I_MPI_DEBUG 2,pid,host ./a.out
```

出力は次のようになります。

```
[0#1986@mpicluster001] MPI startup(): shared memory data transfer mode
```

注意

mpicc -g オプションでコンパイルすると、かなりの量のデバッグ情報が出力されます。

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

<arg>	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。
<file_name>	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_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=<value>`

引数

`<value>`	デフォルトのプロセス配置を定義します。
`<n> > 0`	ノードごとに <n> プロセス。
`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)

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。コマンドラインで `-perhost` オプションが指定されると、`I_MPI_PERHOST` 環境変数は無視されます。この環境変数が定義されていない場合、`-perhost` オプションは `I_MPI_PERHOST` 環境変数に値を反映します。

注意

`I_MPI_PERHOST` が定義され、`mpiexec -host` オプションが指定されると、`I_MPI_PERHOST` は無視されます。

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

`<arg>`	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) にするには環境変数を設定します。

(SDK のみ) I_MPI_JOB_TRACE_LIBS (MPIEXEC_TRACE_LIBS)

`-trace` オプションを介して事前ロードするライブラリーを選択します。

構文

`I_MPI_JOB_TRACE_LIBS=<arg>`

廃止された構文

`MPIEXEC_TRACE_LIBS=<arg>`

引数

<arg>	文字列パラメーター。
<list>	スペース (空白で) 区切られた、事前ロードするライブラリー。デフォルト値は vt です。

説明

-trace オプションを介して事前ロードする代替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。

(SDK のみ) I_MPI_JOB_CHECK_LIBS

-check_mpi オプションを介して事前ロードするライブラリーを選択します。

構文

I_MPI_JOB_CHECK_LIBS=<arg>

引数

<arg>	文字列パラメーター。
<list>	スペース (空白で) 区切られた、事前ロードするライブラリー。デフォルト値は vtmc です。

説明

-check_mpi オプションを介して事前ロードする代替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。

I_MPI_JOB_STARTUP_TIMEOUT

mpiexec のジョブ開始のタイムアウト時間を設定します。

構文

I_MPI_JOB_STARTUP_TIMEOUT=<timeout>

引数

<timeout>	mpiexec のタイムアウト時間を秒単位で指定します。
<n> >= 0	デフォルトのタイムアウト値は、20 秒です。

説明

この環境変数は、mpiexec がジョブの起動後 <timeout> 秒でジョブを強制終了する時間を設定します。<timeout> 値は、ゼロよりも大きくなければいけません。そうでない場合、環境変数の設定は無視され、警告が発せられます。ノード数の多い大規模なクラスター上でジョブの起動時間がデフォルトを越えるような場合、この環境変数を設定します。

注意

mpiexec コマンドを実行する前に、シェル環境で I_MPI_JOB_STARTUP_TIMEOUT 環境変数を設定します。<timeout> 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 (MPIEXEC_TIMEOUT)

mpiexec のタイムアウト時間を設定します。

構文

I_MPI_JOB_TIMEOUT=<timeout>

廃止された構文

MPIEXEC_TIMEOUT=<timeout>

引数

<timeout>	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 <timeout> 秒でジョブを強制終了する時間を設定します。<timeout> 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。<timeout> 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=<number>

廃止された構文

MPIEXEC_TIMEOUT_SIGNAL=<number>

引数

<number>	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

環境変数 I_MPI_JOB_TIMEOUT で指定されるタイムアウト時間に応じて、タスクの終了に使用するシグナルを定義します。システムがサポートし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=<number>

引数

<number>	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=<arg>

廃止された構文

MPIEXEC_SIGNAL_PROPAGATION=<arg>

引数

<arg>	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、MPD デーモンが受信するシグナル (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_OUTPUT_CHUNK_SIZE

stdout/stderr 出力バッファーのサイズを設定します。

構文

I_MPI_OUTPUT_CHUNK_SIZE=<size>

引数

<size>	K バイト単位で出力チャンクのサイズを定義します。
<n> > 0	デフォルトのチャンクサイズは、1KB です。

説明

プロセスからの標準出力と標準エラー出力を受け取るためのバッファーサイズを大きくするには、この環境変数を設定します。<size> がゼロ以下の場合、環境変数の値は無視され警告が発せられます。

異なるプロセスから大量の出力を行うアプリケーションでは、この設定を使用します。mpiexec に -ordered-output オプションを追加すると、文字化けを防止するのに役立ちます。

注意

mpiexec コマンドを実行する前に、シェル環境で I_MPI_OUTPUT_CHUNK_SIZE 環境変数を設定します。<size> 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_PMI_EXTENSIONS

インテル® MPI ライブラリーのプロセス管理インターフェイス (PMI) 拡張の使用を on/off にします。

構文

I_MPI_PMI_EXTENSIONS=<arg>

引数

<arg>	バイナリー・インジケーター。
enable | yes | on | 1	PMI 拡張を有効にします。
disable | no | off | 0	PMI 拡張を無効にします。

説明

インテル® MPI ライブラリーは、プロセス管理が PMI 拡張をサポートするかどうかを自動認識します。サポートされている場合、PMI 拡張によりタスクの起動時間が短縮されます。プロセス管理がこの拡張をサポートしていない場合、環境変数 I_MPI_PMI_EXTENSIONS を disable に設定します。

I_MPI_PMI_LIBRARY

サードパーティーによる実装の PMI ライブラリーの名称を指定します。

構文

I_MPI_PMI_LIBRARY=<name>

引数

<name>	サードパーティ PMI ライブラリーへのフルパス名。

説明

環境変数 I_MPI_PMI_LIBRARY に、サードパーティー PMI ライブラリー名を設定します。この環境変数には、フルパスでライブラリー名を指定してください。

I_MPI_JOB_FAST_STARTUP (I_MPI_PMI_FAST_STARTUP)

インテル® MPI ライブラリーの高速プロセス起動アルゴリズムを on/off にします。

構文

I_MPI_JOB_FAST_STARTUP=<arg>

廃止された構文

I_MPI_PMI_FAST_STARTUP=<arg>

引数

<arg>	バイナリー・インジケーター。
enable | yes | on | 1	高速スタートアップのアルゴリズムを有効にします。これは、デフォルト値です。
disable | no | off | 0	高速スタートアップのアルゴリズムを無効にします。

説明

新しいアルゴリズムは、アプリケーションの起動時間を大幅に短縮します。一部の DAPL プロバイダーは、大量のプロセスの起動時 (512 プロセスを超える場合) に過負荷状態になります。この問題を避けるため、I_MPI_JOB_FAST_STARTUP 環境変数を disable に設定してこのアルゴリズムを無効にします。

TOTALVIEW

使用する特定の TotalView* 実行ファイルを選択します。

構文

TOTALVIEW=<path>

引数

<path>	デフォルトの TotalView* に代わり、TotalView 実行ファイルへのパス/名前を指定します。

説明

特定の TotalView* 実行ファイルを選択するため、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=<platform>

引数

<platform>	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。選択した結果とは異なりノード間となる場合、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。
knl	インテル® Xeon Phi™ プロセッサー x200 製品ファミリー (コード名: Knights Landing) 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

<arg>	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=<threadlevel>

引数

<threadlevel>	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

初期化のために MPI_Init() を使用している場合に、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。

注意

I_MPI_THREAD_LEVEL_DEFAULT 環境変数は、MPICH_THREADLEVEL_DEFAULT 環境変数と等価です。

2.6.2. 設定ファイル

$HOME/.mpd.conf

このオプションの設定ファイルは、mpd デーモンのパスワードを含んでいます。mpd デーモンを起動する前に作成してください。これにより、さまざまなインテル® MPI ライブラリーのユーザーからデーモンを制御できるようになります。

構文

このファイルは 1 行で次のいずれかの形式です。

secretword=<mpd password>

または

MPD_SECRETWORD=<mpd password>

説明

任意の <mpd password> 文字列を使用して、さまざまなクラスターユーザーによる MPD デーモンへのアクセスを制御します。ここでは、Linux* のログインパスワードを使用しないでください。

$HOME/.mpd.conf ファイルをマウントされたネットワーク・ファイル・システムに配置するか、クラスターのすべてのノード上で $HOME/.mpd.conf としてアクセスできるようにします。

mpdboot が root 権限を持たない <user> によって実行される場合、このファイルは対応するユーザー <user> と <<user>'s group> の所有権を持っている必要があります。

アクセス権限は 600 (ユーザーの読み書き可能) に設定される必要があります。

注意

MPD_SECRETWORD は、secretword と同義です。

mpd.hosts

このファイルには、mpdboot コマンドが mpd デーモンを起動する際に使用するノード名のリストを記述します。このファイルは、mpdboot コマンドを実際に起動するノード上で、mpdboot を実行するユーザーがアクセス可能であることを確認してください。

構文

mpd.hosts ファイルの形式は、ノード名のリストを 1 行に 1 つ定義します。空白行と # に続く行は無視されます。

2.6.3. 環境変数

I_MPI_JOB_CONFIG_FILE (I_MPI_MPD_CONF)

mpd 設定ファイルのパス/名前を設定します。

構文

I_MPI_JOB_CONFIG_FILE=<path/name>

廃止された構文

I_MPI_MPD_CONF=<path/name>

引数

<path/name>	MPD 設定ファイルへの絶対パス。

説明

この環境変数には、デフォルトの ${HOME}/.mpd.conf に代わって、mpdboot スクリプトが使用するファイルのフルパスを定義します。

I_MPI_JOB_CONTEXT (MPD_CON_EXT)

mpd コンソールファイルの一意な名前を設定します。これにより、同じユーザーアカウントで、複数の mpd リングを実行することが可能になります。

構文

I_MPI_JOB_CONTEXT=<tag>

廃止された構文

`MPD_CON_EXT=<tag>`

引数

`<tag>`	一意な MPD 識別子。

説明

複数の `mpd` リングが共存できるよう、異なる固有値にこの環境変数を設定します。それぞれの MPD リングは、異なる `I_MPI_JOB_CONTEXT` 値に関連付けられます。この環境変数が一度設定されると、1 つの MPD リングを開始でき、その他の MPD リングの影響を受けずに動作します。特定の MPD リングで動作するよう、`I_MPI_JOB_CONTEXT` に適切な値を設定します。複数のインテル® MPI ライブラリーのジョブを一度に起動する方法は、「簡素化されたジョブ起動コマンド」をご覧ください。

I_MPI_JOB_TAGGED_PORT_OUTPUT

タグ付きの `mpd` ポート出力を on/off にします。

構文

`I_MPI_JOB_TAGGED_PORT_OUTPUT=<arg>`

引数

`<arg>`	バイナリー・インジケーター。
`enable | yes | on | 1`	タグの出力を on にします。これは、デフォルト値です。
`disable | no | off | 0`	タグの出力を `off` にします

説明

出力形式のタグ付けは `mpdbot` ステージで動作し、起動時に `ssh` などのリモートシェルからの予期しない出力を分かりやすくできます。`mpdboot` は、この環境変数を自動的に 1 に設定します。この機能を必要としない場合、`I_MPI_JOB_TAGGED_PORT_OUTPUT` に `disable` を設定します。

I_MPI_MPD_CHECK_PYTHON

MPD リング開始時に Python* のバージョン・チェックを on/off します。

構文

`I_MPI_MPD_CHECK_PYTHON=<arg>`

引数

`<arg>`	バイナリー・インジケーター。
`enable | yes | on | 1`	Python* バージョンの互換性をチェックします。
`disable | no | off | 0`	Python* バージョンの互換性をチェックしません。これは、デフォルト値です。

説明

クラスターのノードにインストールされている Python* のバージョン互換性チェックを有効にするには、この環境変数を設定します。これにより、MPD リングの起動時間が長くなります。互換性のないバージョンの Python* がクラスターにインストールされている場合、MPD の動作は未定義です。

`I_MPI_MPD_CHECK_PYTHON` が、`enable` に設定され、互換性チェックに失敗すると、`mpdboot` は異常終了して診断メッセージを出力します。MPD リングは開始されません。

I_MPI_MPD_RSH

`mpd` デーモンを起動するリモートシェルを設定します。

構文

```
I_MPI_MPD_RSH =<arg>
```

引数

`<arg>`	文字列パラメーター。
`<remote shell>`	リモートシェルを使用します。

説明

この環境変数を設定して、`–rshmpdboot` オプションに適用されるデフォルトの値を定義します。コマンドラインで `-rsh` オプションが指定されると、`I_MPI_MPD_RSH` 環境変数は無視されます。`-rsh` オプションが指定されない場合、`I_MPI_MPD_RSH` 環境変数の値が想定されます。

I_MPI_MPD_TMPDIR
TMPDIR

MPD サブシステムの一時ディレクトリーを設定します。

構文

```
I_MPI_MPD_TMPDIR=<arg>
```

```
TMPDIR=<arg>
```

引数

`<arg>`	文字列パラメーター。
`<directory name>`	一時ディレクトリーの場所を指す文字列デフォルト値は `/tmp` です。

説明

代替の一時ディレクトリーの場所を指定するには、これらの環境変数のいずれかを設定します。MPD サブシステムは、これらの環境変数で指定されたディレクトリーにファイルを作成します。2 つの環境変数が異なる場所を指す場合、`TMPDIR` 環境変数の値は無視されます。

注意

一部のオペレーティング・システムでは、`mpd2.console_*` ファイルパスの長さが制限されます。次のような診断メッセージが表示された場合、問題を回避するには `<directory name>` の文字列長を減らしてください: `socket.error: AF_UNIX path too long`。

注意

<arg> が、分散ファイルシステム (PANFS、PVFS など) を示す場合、mpd デーモンは開始されません。この問題が生じた場合、ext2、ext3、NFS などの標準ファイルシステムを指すように I_MPI_MPD_TMPDIR と TMPDIR を設定してください。

I_MPI_MPD_CLEAN_LOG

MPD デーモン終了時のログファイルの削除を制御します。

構文

```
I_MPI_MPD_CLEAN_LOG=<value>
```

引数

<value>	値を定義します。
enable | yes | on | 1	ログファイルを削除します。
disable| no|off| 0	ログファイルを保持します。これは、デフォルト値です。

説明

この環境変数を設定して、mpdallexit の動作を定義します。この環境変数を enable に設定すると、mpdallexit は実行中に作成したログファイルを削除します。この環境変数を disable に設定すると、mpdallexit は実行中に作成したログファイルを保持します。

2.7. 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 [[-]<options>]]
```

引数

<options>	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します。 • 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. チューニング・リファレンス

インテル® MPI ライブラリーは、多くの環境変数の適切な値の選択を支援する実行時のプログラムの動作やパフォーマンスを自動的に調整する自動チューニング・ユーティリティーとを提供しています。

3.1. 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 \"<application command line>\" ] [ -of <file-name> ] \
[ -t \"<test_cmd_line>\" ] [-cm ] [ -d ] [ -D] \
[ -dl [d1[,d2...[,dN]]] ] [ -fl [f1[,f2…[,fN]]] ] [ -er ] \
[ -hf <hostsfile> ] [ -h ] [ -hr {min:max|min:|:max} ] \
[ -i <count> ] [ -mr {min:max|min:|:max}] [ -od <outputdir> ] \
[ -odr <outputdir> ] [ -r <rshcmd>] [ -pr {min:max|min:|:max}] \
[ -sf [file-path] ] [ -ss ] [ -s ] [ -td <dir-path> ] \
[ -tl <minutes>] [ -mh ] [ -os <opt1,...,optN> ] \
[ -oe <opt1,...,optN> ] [ -V ] [ -vi {percent} | -vix {X factor} ]\
[ - zb ] [ -t ] [ -so ] [ -ar \"reg-expr\" ] [ -trf <appoutfile> ]\
[ -m {base|optimized} ] [ -avd {min|max} ][ -pm {mpd|hydra} ]\
[ -co ] [ -sd ] [ -soc ]
```

または

```
mpitune [ --application \"<app_cmd _ line>\" ] [ --output-file <file-name> ]\
[ --test \"<test_cmd_line>\" ] [ --cluster-mode ] [ --debug ] \
[ --distinct ] [ --device-list [d1[,d2,… [,dN]]] ] \
[ --fabric-list [f1[,f2...[,fN]]] ][ --existing-ring ]\
[ --host-file <hostsfile> ] [ --help ] \
[ --host-range {min:max|min:|:max} ] [ --iterations <count> ] \
[ --message-range {min:max|min:|:max} ] \
[ --output-directory <outputdir> ] \
[ --output-directory-results <outputdir> ] [ --rsh <rshcmd> ] \
[ --ppn-range {min:max|min:|:max} | --perhost-range {min:max|min:|:max} ] \
[ --session-file [file-path] ] [ --show-session ] [ --silent ] \
[--temp-directory <dir-path> ] [ --time-limit <minutes> ] \
[ --master-host ] [ --options-set <opt1,...,optN> ] \
[ --options-exclude <opt1,...,optN> ] [ --version ] \
[ --valuable-improvement | --valuable-improvement-x {X factor} ]\
[ --zero-based ] [ --trace] [ --scheduler-only ] \
[ --application-regexp \"reg-expr\" ] \
[ --test-regexp-file <appoutfile> ] [ --model {base|optimized} ] \
[ --application-value-direction {min|max} ] \
[ --process-manager {mpd|hydra} ] [ -co ] [ -sd ] [ -soc ]
```

引数

-a \"<app_ cmd_line>\"| --application \"<app_cmd_line>\"	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 <file-name> | --output-file <file-name>	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、$PWD/app.conf です。
-t \"<test_cmd_line>\" | --test \"<test_cmd_line>\"	クラスター固有モードで指定するベンチマーク・プログラムを、インテル® MPI Benchmarks と入れ替えます。バックスラッシュを含む完全なコマンドラインを入力します。
-cm {exclusive|full} | --cluster-mode {exclusive|full}	クラスター利用モードを設定します。 • full - 最大数のタスクが実行されます。これはデフォルトのモードです。 • exclusive - クラスター上で 1 つのタスクのみが同時実行されます。
-d | --debug	デバッグ情報を表示します。
-D | --distinct	すべてのオプションを個別にチューニングします。この引数はクラスター固有モードでのみ有効です。
-dl [d1[,d2...[,dN]]] | --device-list[d1[,d2,…[,dN]]]	チューニングするデバイスを選択します。以前に設定したデバイスは無視されます。 デフォルトでは、<installdir>/<arch>/etc/devices.xml ファイルにリストされるすべてのデバイスを使用します。
-fl [f1[,f2...[,fN]]] | --fabric-list [f1[,f2…[,fN]]]	チューニングするファブリックを選択します。以前に設定したファブリックは無視されます。 デフォルトでは、<installdir>/<arch>/etc/fabrics.xml ファイルにリストされるすべてのファブリックを使用します。
-er | --existing-ring	既存の MPD リングを使用します。デフォルトでは、新しい MPD リングが作成されます。この引数は、I_MPI_PROCESS_MANAGER 環境変数に mpd が設定されている場合にのみ有効です。
-hf <hostsfile> | --host-file <hostsfile>	代替のホストファイル名を指定します。 デフォルトは、$PWD/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。デフォルトの min 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 <count> | --iterations <count>	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 <outputdir> | --output-directory <outputdir>	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 <outputdir> | --output-directory-results<outputdir>	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 <installdir>/<arch>/etc です。 クラスター固有モードで、<installdir>/<arch>/etc が利用できない場合、$PWD が選択されます。
-r <rshcmd> | --rsh <rshcmd>	デーモン (必要であれば) とジョブを開始するリモートシェルを指定します。デフォルト値は ssh です。
-pr {min:max|min:|:max} | --ppn-range {min:max|min:|:max} | --perhost-range {min:max|min:|:max}	ホストごとの最大プロセッサー数を設定します。デフォルトの min 値は 1 です。デフォルトの max 値は、プロセッサーのコア数です。min: または :max 形式は、必要に応じてデフォルト値を取ります。
-sf [file-path] | --session-file [file-path]	[file-path] セッション・ファイルに保存されている状態から、チューニングを再開します。
-ss | --show-session	セッションファイルと終了に関する情報を表示します。このオプションは、-sf オプションと併用する場合にのみ効果があります。
-s | --silent	すべての診断を抑制します。
-td <dir-path> | --temp-directory <dir-path>	一時データが使用するディレクトリー名を指定します。 デフォルトで、$PWD/mpitunertemp 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 <minutes> | --time-limit <minutes>	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 {percent} | --valuable-improvement {percent} -vix{X factor} | --valuable-improvement- x {X factor}	パフォーマンス向上のしきい値を制御します。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 \"reg-expr\" | --application-regexp \"reg-expr\"	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。reg-expr (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すことができます。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 <appoutfile> | --test-regexp-file <appoutfile>	reg-expr (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。

MPI tuner 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。

3.1.1. 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。

`<installdir>/<arch>/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。

  ```
  $mpirun -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設定オプションを検索します。`<installdir>/<arch>/etc` への書き込み権限がある場合、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。そうでない場合、現在の作業ディレクトリーに保存されます。

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。

```
$ mpirun -tune -genv I_MPI_FABRICS shm:dapl -ppn 8 -n 32 ./myprog
```

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。

   ```
   $mpirun -tune -n 32 ./myprog
   ```

3.1.2. アプリケーション固有のチューニング

チューナーにコマンドラインを指定することで、任意のアプリケーションのチューニングを実行します。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、もっと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、同様な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 \"mpirun -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。

```
$mpirun -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) 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、異なる名前で保存し、必要な時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。

高速チューニング

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

構文

`--fast [<value>]` または `-f [<value>]`

または

`I_MPI_TUNE_FAST=<value>`

引数

`<value>`	バイナリー・インジケーター。
`enable | yes | on | 1`	高速アプリケーション・チューニングを有効にします。この値は、アプリケーションのチューニングモードでのみ有効です。
`disable | no | off | 0`	高速アプリケーション・チューニングを無効にします。これは、デフォルト値です。

説明

I_MPI_TUNE_FAST を enable に設定すると、mpitune ユーティリティーは代替の高速アプリケーション・チューニング手順を実行します。高速アプリケーション・チューニングは、前回のチューニングに使用した設定ファイルを使用します。

例 1

```
$ mpitune --application \”mpirun …\” --fast
```

例 2

```
$ export I_MPI_TUNE_FAST=enable
$ mpitune --application \”mpirun …\”
```

注意

-help と -fast オプションを指定すると、app_tune に関するヘルプが表示されます。

トポロジーを考慮したアプリケーションのチューニング

ここでは、mpitune ユーティリティーを使用してトポロジーを考慮したチューニングを行う方法を説明します。ダイナミック・メソッドでこのチューニングを行う場合、「-use-app-topology」と「I_MPI_HYDRA_USE_APP_TOPOLOGY」の説明をご覧ください。

I_MPI_TUNE_RANK_PLACEMENT

構文

`--rank-placement [<value>]` または `-rp [<value>]`

または

`I_MPI_TUNE_RANK_PLACEMENT=<value>`

引数

`<value>`	バイナリー・インジケーター。
`enable | yes | on | 1`	mpitune ユーティリティーをトポロジー・チューニング・ツールに切り替えます。この値は、アプリケーションのチューニング・モードでのみ有効です。例えば、–application オプションもしくはアプリケーション通信グラフ (ACG) とオプションのハードウェア・トポロジー/グラフ (HTG) は、追加オプション -acg と -htg で渡されます。
`disable | no | off | 0`	トポロジー・チューニング・ツールを off に切り替えます。これは、デフォルト値です。

説明

I_MPI_TUNE_RANK_PLACEMENT を enable に設定すると、mpitune ラッパーは代替トポロジーツール (mpitune_rank_placement) を実行します。

例

```
$ mpitune --application \”mpirun … \” --rank-placement
$ mpitune --application \”mpirun … \” --rank-placement enable
$ mpitune --application \”mpirun … \” -rp -acg <path to acg_file> -htg <path to htg_file>
```

結果は、host ファイルと host ファイルを使用して自動的に記録される設定ファイルです。

注意

--help と --rank-placement オプションを指定すると、mpitune_rank_placement に関するヘルプが表示されます。

I_MPI_TUNE_APPLICATION_STATISTICS

構文

`--application-statistics [<value>]` または `-s [<value>]`

または

`I_MPI_TUNE_APPLICATION_STATICSTICS=<value>`

引数

`<value>`	インテル® MPI ライブラリーのネイティブ統計ファイルのレベル 1 以上へのパス。この設定は、--rank-placement オプションと同時に指定された場合にのみ適用されます。

説明

インテル® MPI ライブラリーの統計ファイルを mpitune (mpitune_rank_placement) へ渡すと、チューニング時間を短縮できます。

例

```
$ mpitune -rp -s <path to statistics file>
```

I_MPI_TUNE_APPLICATION_COMMUNICATION_GRAPH

構文

```
--application-communication-graph [<value>] または -acg [<value>]
```

または

```
I_MPI_TUNE_APPLICATION_COMMUNICATION_GRAPH=<value>
```

引数

<value>	ACG ファイルへのパス。この設定は、--rank-placement オプションと同時に指定された場合にのみ適用されます。

説明

ACG ファイルを mpitune (mpitune_rank_placement) へ渡すと、チューニング時間を短縮できます。

例

```
$ mpitune -rp -acg <path to acg_file>
```

I_MPI_TUNE_HARDWARE_TOPOLOGY_GRAPH

構文

```
--hardware-topology-graph [<value>] または -htg [<value>]
```

または

```
I_MPI_TUNE_HARDWARE_TOPOLOGY_GRAPH=<value>
```

引数

<value>	ハードウェア・トポロジー・グラフが記述されたファイルへのパス。この設定は、--rank-placement オプションと同時に指定された場合にのみ適用されます。

説明

HTG ファイルを mpitune (mpitune_rank_placement) へ渡すと、チューニング時間を短縮できます。

例

```
$ mpitune -rp -acg <path to acg_file> -htg <path to htg_file>
```

3.1.3. 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。

3.2. 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。

3.2.1. 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`

3.2.2. 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。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するか、論理 CPU 番号を特定するため `cat /proc/cpuinfo` コマンドを実行します。

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

論値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。

3.2.3. 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=<arg>
```

引数

<arg>	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_MODE

ピニングの方式を選択します。

構文

```
I_MPI_PIN_MODE=<pinmode>
```

引数

<pinmode>	CPU ピニングモードを選択します。
mpd|pm	関連するプロセス管理 (多目的デーモン/MPD や Hydra) 内部のプロセスをピニングします。これは、デフォルト値です。
lib	インテル® MPI ライブラリー内部でプロセスをピニングします。

説明

ピニング方式を選択するには、I_MPI_PIN_MODE 環境変数を設定します。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。

mpd デーモンや Hydra プロセスランチャーがシステムで提供される方法でプロセスをピニングするようにするには (可能な場合)、I_MPI_PIN_MODE 環境変数を mpd または pm に設定します。ピニングは、MPI プロセスが起動される前に行います。これにより、CPU とメモリーにプロセスを配置できます。ピニングは、Altix* などの非均一メモリー・アーキテクチャー (NUMA) システムでは利点があります。NUMA 環境では、プロセッサーは自身のローカルメモリーに高速にアクセスできます。

I_MPI_PIN_MODE 環境変数を lib に設定すると、インテル® MPI ライブラリーはプロセスをピニングします。このモードは、CPU とメモリーの同じ場所に配置されたプロセスには機能しません。

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=<value>

I_MPI_PIN_DOMAIN 環境変数には以下の構文があります。

1. <proclist>
2. [<procset>][:[grain=<grain>][,shift=<shift>][,preoffset=<preoffset>][,postoffset =<postoffset>]
3. [<procset>][:map=<map>]

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=<proclist>

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=<proclist>
```

引数

<proclist>	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=[<procset>][:[grain=<grain>][,shift=<shift>][,preoffset
=<preoffset>][,postoffset=<postoffset>]
```

引数

<procset>	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。

<grain>	定義された <procset> に、セルをピニング (固定)する粒度を指定します。最小 <grain> 値は、<procset> の単一要素です。最大 <grain> 値は、ソケットの <procset> 要素の数です。<grain> 値は、<procset> 値の倍数でなければいけません。 そうでない場合、最小 <grain> 値が想定されます。デフォルトは、最小 <grain> 値です。

<shift>	<procset> 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。 <shift> は、定義された <grain> ユニットを基準とします。 <shift> 値は、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。
<preoffset>	<preoffset> 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 <procset> の巡回シフトを指定します。値は、定義された <grain> ユニットを基準とします。 <preoffset> 値は、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す。
<postoffset>	<postoffset> 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 <procset> の巡回シフトを指定します。値は、定義された <grain> ユニットを基準とします。 <postoffset> 値は、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す。

次の表は、<grain>、<shift>、<preoffset> および <postoffset> 向けの値を示します。

<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ユニットと同じ数のパラメータ値を指定します。
cache1	L1 キャッシュを共有するパラメータユニットと同じ数のパラメータ値を指定します。
cache2	L2 キャッシュを共有するパラメータユニットと同じ数のパラメータ値を指定します。
cache3	L3 キャッシュを共有するパラメータユニットと同じ数のパラメータ値を指定します。
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ユニットと同じ数のパラメータ値を指定します。
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=[<procset>][:map=<map>]
```

引数

<map>	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	共有リソース (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

- grain/shift/offset マッピング。この方式は、<shift>*<grain> に等しいステップと、末端が <offset>*<grain> 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、<shift> 回繰り返されます。

  例: grain = 2、論理プロセッサー、shift = 2 grains、offset = 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。2 つのシナリオがあります: `bunch` と `scatter`。

`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、プロセスは共有リソース (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。

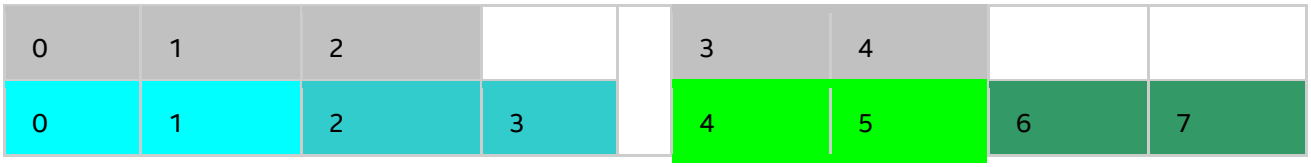
凡例:

灰色 - MPI プロセス

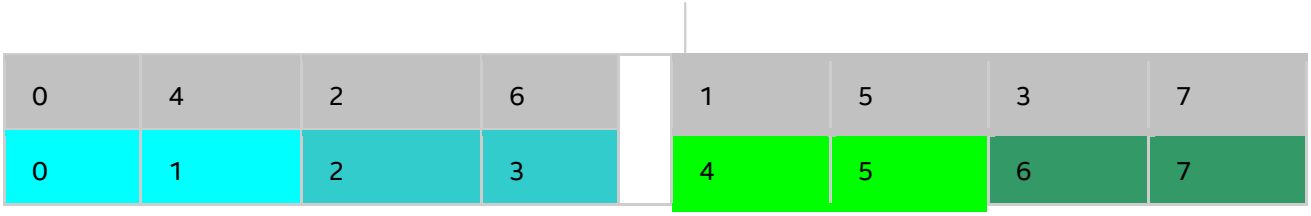
水色 - 最初のソケットのプロセッサー

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定義します



5 プロセスでの bunch シナリオ



すべてを使用する scatter シナリオ

例

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。

```
$ mpirun -genv I_MPI_PIN_PROCESSOR_LIST 0,3 \
-n <# of processes> <executable>
```

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。

```
$ mpirun -host host1 -env I_MPI_PIN_PROCESSOR_LIST 0,3 \
-n <# of processes> <executable> : \
-host host2 -env I_MPI_PIN_PROCESSOR_LIST 1,2,3 \
-n <# of processes> <executable>
```

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。

```
$ mpirun -genv I_MPI_DEBUG 4 -m -host host1 \
-env I_MPI_PIN_PROCESSOR_LIST 0,3 -n <# of processes> <executable> :\
-host host2 -env I_MPI_PIN_PROCESSOR_LIST 1,2,3 \ -n <# of processes> <executable>
```

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_PROCESSOR_EXCLUDE_LIST

意図するホスト上でピニングを使用するため、除外する論理プロセッサーのサブセットを定義します。

構文

`I_MPI_PIN_PROCESSOR_LIST=<proclist>`

引数

`<proclist>`	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。
`<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。

説明

意図するホスト上でインテル® MPI ライブラリーがピニングに使用しない論理プロセッサーを定義するには、この環境変数を設定します。論理プロセッサーは、`/proc/cpuinfo` のように番号付けされます。

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=<cell>`

引数

`<cell>`	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。

I_MPI_PIN_RESPECT_CPUSET

プロセスのアフィニティー・マスクが順守されます。

構文

```
I_MPI_PIN_RESPECT_CPUSET=<value>
```

引数

<value>	バイナリー・インジケーター。
enable | yes | on | 1	プロセスのアフィニティー・マスクを順守します。これは、デフォルト値です。
disable | no | off | 0	プロセスのアフィニティー・マスクを順守しません。

説明

I_MPI_PIN_RESPECT_CPUSET=enable に設定すると、Hydra プロセスランチャーは、インテル® MPI ライブラリーのピニングに適用する各ホスト上の論理プロセッサーを決定するため、プロセス・アフィニティー・マスクを使用します。

I_MPI_PIN_RESPECT_CPUSET=disable に設定すると、Hydra プロセスランチャーは、インテル® MPI ライブラリーのピニングに適用する各ホスト上の論理プロセッサーを決定するため、プロセス・アフィニティー・マスクを使用しません。

I_MPI_PIN_RESPECT_HCA

Infiniband* アーキテクチャーのホスト・チャネル・アダプター (IBA* HCA*) が存在する場合、IBA HCA の位置に応じてピニングを調整します。

構文

```
I_MPI_PIN_RESPECT_HCA=<value>
```

引数

<value>	バイナリー・インジケーター。
enable | yes | on | 1	可能であれば、IBA HCA の位置を使用します。これは、デフォルト値です。
disable | no | off | 0	IBA HCA の位置を使用しません。

説明

I_MPI_PIN_RESPECT_HCA=enable に設定すると、Hydra プロセスランチャーは、インテル® MPI ライブラリーのピニングに適用するため、各ホスト上の IBA HCA を位置を使用します。

I_MPI_PIN_RESPECT_HCA=disable に設定すると、Hydra プロセスランチャーは、インテル® MPI ライブラリーのピニングに適用するため、各ホスト上の IBA HCA を位置を使用しません。

3.2.4. 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。図を参照してください。

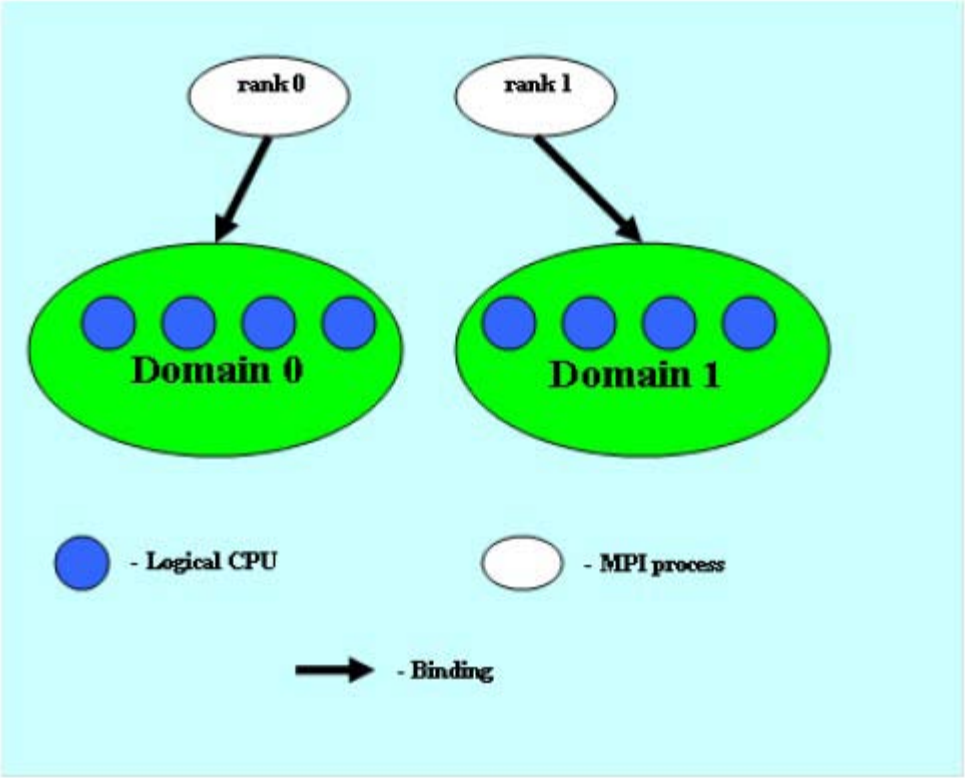


図 3.2-1 ドメインの例

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 `<mc-shape>` 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・レイアウト `<size>[:<layout>]` を介したドメイン定義
- ビットマスク `<masklist>` 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=<mc-shape>

<mc-shape>	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=<size>[:<layout>]

<size>	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 10 進数 <n> でドメインサイズを指定します。

<layout>	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>

<masklist>	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]	<masklist> 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** <masklist> の設定が正しく解釈されることを確実にするため、<masklist> 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。

注意

次の設定は、ピニングが行われていない場合と同じ効果を持ちます。

- `I_MPI_PIN_DOMAIN=auto` に設定し、ノードで単一プロセスが実行されている場合 (例えば、`I_MPI_PERHOST=1` によって)
- `I_MPI_PIN_DOMAIN=node`

マルチソケット・プラットフォーム上でソケット間でプロセスを移行させたくない場合、`I_MPI_PIN_DOMAIN=socket` または小さな値にドメインサイズを設定します。

また、各ランク (アフィニティー・マスクは自動的に IBA* HCA に調整されます) に単一の CPU プロセスのアフィニティー・マスクを生成するため `I_MPI_PIN_PROCESSOR_LIST` を使用できます。

SMP ノードのモデルを以下に示します。

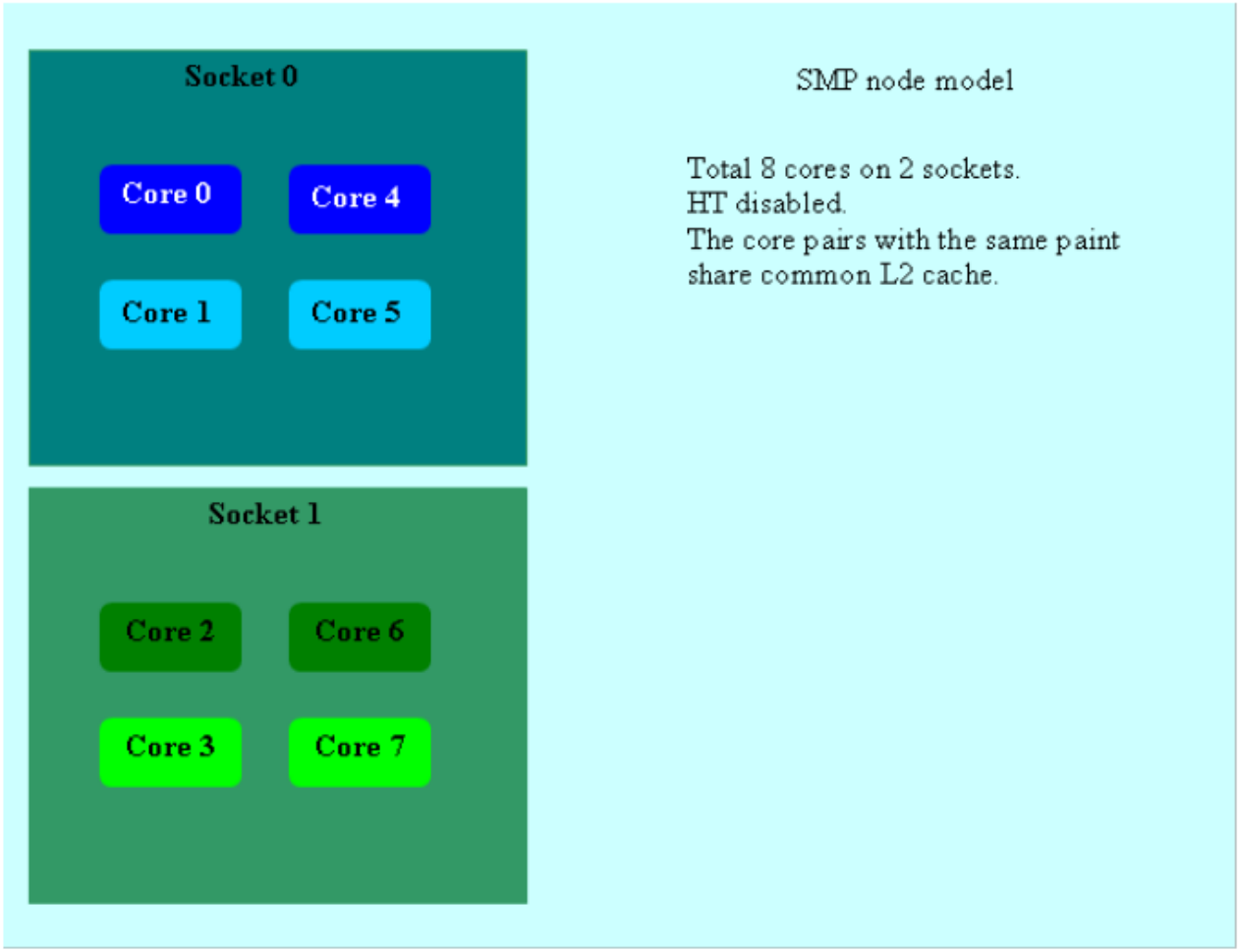


図 3.2-2 ノードのモデル

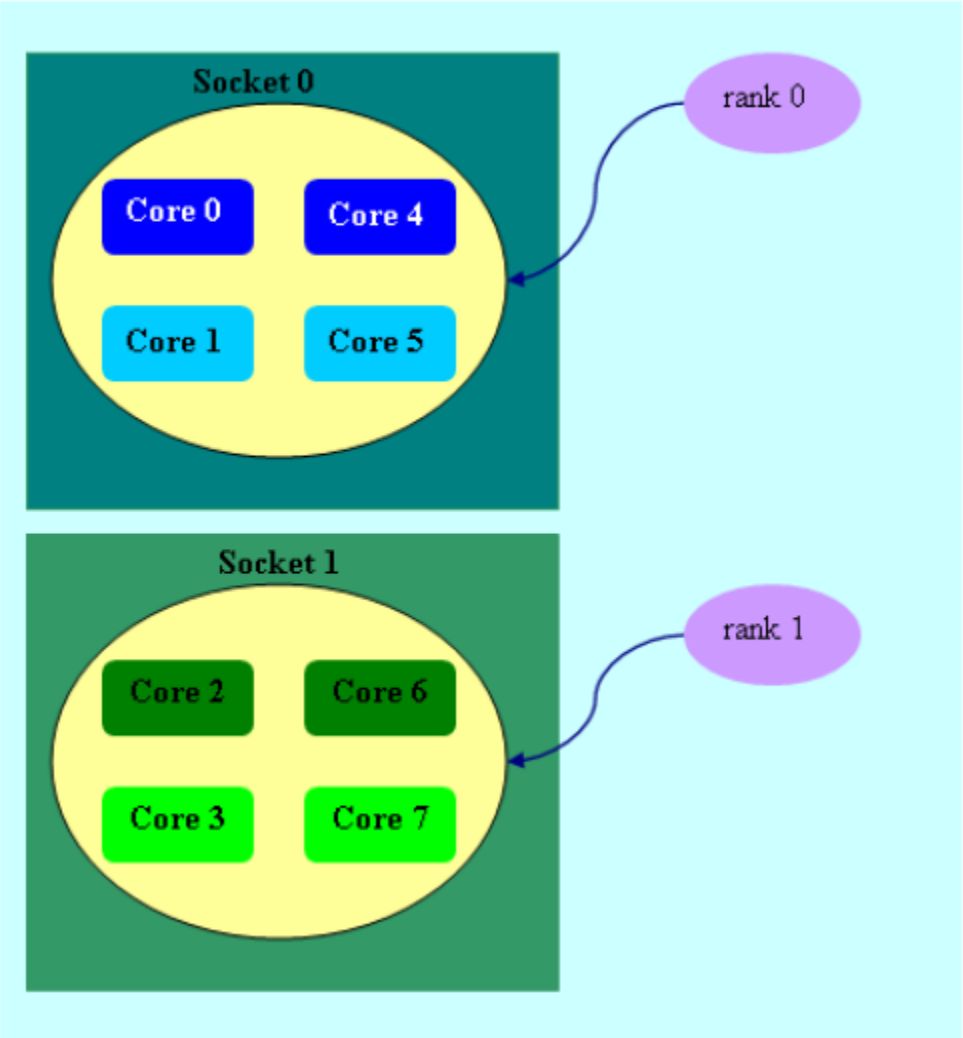


図 3.2-3 `mpirun -n 2 -env I_MPI_PIN_DOMAIN socket ./a.out`

図 3.2-3 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。

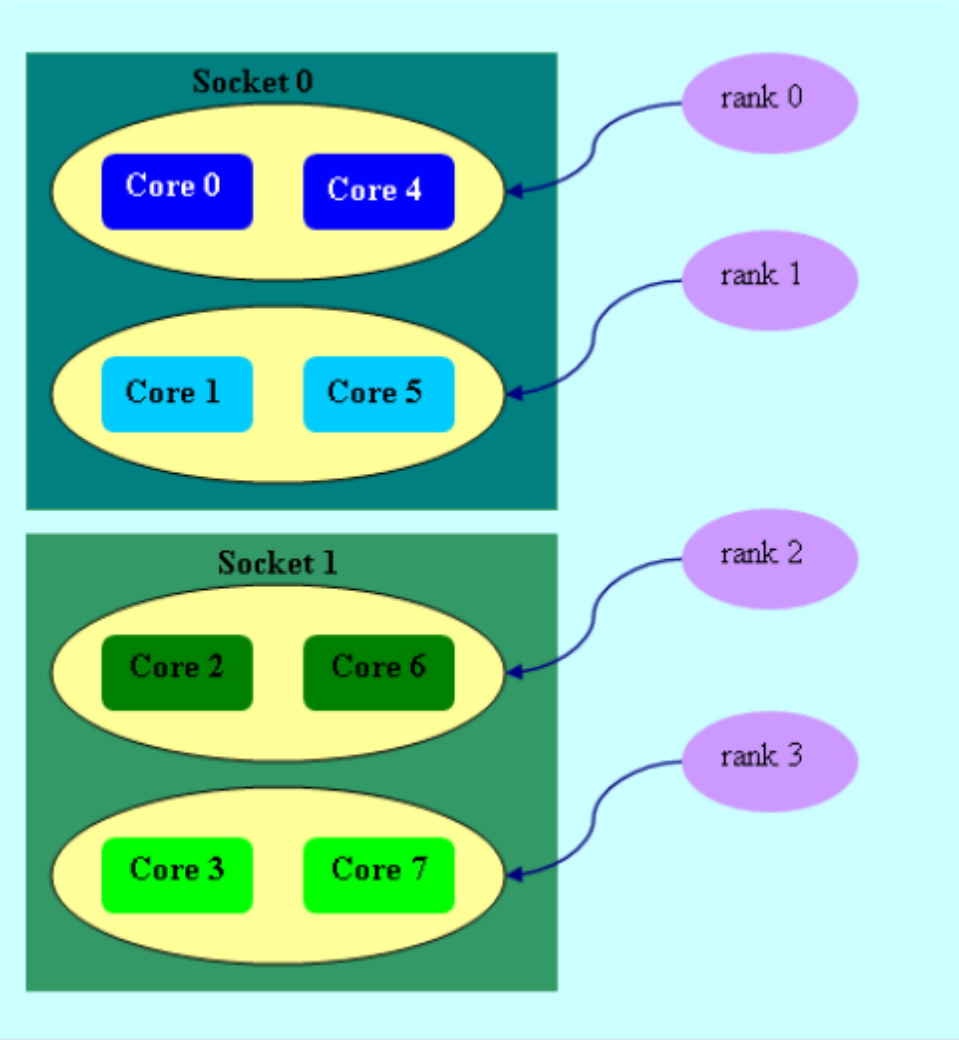


図 3.2-4 `mpirun -n 4 -env I_MPI_PIN_DOMAIN cache2 ./a.out`

図 3.2-4 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。

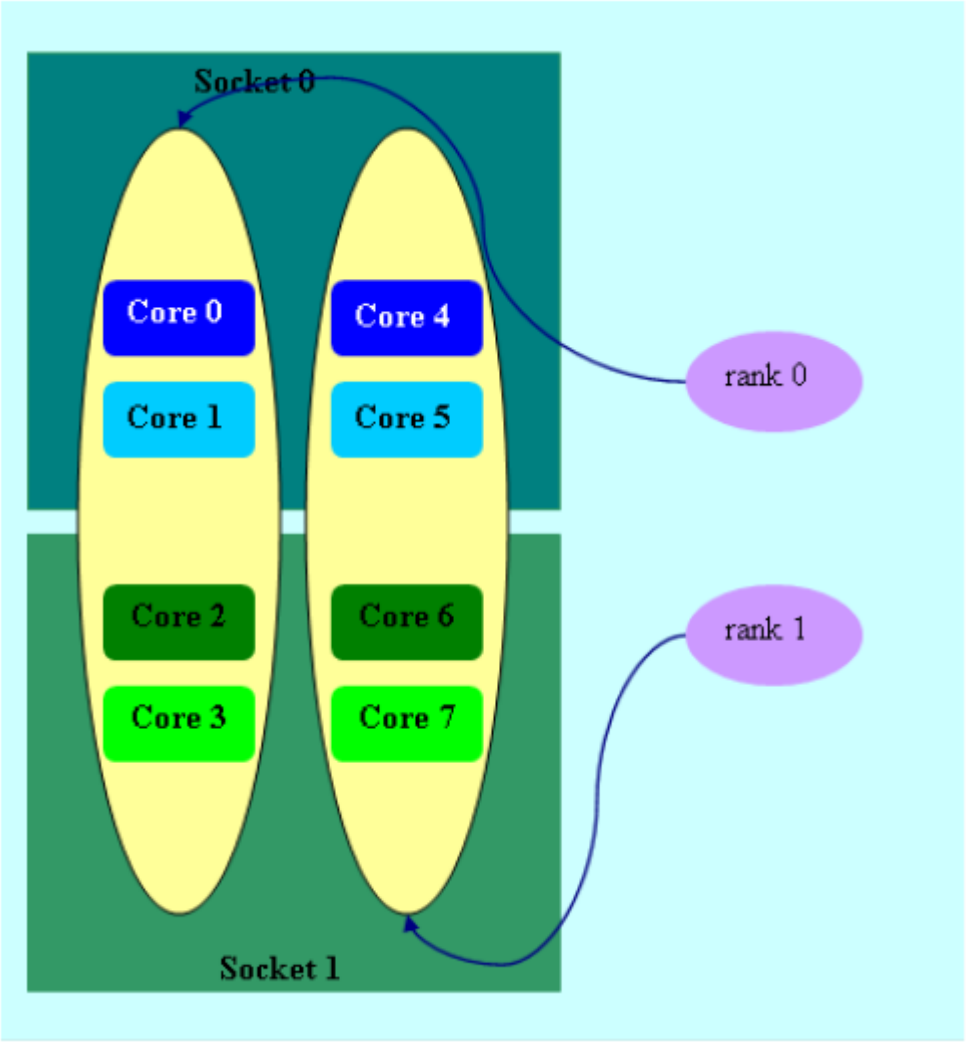


図 3.2-5 `mpirun -n 2 -env I_MPI_PIN_DOMAIN 4:platform ./a.out`

図 3.2-5 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。

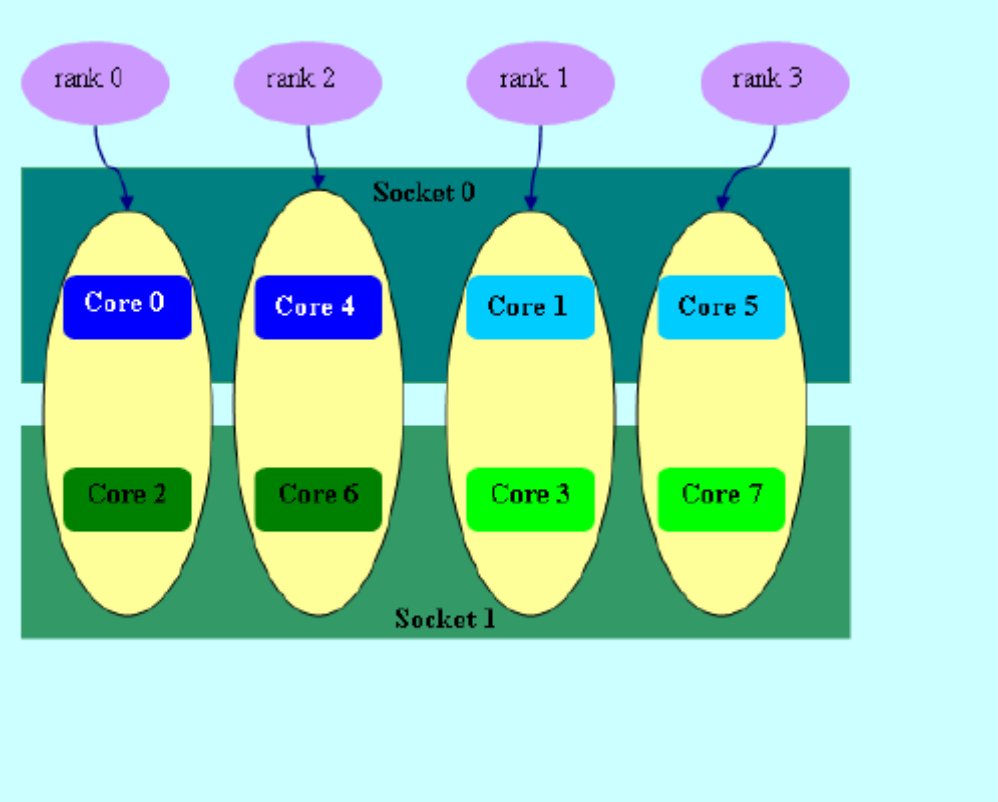


図 3.2-6 `mpirun -n 4 -env I_MPI_PIN_DOMAIN auto:scatter ./a.out`

図 3.2-6 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。

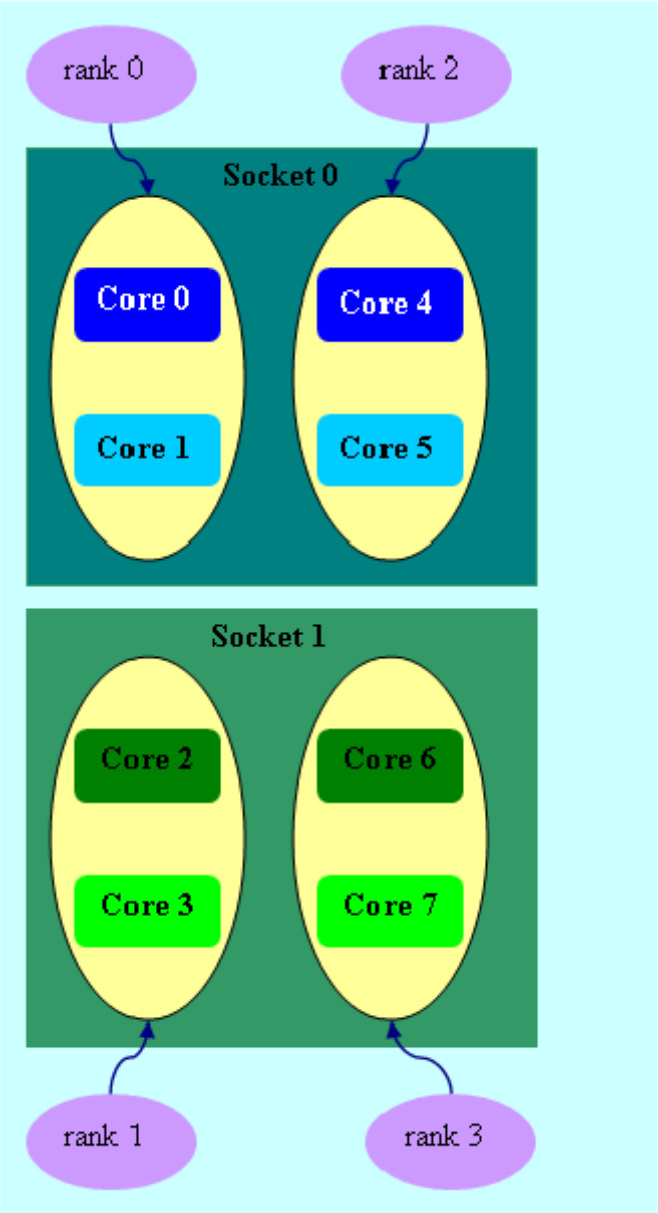


図 3.2-7 setenv OMP_NUM_THREADS=2

```
mpirun -n 4 -env I_MPI_PIN_DOMAIN omp:platform ./a.out
```

図 3.2-7 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。

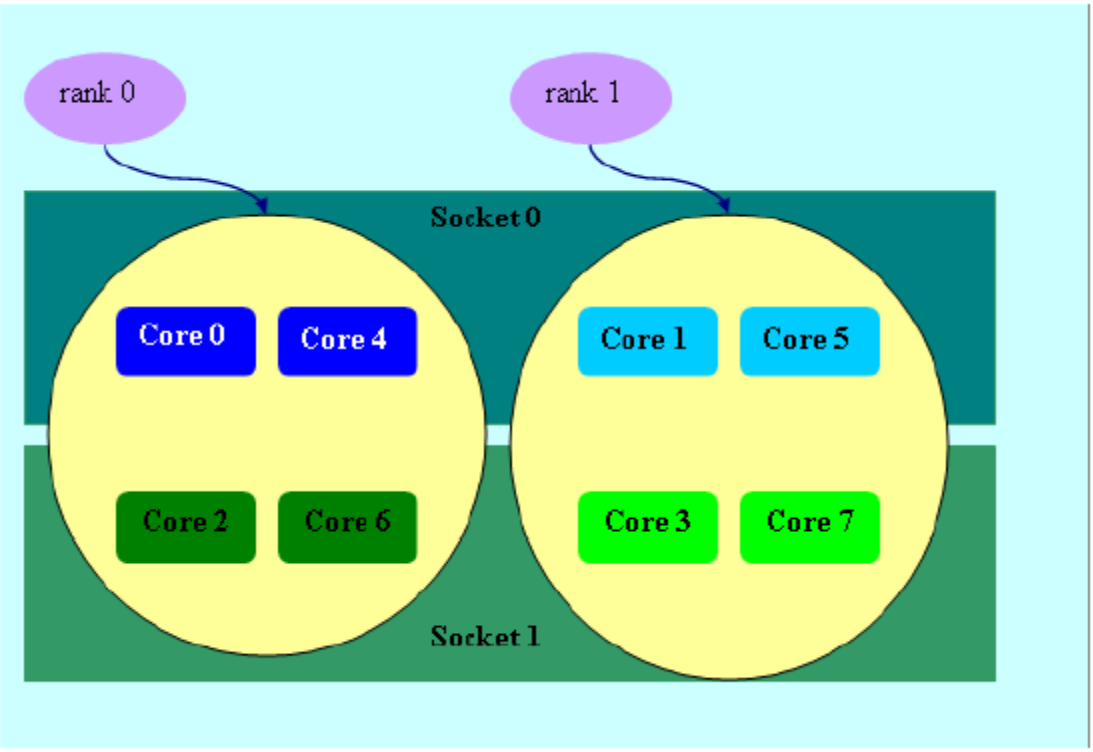


図 3.2-8 `mpirun -n 2 -env I_MPI_PIN_DOMAIN [0x55,0xaa] ./a.out`

図 3.2-8 (`I_MPI_PIN_DOMAIN=<masklist>` 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 MPI プロセスの順番割り当てを定義します。

構文

```
I_MPI_PIN_ORDER=<order>
```

引数

`<order>`	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
    ```

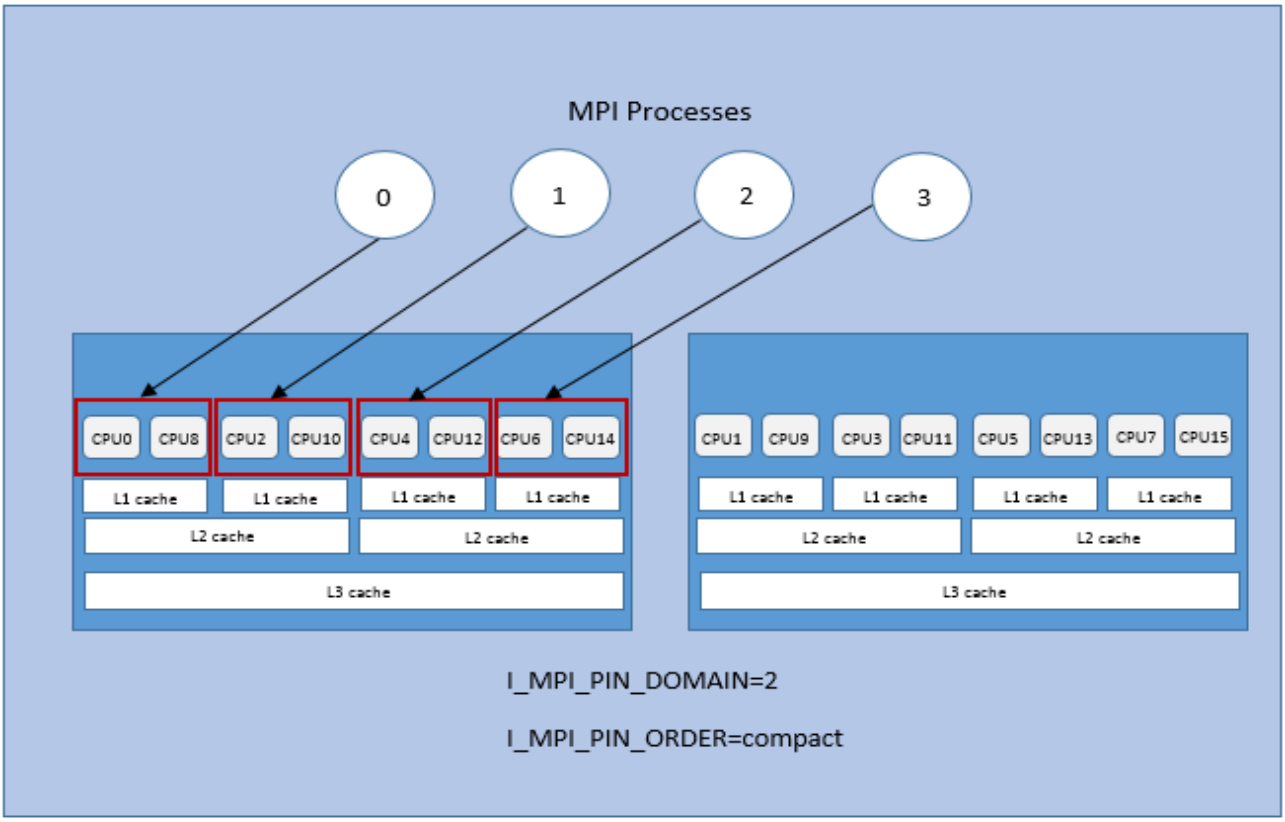


図 3.2-9 Compact オーダーの例

- scatter の場合:

  ```
  I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=scatter
  ```

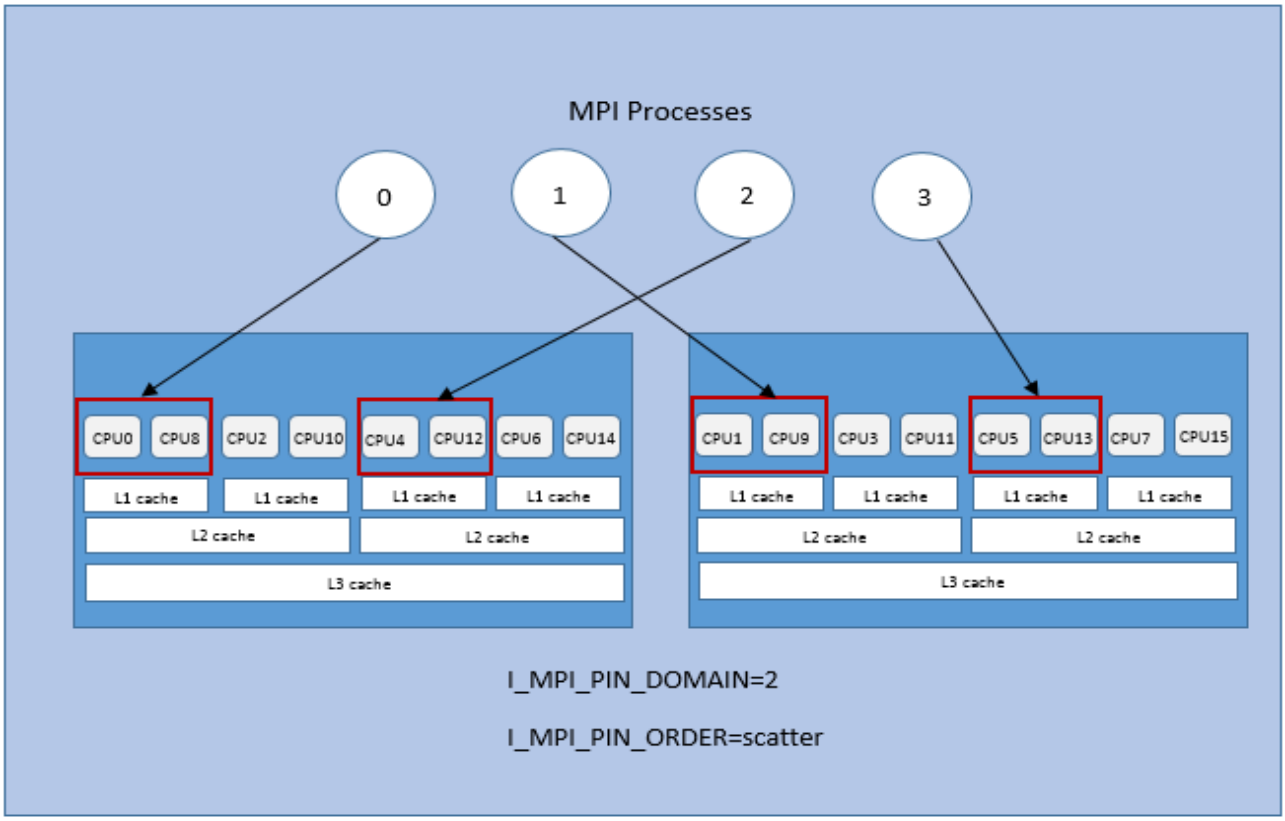


図 3.2-10 Scatter オーダーの例

- spread の場合:

  ```
  I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=spread
  ```

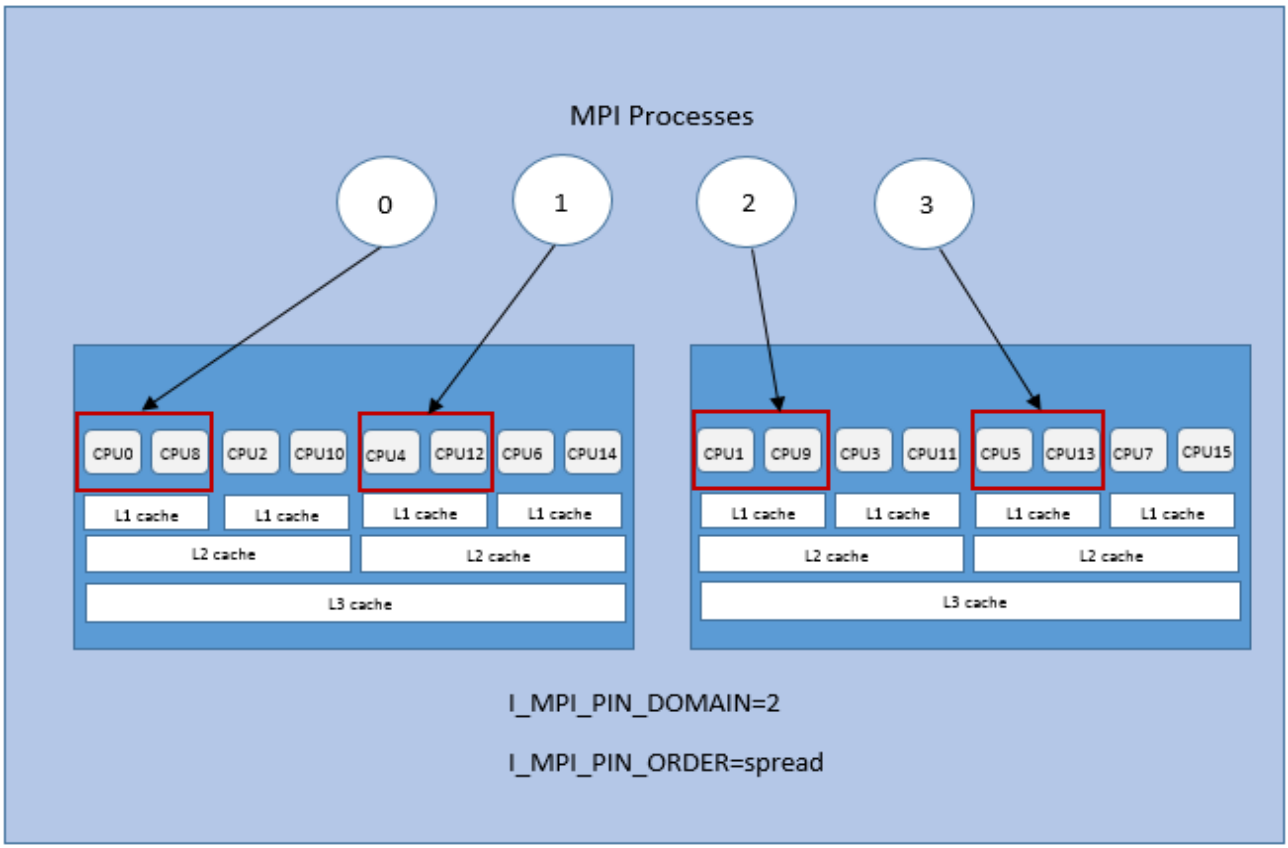


図 3.2-11 Spread オーダーの例

- bunch の場合:

```
I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=bunch
```

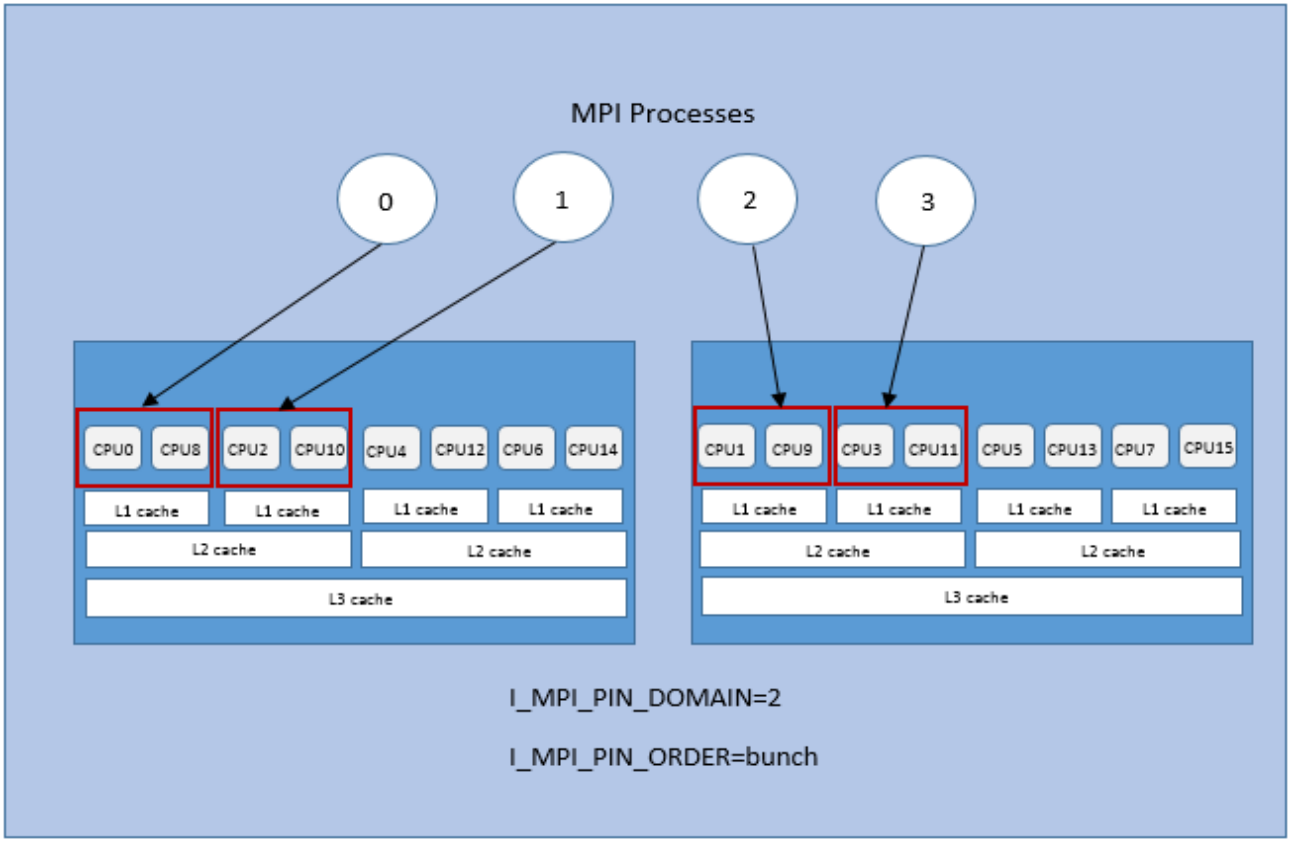


図 3.2-12 Bunch オーダーの例

3.3. 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
- DAPL UD 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
- TMI ネットワーク・ファブリック
- OFA* ネットワーク・ファブリック
- OFI ネットワーク・ファブリック

3.3.1. 通信ファブリック制御

I_MPI_FABRICS (I_MPI_DEVICE)

特定のファブリックを選択します。

構文

```
I_MPI_FABRICS=<fabric>|<intra-node fabric>:<inter-nodes fabric>
<fabric> := {shm, dapl, tcp, tmi, ofa, ofi}
<intra-node fabric> := {shm, dapl, tcp, tmi, ofa, ofi}
<inter-nodes fabric>:= {dapl, tcp, tmi, ofa, ofi}
```

廃止された構文

```
I_MPI_DEVICE=<device>[:<provider>]
```

引数

`<fabric>`	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。
`tmi`	インテル® True Scale や Myrinet* (タグマッチ・インターフェイスを介して) を含む、TMI ネットワーク・ファブリック。
`ofa`	InfiniBand* (OFED* を介して) を含む OFA ネットワーク・ファブリック。
`ofi`	インテル® True Scale ファブリックと TCP (OFI* API を介した) を含む OFI (OpenFabrics Interfaces*) ネットワーク・ファブリック。

`I_MPI_DEVICE` に対応

`<device>`	`<fabric>`
`sock`	`tcp`
`shm`	`shm`
`ssm`	`shm:tcp`
`rdma`	`dapl`
`rdssm`	`shm:dapl`
`<provider>`	オプションの DAPL* プロバイダー名 (`rdma` と `rdssm`)。 `I_MPI_DAPL_PROVIDER=<provider>` または `I_MPI_DAPL_UD_PROVIDER=<provider>`

`{rdma,rdssm}` デバイス向けにのみ `<provider>` 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 <# of processes> \
-env I_MPI_DEVICE rdssm:OpenIB-cma <executable>
```

これらのデバイスは、<provider> 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。

```
$ mpirun -n <# of processes> -env I_MPI_FABRICS shm <executable>
```

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。

```
$ mpirun -n <# of processes> -env I_MPI_FABRICS shm:dapl <executable>
```

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。

```
$ mpirun -n <# of procs> -perhost <# of procs per host> <executable>
```

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。

詳細については、「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=<fabrics list>
```

ここで、<fabrics list>:= <fabric>,...,<fabric> は次のとおりです。

```
<fabric> := {dapl, tcp, tmi, ofa, ofi}
```

引数

<fabrics list>	ファブリックのリストを指定します。
dapl,ofa,tcp,tmi,ofi	これは、デフォルト値です。
dapl,tcp,ofa,tmi,ofi	I_MPI_WAIT_MODE=enable に設定した場合、これはデフォルト値です。
tmi,dapl,tcp,ofa,ofi	これは、インテル® True Scale ファブリックを持ち、それ以外のインターコネクト・カードを持たない場合のデフォルトです。ホストが複数の HCA タイプを持つ場合、これは適用されません。

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=<arg>

廃止された構文

I_MPI_FALLBACK_DEVICE=<arg>

引数

<arg>	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_LARGE_SCALE_THRESHOLD

スケーラブルな最適化を有効にするしきい値を変更します。

構文

I_MPI_LARGE_SCALE_THRESHOLD=<arg>

引数

<nprocs>	スケールのしきい値を定義します。
> 0	デフォルト値は 4096 です。

説明

この環境変数は、DAPL UD IB 拡張が自動的に有効になっている場合、プロセス数を定義します。

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=<nbytes>

引数

<nbytes>	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 <nbytes> 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、<nbytes> 以下の場合、eager プロトコルが使用されます。
- メッセージが、<nbytes> 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=<nbytes>

引数

<nbytes>	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 <nbytes> 値は、262144 バイトです。shm の場合、カットオーバー・ポイントは、I_MPI_SHM_CELL_SIZE 環境変数の値と等しくなります。

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、<nbytes> 以下の場合、eager プロトコルが使用されます。
- メッセージが、<nbytes> 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=<scount>

引数

<scount>	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 <scount> 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、<scount> 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。<scount> 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。

<scount> の最適な値の選択は、経験に依存します。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=<arg>

引数

<arg>	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 プロセッサー以下のデフォルトです。

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=<arg>

引数

<arg>	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=<arg>

廃止された構文

I_MPI_USE_DYNAMIC_CONNECTIONS=<arg>

引数

<arg>	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。

3.3.2. 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=<arg>

廃止された構文

I_MPI_CACHE_BYPASS=<arg>

引数

<arg>	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サイズ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=<arg>
```

引数

<arg>	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・ファストボックスの持っています。

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=<nbytes>
```

引数

<nbytes>	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 <nbytes> 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=<num>
```

引数

<num>	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=<nbytes>
```

引数

<nbytes>	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 <nbytes> 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=<arg>
```

引数

<arg>	バイナリー・インジケーター。
shm	共有メモリーコピー LMT メカニズムを on にします。
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します。
- Linux* カーネルのバージョンが、クロスメモリーアタッチ (CMA) 機能をサポートする 3.2 以降の場合、中間バッファーなしでメッセージをコピーする直接コピーメカニズムを使用します。

I_MPI_SHM_LMT 環境変数を direct に設定し、オペレーティング・システムが CMA をサポートしていない場合、shm LTM メカニズムが使用されます。

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=<num>
```

廃止された構文

```
I_MPI_SHM_NUM_BUFFERS=<num>
```

引数

<num>	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=<nbytes>
```

廃止された構文

`I_MPI_SHM_BUFFER_SIZE=<nbytes>`

引数

`<nbytes>`	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 `<nbytes>` 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 =<arg>`

引数

`<arg>`	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=<num>`

引数

`<num>`	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_LMT_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=<nbytes>
```

引数

<nbytes>	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 <nbytes> 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

代替の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=<arg>
```

引数

<arg>	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=<arg>
```

廃止された構文

```
I_MPI_INTRANODE_SHMEM_BYPASS=<arg>
I_MPI_USE_DAPL_INTRANODE=<arg>
```

引数

<arg>	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、ネットワーク・ファブリック・レイヤーを使用して転送されます。

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:<fabric> 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=<shm_scount>

引数

<scount>	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	ノード間通信に tcp ファブリックを使用する場合、デフォルトの <shm_scount> 値は 100 スピンルールと等価です。 ノード間通信に ofa、tmi、ofi、または dapl ファブリックを使用する場合、デフォルトの <shm_scount> 値は 10 スピンルールと等価です。 デフォルト値は、2147483647 と等価です。

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_scount> 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_scount> 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_scount> 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。

3.3.3. 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=<name>
```

引数

`<name>`	ロードする DAPL プロバイダーの名前を定義します。

説明

ロードする DAPL プロバイダーの名前を定義するには、この環境変数を設定します。この名前は、`dat.conf` 設定ファイルでも定義されています。

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=<library>
```

引数

`<library>`	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 `libdat.so` または `libdat.so.1`、DAPL* 2.0 プロバイダーには `libdat2.so` または `libdat2.so.2` 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 と DAPL UD 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=<arg>
```

廃止された構文

```
I_MPI_RDMA_TRANSLATION_CACHE=<arg>
```

引数

`<arg>`	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=<arg>
```

引数

<arg>	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、OFA 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=<nbytes>
```

廃止された構文

```
I_MPI_RDMA_EAGER_THRESHOLD=<nbytes>
RDMA_IBA_EAGER_THRESHOLD=<nbytes>
```

引数

<nbytes>	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 <nbytes> 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 <nbytes> 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、<nbytes> 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変数は、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=<arg>
```

引数

<arg>	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=<arg>
```

廃止された構文

```
I_MPI_DYNAMIC_CONNECTION_MODE=<arg>
I_MPI_DYNAMIC_CONNECTIONS_MODE=<arg>
```

引数

<arg>	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。このモードは、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=<arg>
```

廃止された構文

```
I_MPI_RDMA_SCALABLE_PROGRESS=<arg>
```

引数

`<arg>`	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=<nbuf>
```

廃止された構文

```
I_MPI_RDMA_BUFFER_NUM=<nbuf>
NUM_RDMA_BUFFER=<nbuf>
```

引数

`<nbuf>`	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=<nbytes>
```

廃止された構文

```
I_MPI_RDMA_BUFFER_SIZE=<nbytes>
I_MPI_RDMA_VBUF_TOTAL_SIZE=<nbytes>
```

引数

<nbytes>	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、<nbytes> を最適値にバッファーをアライメント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=<arg>
```

廃止された構文

```
I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT=<arg>
I_MPI_RDMA_RNDV_BUF_ALIGN=<arg>
```

引数

<arg>	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 します。

構文

```
I_MPI_DAPL_RDMA_RNDV_WRITE=<arg>
```

廃止された構文

```
I_MPI_RDMA_RNDV_WRITE=<arg>
I_MPI_USE_RENDEZVOUS_RDMA_WRITE=<arg>
```

引数

`<arg>`	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=<arg>
```

廃止された構文

```
I_MPI_RDMA_CHECK_MAX_RDMA_SIZE=<arg>
```

引数

`<arg>`	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=<nbytes>

廃止された構文

I_MPI_RDMA_MAX_MSG_SIZE=<nbytes>

引数

<nbytes>	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 <nbytes> 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 <nbytes> 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 <nbytes> 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=<size>

廃止された構文

I_MPI_RDMA_CONN_EVD_SIZE=<size>
I_MPI_CONN_EVD_QLEN=<size>

引数

<size>	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、<size> 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=<arg>
```

引数

<nbytes>	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 <nbytes> 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、<nbytes> 以下のメッセージ。
- メッセージ・サイズが <nbytes> より大きいと、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=<nbuf>
```

引数

<nbuf>	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=<arg>
```

廃止された構文

```
I_MPI_RDMA_WRITE_IMM=<arg>
```

引数

<arg>	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てアルゴリズムが有効になると、特定の DAPL プロバイダーの属性は即時データ IB 拡張を持つ RDMA 書き込みがサポートされることを示します。

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=<num_procesess>

引数

<num_procesess>	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 <num_procesess> 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、<num_procesess> 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、<num_procesess> 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=<arg>

引数

<arg>	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、多数プロセスによる MPI ジョブで顕著です。

3.3.4. DAPL UD ネットワーク・ファブリック制御

I_MPI_DAPL_UD

DAPL UD パスの使用を有効/無効にします。

構文

`I_MPI_OFA_USE_XRC=<arg>`

引数

`<arg>`	バイナリー・インジケーター。
`enable | yes | on | 1`	DAPL UD IB 拡張の使用を on にします。
`disable | no | off | 0`	DAPL UD IB 拡張の使用を `off` にします。これは、デフォルト値です。

説明

データを転送するため DAPL UD 経路を有効にするには、この環境変数を設定します。この環境変数が設定されてアルゴリズムが有効になると、特定の DAPL プロバイダーの属性は UD IB 拡張がサポートされることを示します。

I_MPI_DAPL_UD_PROVIDER

IB UD トランスポートと動作する DAPL プロバイダーを定義します。

構文

`I_MPI_DAPL_UD_PROVIDER=<name>`

引数

`<name>`	ロードする DAPL プロバイダーの名前を定義します。
`> 0`	デフォルトの `<nbytes>` 値は、`16456` バイトです。

説明

ロードする DAPL プロバイダーの名前を定義するには、この環境変数を設定します。この名前は、`dat.conf` 設定ファイルでも定義されています。

指定する DAPL プロバイダーが UD IB 拡張をサポートすることを確認してください。

I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD

DAPL UD 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_UD_DIRECT_COPY_THRESHOLD=<nbytes>
```

引数

<nbytes>	DAPL UD 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 <nbytes> 値は、16456 バイトです。

説明

DAPL UD 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 <nbytes> 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、<nbytes> 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。

I_MPI_DAPL_UD_RECV_BUFFER_NUM

サービスメッセージを受信するため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。

構文

```
I_MPI_DAPL_UD_RECV_BUFFER_NUM=<nbuf>
```

引数

<nbuf>	メッセージを受信するバッファーの数を定義します。
> 0	デフォルト値は、16 + n*4 です。n は、MPI ジョブの合計プロセス数です。

説明

サービスメッセージを受信するため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。これらのバッファーは、すべての接続やプロセスペアで共通です。

注意

事前定義バッファーはメモリーを使用しますが、パケットの損失を回避するのに役立ちます。

I_MPI_DAPL_UD_SEND_BUFFER_NUM

メッセージを送信するための内部事前登録 UD 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。

構文

```
I_MPI_DAPL_UD_SEND_BUFFER_NUM=<nbuf>
```

引数

<nbuf>	メッセージを送信するバッファーの数を定義します。
> 0	デフォルト値は、16 + n*4 です。n は、MPI ジョブの合計プロセス数です。

説明

サービスメッセージを送信するため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。これらのバッファーは、すべての接続やプロセスペアで共通です。

注意

事前定義バッファーはメモリーを使用しますが、パケットの損失を回避するのに役立ちます。

I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE

メッセージを送信するための ACK UD 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。

構文

```
I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE=<nbuf>
```

引数

<nbuf>	メッセージを送信するための ACK UD 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。
> 0	デフォルト値は 256 です。

説明

サービスメッセージを送信するための内部事前登録 ACK 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。これらのバッファーは、すべての接続やプロセスペアで共通です。

I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE

メッセージを受信するための ACK UD 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。

構文

```
I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE=<nbuf>
```

引数

<nbuf>	メッセージを受信するための ACK UD 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。
> 0	デフォルト値は、512 + n*4 です。n は、MPI ジョブの合計プロセス数です。

説明

サービスメッセージを受信するための内部事前登録 ACK 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。これらのバッファーは、すべての接続やプロセスペアで共通です。

I_MPI_DAPL_UD_TRANSLATION_CACHE

DAPL UD 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_UD_TRANSLATION_CACHE=<arg>
```

引数

<arg>	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 UD 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。

キャッシュを使用すると、パフォーマンスが向上します。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_UD_TRANSLATION_CACHE_AVL_TREE

DAPL UD 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_UD_TRANSLATION_CACHE_AVL_TREE=<arg>
```

引数

<arg>	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 UD 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_UD_REQ_EVD_SIZE

データ転送操作を送信するための DAPL UD 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_UD_REQ_EVD_SIZE=<size>
```

引数

<size>	イベントキューの長さを定義します。
> 0	デフォルト値は 2,000 です。

説明

DAPL UD データ転送操作 (DTO) の送信完了を処理する 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、<size> 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。プロバイダーには、少なくとも計算された値に等しいキューサイズを提供する必要がありますが、大きなキューサイズを提供することもできます。

I_MPI_DAPL_UD_CONN_EVD_SIZE

接続のための DAPL UD 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_UD_CONN_EVD_SIZE=<size>
```

引数

<size>	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、<size> 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。プロバイダーには、少なくとも計算された値に等しいキューサイズを提供する必要がありますが、大きなキューサイズを提供することもできます。

I_MPI_DAPL_UD_RECV_EVD_SIZE

接続のための DAPL UD 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_UD_RECV_EVD_SIZE=<size>
```

引数

<size>	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、UD と ACK バッファー数に依存します。

説明

DAPL UD データ転送操作 (DTO) の受信完了を処理する 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、<size> 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。プロバイダーには、少なくとも計算された値に等しいキューサイズを提供する必要がありますが、大きなキューサイズを提供することもできます。

I_MPI_DAPL_UD_RNDV_MAX_BLOCK_LEN

DAPL UD 直接コピープロトコルの 1 反復で渡される最大ブロックサイズを定義します。

構文

```
I_MPI_DAPL_UD_RNDV_MAX_BLOCK_LEN=<nbytes>
```

引数

<arg>	DAPL UD 直接コピープロトコルの 1 反復で渡される最大ブロックサイズを定義します。
> 0	デフォルト値は 1,048,576 です。

DAPL UD 直接コピープロトコルの 1 反復で渡される最大ブロックサイズを定義するには、この環境変数を設定します。直接コピープロトコル内のメッセージサイズが、指定された値よりも大きい場合、メッセージは複数のブロックに分割され、複数の操作で渡されます。

I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT

DAPL 直接コピー転送向けの送信バッファーのアライメントを定義します。

構文

```
I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT=<arg>
```

引数

<arg>	送信バッファーのアライメントを定義します。
> 0 かつ 2 の累乗	デフォルト値は 16 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_UD_RNDV_COPY_ALIGNMENT_THRESHOLD

DAPL 直接コピー転送向けの送信バッファーのアライメントのしきい値を定義します。

構文

```
I_MPI_DAPL_UD_RNDV_COPY_ALIGNMENT_THRESHOLD=<nbytes>
```

引数

<nbytes>	送信バッファーのアライメントのしきい値を定義します。
> 0 かつ 2 の累乗	デフォルト値は 32768 です。

DAPL 直接コピー転送向けの送信バッファーのアライメントのしきい値を定義するため、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_UD_RNDV_DYNAMIC_CONNECTION

直接コピープロトコルで使用される DAPL UD 終了ポイントのダイナミック接続確立のアルゴリズムを制御します。

構文

```
I_MPI_DAPL_UD_RNDV_DYNAMIC_CONNECTION=<arg>
```

引数

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

直接コピープロトコルで使用される DAPL UD 終了ポイントのダイナミ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。

ダイナミック接続モードを無効にした場合、すべての接続は MPI 起動時に確立されます。

このモードを有効にすると、アプリケーションが 1 つのプロセスから別のプロセスへデータを渡すため MPI 関数を呼び出した時に、接続が確立されます。

注意

RNDV ダイナミック接続モードでは、データ中の渡されるメッセージサイズは、I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD 環境変数に設定された値よりも大きくなります。

I_MPI_DAPL_UD_EAGER_DYNAMIC_CONNECTION

eager プロトコルで使用される DAPL UD 終了ポイントのダイナミック接続確立のアルゴリズムを制御します。

構文

I_MPI_DAPL_UD_EAGER_DYNAMIC_CONNECTION=<arg>

引数

<arg>	バイナリー・インジケーター。
enable | yes | on | 1	ダイナミック接続を on にします。プロセス数が 64 より多い場合、これがデフォルトになります。
disable | no | off | 0	ダイナミック接続を off にします。

eager プロトコルで使用される DAPL UD 終了ポイントのダイナミ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。eager プロトコルは、内部の事前定義バッファーを介してメッセージを転送する際に使用されます。

このモードを無効にした場合、すべての接続は MPI 起動時に確立されます。

このモードを有効にすると、アプリケーションが 1 つのプロセスから別のプロセスへデータを渡すため MPI 関数を呼び出した時に、接続が確立されます。

注意

eager ダイナミック接続モードでは、データ中の渡されるメッセージサイズは、I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD 環境変数に設定された値と等しいか小さくなります。

I_MPI_DAPL_UD_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_UD_DESIRED_STATIC_CONNECTIONS_NUM=<num_procesess>
```

引数

<num_procesess>	同時に DAPL UD 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	デフォルト値は、200 と等価です。

説明

DAPL UD 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、<num_procesess> 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、<num_procesess> 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_DAPL_UD_RDMA_MIXED

DAPL UD/RDMA の混在通信を制御します。

構文

```
I_MPI_DAPL_UD_RDMA_MIXED =<arg>
```

引数

<arg>	バイナリー・インジケーター。
enable | yes | on | 1	DAPL UD/RDMA の混在通信を有効にします。
disable | no | off | 0	DAPL UD/RDMA の混在通信を無効にします。これは、デフォルト値です。

説明

データを転送するため DAPL UD/RDMA 混在モードを有効にするには、この環境変数を設定します。DAPL UD/RDMA 混在モードでは、短いメッセージは UD トランスポート経由で、そして長いメッセージは RDMA トランスポート経由で渡されます。I_MPI_DAPL_UD_RDMA_MIXED 環境変数を設定し、特定の DAPL プロバイダー属性が UD IB 拡張をサポートすることを示す場合、DAPL UD/RDMA 混在モードが有効になります。

次の I_MPI_DAPL_UD 環境変数のセットも同様に DAPL UD/RDMA 混在モードを制御します。

- I_MPI_DAPL_UD_PROVIDER
- I_MPI_DAPL_UD_EAGER_DYNAMIC_CONNECTION
- I_MPI_DAPL_UD_RNDV_DYNAMIC_CONNECTION
- I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD
- I_MPI_DAPL_UD_RECV_BUFFER_NUM
- I_MPI_DAPL_UD_SEND_BUFFER_NUM
- I_MPI_DAPL_UD_NUMBER_CREDIT_UPDATE
- I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE

- I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE
- I_MPI_DAPL_UD_RESENT_TIMEOUT
- I_MPI_DAPL_UD_MAX_MSG_SIZE
- I_MPI_DAPL_UD_SEND_BUFFER_SIZE
- I_MPI_DAPL_UD_REQ_EVD_SIZE
- I_MPI_DAPL_UD_REQUEST_QUEUE_SIZE
- I_MPI_DAPL_UD_MULTIPLE_EAGER_SEND
- I_MPI_DAPL_UD_NA_SBUF_LIMIT
- I_MPI_DAPL_UD_RECV_EVD_SIZE
- I_MPI_DAPL_UD_CONNECTION_TIMEOUT
- I_MPI_DAPL_UD_PORT
- I_MPI_DAPL_UD_CREATE_CONN_QUAL,
- I_MPI_DAPL_UD_FINALIZE_RETRY_COUNT
- I_MPI_DAPL_UD_FINALIZE_TIMEOUT
- I_MPI_DAPL_UD_TRANSLATION_CACHE
- I_MPI_DAPL_UD_TRANSLATION_CACHE_AVL_TREE
- I_MPI_DAPL_UD_TRANSLATION_CACHE_MAX_ENTRY_NUM
- I_MPI_DAPL_UD_TRANSLATION_CACHE_MAX_MEMORY_SIZE
- I_MPI_DAPL_UD_PKT_LOSS_OPTIMIZATION
- I_MPI_DAPL_UD_DFACTOR
- I_MPI_DAPL_UD_DESIRED_STATIC_CONNECTIONS_NUM
- I_MPI_DAPL_UD_CONN_EVD_SIZE
- I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT
- I_MPI_DAPL_UD_RNDV_COPY_ALIGNMENT_THRESHOLD

DAPL UD/RDMA 混在モード向けに、次の環境変数を設定します。

- I_MPI_DAPL_UD_MAX_RDMA_SIZE
- I_MPI_DAPL_UD_MAX_RDMA_DTOS

I_MPI_DAPL_UD_MAX_RDMA_SIZE

DAPL UD/RDMA 混在モード向けの RDMA プロトコルを介して送信できる最大メッセージサイズを制御します。

構文

```
I_MPI_DAPL_UD_MAX_RDMA_SIZE =<nbytes>
```

引数

<nbytes>	断片化なしで RDMA 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	デフォルトの <nbytes> 値は、4 バイトです。

説明

DAPL UD/RDMA 混在モード向けの RDMA プロトコルを介して送信できる最大メッセージサイズを定義するには、この環境変数を設定します。

メッセージサイズがこの値よりも大きい場合、メッセージは複数の断片に分割され、複数の RDMA 操作で送信されます。

I_MPI_DAPL_UD_MAX_RDMA_DTOS

DAPL UD/RDMA 混在モード向けに接続ごとの完了しない RDMA 操作の最大数を制御します。

構文

```
I_MPI_DAPL_UD_MAX_RDMA_DTOS=<arg>
```

引数

`<arg>`	接続ごとの RDMA 操作の最大数を定義します。
`> 0`	デフォルト `<arg>` の値は 8 です。

説明

DAPL UD/RDMA 混在モード向けに接続ごとの完了しない RDMA 操作の最大数を定義するには、この環境変数を設定します。

3.3.5. 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=<arg>
```

引数

`<arg>`	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: `ibor 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。
`<interface_name>`	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。
`<network_address>`	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。
`<network_address/ <netmask>`	ネットワーク・アドレス。`<netmask>` 値には、ネットマスクのレングスを指定します。
`<list of interfaces>`	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 ネットマスクを暗示します。

  ```
  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=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=<nbytes>
```

引数

<nbytes>	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 <nbytes> 値は、使用する Linux* システムの TCP ソケットのデフォルト・バッファー・サイズの値と等価です。

説明

TCP ソケットバッファーのサイズを手動で定義するには、この環境変数を設定します。TCP ソケットのバッファーサイズは、Linux* システム上の既存の TCP の設定によって制限されます。

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 10 Gb 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)。

I_MPI_TCP_POLLING_MODE

この環境変数を設定して、ポーリングのモードを定義します。

構文

```
I_MPI_TCP_POLLING_MODE=<mode>
```

引数

<mode>	ポーリングモードを指定します。
poll	poll() 関数をベースにしたポーリングモード。これは、デフォルト値です。
epoll[:edge]	エッジトリガー・インターフェイスによる epoll() 関数に基づくポーリングモード。
epoll:level	レベルトリガー・インターフェイスによる epoll() 関数に基づくポーリングモード。

tcp ファブリックのポーリングモードを選択するには、この環境変数を設定します。

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_TCP_POLLING_MODE 環境変数を使用します。経験則に従って、最良のポーリングモードを選択できます。最良のモードは、アプリケーションとプロセス数に依存します。epoll ポーリングモードは、次の状況に適しています。

- プロセス数が非常に多い
- クライアントサーバー型のアプリケーション
- MPI_ANY_SOURCE タグマッチング

3.3.6. TMI ネットワーク・ファブリック制御

I_MPI_TMI_LIBRARY

特定の TMI ライブラリーを選択します。

構文

```
I_MPI_TMI_LIBRARY=<library>
```

引数

<library>	デフォルトの libtmi.so の代わりに使用する TMI ライブラリーを指定します。

説明

TMI 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 TMI ライブラリーを指定します。

I_MPI_TMI_PROVIDER

ロードする TMI プロバイダーの名前を定義します。

構文

```
I_MPI_TMI_PROVIDER=<name>
```

引数

<name>	ロードする TMI プロバイダーの名前を定義します。

説明

ロードする TMI プロバイダーの名前を定義するには、この環境変数を設定します。この名前は、dat.conf 設定ファイルでも定義されている必要があります。

I_MPI_TMI_NBITS_RANK

TMI レベルで MPI ランクの値を格納する予約ビット数を定義します。

構文

I_MPI_TMI_NBITS_RANK=<num_bits>

引数

<num_bits>	MPI ランクを保存する予約ビット数。
<=32 および > 0	デフォルト値は 24 です。

説明

I_MPI_TMI_NBITS_RANK の値は、TMI レベルで参照および区分可能な MPI ランク数を指定します。この環境変数のデフォルト値 I_MPI_TMI_NBITS_RANK=24 を指定すると、ジョブを実行可能なランク数は、2^{24}=16M ランクとなります。

注意

I_MPI_TMI_NBITS_RANK の値は、MPI_TAG_UB に関連します。I_MPI_TMI_NBITS_RANK に大きな値を指定するほど、MPI_TAG_UB でサポートされるタグ値は少なくなります。I_MPI_TMI_NBITS_RANK に小さな値を指定するほど、MPI_TAG_UB でサポートされるタグ値は多くなります。通常の MPI アプリケーションは、サポートされる最大タグ値を常に照会する必要があります。

I_MPI_TMI_DSEND

TMI netmod における直接送信機能を制御します。

構文

I_MPI_TMI_DSEND=<arg>

引数

<arg>	バイナリー・インジケーター。
enable | yes | on | 1	直接送信を有効にします。これは、デフォルト値です。
disable | no | off | 0	直接送信を無効にします。

説明

直接送信機能は、ブロック化された MPI_Send 呼び出しにのみ使用します。直接送信機能を使用する前に、シングルスレッドのアプリケーションでネットワーク・ファブリックに TMI が選択されていること (I_MPI_FABRICS=tmi が設定済みであること) を確認してください。

注意

直接送信機能は、TMI バージョン 1.1 以降でのみサポートされます。それ以前のバージョンの TMI を使用する場合、I_MPI_TMI_DSEND に設定された値は無視されます。

I_MPI_TMI_DRECV

TMI ファブリックにおける直接受信機能を制御します。

構文

```
I_MPI_TMI_DRECV=<arg>
```

引数

<arg>	バイナリー・インジケーター。
enable | yes | on | 1	直接受信を有効にします。これは、デフォルト値です。
disable | no | off | 0	直接受信を無効にします。

説明

直接受信機能は、ブロック化された MPI_Recv 呼び出しにのみ使用します。直接送信機能を使用する前に、シングルスレッドのアプリケーションでネットワーク・ファブリックに TMI が選択されていること (I_MPI_FABRICS=tmi が設定済みであること) を確認してください。

3.3.7. OFA ネットワーク・ファブリック制御

I_MPI_OFA_NUM_ADAPTERS

接続アダプターの数を設定します。

構文

```
I_MPI_OFA_NUM_ADAPTERS=<arg>
```

引数

<arg>	使用するアダプターの最大数を定義します。
>0	指定されたアダプター数を使用します。デフォルト値は 1 です。

説明

使用するアダプターの数を設定します。利用可能なアダプター数よりも大きな値が指定された場合、すべての利用可能なアダプターが使用されます。

I_MPI_OFA_ADAPTER_NAME

使用するアダプターの名前を設定します。

構文

```
I_MPI_OFA_ADAPTER_NAME=<arg>
```

引数

<arg>	アダプターの名前を定義します。
Name	指定されたアダプターを使用します。デフォルトでは、すべてのアダプターが使用できます。

説明

使用するアダプターの名前を設定します。指定されたアダプターが存在しない場合、ライブラリーはエラーを返します。これは、I_MPI_OFA_NUM_ADAPTERS=1 のときにのみ有効です。

I_MPI_OFA_NUM_PORTS

各アダプターが使用するポート数を設定します。

構文

I_MPI_OFA_NUM_PORTS=<arg>

引数

<arg>	各アダプター上で使用されるポート数を定義します。
> 0	指定されたポート数を使用します。デフォルト値は 1 です。

説明

各アダプターが使用するポート数を設定します。利用可能なポート数よりも大きな値が指定された場合、すべての利用可能なポートが使用されます。

I_MPI_OFA_NUM_RDMA_CONNECTIONS

rdma 交換プロトコルで使用できる接続の最大数を設定します。

構文

I_MPI_OFA_NUM_RDMA_CONNECTIONS=<num_conn>

引数

<num_conn>	rdma 交換プロトコルで使用できる接続の最大数を定義します。
>= 0	rdma 交換プロトコルで使用する接続の最大数を指定して作成します。その他のプロセスは、送信/受信交換プロトコルを使用します。
-1	log2(プロセス数) rdma 接続を作成します。
>= プロセス数	すべてのプロセスに rdma 接続を作成します。これは、デフォルト値です。

説明

2 つのプロセス間では 2 つのプロトコルが使用できます: 送信/受信と rdma。この環境変数は、rdma プロトコルで使用する接続の最大数を指定します。

RDMA プロトコルは高速ですが多くのリソースを必要とします。大規模なアプリケーション向けに、頻繁にデータ交換を行うプロセスだけが RDMA プロトコルを利用できるよう、RDMA プロトコルを使用する接続数を制限することができます。

I_MPI_OFA_SWITCHING_TO_RDMA

プロセスが接続を RDMA 交換プロトコルに切り替える前に受信すべきメッセージ数を設定します。

構文

```
I_MPI_OFA_SWITCHING_TO_RDMA=<number>
```

引数

<number>	プロセスが接続を RDMA 交換プロトコルに切り替える前に受信すべきメッセージ数を定義します。
>= 0	プロセスが <number> のメッセージを受信すると、rdma プロトコルの利用を開始します。

説明

特定のプロセスからの受信メッセージをカウントします。接続が指定された数に達した場合、そのプロセスと交換を行うため rdma プロトコルへの切り替えを試みます。I_MPI_OFA_NUM_RDMA_CONNECTIONS に定義される RDMA 交換プロトコルを使用する接続の最大数に達した場合は、接続は rmda プロトコルに切り替わりません。

I_MPI_OFA_RAIL_SCHEDULER

短いメッセージ向けのレールを選択する方式を設定します。

構文

```
I_MPI_OFA_RAIL_SCHEDULER=<arg>
```

引数

<arg>	モードセレクター。
ROUND_ROBIN	次回、次のレールを使用します。
FIRST_RAIL	短いメッセージには常に最初のレールを使用します。
PROCESS_BIND	プロセス向けに常に特定のレールを使用します。

説明

短いメッセージ向けのレールを選択する方式を設定します。アルゴリズムは次のスキームに従って選択されます。

- ROUND_ROBIN モード、最初のメッセージは最初のレールを使用して送信され、次のメッセージは 2 番目のレールを使用して送信されます。
- FIRST_RAIL モード、短いメッセージには常に最初のレールが使用されます。
- PROCESS_BIND モード、プロセスの最小ランクは最初のレールを使用し、次は 2 番目のレールを使用します。

I_MPI_OFA_TRANSLATION_CACHE

メモリー登録キャッシュを on/off にします。

構文

```
I_MPI_OFA_TRANSLATION_CACHE=<arg>
```

引数

<arg>	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

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_OFA_TRANSLATION_CACHE_AVL_TREE

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_OFA_TRANSLATION_CACHE_AVL_TREE=<arg>

引数

<arg>	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、OFA 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_OFA_DYNAMIC_QPS

ライブラリーがキューのペア (QP) を動的に作成するのを制御します。

構文

I_MPI_OFA_DYNAMIC_QPS=<arg>

引数

<arg>	バイナリー・インジケーター。
enable | yes | on | 1	QP をダイナミックに作成します。プロセス数が 2000 以上のデフォルト値です。
disable | no | off | 0	初期ステージですべての QP を作成します。プロセス数が 2000 未満のデフォルト値です。

説明

QP のダイナミックな作成を on/off にするため、この環境変数を使用します。

I_MPI_OFA_PACKET_SIZE

送信されるパケットのサイズを設定します。

構文

```
I_MPI_OFA_PACKET_SIZE=<arg>
```

引数

<arg>	パケットのサイズをバイト単位で定義します。
> 0	指定されたパケットサイズを使用します。デフォルト値は 8192 です。

説明

バイト単位でパケットサイズを設定します。負の値が指定された場合、サイズは 8 に設定されます。

I_MPI_OFA_LIBRARY

使用する OFA ライブラリーの名前を設定します。

構文

```
I_MPI_OFA_LIBRARY=<arg>
```

引数

<arg>	ロードする OFA ライブラリーの名前を定義します。
Name	指定されたライブラリーを使用します。デフォルトは、libibverbs.so です。

説明

使用する InfiniBand* (IB*) ライブラリーの名前を設定します。指定された名前が存在しない場合、ライブラリーはエラーを返します。

I_MPI_OFA_NONSWITCH_CONF

ポート接続に非標準のテンプレートを定義します。

構文

```
I_MPI_OFA_NONSWITCH_CONF=<arg>
```

引数

<arg>	ポート接続にテンプレートを定義します。
Name	指定されたテンプレートを使用します。

説明

クラスター内のノードは、$port_i$ のノードがその他のすべてのノードの $port_i$ にアクセスできるよう、常に接続されています。ポートがこの方法で接続されていない場合、この環境変数を使用します。以下にテンプレートの例を示します

```
host1@port11#port12#...#host2@port21#port22....
```

$port_i^j$ は、$host_i$ から $host_j$ ホストへ送信するために使用するポートを定義します。

例:
node1@1#1#2#node2@2#1#1#node3@1#2#1#

このサンプルは、次の設定を指定します。

- node1 の port1 は node2 の port2 に接続されます。
- node1 の port2 は node3 の port1 に接続されます。
- node2 の port1 は node3 の port2 に接続されます。
- node2 の port2 は node2 の port1 に接続されます。
- node3 の port1 は node1 の port2 に接続されます。
- node3 の port2 は node2 の port1 に接続されます。

port1 は、常に自身の通信に使用されます (ループバック)。

OFA* デバイスのフェイルオーバーのサポート

インテル® MPI ライブラリーは、ハードウェアの問題を検出するため、次のイベントを認識します。

- IBV_EVENT_QP_FATAL: QP でエラーが発生し、エラー状態へ移行しました。
- IBV_EVENT_QP_REQ_ERR: 不正な要求によるローカル・ワーク・キューのエラー。
- IBV_EVENT_QP_ACCESS_ERR: ローカルアクセス違反エラー。
- IBV_EVENT_PATH_MIG_ERR: 接続が代替パスへの移行に失敗。
- IBV_EVENT_CQ_ERR: CQ がエラー状態 (CQ オーバーラン)。
- IBV_EVENT_SRQ_ERR: SRQ でエラーが発生。
- IBV_EVENT_PORT_ERR: ポートにリンクできません。
- IBV_EVENT_DEVICE_FATAL: CA が致命的状態です。

インテル® MPI ライブラリーは、これらの問題を検出すると、ポートの利用またはアダプター自体の利用を停止します。アプリケーションがマルチレール・モードで実行されている場合、利用可能なポートやアダプターを介して通信を継続します。利用可能なポートやアダプターがない場合、アプリケーションは異常終了します。

インテル® MPI ライブラリーは、また次のイベントを認識します。

- IBV_EVENT_PORT_ACTIVE: リンクはポート上でアクティブ

このイベントは、ポートが再び利用可能で通信を確立可能なことを意味します。

3.3.8. OFI* ネットワーク・ファブリック制御

I_MPI_OFI_LIBRARY

特定の OpenFabrics Interfaces* (OFI*) ライブラリーを選択します。

構文

I_MPI_DAT_LIBRARY=<library>

引数

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

説明

OFI 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 OFI ライブラリーを指定します。

I_MPI_OFI_PROVIDER

ロードする DAPL プロバイダーの名前を定義します。

構文

```
I_MPI_OFI_PROVIDER=<name>
```

引数

`<name>`	ロードする OFI プロバイダーの名前を定義します。

説明

ロードする OFI プロバイダーの名前を定義するには、この環境変数を設定します。この変数を設定していない場合、プロバイダーは自動的に OFI ライブラリーを選択します。`I_MPI_OFI_PROVIDER_DUMP` 環境変数を使用して、利用可能なすべてのプロバイダーをチェックできます。

I_MPI_OFI_PROVIDER_DUMP

すべての OFI プロバイダーと OFI ライブラリーからそれらの属性に関する情報を表示する機能を制御します。

構文

```
I_MPI_OFI_PROVIDER_DUMP=<arg>
```

引数

`<arg>`	バイナリー・インジケーター。
`enable | yes | on | 1`	すべての OFI プロバイダーのリストと OFI ライブラリーからそれらの属性を表示します。
`disable | no | off | 0`	何もしません。これは、デフォルト値です。

説明

すべての OFI プロバイダーと OFI ライブラリーからそれらの属性に関する情報を表示する機能を制御するには、この環境変数を設定します。

3.4. 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。

3.4.1. 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>="<algid>[:<conditions>][;<algid>:<conditions>[...]]"
```

引数

<algid>	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。

<conditions>	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> 以上 <q> 以下のプロセス数。

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 3.4-1 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 3.4-2 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	MPI_IN_PLACE が使用される場合 recv_count*recv_type_size 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	MPI_IN_PLACE が使用される場合 send_count*send_type_size それ以外は recv_count*recv_type_size
MPI_Scatterv	n/a

例

MPI_Reduce 操作向けの第二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_Reduce_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=<block_size>|<algid>:<block_size>[,<algid>:<block_size>[...]]
```

引数

<algid>	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。
<block_size>	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。

<algid> 値が設定されていなければ、<block_size> 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=<block_size>|<algid>:<block_size>[,<algid>:<block_size>[...]]
```

引数

<algid>	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。
<block_size>	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。

<algid> 値が設定されていなければ、<block_size> 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=<radix>
```

引数

<radix>	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=<radix>
```

引数

<radix>	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=<radix>
```

引数

<radix>	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=<radix>
```

引数

<radix>	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=<radix>
```

引数

<radix>	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=<radix>
```

引数

<radix>	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=<radix>
```

引数

<radix>	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=<radix>
```

引数

<radix>	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=<radix>
```

引数

<radix>	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=<radix>
```

引数

<radix>	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。

3.4.2. 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

適切な集団アルゴリズムを選択する際のデフォルトのライブラリーの動作を制御します。

構文

I_MPI_FAST_COLLECTIVES=<arg>

引数

<arg>	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=<nproc>

引数

<nproc>	MPI_Bcast 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_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=<nbytes1,nbytes2>

引数

<nbytes1,nbytes2>	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 nbytes2 >= nbytes1	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 3.4-1 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 <nbytes1> 未満か、操作するプロセス数が <nproc> 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 <nbytes1> 以上 <nbytes2> 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=<nproc>

引数

<nproc>	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=<nbytes1,nbytes2>

引数

<nbytes1,nbytes2>	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 nbytes2 >= nbytes1	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 3.4-1 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 <nbytes1> 以上で、操作するプロセス数が <nproc> 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 <nbytes1> より大きく <nbytes2> 以下の場合、またはメッセージサイズが <nbytes2> 未満で操作するプロセス数が <nproc> 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=<nbytes1,nbytes2>

引数

<nbytes1,nbytes2>	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 nbytes2 >= nbytes1	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 3.4-1 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 <nbytes2> 未満で、操作するプロセス数が 2 の累乗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 <nbytes1> 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=<nbytes>

引数

<nbytes>	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 3.4-1 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 <nbytes> 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=<nbytes1,nbytes2>

引数

<nbytes>	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

以下のスキームに従って、利用可能な 3 つの MPI_Reduce_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 3.4-1 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、<nbytes2> 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 <nbytes2> 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 <nbytes1> 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=<nbytes>

引数

<nbytes>	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 3.4-1 をご覧ください)。

メッセージサイズが <nbytes> 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=<nbytes>

引数

<nbytes>	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 3.4-1 をご覧ください)。

メッセージサイズが <nbytes> 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。

3.5. 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。

- タイマー制御
- 互換性制御
- ダイナミック・プロセスのサポート
- フォールトトレラント
- 統計収集モード
- ILP64 サポート
- ユニファイド・メモリー管理
- ファイルシステムのサポート
- マルチスレッド化された memcpy のサポート

3.5.1. タイマー制御

I_MPI_TIMER_KIND

MPI_Wtime と MPI_Wtick 呼び出して使用されるタイマーを選択します。

構文

I_MPI_TIMER_KIND=<timername>

引数

<timername>	タイマーのタイプを定義します。
gettimeofday	この設定を選択した場合、MPI_Wtime と MPI_Wtick 関数は、gettimeofday(2) 関数を介して動作します。これは、デフォルト値です。
rdtsc	この設定を選択した場合、MPI_Wtime と MPI_Wtick 関数は、高精度の RDTSC タイマーを介して動作します。

説明

通常のまたは RDTSC タイマーのいずれかを選択するには、この環境変数を設定します。

デフォルトの gettimeofday(2) タイマーの解像度は、一部のプラットフォームでは十分でない可能性があります。

3.5.2. 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=<value>

引数

<value>	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、ライブラリーは 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 環境変数に 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 環境変数に 3 に設定します。

3.5.3. 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

生成されたプロセスのデフォルトの配置は、ラウンドロビン・スケジュールを使用します。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。

```
$ mpirun -n 1 -gwdir <path_to_executable> -genv I_MPI_FABRICS shm:tcp <spawn_app>
```

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。

mpd.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
```

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。

```
$ mpirun -n 1 -genv I_MPI_FABRICS shm:dapl <server_app> > <port_name>
```

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。

```
$ mpirun -n 1 -genv I_MPI_FABRICS shm:dapl <client_app> < <port_name>
```

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、サーバーホスト上で次のコマンドラインを実行します。

```
$ mpirun -n 1 -genv I_MPI_FABRICS shm:ofa <join_server_app> < <port_number>
$ mpirun -n 1 -genv I_MPI_FABRICS shm:ofa <join_client_app> < <port_number>
```

3.5.4. フォールトトレラント

インテル® MPI ライブラリーは、MPI アプリケーションでフォールトトレラントのサポートを有効にする特別な機能を提供します。MPI 標準は、MPI アプリケーションの 1 つもしくはいくつかのプロセスが異常終了した際の MPI 実行の動作を定義していません。デフォルトでは、インテル® MPI ライブラリーは、プロセスが停止するとアプリケーション全体を異常終了します。

環境変数 I_MPI_FAULT_CONTINUE を設定すると、この動作を変更できます。

例:

```
$ mpirun -env I_MPI_FAULT_CONTINUEon -n 2 ./test
```

MPI プロセスに問題が発生した場合、アプリケーションが処理を継続するには、問題は次の要件を満たす必要があります。

- アプリケーションはエラーハンドラー MPI_ERRORS_RETURN をコミュニケーター MPI_COMM_WORLD に設定する (すべての新しいコミュニケーターはそこからエラーハンドラーを継承)。
- アプリケーションは、マスター・スレーブ・モデルを採用している。この場合、アプリケーションはマスターが完了した、もしくは反応しないときにのみ停止します。
- アプリケーションは、マスターとスレーブ間でポイントツーポイント通信のみを使用します。この場合、スレーブ間での通信や MPI 集団通信を行いません。
- ランクの異なる通信を回避するため、アプリケーションのスレーブランクの特定のエラーとポイントツーポイント操作の MPI エラーコードを処理します。スレーブランクは、ブロック/非ブロック送信、受信、調査、およびテストを行うことができます。
- 任意の通信操作をサブセット通信システム上で使用できます。集団操作でエラーが発生した場合、このコミュニケーター内部の通信はすべて禁止されます。
- マスターの失敗はジョブの停止を意味します。

- フォールトトレラント機能は、スポーンされたプロセスでは利用できません。

環境変数

I_MPI_FAULT_CONTINUE

フォールトトレラントのアプリケーションサポートを on/off にします。

構文

`I_MPI_FAULT_CONTINUE=<arg>`

引数

`<arg>`	バイナリー・インジケーター。
`enable |yes |on|1`	フォールトトレラントのアプリケーションサポートを on にします。
`disable| no|off| 0`	フォールトトレラントのアプリケーションサポートを off にします。これは、デフォルト値です。

説明

フォールトトレラント型のアプリケーションをサポートするには、この環境変数を設定します。

使用モデル

アプリケーションは MPI_ERRORS_RETURN エラーハンドラーをセットし、各通信呼び出し後にエラーコードをチェックします。通信呼び出しが MPI_SUCCESS を返さない場合、送信先のプロセスが到達不能とマークされ、通信から除外します。次に例を示します。

```
if(live_ranks[rank]) {
    mpi_err = MPI_Send(buf, count, dtype, rank, tag, MPI_COMM_WORLD);
    if(mpi_err != MPI_SUCCESS) {
        live_ranks[rank] = 0;
    }
}
```

非ブロッキング通信の場合、待機もしくはテスト中にエラーが表示されることがあります。

3.5.5. 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。

環境変数を利用するほかにも、MPI Performance Snapshot の `-mps` オプションを使用して、ネイティブ統計情報を収集できます。次に例を示します。

```
$ mpirun -mps –n 2 ./myApp
```

詳細は、-mps の説明をご覧ください。

I_MPI_STATS

統計収集を制御します。既存の値に加え、`I_MPI_STATS` 環境変数の値を拡張します。

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されます。
`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を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 `m` まで (`n` と `m` を含む) の統計を出力します。

`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="<subsystem>[:<ops>][;<subsystem>[:<ops>][...]]"
```

引数

<subsystem>	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=<msg>[@<proc>][,<msg>[@<proc>]]...
```

引数

<msg>	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>。

<proc>	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=<name>
```

引数

<name>	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、またオプションの領域機能をサポートします。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けされた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, <name>) 呼び出しで開かれた領域は、MPI_Pcontrol(-1, <name>) 呼び出しで閉じられます。<name> は、NULL で終わる文字列の領域名です。<name> 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_<name>.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=<level>

引数

<level>	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=<name>

引数

<name>	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="<subset>[;<subset>[;…]]」

引数

<subset>	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 3.5-1 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	**one2all** MPI_Bcast MPI_Scatter MPI_Scatterv MPI_Ibcast MPI_Iscatter MPI_Iscatterv **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_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
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_Win_fence
MPI_Win_free
MPI_Win_get_group
MPI_Win_lock
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_Comm_connect
MPI_Comm_disconnect
MPI_Comm_get_parent
MPI_Comm_join
MPI_Comm_spawn
MPI_Comm_spawn_multiple
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

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_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

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
MPI_Type_get_extent
MPI_Type_get_true_extent
MPI_Type_indexed
MPI_Type_size
MPI_Type_vector
MPI_Unpack_external
MPI_Unpack

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	

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=<percentage>

引数

<percentage>	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 ./a.out
```

統計出力:

```
################################################################################
#
# command : ./a.out (completed)
# host : svlmpihead01/x86_64_Linux mpi_tasks : 4 on 1 nodes
# start : 05/25/11/05:44:13 wallclock : 0.092012 sec
# stop : 05/25/11/05:44:13 %comm : 98.94
# gbytes : 0.00000e+00 total gflop/sec : NA
#
################################################################################
```

コマンド:

```
mpiexec -n 4 -env I_MPI_STATS ipm ./a.out
```

統計出力:

```
################################################################################
#
# command : ./a.out (completed)
# host : svlmpihead01/x86_64_Linux mpi_tasks : 4 on 1 nodes
# start : 05/25/11/05:44:13 wallclock : 0.092012 sec
# stop : 05/25/11/05:44:13 %comm : 98.94
# gbytes : 0.00000e+00 total gflop/sec : NA
#
################################################################################
# region : * [ntasks] = 4
#
# [total] <avg> min max
# entries 4 1 1 1
# wallclock 0.332877 0.0832192 0.0732641 0.0920119
# user 0.047992 0.011998 0.006999 0.019996
# system 0.013997 0.00349925 0.002999 0.004
# mpi 0.329348 0.082337 0.0723064 0.0912335
# %comm 98.9398 98.6928 99.154
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Init 0.236192 4 71.71 70.95
# MPI_Reduce 0.0608737 8000 18.48 18.29
# MPI_Barrier 0.027415 800 8.32 8.24
# MPI_Recv 0.00483489 1 1.47 1.45
# MPI_Send 1.50204e-05 1 0.00 0.00
# MPI_Wtime 1.21593e-05 8 0.00 0.00
# MPI_Finalize 3.33786e-06 4 0.00 0.00
# MPI_Comm_rank 1.90735e-06 4 0.00 0.00
# MPI_TOTAL 0.329348 8822 100.00 98.94
################################################################################
# region : reduce [ntasks] = 4
#
# [total] <avg> min max
# entries 8 2 2 2
```

```
# wallclock 0.0638561 0.015964 0.00714302 0.0238571
# user 0.034994 0.0087485 0.003999 0.015997
# system 0.003999 0.00099975 0 0.002999
# mpi 0.0608799 0.01522 0.00633883 0.0231845
# %comm 95.3392 88.7417 97.1808
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Reduce 0.0608737 8000 99.99 95.33
# MPI_Finalize 3.33786e-06 4 0.01 0.01
# MPI_Wtime 2.86102e-06 4 0.00 0.00
# MPI_TOTAL 0.0608799 8008 100.00 95.34
################################################################################
# region : send [ntasks] = 4
#
# [total] <avg> min max
# entries 1 0 0 1
# wallclock 2.89876e-05 7.24691e-06 1e-06 2.59876e-05
# user 0 0 0 0
# system 0 0 0 0
# mpi 1.50204e-05 3.75509e-06 0 1.50204e-05
# %comm 51.8165 0 57.7982
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Send 1.50204e-05 1 100.00 51.82
################################################################################
# region : ipm_noregion [ntasks] = 4
#
# [total] <avg> min max
# entries 13 3 3 4
# wallclock 0.26898 0.0672451 0.0661182 0.068152
# user 0.012998 0.0032495 0.001 0.004999
# system 0.009998 0.0024995 0 0.004
# mpi 0.268453 0.0671132 0.0659676 0.068049
# %comm 99.8039 99.7721 99.8489
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Init 0.236192 4 87.98 87.81
# MPI_Barrier 0.027415 800 10.21 10.19
# MPI_Recv 0.00483489 1 1.80 1.80
# MPI_Wtime 9.29832e-06 4 0.00 0.00
# MPI_Comm_rank 1.90735e-06 4 0.00 0.00
# MPI_TOTAL 0.268453 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取された場合、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。

3.5.6. 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。

- 分割コンパイル時、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
  ```

- ILP64 インターフェイスを使用するには、mpirun -ilp64 オプションを使用します。

  例:
  ```
  $mpirun -ilp64 -n 2 ./myprog
  ```

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 および 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 ライブラリーの 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。

3.5.7. 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。

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
}
```

3.5.8. ファイルシステムのサポート

インテル® MPI ライブラリーは、次のファイルシステムをサポートするため、ロード可能な共有モジュールを提供しています。

- Panasas* ActiveScale* File System (PanFS)
- Parallel Virtual File System*, Version 2 (Pvfs2)
- Lustre* File System
- IBM* General Parallel File System* (GPFS*)

Parallel File System のサポートを有効にするには、I_MPI_EXTRA_FILESYSTEM 環境変数を on に設定します。特定のファイルシステムのネイティブサポートを要求するには、I_MPI_EXTRA_FILESYSTEM_LIST 環境変数を設定します。例えば、Panasas* ActiveScale* File System のネイティブサポートを要求するには、次のように設定します。

```
$ mpiexec -env I_MPI_EXTRA_FILESYSTEM on \
-env I_MPI_EXTRA_FILESYSTEM_LIST panfs -n 2 ./test
```

環境変数

I_MPI_EXTRA_FILESYSTEM

ネイティブ parallel file system のサポートを on/off にします。

構文

```
I_MPI_EXTRA_FILESYSTEM=<arg>
```

引数

<arg>	バイナリー・インジケーター。
enable| yes|on| 1	parallel file system 向けのネイティブサポートを on にします。
disable| no|off| 0	parallel file system 向けのネイティブサポートを off にします。これは、デフォルト値です。

説明

Parallel File System のサポートを有効にするには、この環境変数を設定します。特定のファイルシステムのネイティブサポートを要求するには、I_MPI_EXTRA_FILESYSTEM_LIST 環境変数を設定する必要があります。

I_MPI_EXTRA_FILESYSTEM_LIST

特定のファイルシステムのサポートを選択します。

構文

```
I_MPI_EXTRA_FILESYSTEM_LIST=<fs>[, <fs>, ..., <fs>]
```

引数

<fs>	ターゲットのファイルシステムを定義します。
panfs	Panasas* ActiveScale* File System。
pvfs2	Parallel Virtual File System, Version 2。

lustre	Lustre* File System。
gpfs	IBM* General Parallel File System* (GPFS*)。

説明

特定のファイルシステムのネイティブサポートを要求するには、この環境変数を設定します。この環境変数は、I_MPI_EXTRA_FYLESYSTEM が有効なときにのみ効果があります。インテル® MPI ライブラリーは、I_MPI_EXTRA_FILESYSTEM_LIST で指定されるファイルシステムをサポートするため、共有モジュールをロードしようとします。

3.5.9. マルチスレッド化された memcpy のサポート

ここでは、インテル® Xeon Phi™ コプロセッサー向けのインテル® MPI ライブラリーでマルチスレッド版の memcpy を使用する方法を説明します。ある種のアプリケーション向けに共有メモリーを介して高いバンド幅を達成する、この実験的な機能を使用できます。

I_MPI_MT_MEMCPY

マルチスレッド化された memcpy の使用法を制御します。

構文

I_MPI_MT_MEMCPY=<value>

引数

<value>	マルチスレッド化された memcpy の使用法を制御します。
enable | yes | on | 1	シングルスレッド版のインテル® MPI ライブラリーでマルチスレッド版の memcpy の利用を有効にします (MPI_THREAD_SINGLE)。この設定は、スレッドセーフ版のインテル® MPI ライブラリーでは無視されます。
disable | no | off | 0	マルチスレッド化された memcpy の使用を無効にします。これは、デフォルト値です。

説明

ノード内通信向けにマルチスレッド版の memcpy を使用するかどうかを制御するには、この環境変数を設定します。

I_MPI_MT_MEMCPY_NUM_THREADS

マルチスレド版の memcpy を行うために実行されるスレッド数を変更します。

構文

I_MPI_MT_MEMCPY_NUM_THREADS=<num>

引数

<num>	マルチスレド版の memcpy を行うために実行されるスレッド数。
>0	デフォルト値は、8 または MPI プロセスがピニングするドメインの物理コア数のどちらか小さい値です。

説明

各 MPI ランクごとに memcpy を実行するスレッド数を設定するには、この環境変数を使用します。値を 1 に設定すると、I_MPI_MT_MEMCPY=disable にするのと等価です。

I_MPI_MT_MEMCPY_THRESHOLD

マルチスレッド化された memcpy を使用するしきい値を変更します。

構文

I_MPI_MT_MEMCPY_THRESHOLD=<nbytes>

引数

<nbytes>	マルチスレッド化された memcpy を使用するしきい値をバイト単位で定義します。
>0	デフォルト値は 32768 です。

説明

マルチスレッド版の memcpy の使用を制御するには、この環境変数を設定します。しきい値が、共有メモリーバッファーのサイズより大きい場合 (例えば、「I_MPI_SHM_LMT_BUFFER_SIZE」もしくは「I_MPI_SSHM_BUFFER_SIZE」を参照)、マルチスレッド版の memcpy が利用されることはありません。マルチスレッド版の memcpy の利用は、次のスキームで選択されます。

- バッファーが、<nbytes> より短い場合、シリアル版の memcpy が使用されます。このアプローチは、ショートバッファーとミディアムバッファーでは高速です。
- バッファーが、<nbytes> より長い場合、マルチスレッド版の memcpy が使用されます。このアプローチは、ラージバッファーでは高速です。

I_MPI_MT_MEMCPY_SPIN_COUNT

スピンカウント値を制御します。

構文

I_MPI_MT_MEMCPY_SPIN_COUNT=<scount>

引数

<scount>	スレッドが、データコピーを待機する際にスリープする前のスピンカウントを定義します
>0	デフォルト値は、100000 と等価です。最大値は、2147483647 と等価です。

説明

スレッドによってデータがコピーされるのを待機するため、ループのスピンカウントを設定します。上限値を越え、コピーするデータがない場合、スレッドはスリープします。

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_MT_MEMCPY_SPIN_COUNT 環境変数を使用します。<scount> の最適な値の選択は、経験に依存します。それは、計算環境やアプリケーションに依存します。

4. 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。

5. 索引

$

$HOME/.mpd.conf, 91

{

-{cc,cxx,fc,f77,f90}=<compiler>, 10

A

-a <alias>, 79

B

-binding, 32
-bootstrap <bootstrap server>, 31
-bootstrap jmi, 32
-bootstrap persist, 32
-bootstrap-exec <bootstrap server>, 32
-bootstrap-exec-args <args>, 32
-branch-count <num>, 23

C

-check_mpi, 9
-check_mpi [<checking_library>], 22, 78
-ckpoint <switch>, 51
-ckpoint-interval <sec>, 52
-ckpointlib <lib>, 53
-ckpoint-logfile <file>, 53
-ckpoint-num <N>, 52
-ckpoint-prefix <dir>, 53
-ckpoint-preserve <N>, 52
-ckpoint-tmp-prefix <dir>, 53
-cleanup, 30
-compchk, 11
-config=<*name*>, 8
-configfile <filename>, 22, 77
cpuinfo, 95

D

-dapl, 37, 74
-DAPL, 37, 74
-demux <mode>, 28
-disable-x, 28
-dynamic_log, 9

E

-ecfn <filename>, 80
-echo, 10
-enable-x, 28
-env <ENVVAR> <value>, 35, 80
-envall, 35, 80
-envexcl <list of env var names>, 80
-envlist <list of env var names>, 36, 80
-envnone, 35, 80
-envuser, 80

F

-f <hostfile>, 20
-fast, 10

G

-g, 9
-g<l-option>, 77
-gcc-version=<nnn>, 11
-gdb, 23, 79
-gdba <jobid>, 79
-gdba <pid>, 24
-genv <ENVVAR> <value>, 21, 77
-genvall, 21, 77
-genvlist <list of genv var names>, 21
-genvnone, 21, 77
-genvuser, 77
-grr <# of processes>, 21, 76
-gtool, 24
-gtoolfile <gtool_config_file>, 27

H

-h, 75
-help, 75
--help, 75
-host <nodename>, 36, 81
-hostfile <hostfile>, 20
-hostos <host OS>, 36
-hosts <nodelist>, 28
Hydra, 32
Hydra, 16, 19, 30, 31, 46, 48, 49, 54, 66, 110, 117

I

I_MPI_ HYDRA_JMI_LIBRARY, 46
I_MPI_{CC,CXX,FC,F77,F90}, 13
I_MPI_{CC,CXX,FC,F77,F90}_PROFILE, 12
I_MPI_ADJUST_<opname>, 174
I_MPI_ADJUST_ALLGATHER_KN_RADIX, 180
I_MPI_ADJUST_ALLREDUCE_KN_RADIX, 180
I_MPI_ADJUST_BCAST_KN_RADIX, 180

I_MPI_ADJUST_BCAST_SEGMENT, 179
I_MPI_ADJUST_GATHERV_KN_RADIX, 181
I_MPI_ADJUST_IALLREDUCE_KN_RADIX, 181
I_MPI_ADJUST_IBCAST_KN_RADIX, 182
I_MPI_ADJUST_IGATHER_KN_RADIX, 182
I_MPI_ADJUST_IREDUCE_KN_RADIX, 182
I_MPI_ADJUST_ISCATTER_KN_RADIX, 182
I_MPI_ADJUST_REDUCE_KN_RADIX, 181
I_MPI_ADJUST_REDUCE_SEGMENT, 179
I_MPI_ALLGATHER_MSG, 185
I_MPI_ALLREDUCE_MSG, 185
I_MPI_ALLTOALL_MSG, 184
I_MPI_ALLTOALL_NUM_PROCS, 184
I_MPI_BCAST_MSG, 184
I_MPI_BCAST_NUM_PROCS, 183
I_MPI_CACHE_BYPASS, 138
I_MPI_CACHE_BYPASS_THRESHOLDS, 138
I_MPI_CHECK_COMPILER, 13
I_MPI_CHECK_DAPL_PROVIDER_COMPATIBILITY, 153
I_MPI_CHECK_PROFILE, 13
I_MPI_CKPOINT, 54
I_MPI_CKPOINT_INTERVAL, 55
I_MPI_CKPOINT_LOGFILE, 55
I_MPI_CKPOINT_NUM, 56
I_MPI_CKPOINT_PREFIX, 54
I_MPI_CKPOINT_PRESERVE, 55
I_MPI_CKPOINT_TMP_PREFIX, 55
I_MPI_CKPOINTLIB, 54
I_MPI_COMPATIBILITY, 188
I_MPI_COMPILER_CONFIG_DIR, 14
I_MPI_CONN_EVD_QLEN, 151
I_MPI_DAPL_BUFFER_NUM, 148
I_MPI_DAPL_BUFFER_SIZE, 149
I_MPI_DAPL_CHECK_MAX_RDMA_SIZE, 150
I_MPI_DAPL_CONN_EVD_SIZE, 151
I_MPI_DAPL_DESIRED_STATIC_CONNECTIONS_NUM, 153
I_MPI_DAPL_DIRECT_COPY_THRESHOLD, 146
I_MPI_DAPL_DYNAMIC_CONNECTION_MODE, 147
I_MPI_DAPL_EAGER_MESSAGE_AGGREGATION, 147
I_MPI_DAPL_LOCALITY_THRESHOLD, 62
I_MPI_DAPL_MAX_MSG_SIZE, 151
I_MPI_DAPL_PROVIDER, 145
I_MPI_DAPL_PROVIDER_LIST, 61
I_MPI_DAPL_RDMA_RNDV_WRITE, 150
I_MPI_DAPL_RDMA_WRITE_IMM, 152
I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT, 149
I_MPI_DAPL_SCALABLE_PROGRESS, 148
I_MPI_DAPL_SR_BUF_NUM, 152
I_MPI_DAPL_SR_THRESHOLD, 152
I_MPI_DAPL_TRANSLATION_CACHE, 145
I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE, 146
I_MPI_DAPL_UD, 154
I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE, 156
I_MPI_DAPL_UD_CONN_EVD_SIZE, 158
I_MPI_DAPL_UD_DESIRED_STATIC_CONNECTIONS_NUM, 161
I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD, 155
I_MPI_DAPL_UD_EAGER_DYNAMIC_CONNECTION, 160
I_MPI_DAPL_UD_MAX_RDMA_DTOS, 163
I_MPI_DAPL_UD_MAX_RDMA_SIZE, 162
I_MPI_DAPL_UD_PROVIDER, 154
I_MPI_DAPL_UD_RDMA_MIXED, 161
I_MPI_DAPL_UD_RECV_BUFFER_NUM, 155
I_MPI_DAPL_UD_RECV_EVD_SIZE, 158
I_MPI_DAPL_UD_REQ_EVD_SIZE, 157
I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT, 159
I_MPI_DAPL_UD_RNDV_COPY_ALIGNMENT_THRESHOLD, 159
I_MPI_DAPL_UD_RNDV_DYNAMIC_CONNECTION, 159
I_MPI_DAPL_UD_RNDV_MAX_BLOCK_LEN, 158
I_MPI_DAPL_UD_SEND_BUFFER_NUM, 155
I_MPI_DAPL_UD_TRANSLATION_CACHE, 157
I_MPI_DAPL_UD_TRANSLATION_CACHE_AVL_TREE, 157
I_MPI_DAT_LIBRARY, 145
I_MPI_DEBUG, 81
I_MPI_DEBUG_INFO_STRIP, 15
I_MPI_DEBUG_OUTPUT, 83
I_MPI_DEVICE, 131
I_MPI_DYNAMIC_CONNECTION, 137
I_MPI_DYNAMIC_CONNECTION_MODE, 147
I_MPI_DYNAMIC_CONNECTIONS_MODE, 147
I_MPI_EAGER_THRESHOLD, 135
I_MPI_ENV_PREFIX_LIST, 63
I_MPI_EXTRA_FILESYSTEM, 207
I_MPI_EXTRA_FILESYSTEM_LIST, 207
I_MPI_FABRICS, 131
I_MPI_FABRICS_LIST, 133
I_MPI_FALLBACK, 134
I_MPI_FALLBACK_DEVICE, 134
I_MPI_FAST_COLLECTIVES, 183
I_MPI_FAULT_CONTINUE, 190
I_MPI_GATHER_MSG, 187
I_MPI_GTOOL, 48
I_MPI_HYDRA_BOOTSTRAP, 41
I_MPI_HYDRA_BOOTSTRAP_AUTOFORK, 42
I_MPI_HYDRA_BOOTSTRAP_EXEC, 42

I_MPI_HYDRA_BOOTSTRAP_EXEC_EXTRA_ARGS, 42
I_MPI_HYDRA_BRANCH_COUNT, 45
I_MPI_HYDRA_CLEANUP, 47
I_MPI_HYDRA_DEBUG, 39
I_MPI_HYDRA_DEMUX, 46
I_MPI_HYDRA_ENV, 39
I_MPI_HYDRA_GDB_REMOTE_SHELL, 45
I_MPI_HYDRA_HOST_FILE, 38
I_MPI_HYDRA_IFACE, 46
I_MPI_HYDRA_PMI_AGGREGATE, 45
I_MPI_HYDRA_PMI_CONNECT, 43
I_MPI_HYDRA_RMK, 43
I_MPI_HYDRA_USE_APP_TOPOLOGY, 49
I_MPI_INTRANODE_EAGER_THRESHOLD, 135
I_MPI_INTRANODE_SHMEM_BYPASS, 143
I_MPI_JOB_ABORT_SIGNAL, 40, 86
I_MPI_JOB_CHECK_LIBS, 44, 85
I_MPI_JOB_CONFIG_FILE, 92
I_MPI_JOB_CONTEXT, 92
I_MPI_JOB_FAST_STARTUP, 88
I_MPI_JOB_RESPECT_PROCESS_PLACEMENT, 47
I_MPI_JOB_SIGNAL_PROPAGATION, 41, 87
I_MPI_JOB_STARTUP_TIMEOUT, 85
I_MPI_JOB_TAGGED_PORT_OUTPUT, 93
I_MPI_JOB_TIMEOUT, 39, 86
I_MPI_JOB_TIMEOUT_SIGNAL, 40, 86
I_MPI_JOB_TRACE_LIBS, 44, 84
I_MPI_LARGE_SCALE_THRESHOLD, 135
I_MPI_LINK, 15
I_MPI_MIC, 60
I_MPI_MIC_POSTFIX, 61
I_MPI_MIC_PREFIX, 60
I_MPI_MPD_CHECK_PYTHON, 93
I_MPI_MPD_CLEAN_LOG, 95
I_MPI_MPD_CONF, 92
I_MPI_MPD_RSH, 94
I_MPI_MPD_TMPDIR TMPDIR, 94
I_MPI_MPIEXEC_TIMEOUT, 39
I_MPI_MPIRUN_CLEANUP, 17
I_MPI_MT_MEMCPY, 208
I_MPI_MT_MEMCPY_NUM_THREADS, 208
I_MPI_MT_MEMCPY_SPIN_COUNT, 209
I_MPI_MT_MEMCPY_THRESHOLD, 209
I_MPI_NETMASK, 163
I_MPI_OFA_ADAPTER_NAME, 167
I_MPI_OFA_DYNAMIC_QPS, 170
I_MPI_OFA_LIBRARY, 171
I_MPI_OFA_NONSWITCH_CONF, 171
I_MPI_OFA_NUM_ADAPTERS, 167
I_MPI_OFA_NUM_RDMA_CONNECTIONS, 168
I_MPI_OFA_PACKET_SIZE, 171
I_MPI_OFA_RAIL_SCHEDULER, 169
I_MPI_OFA_SWITCHING_TO_RDMA, 169
I_MPI_OFA_TRANSLATION_CACHE, 169
I_MPI_OFA_TRANSLATION_CACHE_AVL_TREE, 170
I_MPI_OFI_LIBRARY, 172
I_MPI_OFI_PROVIDER, 173
I_MPI_OFI_PROVIDER_DUMP, 173
I_MPI_OUTPUT_CHUNK_SIZE, 87
I_MPI_PERHOST, 44, 83
I_MPI_PIN, 109
I_MPI_PIN_CELL, 116
I_MPI_PIN_DOMAIN, 118
I_MPI_PIN_MODE, 109
I_MPI_PIN_ORDER, 127
I_MPI_PIN_PROCESSOR_EXCLUDE_LIST, 116
I_MPI_PIN_PROCESSOR_LIST, 110
I_MPI_PIN_PROCS, 110
I_MPI_PIN_RESPECT_CPUSET, 117
I_MPI_PIN_RESPECT_HCA, 117
I_MPI_PLATFORM, 89
I_MPI_PLATFORM_CHECK, 90
I_MPI_PMI_EXTENSIONS, 88
I_MPI_PMI_FAST_STARTUP, 88
I_MPI_PMI_LIBRARY, 88
I_MPI_PRINT_VERSION, 84
I_MPI_PROCESS_MANAGER, 17
I_MPI_RDMA_BUFFER_NUM, 148
I_MPI_RDMA_BUFFER_SIZE, 149
I_MPI_RDMA_CHECK_MAX_RDMA_SIZE, 150
I_MPI_RDMA_CONN_EVD_SIZE, 151
I_MPI_RDMA_EAGER_THRESHOLD, 146
I_MPI_RDMA_MAX_MSG_SIZE, 151
I_MPI_RDMA_RNDV_BUF_ALIGN, 149
I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT, 149
I_MPI_RDMA_RNDV_WRITE, 150
I_MPI_RDMA_SCALABLE_PROGRESS, 148
I_MPI_RDMA_TRANSLATION_CACHE, 145
I_MPI_RDMA_VBUF_TOTAL_SIZE, 149
I_MPI_RDMA_WRITE_IMM, 152
I_MPI_REDSCAT_MSG, 186
I_MPI_RESTART, 56
I_MPI_ROOT, 14
I_MPI_SCALABLE_OPTIMIZATION, 136
I_MPI_SCATTER_MSG, 186
I_MPI_SHM_BUFFER_SIZE, 141

I_MPI_SHM_BYPASS, 143
I_MPI_SHM_CACHE_BYPASS, 138
I_MPI_SHM_CACHE_BYPASS_THRESHOLDS, 138
I_MPI_SHM_CELL_NUM, 140
I_MPI_SHM_CELL_SIZE, 140
I_MPI_SHM_FBOX, 139
I_MPI_SHM_FBOX_SIZE, 140
I_MPI_SHM_LMT, 141
I_MPI_SHM_LMT_BUFFER_NUM, 141
I_MPI_SHM_LMT_BUFFER_SIZE, 141, 143
I_MPI_SHM_NUM_BUFFERS, 141
I_MPI_SHM_SPIN_COUNT, 144
I_MPI_SPIN_COUNT, 136
I_MPI_SSHM, 142
I_MPI_SSHM_BUFFER_NUM, 142
I_MPI_SSHM_DYNAMIC_CONNECTION, 143
I_MPI_STATS, 191, 197, 205
I_MPI_STATS_ACCURACY, 202
I_MPI_STATS_BUCKETS, 194
I_MPI_STATS_FILE, 194, 197
I_MPI_STATS_SCOPE, 191, 198
I_MPI_TCP_BUFFER_SIZE, 164
I_MPI_TCP_NETMASK, 163
I_MPI_TCP_POLLING_MODE, 165
I_MPI_THREAD_LEVEL_DEFAULT, 91
I_MPI_TIMER_KIND, 187
I_MPI_TMI_DRECV, 167
I_MPI_TMI_DSEND, 166
I_MPI_TMI_LIBRARY, 165
I_MPI_TMI_NBITS_RANK, 166
I_MPI_TMI_PROVIDER, 165
I_MPI_TMPDIR, 47
I_MPI_TRACE_PROFILE, 12
I_MPI_TUNE_APPLICATION_COMMUNICATION_GRAPH, 107
I_MPI_TUNE_APPLICATION_STATISTICS, 106
I_MPI_TUNE_HARDWARE_TOPOLOGY_GRAPH, 107
I_MPI_TUNE_RANK_PLACEMENT, 106
I_MPI_USE_DAPL_INTRANODE, 143
I_MPI_USE_DYNAMIC_CONNECTIONS, 137
I_MPI_USE_RENDEZVOUS_RDMA_WRITE, 150
I_MPI_WAIT_MODE, 137
I_MPI_YARN, 18
-ib, 37, 74
-IB, 37, 74
-iface <interface>, 28
-ifhn <interface/hostname>, 80
-ilp64, 9, 29
-info, 30
IPM, 197

L

-l, 28, 79
LD_LIBRARY_PATH, 9, 46
libjmi.so, 32
-link_mpi=<arg>, 10
LMT, 141
-localhost, 30

M

-m, 79
-machine <machine file>, 20
-machinefile <machine file>, 20, 76
max_rdma_size, 150, 151
mpd, 65
MPD, 16, 17, 65, 66, 72, 80, 87
mpd.hosts, 92
MPD_CON_EXT, 92
MPD_SECRETWORD, 92
mpdallexit, 68
mpdboot, 66
mpdcheck, 69
mpdcleanup, 68
mpdexit, 67
mpdhelp, 72
mpdkilljob, 71
mpdlistjobs, 70
mpdringtest, 70
mpdsigjob, 71
mpdtrace, 69
MPI{CC,CXX,FC,F77,F90}_PROFILE, 12
MPICH_{CC,CXX,FC,F77,F90}, 13
mpicleanup, 50
mpiexec, 16, 18, 72, 73, 75, 76, 81, 83, 85, 86
mpiexec.hydra, 16, 19, 39, 44, 50, 56
MPIEXEC_SIGNAL_PROPAGATION, 41, 87
MPIEXEC_TIMEOUT, 39, 86
MPIEXEC_TIMEOUT_SIGNAL, 40, 86
MPIEXEC_TRACE_LIBS, 84
mpirun, 15
mpitune, 99
-mps, 22
-mx, 37, 74
-MX, 38, 74

N

-n <# of processes>, 35, 80

-no_ilp64, 9
-noconf, 29, 80
-nolocal, 28, 75
-nostrip, 8
-np <# of processes>, 35, 80
NUM_RDMA_BUFFER, 148

O

-O, 10
-ofi, 38
-OFI, 38
-ordered-output, 29, 79

P

-path <directory>, 29, 36, 81
-perhost <# of processes>, 21, 75
-pmi-aggregate, 23
-pmi-connect <mode>, 21
-pmi-noaggregate, 23
-ppn <# of processes>, 21, 76
-print-all-exitcodes, 35
-print-rank-map, 35
-profile=<profile_name>, 8
-psm, 38, 75
-PSM, 38, 75
-psm2, 38
-PSM2, 38

R

-rdma, 36, 73
-RDMA, 37, 73
RDMA_IBA_EAGER_THRESHOLD, 146
-restart, 52
-rmk <RMK>, 35
-rr, 21, 76

S

-s <spec>, 29, 79
-show, 10
-show_env, 10
-static, 8
-static_mpi, 8

T

-t, 8
-t [<profiling_library>], 21, 77
-tmi, 37, 74
-TMI, 74
-TMI, 37
-tmpdir, 30
TMPDIR, 47, 50, 94
TotalView, 23, 78
TOTALVIEW, 89
-trace, 8
-trace [<profiling_library>], 21, 77
-trace-collectives, 22
-trace-pt2pt, 22
-tune [<arg >], 28, 75
-tv, 23, 78
-tva <jobid>, 78
-tva <pid>, 23
TVDSVRLAUNCHCMD, 23
-tvsu, 79

U

-umask <umask>, 36, 81
-use-app-topology <value>, 30

V

-v, 11, 35
-V, 30, 75
-verbose, 35
-version, 30, 75
VT_ROOT, 9, 14

W

-wdir <directory>, 36, 81

ら

ラージメッセージ転送, 141