Universität Heidelberg
Institut für Technische Informatik

Holger Fröning
Benjamin Baumann

Introduction to High Performance Computing
Term 2014/2015 (Winter)

Exercise 6

- **Return electronically (MOODLE) until Monday, 01.12.2014 23:55.**
- **Include name and your account (introhpc[xx]) on the top sheet.**
- **A maximum of two students is allowed to work jointly on the exercises.**

6.1 Heat Relaxation II – Parallel implementation based on 1D-row partitioning

Use the program developed in exercise 4 to start with this exercise. The configuration is the same as before, but this time the goal is to parallelize this program for MPI.

- In this exercise, perform a 1D row partitioning. Thus, each process has to exchange only consecutive addresses (C has a row-based layout in memory) to its neighbor, simplifying the communication at the cost of additional communication per process.
- Make a new communicator with topology information attached. See `MPI_Cart_create`. The topology information helps to determine processes working on neighbor data domains. See `MPI_Cart_shift`.
- Ensure that you use two arrays for the current and previous grid. Do not exchange these two arrays by copying (after one iteration), instead just exchange the pointers to them. Saving one copy saves plenty of processor cycles.
- Try to maximize overlap by first calculating the borders and send them out in a non-blocking fashion.
- Write a suitable parallel program that performs the described calculation. Test your program extensively. Compare results against the sequential implementation.

(50 points)

Holger Fröning
Benjamin Baumann

6.2 Heat Relaxation II - Experiments

- Measure the average time for grid sizes of **128 x 128, 512 x 512, 1024 x 1024, 2k x 2k, 4k x 4k**. Report the average time of one iteration by performing for instance 100 iterations, measuring the time with a suitable function (e.g., *gettimeofday()* in Linux) and dividing by the number of iterations. Do not include time for initialization or output. Use compiler-specific optimizations to minimize the runtime.
- Increase the number of processes from 2 to 12 for these grid sizes.
- Report how mapping was done and why you chose it this way!
- Fill out the following tables and interpret results!

	Time/iteration					
Grid size	NP=2	NP=4	NP=6	NP=8	NP=10	NP=12
128 x 128						
512 x 512						
1024 x 1024						
2k x 2k						
4k x 4k						

	Speedup (compared to sequential execution time)					
	Sequential execution time:					
Grid size	NP=2	NP=4	NP=6	NP=8	NP=10	NP=12
128 x 128						
512 x 512						
1024 x 1024						
2k x 2k						
4k x 4k						

	Efficiency (compared to sequential execution time)					
Grid size	NP=2	NP=4	NP=6	NP=8	NP=10	NP=12
128 x 128						
512 x 512						
1024 x 1024						
2k x 2k						
4k x 4k						

(20 points)

Note: Ensure that your program is running without problems even with different configurations, like more processes or larger grids.

Total: 70 points