Benchmarking Generative AI in HPC

Can state of the art LLMs optimize N-Body C++ Simulations?

TECHNISCHE HOCHSCHULE DEGGENDORF

MAX PLANCK INSTITUTE FOR INFORMATICS

Dr. Markus Rampp, Rohan Sharma, Hardeepsinh Jadeja, Chaitanya Koneru, Sai Charan, Mayuri Patil

Introduction

Generative AI is transforming software engineering, but can it handle the rigorous demands of High-Performance Computing (HPC)? We evaluated the "innate intelligence" of state-of-the-art LLMs by tasking them to optimize an $O(N^2)$ C++ N-Body simulation.

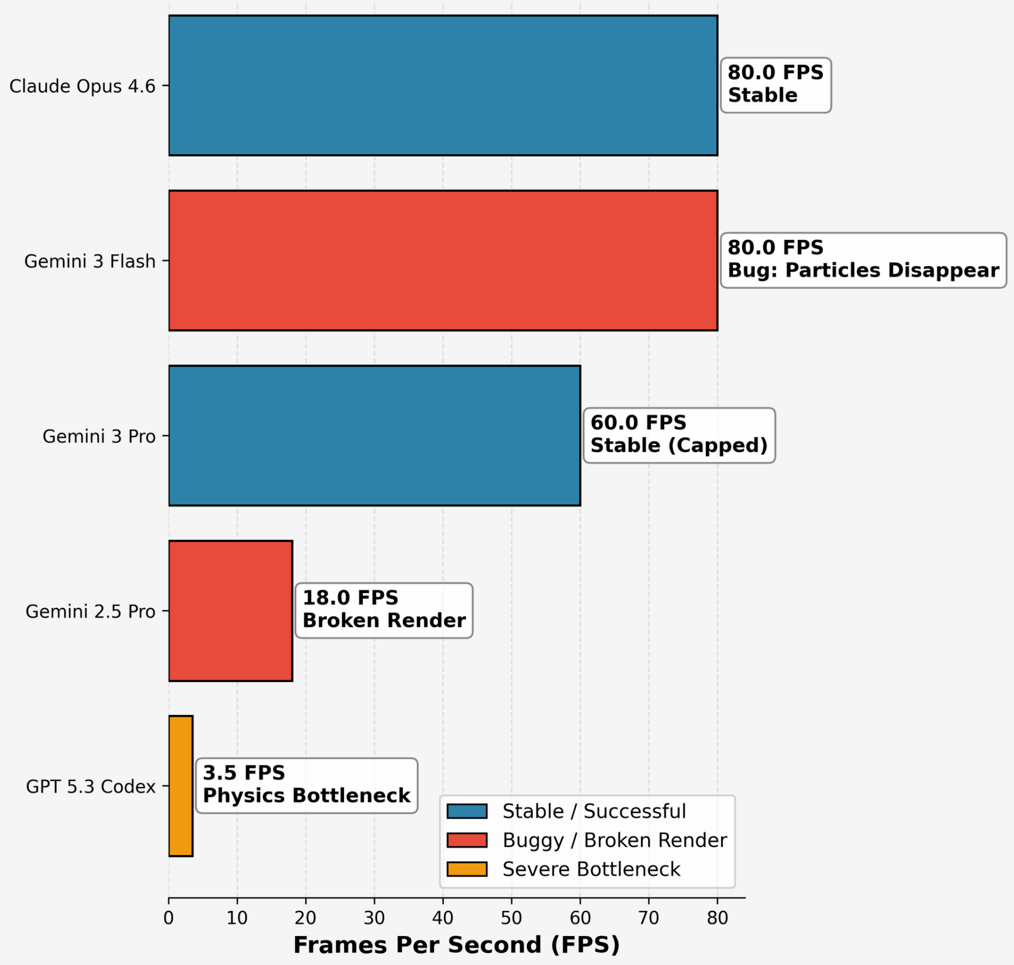
Rather than explicitly requesting specific technologies (e.g., SIMD, Vertex Arrays), we designed prompts that described system bottlenecks and computational goals. This approach reveals whether models truly "understand" industry-standard HPC solutions or merely regurgitate code, testing their ability to autonomously apply optimizations.

Methodology

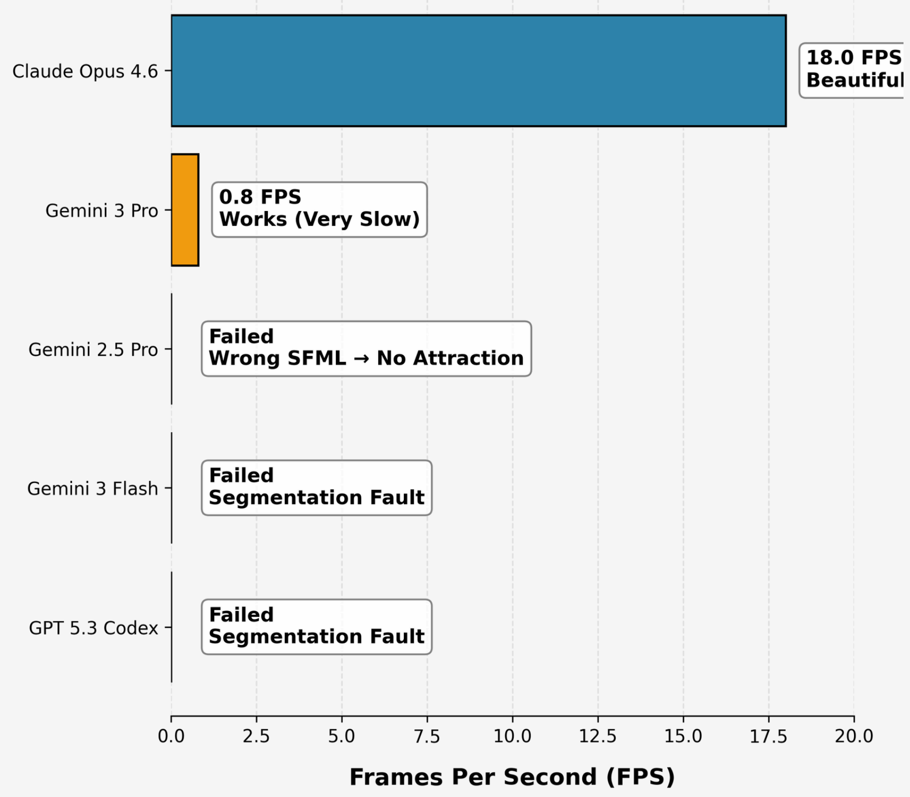
Prompt 1: The Foundation
↓
Prompt 2: Physical Accuracy
↓
Prompt 3: Performance Profiling
↓
Prompt 4: Shared-Memory Parallelism
↓
Prompt 5: Memory Architecture Optimization
↓
Prompt 6: Extreme Mathematical Optimization
↓
Prompt 7: Galaxy Initial Conditions
↓
Prompt 8: High-Efficiency Rendering
↓
Prompt 9: Visual Aesthetics and Dynamics
↓
Prompt 10: Production-Grade Refactoring

Results & Analysis

Multi-Prompt Strategy



Single-Prompt Strategy



Key Insights

- Memory Architecture: Most LLMs lack innate architectural foresight. They fail to spontaneously refactor from Array of Structures (AoS) to Structure of Arrays (SoA) unless explicitly prompted about CPU memory controllers and cache utilization.
- Physical Accuracy: Advanced models demonstrated strong domain knowledge, autonomously suggesting "softening" (adding a small epsilon constant to the gravity formula) to prevent NaN errors during close particle encounters.
- Brute Force vs. Elegance: When forced to optimize in a single prompt, most models attempt to parallelize a bad algorithm. Only Claude Opus 4.6 realized an algorithmic shift (Barnes-Hut) was superior to micro-optimizations.
- Unlike models that crashed or defaulted to brute-force $O(N^2)$ physics, Claude Opus 4.6 autonomously implemented an elegant Barnes-Hut spatial partitioning tree ($O(N \log N)$) from a single prompt. The simulation (below) successfully handles 10,000 particles and features an LLM-generated quadtree visualization alongside a custom, highly detailed performance profiling overlay.

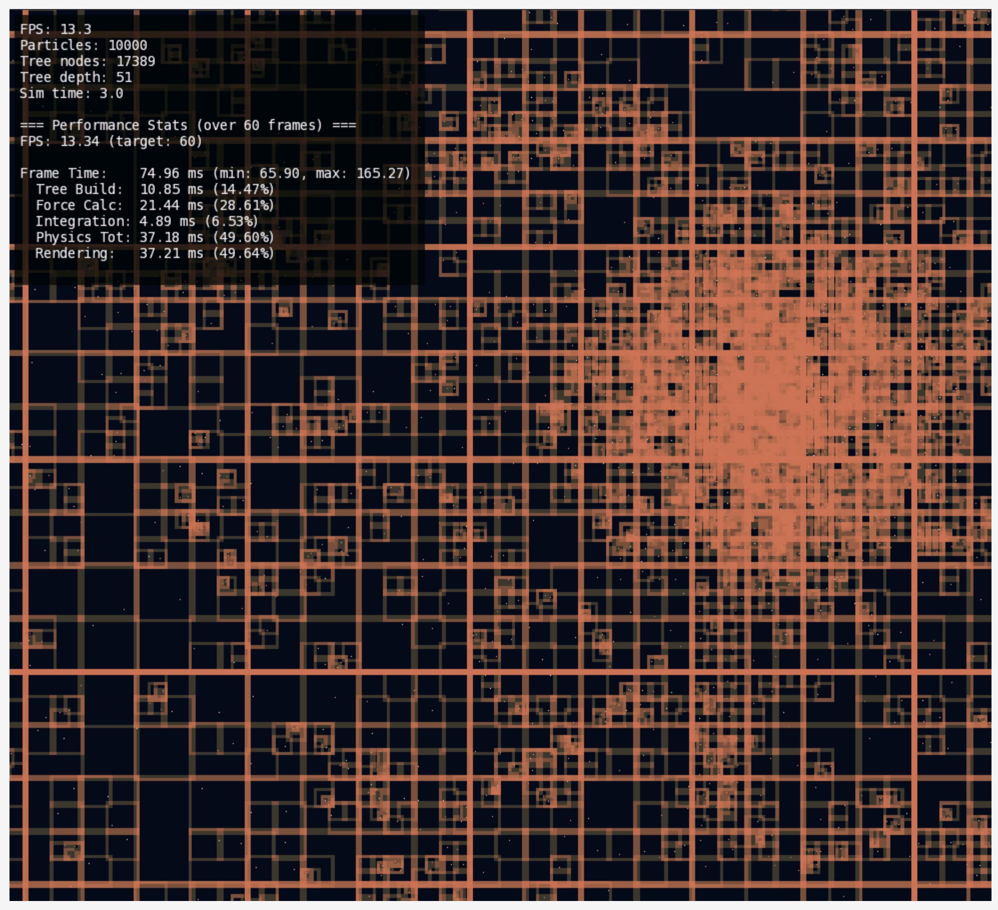


Fig. 1: Zero-Shot Algorithmic Leap.

Challenges

- LLM Hallucinations: Models frequently invent non-existent C++ libraries or incorrect SIMD/AVX intrinsics when pushed for extreme optimization without step-by-step guidance.
- Lack of Global Foresight: Models optimize locally but rarely plan holistic data layouts for cache locality early in the generation cycle.

Conclusion

State-of-the-art LLMs possess deep latent knowledge of High-Performance Computing paradigms, but their innate application of this knowledge is fundamentally flawed. They succeed not by being told how to optimize, but by being iteratively guided through what is bottlenecking the system. Multi-prompting transforms LLMs from naive code generators into highly capable HPC optimization assistants, though models still struggle with context degradation, system-level asset management, and identifying when an algorithmic shift is required over hardware-level micro-optimizations.

Full references and codebase available on :