CS677: Lecture 4

Parallel Programming

August 13, 2024

References for MPI

- (CSA) DE Culler, JP Singh and A Gupta, Parallel Computer Architecture: A Hardware/Software Approach Morgan-Kaufmann, 1998.
- (GGKK) A Grama, A Gupta, G Karypis, and V Kumar, Introduction to Parallel Computing. 2nd Ed., Addison-Wesley, 2003.
- (MPI) Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker and Jack Dongarra, MPI - The Complete Reference, Second Edition, Volume 1, The MPI Core.
- (GLS) William Gropp, Ewing Lusk, Anthony Skjellum, Using MPI: portable parallel programming with the message-passing interface, 3rd Ed., Cambridge MIT Press, 2014.
- (PP) Peter S Pacheco, An Introduction to Parallel Programming, Morgan Kaufmann, 2011.

MPI Implementations

MPI report

- MPICH (ANL)
- MVAPICH (OSU)
- OpenMPI
- Intel MPI
- Cray MPI

H.W.: Install MPI on your Laptop

- Linux or Linux VM on Windows
  - apt/snap/yum/brew
- Windows
  - No support

- https://www.mpich.org/documentation/guides/

Programming

- Shell scripts (e.g. bash)
- ssh basics
  - E.g. ssh –X
  - …
- Mostly in C/C++
- Compilation, Makefiles, ...
- Linux environment variables
  - PATH
  - LD_LIBRARY_PATH
  - …

MPI Programming

```c
#include <stdio.h>
#include "mpi.h"

int main(int argc, char *argv[])
{
  // initialize MPI
  MPI_Init (&argc, &argv);

  printf ("Hello, world!\n");

  // done with MPI
  MPI_Finalize();
}

~
~
```

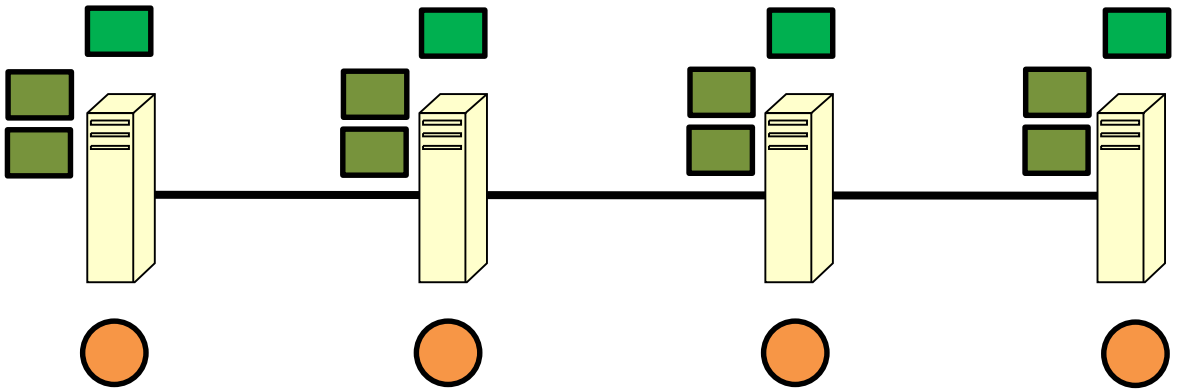
mpicc -o program.x program.c
mpirun –np 2 ./program.x

MPI Programming

```c
#include <stdio.h>
#include "mpi.h"

int main(int argc, char *argv[])
{
  int numtasks, rank, len;
  char hostname[MPI_MAX_PROCESSOR_NAME];

  // initialize MPI
  MPI_Init (&argc, &argv);

  // get number of tasks
  MPI_Comm_size (MPI_COMM_WORLD, &numtasks);

  // get my rank
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);

  // this one is obvious
  MPI_Get_processor_name (hostname, &len);

  printf ("Number of tasks=%d My rank=%d Running on %s\n", numtasks, rank, hostname);

  // done with MPI
  MPI_Finalize();
}
```

mpicc -o program.x program.c
mpirun –np 2 ./program.x

Communication using Messages



Core
Process
Memory

Local data
Instruction 1
Instruction 2
…

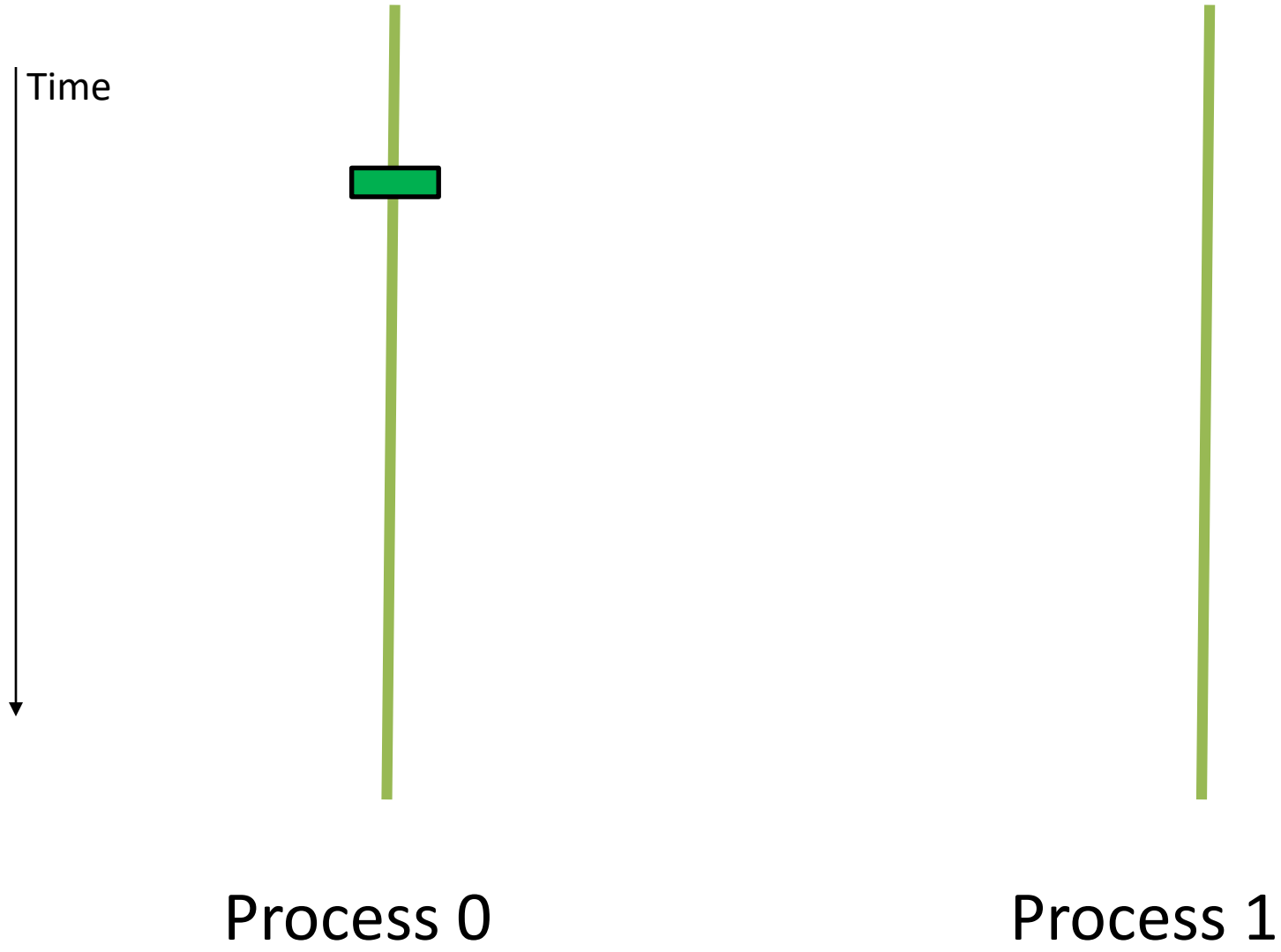
Local data
Instruction 1
Instruction 2
…

Local data
Instruction 1
Instruction 2
…

Local data
Instruction 1
Instruction 2
…

SIMD

Message Passing

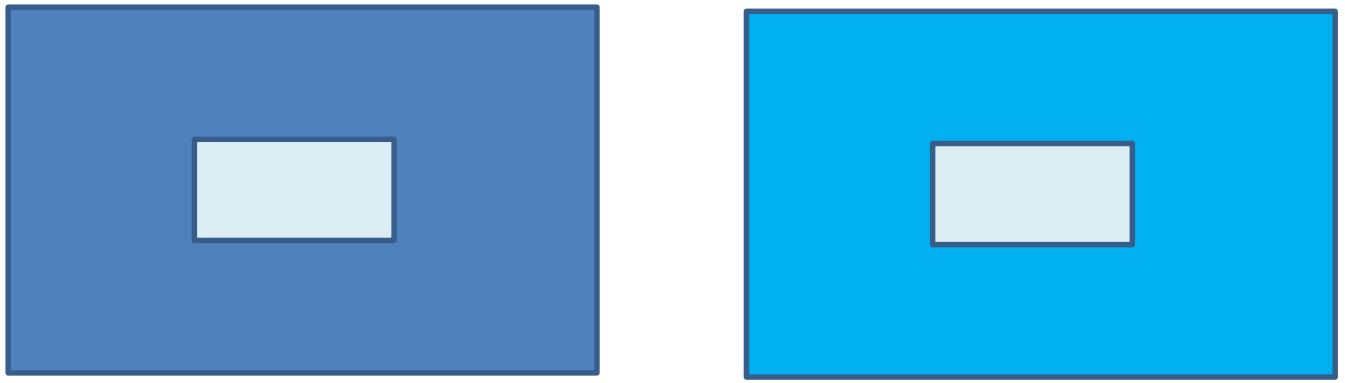


Simplest Communication Primitives

- MPI_Send
- MPI_Recv

MPI Programming

int MPI_Send (const void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)



int MPI_Recv (void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)

MPI Programming

```
MPI_Comm_rank (MPI_COMM_WORLD, &myrank);

// Sender process
if (myrank == 0)    /* code for process 0 */
{
  strcpy (message,"Hello, there");
  MPI_Send (message, strlen(message)+1, MPI_CHAR, 1, 99,
MPI_COMM_WORLD);
}

// Receiver process
else if (myrank == 1)  /* code for process 1 */
{
  MPI_Recv (message, 20, MPI_CHAR, 0, 99, MPI_COMM_WORLD,
&status);
  printf ("received :%s\n", message);
}
```

MPI – Parallel Sum

Assume the data array resides in the memory of process 0 initially

```
MPI_Comm_rank (MPI_COMM_WORLD, &myrank);

// Sender process
if (myrank == 0)  /* code for process 0 */
{
 for (int rank=1; rank<SIZE ; rank++) {
   start = rank*N/size*sizeof(int);
   MPI_Send (data+start, N/size, MPI_INT, rank, 99, MPI_COMM_WORLD);
 }
}
else /* code for processes 1 … SIZE */
{
  MPI_Recv (data, N/size, MPI_CHAR, 0, 99, MPI_COMM_WORLD, &status);
}
```

MPI Timing

```
double stime = MPI_Wtime();
  …
  …
  …
double etime = MPI_Wtime();
```

Code 1

- Compile
  - mpicc –o simple simple.c
- Execute
  - mpirun –np 2 ./simple 4

Code 2

- Compile
  - mpicc –o findMin findMin.c
- Execute
  - mpirun –np 2 ./findMin 4
  - time mpirun –np 2 ./findMin 4
  - time mpirun –np 2 ./findMin 4000
  - time mpirun –np 8 ./findMin 4000

Performance Measure

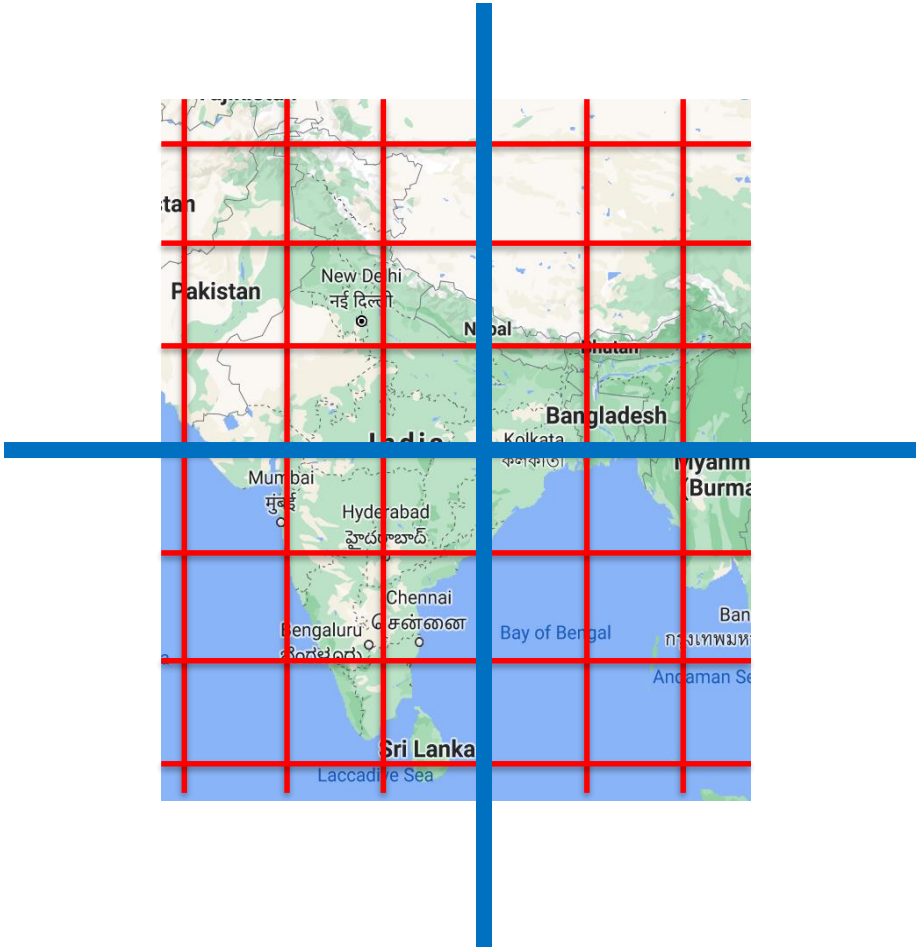
- Speedup

$$S_P = \frac{\text{Time (1 processor)}}{\text{Time (P processors)}}$$

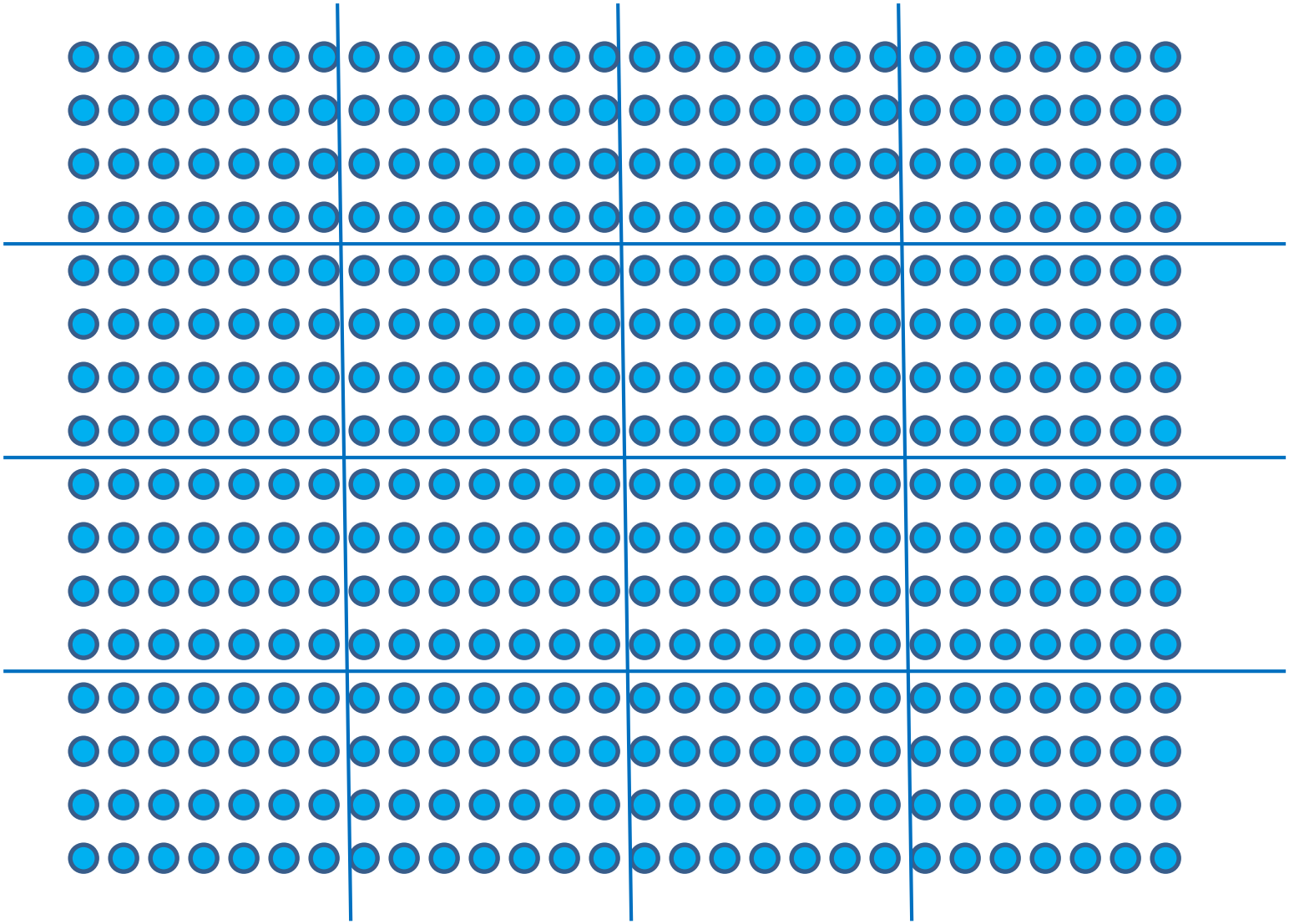
- Efficiency

$$E_P = \frac{S_P}{P}$$

Domain Decomposition



Nearest Neighbour Computations



Code 3

- Compile
  - mpicc –o findMinNeighbour findMinNeighbour.c
- Execute
  - mpirun –np 2 ./findMinNeighbour 4
  - time mpirun –np 2 ./findMinNeighbour 4
  - time mpirun –np 2 ./findMinNeighbour 4000
  - time mpirun –np 8 ./findMinNeighbour 4000

Thank You