

MPI Quick Reference

MPI, the **Message Passing Interface**, is a standardized message-passing system for writing portable parallel programs.
Implementations: LAM/MPI (the one we use), MPICH, ...

start mpi with lamboot <hostfile>
lamhalt, lamnodes, lamclean
run programs: mpirun <mpirun arguments> <program> <program arguments>
mpirun arguments: C or -np 4

process ID: **rank** (0, 1, 2, ...)

MPI_Init() MPI_Finalize()
MPI_Comm_rank() MPI_Comm_size()

message envelop: source, destination, tag (message type) and communicator (MPI_Comm_world = all processes)

Point-to-point communication: MPI_Send & MPI_Recv
- messages are non-overtaking, but fairness is not guaranteed
- types must match!
- MPI_Recv: - count is upperbound
 - use MPI_ANY_TAG, MPI_ANY_SOURCE
- MPI_Probe, MPI_IProbe (non-blocking): polling for messages
- datatypes: MPI_INT, MPI_CHAR, MPI_FLOAT, MPI_DOUBLE, ...

Varia
- MPI_Status: record with fields
MPI_SOURCE, MPI_TAG and MPI_ERROR
get count with MPI_Get_count
- start processes MPI_Comm_spawn
- errors codes: different than MPI_SUCCESS
get message with MPI_Error_string

Communication optimization:

- **standard**: send or buffered (when necessary) & blocked
- **non-blocking**: post > comm. in background > test-for-completion
 - send buffer may not be accessed!
- **modes**
 - **buffered (B)**: message is copied (MPI_Buffer_attach to specify buffer)
 - **synchronous (S)**: rendez-vous of sender & receiver
 - **ready-mode (R)**: receiver is ready => sender can send immediately
- MPI_Cancel: cancellation of non-blocking communications

posting: MPI_Isend and MPI_Irecv()
test: MPI_Wait(), MPI_Test(), MPI_Request_free() ...

<i>blocking</i>	<i>non-blocking</i>
MPI_Bsend	MPI_Ibsend
MPI_Ssend	MPI_Issend
MPI_Rsend	MPI_Irsend

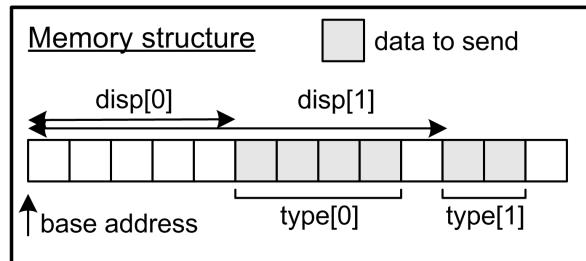
MPI_Send_recv & MPI_Send_recv_replace (same buffer) (both blocking)
eg. avoid deadlock

User-defined datatypes:

for non-contiguous memory locations

`Typemap = {(type0, displacement0), ...}`

specifies message buffer together with base address (passed with send/receive command)



Define types with (and commit with `MPI_Type_commit!`)

`MPI_Type_contiguous`: replication of a datatype

`MPI_Type_vector`: equally spaced blocks (stride)

`MPI_Type_create_hvector` stride is arbitrary nbr of bytes

`MPI_Type_indexed` different #blocks, # displacements

`MPI_Type_hindexed`

`MPI_Type_create_struct` diff datatypes (most general)

`MPI_Get_address` implementation-independent & operator

pack/unpack alternative

make a copy to a buffer: more memory + extra time

when buffer layout is data dependent

`MPI_Pack` copies into buffer, send with type `MPI_PACKED`

`MPI_Unpack` gradually unpack buffer, outcount must be actual number

Error handling

- `MPI_Comm_set_errhandler`: change error handling

`MPI_ERRORS_FATAL` is default, `MPI_ERRORS_RETURN` MPI call returns, check return value for error

- errors codes: different than `MPI_SUCCESS`, get message with `MPI_Error_string`

Communicators

> subdivide processes into **groups** (type `MPI_GROUP`)

process has (different) rank in each group it belongs to

`MPI_Group_rank`: `MPI_UNDEFINED` if no member

`MPI_Group_size`

`MPI_Comm_group`: get group of communicator

...

> **communicator** is associated with one or two groups

communication only happens within a communicator

intracommunicator: within a group

intercommunicator: P2P between 2 disjoint groups

`MPI_Comm_world` = communicator of all processes

...

Collective Communications (for optimization)

`MPI_Barrier`: returns when all processes entered the call

`MPI_Bcast`: broadcast (one-message-to-all), root also!

`MPI_Gather`: all-to-one (root included)

`MPI_Gatherv`: with stride

`MPI_Scatter`: diff-messages-to-all (`MPI_Scatterv`)

`MPI_Allgather`: all-to-all (`MPI_Allgatherv`)

`MPI_Alltoall`: personalized messages all-to-all

`MPI_Alltoallv/w`

`MPI_Reduce`: global reduce operation all-to-one

`MPI_MAX`, `MPI_MIN`, `MPI_SUM`, `MPI_PROD`, ...

`MPI_Allreduce`: result to all

`MPI_Reduce_scatter`: scatter the result (array)

`MPI_Scan`

`MPI_Op_create`: user-defined operation