

## Concurrency and Parallel Programming

### Lab 2: MPI

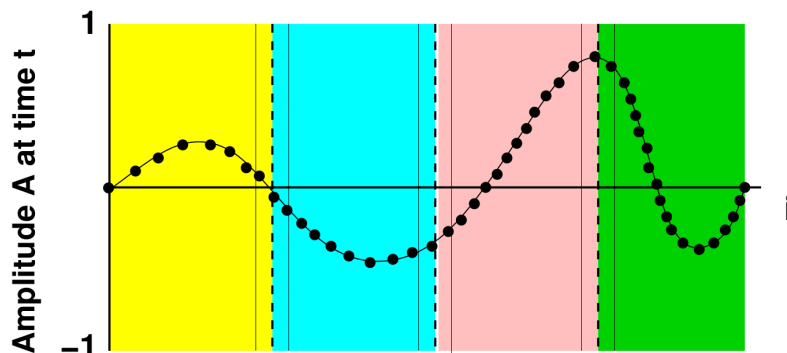
#### Parallel Programming for Distributed Memory Systems

dr. Clemens Grelck

2019

#### Assignment 2.1: Wave equation simulation with MPI

Reconsider the 1-dimensional wave equation studied in assignment 1.1. Write a distributed MPI program in C that uses multiple compute nodes to simulate the wave equation following the domain decomposition approach as illustrated below and explained during the lecture.



Each MPI process shall only store the relevant parts of the three amplitude buffers in its local memory. Add halo cells as necessary and exchange their values between time steps as needed. Aim at an efficient implementation that overlaps communication with computation as demonstrated during the lecture.

Repeat your experiments of assignment series 1 with three different system configurations: up to 8 MPI processes on a single node, up to 8 nodes with a single MPI process per node and up to 8 nodes with 8 MPI processes per node. Compare the results of the new experiments with those of assignments 1.1 and 1.2.

## Assignment 2.2: Overlapping communication and computation

Overlapping communication (overhead) with computation (productive processing) is key to achieve good performance. Develop two versions of the 1-d wave equation that more effectively overlap the communication with computation than the example shown in the lecture:

1. Solely with the standard non-blocking send / blocking receive.
2. With making use of asynchronous communication.

Use pseudo code notation in analogy to the lecture and explain your pseudo code in plain text.

## Assignment 2.3: Collective communication

Collective communication is a form of structured communication, where, instead of a dedicated sender and a dedicated receiver, all MPI processes of a given communicator participate. A simple example is broadcast communication where a given MPI process sends one message to all other MPI processes in the communicator. Write an MPI function in C:

```
int MYMPI_Bcast (
    void *buffer,           // INOUT : buffer address
    int count,              // IN    : buffer size
    MPI_Datatype datatype, // IN    : datatype of entry
    int root,               // IN    : root process (sender)
    MPI_Comm communicator // IN    : communicator
)
```

The function should implement broadcast communication by means of point-to-point communication. Each MPI process of the given communicator is assumed to execute the broadcast function with the same message parameters and root process argument. The root process is supposed to send the content of its own buffer to all other processes of the communicator. Any non-root process uses the given buffer for receiving the corresponding data from the root process.

A particular advantage of collective communication operations is that their implementation can take advantage of the underlying physical network topology without making an MPI-based application program specific to any such topology. For your implementation of broadcast assume a 1-dimensional ring topology, where each node only has communication links with its two direct neighbours with (circularly) increasing and decreasing MPI process ids. While any MPI process may, nonetheless, send messages to any other MPI process, messages may need to be routed through a number of intermediate nodes. Communication cost can be assumed to be linear in the number of nodes involved. Aim for an efficient implementation of your function on an (imaginary) ring network topology.

No experiments are required for this assignment, but make sure your solution actually works and describe your code in detail in your report: how it is adapted to the given network topology and how many atomic communication events (sending a message from one node to a neighbouring node) are required to complete one broadcast.

## Additional Information

### Grade weights

Table 1: Weights of different parts of the assignments used in calculating the final grades.

Assignment	Name	Code	Report
2.1	Wave equation simulation with MPI	35%	20%
2.2	Overlapping communication and computation	0%	20%
2.3	Collective communication	15%	10%
		50%	50%

### Instructions for submission

For each assignment we expect two kinds of deliverables:

1. Source code in the form of a tar-archive of all relevant files that make up the solution of a programming exercise with an adequate amount of comments ready to be compiled.
2. A report in the form of a pdf file that explains the developed solution at a higher level of abstraction, illustrates and discusses the outcomes of experiments and draws conclusions from the observations made.

Please submit one archive containing both your code and your report (one report for the entire assignment).