TDT4200 Problem Set 2

Graded MPI

Maren Wessel-Berg and Anne C. Elster

Deadline: September 27, 2022 by 22:00 in Blackboard
Evaluation: Graded (20%)

- **This assignment is graded and will count towards 20% of your final course grade.** The assignment must receive a pass grade for you to take the final exam in the course.
- **This assignment must be done individually and without help from anyone but the TDT4200 staff.** All sources found on the internet or elsewhere must be referenced. We encourage that you post clarification questions on the Forum in BB so all can benefit. However, make sure you do not post full or partial solutions on the BB Forum.
- **This assignment has two parts.** In the first part, you will work on an MPI implementation of the Finite Difference Method (FDM) for solving the 2D shallow water equations. In the second part, you will answer four questions about your implementation.
- **Solution requirements** are stated in the Evaluation section.
- **Do not deliver any other files than those specified in the Delivery section.** Code should not use external dependencies aside from the ones already included in the handout code.

Finite difference approximation of the 2D shallow water equations using MPI

In this assignment, you will work on an MPI implementation of the Finite Difference Method (FDM) for solving the 2D shallow water equations. We provide a partially complete MPI implementation in the handout code, and your job is to complete the implementation using MPI. Information on solving the shallow water equations using FDM and on parallelization with MPI is described in the lecture slides.

The partial parallel implementation can be found in `shallow_water_parallel.c`. The approach you should follow is described in the Tasks section. A serial solution is also provided as a reference, and can be found in `shallow_water_serial.c`.

Program description

Information on how to install the dependencies and run the program is provided in the `README.md` of the handout code.

1 Tasks

The following tasks all have corresponding TODO-comments in `shallow_water_parallel.c`. We recommend that you ensure the program compiles after completing a task even though the program might not run correctly between tasks.

1. (15%) Create a communicator named *cart* with cartesian topology.
2. (10%) In the function `domain_init()`, find the number of columns and rows in each process' sub-grid and find each process' offset to calculate the correct initial values.

   Hint You can get useful information from the cartesian communicator.

   Note You are allowed to assume that the grid size is divisible by the number of processes.

   After 1. and 2. The functions `domain_save()` and `create_types()` depend on the cartesian communicator *cart*, so you will have to uncomment the calls to `domain_save()` and `create_types()` after you have initialized the cartesian communicator.
3. (5%) Change the function `time_step()` so that each process only iterates over its own sub-grid.
4. (15%) Change the function `boundary_condition()` so that the application of boundary conditions matches the cartesian topology.

5. (25%) As was the case with FDM in 1D, there is a dependency between the processes where a process needs to use values from the other processes' sub-grids to perform time step computations in their own sub-grid. Therefore, the border values of the mass and the mass velocity in the x- and y-direction must be communicated in each iteration. This is described in the slides from the recitation lecture. Implement the border exchange.

2 Questions

- (5%) How did you avoid a deadlock during the border exchange?
- (5%) How could creating derived MPI datatypes be useful for the border exchange?
- (5%) Describe the pros and cons of using `MPI_Cart_shift` in connection with cartesian communicators?
- (5%) Look at the execution time of the time loop when running with 1 processes. Look at the execution time of the time loop when running with 2, 4 and 8 processes and document the *speedup* you get compared to running with a single process. Is the result as you would expect? Why/why not?

Evaluation

This assignment is graded and will count towards 20% of your final course grade and is graded as follows:

- **MPI implementation: 70%**

  The MPI implementation will count towards 70% of the problem set grade, i.e., 14% of your final course grade.

  – Your implementation should pass the correctness test for 1, 2, 4, and 8 processes.

  – Partial solutions will be given some percentage points if the intent behind the code is clear and well-documented. The possible percentage points for each task is indicated in the Tasks section.

- **Documentation and code clarity: 10%**

  Documentation and clarity of the code **you write** will count towards 10% of the problem set grade, i.e., 2% of your final course grade. You should document each function and what is does, use sensible variable names, etc.

- **Questions: 20%** Answers to the questions will count towards 20% of the problem set grade, i.e., 4% of your final course grade. The possible percentage points for each question is indicated in the Questions section. You should answer each question properly, but there is no need to write long essays.

Delivery

Deliver the file `shallow_water_parallel.c` in Blackboard. Answers to the questions should also be delivered in Blackboard in a separate PDF file. Do **NOT** upload a ZIP file.