Universidade de Aveiro

Mestrado Integrado em Engenharia de Computadores e Telemática

Arquitectura de Computadores Avançada

Performance Measurements

Academic year 2019/2020 Adaptation of the exercise guide by Nuno Lau/José Luís Azevedo

1. The `perf-meas.zip` archive (available at the course elearning site) contains the following files
   - `dhrystone.c` – C code for an implementation of the synthetic Dhrystone benchmark
   - `whetstone.c` – C code for an implementation of the synthetic Whetstone benchmark
   - `prime_v1.c` and `prime_v2.c` – C code of two programs that compute the number of primes that exist up to a given number, using simple algorithms; these programs will be used as examples of toy benchmarks
   - `crypt.c` and `newCrypt.c` – C code of two programs that access a memory region in two different ways; these programs will also be used as examples of toy benchmarks
   - `perf-meas.xls` – Excell file containing a table to be filled in with the results.

   1.1. Download and decompress the `perf-meas.zip` archive to your workspace under Linux.

   1.2. Compile the programs, first without optimization and then with the `-O3 -march=native` flags (add `_O3` to the name of the executable file when optimization is being used). Run each executable file five times, write down in the table the respective execution times and compute their average value and the speed up obtained by compiler optimization.

   1.3. Repeat the operations above, now under Windows.

   1.4. Compare the execution times under both operating systems.

   1.5. Discuss the results.

2. Using the Linux command *gprof* it is possible to check the time spent by a generic program in each of its procedures. This operation, called *profiling*, allows one to pinpoint the code regions where the program spends most of its time during execution. Thus, enabling one to perform code optimization manually. The steps to carry out this analysis are the following
   - compile the program using the `-pg` switch

     ```
     gcc -pg dhrystone.c -o dhry_prof
     gcc -pg whetstone.c -o whet_prof
     ```

     
   - run the program

     ```
     ./dhry_prof
     ./whet_prof
     ```
   - run *gprof* to summarize the results

     ```
     gprof dhry_prof > dhry_gprof.out
     gprof whet_prof > whet_gprof.out
     ```

   The file `*_gprof.out` now contains all the profiling information about the execution of the program. Read it carefully.

   2.1. In order to optimize the programs, based on the *gprof* results, which procedures would you try to improve in the first place?

   2.2. What is the maximum overall speed up that can be achieved through the optimization of those procedures? Correlate the results with *Amdhal law*.

3. Using the Linux command *perf* a more detailed profiling of `dhrystone`, `whetstone`, `crypt` and `newCrypt` programs can be carried out. Read carefully the man page of *perf* and execute the commands `perf record <file name>`, `perf report` and `perf stat <file name>`. Discuss the results.