

# **Introduction to Parallel and Distributed Computing Exercise 4 (June 29)**

Wolfgang Schreiner  
Wolfgang.Schreiner@risc.jku.at

The result is to be submitted by the deadline stated above via the Moodle interface as a .zip or .tgz file which contains

- a single PDF (.pdf) file with
  - a cover page with the title of the course, your name, Matrikelnummer, and email-address,
  - a section with the source code of the program benchmarked, the output of the parallelizing compiler, and an explanation of the output,
  - a section with the raw data of the benchmarks,
  - a section with a summary table and graphical diagrams of the benchmarks.
- the source (.c) file(s) of the programs.

## Exercise 4: Message Passing Programming in MPI

The goal of this exercise is to develop a MPI-based parallel solution to the “all pairs shortest paths” problem described in Exercise 1.

**MPI Program** Initially process 0 broadcasts  $D := W$  to every process (MPI\_Bcast). The program then runs in multiple rounds where in each round  $D := D \times D$  is computed as follows:

- every process computes some rows of  $D \times D$  (row-wise parallelization),
- every process gathers the results of all other processes (MPI\_Allgather) such that every process holds again the complete  $D := D \times D$ .

The dimension  $n$  of the matrix is not necessarily a multiple of the number  $n$  of processes. To simplify the gathering, one may thus extend the matrix to a dimension  $n' = p \cdot \lceil n/p \rceil$  and let  $n'/p$  rows be gathered from every process.

**Scalability Analysis** Furthermore, perform a scalability analysis of the program, i.e.:

- determine the basic execution time  $T_n$ ;
- determine the parallelization overhead  $P_{p,n}$ ;
- determine the solution  $n_p$  of the constraint  $T_{n_p} = K_E \cdot P_{p,n_p}$ ;
- determine the isoefficiency function  $I_p^E = K_E \cdot P_{p,n_p}$ .

For determining  $P_{p,n}$ , you just need to consider the communication overhead. Here it suffices to use a simple communication model where sending a message of size  $m$  takes time  $\Theta(m)$ . Furthermore assume that broadcasting a message to  $p$  processors is implemented by sending  $p$  individual messages to each processor, i.e., takes time  $\Theta(p \cdot m)$ ; likewise, scattering a message of size  $m$  among  $p$  processors takes the same time as  $p$  times broadcasting a message of size  $m/p$ , i.e., it takes time  $\Theta(p \cdot p \cdot m/p) = \Theta(p \cdot m)$ .

**Benchmarking** Finally, benchmark the program as follows:

- Take the sequential solution and benchmark it with two appropriate values  $N_1, N_2$  for the matrix dimension (at least one should run for at least one minute).
- Benchmark the MPI version of the program for  $N_1$  and  $N_2$  and  $P = 1, 4, 8, 16, 32, 64$  processes. Do not forget to set the environment variable MPI\_DSM\_CPULIST to pin processes to separate physical processor cores.
- Apply the result of the scalability analysis to scale the larger of  $N_1$  or  $N_2$  for  $P = 1, 4, 8, 16, 32, 64$  processors; benchmark for these values both the sequential and the parallel program. Do the benchmark results confirm the results of the scalability analysis (i.e., is the efficiency preserved at a high level)?

Perform your benchmarks and present the results in the same way as in Exercise 1.