On the parallelization of a N-body simulation

ANDRÉ MENDES, Universidade do Minho, Portugal
BRUNO JARDIM, Universidade do Minho, Portugal
ROBERTO FIGUEIREDO, Universidade do Minho, Portugal

This report documents the parallelization techniques implemented on a N-body simulation. This was achieved through the OpenMP API. After carefully analyzing and tested different optimization locations, the resulting program had a xxx% speedup compared to the original, sequential implementation. This result shows that physical, N-body simulations are an excellent target for parallel code optimizations. As even very simple optimization techniques are able to produce a huge speedup.

Additional Key Words and Phrases: Parallel, Parallelization, Code Optimization, Simulation, Physics Simulation, OpenMP

ACM Reference Format:
André Mendes, Bruno Jardim, and Roberto Figueiredo. 2026. On the parallelization of a N-body simulation. 1, 1 (January 2026), 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

With the gradual decline of Moore's Law[1], the time of swapping an older chip for a newer one and experiencing exponential speedups is coming to an end. This, in a sense, limits the computing power a certain problem can have to still be tractable. That is, if we were dealing with a single computing core on a machine.

Fortunately, that is not the case. Nowadays, even the simplest computers have multiple computing cores. Unfortunately, this parallelism is often misused or even neglected, making the device, essentially, a single-core computer.

As such, this demands a narrative shift to parallelism as design principle. In fact, this is precisely the direction in which hardware is going, with more cores instead of faster ones, as we are reaching the physical limits of silicon. Thus, performance gains are not dependent on how fast a given CPU is, but how the parallelism of a given CPU can be exploited.

This issue becomes quite significant when it comes to N-body problems. These problems are notorious for their computational complexity, this is due to the pairwise calculations between every body in the system. As such, this requires a $O(N^2)$ time complexity, meaning that even small increases in particle count could dramatically increase the runtime of such an algorithm. In real-world scenarios, these applications require millions, if not billions, of particles in order to simulate any interesting phenomena. Thus making serial implementations of these applications impractical. This necessitates parallelization for these kinds of problems. However, parallelism isn't a solution without any drawbacks, there are a lot of considerations when parallelizing any algorithm. Such as the synchronization overheads, load imbalances, cache coherence effects, and even, memory bandwidth limitations. This article goes over the parallelization of a N-body simulation on multicore CPUs, focusing on memory behavior, scalability and algorithmic design choices.

Authors' Contact Information: André Mendes, andre@mail.com, Universidade do Minho, Braga, Portugal; Bruno Jardim, bruno.f.jardim@inesctec.pt, Universidade do Minho, Braga, Portugal; Roberto Figueiredo, roberto@mail.com, Universidade do Minho, Braga, Portugal.

ACM XXXX-XXXX/2026/1-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 Background

2.1 Overview of the N-body Problem

The N-body problem aims to simulate the time evolution of a given system with N particles, where all particles affect one another. Each time step necessitates the computation of the net forces that act on every single particle based on the states of all others and then updating their position and velocity. This results in an $O(N^2)$ complexity on every time step. Making the problem increasingly more expensive as N grows.

Due to this design the N-body problem is a well-suited candidate for parallelization. This is further reinforced by the fact that the force computations on individual particles are independent within that time step. However, most implementations on multicore CPUs are constrained by memory bottlenecks, cache coherence effects and even synchronization overheads. Thus, achieving linear speedups is not as trivial as it might seem.

Performance is heavily influenced by memory access patterns along with the in-memory representation of the given datastructures. As they can significantly improve cache utilization and throughput. All of these characteristics make the N-body problem a very useful benchmark for parallel performance on modern CPU architectures.

References

[1] G.E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.