TACC
TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

MPI Lab

IHPCSS

Parallel Programming: Classic Track

July 7-12, 2019

PRESENTED BY:

John Cazes

cazes@tacc.utexas.edu

Lecture and Lab slides available at: tinyurl.com/tacc-2019-ihpcss

Lab Exercises

Login to bridges.psc.edu

Untar the lab source code

```
% tar -xvf ~jcazes/ihpcss_2019_mpi.tar
% cd ihpcss_2019_mpi
```

Part 4: Using derived datatypes

Part 5: Creating a cartesian communicator

Part 6: (Optional) Running the stommel model

Running Interactively

If you would like to follow along using the examples during the lecture, you may start an interactive session on Bridges or Comet.

Bridges:

```
# Monday
interact -p RM -N 1 -n 4 -t 4:00:00 -A ac560tp -R mpi
# Tuesday
interact -p RM -N 1 -n 4 -t 4:00:00 -A ac560tp -R mpi2
```

Comet:

```
srun -p compute -N 1 --ntasks-per-node=16 -t 4:00:00 \
--wait=0 --export=all --pty /bin/bash
```

Part 4: Using derived datatypes

The MPI examples in this directory are the examples covered in the slides. There may be minor differences between the slides and these examples.

Enter the examples directory

```
cd mpi_examples
```

To build all the examples:

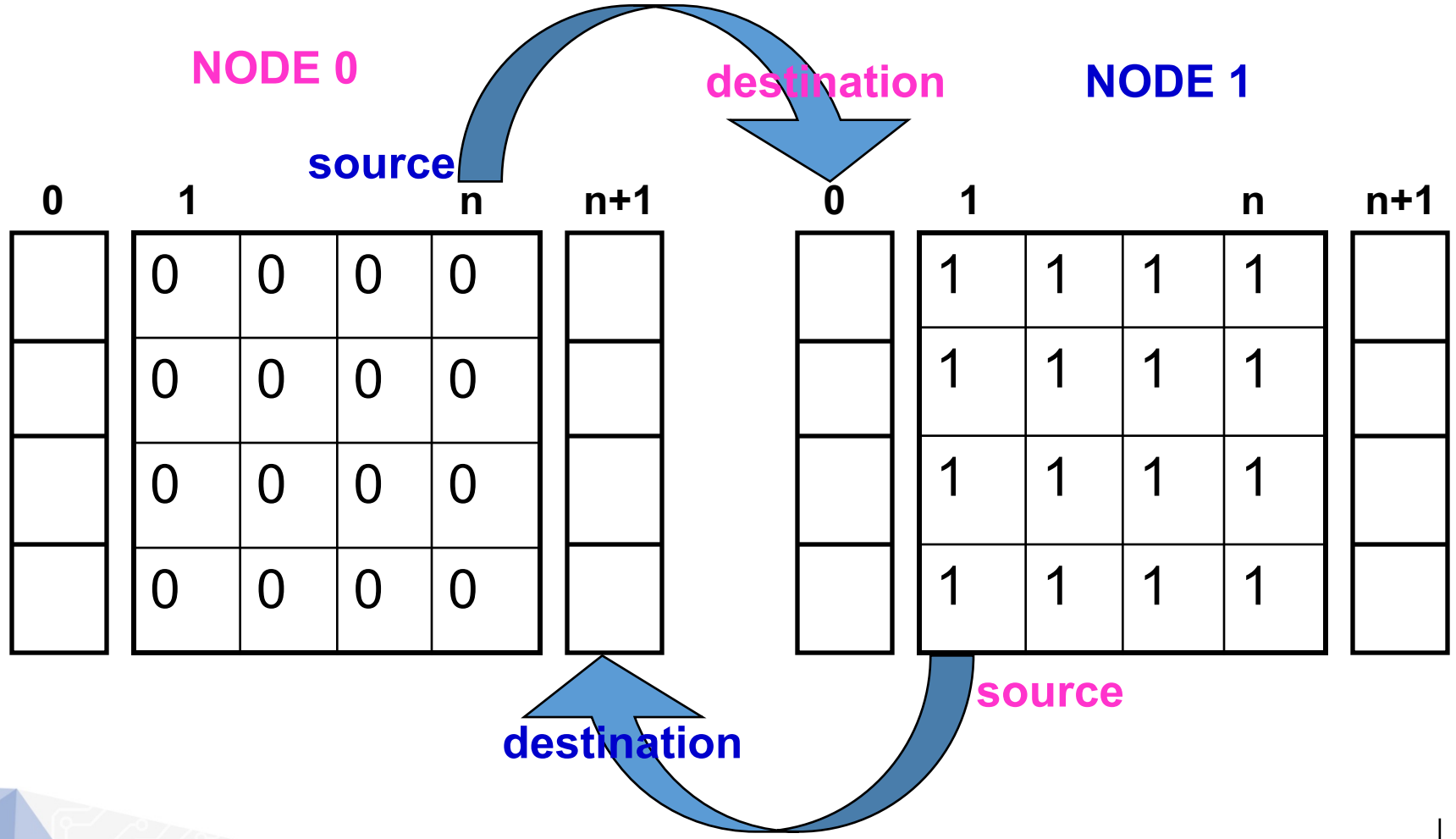
```
make
```

To run interactively

```
mpirun ./<executable>    #Bridges
ibrun ./<executable>     #Comet
```

Part 4: Using derived datatypes

Exchange ghost cell data using derived datatypes



Derived datatypes in ghost cell exchange

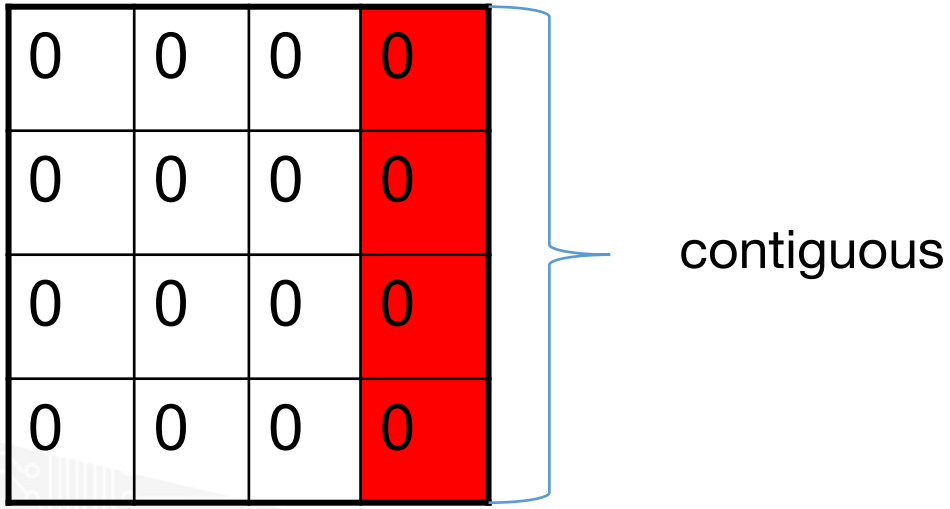
Set up the simplest derived datatype, contiguous.

C_ghost_exchange_derived_datatype.c

```
ierr=MPI_Type_contiguous(<number_elements>,<base_data_type>,<name>);
ierr=MPI_Type_commit(<name>);     //  You must commit before you can use it
```

F_ghost_exchange_derived_datatype.f90

```
call MPI_Type_contiguous(<number_elements>,<base_data_type>,<name>,ierr)
call MPI_Type_commit(<name>,ierr)  !  You must commit before you can use it
```



Derived datatype – ghost cell exchange

As before, fix the MPI_Sendrecv calls by filling in the missing data.
What should the count be for the derived datatype?

C_ghost_exchange_derived_datatype.c

```
ierr=MPI_Sendrecv(
         &A(1,  j2),      <send_count>, <send_datatype>, idest,9,
         &A(1,  jghost1),<recv_count>, <recv_datatype>, isrc ,9,
         MPI_COMM_WORLD, &status);
```

F_ghost_exchange_derived_datatype.f90

```
call MPI_Sendrecv(  &
         A(1,  j2),      <send_count>, <send_datatype>, idest,9, &
         A(1, jghost1), <recv_count>, <recv_datatype>, idest,9, &
         MPI_COMM_WORLD, MPI_STATUS_IGNORE, ierr)
```

Domain Decomposition – Ghost cell exchange

Compile:

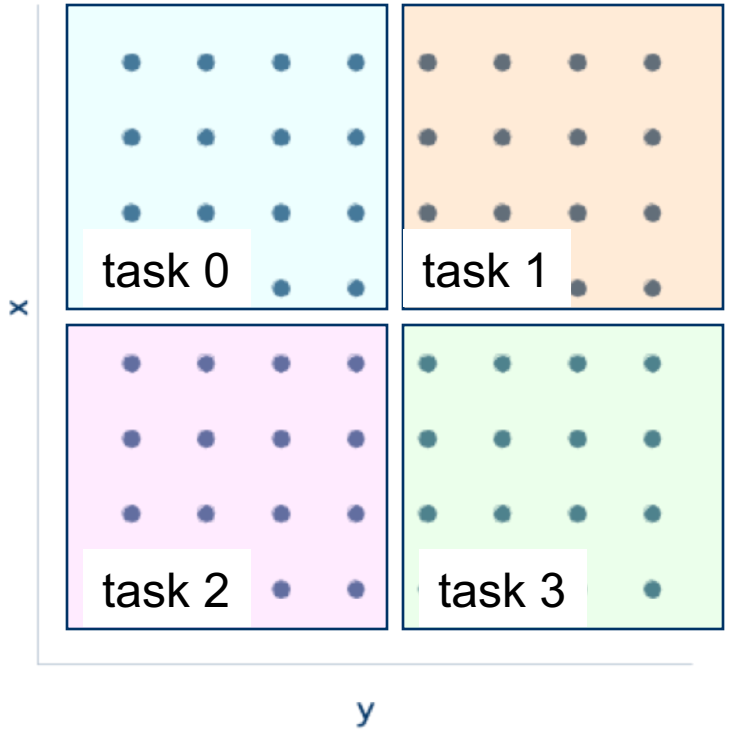
```
make C_ghost_exchange_derived_datatype  #C
make F_ghost_exchange_derived_datatype  #Fortran
```

Run:

```
mpirun ./C_ghost_exchange_derived_datatype #C
mpirun ./F_ghost_exchange_derived_datatype #Fortran
```

Output should be the same as from `ghost_exchange` from the first lab

Part 5: Creating a cartesian communicator



- Assemble tasks into 2D process grid (see *MPI_Cart_create, MPI_Cart_get*, *MPI_Cart_shift, MPI_Cart_coords* for further reference) or do this by hand

-OR-

- Get task coordinate in process grid, e.g., task 2 has coordinates (row,col)=(1,0)
- Create non-overlapping communicators along rows and columns

- Create column communicator:

```
Call MPI_COMM_SPLIT(MPI_COMM_WORLD,col,row,MPI_COL_COMM,mpi_err)
Call MPI_COMM_SIZE(MPI_COL_COMM,npes_c,mpi_err)
Call MPI_COMM_RANK(MPI_COL_COMM,mype_c,mpi_err)
top=mod(mype_c+1,npes_c); bottom=mod(mype_c-1,npes_c)
```

Part 5: Creating a cartesian communicator

Fill in the place holders to create the cartesian communicator

C_cartCreate.c

```
ierr=MPI_Cart_create(MPI_COMM_WORLD,
                     <# dims>, <# tasks each dimension>,
                     periods,  reorder, <communicator name>);
```

F_cartCreate.f90

```
call MPI_CART_CREATE(MPI_COMM_WORLD, &
                     <# dims>, <# tasks each dimension>, &
                     periods,  reorder, <communicator name>, ierr)
```

Part 5: Creating a cartesian communicator

Examine the commands to get your coordinate and the rank of the task in the shifted cartesian directions

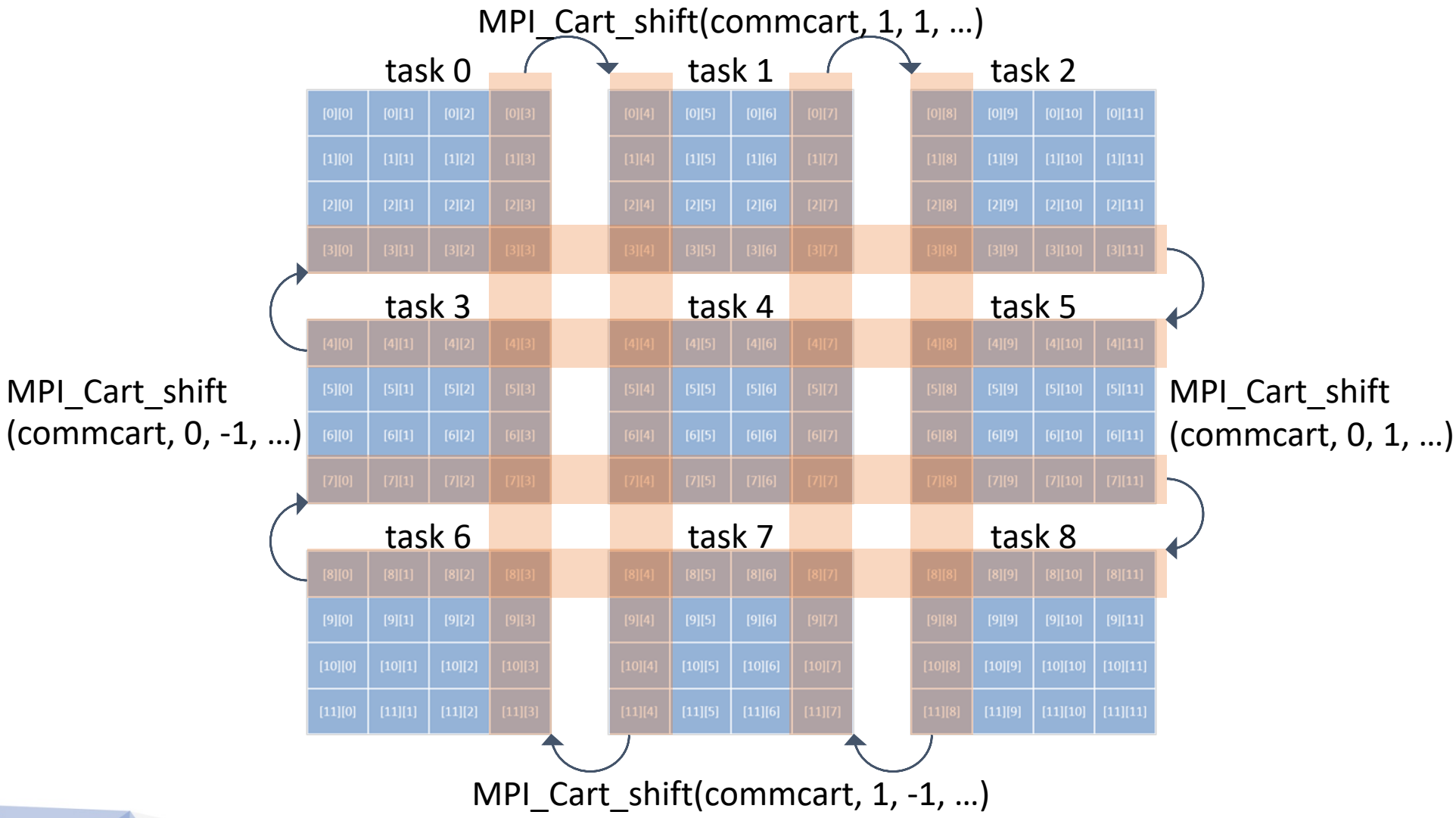
C_cartCreate.c

```
ierr=MPI_Cart_coords(comm2d,mytask,ndim,coords);
ierr=MPI_Cart_shift(comm2d,0,1,&top,&bottom);
ierr=MPI_Cart_shift(comm2d,1,1,&left,&right);
```

F_cartCreate.f90

```
call MPI_CART_COORDS(comm2d,irank,ndim,coords,ierr)
call MPI_CART_SHIFT(comm2d,0,1,top,bottom,ierr)
call MPI_CART_SHIFT(comm2d,1,1,left,right,ierr)
```

Shift Example (non-periodic)



Part 5: Creating a cartesian communicator

Compile using the command:

```
make C_cartCreate   #C
make F_cartCreate   #Fortran
```

Run:

```
mpirun C_cartCreate
mpirun F_cartCreate
```

This will print out three lines from each task reporting the task's coordinates, and the ranks of its neighbors on the cartesian grid.

Part6 – (Optional) Stommel Ocean model

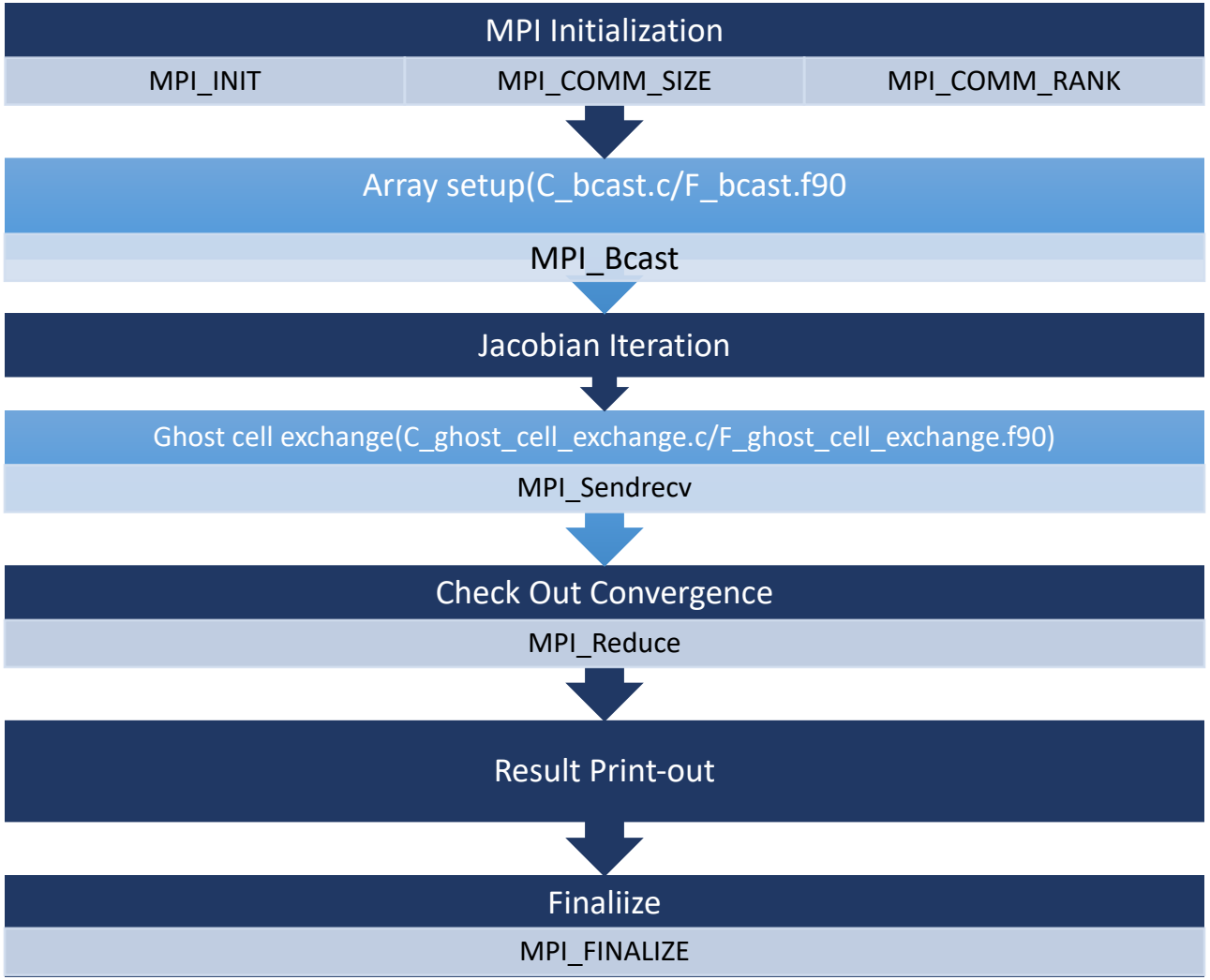
Use the updated MPI calls from the first lab

C_ghost_exchange.c C_bcast.c
F_ghost_exchange.f90 F_bcast.c

to update the same calls in the Stommel Ocean model

C_stommel.c
F_stommel.f90

Part6: (Optional) Stommel Ocean model



Part6 – (Optional) Stommel Ocean model

As before, compile using the command:

```
make C_stommel   #C
make F_stommel   #Fortran
```

Run:

```
mpirun C_stommel < stommel.in
mpirun F_stommel < stommel.in
```

The model is set to print out convergence criteria.

This a very small grid, 16x16. Try running with the large grid, `stommel.in.large`, 1600x1600, using 24 tasks instead of 4.

You've increased your computational grid size by a factor of 10^6. Did it slow down?