High Performance Computing – 2025/2026

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 comes with a ZIP archive with related resources (nbody-pthreads-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.

The work involves several stages; some may be executed on your personal computer, others must be executed in a node of the cluster used in the classes, and others may be executed on either systems; read the specific instructions for each stage regarding the required execution environment.

a) Compiling and Testing the Serial Code
[this stage may be done on you personal computer, in the cluster frontend or in your cluster node]

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, edit the Makefile and (un)comment the appropriate lines, depending if you want to test or not the GUI version (this version is just for demonstration and requires the SDL bgi library [3] preinstalled; never use it for benchmarking). To compile, issue the make command.

To execute the serial version type ./nbody-serial.exe (non-GUI version) or ./nbody-serial-gui.exe (GUI version); default, this will run the N-Body simulation for 15 bodies, 30000 time steps, and a grid of 800x800; these 4 values (numBodies, numSteps, windowWidth, windowHeight) can be changed in the source code, or passed to the program as command line parameters; for instance: /nbody-serial.exe 15 30000 800 800 . To run the GUI version in the frontend, connect via mobaxterm, go (cd) to the folder where nbody-serial-gui.exe is located and execute export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib; ./nbody-serial-gui.exe; however, execution will be very slow compared to execution in your local Linux system; in addition, if you connect to the frontend from a Linux system, via ssh -X ..., the GUI version won't work at all; also, the GUI version will not work in the cluster nodes, whatever method you use to connect to the cluster.

b) Benchmarking the non-GUI Serial Version
[this stage must be done on the specific cluster node assigned to your group; this is mandatory]

Ensure the definitions in the Makefile are adequate to generate an executable suitable for benchmarking (uncomment CFLAGS=-Wall -O2 and comment CFLAGS=-Wall -O0 -g), and if necessary regenerate the executable of the non-GUI serial version (make clean; make). Run this version 3 times with time ./nbody-serial.exe 300 30000 2> nbody-serial-300-30000.txt, register the 3 execution times (only real times) and backup the last file nbody-serial-300-30000.txt generated.

Note: each run should take less than 3m30s; do not change the code to show additional output during execution, or execution may slow down considerably, contaminating the measurements.

In you report, provide the three execution times measured (only the real times).

c) Profiling the non-GUI Serial Version
[this stage must be done on the specific cluster node assigned to your group; this is mandatory]

Note: in this task you are required to use gprofng [4] (non-GUI version); to discover the options that you need to use, read the documentation on gprofng provided in the Virtual e-learning system.

Modify the Makefile to compile the code properly for profiling (comment CFLAGS=-Wall -O2 and uncomment CFLAGS=-Wall -O0 -g), regenerate the nbody-serial.exe executable (make clean; make) and run it through gprofng to capture the profiling data. Ensure numBodies=300 and numSteps=3000; with these parameters, the profiling should take approximately 33 seconds.

In you report, provide the command used to run nbody-serial.exe through gprofng.

After the profiling finishes, use again gprofng to display the calltree based on the collected data.

In you report, provide the command used to display the calltree, and also the calltree itself.

Analyze the calltree and identify the main hot-spots **(>=10% of the total time)**. In particular, verify if there are opportunities to improve performance either than resorting to parallelization. For instance, are there functions called, whose task could be implemented in a more efficient way ?

In you report, state which functions did you identify that represent important hot-spots and explain how would you change the nbody-serial.c code to improve performance still without parallelization.

d) Development, Benchmarking and Profiling of the non-GUI Serial Version 2
[this stage must be done on the specific cluster node assigned to your group; this is mandatory]

Copy nbody-serial.c to nbody-serial2.c, and change the Makefile to also compile the later. Then apply to nbody-serial2.c the changes you envisioned before, that could improve the performance. Next, benchmark nbody-serial2.c the same way you did in **b)** for nbody-serial.c. Very Important: you must make sure the output of the improved version matches the output of the original version !

In you report, provide the three nbody-serial2.exe execution times measured (only the real times).

Provide also the measure of the speedup achieved by nbody-serial2.exe against nbody-serial.exe.

Now you must profile the improved serial version nbody-serial2.exe, following the same procedure used to profile the original serial version nbody-serial.exe in **c)**.

In you report, provide the calltree generated by gprofng upon the profiling of nbody-serial2.exe.

Analyze the calltree and identify the main hot-spot functions **(>=10% of the total time)**. Then analyze their code and identify those that you think can be easily parallelizable.

In you report, state the functions you chose to parallelize and explain why they are easily parallelizable. Also, provide the aggregated weight (% of total execution time) of those functions.

e) Applying Amdahl's Law

Considering the aggregated weight (%) of the code that you chose to parallelize, use Amdahl's Law to calculate the theoretical speedup S_T of the parallelization for a number of threads N=2,...,8.

In your report, provide the next table with the missing values (?) of the S_T theoretical 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.

f) Parallelization

The code for the Pthreads version must be in the file nbody-threads.c . This file is to be initially created as a copy of nbody-serial2.c that will change progressively to produce the parallel version.

Modify the Makefile to ensure that, from now on, nbody-threads.c is compiled, and the flags for benchmarking are used (uncomment CFLAGS=-Wall -O2 and comment CFLAGS=-Wall -O0 -g). Also, pass to nbody-threads.exe the same parameters as in **b)**: numBodies=300, numSteps=30000.

f.1) Development

[this stage may be done on you personal computer or in the cluster frontend]

Decide how to divide the work (the hot-spot functions that you choose to parallelize) by threads and how to synchronize them (balancing the work uniformly by threads, and minimizing their synchronization points, will favor performance). Explain your parallelization strategy in the report.

Hint: start with 2 threads and compare the output of the simulation (nbody-threads-300-30000.txt) with that of the serial versions (nbody-serial-300-30000.txt), to ensure the results are the same, or at least very similar; to compare compare the outputs use the `diff` command or the Meld [5] program; only if the outputs are similar, should you increment the number of threads created by your code.

Note: with threads, the output may not match exactly the one from the serial version because of rounding errors introduced by a different order of the calculations that involve floating point values.

f.2) Benchmarking

[this stage must be done on the specific cluster node assigned to your group; this is mandatory]

When ready for benchmarking, run time ./nbody-threads.exe 300 30000 3 times for each different number of threads N=1,...,8 and take note of the smallest real time T_R for each value of N. In the 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)	? (*)	?	?	?	?	?	?	?

(*) this value should be similar to the value measured in b); if very different, that is a red flag !

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 (note: the reference value is the one from d)).

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.

g) 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 ?

h) Deadline and Submission

The report (PDF) and code (all zipped) should be submitted in Virtual until **November 23rd 2025**.

i) Plagiarism and Usage of AI Tools: If detected, Plagiarism or use of AI tools will dictate the work annulment. The report should not be written with the aid of AI tools, and the required code (2nd serial and parallel versions) should not be generated by AI tools. This is your work, not other's.

j) References

[1] https://en.wikipedia.org/wiki/N-body_simulation
[2] https://rosettacode.org/wiki/N-body_problem#C
[3] https://sdl-bgi.sourceforge.io/
[4] https://www.gnu.org/software/gprofng-gui/manual/gprofng.html
[5] https://meldmerge.org/