Advanced Computing – 2022/2023

MsC in Informatics – ESTiG/IPB

Practical Work 1:
N-Body Acceleration with PThreads

Goal: To accelerate a simulation of the N-Body scenario using the Pthreads programming model.

Preliminary Note:
This work is accompanied by a ZIP archive with related resources (ac-nbody-resources.zip).

Introduction:
“In physics and astronomy, an N-body simulation is a simulation of a dynamical system of particles, usually under the influence of physical forces, such as gravity [...]. N-body simulations are widely used tools in astrophysics, from investigating the dynamics of few-body systems like the Earth-Moon-Sun system to understanding the evolution of the large-scale structure of the universe. In physical cosmology, N-body simulations are used to study processes of non-linear structure formation such as galaxy filaments and galaxy halos from the influence of dark matter. Direct N-body simulations are used to study the dynamical evolution of star clusters." [1]

Details and Methodology:

In this work you are given a serial implementation of an N-Body simulation and you are required to develop a parallel version based on the Pthreads programming model, faster than the serial version. The serial version (non-GUI and GUI variants) is the one provided in [2] with some modifications.

You can use your personal computer if you have Linux installed (either a bare-metal installation or in a virtual machine). Mind you, however, that a limited number of CPU-cores will limit the scalability tests. Therefore, for the final benchmarking it is recommended to use the frontend of the cluster or at least a computing system that supports no less than eight simultaneous threads.

Advice: to ensure that the various version of the code developed throughout the practical work are robust, run the code via valgrind [3] to detect memory-leaks or memory access violations; note that executing through valgrind should only be done for validation, never when benchmarking.

a) Compiling and Running the Serial Code

The companion ZIP archive contains 4 files: a project file (Makefile), the source code of the serial version (nbody-serial.c), the source code of GUI-based serial version (nbody-serial-gui.c) and the starting point for the source code of the parallel version (nbody-threads.c, a copy of nbody-serial.c).

Extract the ZIP archive file in the system you choose to develop the work. Then, edit the Makefile and comment/uncomment the appropriate lines. To compile, execute make. Note that if you don't have the SDL bgi library [4] installed, you won't be able to compile and run the serial GUI version.

To execute the serial version just type ./nbody-serial.exe ; this will run the simulation for 15 bodies along 30000 time steps; these are default values that can be changed in the source code; other values that may be changed are the width and height of the simulation grid (these are the also the

dimensions, in pixels, of the output window in the GUI version), but stick with the defaults (800); these 4 parameters (numBodies, numSteps, windowWidth, windowHeight) may also be passed to the program as command line parameters; for instance: /nbody-serial.exe 300 30000 800 800 .

To run the GUI version with the same default parameters of the non-GUI version just type ./nbody-serial-gui.exe ; you may also pass it the same command line parameters of the non-GUI version; however, the GUI version is just for fun - it is not meant for benchmarking neither parallelization.

b) Optimizations to the Serial Version

Note: this step is optional (you may skip it); however, it provides extra-points.

Inspect the serial code (nbody-serial.c) carefully, to apprehend its basic structure and determine if there are some opportunities to improve its performance, still without resorting to parallelization. If you find possible optimizations, produce a new serial version (nbody-serial-opt.c) with them applied. Then, measure the execution time of both serial versions (original and optimized):

```
time ./nbody-serial.exe 300 30000 2> nbody-serial-300-30000.txt
time ./nbody-serial-opt.exe 300 30000 2> nbody-serial-opt-300-30000.txt
```

The .txt output files of both versions should be equal or very similar (if not, the optimized version is not reliable and must be corrected). Use the `diff` command or Meld [5] to compare the .txt files.

Take 3 measurements for each version and use the lowest times to calculate the optimized version speedup. In your report, explain the optimizations applied and present the speedup value achieved.

c) Profiling with valgrind

Note: if you developed the optional optimized serial version, use it as the starting point for this stage; otherwise, use the original serial version; both are referred as nbody-serial.exe from now on.

Modify the Makefile to compile the code properly to run through valgrind, and use its callgrind tool to capture the profiling data. Important: for profiling, use numBodies=300 and numSteps=3000 (do not change any other parameters). After running nbody-serial.exe through valgrind (which may take considerable time - for reference, it takes ≈14m in a i7-8665 CPU) use the kcachegrind [6] utility to visualize the call graph and identify the most relevant code hot-spots.

In your report you must: i) provide the command used to run nbody-serial.exe through valgrind; ii) provide one or more screenshots of the kcachegrind windows that shows the hot-spots (left panel) and the call graph of `main`; iii) identify the biggest hot-spots and their relative importance (%).

Also, analyze the code of the biggest hot-spots and identify those that seem to be easily parallelizable; in your report, state those functions and explain why they are easily parallelizable.

d) Applying Amdahl's Law

Considering the aggregated weight (%) of the code that you decided to parallelize, apply Amdahl's Law to calculate the theoretical speedup S_T of the parallelization for a number of threads N=1,...,8. Explain and justify, in your report, the value given to the different factors of the Amdahl's Law formula and provide a table like the following with the missing values (?) of the S_T speedups.

N	1	2	3	4	5	6	7	8
S_T	?	?	?	?	?	?	?	?

Table 1 - Theoretical Speedups

Based on the Table 1 values, provide also a table with the corresponding theoretical efficiency E_T :

N	1	2	3	4	5	6	7	8
E_T (%)	?	?	?	?	?	?	?	?

Table 2 - Theoretical Efficiency (in percentage)

Finally, determine the theoretical speedup limit and also present that calculation in your report.

e) Run the Serial Version without Profiling

Note: you only need to do this step if you have not done step b); otherwise, you should already have measured the execution times of the serial version and you should have the .txt output file

Later, to evaluate the correctness and performance of the parallel version, you will need to compare its execution time and its output with that of the serial version. Therefore, before developing the parallel version do the following: edit the Makefile and do the necessary changes to compile the non-GUI serial version for benchmarking (not for profiling); remake the executables (make clean; make) and run the serial version with time ./nbody-serial.exe 300 30000 2> nbody-serial-300-30000.txt (note that numSteps is now 30000 and not 3000); take note of the real time (if possible run two more times and take note of the smallest of the three real times observed); preserve the file nbody-serial-300-3000.txt to be later compared with the output of the parallel version.

f) Parallelization

f.1) Development

Now it's time to parallelize the code. Decide how will you divide the work (the hot-spot functions that you choose to parallelize) by threads and how will you synchronize them. Explain your strategy in the report. The code should balance the work by the available threads as equally as possible. The code must also be prepared to execute with any number of threads (N), irregardless of the amount of work units available (e.g., if you have more threads than work units, some threads will be idle). Try to minimize the synchronization points of the threads as they will impair performance.

Hint: start with 2 threads and compare the output of the simulation with the serial version, to ensure the results are the same (or at least very similar - a perfect matching will depend on some choices you will have to make during the implementation); then, increment the number of threads; during development use the same parameters used for profiling (numBodies=300 and numSteps=3000); again, to compare the outputs (.txt files) of the simulations use the `diff` command or Meld [5].

The code for the Pthreads version must be in the file nbody-threads.c . This file is initially supplied as a copy of nbody-serial.c and you will change it progressively to produce your Pthreads version.

e.2) Benchmarking

When ready for benchmarking, make sure the Makefile is properly configured and generate the final ./nbody-threads.exe executable. Then run time ./nbody-threads.exe 300 30000 at least 3 times for each different number of threads N=1,...,8 and take note of the smallest real time T_R for each N. In you report, provide the following table filled with the smallest real times (? in seconds) observed:

N	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
T_R (s)	?	?	?	?	?	?	?	?

Table 3 - Real Execution Times of the Parallel Version (in seconds)

Now it's time to calculate the real speedups S_R. Provide in your report the following table:

N	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
S_R	--	?	?	?	?	?	?	?

Table 4 - Real Speedups of the Parallel Version

Provide also in the report a table with the real efficiency E_R achieved by the parallel version, and the ratio E_R/E_T (this ratio tells how close is the parallel implementation to the Amdahl's Law prediction):

N	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
E_R (%)	--	?	?	?	?	?	?	?
E_R/E_t (%)	--	?	?	?	?	?	?	?

Table 5 - Efficiency of the Parallel Version (in %) and closeness of the real to ideal efficiency (in %)

For each table, provide also a companion graph, to make the data easier to understand and interpret.

f) Discussion

Discuss the results achieved. Do they correspond to the predictions ? Are they behind expectations (and if so, what could be the motive(s)) ? Are there some more optimizations that can still be done ?

-- / --

Groups
Students may only form groups of 2 elements. Any other cardinality requires teacher authorization.

Deadline:
The report (PDF) and code (all zipped) should be sent to rufino@ipb.pt until December 18th 2022.

Plagiarism:
Plagiarism will not be tolerated and will be firmly handled by the current regulations of ESTiG/IPB.

References:
[1] https://en.wikipedia.org/wiki/N-body_simulation
[2] https://rosettacode.org/wiki/N-body_problem#C
[3] https://valgrind.org/
[4] https://sdl-bgi.sourceforge.io/
[5] https://meldmerge.org/
[6] https://kcachegrind.github.io/