MPI-3 Standard and Support in Intel MPI 5.0 beta

Agenda

What's in MPI-3?

Use cases

- Complete communication/computation overlap
- Sparse communication
- One-sided communication sounds cool (in theory)
- Large messages
- What else?

Intel® MPI Library

- What is it?
- What is supported? (MPI-3)

How do you spell MPI?

A de facto standard for communicating between processes of a parallel program on a distributed memory system

- Standardized
  - Supported on almost all platforms
- Portable
  - No need to modify your code when porting
- Performance opportunities
  - Vendor MPIs can exploit native hardware features
- Functionality
  - Over 125 routines defined by a committee

Example (C)

```
#include "mpi.h"

int main(argc,argv){
MPI_Init(&argc,&argv);

MPI_Comm_size(MPI_COMM_WORLD,&numtasks);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
printf ("Number of tasks= %d \
     My rank= %d\n",ntasks,rank);

/*******  do some work **

MPI_Finalize();
}
```

MPI include file

Initialize MPI environment

Do work and make MPI calls

Terminate MPI environment

What is in MPI-3?

Topic	Motivation	Main Result
Collective Operations	Collective performance	Non-Blocking & Sparse Collectives
Remote Memory Access	Cache coherence, PGAS support	Fast RMA
Backward Compatibility	Buffers > 2 GB	Large buffer support, const buffers
Fortran Bindings	Fortran 2008	Fortran 2008 bindings Removed C++ bindings
Tools Support	PMPI Limitations	MPIT Interface
Hybrid Programming	Core count growth	MPI_Mprobe, shared memory windows
Fault Tolerance	Node count growth	None. Next time?

I want a complete comm/comp overlap

Problem

- Computation/communication overlap is not possible with the blocking collective operations

Solution: Non-blocking Collectives

- Add non-blocking equivalents for existing blocking collectives
- Do not mix non-blocking and blocking collectives on different ranks in the same operation

Example (C)

```
// Start synchronization
MPI_Ibarrier(comm, &req);

// Do extra computation
…

// Complete synchronization
MPI_Test(&req, …);
```

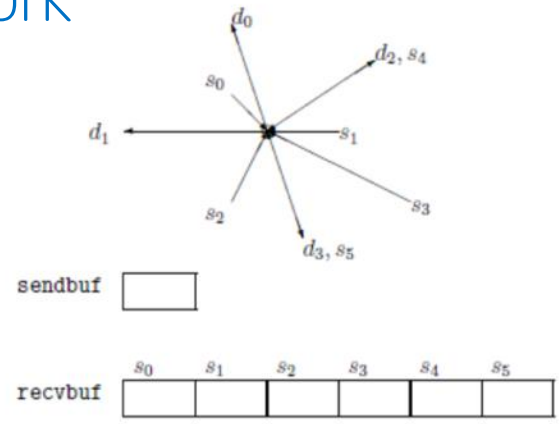
I have a sparse communication network

Problem

- Neighbor exchanges are poorly served by the current collective operations (memory and performance losses)

Solution: Sparse Collectives

- Add blocking and non-blocking `Allgather*` and `Alltoall*` collectives based on neighborhoods



Example (FORTRAN)

```
call MPI_NEIGHBOR_ALLGATHER(&
& sendbuf, sendcount, sendtype,&
& recvbuf, recvcount, recvtype,&
& graph_comm, ierror)
```

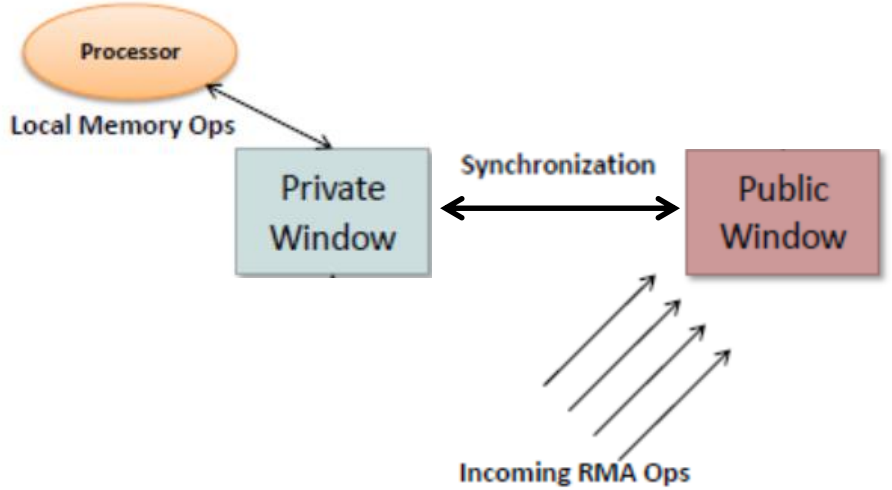
I want to use one-sided calls to reduce sync overhead

Problem

- MPI-2 one-sided operations are too general to work efficiently on cache coherent systems and compete with PGAS languages

Solution: Fast Remote Memory Access

- Eliminate unnecessary overheads by adding a 'unified' memory model
- Simplify usage model by supporting the `MPI_Request` non-blocking call, extra synchronization calls, relaxed restrictions, shared memory, and much more



Example (FORTRAN)

```
call MPI_WIN_GET_ATTR(win, MPI_WIN_MODEL, &
memory_model, flag, ierror)
if (memory_model .eq. MPI_WIN_UNIFIED) then
! private and public copies coincide
```

I'm sending *very* large messages

Problem

- Original MPI counts are limited to 2 Gigaunits, while applications want to send much more

Solution: Large Buffer Support

- "Hide" the long counts inside the derived MPI datatypes
- Add new datatype query calls to manipulate long counts

Example (FORTRAN)

```
// mpi_count may be, e.g.,
//    64-bit long
MPI_Get_elements_x(&status,
datatype, &mpi_count);
```

None of these apply to me. What else you got?

I have a hybrid application

- Create a communicator inside a shared memory domain (intranode, via `MPI_Comm_split_type`)
- Use the new `MPI_Mprobe` calls

I need to know what architecture I'm running on

- Predefined info object `MPI_INFO_ENV` allows for environment query

I'm using the C++ bindings

- Tough luck. C++ bindings have been removed from the standard.

Tell me more about this Intel® MPI Library

Optimized MPI application performance

- Application-specific tuning
- Automatic tuning

Lower Latency and Multi-vendor interoperability

- Optimized support for latest OFED* features

Faster MPI communication

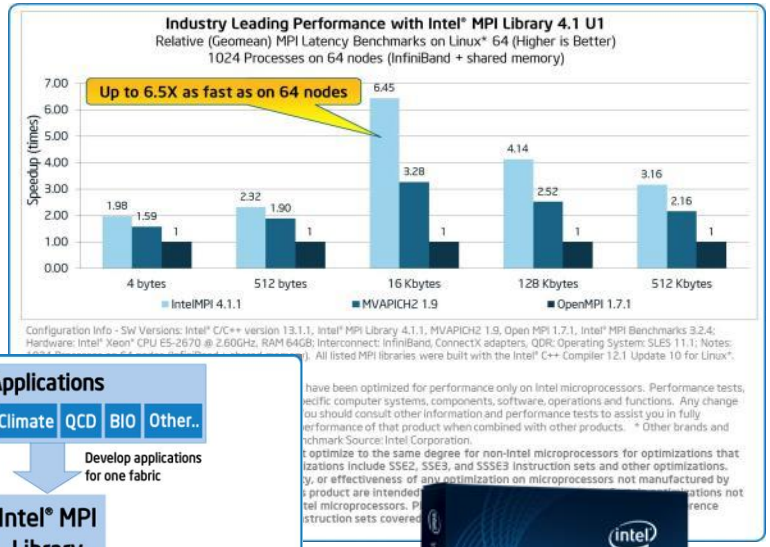
- Optimized collectives

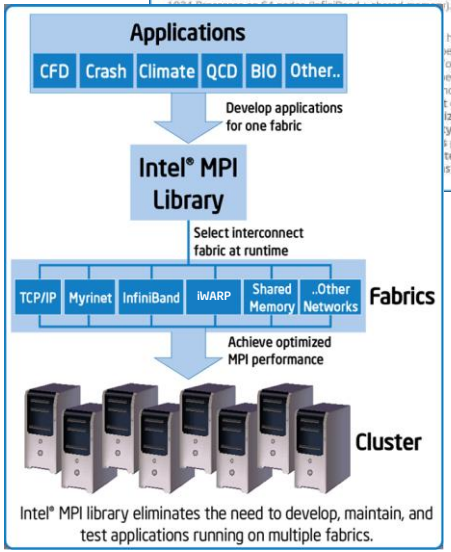
Sustainable scalability beyond 120K cores

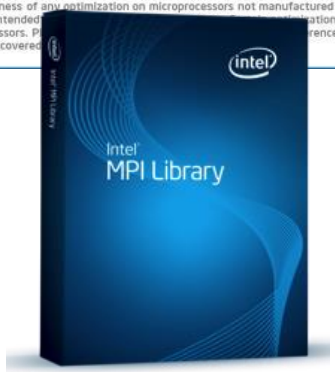
- Native InfiniBand* interface allows for reduced memory load and higher bandwidth

Simply and Accelerate Clusters

- Intel® Cluster Ready compliance







Intel® MPI Library 5.0 & Intel® Trace Analyzer and Collector 9.0
Beta Nov 2013

Intel® MPI Library

Initial MPI-3.0 Support

- Non-blocking Collectives
- Fast RMA
- Large Counts

ABI compatibility with existing Intel® MPI Library applications

Intel® Trace Analyzer and Collector

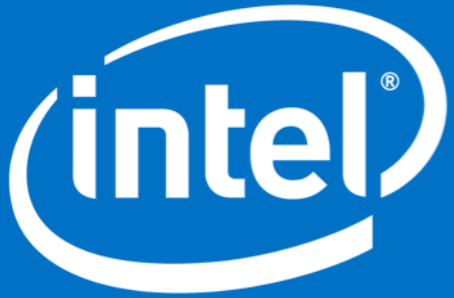
Initial MPI-3.0 Support

Automatic Performance Assistant

- Detect common MPI performance issues
- Automated tips on potential solutions

What is supported in Intel® MPI Library 5.0 Beta?

Topic	Motivation	Main Result	Supported in 5.0 Beta?
Collective Operations	Collective performance	Non-Blocking & Sparse Collectives	Yes
Remote Memory Access	Cache coherence, PGAS support	Fast RMA	Yes
Backward Compatibility	Buffers > 2 GB	Large buffer support, const buffers	Yes, partial
Fortran Bindings	Fortran 2008	Fortran 2008 bindings Removed C++ bindings	No support in MPICH3.0
Tools Support	PMPI Limitations	MPIT Interface	Yes
Hybrid Programming	Core count growth	MPI_Mprobe, shared memory windows	Yes


intel