社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－ARC－182 (20)
2009－HPC－119 (20)
2009／2／27

並列プログラムの実行可搬性を実現するMPI通信ライブラリの設計

住元 真司[†] 中島 耕太[†] 成瀬 彰[†]
久門 耕一[†] 安井 隆[††] 鴨志田 良和[†††]
松葉 浩也[†††] 堀 敦史[†††] 石川 裕[†††]

本論文では、複数の実行環境が異なるクラスタシステム間での並列アプリケーションバイナリの可搬性を実現するための MPI 通信ライブラリ（MPI-Adaptor）の設計について述べる。実行バイナリの可搬性を確保するためには、MPI の実装毎に異なる引数と返り値を変換する必要がある。実現の有効性を判断するため、最小限度の変換機能をもつ MPI-Adaptor のプロトタイプを実装した。簡易の評価結果、Xeon 3.8GHz の PC クラスタにおいて、MPI の変換オーバヘッドは 送受信関数 1 回あたり 0.028μs 程度と小さく、MPI-Adaptor 実現の有用性を示した。

A Design of MPI Communication Library for Execution Portability of Parallel Program Executables

Shinji Sumimoto,[†] Kohta Nakashima,[†] Akira Naruse,[†]
Kouichi Kumon,[†] Takashi Yasui,[††] Yoshikazu Kamoshida,[†††]
Hiroya Matsuba,[†††] Atsushi Hori[†††] and Yutaka Ishikawa [†††]

This paper presents a design of MPI library for execution portability of parallel program to realize seamless runtime environment among PC clusters with different runtime environments, called MPI-Adaptor. To realize the execution portability, MPI-Adaptor needs to translate arguments and return value of MPI functions which are different among MPI implementations. We have implemented a research prototype to evaluate effectiveness of MPI-Adaptor with minimum functionality. Our evaluation results using Xeon Processor (3.8GHz) based PC cluster show that MPI translation overhead of MPI sending(receiving) is around 0.028μs and MPI-Adaptor is effective for seamless runtime environment.

1. はじめに

PC クラスタは高性能計算において広く使われるようになり、大学、研究機関、企業での普及が進んでいる。さらにインターネットの普及により PC クラスタの遠隔利用も一般的になっている。大学や研究機関の計算機センターにおいても、理研スーパコンバインドクラスタ [1] から始まり、T2K[2] に代表される標準仕様の大規模 PC クラスタの導入が進んでいる。

複数の大規模 PC クラスタを持つ計算センターを利用するユーザが増えてくると、利用形態が大きく広がると我々は考えている。例えば、開発とデバッグは研究室の小規模システム、大規模実行は大学のセンターマシンでの実行、あるいは、複数の計算機センター間で空き状況や実行時間あたりのコストを考えたシームレスなジョブ実行が進むであろう。

計算機センター間でシームレスなジョブ実行を行うには、実行バイナリをはじめ実行環境が統一されているのが望ましい。しかし、現状の大規模 PC クラスタは、納入ベンダーや運用方針などにより、必ずしも実行環境が統一されていない。例えば、T2K のように、ノード内構成と OS が同じであっても、MPI の実行環境が異なるためユーザは各計算センターの実行環境上で再コンパイルする必要がある (図 1)。さらには、PC クラスタ環境では、ベンダーが提供する MPI の実行環境だけでなく、MPICH2[3] や OpenMPI[4] といったオープンソースの MPI 実行環境が使われることがある。また、MPI の実行バイナリをダウンロードして実行することも想定される。このような場合にも、実行時に再コンパイルすることなく、MPI の実行環境が選択可能なことが望まれる。

本論文では、並列プログラムバイナリのシームレスな可搬性を実現する MPI 通信ライブラリ (MPI-Adaptor) の設計について述べる。MPI-Adaptor の

† 富士通研究所
FUJITSU LABORATORIES
†† 日立製作所
HITACHI
††† 東京大学
The University of Tokyo

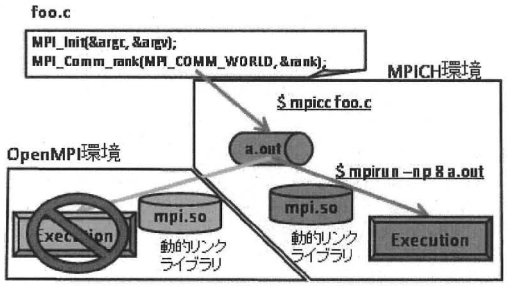

図 1　実行環境が異なり再コンパイルが必要な場合

実現においては、MPI の実装毎に異なる引数と返り値を変換することが必要になる。これを、動的リンクライブラリを用いて実行環境の非依存性を確保し、MPI-Adaptor が MPI 変換を実現する。(図 2)

実現の有効性を判断するため、特定の 2 つの MPI 実装間での最小限度の変換機能をもつ MPI-Adaptor のプロトタイプを実装した。簡易評価の結果、Xeon 3.8GHz の PC クラスタにおいて、MPI の変換オーバヘッドは MPI の送受信関数 1 回あたり 0.028μs と小さく、MPI-Adaptor 実現の有用性を示した。

本論文の構成は次に示す。第 2 章では、既存の MPI の実装を比較し、MPI-Adaptor を実現するために考慮すべき点について述べる。第 3 章で MPI-Adaptor の目標と課題を述べる。第 4 章は実現する MPI-Adaptor の設計、第 5 章では MPI-Adaptor の実装、第 6 章では評価、第 7 章で関連研究について述べる。

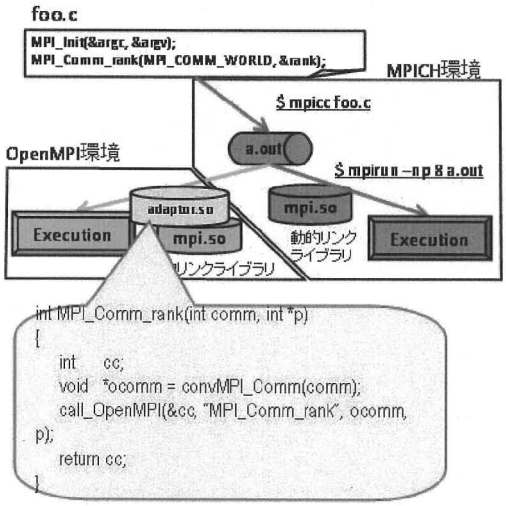

図 2　MPI-Adaptor によるシームレスな実行イメージ

2. MPI 実装の違いとプログラム可搬性実現に必要な機能

2.1 MPI 実装の違いとプログラム可搬性への影響

MPI 仕様では型と値が定義されており、この型と値を用いてプログラムを行う。表 1 に MPI 使用で定義されている型と値の例を示す。MPI forum[5] の ABI WG において、各 MPI の実装の違いが情報が整理されているが、その大きな違いは次の点である。

(1) MPI 規格定義の型と実装での表現手法：MPI_Comm、MPI_Status（表 1、構造体の場合は内部のメンバの違いも関係）
(2) 型のとりうる値の違い: MPI_COMM_WORLD、エラーコード番号など

以上の定義は、MPI ヘッダで定義されているため、プログラムをコンパイルする際に MPI の定義は MPI の実装依存の定義に置き換えられる。これは、コンパイル時にプログラムバイナリに埋め込まれるため、そのまま、プログラムを実行すると実装が異なる MPI ライブラリでは間違って解釈される。

それ故、実装が異なる場合には、MPI の定義の違いを変換してプログラムを実行する必要がある。

表 1　MPI 規格で定義されている型と値の例

用途	型 (値の例)
Communicator	MPI_Comm (MPI_COMM_WORLD)
Group	MPI_Group
Request	MPI_Request
Status	MPI_Status
Data type	MPI_Datatype (MPI_INT)
Operation	MPI_Op (MPI_SUM)
Window	MPI_Win
File	MPI_File
Info	MPI_Info
Pointer の差分	MPI_Aint
Offset	MPI_Offset
Error Handler	MPI_Errhandler

2.2 MPICH2 と OpenMPI の実装の違い

本節では、具体的な MPI の定義の違いを MPICH2[3] と OpenMPI[4] を例にあげ、どのような MPI の定義がされ、MPI-Adaptor 実現時にどのような点を考慮する必要があるのかを整理する。

表 2 は、MPICH2 と OpenMPI の C 言語における MPI の型定義の展開例である。MPI の実装の方針として MPICH は MPI の型定義を int で定義しているのに対し、OpenMPI では構造体で定義している。これは、MPI-Adaptor 実現時に MPI 定義の型変換が必要である。また、表 2 における MPI Error Class の定義では、型自体は int と同じ型で定義されているが、値が一部異なるため、値の変換も同様に必要である。

表 3 は、MPICH2 と OpenMPI の Fortran 言語に

表 2　MPI の型実装の違いの例 (C 言語)

MPI 値の例	MPICH2	OpenMPI
MPI_COMM_WORLD	0x44000000	&ompi_mpi_comm_world
MPI_INT	0x4c000405	&ompi_mpi_int
MPI_INTEGER	0x4c00041b	&ompi_mpi_intger
MPI_SUCCESS	0	0
MPI_ERR_TRUNCATE	14	15

おける MPI の型の展開例を示す。MPI の実装方針として MPICH と OpenMPI ともに MPI の定義の型を int で定義しているが値が異なる場合がある。従って MPI-Adaptor 実現には、値変換が必要である。なお、いずれも int 型で定義されているのは、Fortran90 より前の言語仕様が、ポインタや構造体を扱えないために int の型定義を採用しているものと考えられる。

表 3　MPI の型実装の違いの例 (Fortran 言語)

MPI 値の例	MPICH2	OpenMPI
MPI_COMM_WORLD	0x44000000	0
MPI_INTEGER	0x4c00041b	7
MPI_SUCCESS	0	0
MPI_ERR_TRUNCATE	14	15

表 2 と表 3 を比較する。MPICH2 の場合、2 つの定義を比較すると同じ型と値が定義されている。これは、MPICH2 では、C 言語と Fortran 言語の内部構造が同じである想定される。一方、OpenMPI では、型宣言は異なるため、内部で変換関数を用いて型変換する実装となっている。なお、MPI Error Class については、そのまま同じ値を用いている。

2.3 他の MPI 実装

MPI forum ABI working group[6] に、ベンダ製 MPI を含む MPI の実装の違いを整理したデータがある。表 4 に MPI 実装の違いをまとめる。

表 4　各 MPI 実装における実装

MPI 実装	description
Intel MPI	MPICH2 の実装に準拠
MS MPI	MPICH2 の実装に準拠
HP MPI	独自、型はポインタ型
LAMPI	独自、型はポインタ型と数値混在
NEC MPI	独自、型に数値型

表 4 より MPICH2 ベースの実装を採用しているのが Intel MPI と MS(MicroSoft) MPI であり、他のベンダーは歴史的にはどうであれ、現状は独自の実装となっている。以上より、型変換については、ポインタ型と数値型の 2 つの型の変換機能で対応可能である。

3. MPI-Adaptor の目標と課題

MPI-Adaptor の実現目標は、異なる実行環境を持つ PC クラスタ間において、相互の実行バイナリの実行可搬性を最小の変換コストで実現することである。最終的には様々な MPI の実装について、mpi.h(mpif.h) を読み込めば自動的に変換用のライブラリが生成され、このライブラリを用いて異なる実行環境上で相互実行が可能なものを目指している。

2 章で述べた様々な MPI の実装について相互実行が可能な MPI-Adaptor を実現するためには、次の課題がある。

- 各 MPI 実装で定義している型と値定義の把握
- MPI 関数引数と返り値を変換する機構の実現
- 各 MPI 実装が持つ非互換性の扱い

課題について、次の方針で考えるものとする。

各 MPI 実装で定義している型と値定義の把握 については、mpi.h(mpif.h) にすべての情報は存在する。人手でデータを抽出する方法と、プログラムによる抽出する方法がある。最初のプロトタイプでは人手による抽出とするが、将来的にはプログラムによる自動生成を考える。

MPI 関数引数と返り値を変換する機構の実現については、基本的には単純変換で対応可能であるが、MPI のコミュニケータを生成するなど、動的に型が生成される場合は、型の変換についても動的に生成して、変換可能にする必要がある。

各 MPI 実装が持つ非互換性の扱いについては、2 つの考え方がある。

(1)　非互換性はエミュレートしない
(2)　非互換性を含めエミュレートする

MPI-Adaptor の目標は、バイナリの差を意識しないでプログラムを実行することである。この主旨からすると、課題は次の 2 つに分類される。

(1)　実行環境が異なる PC クラスタシステムでのプログラム実行において、数値演算の精度を含んで実行の互換性を確保するか？
(2)　MPI 実装の違いでプログラムの動作が異なる（正常に実行されない）。

前者の数値演算の精度の保証については、保証しないこととする。これは、MPI-Adaptor の機能レベルだけでなく、並列度やシステムの持つ算術演算ライブラリの違いによる誤差も考えられ、MPI ライブラリに留まらないからである。周辺環境を含む実行環境全体で考える必要がある。

後者の MPI 実装の違いによるプログラム動作が異なる点については、「MPI プログラムは本来実装間の違いに依存なく、動作するはずである。」という前提で考える。しかし、そうはいっても、重要なアプリケー

ションが動かないことも想定される。したがって、その問題の影響度に応じて、問題を回避するような変換は考えることにする。

4. MPI-Adaptor の設計

本章では、第 3 章で述べた MPI-Adaptor の設計について次の 3 点について議論する。

- MPI-Adaptor のアーキテクチャ
- MPI 実装間の変換設計
- 低オーバヘッドでの実現

4.1 MPI-Adaptor のアーキテクチャ

MPI-Adaptor はプログラムがコンパイルされた MPI 実行環境（オリジナル環境）から、実際に実行する MPI 実行環境（ターゲット環境）への変換を行う実行ランタイムである。これを、実現するためには次の点を実現可能なアーキテクチャとする必要がある。

- オリジナルの MPI 実行環境依存性の排除
- オリジナル側とターゲット側のライブラリが持つ同名の関数の呼び出しの実現
- 高速な型変換の組み込み

オリジナルの MPI 実行環境依存性の排除 については、動的リンクライブラリ対応の MPI 実装を前提として考える。ほとんどの MPI 実装が、動的リンクライブラリ対応になっているが、プログラム生成時に個別に静的リンクされた実行バイナリは対応できない。

オリジナル側とターゲット側のライブラリが持つ同名の関数の呼び出しの実現 については、オリジナル側の実行バイナリもターゲット側の MPI ライブラリも同じ関数名のシンボルを保持している。このため、シンボルが衝突しない変換の仕組みが必要である。これを実現するために、動的リンクライブラリ向けのライブラリの機能を用いてターゲット側関数へのポインタを取得、変換テーブルに格納した上で、この変換テーブルベースに関数を呼び出す作りとする。

高速な型変換の組み込み については、複数の MPI 実装で高速に型変換が扱えるアーキテクチャが必要である。第 2.2 節での議論により、数値型とポインタ型の相互変換機能を基本とした作りとする。

4.2 MPI 実装間の変換設計

MPI 実装のオリジナル側とターゲット側の型変換を設計する上で考慮すべき型変換時の局所性について考える。表 1 で示される MPI の型については、アプリケーションにより局所性があると考えられる。表 5 にそれぞれの MPI の型毎の局所性について整理する。

表 5 より、利用頻度が高い型と値については優先的に高速化すると共に、アプリケーションにより変換の偏りがある場合には LRU などの手法を採用して、変換オーバヘッドを抑える必要があることがわかる。

表 5　MPI の型利用の局所性

用途	利用の局所性と対処
Communicator	MPI_COMM_WORLD が多用 変換の偏りがあると想定
Request	非同期通信の際に多用
Status	構造体と構造体間のマッチング MPI_SUCCESS が多い
Data type Operation	プログラムで利用頻度の偏りがあると想定 LRU などの手法が効果的
Window File Pointer の差分 Offset	MPI2 規格の通信利用の場合、新規生成想定
Info Group Error Handler	利用頻度少ないと想定

4.3 低オーバヘッドでの実現

MPI-Adaptor のオーバヘッドは、最低でも動的リンクの関数 1 回分の呼び出しと関数変換テーブルを用いた関数呼び出しオーバヘッドに加え、型（値）と変換のオーバヘッドが加わる。それぞれのオーバヘッドを定量的に把握して実装設計を行う必要がある。

5. MPI-Adaptor のプロトタイプ実装

本章では MPI-Adaptor 実現時の実用性を評価するために開発した MPI-Adaptor プロトタイプの実装について述べる。プロトタイプ実装の指針としては、機能を最小限に抑え最小限の MPI アプリケーションが実行できる程度のものを実装することとする。

5.1 MPI-Adaptor プロトタイプの機能と構成

MPI-Adaptor プロトタイプの実現している機能と構成について述べる。

MPI-Adaptor プロトタイプの機能

MPI-Adaptor プロトタイプのもつ機能を述べる。

- OpenMPI 環境でコンパイルされたバイナリプログラムを MPICH2/SCore で実行可能
- C プログラム対応 (Fortran は未対応)
- MPI-1 規格準拠の関数
- Communicater など型の新規生成には未対応

最小限のアプリケーションが稼働できる機能に限定している。

MPI-Adaptor プロトタイプの構成

MPI-Adaptor プロトタイプの動作イメージは、図 2 であり、MPI-Adaptor プロトタイプで実現している機能としては

- MPI_Init 時に、ターゲットの動的リンクライブラリを動的リンクライブラリ向けの関数 (dlopen(), dlsym()) を用いて、使う関数をへのポインタを抽出して関数変換テーブル作成する。
- MPI 定義の 305 関数のエントリを作成し、引数と

返り値変換の関数を準備して型変換を実行、作成した関数変換テーブルを用いてターゲットのMPIライブラリ関数を呼び出す。

変換関数の例を以下に示す。d_MPI_Comm はターゲットにおける MPI_Comm の型を示し、mpiconv_s2d_comm() は、オリジナルからターゲットへの MPI_Comm 型を変換する関数、mpiconv_d2s_serrcode() は返り値をターゲットからオリジナルに変換する関数である。ftables[] がターゲットへの関数変換テーブルである。

```
#include "mpi.h"
int MPI_Comm_rank(MPI_Comm comm, int *rank) {
        int dret;
        d_MPI_Comm dcomm = mpiconv_s2d_comm(comm);
        dret = (*ftables[OP_MPI_Comm_rank].funcp)
                                        (dcomm, rank);
        return mpiconv_d2s_serrcode(dret);
}
```

5.2 misc ライブラリ対応

OpenMPI では、mpi だけに関係するライブラリの他、ランタイムまわりの動的ロードライブラリも参照しているため、MPI-Adaptor によるプログラム実行時にも、次のコマンド実行例のように追加のライブラリ (libopen-rte.so.0 , libopen-pal.so.0) が必要になる。

```
[s-sumi@pdspccs90 MPI-trans]$ make sample
/opt/OpenMPI3.0/bin/mpicc sample.c  -o sample
[s-sumi@pdspccs90 OpenMPI]$ ldd sample
  libmpi.so.0 (0x00002aaaaaaad000)
  libopen-rte.so.0 (0x00002aaaaad48000)
  libopen-pal.so.0 (0x00002aaaaaf8e000)
  /lib64/ld-linux-x86-64.so.2 (0x00000030f7a00000)
```

このため、MPI-Adaptor 用のライブラリとして libmpi.so.0 を準備、その他はダミーライブラリを生成しプログラム実行時に参照している。

5.3 そ の 他

未定義関数 ：OpenMPI から、MPICH2 への変換においては、C とフォートラン間での変換関数が定義されていなかった。MPICH2 では、mpi.h で即値変換しているため、変換関数はなくても問題にならないはずである。現状は未定義関数が呼ばれるとエラーになる実装としている。

グローバル変数 ： OpenMPI で用いているグローバル変数は構造体の大きさを確保した上で、MPI-Adaptor のライブラリ内で定義している。

動的ライブラリの切り替え ：環境変数 (LD_LIBRARY_PATH) を用いて切り替えている。

6. MPI-Adaptor の評価

MPI-Adaptor の評価として、MPI-Adaptor 自体の挿入オーバヘッド、転送遅延、転送バンド幅、アプリケーションへの影響について評価した後、詳細な処理コストについて評価する。

6.1 評 価 環 境

表 6 評価環境

Computation Node	DUAL Xeon 3.8GHz Primargy RX200S2
Ethernet (1Gbps)	Intel E1000 NIC Netgear 48port GigE Switch
OS	CentOS 5.1 x86_64 (2.6.18-8.el5 kernel) SCore7.0

MPI-Adaptor の評価環境として、SCore[7] 搭載のクラスタで評価する。表 6 に評価環境を示す。ネットワークは Gigabit Ethernet であり、低レベル通信層は PMX/Etherhxb である。MPI は MPICH2/SCore を用いている。評価では OpenMPI バイナリを MPI-Adaptor を用いて MPICH2/SCore で実行した結果と MPICH2/SCore の Native 実行結果を比較する。

6.2 Pingpong プログラムによる MPI-Adaptor のオーバヘッド

MPI の pingpong プログラムを用いて MPI-Adaptor のオーバヘッドを測定した。2 つのプログラムは MPI_send と MPI_recv 関数を用いた簡単のものである.

表 7 MPI ラウンドトリップ (RTT) 時間

	RTT	Ratio
MPICH2/SCore	43.328 μs	100.0%
OpenMPI+MPI-Adaptor	43.440 μs	100.2%

注) Switch の 1 段あたりの往復遅延 6.0 μs を含む

表 7 に、MPI の pingong プログラムを用いた RTT 時間の測定結果を示す。表 7 より、MPICH2/SCore と OpenMPI のバイナリを MPI-Adaptor を介した MPICH2/SCore の差である MPI-Adaptor のオーバヘッドは、0.112 μs であり、その時間の占める割合も 0.2%と小さい。MPI_send, MPI_recv 関数 1 回あたりの呼び出しオーバヘッドは 0.028μs と特に問題になる値ではないことが分かった。

6.3 Burst プログラムによる MPI-Adaptor の転送性能

MPI の burst 転送プログラムを用いて通信バンド幅を測定した。図 3 に測定結果を示す。MPI-Adaptor を用いた場合は 512 バイトから 1024 バイトあたりの立ち上がりが少し鈍くなっているが最大バンド幅への影響は特に大きくないと言える。

6.4 NPB IS による性能評価

MPI-Adaptor を用いた場合のアプリケーションとして NAS 並列ベンチマークの IS を用いて評価した。表 8 に実行結果を示す。ノード数は 8 ノード（1 ノードあたり 1 プロセス）である。

表 8 の結果より、オリジナルの方が性能が少し速い。

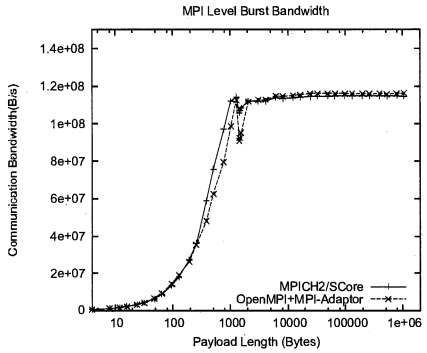

図 3 MPI レベルの転送バンド幅性能

表 8 NPB IS Class A 8node

	Mops Total
MPICH2/SCore	50.75
OpenMPI+MPI-Adaptor	48.47

性能差の原因については詳細な調査が必要である。

6.5 MPI-Adaptor の構成要素のコスト調査

MPI-Adaptor のオーバヘッドを決める各構成要素の詳細な実行コストを測定した.

表 9 MPI-Adaptor の構成要素コスト

	コスト
ローカル関数呼び出し	3.1 ns
共有ライブラリ関数呼び出し	4.1 ns
共有ライブラリ+変換関数 if 文 1 回	4.6 ns
共有ライブラリ+変換関数 if 文 48 回	17.3 ns
if 文 1 回あたり	0.27 ns

表 9 に MPI-Adaptor の構成要素コスト測定結果を示す。表 9 より、新たな共有ライブラリの呼び出しによるコストは、4.1 ns とローカル関数呼び出しコストの 3.1 ns に比べ 1.0ns の違いであった。よって、本結果からは、静的リンクか動的リンクかによるオーバヘッドは使い方にもよるが、MPI-Adaptor の実現という意味からは問題になるレベルではないと考える。また、if 文 1 回の処理コストは、0.27 ns と小さいが比較対象が 48 回だと 0.27ns のコストに対して 13.0ns 増えることがわかる。MPI_TYPE の型は 48 種あるので、今後の実装には考慮が必要である。

7. 関連研究

関連する研究として、MPI フォーラムにおける ABI の統一化と MorphMPI がある。

- MPI フォーラムにおける ABI の統一化: MPI Forum では、次期の MPI の規格である MPI 3.0 の規格検討を進めており、その中で、Application Binary Interface Working Group[6] において、ABI の統一化の検討が進められている。
- MorphMPI[8]、共通の mpi.h(mpif.h) を実現し、かつ、実行ランタイム機能を含めて実現している。実際のつくりは、単純に mpi.h(mpif.h) で MPI 関連のシンボルを置き換え、変換ライブラリで MPI 関数を呼び出すことにより、シンボルの衝突を避けている。

以上の、既存研究と異なり、MPI-Adaptor のアプローチは個々の MPI プログラムの MPI ランタイム実装はそのままで、バイナリ実行の可搬性を実現して、ターゲットの環境で動かそうとしている点が異なる。変換部の実装という視点で見ると、MorphMPI は 1 対多に対して、MPI-Adaptor は多対多の変換になることが異なる.

8. まとめ

本論文では、バイナリレベルで実行可搬性を実現する MPI 通信ライブラリ (MPI-Adaptor) の設計について述べた。MPI-Adaptor の実現では、MPI の実装毎に異なる引数と返り値の変換が必要になる。これを、動的リンクライブラリを用いて実行環境の非依存性を確保し、MPI-Adaptor が MPI 変換を実現する。

実現の有効性を判断するため、OpenMPI と MPICH2/SCore の MPI 実装間での最小限度の変換機能をもつ MPI-Adaptor のプロトタイプを実装した。簡易評価の結果、Xeon 3.8GHz の PC クラスタで、MPI の変換オーバヘッドは MPI の送受信関数 1 回あたり 0.028μs と小さく、MPI-Adaptor の実現可能性を示した。

今後の予定として、未実装部分の実装、Fortran 対応、MPICH2 から OpenMPI 対応、他 MPI の対応を行う予定である。開発されたプログラムは SCore[7] の distribution に含まれる予定である。

謝辞

本研究は、文部科学省「e サイエンス実現のためのシステム統合・連携ソフトウェアの研究開発」からの支援を受けている。

参考文献

1) RSCC: RIKEN Super Combined Cluster System: http://w3cic.riken.go.jp/rscc/.
2) T2K Open Supercomputer Alliance: http://www.open-supercomputer.org/.
3) MPICH2: http://www.mcs.anl.gov/research/projects/mpich2/.
4) OpenMPI: http://www.open-mpi.org/.
5) The Message Passing Interface (MPI) standard: http://www.mpi-forum.org.
6) Application Binary Interface Working Group: https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/AbiWikiPage.
7) SCore Cluster System Software: http://www.pccluster.org/.
8) MorphMPI: http://morphmpi.sourceforge.net/.