20 Years of the MPI Standard

Now With a Common Application Binary Interface

By Alexander Supalov *Intel Cluster Tools Architect,* **and Artem Yalozo,** *Software Engineer, Intel Corporation*

Introduction

Message Passing Interface (MPI)[1]—the de facto industry standard for distributed memory computing—celebrates its 20th anniversary this year. There are few interfaces that can compete with MPI on the critical performance required for high performance computing (HPC).

MPI has come a long way since its first introduction by the MPI Forum in May 1994. In September 2012, a major new version, the MPI-3 standard,[2] was voted into existence. The latest version of the standard adds fast one-sided communication, non-blocking collective operations, and quite a few other features.

Based on this new standard, Intel Corporation, in cooperation with Argonne National Laboratory, Cray Corporation, and IBM Corporation, formed an initiative that introduced the long-desired common MPI Application Binary Interface (ABI) at SuperComputing 2013. This ABI will be available in Intel® MPI Library 5.0, shipping in June 2014, as in a number of compatible third-party products, including MPICH*, Cray MPI*, and IBM POE*.

Motivation

Compatibility has long been a key concern for the Intel MPI Library, a multi-fabric message passing library that is used on high performance computing clusters.[3] Our customers have always expected it to preserve compatibility, even between major product releases. This allowed for a binary-compatible upgrade path to a subsequent version of the product library, making the transitions to new versions easier and more successful. Without ABI compatibility, any released software building on top of an existing version needs to be re-released for every new product version.

Reasons for Potential Incompatibility

The root cause for prior MPI library incompatibilities stemmed from the use of different source bases. Different versions were based on different MPICH[4] source codes and implemented different versions of the MPI standard. For example, the basis for Intel MPI 4.1 was MPICH2, and hence it implemented the MPI-2.2 standard. The basis for Intel MPI 5.0 is MPICH3 and hence the new MPI-3 standard.

Application binary interface is the low-level interface between two program modules. It describes the binary interface to a library, and should not be confused with the Application Programming Interface (API). The ABI determines such details as how functions are called and the size, layout, and alignment of data types. With ABI, compatible programs conform to the same set of runtime conventions. A library is binary compatible if a program linked to a previous version of the library continues to work with a newer version without being recompiled.

The second kind of compatibility implies that all library functions do the same thing in the newer version that they did in older versions. Of course, there is some flexibility here. The old behavior could be at some point incorrect, so it could be changed in the new release of the software. And this could be a root cause of broken behavior compatibility. This could also be caused by the changed MPI standard.

Compatibility Scenarios, Issues, and Solutions

Ideally, both backward and forward compatibility of the software could be ensured. In practice, products provide a reasonable balance between backward compatibility and the need to implement newer standards.

The main compatibility scenario for the Intel MPI Library is the backward compatibility scenario. In this case, an application built with a new Intel MPI version runs with previous library runtime. Forward compatibility is considered, but not claimed.

ABI compatibility problems may cause crashes in the client code. They may also cause data corruption. They may even go unnoticed. Because of this, a careful investigation of potential incompatibilities should be performed before trying to catch the incompatibility crashes during testing.

There are several language bindings provided by the Intel MPI Library: C, C++, and Fortran 77/90. Changes in each interface are the potential cause for binary incompatibility. **Table 1** summarizes reasons and possible solutions for binary incompatibility cases discovered between Intel MPI Library 5.0 (based on MPICH3) and 4.1 (MPICH2).

Reason (changes between Intel MPI 4.1 and MPICH3)	**Language binding**	**Solution**
Predefined constant changed: **MPI_MAX_ERROR_STRING**	All	Return to 4.1 ABI
Constants/new definitions added: **MPICH_ATTR_FAILED_PROCESSES** **MPI_UNWEIGHTED** **MPICH_ERR_LAST_CLASS**	C	Include constant, take new definition
Predefined constant added/changed: **MPI_COMBINER_HINDEXED_BLOCK** **MPI_COMBINER_***	C Fortran	Assign next value to **HINDEXED_BLOCK**, leave others unchanged
MPI_Status items order changed (**count** and **cancelled** moved down). Type of **count** changed from **int** to 64-bit **MPI_Count.** **typedef struct MPI_Status {** int MPI_SOURCE; int MPI_TAG; int MPI_ERROR; **MPI_Count count;** **int cancelled;** } MPI_Status;	C	Move **count** and **cancelled** up. Interpret 32-bit **count** and **cancelled** fields as a 63-bit **count** and a 1-bit **cancelled** values
Predefined constant added/removed: **MPI_2COMPLEX,** **MPI_2DOUBLE_COMPLEX**	Fortran	Include constant
Common blocks removed/added: **MPIPRIV1, MPIPRIV2, MPIPRIVC MPIFCMB5, MPIFCMB9**	Fortran	Return to 4.1 ABI, add new common block(s) for all MPI-3 things
Virtual functions table changed (order changed): **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++	Return to 4.1 ABI

1 Binary compatibility issues and their resolution.

The Intel MPI Library 5.0 Beta provided ABI compatibility with Intel MPI Library 4.x by introducing the necessary changes in the source codes with regards to discovered compatibility issues. Originally, Intel MPI Library 5.0 Beta was not compatible with MPICH3. However, the ABI changes could be controlled during the library build stage (optionally), and an MPICH3 compatible library could still be built. However, Intel MPI 5.0 is compatible with MPICH3.

Table 2 summarizes reasons and possible solutions for discovered behavior incompatibility between Intel MPI Library 4.1 (based on MPICH2 1.4) and 4.0 (MPICH2 1.1).

Reason (changes between Intel MPI 4.1 and MPICH2 1.4)	**Language binding**	**Solution**
Behavior for some functions changed by MPI standard: **MPI_Cart_create** **MPI_Cart_map** **MPI_Graph_create** **MPI_Win_get_attr**	C	Change function behavior based on I_MPI_COMPATIBILITY environment variable value
Behavior for some collective operations changed by MPI standard: **Gather** **Allgather** **et al.**	C	Change function behavior based on I_MPI_COMPATIBILITY environment variable value

2 Behavior compatibility issues and their resolution.

Compatibility in behavior for the Intel MPI Library is not provided by default because of the need to implement the newer standard that changed the behavior. However, backward compatibility in behavior can be achieved at runtime by setting up a special environment variable `I_MPI_COMPATIBILITY` to the value of 4.

As a result, within a single library the Intel MPI introduces support for the newer standard and still remains binary compatible with the majority of the applications built using older versions of the Intel MPI Library.

Conclusion

The Intel MPI Library provides compatibility with all previous versions. This is one of its significant advantages. Meanwhile, the new MPI ABI will be adopted by all partners by the summer of 2014. (For more details on the new ABI, visit the MPICH ABI Compatibility Initiative[5] at: **http://www.mpich.org/abi/**.) Once this occurs, Intel MPI Library, MPICH, Cray MPI, and IBM POE will become fully interchangeable at runtime. Moreover, this ABI will allow both MPI-2.x and MPI-3 standards to be supported by one MPI library. This will preserve customer investment into the earlier MPI standards and related applications, and simplify the life of the MPI implementers by allowing them to support only one package in the field. This will also allow our binary compatibility to extend across our collaborators' MPI implementations, which is important both to our customers and to our partners.

The ABI and the resulting ability to switch between libraries can help every developer by letting different MPI implementations compete based on performance alone. This will also open up the MPI space to the associated tools currently built for a specific MPI distribution and unable to reach outside of this box.

References

1. Message Passing Interface (MPI): **http://www.mpi-forum.org/docs**
2. Message Passing Interface Forum. MPI: A Message-Passing Interface Standard, Version 3.0. September 21, 2012. **http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf**
3. The Intel® MPI Library: **www.intel.com/go/mpi**
4. MPICH: **http://www.mpich.org**
5. MPICH ABI Compatibility Initiative: **http://www.mpich.org/abi/**