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:**
  - **Selecting Clusters:** for development and production
  - **Binary Distribution:** for ISV and developer
  - **Changing Runtime Environment:** 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

- 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.
  - The differences are the reason of lack of ABI compatibility.

<table>
<thead>
<tr>
<th>Objects</th>
<th>Types (a pre-defined value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communicator</td>
<td>MPI_Comm</td>
</tr>
<tr>
<td></td>
<td>(MPI_COMM_WORLD)</td>
</tr>
<tr>
<td>Group</td>
<td>MPI_Group</td>
</tr>
<tr>
<td>Request</td>
<td>MPI_Request</td>
</tr>
<tr>
<td>Status</td>
<td>MPI_Status</td>
</tr>
<tr>
<td>Data type</td>
<td>MPI_Datatype</td>
</tr>
<tr>
<td></td>
<td>(MPI_Int,)</td>
</tr>
<tr>
<td>Operation</td>
<td>MPI_Op (MPI_MAX)</td>
</tr>
<tr>
<td>Window</td>
<td>MPI_Win</td>
</tr>
<tr>
<td>File</td>
<td>MPI_File</td>
</tr>
<tr>
<td>Info</td>
<td>MPI_Info</td>
</tr>
<tr>
<td>Pointer diffs.</td>
<td>MPI_Aint</td>
</tr>
<tr>
<td>Offset</td>
<td>MPI_Offset</td>
</tr>
<tr>
<td>Error Handler</td>
<td>MPI_Errorhandler</td>
</tr>
</tbody>
</table>
Differences of Pre-defined Values between MPICH2 and Open MPI

<table>
<thead>
<tr>
<th>Pre-defined Values</th>
<th>MPICH2</th>
<th>Open MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI_COMM_WORLD</td>
<td>0x44000000</td>
<td>&amp;ompi_mpi_comm_world</td>
</tr>
<tr>
<td>MPI_INT</td>
<td>0x4c000405</td>
<td>&amp;ompi_mpi_int</td>
</tr>
<tr>
<td>MPI_INTEGER</td>
<td>0x4c00041b</td>
<td>&amp;ompi_mpi_integer</td>
</tr>
<tr>
<td>MPI_SUCCESS</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MPI_ERR_TRUNCATE</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

- 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.
## Differences of MPI Implementations

### Survey of ABI Working Group (MPI Forum)

<table>
<thead>
<tr>
<th>Implementations</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel MPI</td>
<td>MPICH2 based (Integer)</td>
</tr>
<tr>
<td>MS MPI</td>
<td>MPICH2 based (Integer)</td>
</tr>
<tr>
<td>HP MPI</td>
<td>Original (Structure Based)</td>
</tr>
<tr>
<td>LAMPI</td>
<td>Original (Integer and Structure)</td>
</tr>
<tr>
<td>NEC MPI</td>
<td>Original? (Integer)</td>
</tr>
</tbody>
</table>

- 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

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 dlopen() and dlsym().

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

```c
#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

```c
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);
        }
    }
}
```

Function Call Table

```
int MPI_Comm_rank(int comm, int *p) {
    int cc;
    void *ocomm = convMPI_Comm(comm);
    call_MPICHMPIL(&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.

- Conversion Skeleton Codes:
  - Replacing original and target MPI ABI information using C pre-processor

Tool output for Original Open MPI

```c
#define d_MPI_COMM_SELF ((d_MPI_Comm)0x44000001)
#define d_MPI_COMM_NULL ((d_MPI_Comm)0x04000000)
#define d_MPI_COMM_WORLD (d_MPI_Comm)0x44000000)
#define s_MPI_COMM_SELF (&ompi_mpi_comm_self)
#define s_MPI_COMM_NULL (&ompi_mpi_comm_null)
#define s_MPI_COMM_WORLD (&ompi_mpi_comm_world)
```

Tool output for Target MPICH2

```c
#define d_MPI_COMM_SELF (d_MPI_Comm)0x44000001)
#define d_MPI_COMM_NULL ((d_MPI_Comm)0x04000000)
#define d_MPI_COMM_WORLD ((d_MPI_Comm)0x440000001)
```

```c
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.

```c
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

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

Options:

- `-S`: type of original MPI `mpiname` (Example: `mpich2`)
- `-d`: type of target (`mpirun`) `mpiname` (Example: `OMPI`)  

Example: `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

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

- MPI-Adapter works well among these clusters
# MPI-Adapter Overhead Evaluation on Fujitsu RX200 Cluster

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

<table>
<thead>
<tr>
<th>usec</th>
<th>Fortran</th>
<th>C</th>
<th>Overhead (/MPI call)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open MPI+ MPI-Adaptor</td>
<td>3.154</td>
<td>3.065</td>
<td>0.082 (0.022)</td>
</tr>
<tr>
<td>MPICH2/SCore</td>
<td>3.103</td>
<td>3.055</td>
<td>0.048 (0.012)</td>
</tr>
<tr>
<td>Overhead (/MPI)</td>
<td>0.051 (0.013)</td>
<td>0.010 (0.0025)</td>
<td>0.034 (0.0085)</td>
</tr>
</tbody>
</table>

- **Fortran to C ABI Translation Overhead**
  - MPICH2=0.012 usec, Open MPI=0.022 usec

- **MPI-Adapter Overhead (Open MPI → MPICH2)**
  - Fortran (INT to INT)=0.013 usec, C (Pointer to INT)=0.0025 usec

- **Overhead of inserting MPI-Adapter is quite small**

---

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

<table>
<thead>
<tr>
<th>Class C</th>
<th>BT</th>
<th>CG</th>
<th>FT</th>
<th>LU</th>
<th>MG</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open MPI</td>
<td>0.5%</td>
<td>0.3%</td>
<td>1.0%</td>
<td>2.3%</td>
<td>0.8%</td>
<td>-1.3%</td>
</tr>
<tr>
<td>HP MPI</td>
<td>0.5%</td>
<td>0.6%</td>
<td>0.2%</td>
<td>-0.3%</td>
<td>2.7%</td>
<td>-1.1%</td>
</tr>
</tbody>
</table>

256 PE, Fortran=gfortran

- 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), CentOS 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