

Introduction to Parallel and Distributed Computing Exercise 1 (April 16)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

March 31, 2008

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

- a 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 of the benchmarks.
- the source (.c/.f) file(s) of the programs.

Exercise 1: Benchmarking

Take *either* the Fortran program `pathf.f` *or* the C program `pathc.c` (file “Example Programs” from the course web site, subdirectory “auto”) for solving the “all pairs shortest paths” problem and benchmark the program as follows:

Instrumentation Instrument the source code of the program to measure the real (“wall clock”) time spent (only) in the execution of function/subroutine `path` (in milliseconds) and print it to the standard output.

In a C program, you may do this with the help of the library function `clock_gettime` (as demonstrated in the example program `time.c` in “Example Programs”).

In a Fortran program, you may do this with the help of the intrinsic function `SYSTEM_CLOCK` described in the “Intel Fortran Libraries Reference” available in the web¹.

¹http://www.intel.com/software/products/compilers/techttopics/for_lib.htm

Benchmark Sequential Execution Compile the sequential program with the Intel Fortran/C compiler and optimization option `-O3`. Benchmark the program for $N \in \{512, 768, 1024\}$ (and L adapted such that $N \leq 2^L$).

Benchmark Parallel Execution Compile the program with the Intel Fortran/C compiler and parallelization option `-O3 -parallel -par-report3`. Investigate and explain the compiler output.

Benchmark the parallel program for $N \in \{512, 768, 1024\}$ and (and L adapted such that $N \leq 2^L$) for 1, 2, 4, 8, 12, 16 processors.

Repetition Repeat the sequential and the parallel benchmark 5 times and collect all results. For automating this process, the use of a shell script is recommended. For instance, a shell script `loop.sh` with content

```
#!/bin/bash
for p in 1 2 4 8 12 16 ; do
  echo $p
done
```

can be executed as `bash loop.sh` to print out a sequence of values.

For each run of a program, before starting the program, it may be helpful to record the output of

```
cat /proc/loadavg
```

which prints a line of form

```
213.73 221.78 218.23 197/3736 7429
```

where the first three numbers represent the average load of the system in the last minute, the last 5 minutes, and the last 15 minutes, and the number 197 is the number of threads currently being executed (from 3736 existing ones). In this way, it may be determined after the benchmark whether for a particular program run a sufficient numbers of free processors was available.

Present all timings in an adequate form in the report.

Summary Construct a summary table that reports for each value of N

- the average execution time of three runs of the sequential program (excluding those two runs that took shortest and longest) and

for each value of N and P

- the average execution time of three runs of the parallel program (excluding those two runs that took shortest and longest),
- the average speedup, i.e., the average parallel execution time divided by the average sequential execution time, and
- the average efficiency, i.e., the average speedup divided by the number of processors used.