# MPI Programming

Intel Software College

# Course Objectives

After successfully completing this module, you will be able to:

- Compile and run MPI programs using the MPICH implementation
- Write MPI programs using the core library

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Agenda

What is MPI?

Core MPI Library

Basic Communication

Non-Blocking Communications

Collective Communications

Advanced MPI

- Useful Features
- Factors Affecting MPI Performance

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Agenda

What is MPI?

Core MPI Library

Basic Communication

Non-Blocking Communications

Collective Communications

Advanced MPI
- Useful Features
- Factors Affecting MPI Performance

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPI

A library - not a language

A library for inter-process communication and data exchange

Function categories

- Initialization/finalization
- Point-to-point communication
- Collective communication
- Communicator topologies
- User-defined data types
- Utilities (e.g.- timing and initialization)

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Parallel Models Compared

| | MPI | Threads | OpenMP* |
|---|---|---|---|
| Portable | ✓ | | ✓ |
| Scalable | ✓ | ✓ | ✓ |
| Performance Oriented | ✓ | | ✓ |
| Supports Data Parallel | ✓ | ✓ | ✓ |
| Incremental Parallelism | | | ✓ |
| High Level | | | ✓ |
| Serial Code Intact | | | ✓ |
| Verifiable Correctness | | | ✓ |
| Distributed Memory | ✓ | | |

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Common MPI Implementations

MPICH* (Argonne National Laboratory)

- Most common MPI implementation
- Derivatives
  - MPICH GM* – Myrinet* support (available from Myricom)
  - MVAPICH* – InfiniBand* support (available from Ohio State University)
  - Intel® MPI – version tuned to Intel Architecture systems
  - LAM/MPI* (Indiana University)
    - Contains many MPI 2.0 features
    - Daemon-based MPI implementation
  - MPI/Pro* (MPI Software Technology)
  - Scali MPI Connect*
    - Provides native support for most high-end interconnects

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Getting Started

OSCAR? NPACI Rocks?

- MPI Implementation(s) pre-configured
- For a manual build see material in backup to these slides

What do you need to run?

- Program – supplied or build with
  `mpicc <program.c>   or`
  `mpif77 <program.f>`
- Which machines to use?  (machines file)
- A way to launch programs on remote machines (ssh for both OSCAR and Rocks)

Run with:

- `mpirun –n xx  <program name>`
  where xx is the number of processes (ranks) to launch

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Absolute Minimum MPI Program Skeleton

Initialize MPI Lib on my node – required

Do work – optional, but important ☺

Wrap it up once we are finished – required

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Compiling an MPI Program

Most MPI implementations supply compilation scripts:

```
mpif77 mpi_prog.f
mpicc mpi_proc.c
mpif90 mpi_prof.f90
mpiCC mpi_prof.C
```

Manual compilation/linking also possible:

```
ifc mpi_prog.f –L/usr/local/mpich-1.2.5/lib –lfmpich –lmpich
```

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPI Machine File

A text file telling MPI where to launch processes

Put separate machine name on each line

Example:

```
compute-0-0
compute-0-1
compute-0-2
compute-0-3
```

Check implementation for multi-processor node formats

Default file found in MPI installation

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

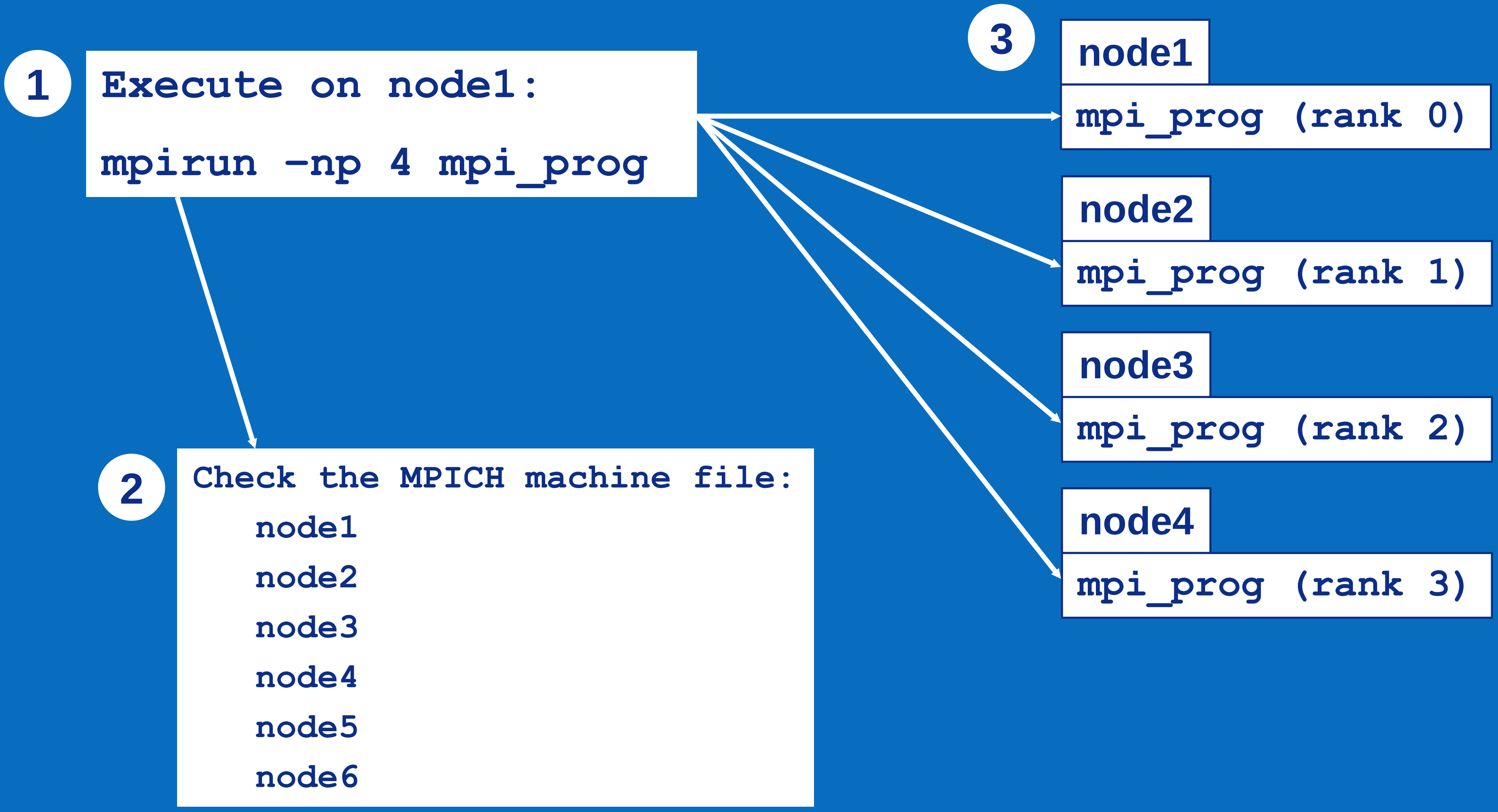
# Parallel Program Execution

Launch scenario for MPIRUN

- Find machine file (to know where to launch)
- Use SSH or RSH to execute a copy of program on each node in machine file
- Once launched, each copy establishes communication with local MPI lib (MP_Init)
- Each copy ends MPI interaction with MPI_Finalize

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Starting an MPI Program



1

Execute on node1:

```
mpirun –np 4 mpi_prog
```

2

Check the MPICH machine file:

```
node1
node2
node3
node4
node5
node6
```

3

**node1**

`mpi_prog (rank 0)`

**node2**

`mpi_prog (rank 1)`

**node3**

`mpi_prog (rank 2)`

**node4**

`mpi_prog (rank 3)`

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Activity 1: “Hello, World” in MPI

Demonstrates how to create, compile and run a simple MPI program on the lab cluster using the Intel MPI implementation

```
#include <stdio.h>
#include “mpi.h”

int main (int argc, char* argv[])
{
   MPI_Init (&argc, &argv); /* Initialize the library */

   printf (“Hello world\n”);

   MPI_Finalize (); /* Wrap it up. */
}
```

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Agenda

What is MPI?

Core MPI Library

Basic Communication

Non-Blocking Communications

Collective Communications

Advanced MPI

- Useful Features
- Factors Affecting MPI Performance

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# The Whole Library

*MPI is large and complex*

- MPI 1.0 – 125 functions
- MPI 2.0 is even larger

But, many MPI features are rarely used

- Inter-communicators
- Topologies
- Persistent communication
- Functions designed for library developers

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# The Absolute Minimum

Six MPI functions:

- `MPI_Init` ✯
- `MPI_Comm_size`
- `MPI_Comm_rank`
- `MPI_Send`
- `MPI_Recv`
- `MPI_Finalize` ✯

Many parallel algorithms can be implemented efficiently with only these functions

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Initializing the MPI Library

```
Fortran:
      MPI_INIT (ierr)

C:
      int MPI_Init (int* argc, char** argv)
```

`MPI_Init` prepares the system for MPI execution

Call to `MPI_Init` may update arguments in C

- Implementation dependent

No MPI functions may be called before `MPI_Init`

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Shutting Down MPI

```
Fortran:
      MPI_FINALIZE (ierr)

C:
      int MPI_Finalize (void)
```

`MPI_Finalize` frees any memory allocated by the MPI library

No MPI functions may be called after calling `MPI_Finalize`

- Exception: `MPI_Init`

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Sizing the MPI Communicator

```
Fortran:
     MPI_COMM_SIZE (comm, size, ierr)
          integer :: comm, size, ierr
C:
     int MPI_Comm_size (
               MPI_Comm comm, int* size)
```

`MPI_Comm_size` returns the number of processes in the specified communicator

The communicator structure, `MPI_Comm`, is defined in `mpi.h`

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Determining MPI Process Rank

```
Fortran:
     MPI_COMM_RANK (comm, rank, ierr)
          integer :: comm, rank, ierr
C:
     int MPI_Comm_rank (
               MPI_Comm comm, int* rank)
```

`MPI_Comm_rank` returns the rank of calling process within the specified communicator

Processes are numbered from 0 to N-1

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Activity 2: “Hello, World” with IDs

Demonstrates how to modify, compile and run a simple MPI program on the lab cluster using the Intel MPI implementation

```
#include <stdio.h>
#include “mpi.h”

int main (int argc, char* argv[])
{
   int numProc, myRank;

   MPI_Init (&argc, &argv); /* Initialize the library */
   MPI_Comm_rank (MPI_COMM_WORLD, &myRank); /* Who am I?” */
   MPI_Comm_size (MPI_COMM_WORLD, &numProc); /*How many? */

   printf (“Hello. Process %d of %d here.\n”, myRank, numProc);

   MPI_Finalize (); /* Wrap it up. */
}
```

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Agenda

What is MPI?

Core MPI Library

Basic Communication

Non-Blocking Communications

Collective Communications

Advanced MPI

- Useful Features
- Factors Affecting MPI Performance

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Sending Data

```
Fortran:
      MPI_SEND (buf, count, datatype,
                        dest, tag, comm, ierr)
      <type> buf(*)
      integer :: count, datatype, ierr,
                        dest, tag, comm
C:
      int MPI_Send (void* buf, int count,
            MPI_Datatype datatype, int dest,
            int tag, MPI_Comm comm)
```

`MPI_Send` performs a blocking send of the specified data to the specified destination

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Receiving Data

```
Fortran:
      MPI_RECV (buf, count, datatype, source,
                    tag, comm, status, ierr)
      <type> buf(*)
      integer :: count, datatype, ierr, source,
                    tag, comm,
                    status(MPI_STATUS_SIZE)
C:
      int MPI_Recv (void* buf, int count,
            MPI_Datatype datatype, int source,
            int tag, MPI_Comm comm,
            MPI_Status* status)
```

`MPI_Recv` performs a blocking receive of the specified data from the specified source

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPI Data Types for C

| MPI Data Type | C Data Type |
|---|---|
| MPI_BYTE | |
| MPI_CHAR | signed char |
| MPI_DOUBLE | double |
| MPI_FLOAT | float |
| MPI_INT | int |
| MPI_LONG | long |
| MPI_LONG_DOUBLE | long double |
| MPI_PACKED | |
| MPI_SHORT | short |
| MPI_UNSIGNED_SHORT | unsigned short |
| MPI_UNSIGNED | unsigned int |
| MPI_UNSIGNED_LONG | unsigned long |
| MPI_UNSIGNED_CHAR | unsigned char |

MPI provides predefined data types that must be specified when passing messages.

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPI Data Types for Fortran

| MPI Data Type | Fortran Data Type |
|---|---|
| `MPI_BYTE` | |
| `MPI_CHARACTER` | CHARACTER |
| `MPI_COMPLEX` | COMPLEX |
| `MPI_DOUBLE_PRECISION` | DOUBLE PRECISION |
| `MPI_INTEGER` | INTEGER |
| `MPI_LOGICAL` | LOGICAL |
| `MPI_PACKED` | |
| `MPI_REAL` | REAL |

MPI provides predefined data types that must be specified when passing messages.

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPI Status Details

## C

type `MPI_Status` has three fields

- `MPI_SOURCE`
- `MPI_TAG`
- `MPI_ERROR`

```
status.MPI_SOURCE

status.MPI_TAG

status.MPI_ERROR
```

## Fortran

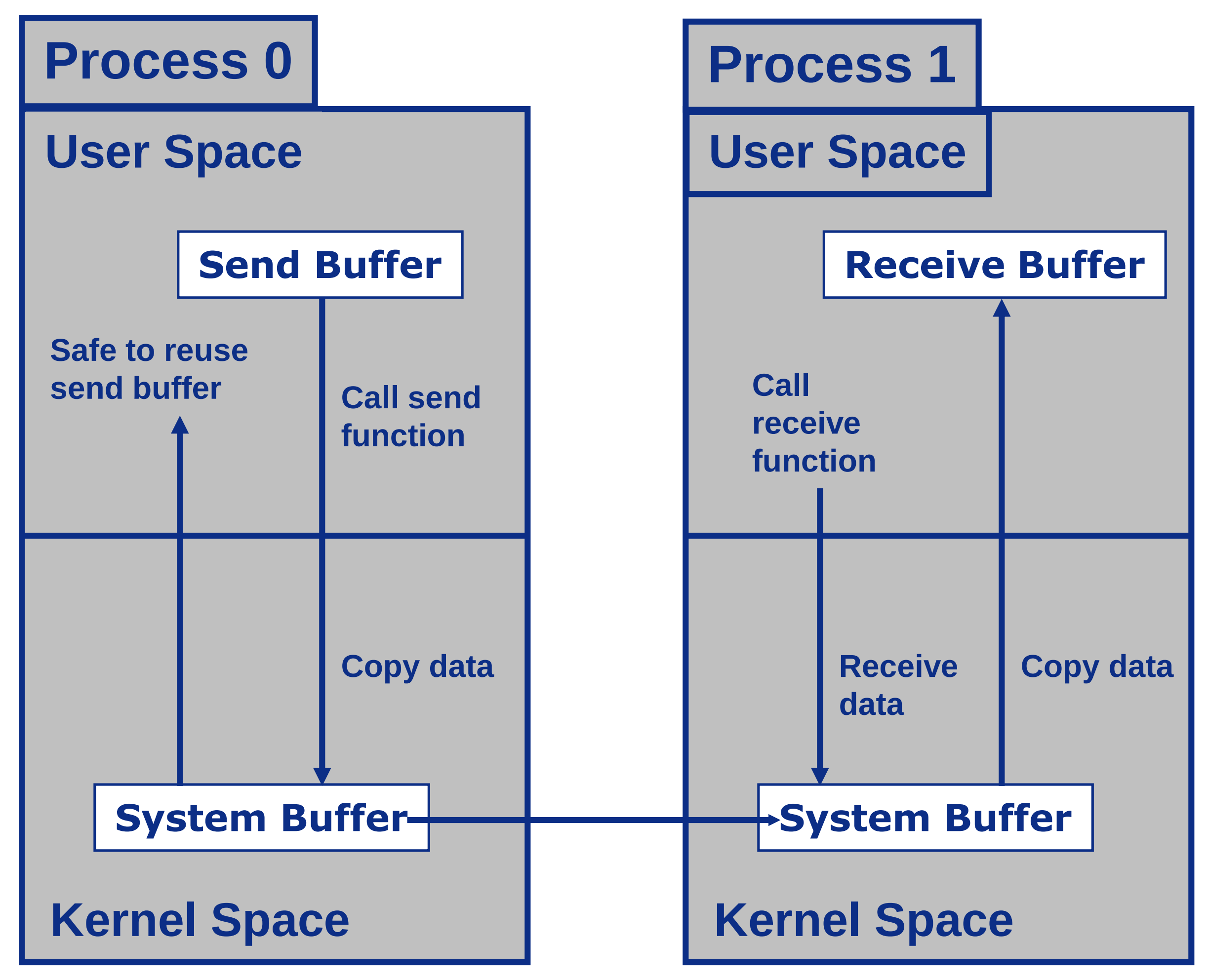
Status is an array indexed by

- `MPI_SOURCE`
- `MPI_TAG`
- `MPI_ERROR`

```
status(MPI_SOURCE)

status(MPI_TAG)

status(MPI_ERROR)
```

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Blocking Communication



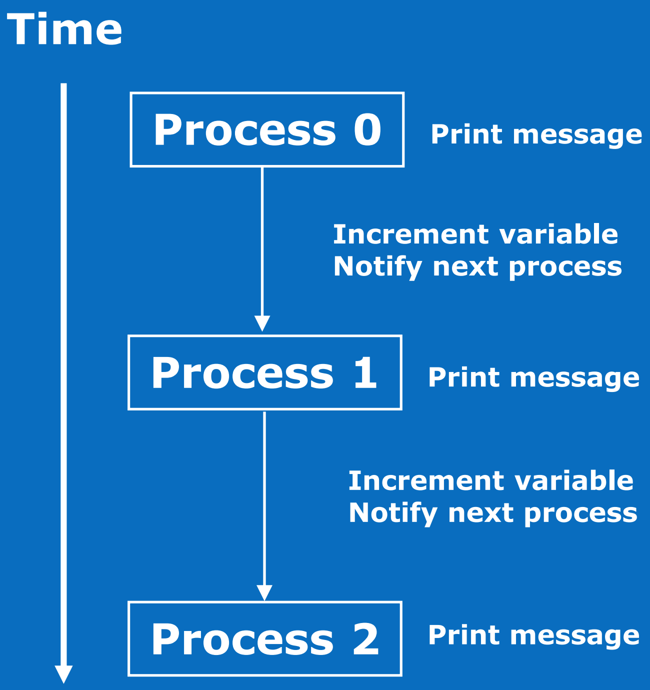
## Key Points

- Sending and receiving data is a paired operation. Ignoring this principle can result in deadlock.
- Communication and synchronization are coupled.

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Activity 3: “Hello, World” with Message

Modify the “Hello, World” example so that the MPI processes pass an integer variable from left to right, reporting and incrementing its value at each step.



**Time**

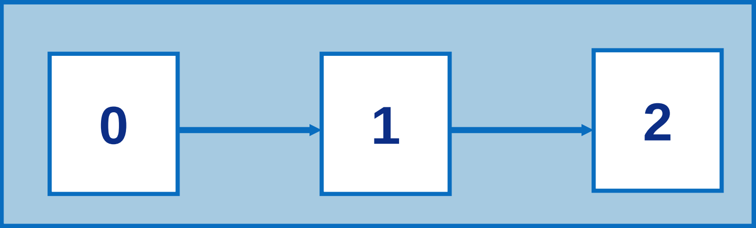
**Process 0** **Print message**

**Increment variable**
**Notify next process**

**Process 1** **Print message**

**Increment variable**
**Notify next process**

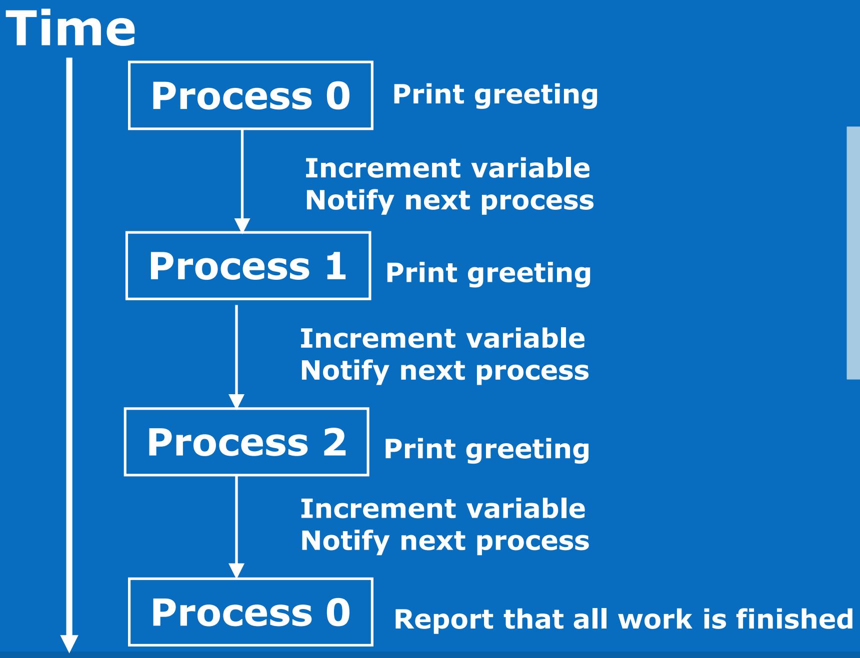
**Process 2** **Print message**



Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Activity 4: "Hello, World" with Finish

Modify the "Hello, World" example so that the MPI processes pass an integer variable from left to right, reporting and incrementing its value at each step. The master process should report when all processes are finished.



**Time**

**Process 0** **Print greeting**

**Increment variable**
**Notify next process**

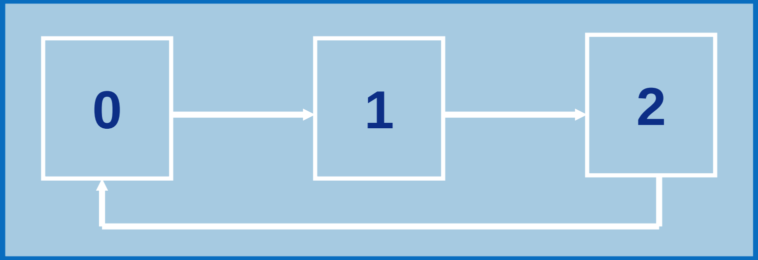
**Process 1** **Print greeting**

**Increment variable**
**Notify next process**

**Process 2** **Print greeting**

**Increment variable**
**Notify next process**

**Process 0** **Report that all work is finished**

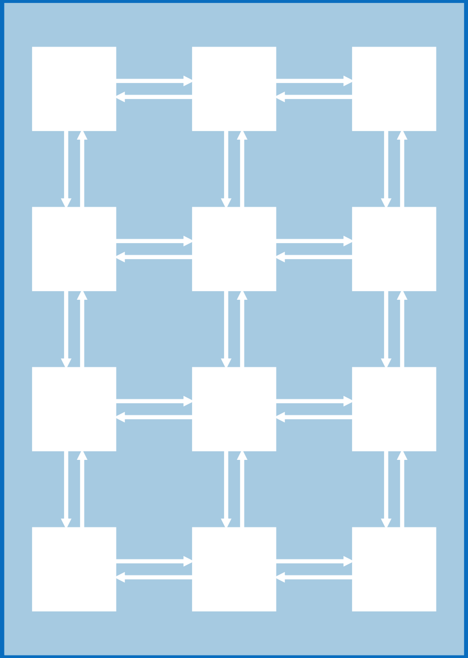


Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Thought Problem

**Consider a 2D mesh with nearest-neighbor communication, coded in MPI:**



```
do i = 1, neighbors
   call MPI_RECV (edge(1,i), len, MPI_REAL,
nbr(i),
                  tag, comm, requests(i),
ierr)
enddo

do i = 1, neighbors
   call MPI_SEND (edge(1,i), len, MPI_REAL,
nbr(i),
                   tag, comm, ierr)
enddo
```

*What is wrong with this code?*

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Definite Deadlock

```
do i = 1, neighbors
   call MPI_RECV (edge(1,i), len, MPI_REAL, nbr(i),
                  tag, comm, requests(i), ierr)
enddo

do i = 1, neighbors
   call MPI_SEND (edge(1,i), len, MPI_REAL, nbr(i),
                   tag, comm, ierr)
enddo
```

This program should always deadlock because each MPI process executes a blocking ‘receive’ before the corresponding ‘send’.

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Possible Deadlock

```
do i = 1, neighbors
   call MPI_SEND (edge(1,i), len, MPI_REAL, nbr(i),
                  tag, comm, ierr)
enddo

do i = 1, neighbors
   call MPI_RECV (edge(1,i), len, MPI_REAL, nbr(i),
                 tag, comm, requests(i), ierr)
enddo
```

This program may deadlock.

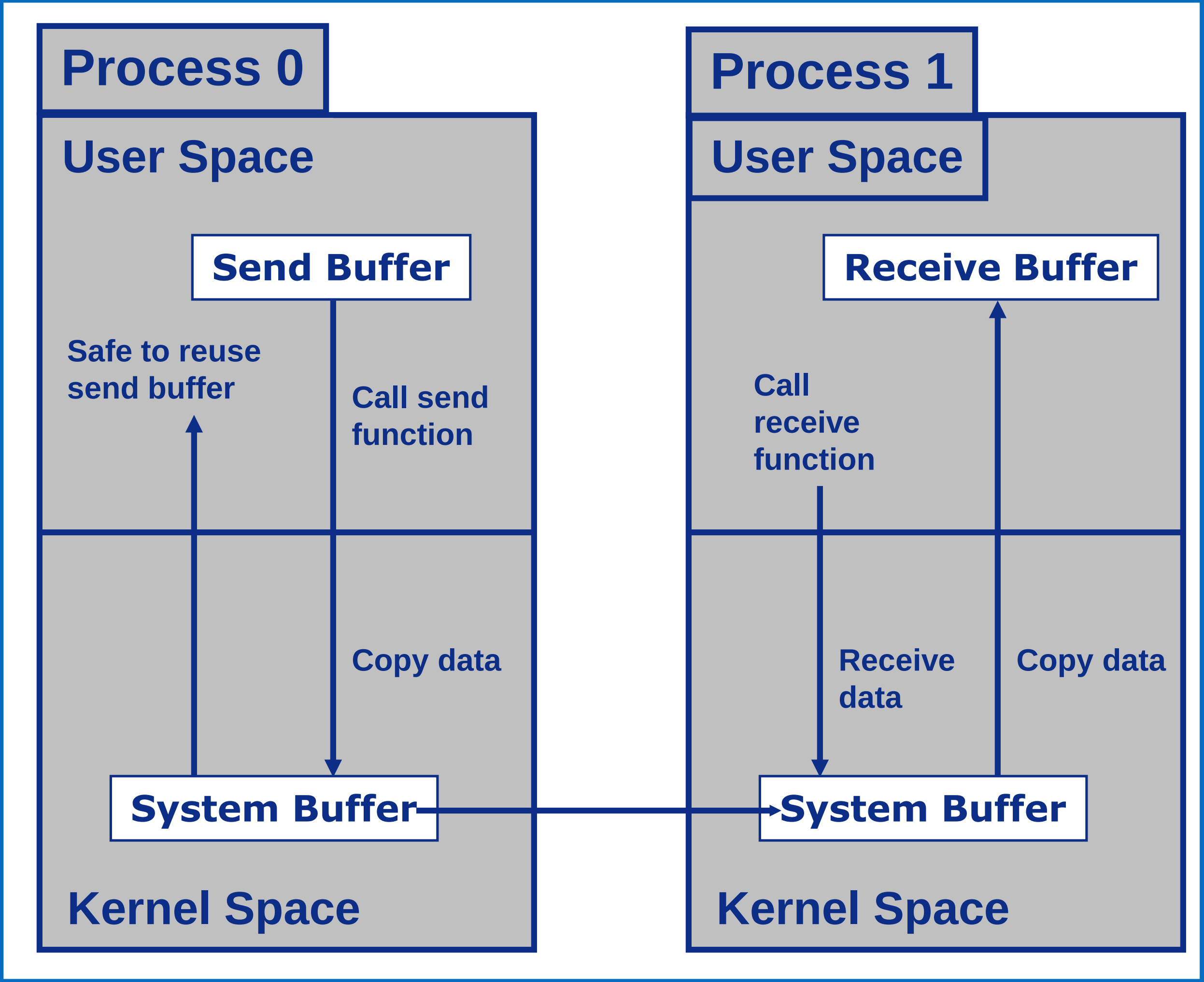
In some MPI implementations, MPI_SEND returns when the send buffer is copied to the system buffer. Result:

- Program might work if messages are small, but would deadlock when messages are larger than the system buffer

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Worth Repeating



## Key Points

- Sending and receiving data is a paired operation.

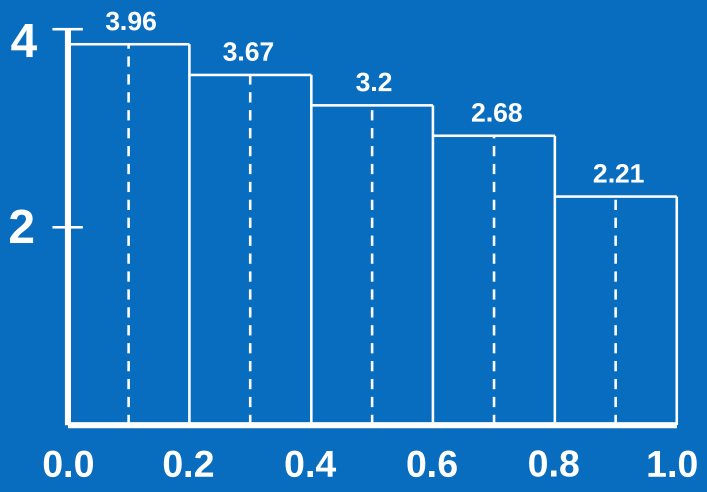
  **Ignoring this principle can result in deadlock.**

- Communication and synchronization are coupled.

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Activity 5: Parallelize Pi Integration

$f(x) = 4 / (1 + x^2)$



$$\frac{3.96 + 3.67 + 3.2 + 2.68 + 2.21}{5 \text{ intervals}} = 3.144$$

```
#include <stdio.h>
#define INTERVALS 100000

int main (int argc, char* argv[])
{
   int i;
   double h, x, pi = 0.0;

   h = 1.0 / (double)INTERVALS

   for (i = 0; i < INTERVALS; i++)
   {
      x = h * ((double)i - 0.5);
      pi += 4.0 / (1.0 + x * x);
   }
   pi *= h;
   printf (“Pi = %f\n”, pi);
}
```

**The serial version**

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

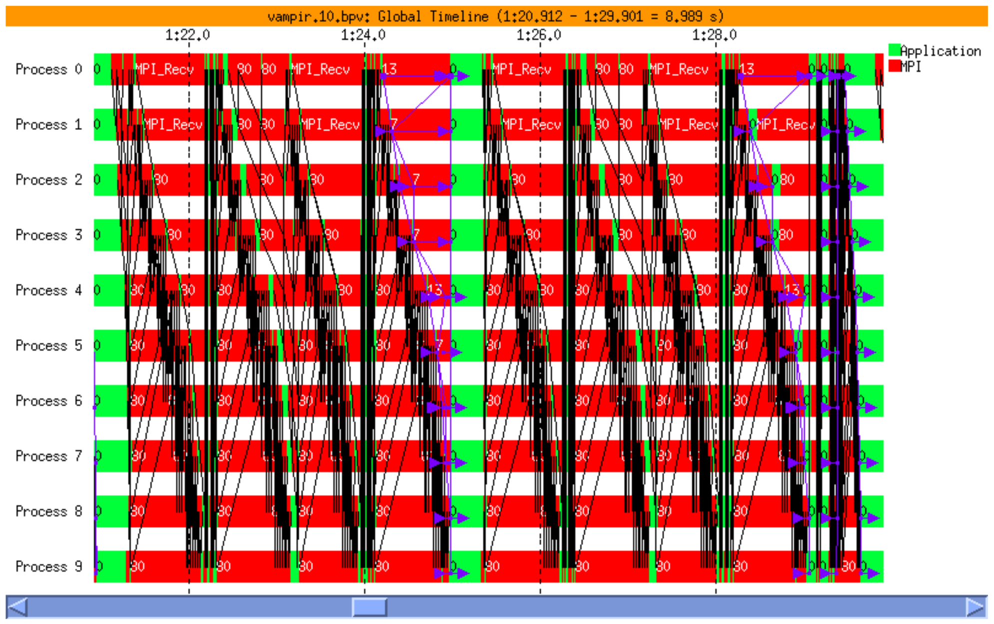
# No Deadlock

```
Process 0 code
do
<Prepare data>
   call MPI_SEND (data, len, MPI_REAL, 1,
                     tag, comm, ierr)
   Call MPI_REVC (buff, len, MPI_REAL, 1,
                 tag, comm, ierr)
enddo
Process 1 code
do
   call MPI_RECV (buff, len, MPI_REAL, 0,
                  tag, comm, ierr)
<Prepare data>
   call MPI_SEND (buff, len, MPI_REAL, 0,
                 tag, comm, ierr)
enddo
```

Though this program will not deadlock, there is another issue . . .

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

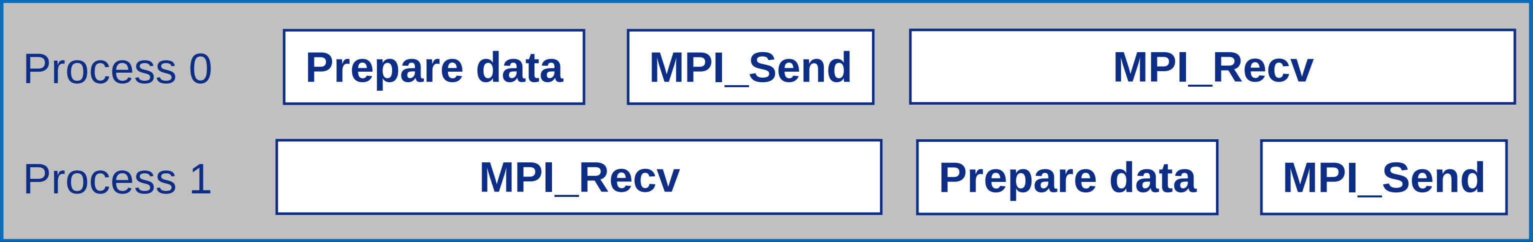
# Communication Bottlenecks



vampir.10.bpv: Global Timeline (1:20.912 - 1:29.901 = 8.989 s)

**Intel® Trace Analyzer MPI timeline showing severe communications bottleneck**

Examination of the code revealed that the application was performing data preparation in serial:



Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Agenda

What is MPI?

Core MPI Library

Basic Communication

Non-Blocking Communications

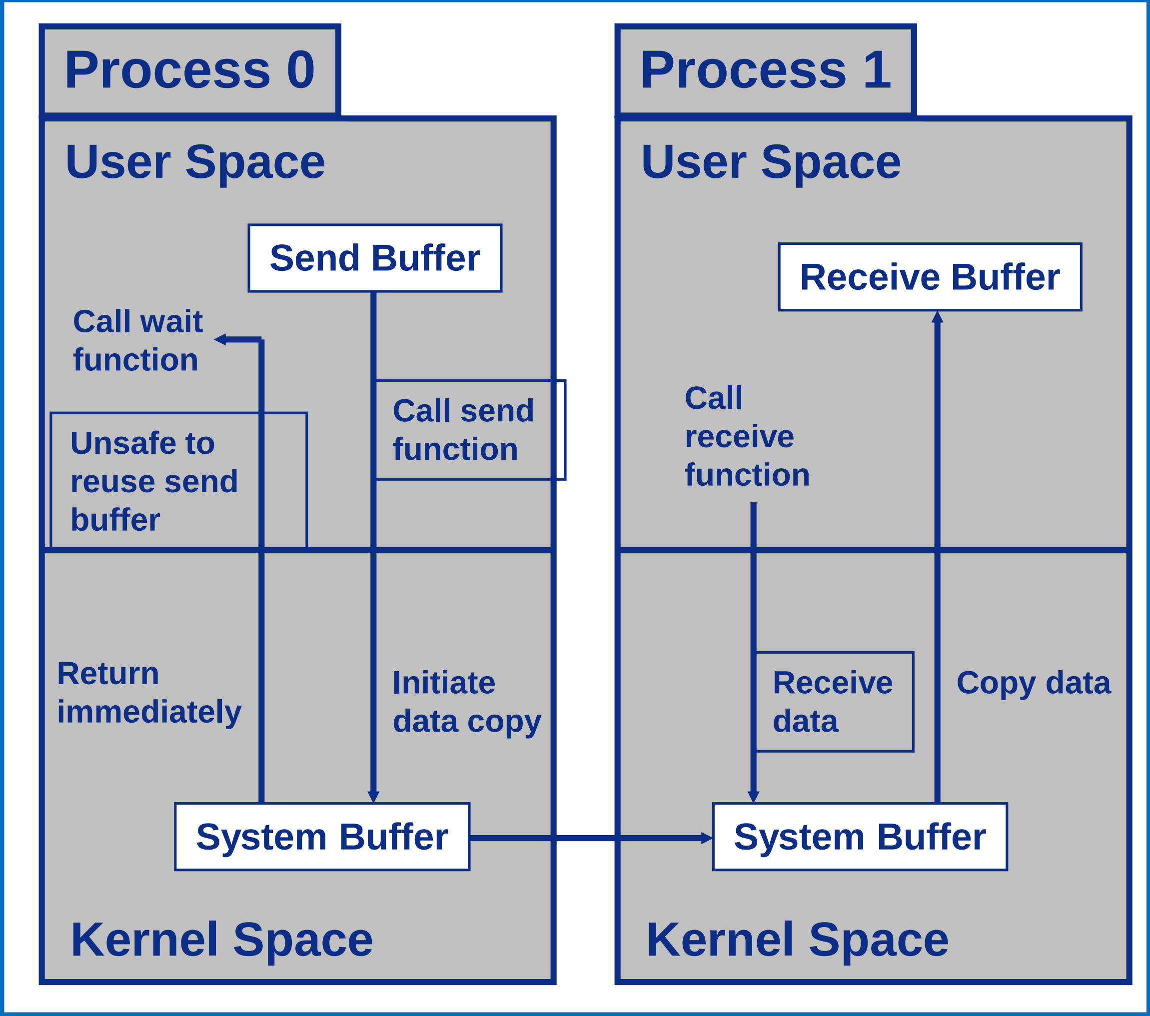
Collective Communications

Advanced MPI

- Useful Features
- Factors Affecting MPI Performance

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Non-Blocking Send



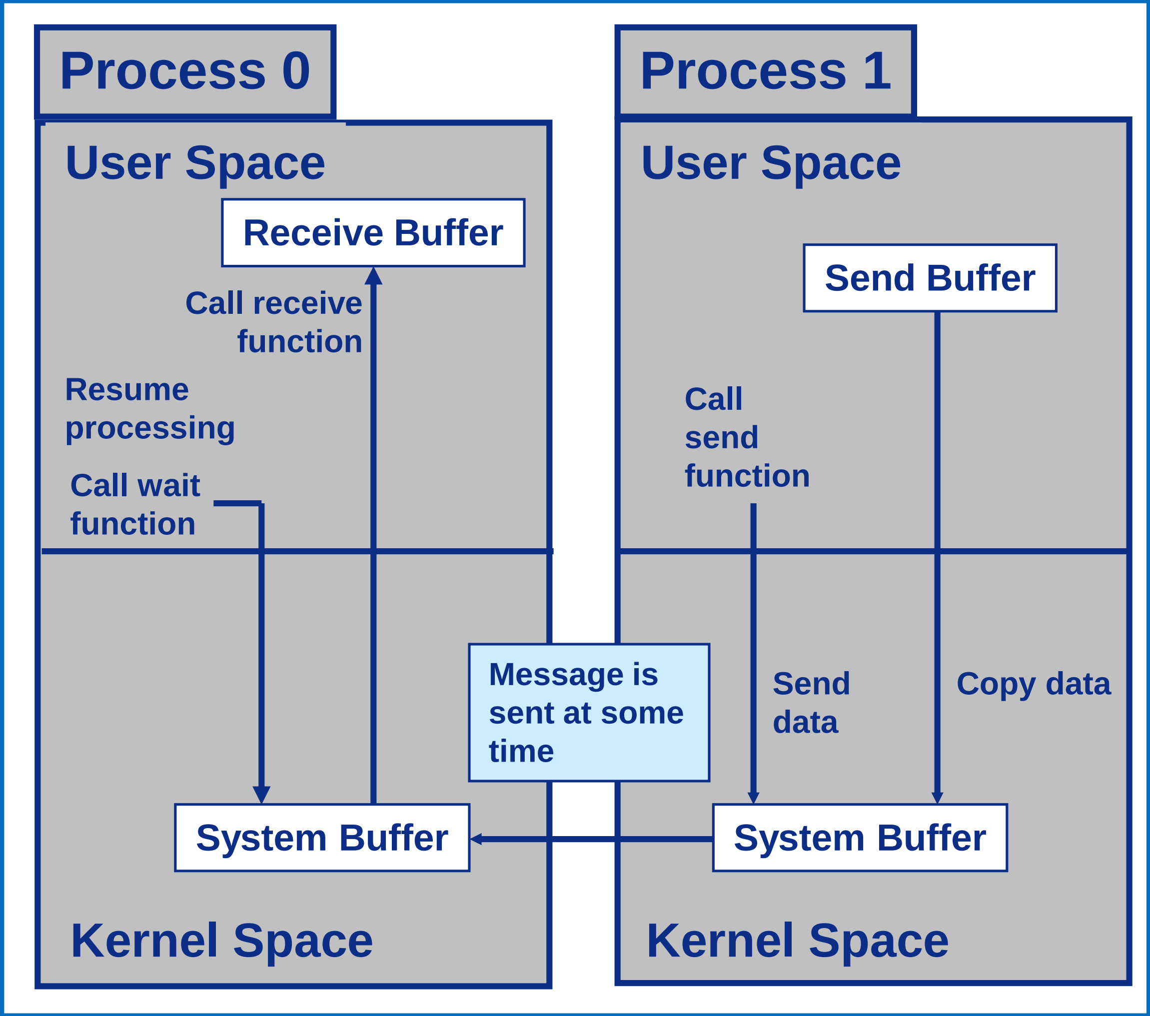
## Key Points

1. The sender does not wait for message buffering to complete.
2. The programmer must guarantee that the send buffer is not modified before the message is delivered.
3. It is difficult to debug MPI programs that ignore the previous point.

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Non-Blocking Receive


Process 0

User Space

Receive Buffer

Call receive function

Resume processing

Call wait function

System Buffer

Kernel Space

Message is sent at some time

Process 1

User Space

Send Buffer

Call send function

Send data

Copy data

System Buffer

Kernel Space

## Key Points

1. The receiver does not wait for message to be received.
2. The program may continue with other processing until message is needed.
3. Call *wait* function to block until message arrives. Cannot guarantee buffer contents until *wait* function returns.

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Immediate Send

```
Fortran:
      MPI_ISEND (buf, count, datatype, dest,
                  tag, comm, request, ierr)
      <type> buf(*)
      integer :: count, datatype, dest, tag,
                       comm, request, ierr
C:
      int MPI_Isend (void* buf, int count,
            MPI_Datatype datatype, int dest,
            int tag, MPI_Comm comm,
            MPI_Request* request)
```

`MPI_Isend` performs a non-blocking send of the specified data to the specified destination

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Immediate Receive

```
Fortran:
      MPI_IRECV (buf, count, datatype, source,
                  tag, comm, request, ierr)
      <type> buf(*)
      integer :: count, datatype, ierr, source,
                  tag, comm, request
C:
      int MPI_Irecv (void* buf, int count,
            MPI_Datatype datatype, int source,
            int tag, MPI_Comm comm,
            MPI_Request* request)
```

`MPI_Irecv` performs a non-blocking receive of the specified data from the specified source

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Confirming Delivery

```
Fortran:
      MPI_WAIT (request, status, ierr)
      integer :: request, ierr,
                  status(MPI_STATUS_SIZE)
C:
      int MPI_Wait (MPI_Request* request,
                         MPI_Status* status)
```

`MPI_Wait` returns when the operation that started the specified request gets completed

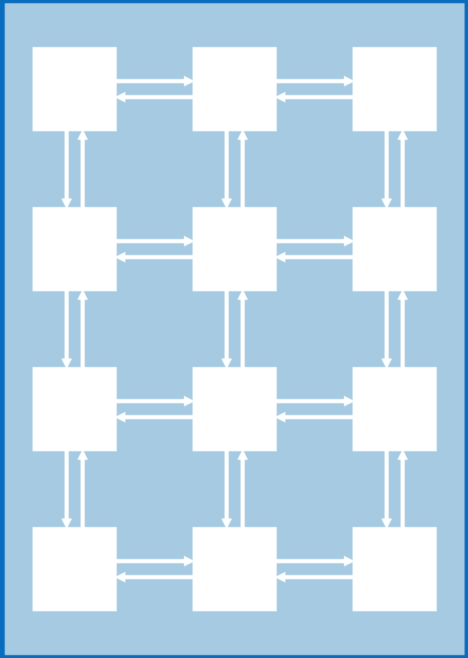
`MPI_Send` or `MPI_Recv` can also be used to confirm delivery

`MPI_Waitall` and `MPI_Waitany` can be used to wait on groups of messages without imposing order

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Thought Problem

Consider a two-dimensional mesh with nearest-neighbor communication, coded in MPI



```
do i = 1, neighbors
   call MPI_IRECV (edge(1,i), len, MPI_REAL,
nbr(i),
                   tag, comm, requests(i),
ierr)
enddo

do i = 1, neighbors
   call MPI_SEND (edge(1,i), len, MPI_REAL,
nbr(i),
                  tag, comm, requests(i),
ierr)
enddo
```

*Why does this code perform poorly?*

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Increased Parallelism

```
Process 0 code
do
<Prepare data>
   call MPI_ISEND (data, len, MPI_REAL, 1,
                   tag, comm, request, ierr)
   call MPI_REVC (buff, len, MPI_REAL, 1,
               tag, comm, ierr)
enddo
Process 1 code
Do
<Prepare data>
   call MPI_ISEND (data, len, MPI_REAL, 0,
                 tag, comm, request, ierr)
   call MPI_RECV (buff, len, MPI_REAL, 0,
                 tag, comm, ierr)

enddo
```

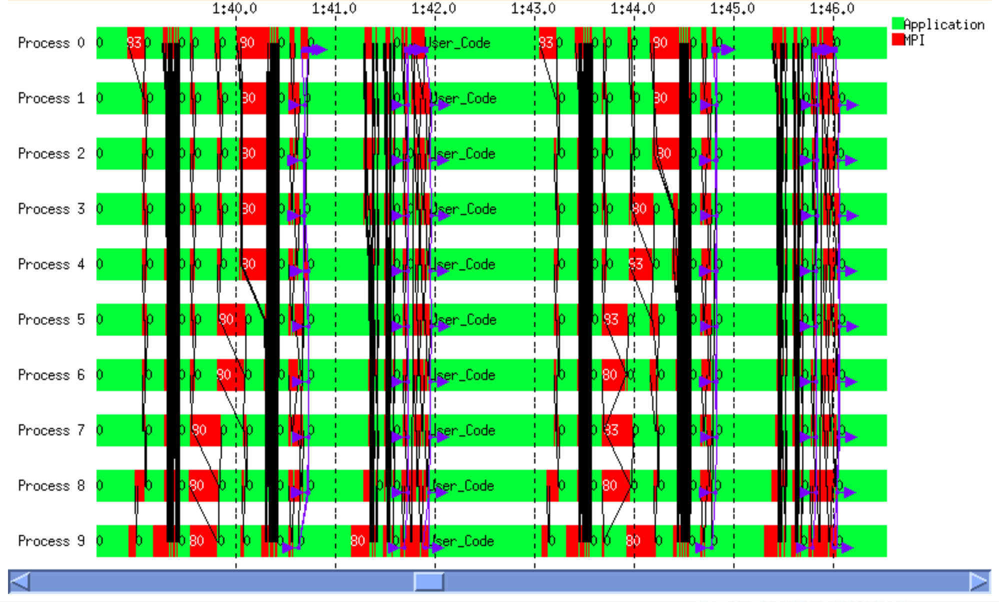
*How does this run?*

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Optimizing Communication

Parallel data preparation alleviates the bottleneck

| | | | |
|---|---|---|---|
| Process 0 | **Prepare data** | **MPI_Isend** | **MPI_Recv** |
| Process 1 | **Prepare data** | **MPI_Isend** | **MPI_Recv** |


vampir.10.bpv: Global Timeline (1:38.605 - 1:46.518 = 7.913 s)

Intel® Trace Analyzer MPI timeline showing improved computation to communication ratio

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Agenda

What is MPI?

Core MPI Library

Basic Communication

Non-Blocking Communications

Collective Communications

Advanced MPI
- Useful Features
- Factors Affecting MPI Performance

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Collective Communication

Functions to facilitate communication among all MPI processes in a communicator

Categories:

- Barrier and broadcast
- Gather and scatter
- Reduction

Collective communication functions can be optimized for the underlying network topology

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Broadcasting Data

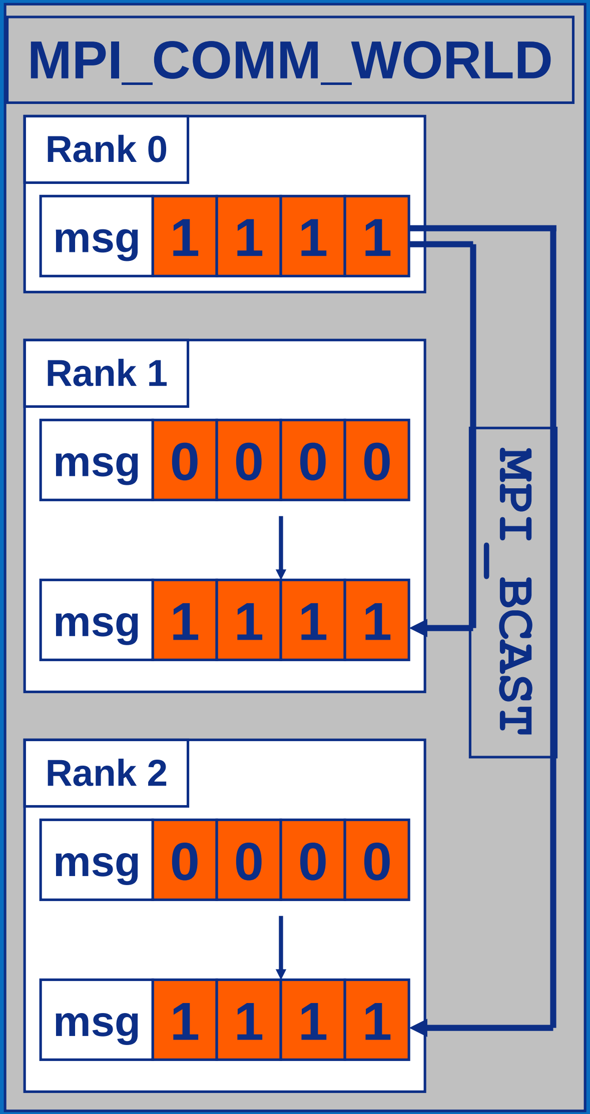
```
Fortran:
      MPI_BCAST (buf, count, datatype, root,
                  comm, ierr)
      <type> buf(*)
      integer :: count, datatype, root, comm,
                  ierr
C:
      int MPI_Bcast (void* buf, int count,
            MPI_Datatype datatype, int root,
            MPI_Comm comm)
```

`MPI_Bcast` sends the specified data to all processes in the communicator

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPI_BCAST Example

```
program BroadcastData
include ‘mpif.h’

integer, dimension(4) :: msg

call MPI_INIT (ierr)
call MPI_COMM_SIZE (MPI_COMM_WORLD, nprocs, ierr)
call MPI_COMM_RANK (MPI_COMM_WORLD, myrank, ierr)

if (myrank == 0) then
   msg = 1
else
   msg = 0
endif

call MPI_BCAST   &
   (msg, 4, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)

call MPI_FINALIZE (ierr)
end program BroadcastData
```



Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Data Reduction

```
Fortran:
      MPI_REDUCE (sendbuf, recvbuf, count,
                  datatype, operation, root,
                  comm, ierr)
      <type> sendbuf(*), recvbuf(*)
      integer :: count, datatype, operation,
                  root, comm, ierr
C:
      int MPI_Reduce (void* sendbuf,
            void* recvbuf, int count,
            MPI_Datatype datatype, MPI_Op op,
            int root, MPI_Comm comm)
```

`MPI_Reduce` performs the specified reduction operation on the specified data from all processes in the communicator

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

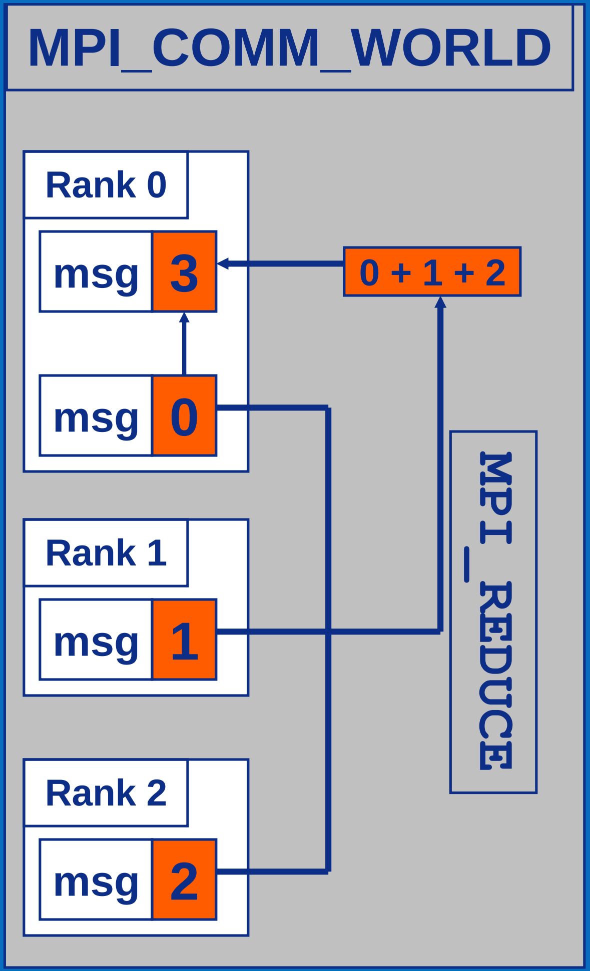
# MPI Reduction Operations

| **Operation** | **Function** |
|---|---|
| `MPI_SUM` | Summation |
| `MPI_PROD` | Product |
| `MPI_MIN` | Minimum value |
| `MPI_MINLOC` | Minimum value and location |
| `MPI_MAX` | Maximum value |
| `MPI_MAXLOC` | Maximum value and location |
| `MPI_LAND` | Logical AND |
| `MPI_BAND` | Bitwise AND |
| `MPI_LOR` | Logical OR |
| `MPI_BOR` | Bitwise OR |
| `MPI_LXOR` | Logical exclusive OR |
| `MPI_BXOR` | Bitwise exclusive OR |
| User-defined | It is possible to define new reduction operations |

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPI_REDUCE Example

```
program ReduceData
include ‘mpif.h’

integer :: msg, sum

call MPI_INIT (ierr)
call MPI_COMM_SIZE (MPI_COMM_WORLD, nprocs, ierr)
call MPI_COMM_RANK (MPI_COMM_WORLD, myrank, ierr)

sum = 0
msg = myrank

call MPI_REDUCE                                &
    (msg, sum, 1, MPI_INTEGER, MPI_SUM, 0,     &
     MPI_COMM_WORLD, ierr)

call MPI_FINALIZE (ierr)

end program ReduceData
```

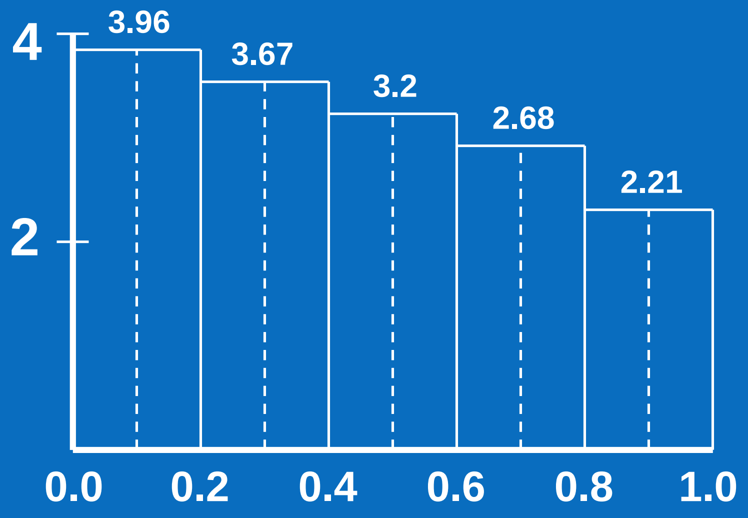


Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Activity 6: Pi Revisited

Simplify your parallel Pi integration using collective communication routines

$$f(x) = 4 / (1 + x^2)$$



$$\frac{3.96 + 3.67 + 3.2 + 2.68 + 2.21}{\text{5 intervals}} = 3.144$$

```
#include <stdio.h>
#define INTERVALS 100000

int main (int argc, char* argv[])
{
   int i;
   double h, x, pi = 0.0;

   h = 1.0 / (double)INTERVALS

   for (i = 0; i < INTERVALS; i++)
   {
      x = h * ((double)i – 0.5);
      pi += 4.0 / (1.0 + x * x);
   }
   pi *= h;
   printf (“Pi = %f\n”, pi);
}
```

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Activity 7: Master-worker

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Review

MPI is best for distributed-memory parallel systems

There are several MPI implementations

The MPI library is large but the core is small

MPI communication:

- Point-to-point, blocking
- Point-to-point, non-blocking
- Collective

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Core MPI

11 Out of 125 Total Functions

Program startup and shutdown

- `MPI_Init, MPI_Finalize`
- `MPI_Comm_size, MPI_Comm_rank`

Point-to-point communication

- `MPI_Send, MPI_Recv`

Non-blocking communication

- `MPI_Isend, MPI_Irecv, MPI_Wait`

Collective communication

- `MPI_Bcast, MPI_Reduce`

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Agenda

What is MPI?

Core MPI Library

Basic Communication

Non-Blocking Communications

Collective Communications

Advanced MPI
- **Useful Features**
- Factors Affecting MPI Performance

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Useful Features

Communicators

Timing routines

User-defined data types

Inter-communicators

Topologies

Persistent communication

New features in MPI 2.0

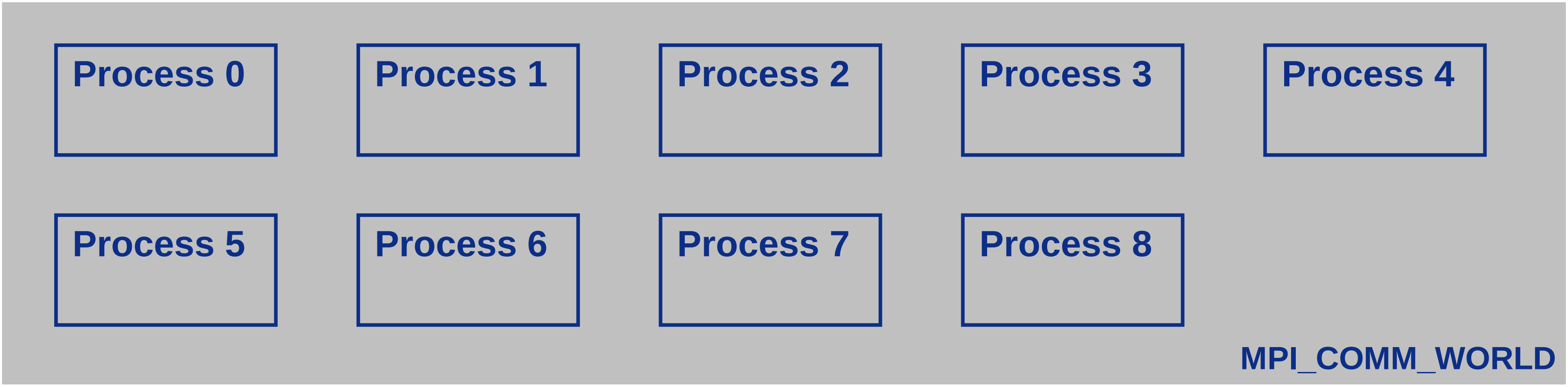
Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Communicators

MPI uses communicators to direct message traffic.

MPI_COMM_WORLD is default communicator of all processes.

Program may define other communicators.



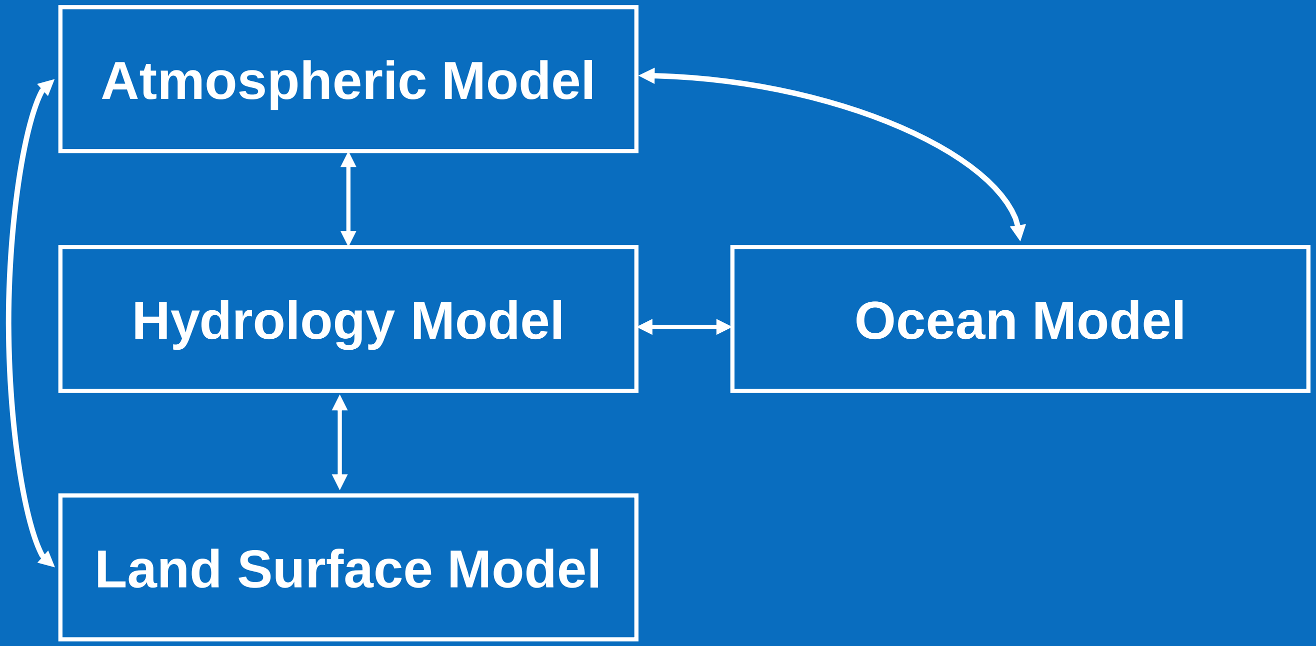
Process 0
Process 1
Process 2
Process 3
Process 4
Process 5
Process 6
Process 7
Process 8

MPI_COMM_WORLD

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Communicators: Example

Each model conveniently maps to its own communicator.



Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Timing Routines

Performance is a major design goal of MPI

Therefore, MPI provides functions to measure runtime:

- `MPI_Wtime` – returns the number of seconds from some arbitrary starting point
- `MPI_Wtick` – used to determine timer resolution

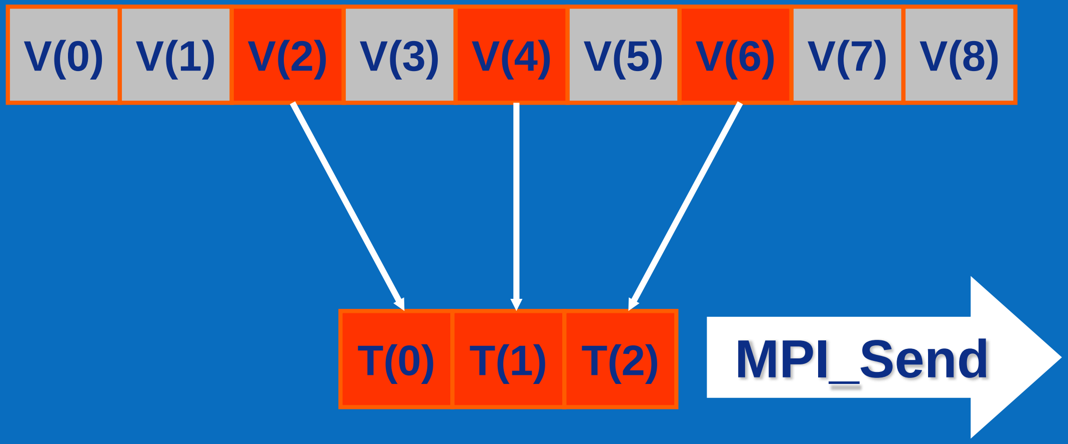
```
int start = MPI_Wtime ();

// Do some work

int end = MPI_Wtime ();
int runtime = end - start;
```

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Handling Non-Contiguous Data

## How to send only the red elements of V?



This method requires inefficient copy of non-contiguous data

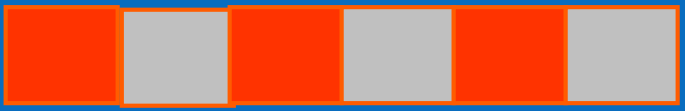
One possibility, copy these elements to a temporary array before sending.

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Derived Data Types

How to send only the red elements of V, while avoiding copying non-contiguous data to a temporary array?

Define a new data type, in this case a vector with stride of two from original.



```
MPI_Type_vector (3, 1, 2, MPI_REAL, &vType);
MPI_Send (V[2], 1, vType, dest, tag, MPI_COMM_WORLD);
```

MPI provides several functions to define new data types. MPICH contains optimizations for user-defined type.

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

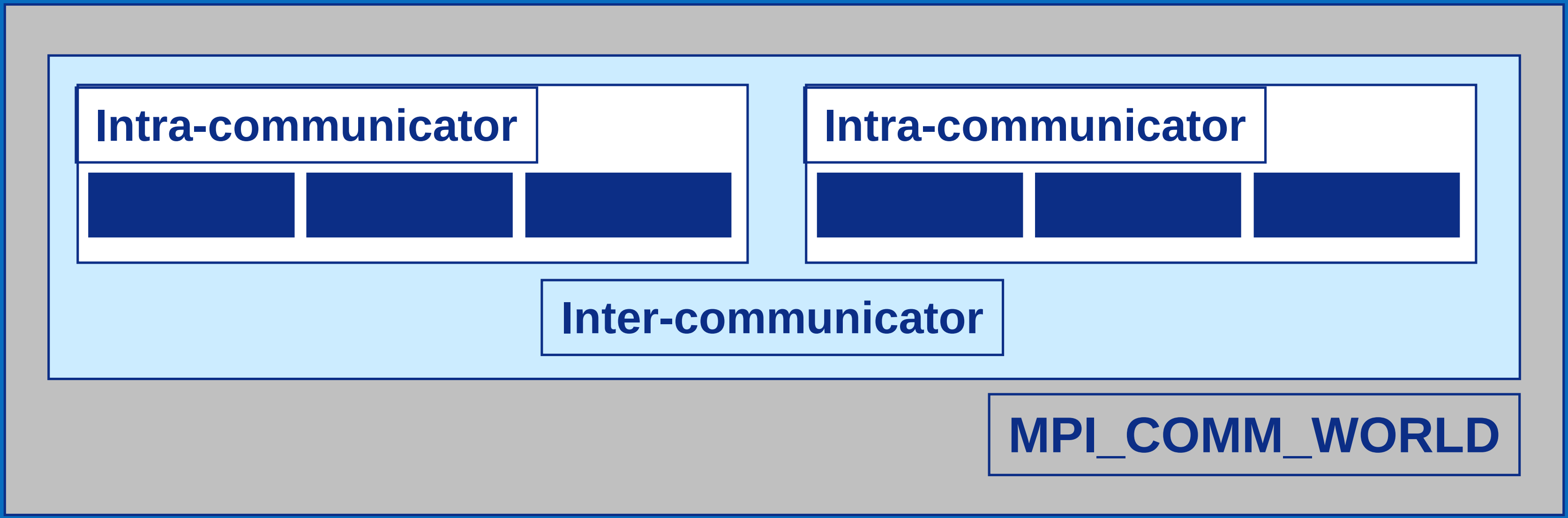
# Inter-communicators

Useful for library development to hide library messages from calling program

Can be used to divide the MPI processes among different parallel computations (for example, multi-block CFD)

Functions:

- `MPI_Comm_create, MPI_Comm_dup, MPI_Comm_split`



Intra-communicator

Intra-communicator

Inter-communicator

MPI_COMM_WORLD

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Topologies

MPI provides functions to map application topology to processor topology

| | | |
|---|---|---|
| **0**<br>**(0,0)** | **1**<br>**(0,1)** | **2**<br>**(0,2)** |
| **3**<br>**(1,0)** | **4**<br>**(1,1)** | **5**<br>**(1,2)** |
| **6**<br>**(2,0)** | **7**<br>**(2,1)** | **8**<br>**(2,2)** |
| **9**<br>**(3,0)** | **10**<br>**(3,1)** | **11**<br>**(3,2)** |

For example, consider an application that maps naturally onto a 4 x 3 Cartesian grid. Nearest neighbors are not obvious from process rank

Use MPI_Cart_create function to create a new communicator with the specified topology

Use MPI_Cart_shift to identify nearest neighbors, even for periodic grids

MPI topologies can be user-defined graphs

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPI 2.0 – More Features

One-sided communication

- MPI_Put, MPI_Get
- Uses Remote Memory Access (RMA)
- Separates communication from synchronization

Dynamic process spawning

- MPI_Spawn function
- Creates a new inter-communicator

Collective communication can be used across disjoint intra-communicators

Parallel I/O

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPI 2.0 – Better Language Support

Fortran 90/95

- Array sections
- Modules

C++

- MPI class library
- MPI functions are methods within MPI classes

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Profiling MPI Communications

MPICH* contains profiling tools that are not installed by default

Jumpshot-2

- Installation
  - Install Java Development Kit
  - Additional MPICH configuration option:
  - `-mpe-opts=--with-java=/usr/local/jdk`
- Usage
  - Add `-mpilog` option to MPICH compilation, e.g.:
  - `mpicc -mpilog -o prog.exe prog.c`

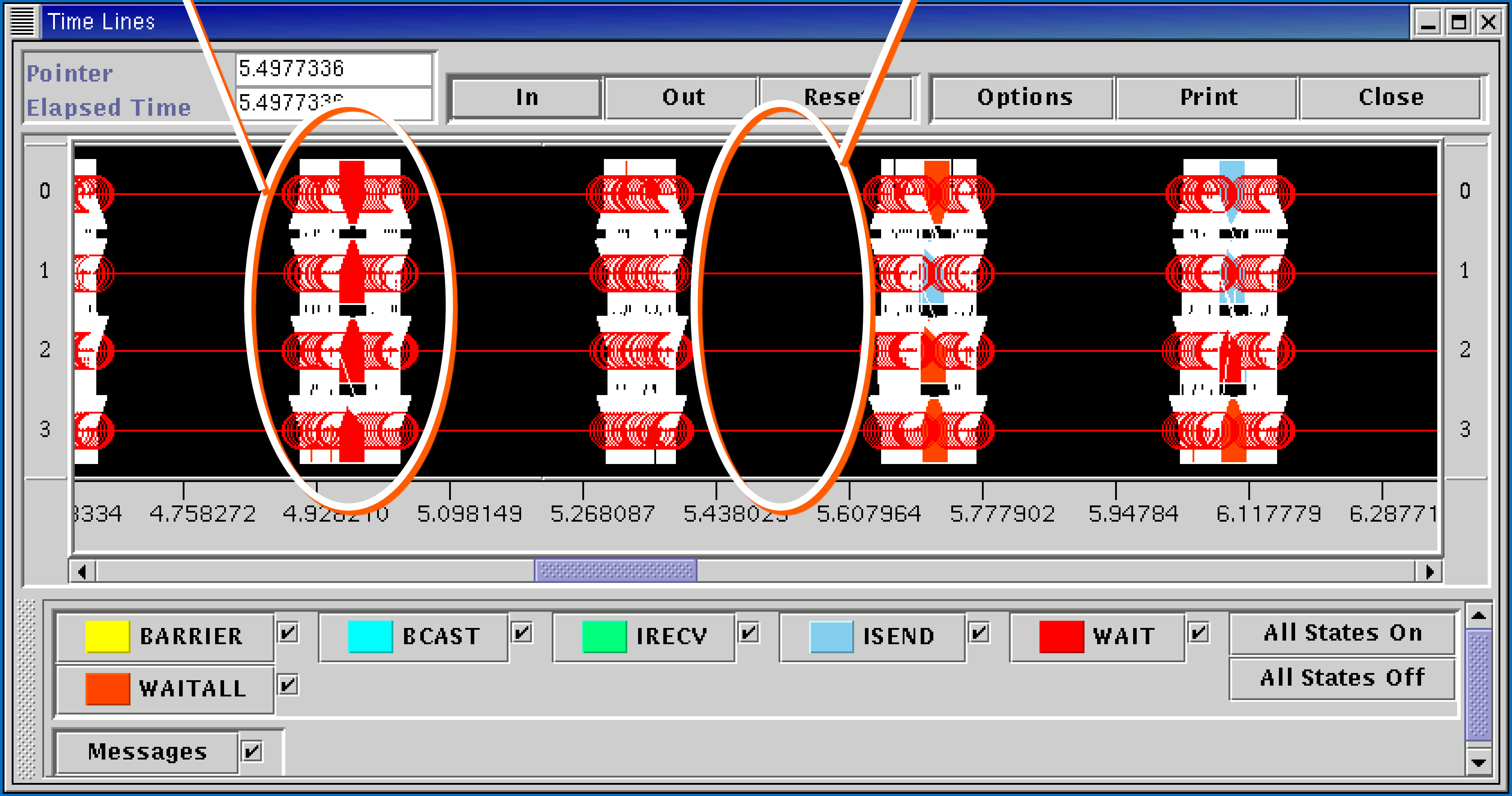
Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Communications Timeline

Communication

Computation



Pointer 5.4977336

Elapsed Time 5.49773

🟨 BARRIER ☑

🟦 BCAST ☑

🟩 IRECV ☑

🟦 ISEND ☑

🟥 WAIT ☑

🟧 WAITALL ☑

Messages ☑

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Agenda

What is MPI?

Core MPI Library

Basic Communication

Non-Blocking Communications

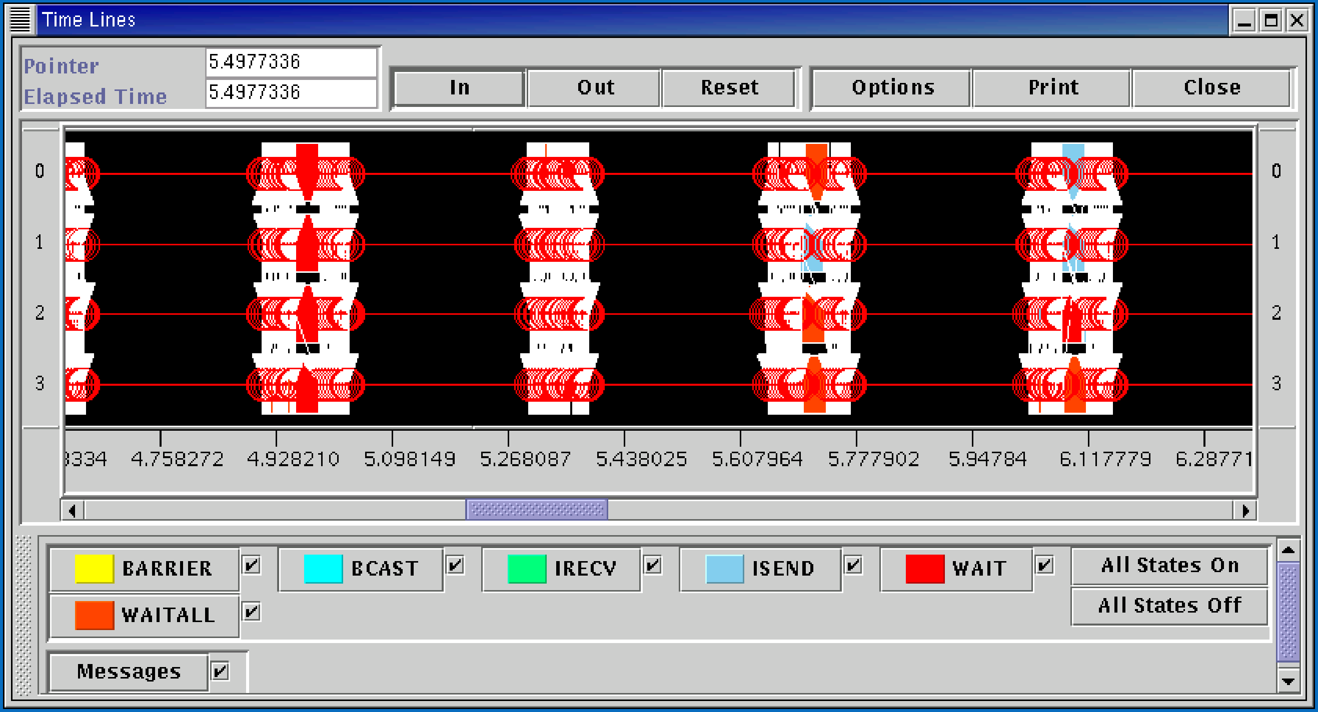
Collective Communications

Advanced MPI
- Useful Features
- **Factors Affecting MPI Performance**

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Network Interconnect

- Timeline shows a repeating pattern of computation and communication but does not indicate a communications bottleneck



Pointer 5.4977336
Elapsed Time 5.4977336

🟨 BARRIER ☑
🟦 BCAST ☑
🟩 IRECV ☑
🟦 ISEND ☑
🟥 WAIT ☑
🟧 WAITALL ☑
Messages ☑

- The only way to improve the performance of this application is to use a faster network interconnect

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Network Interconnect

Fast Ethernet

Gigabit Ethernet

Myrinet*

InfiniBand*

Quadrics*

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Message Size

In general, given the same amount of data, it is better to send fewer large messages than many small messages

- Fewer MPI function calls, less system overhead
- Fewer messages, less TCP/IP overhead

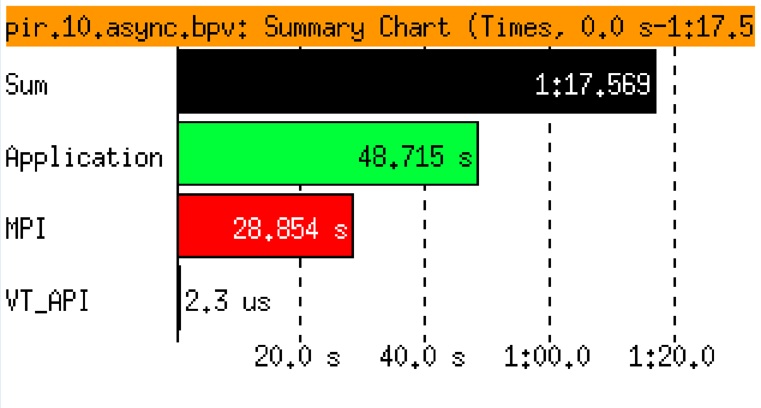
User-defined data types in MPI can help pack several messages into one

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

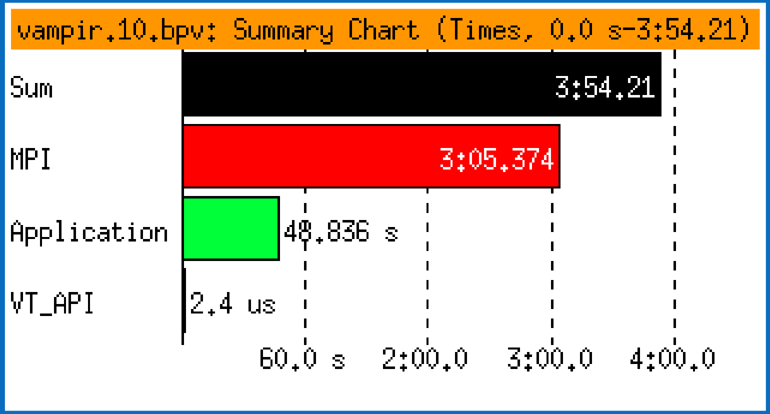
# Computation: Communication

Which of these applications is likely to show better parallel speedup and scalability?

## Application 1



## Application 2



*Obviously, Application 1 has a better ratio of computation to communication!*

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Overlapping Communication & Computation

```
do i = 1, neighbors
   call MPI_IRECV (edge(1,i), len, MPI_REAL, nbr(i),
                   tag, comm, requests(i), ierr)
enddo

do i = 1, neighbors
   call MPI_ISEND (edge(1,i), len, MPI_REAL, nbr(i),
                  tag, comm, requests(i), ierr)
enddo

! Perform computation not requiring message data

call MPI_WAITALL ((2 * neighbors), requests, status, ierr)
```

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPICH Communication Devices

ch_p4

- General communication device
- Support for SMP nodes
- Supports heterogeneous nodes
- MPICH default communication device

ch_p4mpd

- Only supports homogeneous, uni-processor clusters
- Provides faster, more scalable job startup

ch_shmem – Best for machines supporting shmem

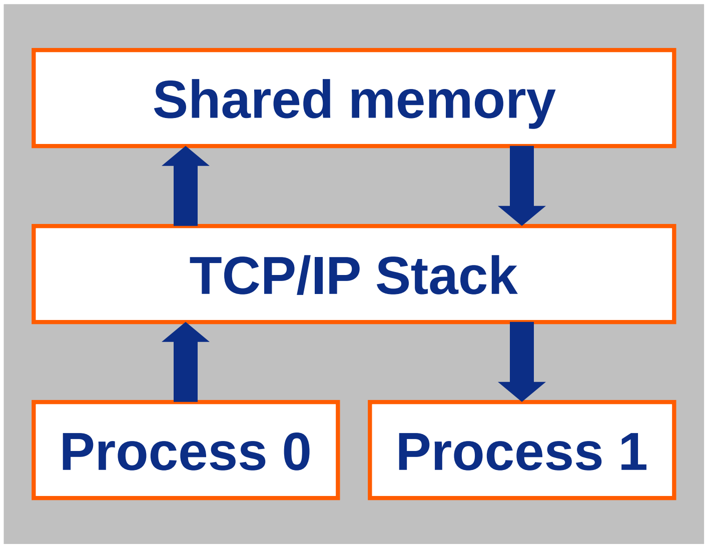
ch_globus2 – Grid-enabled MPICH for Globus

ch_nt – MPICH* for Microsoft Windows*

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

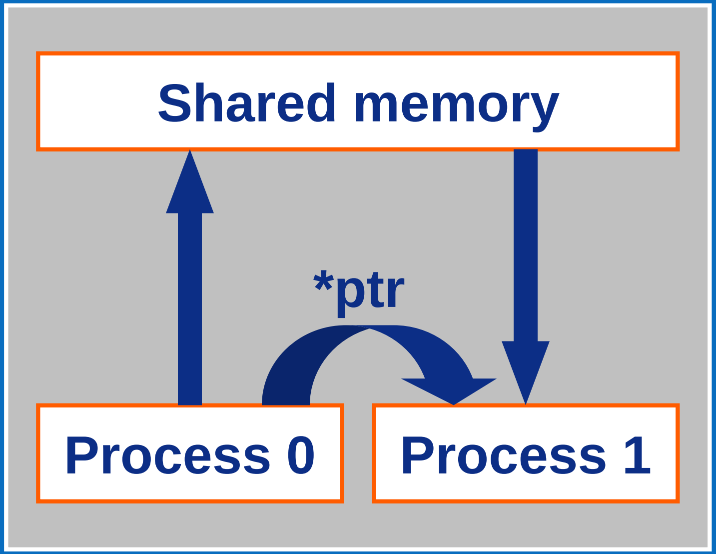
# MPICH SMP Support

**Intra-node communication through TCP/IP**



**Dual-CPU node**

**Intra-node communication through memcpy**



**Dual-CPU node**

**Configure MPICH with the –comm=shared option**
**Set the MPI_MAX_CLUSTER_SIZE environment variable**

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Backup

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Getting Started with MPICH

Step 1: Download most recent package

- ftp://ftp.mcs.anl.gov/pub/mpi/mpich.tar.gz

Step 2: Unpack the archive in /usr/local

Step 3: Patch MPICH

- Download the latest cumulative patch
  - ftp://ftp.mcs.anl.gov/pub/mpi/patch/1.2.5/patch.all
- Copy patch file(s) to /usr/local/mpich-1.2.5
- Execute:

```
patch -p0 < patch.all
```

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Configuring MPICH

Step 4: Configure

- Important configuration options
    - Communication device
    - Compilers
    - SMP support
- Execute:

```
configure -with-device=ch_p4 \
      -cc=icc -fc=ifc -f90=ifc -c++=icc \
      -prefix=/usr/local/mpich-1.2.5
```

- Check the configuration log for errors

Copyright © 2006, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# Installing MPICH

Step 5: Build

- Execute: `make >& make.log`
- Check the make log for errors

Step 6: Install

- Execute: `make install`

At this point, anyone logging into the system should have access to the MPI compiler scripts, libraries, mpirun, and machine files.

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

# MPI_GET_COUNT and Wild Card

**Fortran:**

```
    MPI_GET_COUNT (status, datatype, count, ierr)
    integer :: status(MPI_STATUS_SIZE), count,
               datatype, ierr
```

**C:**

```
    int MPI_Get_count (MPI_Status* status,
             MPI_Datatype datatype, int* count)
```

`MPI_Get_count` returns number of elements received.

`MPI_Recv` may use "wild card" values:

`MPI_ANY_TAG` `MPI_ANY_SOURCE`

Copyright © 2006, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. *Other brands and names are the property of their respective owners.

intel®

Leap ahead™