MPI Forum Japan Meeting
Tools WG: MPI Adapter

Shinji Sumimoto
(Fujitsu Laboratories Ltd.),

Outline of This Presentation

- Background
- Seamless MPI Environment
  - Issues of keeping MPI ABI compatibility
  - MPI-Adapter Approach and its Related Work
- Design and Implementation of MPI-Adapter
- Evaluation
- Future Works of MPI-Adapter

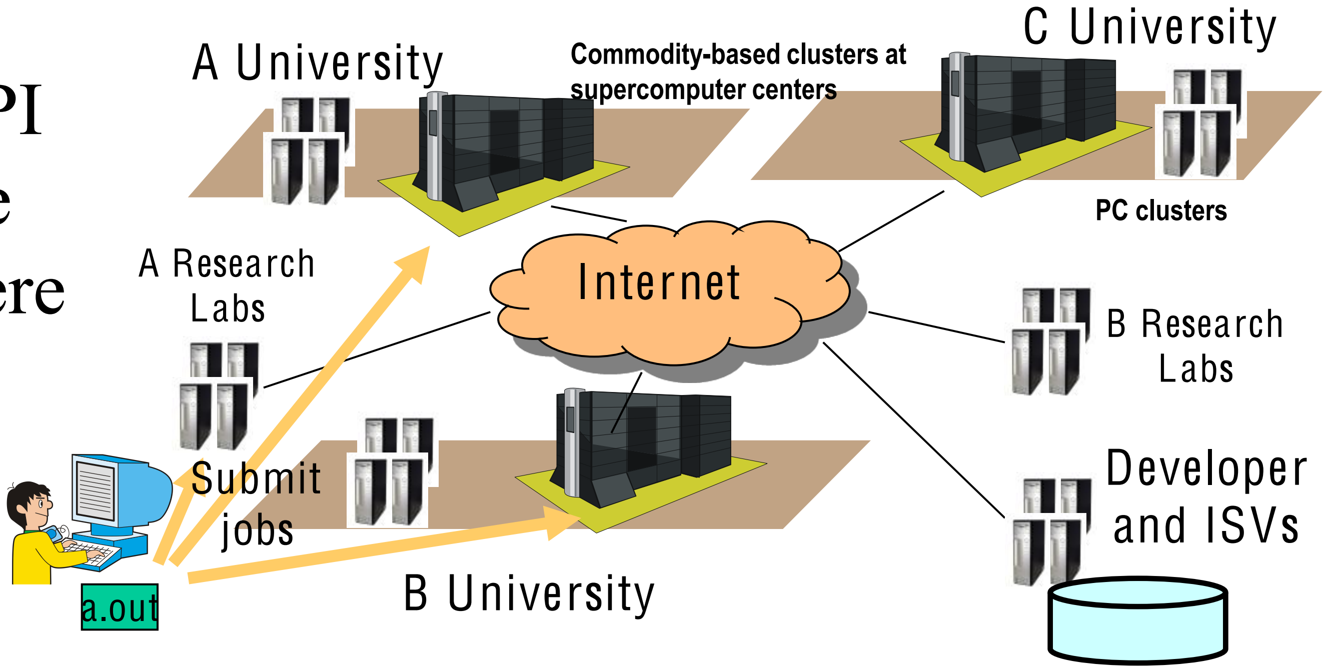
Background

- Commodity-based clusters are widely used for high – performance computing.
  - RSCC, Tsubame, T2K、RICC, etc. in JAPAN
- Users can use several clusters through the Internet.
- However, users must re-compile their program even if using PC clusters (x86 and Linux).
  - This limitation does little to expand PC cluster use.

- ABI compatibilities should be realized on PC Clusters.
  - Seamless MPI Computing Environment

Seamless MPI Computing Environment

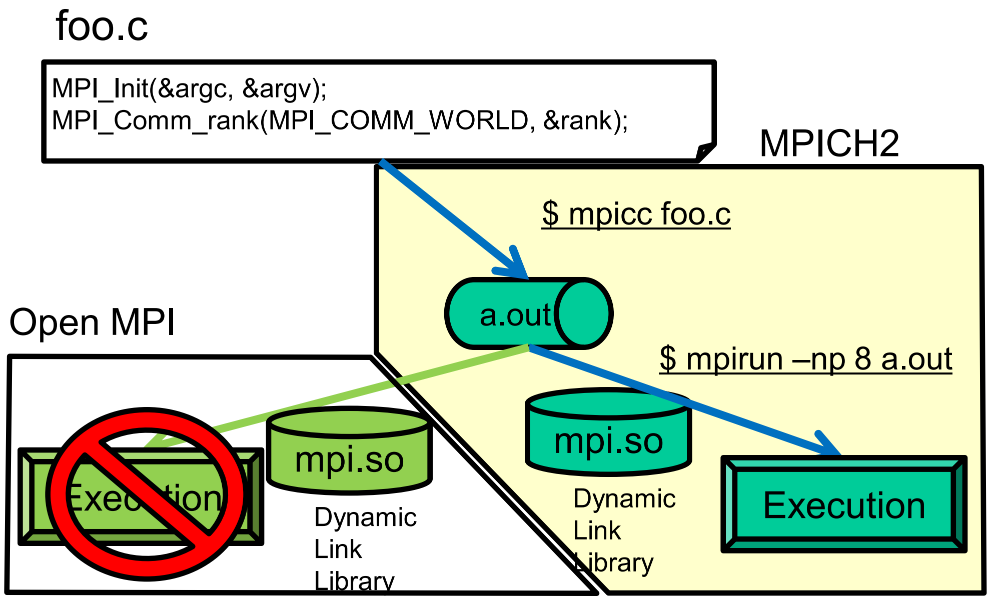
- Goal: Same MPI binaries are able to run everywhere on PC Cluster.



- Use Cases:
  - <u>Selecting Clusters</u>: for development and production
  - <u>Binary Distribution</u>: for ISV and developper
  - <u>Changing Runtime Environment</u>: for Functionality and performance issues

Issue of Seamless MPI Computing Environment

- MPI standard does not define MPI application binary interface (ABI)
  - Ex. MPI_Comm type
Open MPI: address type
MPICH2: 32 bit integer



foo.c

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

- Issue: Providing a mechanism to keep ABI among PC clusters

Objects and Type Definitions on MPI Standard

- MPI standard defines several MPI objects and type definitions.
- Implementation of them depends on MPI runtime.
  - – The differences are the reason of lack of ABI compatibility.

Objects	Types（a pre-defined value）
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_MAX)
Window	MPI_Win
File	MPI_File
Info	MPI_Info
Pointer diffs.	MPI_Aint
Offset	MPI_Offset
Error Handler	MPI_Errorhandler

Differences of Pre-defined Values between MPICH2 and Open MPI

	Pre-defined Values	MPICH2	Open MPI
C Lnag.	MPI_COMM_WORLD	0x44000000	&ompi_mpi_comm_world
	MPI_INT MPI_INTEGER	0x4c000405 0x4c00041b	&ompi_mpi_int &ompi_mpi_integer
	MPI_SUCCESS MPI_ERR_TRUNCATE	0 14	0 15
Fortran	MPI_COMM_WORLD	0x44000000	0
	MPI_INTEGER	0x4c00041b	7
	MPI_SUCCESS MPI_ERR_TRUNCATE	0 14	0 15

- No ABI compatibility between MPICH2 and Open MPI
  - MPICH2: 32bit INT based implementation
  - Open MPI: Structure based implementation
- In Fortran implementation, 32 bit implementation, but values are different between MPICH2 and Open MPI

Difference of MPI_Status Structure

- MPI_Status structure implementation is also different among MPI implementations.
  - Location and Symbols are different between Open MPI and MPICH2.

```
struct ompi_status_public_t {
    int MPI_SOURCE;
    int MPI_TAG;
    int MPI_ERROR;
    int _count;
    int _cancelled;
};
```

Open MPI

```
typedef struct MPI_Status {
    int count;
    int cancelled;
    int MPI_SOURCE;
    int MPI_TAG;
    int MPI_ERROR;
};
```

MPICH2

Differences of MPI Implementations

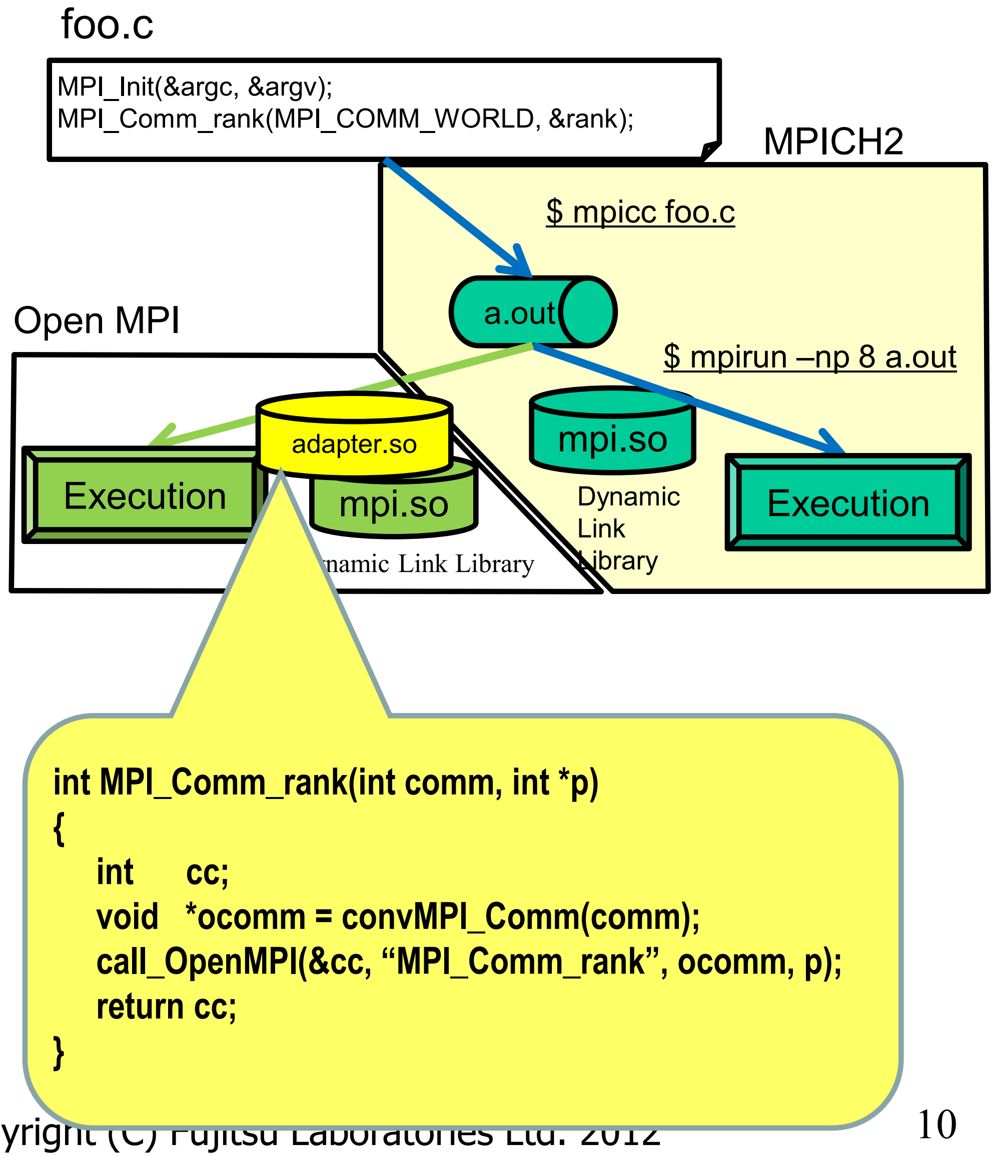
Survey of ABI Working Group (MPI Forum)

	Differences
Intel MPI	MPICH2 based (Integer)
MS MPI	MPICH2 based (Integer)
HP MPI	Original (Structure Based)
LAMPI	Original (Integer and Structure)
NEC MPI	Original? (Integer)

- Two Groups: Integer, Structure or Combination of Integer and Structure Based Implementation

Realizing MPI ABI Compatibility

- Our Approach: MPI-Adapter Translation
  - Inserting MPI-Adapter between two different MPI distributions.
  - Dynamic Link Library Based
  - No need to modify Application Binaries and MPI Runtime Libraries



```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

```
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;
}
```

Related Work

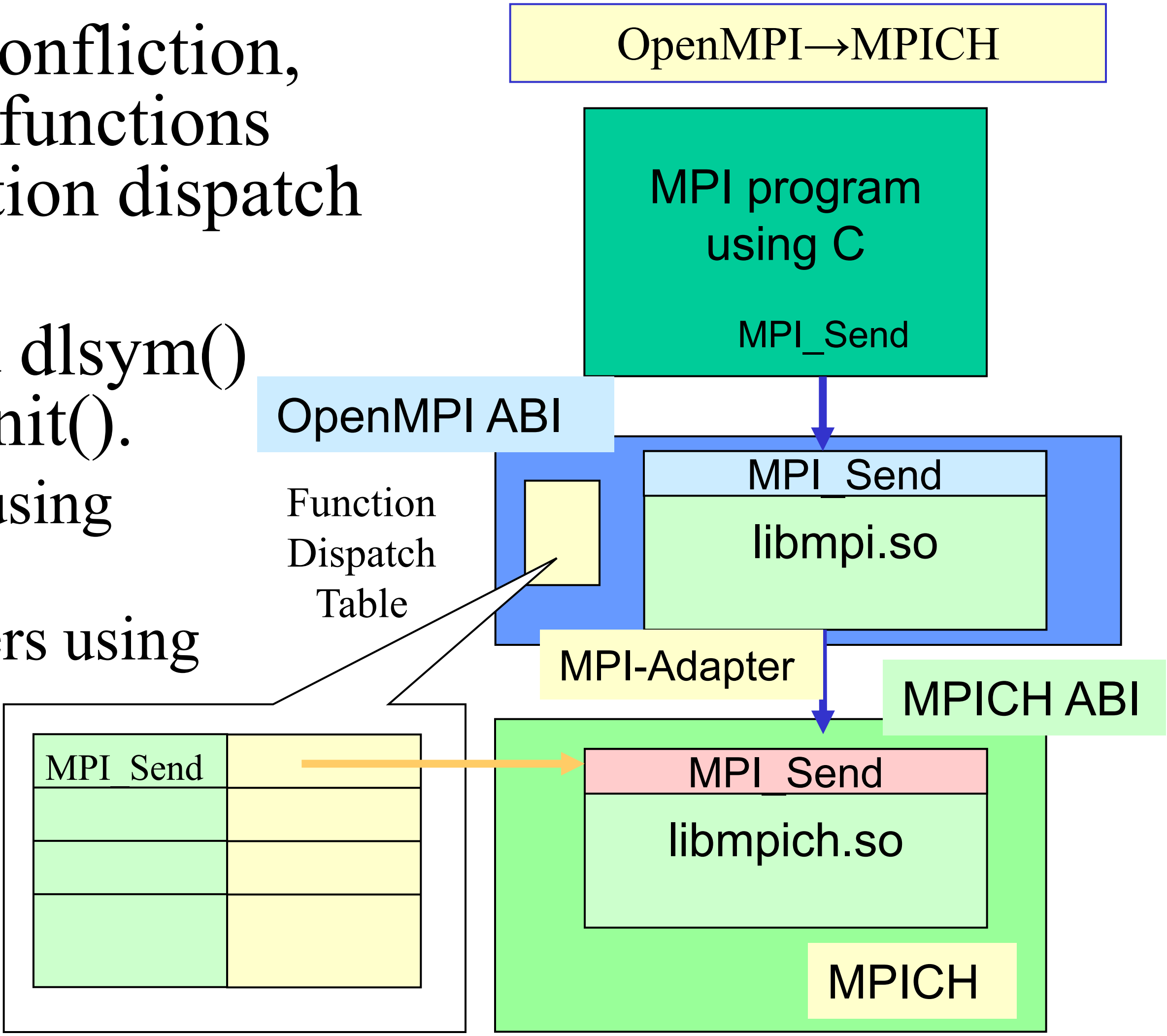
- ABI Working Group (for MPI3.0, MPI Forum):
  - Trying to specify the MPI ABI. Significant Work.
  - After defining the unified MPI ABIs, several years will be needed to implement them and widely used in the world.
- Morgh MPI and GMPI (W. Gropp, 2002):
  - Providing a generic MPI headers
  - Users must re-compile to use generic header.
- MPI-Adapter:
  - No need to modify Application Binaries and MPI Runtime Libraries
  - Does not support static linked binaries

Design of MPI Adapter

- Dynamic Linked Library Based
  - Switched by using LD_LIBRARY_PATH
- Realizing Objects and Types Translation
  - Pointer and Integer
  - MPI_Status structures must be translated
- Issues:
  - How to call target MPI libraries with same symbol?
    - Same function symbols on both MPI-Adapter and Target MPI
    - A Problem with Fortran Libraries
  - How to translate MPI ABI among several MPI implementations automatically?
    - A lot of combinations among MPI implementations: $O(N^2)$

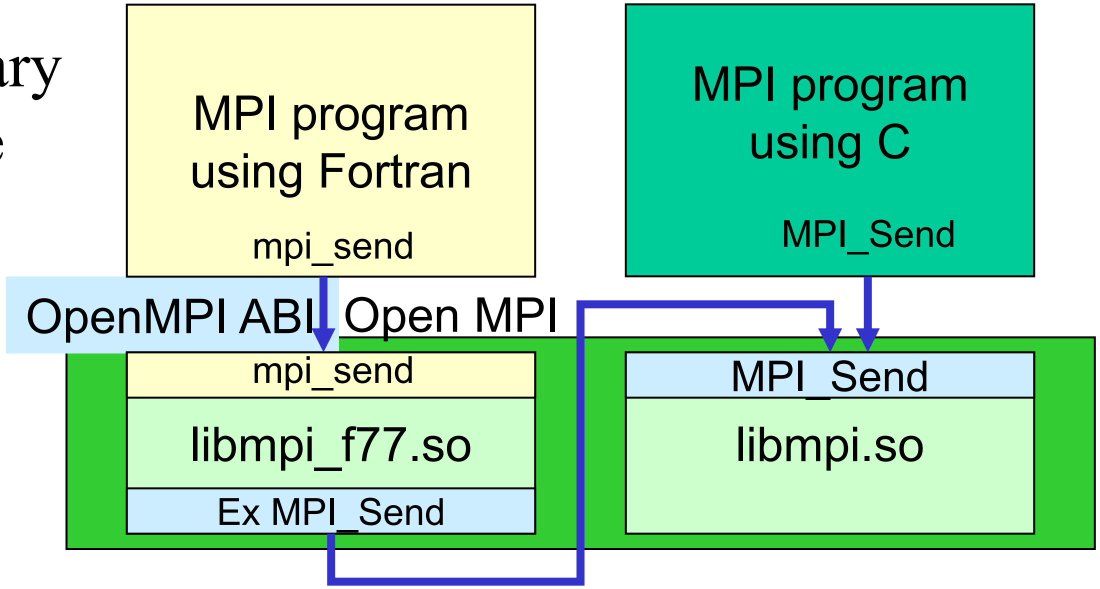
How to call a target MPI library?

- Avoiding symbol confliction, MPI-Adapter calls functions directly using function dispatch table.
- Using dlopen() and dlsym() functions at MPI_Init().
  1. Open libmpich.so using dlopen().
  2. Get function pointers using dlsym().
  3. Store the function pointers to function dispatch table.



C and Fortran Libraries of Open MPI

- MPI Library for C Language and MPI Library for Fortran Language are implemented as separate library for each.
- Open MPI Case:
  - libmpi_f77.so(Fortran)
  - libmpi.so (C)
- MPICH Case:
  - libfmpich.so (Fortran)
  - libmpich.so (C)



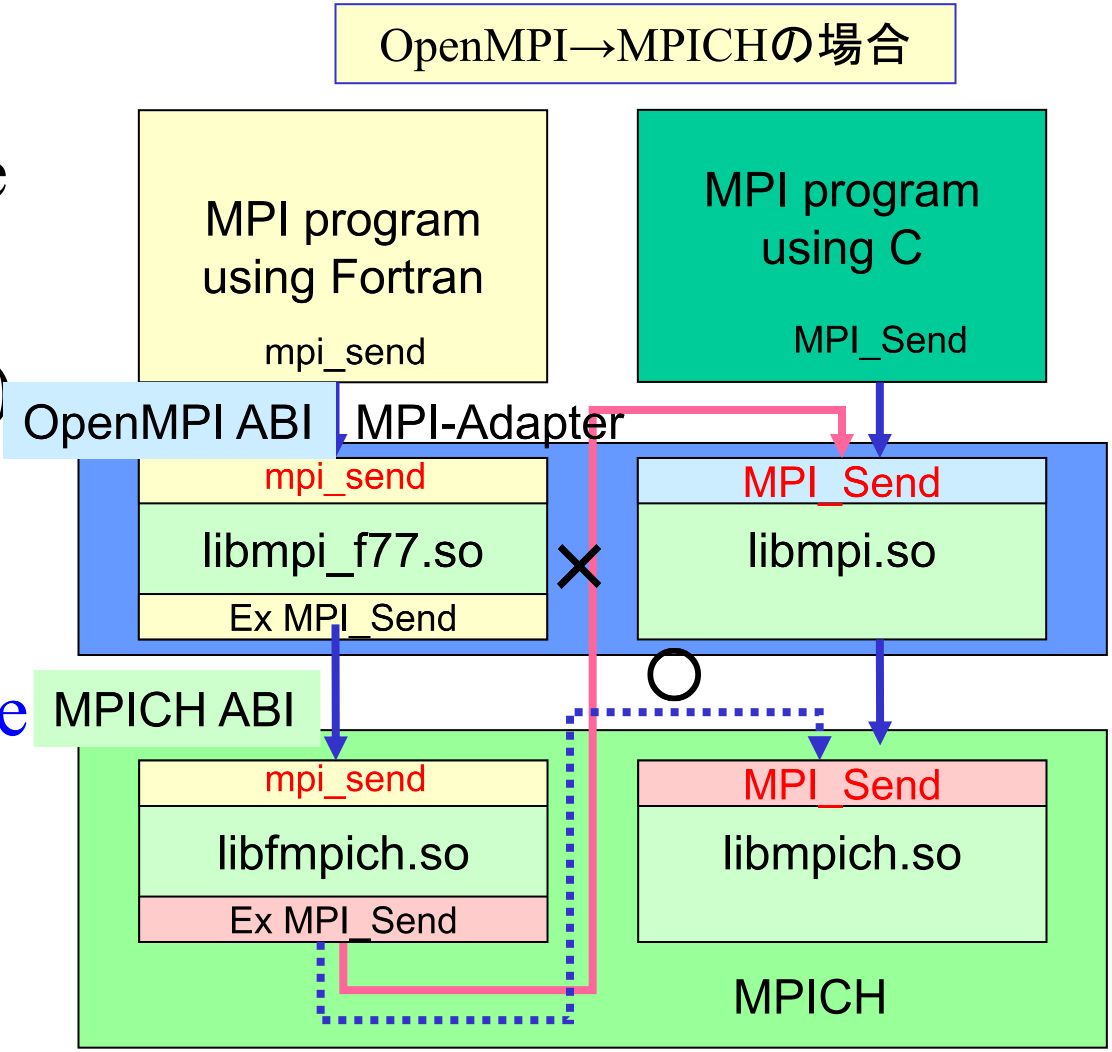
For Open MPI

Same as C++
C++→C

A Problem of MPI Fortran Library

- Functions which have the same names with target MPI library are handled by dlopmen() and dlsym().

- However, functions which are called in the target libraries use original MPI libraries



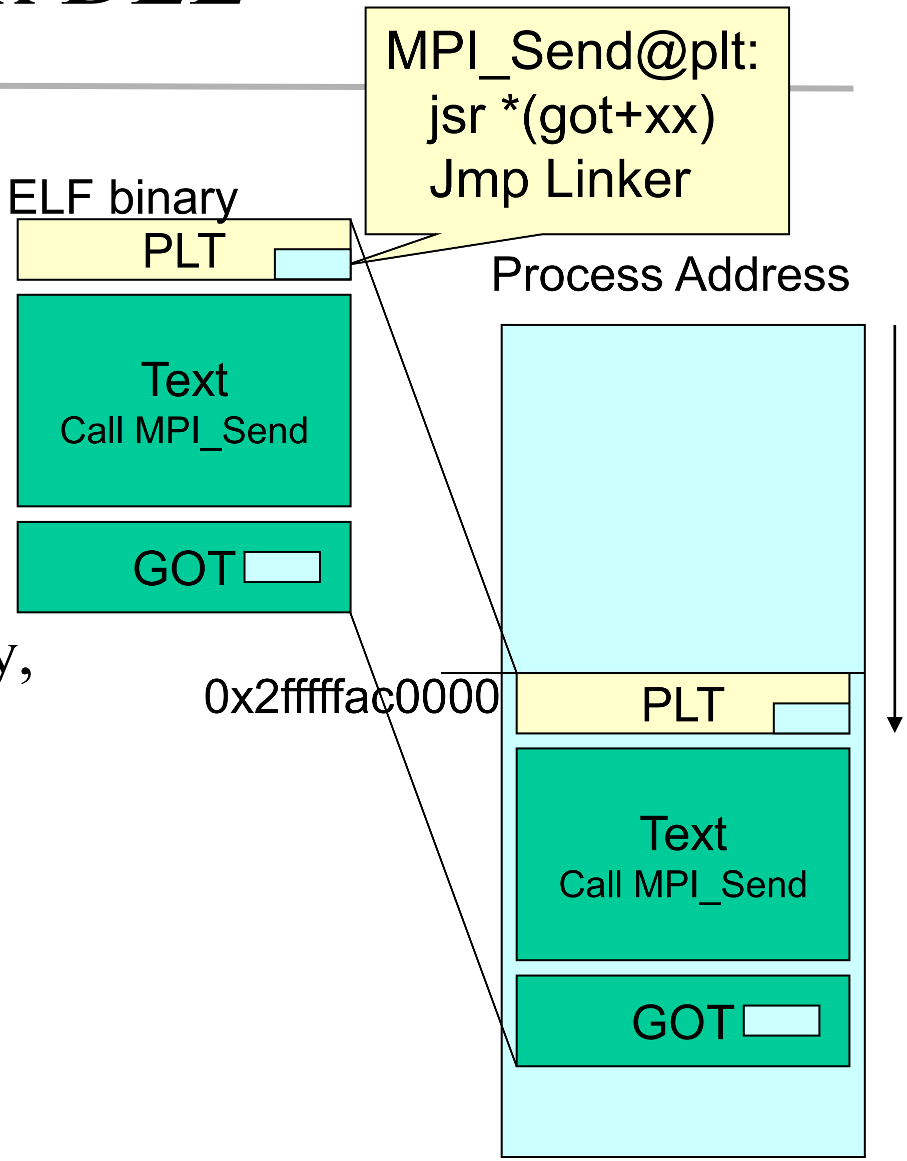
MPI-Forum 2012 in Tsukuba

A Solution To Fix the Problem

- Modifying Call Address Table of DLL （Dynamic Link Library）
  Using Linux DLL Mechanism

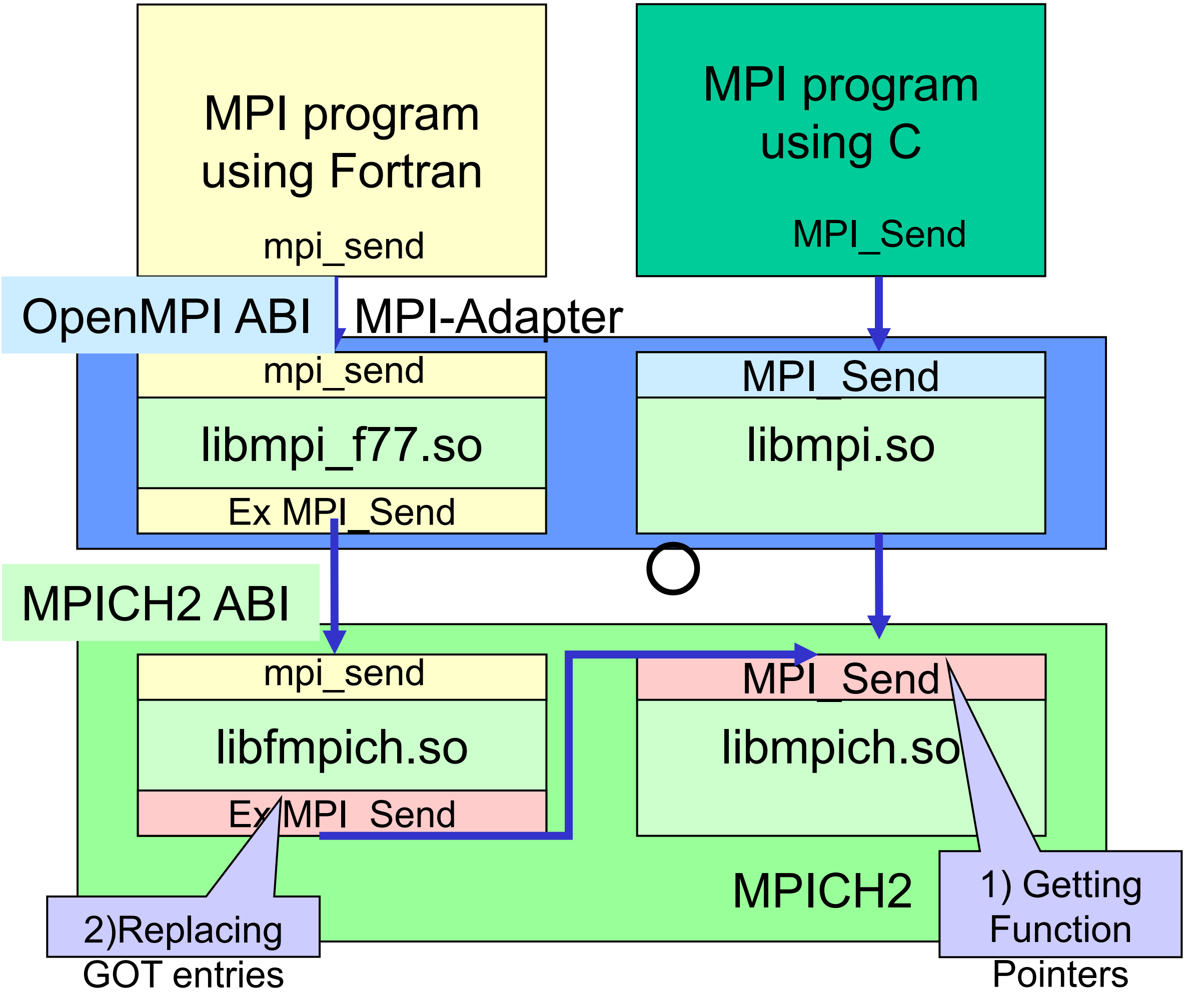
Inside of Linux DLL

- PLT and GOT
  - PLT(Procedure Linkage Table): A call address is fixed using dynamic linker of Linux at first function call
  - GOT(Global Offset Table): After initialization of the library, GOT values are set to the next address of jsr instruction to call Linux linker.
- Linux loader(ld-so) fixes the function address using address table of the process.



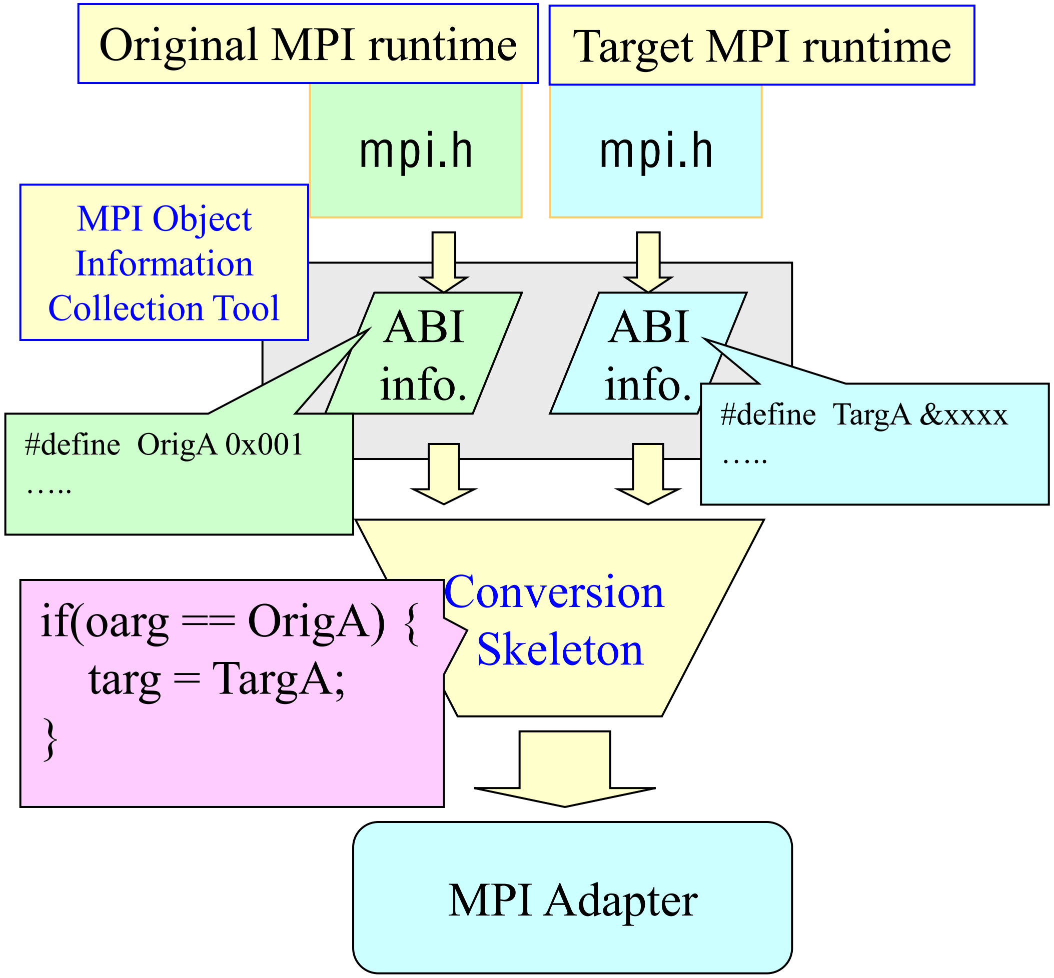
The Solution of the Problem

- MPI-Adapter replaces GOT Table entries to those of the target MPI libraries.
  - １）Getting function pointers
  - ２）Replacing GOT entries.



How to Translate MPI ABI among several MPI Implementations Automatically?

- Getting ABI information from MPI headers (mpi.h, mpif.h) by using MPI Object Information Collection Tool
- Selecting two MPI ABI information and building MPI-Adapter by using Conversion Skeleton.
  - One ABI info. for one MPI implementation.
  - O(N) not O(N^2)



```
#define  OrigA 0x001
…..
```

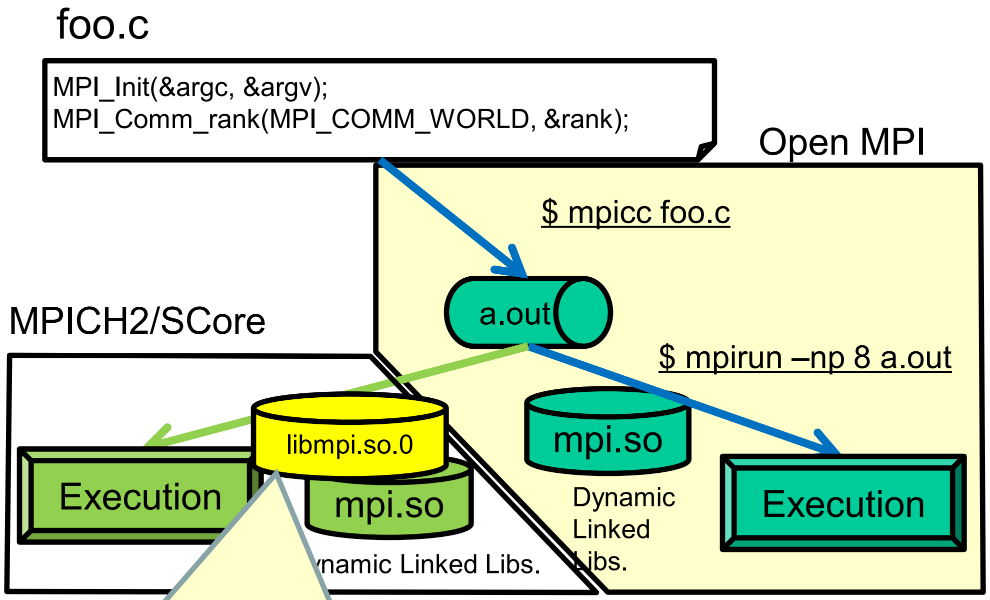
```
#define  TargA &xxxxx
…..
```

```
if(oarg == OrigA) {
    targ = TargA;
}
```

Making Process Flow of MPI Object Information

MPI-Adapter Implementation
Open MPI → MPICH2

- Overview of MPI-Adapter
  - From Open MPI to MPICH2/SCore
  - MPI-Adapter for C program
- Program Steps： 7Kstep
  - For dummy 305 MPI function entries
- Misc. Libraries
  - Resolving dependency of some misc libraries. （libopen-rte.so.0, libopen-pal.so.0
  - Providing dummy libs.



```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

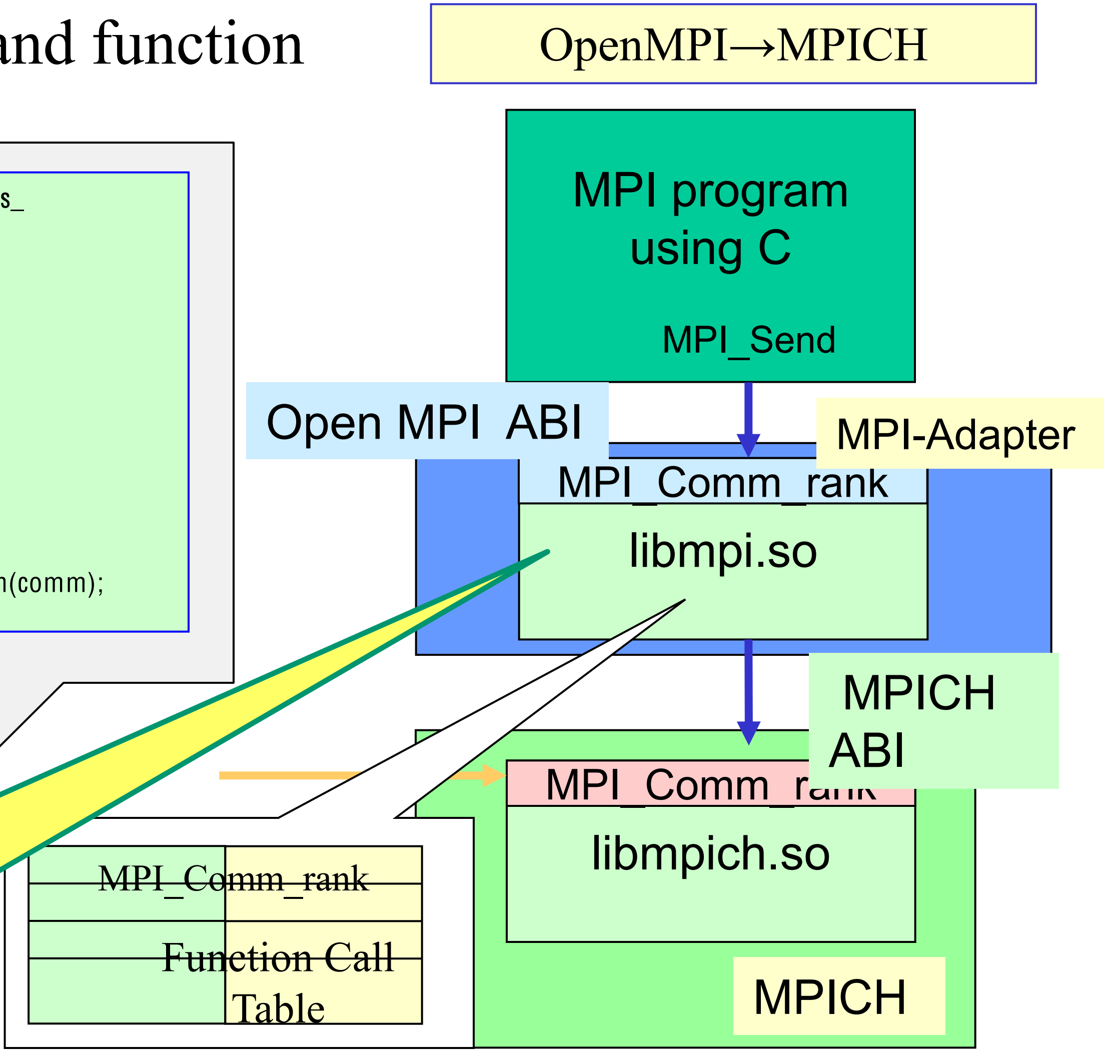
```
#include "mpi.h"
int MPI_Comm_rank(MPI_Comm comm, int *rank) {
    int dret;
    d_MPI_Comm dcomm = mpiconv_s2d_comm(comm);
    dret = (*ftables[OP_MPI_Comm_rank].funcp)(dcomm,
rank);
    return mpiconv_d2s_serrcode(dret);
}
```

Implementation of MPI-Adapter

- ABI translation modules and function call table

```
static inline void mpiconv_s2d_comm(d_MPI_Comm *dcomm, s_
MPI_Comm comm) {
   if(comm == s_MPI_COMM_WORLD)
       *dcomm = d_MPI_COMM_WORLD;
   else if(comm == s_MPI_COMM_NULL)
       *dcomm = d_MPI_COMM_NULL;
   else if(comm == s_MPI_COMM_SELF)
      *dcomm = d_MPI_COMM_SELF;
   else {
      if(sizeof(s_MPI_Comm) >= sizeof(d_MPI_Comm)) {
          *((d_MPI_Comm *)dcomm) = (d_MPI_Comm)comm;
      } else {
         *((d_MPI_Comm *)dcomm) = mpiconv_s2d_comm_hash(comm);
} } }
```

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



MPI Object Information Collection Tool and Conversion Skeleton sample

- MPI Object Information Collection Tool:
  - Implemented using C pre-processor and perl-script
  - Retrieving one ABI information from One MPI implementation.
- Coversion Skeleton Codes:
  - Replacing original and target MPI ABI information using C pre-processor

Tool output for Original Open MPI

```
#define s_MPI_COMM_WORLD (&ompi_mpi_comm_world)
#define s_MPI_COMM_NULL (&ompi_mpi_comm_null)
#define s_MPI_COMM_SELF (&ompi_mpi_comm_self)
```

Tool output for Target MPICH2

```
#define d_MPI_COMM_WORLD (d_MPI_Comm)0x44000000)
#define d_MPI_COMM_NULL ((d_MPI_Comm)0x04000000)
#define d_MPI_COMM_SELF ((d_MPI_Comm)0x44000001)
```

```
static inline void mpiconv_s2d_comm(d_MPI_Comm *dcomm,
s_MPI_Comm comm) {
   if(comm == s_MPI_COMM_WORLD)
       *dcomm = d_MPI_COMM_WORLD;
   else if(comm == s_MPI_COMM_NULL)
       *dcomm = d_MPI_COMM_NULL;
   else if(comm == s_MPI_COMM_SELF)
      *dcomm = d_MPI_COMM_SELF;
   else {
      if(sizeof(s_MPI_Comm) >= sizeof(d_MPI_Comm)) {
          *((d_MPI_Comm *)dcomm) = (d_MPI_Comm)comm;
      } else {
         *((d_MPI_Comm *)dcomm) = mpiconv_s2d_comm_hash(comm);
} } }
```

Built-in Conversion Skeleton Example in MPI-Adapter

- MPI-Adapter code for MPI_Comm translation.

```
static inline void mpiconv_s2d_comm(d_MPI_Comm *dcomm,
MPI_Comm comm) {
    if(comm == (&ompi_mpi_comm_world))
      *dcomm = ((d_MPI_Comm)0x44000000);
    else if(comm == (&ompi_mpi_comm_null))
      *dcomm = ((d_MPI_Comm)0x04000000);
    else if(comm == (&ompi_mpi_comm_self))
      *dcomm = ((d_MPI_Comm)0x44000001);
    else {
      if(sizeof(MPI_Comm) >= sizeof(d_MPI_Comm)) {
         *((d_MPI_Comm *)dcomm) = (d_MPI_Comm)comm;
      }
      else {
       *((d_MPI_Comm *)dcomm) = mpiconv_s2d_comm_hash(comm);
 } } }
```

Usage of MPI-Adapter：Basic

- Simple example

```
% mpirun –np 4 mpi-adapter [options] mpi-bin.exe
```

Options:

-s：type of original MPI mpiname (例: mpich2)

-d: type of target (mpirun)のmpiname (例：ompi)

例：ompi, mvapich, mpich_score

At default, -s ompi, -d mpich_score

Options are able to eliminate when using default values

MPI-Adapter Usages： Samples

- Running Open MPI binary on mpich2/SCore environment

```
% mpirun –np 4 mpi-adapter ompi.exe
% mpirun –np 4 mpi-adapter –s ompi ompi.exe
% mpirun –np 4 mpi-adapter –s ompi –d mpich_score ompi.exe
```

- Running Open MPI binary on mpich2 environment

```
% /opt/MPICH2/bin/mpirun –np 4 mpi-adapter –d mpich2 ompi.exe
% /opt/MPICH2/bin/mpirun –np 4 mpi-adapter –s ompi –d mpich2 ompi.exe
```

Current Status of MPI-Adapter

- Developed a Tool for making ABI information and MPI-Adapter from MPI runtime automatically
- MPI-Adapter works well on several MPI runtimes:
  - MPICH2 based: MPICH2, MPICH2/SCore, MPICH2-MX, MVAPICH
  - Open MPI, HP MPI
- Test Status:
  - Basic MPI Functions are tested, not whole of MPI2 functions.
    - Intel MPI Benchmarks (IMB), NAS Parallel Benchmarks.
    - BT-IO for MPI-IO Testing
  - MPI-Adapter works well on several clusters in Fujitsu Labs and T2K Todai, Tsukuba, Kyoto Cluster.

Some Cluster Environments using MPI-Adapter Portability Testing

	Distribution (Kernel) MPI	Glibc	GCC PE
Flab Cluster 1 RX200(Xeon)	CentOS 5.2 (2.6.18-8) MPICH2/SCore, Open MPI	2.5.12	4.1.1-52 16
Flab Cluster 2 HX600(Opteron)	CentOS 5.2 (2.6.18-92) MVAPICH, Open MPI	2.5-24	4.1.2-42 64
Flab PC Phenom	CentOS 5.3 (2.6.18-164) Open MPI, MPICH2	2.5-34	4.1.2-44 4
Flab PC2 Opteron	FedoraCore 11 (2.6.30-10) MVAPICH2, MPICH2	2.10-2	4.4.1-2 4
T2K Todai HA800	RedHat EL 5.1 (2.6.18-53) MPICH2-MX, HP MPI	2.5.24	4.1.2-14 256

- MPI-Adapter works well among these clusters

MPI-Adapter Overhead Evaluation on Fujitsu RX200 Cluster

Using MPI-Pingpong(mpi_rtt) Program on PMX/Shmem

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)

Unit：usec

- Fortran to C ABI Translation Overhead
  - MPICH2=0.012usec, Open MPI=0.022usec
- MPI-Adapter Overhead (Open MPI → MPICH2)
  - Fortran（INT to INT）=0.013usec, C (Pointer to INT)=0.0025usec
- Overhead of inserting MPI-Adapter is quite small

Performance Difference using MPI-Adapter on MPICH2-MX Runtime at T2K-Todai Cluster

256 PE, Fortran=gfortran

Class C	BT	CG	FT	LU	MG	SP
Open MPI	0.5%	0.3%	1.0%	2.3%	0.8%	-1.3%
HP MPI	0.5%	0.6%	0.2%	-0.3%	2.7%	-1.1%

🟥 Performance UP

- Open MPI binaries were compiled on Flab Cluster 1 and Copied to T2K-Todai Cluster.
- Performance Difference: Less than 2.7%

Summary

- MPI-Adapter for Portable MPI Computing Environment.
  - Keeping MPI ABI compatibility by MPI ABI translator.
  - Implemented and Evaluated on T2K-Todai Cluster and several Fujitsu Clusters
    - Overhead of inserting MPI-Adapter is negligible
    - Works well among MPICH2/SCore, MPICH2, Open MPI, HP MPI runtimes
- Future Work
  - Tested among Three T2K Clusters (Tsukuba, Todai, and Kyoto), and entire MPI functions using MPI test suites.
  - Other Usage: Profiler Interface….
- Acknowledgement: This research was partially supported by the eScience project of the MEXT, Japan.

Thank You.

MPI-Adapter Demonstrations

- Demonstration on VMware environment
  - Intel Core2 Duo(2 core), Cent OS 5.4, SCore7
  - MPI Runtimes: MPICH2, Open MPI, MPICH2/SCore
- Pre-build NAS Parallel benchmark Binaries
  - MPICH2, Open MPI, MPICH2/SCore, HP MPI
- Demonstration
  - Run mpirun program w/ (w/o) inserting MPI-Adapter