


Beyond MPI-1.X

Dr Graham E Fagg
CS-594
Spring 2001


G.E.Fagg UTK-CS594 Spring2001 1



What we are here for

- All the stuff extra stuff in the MPI-2 Spec and a few things in the MPI-1 Spec you have not thought about yet
 - MPI 2 is it really out there ?
 - Do you trust your implementation ?
- Some of this is a warm up for the MetaComputing classes coming up next.


G.E.Fagg UTK-CS594 Spring2001 2



Overview

- MPI-1 what it missed out
 - And why...
- MPI-2 what it included
 - Language issues
 - Process Management
 - Establishing Communication


G.E.Fagg UTK-CS594 Spring2001 3



Overview

- MPI-2 what it included
 - Single sided Communication
 - Intercommunicator Collective Operations
 - I/O including Parallel IO (PIO)


G.E.Fagg UTK-CS594 Spring2001 4



Overview

- MPI-1 the other things
 - How profiling works
 - Using profiling libraries such as JUMPSHOT
 - Building a library yourself
 - How do collectives work
 - Using communicators for distributed computing

G.E.Fagg UTK-CS594 Spring2001 5



Overview

- Some home work
 - Report
- Some practical work
 - Profiling / benchmarking / debugging
 - Collectives

G.E.Fagg UTK-CS594 Spring2001 6

What MPI-1 Missed out

- ⚡ A lot of things that were hard to agree on for a **standard**
- ⚡ Anything that might not run well on a early 90s late 80s MPP
 - ⚡ Remember who made the standard
 - ⚡ NEC, IBM, SGI, Intel, HP.....

G.E.Fagg UTK-CSS94 Spring2001 7

Basics of MPI-1

- ⚡ **All** characteristics of message passing are contained within **communicators**
- ⚡ Communicators contain:
 - ⚡ Process lists or groups
 - ⚡ Connection/communication structures **topologies**
 - ⚡ System derived message tags/envelopes to separate messages from each other

G.E.Fagg UTK-CSS94 Spring2001 8

MPI-1 internals Communicators

- ⚡ All processes in an MPI-1 application belong to a global communicator called **MPI_COMM_WORLD**
- ⚡ All other communicators are derived from this global communicator.
- ⚡ Communication can only occur within a communicator.
 - ⚡ Safe communication

G.E.Fagg UTK-CSS94 Spring2001 9

MPI-1 Internals Processes

- ⚡ All process groups are derived from the membership of the **MPI_COMM_WORLD** communicator.
 - ⚡ i.e.. no external processes.
- ⚡ MPI-1 process membership is **static** not **dynamic** like PVM or LAM 6.X
 - ⚡ simplified consistency reasoning
 - ⚡ fast communication (fixed addressing) even across complex topologies.
 - ⚡ interfaces well to simple run-time systems as found on many **MPPs**.

G.E.Fagg UTK-CSS94 Spring2001 10

MPI-1 Application

The diagram illustrates the MPI-1 application structure. A large red outline represents the **MPI_COMM_WORLD** communicator, which contains several red circular nodes. Within this red boundary, a smaller blue outline represents a **Derived_Communicator**, which contains four blue circular nodes. Each blue node is also surrounded by a red circle, indicating that all processes in the application belong to the global MPI_COMM_WORLD communicator.

G.E.Fagg UTK-CSS94 Spring2001 11

Disadvantages of MPI-1

- ⚡ **Static process model**
 - ⚡ If a process fails, all communicators it belongs to become invalid. I.e. No fault tolerance.
 - ⚡ Dynamic resources either cause applications to fail due to loss of nodes or make applications inefficient as they cannot take advantage of new nodes by starting/spawning additional processes.
 - ⚡ When using a dedicated MPP MPI implementation you cannot usually use off-machine or even off-partition nodes.

G.E.Fagg UTK-CSS94 Spring2001 12

Disadvantages of MPI-1

- ⚡ Multi-language operation is not supported
 - ⚡ MPI specifies both ANSI C and F77 binding
 - ⚡ No agreed standard for data type conversions between languages even upon the same architecture
 - ⚡ Communicators cannot be passed between different language modules due to their representation
 - ⚡ In F77 a communicator is an INTEGER
 - ⚡ In C it is usually a pointer to a structure
 - ⚡ non standard methods exist in different implementations

G.E.Fagg UTK-CSS94 Spring2001 13

MPI-2

- ⚡ Problem areas and needed additional features identified in MPI-1 are being addressed by the MPI-2 forum.
- ⚡ These include:
 - ⚡ inter-language operation
 - ⚡ dynamic process control / management
 - ⚡ parallel IO
 - ⚡ extended collective operations
- ⚡ Support for inter-implementation communication/control is **not** being considered.
 - ⚡ See the MetaComputing lectures for more...

G.E.Fagg UTK-CSS94 Spring2001 14

MPI-2: Language Issues

- ⚡ Think back to MPI-1
 - ⚡ What is a communicator in 'C'?
 - ⚡ What is a communicator in 'Fortran'?
 - ⚡ Does it depend on the implementation?
 - ⚡ How could I write a program in both 'C' and 'Fortran' and have them pass communicators?

G.E.Fagg UTK-CSS94 Spring2001 15

MPI-2: Language Issues

- ⚡ MPI-2 Handle conversion
 - ⚡ Fortran to C/C++
 - ⚡ 'C' Wrappers convert Fortran handles
 - ⚡ C/C++ to Fortran
 - ⚡ Not supported
 - ⚡ Why do you think not ??
 - ⚡ C to C++
 - ⚡ Overloading C++ operators called on the C++ side
 - ⚡ C++ to C
 - ⚡ Not supported
 - ⚡ Want to write a new C++ compiler ??

G.E.Fagg UTK-CSS94 Spring2001 16

MPI-2: Language Issues

- ⚡ All wrappers of the form:
 - ⚡ MPI_<C_CLASS> MPI_<CLASS>_f2c (MPI_Fint handle)
 - ⚡ All handles in fortran are integers (INT*4)

G.E.Fagg UTK-CSS94 Spring2001 17

MPI-2: Language Issues

```

! FORTRAN PROCEDURE
SUBROUTINE MPI_SEND_ONEINT (COMM, I, IERR)
INTEGER COMM, I, IERR
CALL C_ROUTINE (COMM, I, IERR)
RETURN
END

```

G.E.Fagg UTK-CSS94 Spring2001 18

MPI-2: Language Issues

```

void c_routine (MPI_Fint *fh, MPI_Fint *i, MPI_Fint *ierr)
{
    MPI_Comm fcom;

    mycom = MPI_Comm_f2c (fh);
    MPI_Send (i, 1, MPI_INT, 1, 1, mycom);
}

```

G.E.Fagg UTK-CSS94 Spring2001 19

MPI-2: Language Issues

- Added a C++ binding
 - Constants and names the same as C except when they have been made into an object or special class

G.E.Fagg UTK-CSS94 Spring2001 20

MPI-2: Language Issues

- C++ binding
 - C++ has types C does not such as
 - MPI::COMPLEX, MPI::BOOL etc
 - Uses a namespace called MPI that contains all object types such as:
 - Class comm; class Cartcomm, class Datatype..

G.E.Fagg UTK-CSS94 Spring2001 21

MPI-2: Process Management

- In MPI-1
 - You get a set of process after you call MPI_Init ()
 - You keep them until you call MPI_Finalize
 - Or you halt...

G.E.Fagg UTK-CSS94 Spring2001 22

MPI-2: Process Management

- How do you add more nodes to an already running MPI-1 application ??

G.E.Fagg UTK-CSS94 Spring2001 23

MPI-2: Process Management

- How would we handle a node failure ??

G.E.Fagg UTK-CSS94 Spring2001 24

MPI-2: Process Management

- How could we couple two or more applications ??

G.E.Fagg UTK-CSS94 Spring2001 25

MPI-2: Process Management

- How could we couple two or more applications ??
- Running using two or more different MPI implementations, such as one on an IBM SP and the other on a SGI/Cray ??

G.E.Fagg UTK-CSS94 Spring2001 26

MPI-2: Process Management

- MPI-2 provides both a Spawn (or remote start) call
 - Depending on the implementation you have LAM 6.X+ does MPICH 1.3.1 does not
 - Different vendor versions do
 - Such as NEC/SUN

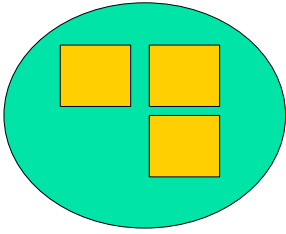
G.E.Fagg UTK-CSS94 Spring2001 27

MPI-2: Process Management

- Two flavors
 - MPI_Comm_spawn ()
 - Starts new processes from a single binary and returns an intercommunicator to them
 - MPI_Comm_spawn_multiple ()
 - Starts new processes from more than one binary

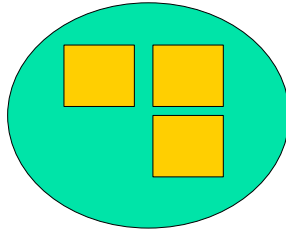
G.E.Fagg UTK-CSS94 Spring2001 28

MPI-2: Process Management



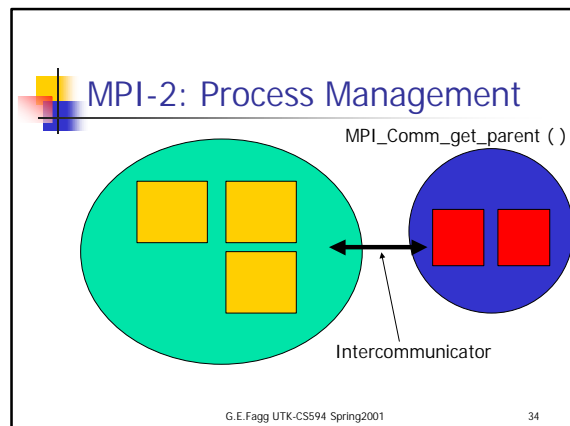
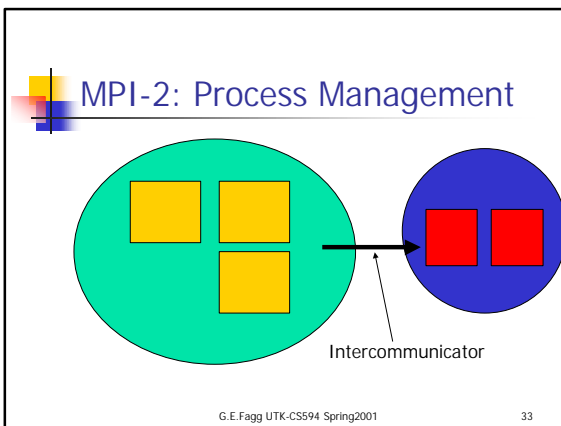
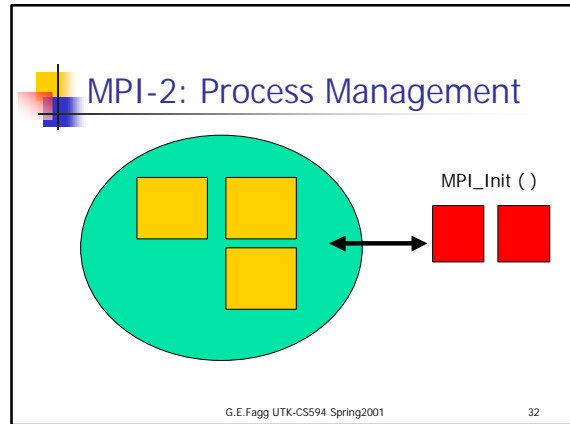
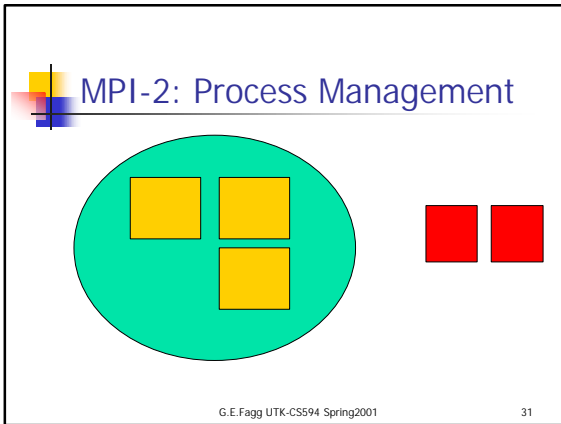
G.E.Fagg UTK-CSS94 Spring2001 29

MPI-2: Process Management



MPI_Comm_spawn (other..)

G.E.Fagg UTK-CSS94 Spring2001 30



MPI-2: Process Management

- ⌚ The creating processes get an intercommunicator to the **child** group as soon as the spawn call returns..
 - ⌚ They can send to it before the children have called MPI_Init ()

G.E.Fagg UTK-CS594 Spring2001 35

MPI-2: Process Management

- ⌚ Is the Intercommunicator functionality enough to allow for a full dynamic process model with efficient communications ?
 - ⌚ Hints
 - ⌚ What are the limits of an intercommunicator
 - ⌚ How are multiple child groups

G.E.Fagg UTK-CS594 Spring2001 36

MPI-2: Process Management

Intercommunicator

G.E.Fagg UTK-CSS94 Spring2001 37

MPI-2: Process Management

G.E.Fagg UTK-CSS94 Spring2001 38

MPI-2: Process Management

Can we do this ??

G.E.Fagg UTK-CSS94 Spring2001 39

MPI-2: Process Management

- Can the two children groups create a direct connection?
- using an MPI communicator create / merge call of some kind?

G.E.Fagg UTK-CSS94 Spring2001 40

MPI-2: Process Management

We do not have any overlapping communicators between the three groups...

G.E.Fagg UTK-CSS94 Spring2001 41

MPI-2: Process Management

Can not do collectives directly

G.E.Fagg UTK-CSS94 Spring2001 42

MPI-2: Establishing communications

- How do we do communications between MPI applications that have not been started by each other?

G.E.Fagg UTK-CSS94 Spring2001 43

MPI-2: Establishing communications

- MPI-2 has a TCP socket style abstraction
- Process can wait and accept connections from other processes
- Server – Client interface

G.E.Fagg UTK-CSS94 Spring2001 44

MPI-2: Establishing communications

- How does the client find the server ?
- MPI does not mandate a nameservice but they did make an interface for one.

G.E.Fagg UTK-CSS94 Spring2001 45

MPI-2: Establishing communications

- Server Side
 - Three routines
 - MPI_Open_port (info, port_name)
 - Creates a port that other can connect to
 - MPI_Close_port (port_name)
 - MPI_Comm_accept (port_name, info, root, comm, newcomm)
 - comm is a intracommunicator
 - newcomm is an intercommunicator

G.E.Fagg UTK-CSS94 Spring2001 46

MPI-2: Establishing communications

- Client side
 - MPI_Comm_connect (port_name, info, root, comm, newcomm)
 - comm is a intracommunicator
 - newcomm is an intercommunicator
 - This call does have a timeout dependant on the implementation

G.E.Fagg UTK-CSS94 Spring2001 47

MPI-2: Establishing communications

G.E.Fagg UTK-CSS94 Spring2001 48

MPI-2: Establishing communications

MPI_Comm_connect

G.E.Fagg UTK-CSS94 Spring2001 49

MPI-2: Establishing communications

MPI_Comm_accept

G.E.Fagg UTK-CSS94 Spring2001 50

MPI-2: Establishing communications

MPI_Close_port

G.E.Fagg UTK-CSS94 Spring2001 51

MPI-2: Establishing communications

Naming service is build on a simple third party publish and lookup nameservice

3 calls exist

MPI_Publish_name (servicename, info, port_name)
 MPI_Unpublish_name (servicename, info, port_name)
 MPI_Lookup_name (servicename, info, port_name)

G.E.Fagg UTK-CSS94 Spring2001 52

MPI-2: Establishing communications

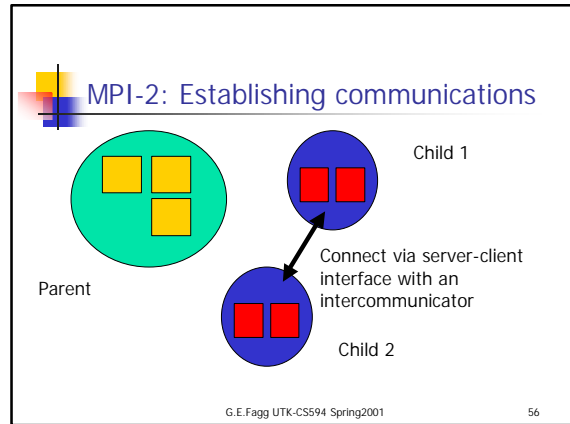
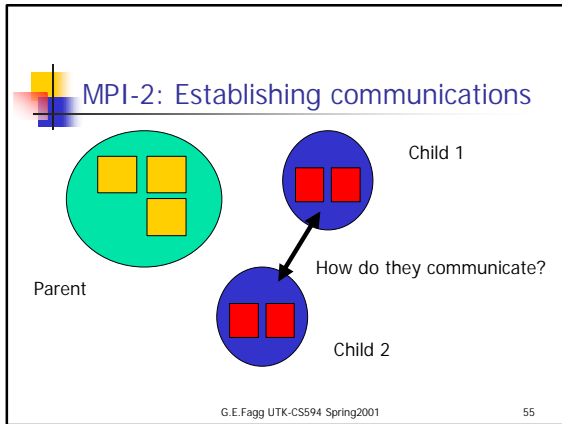
MPI_Open_port (..)
 MPI_Publish_name (service..)
 ...
 MPI_Comm_accept

G.E.Fagg UTK-CSS94 Spring2001 53

MPI-2: Establishing communications

MPI_Lookup_name (service.. port_name)
 MPI_Comm_connect (... port_name)

G.E.Fagg UTK-CSS94 Spring2001 54




- ### MPI-2: Establishing communications
- ⚡ Does this interface solve all of our problems?
 - ⚡ Can we create an intracommunicator between the two child groups?
 - ⚡ Can we now do collectives between children?
 - ⚡ What kind of operations/types of computing is it good for?
- G.E.Fagg UTK-CSS94 Spring2001 57

- ### MPI-2: Establishing communications
- ⚡ Once the communication is finished
 - ⚡ `MPI_Comm_disconnect ()`
 - ⚡ And lastly
 - ⚡ `MPI_Comm_join (fd, intercomm)`
Yes `fd` is a TCP socket descriptor used in cases when your applications can join via other means outside the MPI library
- G.E.Fagg UTK-CSS94 Spring2001 58

- ### MPI-2: Single sided communications
- ⚡ Normal message passing operation needs at least two parties
 - ⚡ A sender who performs a send call
 - ⚡ A receiver who performs a receive call
- Why is this? And what does it have to do with... memory management / protection?
- G.E.Fagg UTK-CSS94 Spring2001 59


- ### MPI-2: Single sided communications
- ⚡ Earlier Cray MPP systems allowed processes to remotely access other processes memory via `shmget` and `shmput` system function calls.
 - ⚡ This is known as Remote Memory Access (RMA)
- G.E.Fagg UTK-CSS94 Spring2001 60



MPI-2: Single sided communications

- Remote Memory Access (RMA)
 - Is fast
 - Can allow for simple program design
 - An operation specifies all the send and receive arguments together


G.E.Fagg UTK-CSS94 Spring2001 61



MPI-2: Single sided communications

- Data (memory) in a fixed range (a window) is made available with a `MPI_Win_create ()` call.
 - Freed with `MPI_Win_free ()`
- Data can then be accessed via
 - `MPI_Put ()`
 - `MPI_Get ()`
 - `MPI_Accumulate ()`


G.E.Fagg UTK-CSS94 Spring2001 62



MPI-2: Single sided communications

- The communication calls (put/get/accumulate) are non-blocking
- The operation occurs sometime after the call BUT before a synchronization point


G.E.Fagg UTK-CSS94 Spring2001 63



MPI-2: Single sided communications

- RMA communication is in two classes
 - Active
 - Memory is moved from one process to another
 - One process calls the move
 - Both must call the synchronization (including the owner of the target memory)
 - Like message passing


G.E.Fagg UTK-CSS94 Spring2001 64



MPI-2: Single sided communications

- RMA communication is in two classes
 - Passive
 - Memory is copied from a target to two other processes
 - Both processes call the copy
 - Both must synchronize (complete) their move, expect the target does not need to synchronize
 - Like shared memory

G.E.Fagg UTK-CSS94 Spring2001 65



MPI-2: Single sided communications

- Semantics of an active target operation cycle are:
 - Sync on a win (RMA target window)
 - **Start of an epoch**
 - Perform zero or more transfer operations
 - Such as `put/get/accumulate`
 - **End of an epoch**
 - Sync on a win
- Passive target operations require no sync operations

G.E.Fagg UTK-CSS94 Spring2001 66

MPI-2: Single sided communications

- MPI_Win_fence ()
 - A collective that can be used across the whole application to control operations so that they occur at the correct time such as exchanging data at the end of a time step

G.E.Fagg UTK-CSS94 Spring2001 67

MPI-2: Single sided communications

The diagram illustrates two processes. The top process performs MPI_Win_fence () followed by MPI_Put (). The bottom process performs MPI_Win_fence () followed by MPI_Put (). A right-pointing arrow connects the MPI_Put () of the top process to the MPI_Win_fence () of the bottom process. A left-pointing arrow connects the MPI_Put () of the bottom process to the MPI_Win_fence () of the top process.

G.E.Fagg UTK-CSS94 Spring2001 68

MPI-2: Single sided communications

The diagram illustrates two processes. The top process performs MPI_Win_fence () followed by MPI_Put (). The bottom process performs MPI_Win_fence () followed by MPI_Put (). Red lines connect the MPI_Win_fence () of the top process to the MPI_Win_fence () of the bottom process, and the MPI_Put () of the top process to the MPI_Put () of the bottom process. Black arrows also connect the MPI_Put () of the top process to the MPI_Win_fence () of the bottom process, and the MPI_Put () of the bottom process to the MPI_Win_fence () of the top process.

G.E.Fagg UTK-CSS94 Spring2001 69

MPI-2: Single sided communications

- MPI_Win_fence () is a collective that allows groups to sync their memory operations.
- Think of a set of MPI_Isends/MPI_Irecvs followed by a MPI_Waitall and a MPI_Barrier

G.E.Fagg UTK-CSS94 Spring2001 70

MPI-2: Single sided communications

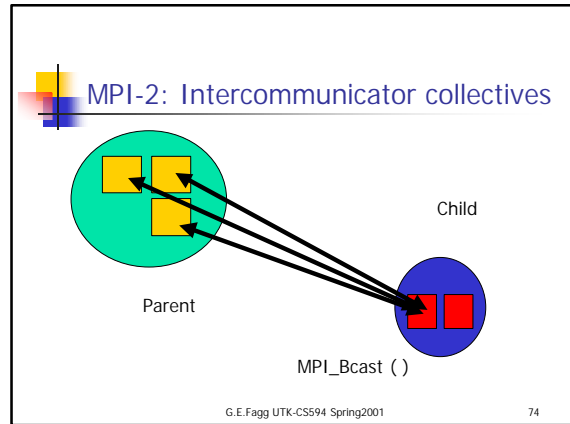
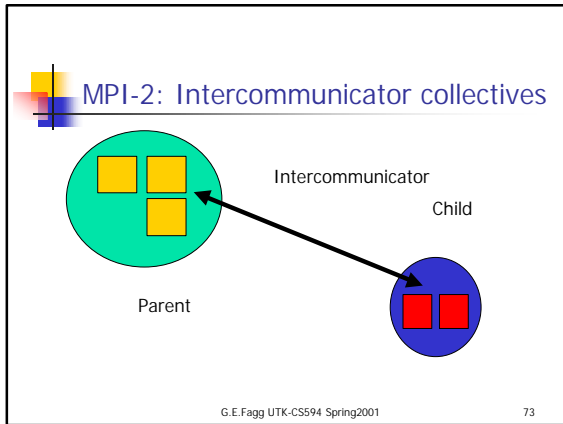
- For fine grain control between pairs of processes only there is a more complex interface
 - MPI_Win_start, MPI_Win_complete, MPI_Win_post, MPI_Win_wait, MPI_Win_lock, MPI_Win_unlock
- Close to some of the Posix threads interface

G.E.Fagg UTK-CSS94 Spring2001 71

MPI-2: Intercommunicator collectives

- Collective operations require an intracommunicator
 - Sometimes it would be useful to perform a collective (such as a broadcast) across an intracommunicator
 - As in the case of when we cannot create an intracommunicator

G.E.Fagg UTK-CSS94 Spring2001 72



- ### MPI-2: Intercommunicator collectives
- ⌚ Semantics of these (MPIX*) calls
 - ⌚ We need to know which is the root group and which is the leaf (target) group
 - ⌚ How do we know the root group?
 - ⌚ All processes in that group other than the real root call the operation with MPI_PROC_NULL as their root
 - ⌚ The root calls the operation with the root set to a special constant MPI_ROOT
 - ⌚ MPIX was a project at MSU that first proposed these extensions to MPI (hence MPIX)
- G.E.Fagg UTK-CSS94 Spring2001 75

- ### MPI-2: MPI I/O
- ⌚ Will be covered in the next lecture
 - ⌚ When you get to compare it against distributed file systems
- G.E.Fagg UTK-CSS94 Spring2001 76

- ### MPI-1
- ⌚ Back to MPI-1 stuff
 - ⌚ Some things you may have missed before
 - ⌚ Many of them to do with using MPI under the covers
 - ⌚ (like reading a book under the covers with the lights turned off)
- G.E.Fagg UTK-CSS94 Spring2001 77

- ### MPI-1: Profiling
- ⌚ Tool writers like to tell you what is happening when running an MPI application so you can improve things
 - ⌚ Three ways of doing this
 - ⌚ Instrument the MPI library
 - ⌚ Common way of doing it
 - ⌚ Instrument your application before it links to the MPI library
 - ⌚ Useful for very complex debuggers
 - ⌚ Catch the runtime calls
 - ⌚ Pardym (do it without the source available)
- G.E.Fagg UTK-CSS94 Spring2001 78

MPI-1: Profiling

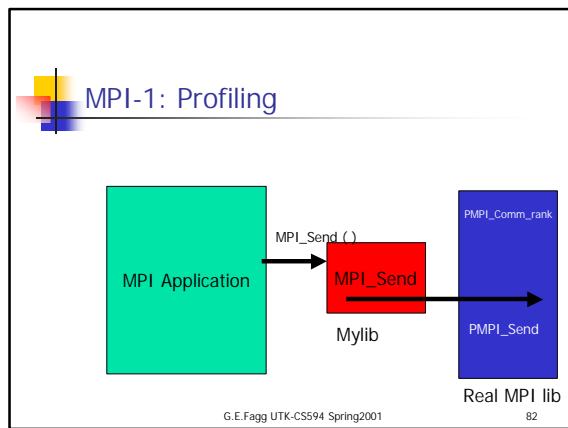
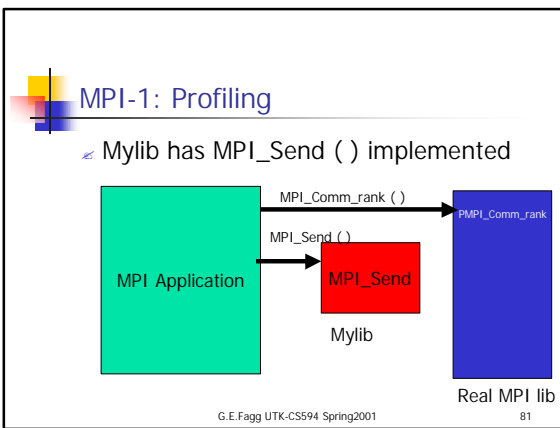
- MPI-1 provides a built in method of doing this
 - Known as the profiling interface
 - Allows instrumentation or replacing of only the required calls at run time
 - Relies on weak linking in most implementations

G.E.Fagg UTK-CSS94 Spring2001 79

MPI-1: Profiling

- How it works
 - All MPI calls have a non-profiled version called:
 - PMPI_<CALL>
 - The standard MPI_<CALL> links to this PMPI_<CALL> unless a replacement MPI_<CALL> is provided.

G.E.Fagg UTK-CSS94 Spring2001 80



MPI-1: Profiling

- The profile library does have full access to the real MPI library, by calling the PMPI_<CALL> calls.
- The tool writer only needs to implement the calls they require
 - Rather than 248+

G.E.Fagg UTK-CSS94 Spring2001 83

MPI-1: Profiling

- An application can pass information to the profile library
 - MPI_Pcontrol (level)
 - Indicated to the profile library what level of debugging you might want
 - Can also use the MPI key attributes as well.

G.E.Fagg UTK-CSS94 Spring2001 84

MPI-1: Profiling

- MPICH comes with multiple profile libraries
 - We are interested in the mpilog library
- Mpicc myfile.c -o myfile -mpilog -Impe ...

G.E.Fagg UTK-CSS94 Spring2001 85

MPI-1: Profiling

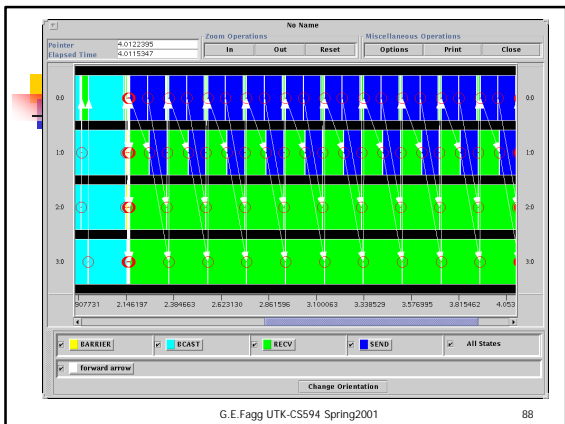
- When executed will produce a
 - <exename>.clog file
- This needs to be converted to an slog (scalable log file) before viewing etc
 - /usr/local/mpich-1.2.1/clog2slog myfile.clog

G.E.Fagg UTK-CSS94 Spring2001 86

MPI-1: Profiling

- Then run the logview on it
 - /usr/local/mpich-1.2.1/bin/logviewer myfile.slog
 - Choose the <MPI Process> button and hit <display>

G.E.Fagg UTK-CSS94 Spring2001 87



MPI-1: Profiling

- That was a barrier followed by a set of broadcasts so that they could be benchmarked.
- You will need to show that you can use the logfile viewer for your homework exercises

G.E.Fagg UTK-CSS94 Spring2001 89

MPI-1: How do collectives work

- If we look back at the logfile there is no bcast marker
 - Or reduce, scatter, etc
 - It shows collectives as being built on point to point operations
- When is this less efficient ?

G.E.Fagg UTK-CSS94 Spring2001 90

MPI-1: How do collectives work

- Collectives can be built using a number of different topologies, such as rings, trees, meshes etc
- Some of these topologies are better than others in terms of latency or obtaining peak bi-sectional bandwidth

G.E.Fagg UTK-CSS94 Spring2001 91

MPI-1: How do collectives work

Flat tree (or sequential broadcast)

G.E.Fagg UTK-CSS94 Spring2001 92

MPI-1: How do collectives work

Binary tree (fan out of 2)

G.E.Fagg UTK-CSS94 Spring2001 93

MPI-1: How do collectives work

Binomial tree

G.E.Fagg UTK-CSS94 Spring2001 94

MPI-1: How do collectives work

Pipeline / ring

G.E.Fagg UTK-CSS94 Spring2001 95

MPI-1: How do collectives work

- What are the properties of these topologies?
 - Number of steps required to complete an operations (maximum latency)
 - How many concurrent operations?

G.E.Fagg UTK-CSS94 Spring2001 96

MPI-1: How do collectives work

- Say we send a message of length X in one step down a pipeline...

G.E.Fagg UTK-CSS94 Spring2001 97

MPI-1: How do collectives work

- Step 1

G.E.Fagg UTK-CSS94 Spring2001 98

MPI-1: How do collectives work

- Step 2

G.E.Fagg UTK-CSS94 Spring2001 99

MPI-1: How do collectives work

- Step 3

G.E.Fagg UTK-CSS94 Spring2001 100

MPI-1: How do collectives work

- Step 4

G.E.Fagg UTK-CSS94 Spring2001 101

MPI-1: How do collectives work

- Can we do better ?
- Well we could segment the message into smaller parts...

 - End to end latency (to get the first part), would it be different?
 - Overall end to end bandwidth would it change?

G.E.Fagg UTK-CSS94 Spring2001 102

MPI-1: How do collectives work

Step 1

G.E.Fagg UTK-CSS94 Spring2001 103

MPI-1: How do collectives work

Step 2

G.E.Fagg UTK-CSS94 Spring2001 104

MPI-1: How do collectives work

Step 3

G.E.Fagg UTK-CSS94 Spring2001 105

MPI-1: How do collectives work

- What happens with a tree? Is it even better.
- Comm systems have peak performances and small message sizes are usually less efficient

G.E.Fagg UTK-CSS94 Spring2001 106

MPI-1: Communicators for distributed computing

If an application is spread across multiple systems but acts as a single MPI application, performance problems can occur.

G.E.Fagg UTK-CSS94 Spring2001 107

MPI-1: Communicators for distributed computing

Say the collective is a ring

G.E.Fagg UTK-CSS94 Spring2001 108

MPI-1: Communicators for distributed computing

But the nodes are distributed...

G.E.Fagg UTK-CS594 Spring2001 109

MPI-1: Communicators for distributed computing

But the nodes are distributed...

G.E.Fagg UTK-CS594 Spring2001 110

MPI-1: Communicators for distributed computing

What you really want...

G.E.Fagg UTK-CS594 Spring2001 111

MPI-1: Communicators for distributed computing

What you really want...

G.E.Fagg UTK-CS594 Spring2001 112

MPI-1: Communicators for distributed computing

You can create multiple new communicators

- One for each of sites
- A new one that is made by merging the new communicators in the right order
 - Works well if you know how your MPI implementation works
 - Or you have to build your own collectives

G.E.Fagg UTK-CS594 Spring2001 113

MPI-1: Communicators for distributed computing

Gets more complex as we go through more layers

- Bottom layer, SMP machines and dedicated MPPs
- Clusters of these machines
- Campus sets of clusters
- Across nation campus computing
- Transatlantic clusters
- How would you perform a broadcast? Or an All2All ?
 - Does MAGPIE by Thilo Keilmann do it the best way?

G.E.Fagg UTK-CS594 Spring2001 114

MPI-1: Profiling

- Exercise 1:
 - Write a profile library that counts every MPI_Bcast function call and displays the count as the program runs
 - Write a test program, that performs 'n' broadcasts, where the root changes from rank 0-n-1 for each broadcast
 - Only the ROOT of each Bcast should print at each point

Keep both codes as you will build on them later.

G.E.Fagg UTK-CSS94 Spring2001 115

MPI-1: Profiling

- Exercise 2:
 - Profile a standard MPICH broadcast for 8 processors
 - Collect an image of the broadcast from the log viewer
 - What topology does it use?

G.E.Fagg UTK-CSS94 Spring2001 116

MPI-1: Communicators for distributed computing

- Exercise 3:
 - Take a copy of your profile library and make a new broadcast function that performs a binary tree
 - I.e. when your test program from exercise one is linked with this it will use your broadcast not the default MPI function
 - Get a screen shot from the logviewer

G.E.Fagg UTK-CSS94 Spring2001 117

MPI-1: Communicators for distributed computing

- Exercise 4:
 - Write a test application with 8 nodes
 - Make them do a pipeline (I.e. 0 sends to 1.. 7 to 0)
 - Then assume that all even ranks are on one system and that all odd are on another (as the distributed example a few slides ago)
 - Using communicator manipulation routines, create two new communicators for each odd/even set, then merge them into an ordered full set
 - (I.e. 0,1,2..n-1 is made from original ranks, 0,2,4..5,7)

G.E.Fagg UTK-CSS94 Spring2001 118

MPI-1: Communicators for distributed computing

- Exercise 5:
 - This is a mix of the previous few exercise
 - We have a single communicator, where odd/even nodes are on the 'different' systems.
 - Write a final profile broadcast library that breaks them into two communicators
 - Then the first root send to the second root
 - Each set then does a 'local' broadcast
 - Get a screen shot again

G.E.Fagg UTK-CSS94 Spring2001 119

MPI-1: Communicators for distributed computing

G.E.Fagg UTK-CSS94 Spring2001 120

