3️⃣

MPI 2

Files	PP_MPI2.pdf
Link	
Status	Done
Type	Lecture

Types of Communication

Blocking vs Non-blocking

> Refers to the state of the sender

Blocking operations: the program "stops" until the operation is "complete" (resources are free to use by another operation

Non-blocking operation: the program initiates what you want, but does not wait until its complete before moving to the next line of code to continue running the program, access to the resources this operation use might be prohibited (maybe even for read-only), gives you an `MPI_Request` object that can be used to check status or wait for completion

`int MPI_Wait(MPI_Request *req, MPI_Status *status)` Blocking, waits until the operation is complete

Synchronous vs Asynchronous

> Refers to the state of receiver

Synchronous: send will start only when the destination is ready to receive, will only complete after the receiver acknowledges that everything is received and is in memory

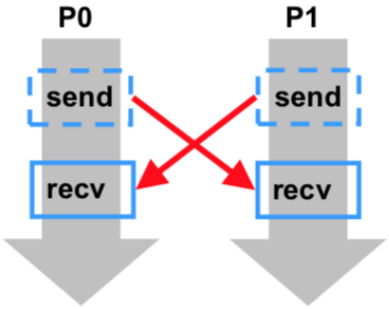
Asynchronous: will send into a buffer is receiver is not ready, only know that message is left

Summary

There are not synchronous and non-asynchronous receive options because the program does not care if it is receiving data from a buffer or another program directly, it just accepts it.

semantics / mode	standard	synchronous	Asynchronous (buffered)
blocking	MPI_Send MPI_Recv	MPI_Ssend	MPI_Bsend
non-blocking	MPI_Isend MPI_Irecv	MPI_Issend	MPI_Ibsend

Deadlocks in Synchronous send operations



Derived Data Types

Functions

Definition	Description	Diagram
int MPI_Type_commit(MPI_Datatype *datatype)	Commit	
int MPI_Type_free(MPI_Datatype *datatype)	Deallocate	
int MPI_Type_contiguous(int count, MPI_Datatype oldtype,MPI_Datatype *newtype)	Groups a series of fixed number of smaller data types	
int MPI_Type_vector(int count, int blocklength, int stride, MPI_Datatype oldtype, MPI_Datatype *newtype)	Extract a fixed number of data blocks at fixed intervals (for example: part of an array)	

Can also group several derived data types, for example use a contiguous within a vector.

Advanced Communicators & Groups

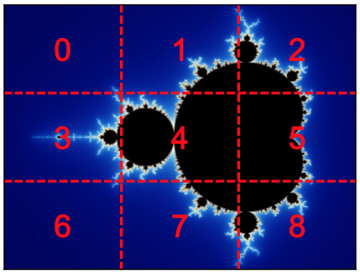
Functions

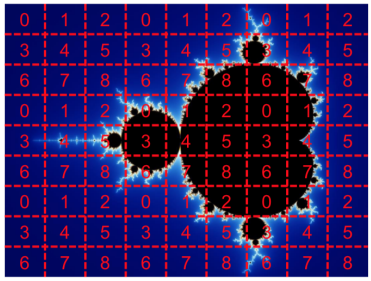
Definition	Description	Diagram
int MPI_Comm_group(MPI_Comm comm, MPI_Group *group)	Extract a group from a communicator	

Definition	Description	Diagram
int MPI_Group_incl(MPI_Group group, int n, int *ranks, MPI_Group *newgroup)	Include a process(es) in a group	
int MPI_Group_excl(MPI_Group group, int n, int *ranks, MPI_Group *newgroup)	Exclude a process(es) in a group	
int MPI_Comm_create(MPI_Comm oldcomm, MPI_Group newgroup, MPI_Comm *newcomm)	Create a communicator from a group	
int MPI_Comm_split(MPI_Comm comm, int color, int key, MPI_Comm *newcomm)	Split a communicator, processes that have the same color end up in the same new communicator, within a communicator rank is reassigned based on the key argument	
int MPI_Comm_free(MPI_Comm *comm)	Marks a communicator for deallocation but it is only deallocated when all operations within it are complete	

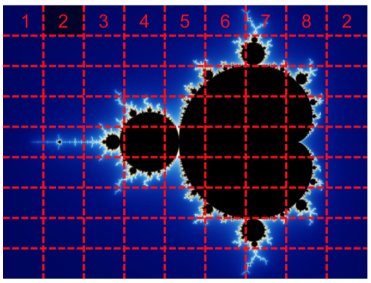
Load Balancing

Approach 1: Smaller sized chunks, but keep in mind, granularity vs high communication cost balance (lecture 1)





Approach 2 (better): Master-Worker-Pattern, once worker is done with chunk asks master for the next piece to work on


1
2
3
4
5
6
7
8
2