



Powered by **Score**

MPI-Adapter: MPIバイナリ互換を 実現するミドルウェア

富士通研究所
住元 真司

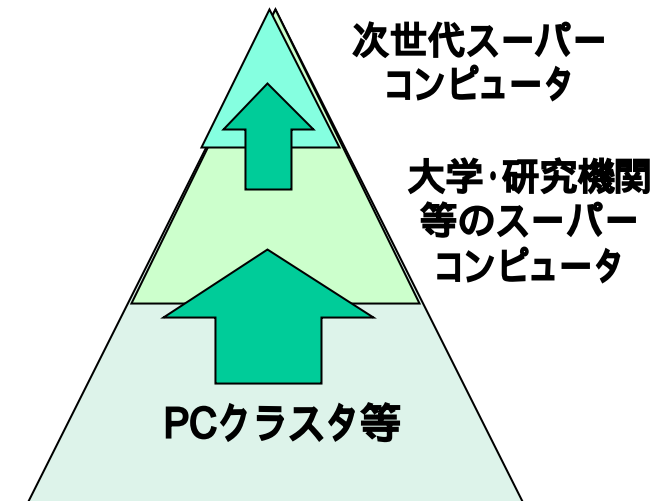
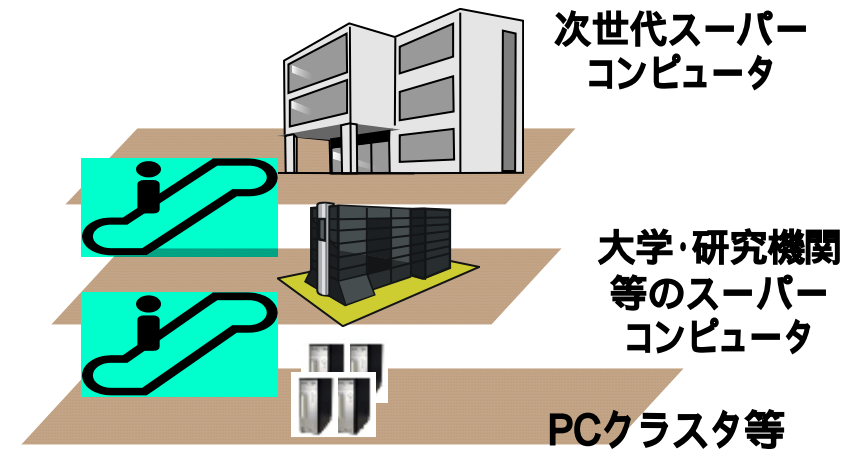


発表の概要

- eScienceプロジェクトの概要
 - 高性能並列プログラミング言語処理系
 - 高生産並列スクリプト言語
 - 高効率・高可搬性ライブラリの開発
- MPIバイナリ互換を実現するミドルウェア
MPI-Adapter
 - Motivation、目標、MPI-Adapterの活用方法
 - MPI-Adapterの技術概要、性能評価
 - SCore 7.0Beta3 でのMPI-Adapterの利用方法

eScienceプロジェクトのめざすもの

- 現状： PCクラスタの利用裾野の広がり 研究室単位での導入
 - さまざまな並列アプリの研究開発が実行されている。
 - しかし、開発されたアプリの可能性が導入クラスタの規模レベルに留まることが多い。
- 目標： 大学、研究機関の研究による並列アプリをより実用的、より大規模に引き上げ、ペタフロップス級の超大規模アプリの開拓



eScienceプロジェクト (Runtime) の課題

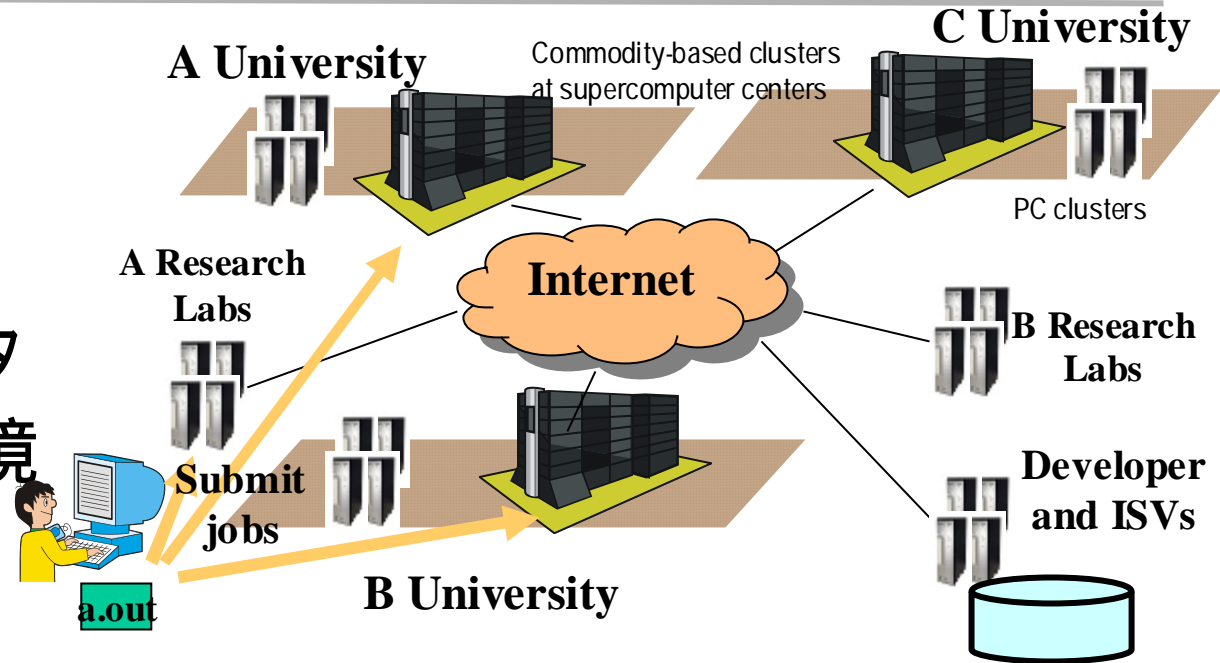
- PCクラスタ毎に実行環境が異なり、ユーザがクラスタ毎の実行環境を意識する必要あり。
 - プログラムの実行の仕方：
 - TSS実行コマンドの違い、バッチスクリプトの記述などの変更が必要
 - 実行プログラムに必要なファイル群へのアクセス：
 - 必要なファイル群を個別に手動での転送が必要
 - 実行プログラムのコンパイル：
 - ソースファイルからの再コンパイルが必要
- MPI-Adapter: ユーザに実行プログラムの再コンパイル不要な環境を提供するために開発

MPI-Adapter開発のMotivation

- PCクラスタの広がり:
 - 計算機センタ (RSCC, T2K)、研究室クラスタ
- 利用方法が拡大:
 - インターネットから様々なクラスタを選択利用
- PCクラスタ毎のMPI実行環境の違い
 - 実行環境毎にソースからのコンパイルが必要
 - 例: T2Kマシンはハードウェア構成がほぼ同等にも関わらず、相互にMPIプログラムを実行する場合再コンパイルが必要
- もっと簡単に、複数のPCクラスタを自由自在に使えないか？

MPI-Adapterの目指すもの： シームレスな MPI 実行環境の提供

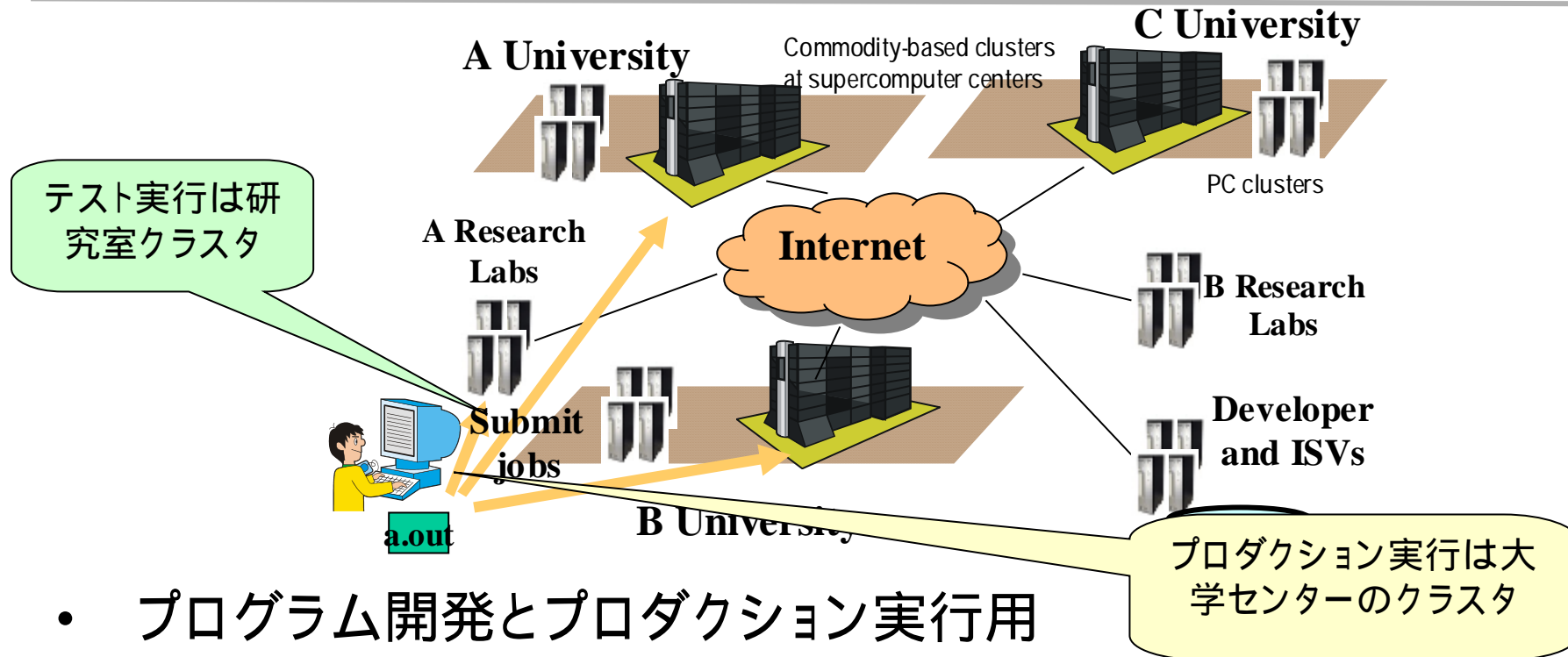
- 目指すもの：
PCクラスタで作成
のMPIバイナリは、
他のどのPCクラスタ
でも実行可能な環境
を実現する



- Use Cases:

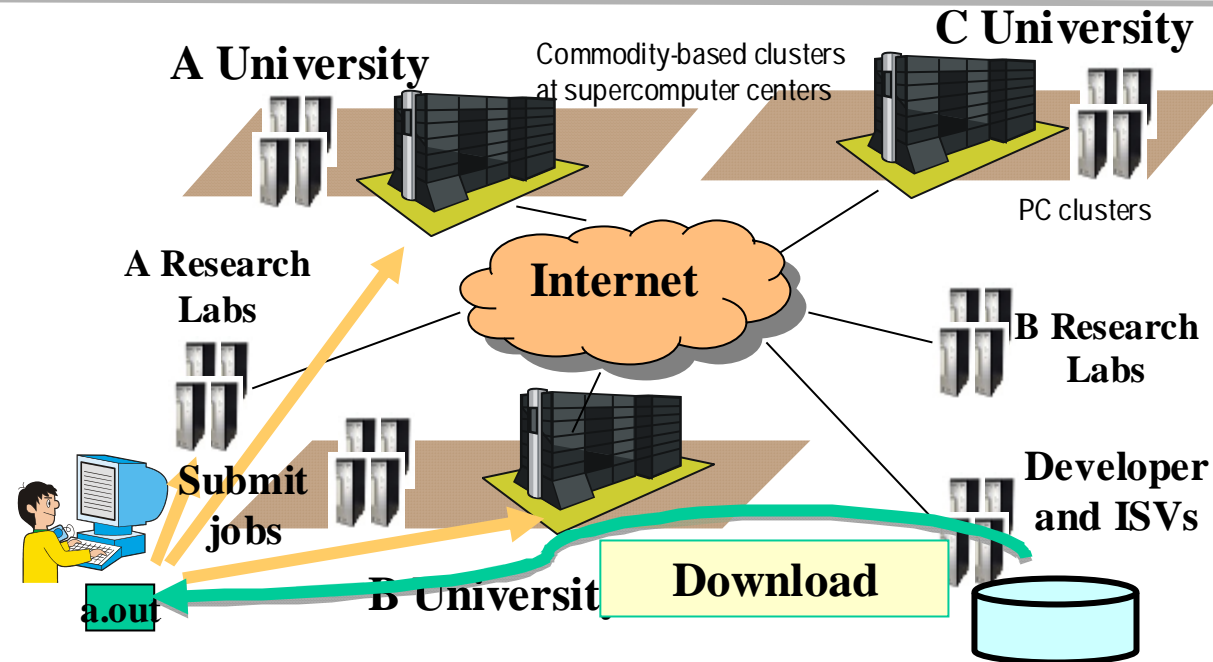
1. 用途によるクラスタの使い分け：
プログラム開発用とプロダクション実行用
2. バイナリ配布： ISV向け、コンパイル不要
3. クラスタ環境の変更： 機能変更、性能向上

Use Case 1:用途によるクラスタの使い分け



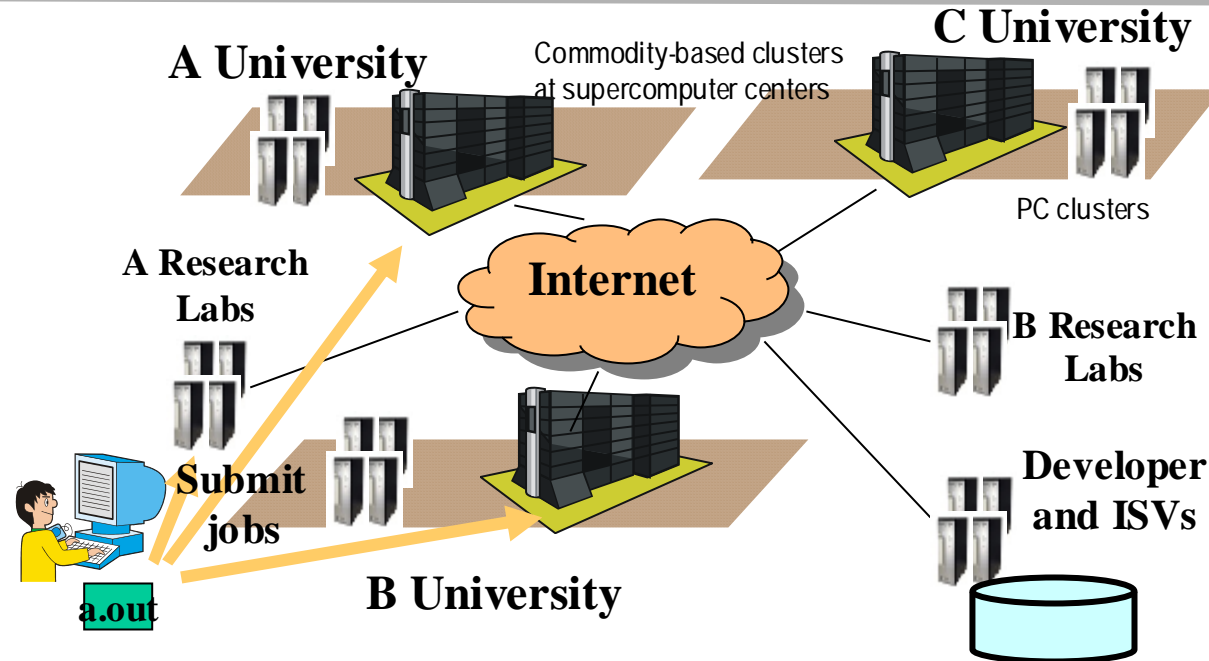
- プログラム開発とプロダクション実行用
 - プログラム開発は、手元のPCクラスタ
 - プロダクション実行用はセンターのPCクラスタ
- 複数のクラスタのシームレスな相互利用:
 - プログラム実行規模と計算リソース、システムの空き具合、プログラム実行時の課金状況により実行するPCクラスタを選択

Use Case 2: バイナリ配布



- 開発したプログラムをMPI実行バイナリとして配布することにより、利用ユーザの利便性を図る
- ISVは一つのMPI実行バイナリの提供： 利用者側の実行環境に合わせてISVプログラムを実行

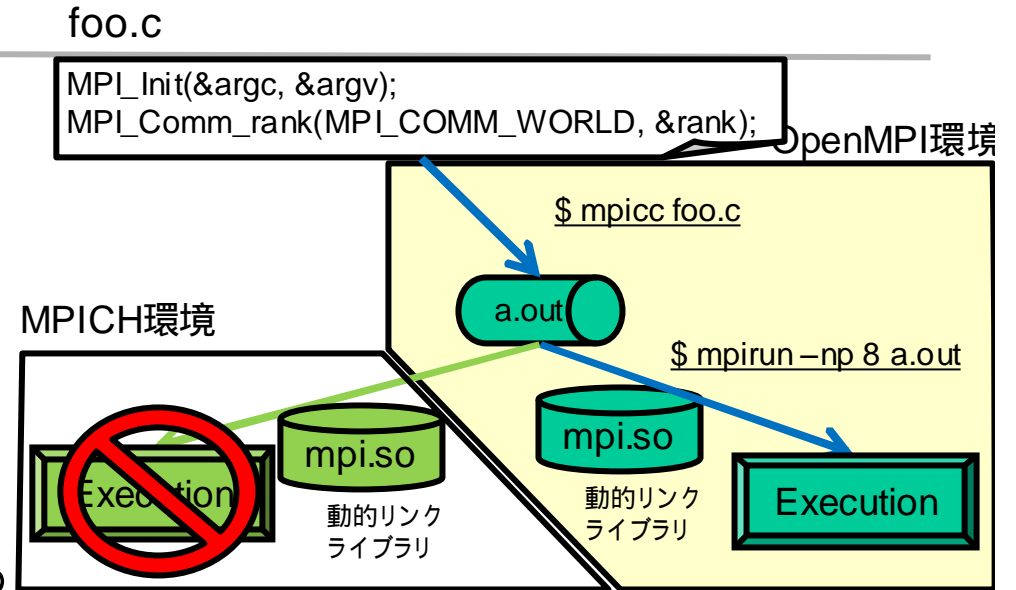
Use Case 3: クラスタ環境の変更



- MPIの機能毎に自由な置き換え(組み合わせ)が可能になる
 - MPIランタイムの個別ユーザの変更は一部の特権ユーザ以外容易ではない、特にプロダクション実行用のクラスタは困難
 - MPIの集合通信のアルゴリズム変更(関数毎のチューニング)
 - デバッグ用のプロファイラの動的挿入、ログ情報の動的挿入を、プログラム本体の再コンパイルなく実施可能

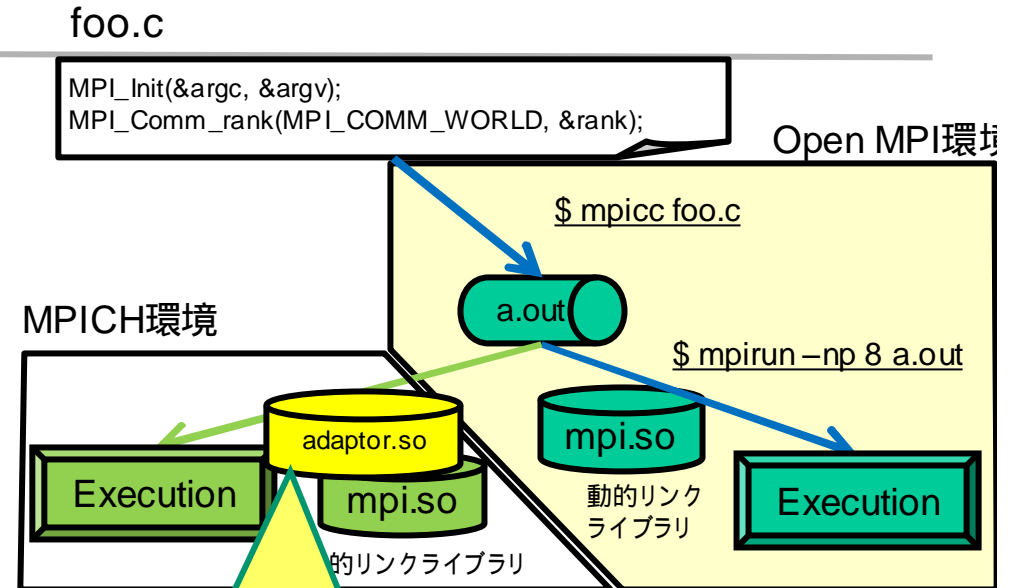
MPI-Adapter実現の課題

- MPI規格書には、ABI (Application Binary Interface) が未定義
- このため、異なる実行環境では正常に動作しない。
- このような場合は、再コンパイルが必要になる。
- 実行可能とするためには、ABIを一致させる必要がある。



MPI-Adapterの動作

- MPI-Adapterを2つの異なるMPI実装の間に挿入しABIの整合性を確保
 - ユーザはMPI実行環境の違いを意識しないで、異なるクラスタ上でのプログラム実行可能



```
int MPI_Comm_rank(int comm, int *p)
{
    int cc;
    void *ocomm = convMPI_Comm(comm);
    call_OpenMPI(&cc, "MPI_Comm_rank", ocomm, p);
    return cc;
}
```

ABI違いの原因：MPI定義の型定義

- MPIの仕様書にはいくつかの型が定義
- MPIの実装時にどのように型を実現するかは実装依存
 - 例： 数値での実装、構造体での実装
- この実装の違いが、バイナリの可搬性の障害になっている

用途	型 (値例)
Communicator	MPI_Comm (MPI_COMM_WORLD)
Group	MPI_Group
Request	MPI_Request
Status	MPI_Status
Data type	MPI_Datatype (MPI_Int,)
Operation	MPI_Op
Window	MPI_Win
File	MPI_File
Info	MPI_Info
Pointerの差分	MPI_Aint
Offset	MPI_Offset
Error Handler	MPI_Errorhandler

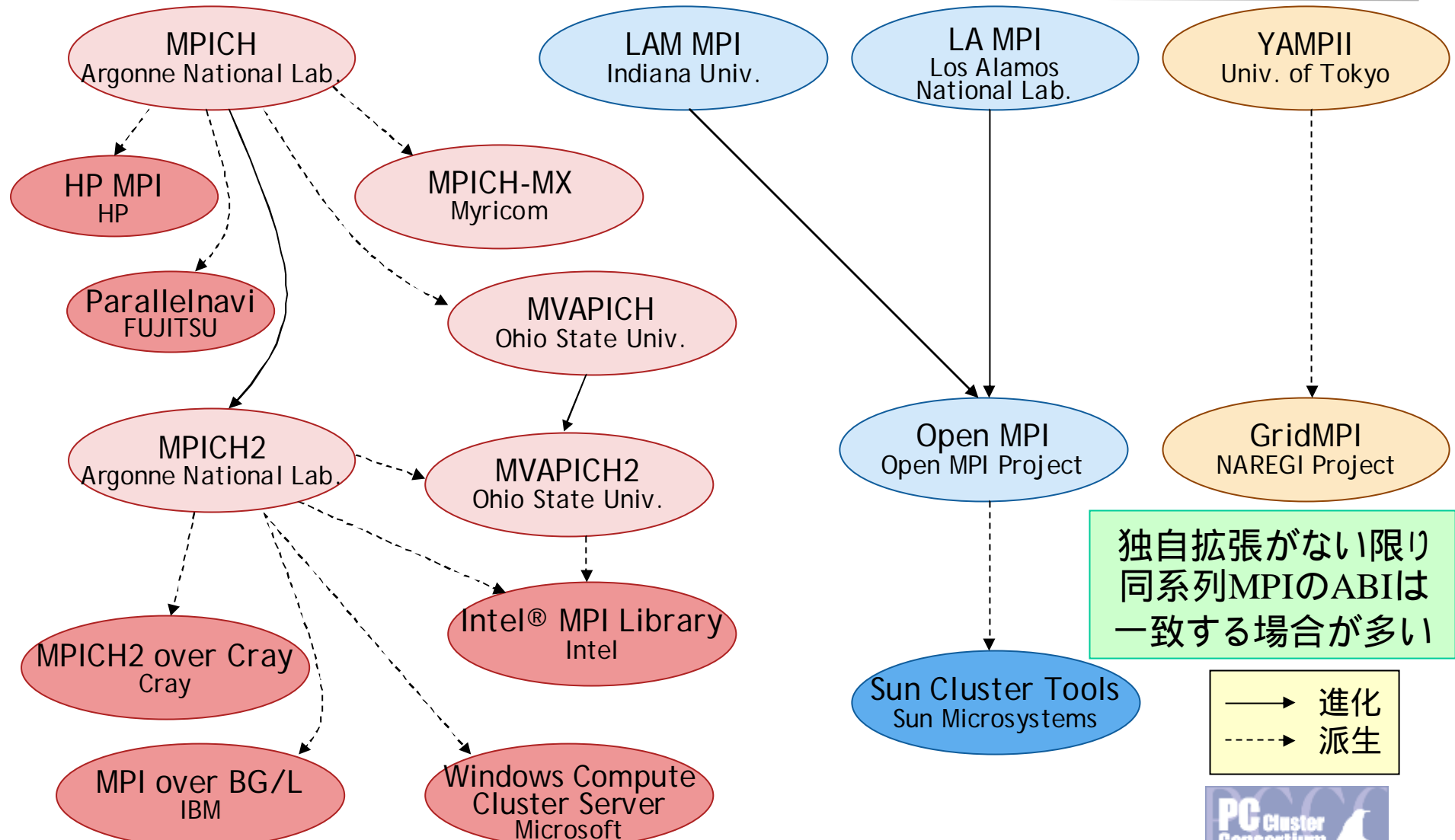
ABI違いの原因：MPI定義の型定義

OpenMPIとMPICH2の実装例

	用途	MPICH2	OpenMPI
C言語	MPI_COMM_WORLD	0x44000000	&ompi_mpi_comm_world
	MPI_INT	0x4c000405	&ompi_mpi_int
	MPI_INTEGER	0x4c00041b	&ompi_mpi_integer
	MPI_SUCCESS	0	0
	MPI_ERR_TRUNCATE	14	15
Fortran言語	MPI_COMM_WORLD	0x44000000	0
	MPI_INTEGER	0x4c00041b	7
	MPI_SUCCESS	0	0
	MPI_ERR_TRUNCATE	14	15

- OpenMPIとMPICH2ではバイナリ可搬性がない
 - MPICH2は32bit INTを用いた実装
 - OpenMPIは構造体を主体とした実装
- Fortranは32bit INTを用いた実装

ABI互換の適用性： MPIライブラリの系譜

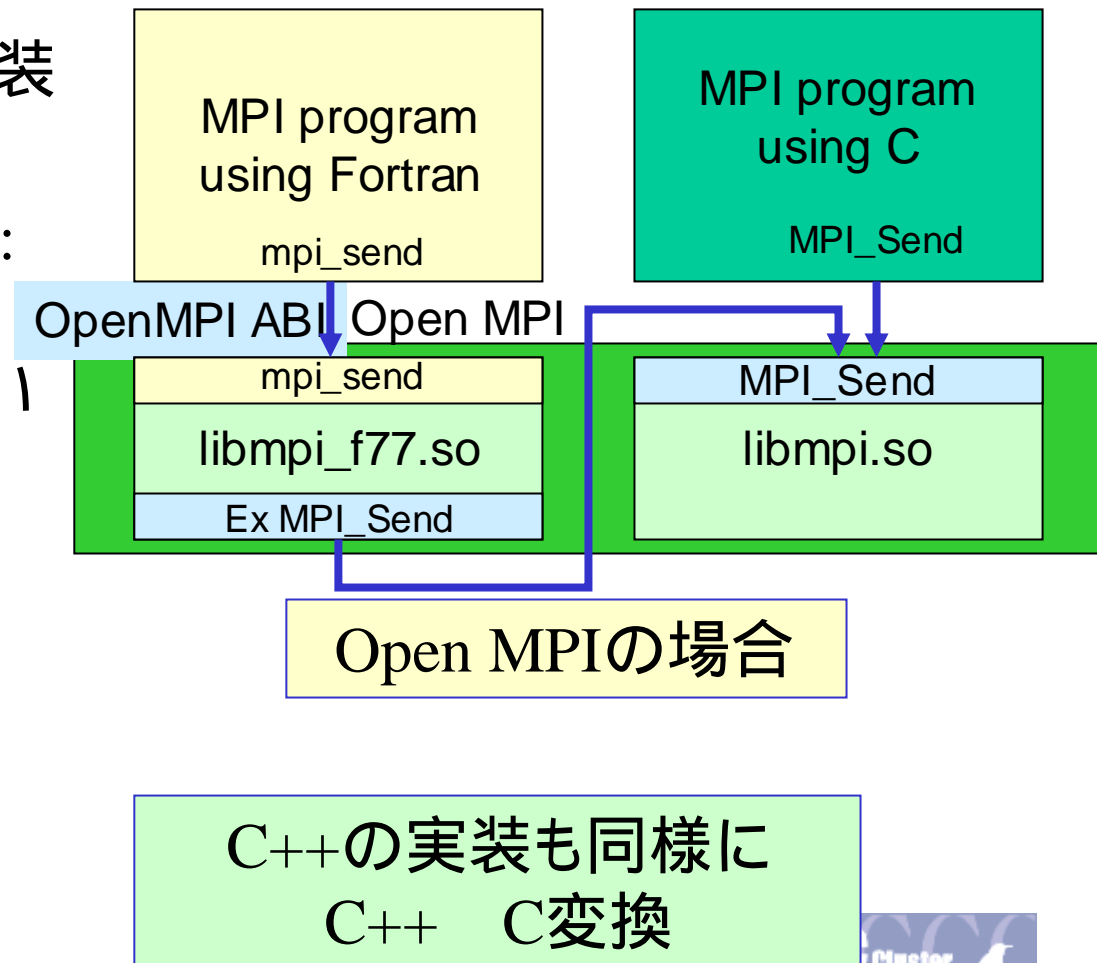


MPI-Adapterの概要

- 簡単な設定で利用可能
- MPI-Adapterのオーバヘッドは無視できる程度
- 多くのMPIランタイムに適用可能
 - 現状、Open MPI, HP MPI MPICH, MPICH2での動作を確認
 - ターゲットのMPIランタイムのヘッダより自動的にABI情報を抽出

MPIライブラリの実装構造例： Open MPIの場合

- C用のMPIライブラリ：
MPIライブラリ本体を実装
- Fortran用MPIライブラリ：
Fortran Cコンバータ
(F2C)による実装が多い
- MPICHの場合
 - libmpich.so (Fortran)
 - libmpich.so (C)



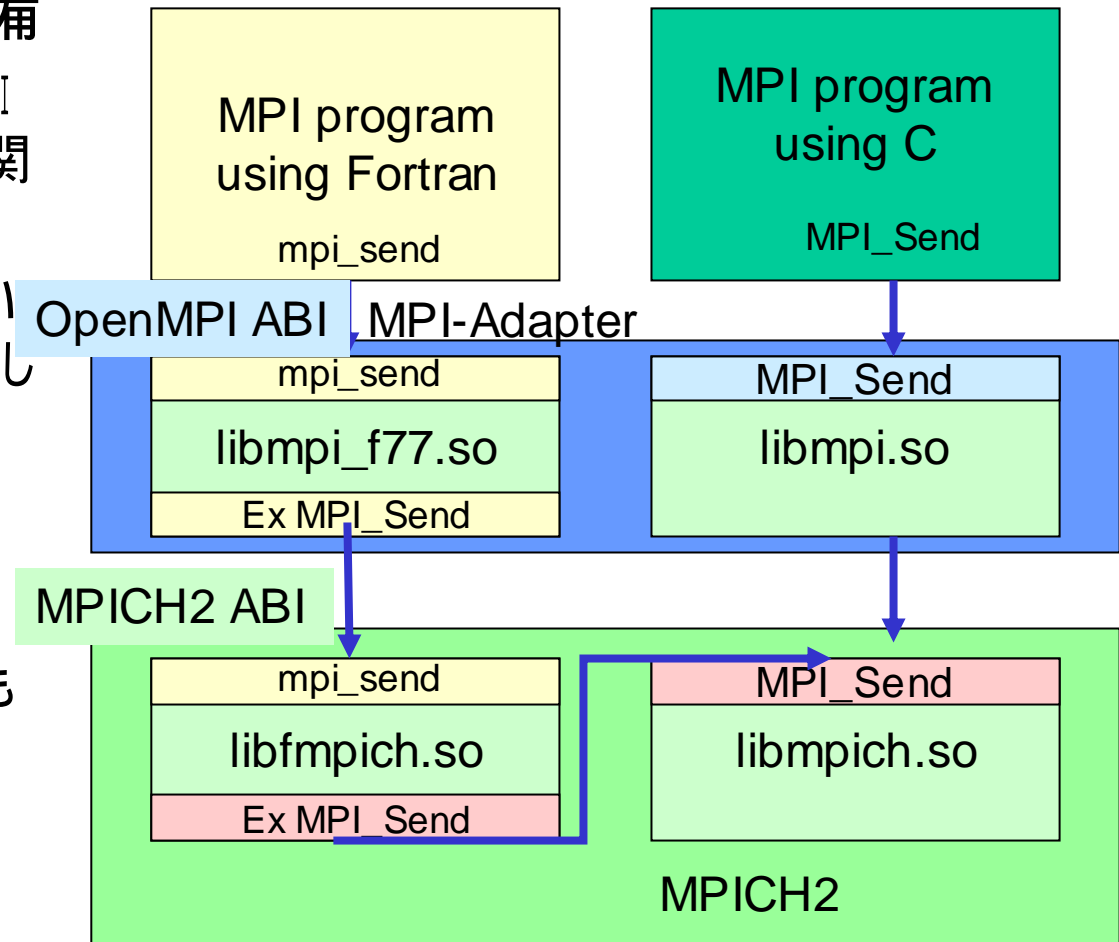
MPI-Adapterの実装と動作

- 変換手順: libmpi.so.0 準備

- MPI_Init時にターゲットMPIのライブラリをdlopen()して関数変換テーブルを作成
- 関数実行時に型変換を行い、関数テーブル経由で呼出し
- 返値も変換して返す

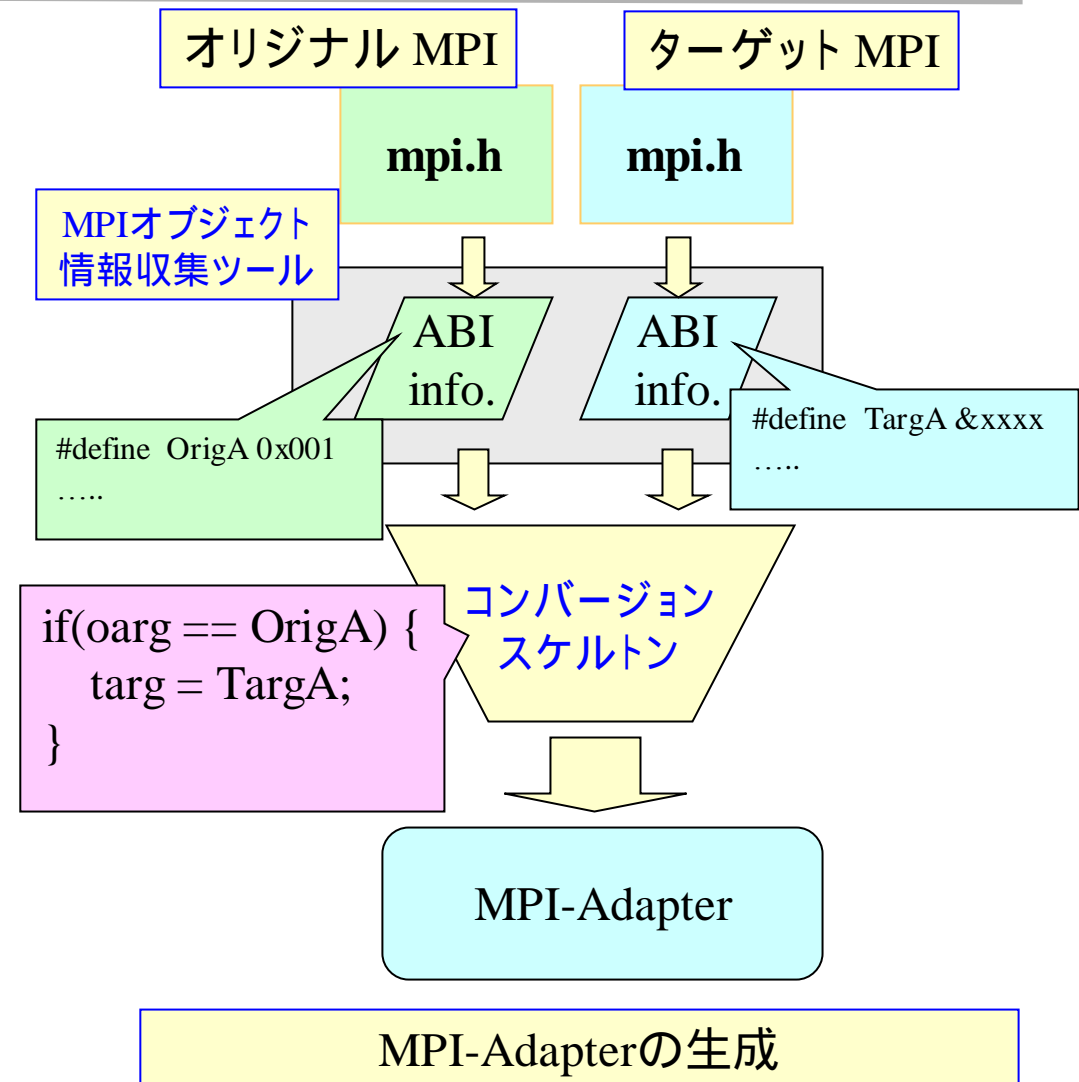
- Miscライブラリ対応

- OpenMPIのmiscライブラリもダミーを準備 (libopen-rte.so.0, libopen-pal.so.0)



複数のMPIランタイムからの MPI ABI変換情報の自動抽出

- MPIオブジェクト情報収集ツールによりMPIヘッダ(mpi.h, mpif.h)からABI情報を抽出する
 - 一つのMPIランタイムにつき一つのABI情報
- 収集したABI情報から2つを選択してMPIコンバージョンスケルトンに導入することによりMPI-Adapterを生成.



MPI-Adapter評価

- MPI-Adapter挿入オーバヘッド評価
 - MPI通信遅延 (共有メモリ利用)
 - NAS並列ベンチマーク (Fortran, Class A,B,C)
- 実行環境
 - RX200S2クラスタ(Xeon 3.8GHz, SCore7.0)
 - ネットワーク : Intel E1000 NIC,
Netgear 48Port Switch
 - 利用MPI MPICH2/SCore w/ PMX/Etherhxb,shmem

MPI遅延によるオーバヘッド評価

MPI-Pingpong(mpi_rtt)プログラムによる測定, PMX/Shmem利用

usec	Fortran	C	Overhead (/MPI call)
Open MPI+ MPI-Adaptor	3.154	3.065	0.082(0.022)
MPICH2/SCore	3.103	3.055	0.048(0.012)
Overhead(/MPI)	0.051(0.013)	0.010(0.0025)	0.034(0.0085)


- Fortran to C ABIトランスレータのオーバヘッド
 - MPICH2=0.012usec, Open MPI=0.022usec
- MPI-Adapterのオーバヘッド(Open MPI MPICH2)
 - Fortran (INT to INT)=0.013usec, C (Pointer to INT)=0.0025usec
- 総じて、挿入オーバヘッドは20ns程度と極めて小さい

単位: usec

NPBによる評価(性能劣化率)

16x1の結果、Fortranはgfortran

16 node	BT	CG	FT	LU	MG	SP	IS
Class A	-0.1%	3.4%	0.2%	0.4%	-3.7%	0.0%	-0.4%
Class B	-2.2%	3.7%	2.9%	-0.5%	-0.8%	1.6%	4.8%
Class C	0.3%	1.2%	0.4%	0.1%	-5.0%	0.6%	0.3%

- C版のMPI-Adapterと同様の傾向  性能向上
 - 最大で3.7%のオーバヘッド
 - MPI-Adapter挿入時の方が速い場合あり (-5.0%)
- 性能の違いは直接的なものではなく、間接的なもの

MPI-Adapterの利用

- SCore 7.0Beta3でのMPI-Adapterのバージョンが含まれています。
 - 動作バージョン: Open MPI MPICH2/SCoreのみ
- 利用法: コマンドラインにmpi-adapterを挿入
 - % mpirun **mpi-adapter** [-T src2dest] mpi_program arg...
 - src は mpi_program をコンパイルした mpi
 - dest は mpi_program を実行する mpi
 - Beta3 で動作するのは ompi2mpich_score のみです。
 - -T を指定しない場合の default は ompi2mpich_score

MPI-Adapterの応用

- 開発技術
 - DLL挿入による同名関数のフッキング
 - ライブラリの一部関数の挙動を変更可能
 - 実行時関数スワップ方式による関数呼び出し変更
 - 実行時に動的に関数の呼び出し先を変更可能
- 応用
 - 実行時プロファイラ、デバッグ、チューニングツール
 - 通信処理の最適化と機能拡張
 - 集合通信の最適化： プロファイラとの連携
(アルゴリズムの動的変更、静的動的)
 - IMPI-Adapter： 複数のMPIランタイムを切り替え利用

まとめ

- MPIバイナリのABI互換性を実現するMPI-Adapter
 - 仕組み、動作、概要について説明
 - PCクラスタ利用をより便利にするツール
 - さまざまな、応用が可能で適用範囲が広がる
- 今後の予定
 - MPIのテストスーツによる網羅的な品質向上
 - 利用環境の充実
 - MPI_{xx}のバージョン毎の違いの吸収(ex. Open MPI)
 - MPIバイナリの自動検出
 - 実運用で実績を積む
- 謝辞: 研究は、eScienceプロジェクトの一部として実施された

参考文献

- 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 (情報処理学会 HOKKE 2009, 2009-ARC-182(20) 2009-HPC-119(20), pp.115-120, 2009.)
住元真司, 中島耕太, 成瀬彰, 久門耕一(富士通研), 安井隆(日立), 鴨志田良和, 松葉浩也, 堀敦史, 石川裕(東大)
- シームレスなMPI環境を実現するMPI-Adapterの設計と性能評価 (情報処理学会 SWoPP 2009, 2009-HPC-121(12), pp. 1-8, 2009)
住元真司, 中島耕太, 成瀬彰, 久門耕一(富士通研), 安井隆(日立), 鴨志田良和, 松葉浩也, 堀敦史, 石川裕(東大)
- The Design of Seamless MPI Computing Environment for Commodity-based Clusters (Euro PVM/MPI 2009, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, Volume 5759/2009, ISBN 978-3-642-03769-6, Page 9-19)
Shinji Sumimoto, Kohta Nakashima, Akira Naruse, Kouichi Kumon, Takashi Yasui, Yoshikazu Kamoshida, Hiroya Matsuba, Atsushi Hori and Yutaka Ishikawa (2009 Best Papers)

Thank you.