MPI Cheat Sheet

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI 1818

FACULTY OF AUTOMATIC CONTROL AND COMPUTERS

Made by Cristian Chilipirea

Arguments from the main function
Called at the start of any MPI program

int MPI_Init(↕ int *, ↕ char *)**

&argc	&argv
NULL	NULL

Called at the end of any MPI program

int MPI_Finalize()

Gives the number of tasks

int MPI_Comm_size (↓ MPI_Comm, ↑ int *)

MPI_COMM_WORLD

&num_tasks

Gives the id (rank) of the current (calling) task

int MPI_Comm_rank (↓ MPI_Comm, ↑ int *)

MPI_COMM_WORLD

&rank

Synchronizez all tasks at the call of the barrier

int MPI_Barrier (↓ MPI_Comm comm)

MPI_COMM_WORLD

Send from buffer **b**, **c** elements of data type d to rank r. The communication is marked with tag **t**.
The function is blocking, b can safely be used after it but data may not have yet been delivered.

int MPI_Send(↓ void *b, ↓ int c, ↓ MPI_Datatype d, ↓ int reiceiver, ↓ int t, ↓ MPI_Comm)

v	sizeof(v)	MPI_INT	[0, **num_tasks**)	[0, ..)	MPI_COMM_WORLD
&v[3]	[0,..)	MPI_CHAR			
&a		MPI_FLOAT			
v+5		MPI_LONG			

Receive in buffer **b**, **c** elements of data type d from rank r. The communication is marked with tag **t**.
The function is bloking, b can be safely used and the data was delivered.

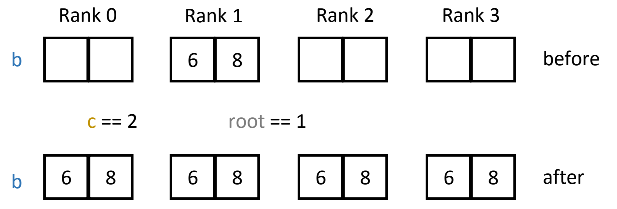
int MPI_Recv(↑ void *b, ↓ int c, ↓ MPI_Datatype d, ↓ int sender, ↓ int t, ↓ MPI_Comm, ↑ MPI_Status *)

v	sizeof(v)	MPI_INT	[0, **num_tasks**)	[0, ..)	MPI_COMM_WORLD	&Stat
&v[3]	[0,..)	MPI_CHAR	MPI_ANY_SOURCE	MPI_ANY_TAG		MPI_STATUS_IGNORE
&a		MPI_FLOAT				***Stat.MPI_SOURCE, Stat.MPI_TAG***
v+5		MPI_LONG				

Sends (Broadcasts) **c** elements of data type d from buffer **b** from rank r to all other tasks in buffer **b**.
All tasks have to call this function with the same value for root.

int MPI_Bcast (↕ void *b, ↓ int c, ↓ MPI_Datatype d, ↓ int root, ↓ MPI_Comm)

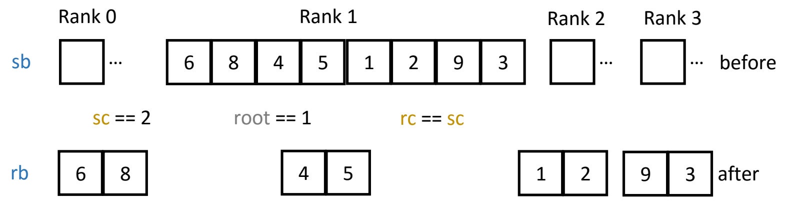
v	sizeof(v)	MPI_INT	[0, **num_tasks**)	MPI_COMM_WORLD
&v[3]	[0,..)	MPI_CHAR		
&a		MPI_FLOAT		
v+5		MPI_LONG		



Splits the elements from **sb** of datatype sd on rank root in num_tasks chunks of size **sc**.
Every task receives its appropriate chunk in **rb**. For simplicity **sc** == **rc**, sd == rd.
All tasks have to call this function with the same value for root.

int MPI_Scatter (↓ void *sb, ↓ int sc, ↓ MPI_Datatype sd, ↑ void *rb, ↓ int rc, ↓ MPI_Datatype rd, ↓ int root, ↓ MPI_Comm)

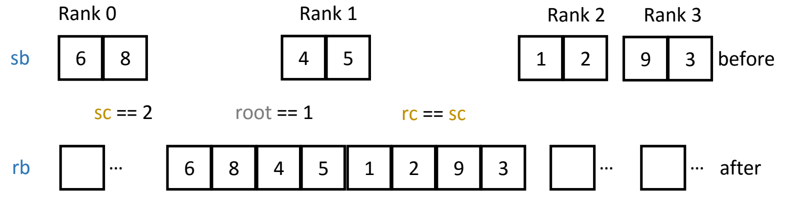
v	sizeof(v)/**num_tasks**	MPI_INT	v	sizeof(v)/**num_tasks**	MPI_INT	[0, **num_tasks**)	MPI_COMM_WORLD
&v[3]	[0,..)	MPI_CHAR	&v[3]	[0,..)	MPI_CHAR		
&a		MPI_FLOAT	&a		MPI_FLOAT		
v+5		MPI_LONG	v+5		MPI_LONG		



Gathers **sc** elements from all **sb** of datatype sd on all tasks and places the **num_tasks** chunks of size **rc** in **rb** on task of rank root.
Every task sends its appropriate chunk in **rb**. For simplicity **sc** == **rc**, sd == rd.
All tasks have to call this function with the same value for root.

int MPI_Gather (↓ void *sb, ↓ int sc, ↓ MPI_Datatype sd, ↑ void *rb, ↓ int rc, ↓ MPI_Datatype rd, ↓ int root, ↓ MPI_Comm)

v	sizeof(v)/**num_tasks**	MPI_INT	v	sizeof(v)/**num_tasks**	MPI_INT	[0, **num_tasks**)	MPI_COMM_WORLD
&v[3]	[0,..)	MPI_CHAR	&v[3]	[0,..)	MPI_CHAR		
&a		MPI_FLOAT	&a		MPI_FLOAT		
v+5		MPI_LONG	v+5		MPI_LONG		


Rank 0
Rank 1
Rank 2
Rank 3
sb
6 8
4 5
1 2
9 3
before
sc == 2
root == 1
rc == sc
rb
…
6 8 4 5 1 2 9 3
…
…
after