Characterizing the Performance of the Implicit Massively Parallel Particle-in-Cell iPIC3D Code

Jeremy J. Williams, Daniel Medeiros, Ivy B. Peng, and Stefano Markidis
Department of Computer Science, EECS, KTH Royal Institute of Technology
Stockholm, Sweden
jjwil,dadm,bopeng,markidis@kth.se

Abstract

Optimizing iPIC3D, an implicit Particle-in-Cell (PIC) code, for large-scale 3D plasma simulations is crucial for space and astrophysical applications. This work focuses on characterizing iPIC3D's communication efficiency through strategic measures like optimal node placement, communication and computation overlap, and load balancing. Profiling and tracing tools are employed to analyze iPIC3D's communication efficiency and provide practical recommendations. Implementing optimized communication protocols addresses the Geospace Environmental Modeling (GEM) magnetic reconnection challenges in plasma physics with more precise simulations. This approach captures the complexities of 3D plasma simulations, particularly in magnetic reconnection, advancing space and astrophysical research.

Keywords: iPIC3D, Magnetic Reconnection, Implicit PIC, Space Weather, Performance Analysis, Profiling and Tracing

ACM Reference Format:
Jeremy J. Williams, Daniel Medeiros, Ivy B. Peng, and Stefano Markidis. 2023. Characterizing the Performance of the Implicit Massively Parallel Particle-in-Cell iPIC3D Code. In *Proceedings of ACM Conference (SC '23).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Introduction. The iPIC3D code is widely used massively-parallel Particle-in-Cell (PIC) for space simulations, particularly in magnetic reconnection studies. Magnetic reconnection is a phenomenon where magnetic field lines rupture and rearrange in three-dimensional space, leading to a restructuring of plasma's magnetic topology. This process converts magnetic energy into kinetic and thermal energy, accelerating charged particles and generating intense energy bursts [3].

SC '23, Tuesday–Thursday, November 14–16, 2023, Denver, CO, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

As an implicit Particle-in-Cell (iPIC) code, iPIC3D is a powerful tool for 3D plasma simulations, allowing for in-depth space exploration of plasma dynamics and complex interactions between electromagnetic fields and charged particles. It provides valuable insights into the conversion activity of magnetic energy (Figure 1) and the acceleration of charged particles, shedding light on complex underlying processes in iPIC3D Plasma Simulations [3].

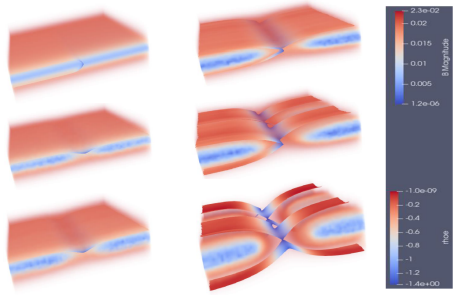

Figure 1. Evolution of streamline magnetic fields in magnetic reconnection with iPIC3D.

This work aims to comprehensively analyze and enhance the performance of iPIC3D by exploring metrics like execution time, memory usage, and computational efficiency in its simulations. Advanced performance analysis tools, including profilers and tracers [6], are employed to examine iPIC3D's performance characteristics, identify hotspots, and gain insights into its behavior in plasma simulations. Optimizations based on these findings seeks to improve iPIC3D's effectiveness and deepen our understanding of plasma dynamics.

Methodology & Experimental Setup. In this work, we use `perf` as a profiler to collect hardware performance counters, focusing on cache and memory performance. We also leverage `CrayPAT` and `Apprentice2` for parallel data processing visualization on Cray architectures [1]. We incorporate `Extrae` and `Paraver`, from the Barcelona Supercomputing Center (BSC), into our workflow for parallel performance tracing and profiling [4]. Finally, we also display our obtained results with `Darshan`, a performance monitoring tool for analyzing serial and parallel I/O workloads [5].

In our experiments, we analyze iPIC3D on two systems: a workstation (`Greendog`) with an i7-7820X processor (8 cores) and `Dardel`, a HPE Cray EX supercomputer with 1270 nodes,

each equipped with 256GB DRAM and two AMD EPYC Zen2 2.25 GHz 64-core processors per node. Further details about these systems are provided in the poster.

Results and Analysis. iPIC3D uses parallel processing within a single node and inter-node communication for larger-scale simulations across multiple nodes. This section analyzes intra-node and inter-node activity of iPIC3D, focusing on up to 32 nodes (4096 MPI Processes) and evaluating its I/O performance.

We begin by assessing iPIC3D's performance and memory system. We used `perf` on the Greendog workstation, leveraging administrative privileges (without restrictions). The impact of the iPIC3D cache size on load misses varies with cache level and workload. Larger cache sizes result in a reduction of L1 Ddcache load misses. For example, 50% Increase Size (6 6 6 6) has 1.99% load misses, Baseline Size (4 4 4 4) has 2.22% load misses, and 50% Reduction Size (2 2 2 2) has 3.79% load misses. Similarly, for LLC load misses, 50% Increase Size has 54.75%, Baseline Size has 58.03%, and 50% Reduction Size has 47.95%.

Next, we employed `Extrae` and `Paraver` on Dardel to analyze and trace iPIC3D's communication pattern. This analysis focused on 8 MPI ranks, with one simulation cycle (`ncycles = 1`) and all other parameters fixed, providing clear communication pattern results for a complete simulation of iPIC3D. During the initial phase to around 50% of the simulation, the iPIC3D MPI ranks remain stable. However, ranks 1 to 7 experience a waiting period for rank 0, leading to workload imbalance and impacting efficiency in each node.

We then analyze iPIC3D parallel performance on Dardel, up to 32 nodes (4096 MPI ranks), using `CrayPAT` for instrumentation and `Apprentice2` for visualizing and exploring performance analysis data. This allows us to gain insights into the performance characteristics of multiple simulations. In the intra-node configuration (128 MPI processes), iPIC3D's computational functions are predominantly utilized, accounting for approximately 74.84% of the execution time. On the other hand, in the inter-node configuration (4096 MPI processes), a higher proportion of MPI functions is observed, due to increased inter-node communication demands. As seen in Figure 2, increasing MPI processes in iPIC3D results in higher communication time, reaching 86.90% with 4096 MPI processes, while computation time decreases to 13.10% due to overhead. Distributing computation improves overall execution time, but ideal linear scaling is not achieved, resulting in a slower execution due to communication overhead, load imbalance, and scalability limitations.

Finally, using `Darshan`, we analyze I/O in iPIC3D plasma simulations. The POSIX interface demonstrates higher bandwidth with more MPI processes, indicating better parallel performance. With 128 MPI processes, the POSIX interface achieves an I/O bandwidth of 212.28 MiB/s, while STDIO only manages 1.07 MiB/s. Increasing the number of MPI processes to 4096, the POSIX interface delivers an impressive

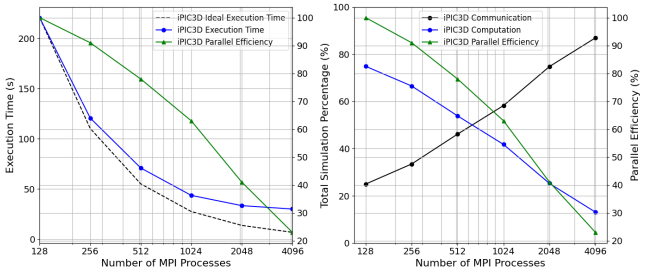

Figure 2. iPIC3D's strong scaling up to 4096 MPI processes.

I/O bandwidth of 4364.49 MiB/s, whereas STDIO lags behind with only 8.27 MiB/s. These results highlight how iPIC3D's POSIX interface outperforms STDIO in I/O bandwidth, even with higher parallelism. Utilizing POSIX can significantly enhance I/O performance, especially in data transfer rates.

Conclusions and Future Work. In this work, we identified communication as a critical factor and bottleneck impacting the performance in iPIC3D, particularly on large runs. Non-blocking MPI communication functions are used to mitigate this issue, however the presence of `MPI_Waitall` can hinder execution and slow down progress. Additionally, file I/O operations (POSIX and logging) contribute to performance overhead.

To enhance communication efficiency in iPIC3D plasma simulations, we propose several strategies. These include optimal node placement [7], communication and computation overlap [2], and load balancing [8]. Additionally, we suggest exploring alternative algorithms and data structures to minimize overhead, drawing insights from further advanced tooling techniques [6].

References

[1] Reuben Budiardja and et al. 2018. *Using CAASCADE and CrayPAT for analysis of HPC applications.* Technical Report. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States).

[2] Vladimir Marjanović and et al. 2010. Overlapping communication and computation by using a hybrid MPI/SMPSs approach. In *Proceedings of the 24th acm International Conference on Supercomputing*. 5–16.

[3] Stefano Markidis and et al. 2010. Multi-scale simulations of plasma with iPIC3D. *Mathematics and Computers in Simulation* 80, 7 (2010), 1509 – 1519. https://doi.org/10.1016/j.matcom.2009.08.038

[4] Harald Servat and et al. 2013. Framework for a productive performance optimization. 39, 8 (2013), 336–353.

[5] Shane Snyder and et al. 2016. Modular HPC I/O characterization with Darshan. In *2016 5th workshop on extreme-scale programming tools (ESPT)*. IEEE, 9–17.

[6] Jeremy J Williams and et al. 2023. Leveraging HPC Profiling & Tracing Tools to Understand the Performance of Particle-in-Cell Monte Carlo Simulations. *arXiv preprint arXiv:2306.16512* (2023).

[7] Mohamed Younis and et al. 2008. Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Networks* 6, 4 (2008), 621–655.

[8] Qi Zhang and et al. 2005. Workload-aware load balancing for clustered web servers. *IEEE Transactions on Parallel and Distributed Systems* 16, 3 (2005), 219–233.