

On the parallelization of a N-body simulation

ALICE, Universidade do Minho, Portugal

BOB, Universidade do Minho, Portugal

CHARLIE, 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:

Alice, Bob, and Charlie. 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: Alice, alice@mail.com, Universidade do Minho, Braga, Portugal; Bob, bob@mail.com, Universidade do Minho, Braga, Portugal; Charlie, charlie@mail.com, Universidade do Minho, Braga, Portugal.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM XXXX-XXXX/2026/1-ART

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

References

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