High Performance Computing Proseminar 2024

Assignment 2

Stefan Wagner & Sebastian Bergner

October 9, 2024

Exercise 1

This exercise consists in writing a parallel application to speed up the computation of π.

There are many ways of approximating π, one being a well-known Monte Carlo method: The ratio of the areas of a square and its incircle is $\pi/4$. Since the exact area of a circle cannot be computed (we don't know the value of π yet), one can instead sample random points, check their distance from the center and compute the ratio of points inside the circle to all sampled points.

- Write a sequential application 'pi_seq' in C or C++ that computes π for a given number of samples (command line argument). Test your application for various, large sample sizes to verify the correctness of your implementation.
- Consider a parallelization strategy using MPI. Which communication pattern(s) would you choose and why?
- Implement your chosen parallelization strategy as a second application 'pi_mpi'. Run it with varying numbers of ranks and sample sizes and verify its correctness by comparing the output to 'pi_seq'.
- Discuss the effects and implications of your parallelization.
- Insert the measured wall time for 10^9 samples for the sequential implementation and on 96 cores for MPI into the comparison spreadsheet

Exercise 2

This exercise consists in parallelizing an application simulating the propagation of heat.

A large class of scientific applications are so-called stencil applications. These simulate time-dependent physical processes such as the propagation of heat or pressure in a given medium. The core of the simulation operates on a grid and updates each cell with information from its neighbor cells.

- A sequential implementation of a 1-D heat stencil is available in heat_stencil_1D_seq.c. Read the code and make sure you understand what happens. See the Wikipedia article on Stencil Codes for more information.
- Consider a parallelization strategy using MPI. Which communication pattern(s) would you choose and why? Are there additional changes required in the code beyond calling MPI functions? If so, elaborate!
- Implement your chosen parallelization strategy as a second application 'heat_stencil_1D_mpi'. Run it with varying numbers of ranks and problem sizes and verify its correctness by comparing the output to 'heat_stencil_1D_seq'.
- Discuss the effects and implications of your parallelization.
- Insert the measured wall time for N=6144 and 96 cores into the comparison spreadsheet