MPI の標準化から 20 年 : 共通のアプリケーション・バイナリー・インターフェイスの実現

Alexander Supalov インテル コーポレーション クラスター・ツール・アーキテクト
Artem Yalozo インテル コーポレーション ソフトウェア・エンジニア

はじめに

分散メモリー・コンピューティングの業界標準であるメッセージ・パッシング・インターフェース (MPI)[1] が誕生してから今年で 20 年になります。ハイパフォーマンス・コンピューティング (HPC) で要求される高度なパフォーマンスにおいて、MPI に勝るインターフェイスはほとんどないでしょう。

1994 年 5 月の MPI Forum で発表されて以来、MPI は大きな進化を遂げました。2012 年 9 月には最新のメジャーバージョンである MPI-3 標準 [2] がリリースされました。このバージョンでは、高速な一方向通信、非ブロッキング集合操作、その他の機能が追加されています。

この最新バージョンに基づいて、インテルは SuperComputing 2013 にて Argonne National Laboratory、Cray Corporation、および IBM Corporation と協力し、長い間待ち望まれていた共通の MPI アプリケーション・バイナリー・インターフェイス (ABI) を作成するためのイニシアチブを結成しました。この ABI は、インテル® MPI ライブラリー 5.0 として 2014 年 6 月にリリースされ、MPICH、Cray* MPI、IBM* POE を含む多くの互換サードパーティー製品でも利用できます。

目的

ハイパフォーマンス・コンピューティング・クラスターで利用されるマルチファブリック・メッセージ・パッシング・ライブラリーであるインテル® MPI ライブラリーにとって、互換性は長年にわたる大きな課題でした。[3] ユーザーは、製品のメジャーバージョン間でさえも常に互換性が保持されることを期待しています。インテル® MPI ライブラリーでは新しいバージョンへ簡単に移行できるようにバイナリー互換のアップグレード・パスを用意しています。ABI 互換でなければ、ライブラリーの新しいバージョンがリリースされるたびに、古いバージョンで開発されたリリース済みソフトウェアをすべて再リリースしなければなりません。

潜在的な互換性問題の原因

以前のインテル® MPI ライブラリーの互換性問題は、異なるソースベースを使用していたことが原因でした。各バージョンで異なる MPICH[4] ソースコードを使用し、異なるバージョンの MPI 標準を実装していました。例えば、インテル® MPI ライブラリー 4.1 は MPICH2、つまり MPI-2.2 標準をベースにしていました。インテル® MPI ライブラリー 5.0 は MPICH3、つまり新しい MPI-3 標準をベースにしています。

アプリケーション・バイナリー・インターフェイスは、2 つのプログラムモジュール間の低レベルのインターフェイスです。ライブラリーへのバイナリー・インターフェイスを定義するものであり、アプリケーション・プログラミング・インターフェイス (API) とは異なります。ABI は関数の呼び出し方法、データ型のサイズ、レイアウト、アライメントを定義します。同じ ABI を利用することで、互換プログラムは同じランタイム規則に従います。プログラムがライブラリーの以前のバージョンにリンクされている場合、ライブラリーはバイナリー互換なので、再コンパイルしなくても新しいバージョンで動作します。

また、新しいバージョンのライブラリー関数はすべて、以前のバージョンと同じ動作になります。当然のことながら、以前のバージョンで動作が正しくない場合は、新しいバージョンでも動作が変わることがあります。これにより動作の互換性が失われる可能性があります。さらに、MPI 標準の変更によって互換性が失われることもあります。

互換性のシナリオ、問題、解決方法

ソフトウェアの下位互換性と上位互換性の両方が保証されることが理想的ですが、実際には下位互換性と新しい標準の実装要件の間で合理的なバランスがとられます。

インテル® MPI ライブラリーで重要なのは下位互換性です。新しいバージョンでビルドされたアプリケーションは、以前のバージョンのランタイムで動作できなければなりません。上位互換性も考慮されますが、保証されません。

ABI の互換性問題はクライアント・コードでクラッシュを引き起こしたり、データを破損する可能性があります。また、問題に気付かないまま見過ごされることもあるでしょう。そのため、テスト時に互換性問題によるクラッシュを見つけるのではなく、その前に潜在的な互換性問題を注意深く調査すべきです。

インテル® MPI ライブラリーには C、C++、および Fortran 77/90 に固有の機能がいくつかあります。各インターフェイスへの変更は、バイナリー互換性問題を引き起こす可能性があります。表 1 に、インテル® MPI ライブラリー 5.0 (MPICH3 ベース) と 4.1 (MPICH2 ベース) の間で見つかっているバイナリー互換性問題の原因と解決方法を示します。

原因 (インテル® MPI ライブラリー 4.1 と MPICH3 間の変更)	**言語固有**	**解決方法**
事前定義定数の変更 : **MPI_MAX_ERROR_STRING**	すべて	インテル® MPI ライブラリー 4.1 の ABI に戻します。
定数/新しい定義の追加: **MPICH_ATTR_FAILED_PROCESSES** **MPI_UNWEIGHTED** **MPICH_ERR_LAST_CLASS**	C	定数を含め、新しい定義を使用します。
事前定義定数の追加/変更: **MPI_COMBINER_HINDEXED_BLOCK** **MPI_COMBINER_***	C Fortran	**HINDEXED_BLOCK** に次の値を割り当て、残りはそのままにします。
MPI_Status 項目の順序変更 (**count** と **cancelled** が下に移動)。**count** の型の **int** から 64 ビットの **MPI_Count** への変更。 **typedef struct MPI_Status** { int MPI_SOURCE; int MPI_TAG; int MPI_ERROR; **MPI_Count count;** **int cancelled;** } MPI_Status;	C	**count** と **cancelled** を上に移動します。 32 ビットの **count** と cancelled を 63 ビットの **count** と 1 ビットの **cancelled** 値として解釈します。
事前定義定数の追加/削除: **MPI_2COMPLEX、** **MPI_2DOUBLE_COMPLEX**	Fortran	定数を含めます。
共通ブロックの削除/追加: **MPIPRIV1、MPIPRIV2、MPIPRIVC、MPIFCMB5、MPIFCMB9**	Fortran	インテル® MPI ライブラリー 4.1 の ABI に戻し、MPI-3 用の新しい共通ブロックをすべて追加します。
仮想関数テーブルの変更 (順序変更): **virtual void Shift()** **Cartcomm Dup()** **virtual int Get_cart_rank()** **virtual void Get_topo()** **virtual int Get_dim()** **virtual int Map()** **virtual Cartcomm Sub()**	C++	インテル® MPI ライブラリー 4.1 の ABI に戻します。

1 バイナリーの互換性問題と解決方法

インテル® MPI ライブラリー 5.0 では、既知の互換性問題が修正され、インテル® MPI ライブラリー 4.x とバイナリー互換性を実現しました。当初、ベータ版インテル® MPI ライブラリー 5.0 は MPICH3 と互換性がありませんでしたが、ビルド段階で ABI 変更を調整して MPICH3 互換ライブラリーをビルドできることが分かったため、MPICH3 との互換性が保持されました。

表 2 に、インテル® MPI ライブラリー 4.1 (MPICH2 1.4 ベース) と 4.0 (MPICH2 1.1 ベース) の間で見つかっている動作の互換性問題の原因と解決方法を示します。

原因 **(インテル® MPI ライブラリー 4.1 と MPICH2 1.4 間の変更)**	**言語固有**	**解決方法**
MPI 標準における一部の関数の動作変更: **MPI_Cart_create** **MPI_Cart_map** **MPI_Graph_create** **MPI_Win_get_attr**	C	I_MPI_COMPATIBILITY 環境変数の値に基づいて関数の動作を変更します。
MPI 標準における一部の集合操作の動作変更: **Gather** **Allgather** **など**	C	I_MPI_COMPATIBILITY 環境変数の値に基づいて関数の動作を変更します。

2 動作の互換性問題と解決方法

インテル® MPI ライブラリーは新しい標準を実装しなければならず、新しい標準では動作が変更される可能性があるため、デフォルトでは動作の互換性を保証していません。ただし、実行時に **`I_MPI_COMPATIBILITY` 環境変数**を 4 に設定することで、動作の下位互換性を確保することができます。

これにより、1 つのライブラリーで新しい標準をサポートしつつ、以前のバージョンでビルドされたほとんどのアプリケーションと下位互換性を保持できます。

まとめ

インテル® MPI ライブラリーは、すべての過去のバージョンと互換性があります。これは、インテル® MPI ライブラリーの優れた利点と言えるでしょう。新しい MPI ABI は、2014 年の夏にパートナー各社で導入される予定です。新しい ABI の詳細は、MPICH ABI Compatibility Initiative[5] の Web サイト (http://www.mpich.org/abi/) を参照してください。導入後、インテル® MPI ライブラリー、MPICH、Cray* MPI、および IBM* POE は実行時に完全互換になります。さらに、この ABI により、1 つの MPI ライブラリーで MPI-2.x と MPI-3 の両方をサポートできるようになります。これは、以前の MPI 標準と関連アプリケーションに対する投資を保護し、1 つのパッケージをサポートするだけで済むため、MPI 実装とその保守が容易になります。また、パートナー各社の MPI 実装にわたってバイナリー互換性が拡張されることも、顧客とパートナーの双方にとって重要です。

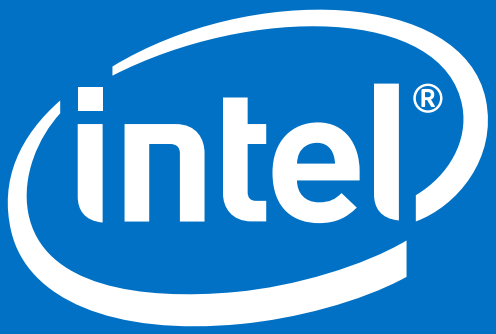
ABI によりライブラリー間の切り替えが可能になるため、開発者は異なる MPI 実装をパフォーマンスに基づいて比較することができます。さらに、現在特定の MPI ライブラリー向けにビルドされているツールがほかの MPI ライブラリーでも利用できるようになるでしょう。

参考文献

1. メッセージ・パッシング・インターフェイス (MPI) ドキュメント : http://www.mpi-forum.org/docs (英語)
2. メッセージ・パッシング・インターフェイス・フォーラム。
   MPI: A Message-Passing Interface Standard, Version 3.0.
   September 21, 2012.
   http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf (英語)
3. インテル® MPI ライブラリー : www.intel.com/go/mpi (英語) ;
   http://www.isus.jp/article/idz/hpc/intel-mpi-library/
4. MPICH: http://www.mpich.org (英語)
5. MPICH ABI Compatibility Initiative: http://www.mpich.org/abi/ (英語)

インテル® MPI ライブラリーを試そう >

Parallel Universe 18 号の全文はこちらでご覧いただけます。



Software

The Parallel Universe