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

Getting Started

Login to bridges.psc.edu

Untar the lab source code

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

Part 1: Getting started with examples

Part 2: Transferring data in a 1-D decomposition

Part 3: Broadcasting data

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 1: MPI Examples

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
```

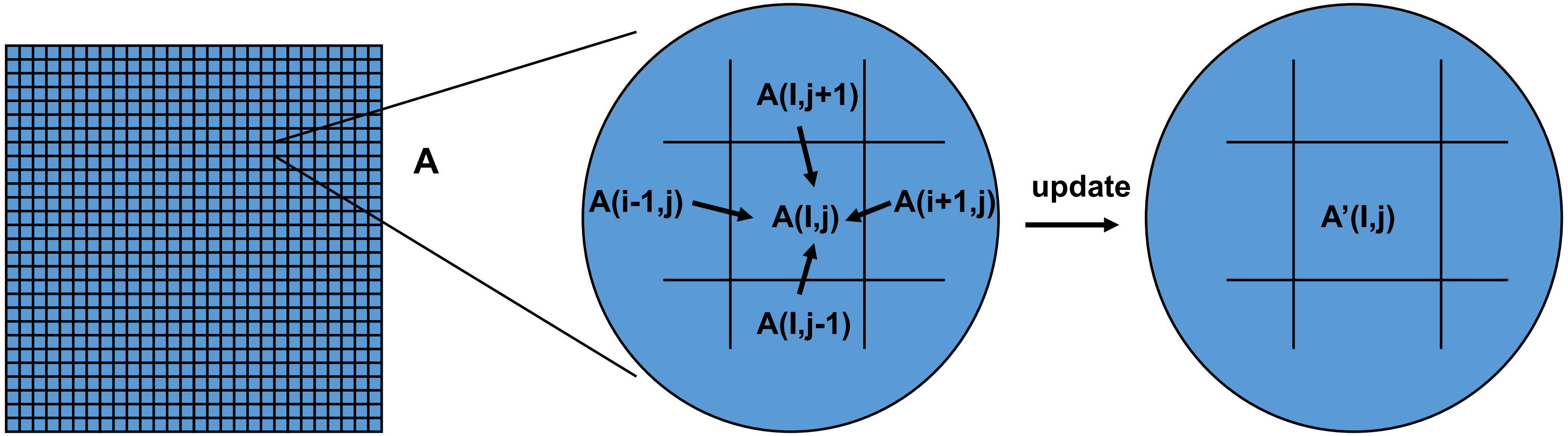
Part 1: MPI Examples

C	Fortran	Description
C_broadcast.c	F_broadcast.f90	Broadcasts from one task to all
C_even_odd.c		Uses MPI_Groups to create even/odd communicators
C_gather.c	F_gather.f90	Creates a matrix on task 0 from distributed vectors
C_gatherv.c	F_gatherv.f90	Creates a matrix in reverse order from distributed vectors
C_isend_irecv.c	F_isend_irecv.f90	Communicates between 0 and 1 using non-blocking comms
C_master_worker.c	F_probe.f90	Creates a intra-communicator for workers
C_probe.c	F_scatter_reduce.f90	Probes incoming message to determine size

Part 2: Domain Decomposition

Solve 2-D partial differential equation (finite difference) – Stommel Ocean Model

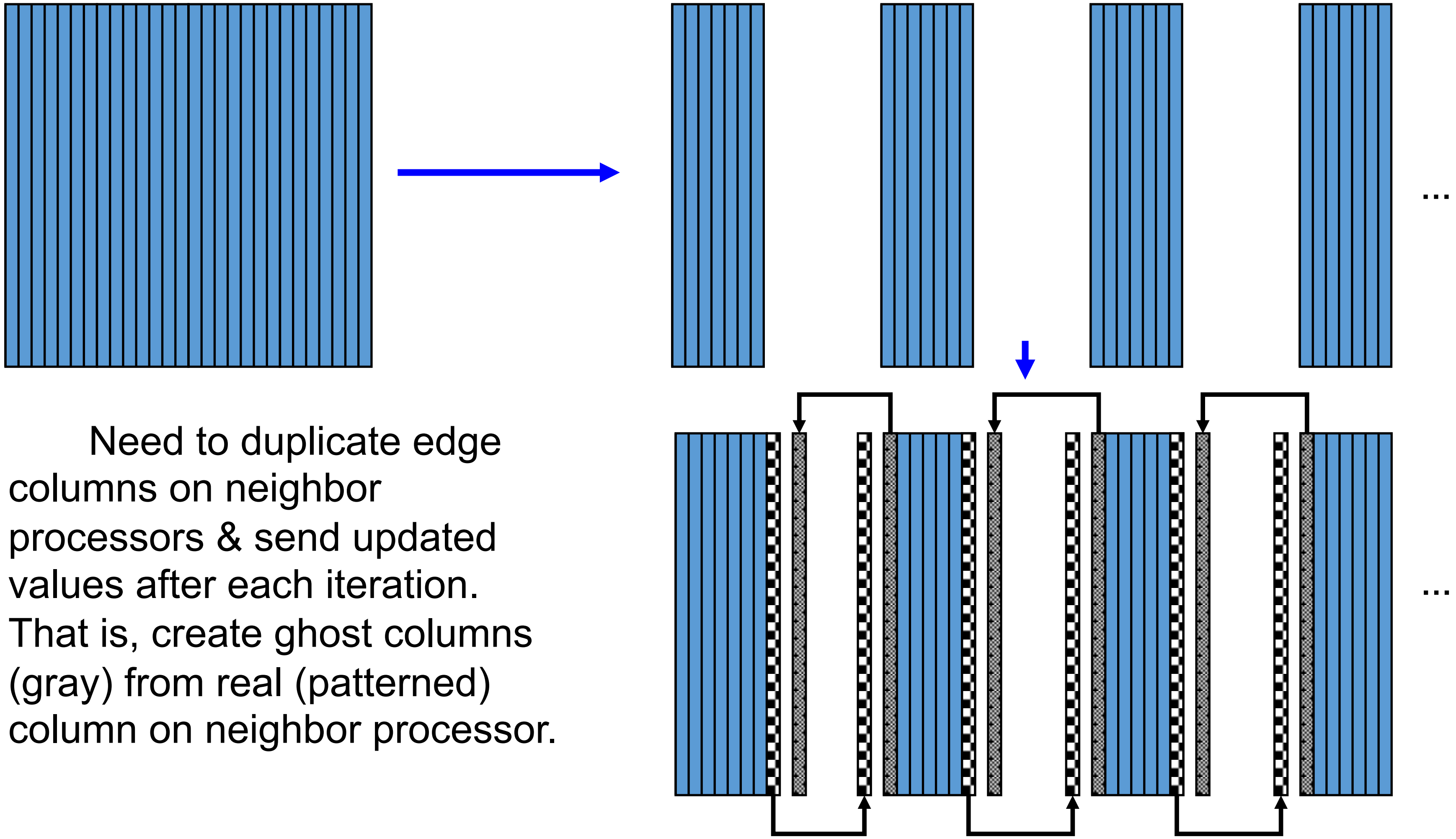
- Represent x-y domain as 2-D Cartesian grid
- Solution Matrix=A(x,y)
- Initialize grid elements with guess.
- Iteratively update Solution Matrix (A) until converged.
- Each iteration uses “neighbor” elements to update A



A’ holds updated elements of A

Domain Decomposition: Sharing Data Across Processors

Decompose 2-D grid into column blocks across *p* processors (1-D decomposition)



Need to duplicate edge columns on neighbor processors & send updated values after each iteration. That is, create ghost columns (gray) from real (patterned) column on neighbor processor.

Domain Decomposition

Matrix Layout with Ghost Cells

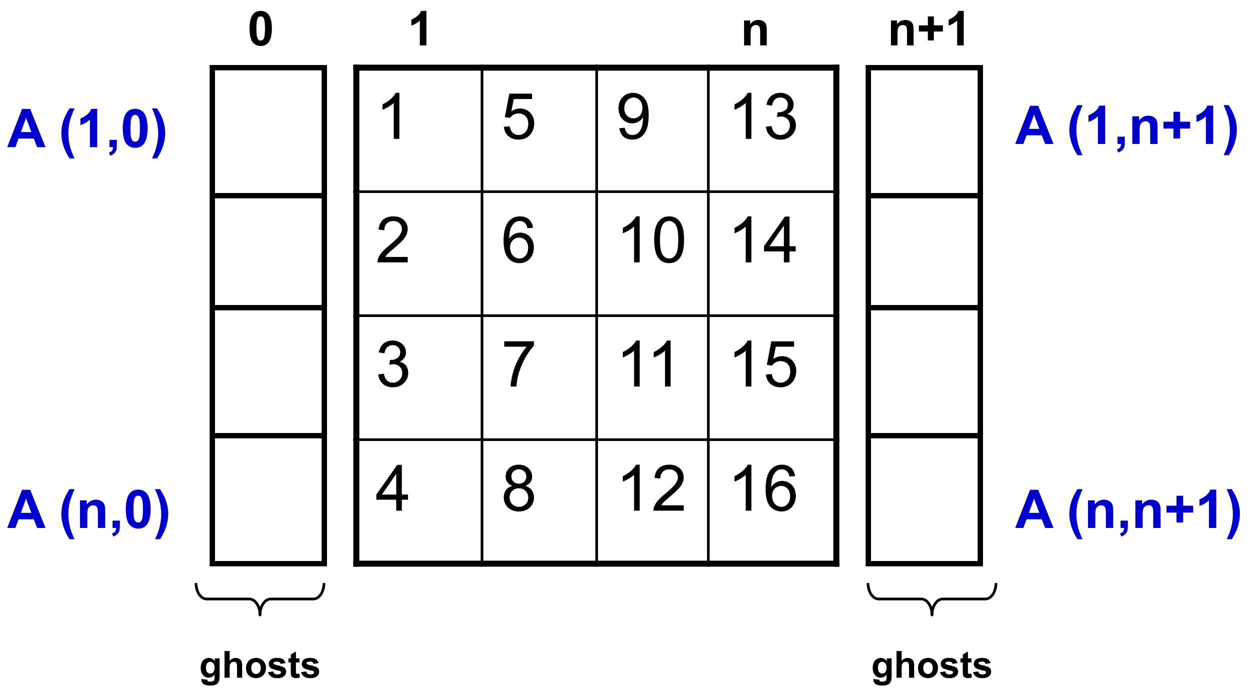
Redefine Array for easy ghost access

```
real*8 :: A(n, 0:n+1)
```

Fortran

```
#define A(i,j)  a( (i-1) + (j)*n )
double a[n*(n+2)];
```

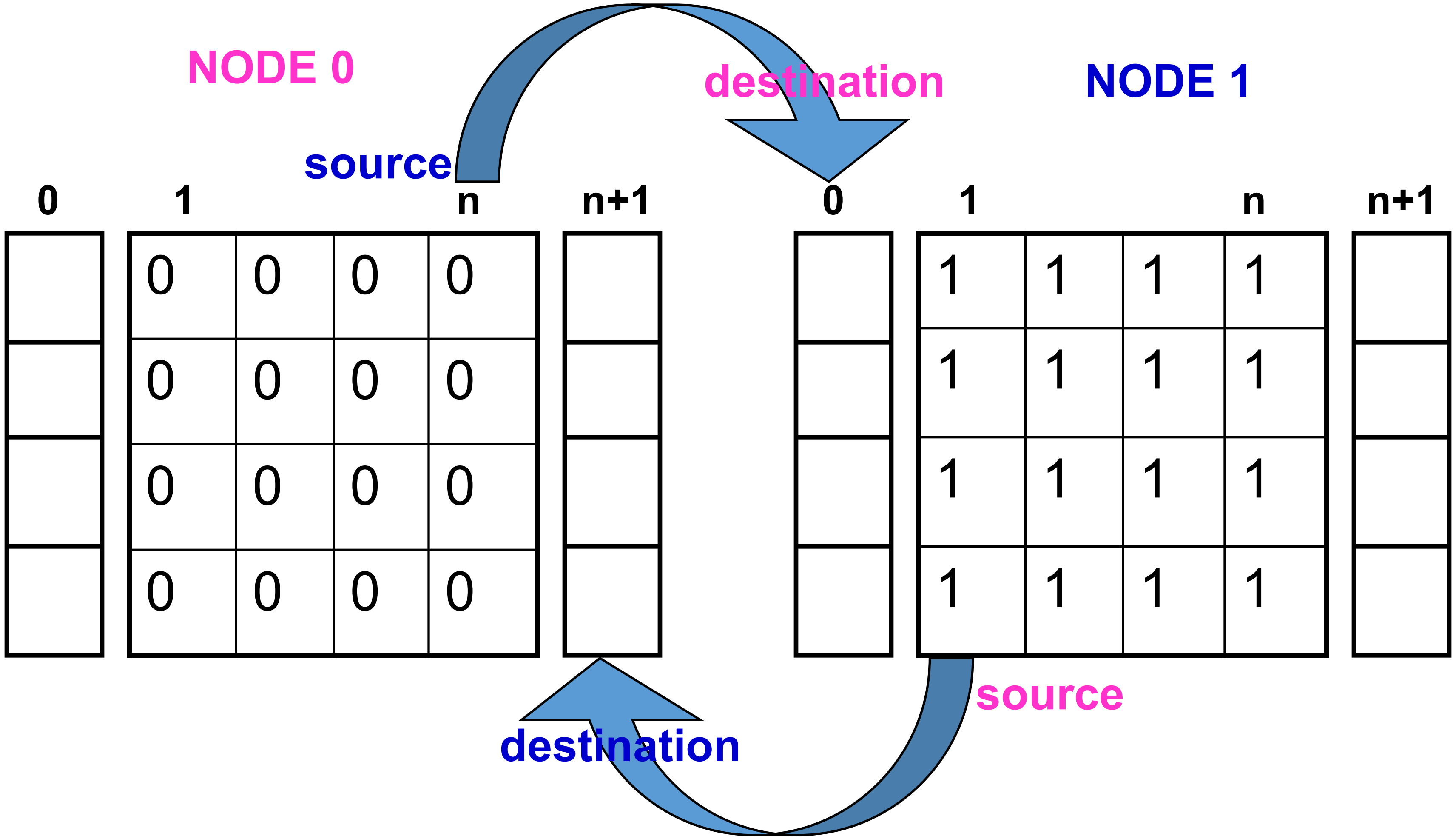
C



0	1			n	n+1
	1	5	9	13	
	2	6	10	14	
	3	7	11	15	
	4	8	12	16	

Domain Decomposition

Exchange ghost cell data



0	1			n	n+1
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	

0	1			n	n+1
	1	1	1	1	
	1	1	1	1	
	1	1	1	1	
	1	1	1	1	

Domain Decomposition – Ghost cell exchange

Fix the MPI_Sendrecv calls by filling in the missing data.

C_ghost_exchange.c

```
ierr=MPI_Sendrecv(
         <send_buf>,<send_count>,<send_MPItype>,<destination>,<send_tag>,
         <recv_buf>,<recv_count>,<recv_MPItype>,<source     >,<recv_tag>,
         MPI_COMM_WORLD, &status);
```

F_ghost_exchange.f90

```
call MPI_Sendrecv(  &
         <send_buf>,<send_count>,<send_MPItype>,<destination>,<send_tag>,   &
         <recv_buf>,<recv_count>,<recv_MPItype>,<source     >,<recv_tag>,   &
         MPI_COMM_WORLD, MPI_STATUS_IGNORE, ierr)
```

Domain Decomposition – Ghost cell exchange

Compile:

```
   make C_ghost_exchange  #C
   make F_ghost_exchange  #Fortran
```

Run:

```
   mpirun ./C_ghost_exchange #C
   mpirun ./F_ghost_exchange #Fortran
```

Part 3: Broadcast Data

Fill in the place holders to send the following scalars to all tasks

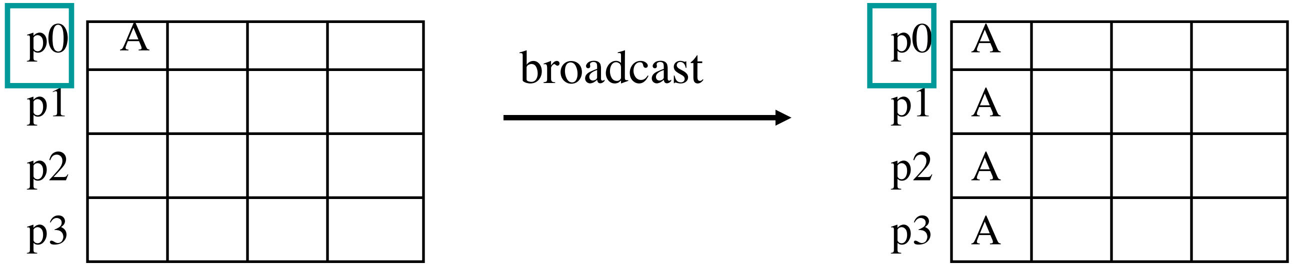
```
    nx, ny, lx, ly, alpha, beta, my_gamma
```

C_bcast.c

```
ierr=MPI_BCAST(<buf>,<count>,<MPItype>,<src>,MPI_COMM_WORLD);
```

F_bcast.f90

```
call MPI_BCAST(<buf>,<count>,<MPItype>,<src>,MPI_COMM_WORLD,mpi_err)
```



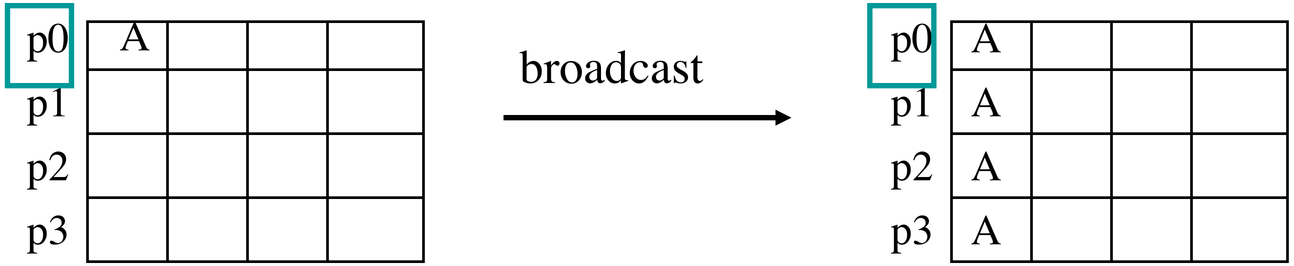
Part 3: Broadcast Data

As before, compile using the command:

```
make C_bcast   #C
make F_bcast   #Fortran
```

Run:

```
mpirun C_bcast < stommel.in
mpirun F_bcast < stommel.in
```



p0	A			
p1				
p2				
p3				

broadcast

p0	A			
p1	A			
p2	A			
p3	A			