MPI Performance Analysis in Vlasiator: Unraveling Communication Bottlenecks

Jennifer Faj, Jeremy J. Williams, Ivy B. Peng, Stefano Markidis
Department of Computer Science, KTH Royal Institute of Technology
Stockholm, Sweden
faj,jjwil,bopeng,markidis@kth.se

Urs Ganse, Markus Battarbee, Yann Pfau-Kempf, Leo Kotipalo, Minna Palmroth
Department of Physics, University of Helsinki
Helsinki, Finland
urs.ganse,markus.battarbee@helsinki.fi
yann.kempf,leo.kotipalo,minna.palmroth@helsinki.fi

Abstract

Vlasiator is a popular and powerful massively parallel code for accurate magnetospheric and solar wind plasma simulations. This work provides an in-depth analysis of Vlasiator, focusing on MPI performance using the Integrated Performance Monitoring (IPM) tool. We show that MPI non-blocking point-to-point communication accounts for most of the communication time. The communication topology shows a large number of MPI messages exchanging data in a six-dimensional grid. We also show that relatively large messages are used in MPI communication, reaching up to 256MB. As a communication-bound application, we found that using OpenMP in Vlasiator is critical for eliminating intra-node communication. Our results provide important insights for optimizing Vlasiator for the upcoming Exascale machines.

Keywords: Vlasiator, Performance Analysis, Profiling, MPI

ACM Reference Format:
Jennifer Faj, Jeremy J. Williams, Ivy B. Peng, Stefano Markidis and Urs Ganse, Markus Battarbee, Yann Pfau-Kempf, Leo Kotipalo, Minna Palmroth. 2023. MPI Performance Analysis in Vlasiator: Unraveling Communication Bottlenecks. In *Proceedings of ACM Conference (SC '23).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

Introduction. Large-scale plasma simulations are critical for understanding plasma dynamics in various plasma environments, from fusion devices and accelerators to space and astrophysical systems. Vlasiator [4] is one of the most powerful and established parallel C++ codes for highly accurate simulations of magnetospheric and solar wind plasmas with application to space weather. The code uses a hybrid approach for the solution of plasma dynamics: a direct Vlasov solver to solve protons' kinetic dynamics (requiring to solve the Vlasov equation for the distribution function) and an equation of state that assumes electrons behaving like a fluid [5]. Since the proton distribution function depends on the position and velocity spaces (the so-called phase-space) in a 3D Vlasiator simulation, we need a six-dimensional simulation grid to solve the Vlasov equations for the protons. The code is parallelized using hybrid MPI and OpenMP for inter-node and intra-node communication. Additionally, adaptivity in the discretization of the velocity space is required, leading to load imbalance. Therefore, libraries are utilized to implement adaptive mesh refinements (`DCCRG` [3]) and to improve the load balance across MPI processes (`Zoltan` [1]). The solution of a partial differential equation with load-balancing techniques is a formidable task, requiring advanced parallelization techniques, including efficient data distribution, communication, and memory utilization. This work aims to provide insights into communication and the impact of load imbalance to identify potential optimization techniques. To achieve this, we perform an in-depth analysis of the profiling results of the Vlasiator code using the IPM tool [2, 7].

Experiments and Analysis. The experiments are executed on the CPU nodes of the Dardel supercomputer. Each node comprises two AMD EPYC 7742 CPUs, for a total of 128 physical cores and 256 GB memory. We perform a Vlasiator test simulation, modeling a magnetosphere with two grid sizes, namely $\sim 8 * 10^4$ and $\sim 8 * 10^5$ grid cells. The simulated time is set to model 3.1 seconds, corresponding to about 40 time steps. To investigate MPI performance and understand the impact of OpenMP, multi-node experiments are performed scaling up to 16 nodes with *(i)* a varying number of MPI ranks with a fixed number of OpenMP threads per rank, and *(ii)* a fixed number of MPI ranks and a varying number of OpenMP threads per rank. Table 1 shows the percentage of communication time spent with the best configuration in both of the aforementioned experiments, with the same number of cores (1024) distributed differently between MPI ranks and OpenMP threads, as well as an additional test with a larger data set and 2048 cores. We note that MPI communication time takes between 48% and 82% of the total time, concluding Vlasiator as a communication-bound application.

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

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

Table 1. IPM report of selected experiments.

Grid Size	AMR cells	Nodes	MPI	OMP	%Comm
85184	13308	8	512	2	82%
		8	64	16	51%
830584	137696	16	128	16	48%

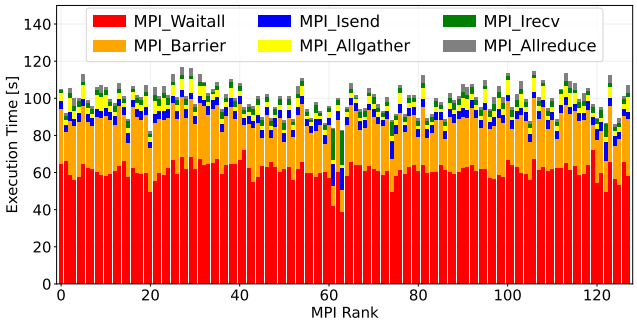

Figure 1. Vlasiator breakdown of MPI calls.

This means that an optimization of the communication part leads to a major improvement of the overall Vlasiator performance, requiring a focus on communication for future improvements. We also note that OpenMP usage is critical to eliminate intra-node communication and improve the overall parallel speed-up. Figure 1 shows the most used MPI functions and the aggregated time spent in them. Approximately 86% of MPI function time is spent in two MPI functions: `MPI_Waitall` and `MPI_Barrier` calls. These two calls are synchronizing MPI calls revealing that a large amount is spent by MPI processes waiting for the slowest processes due to load imbalances. When looking at communication patterns and message sizes, Figure 2 shows the communication topology (which MPI ranks communicate to each MPI rank) with the aggregated message size. The number of messages in a six-dimensional grid and the aggregated message size are relatively high. A closer look in Figure 3 shows the distribution of message size relatively to the number of calls and time spent in the functions. We observe a wide range of message sizes, ranging from a few bytes to 256MB, and a significant amount of time being spent in exchanging messages between 16MB-64MB.

Conclusions. In this work, we analyzed the MPI performance of the plasma simulation code Vlasiator, using the low-overhead IPM tool. We showed that Vlasiator is a communication-bound application, with communication percentages varying from 48% to 82% of the total time. We found that OpenMP is critical to eliminate intra-node communication and improve the overall performance of the code. MPI non-blocking point-to-point communication accounts for most of the Vlasiator communication. In particular, the time spent in the `MPI_Waitall` non-blocking point-to-point synchronization function is the largest, with the collective `MPI_Barrier`

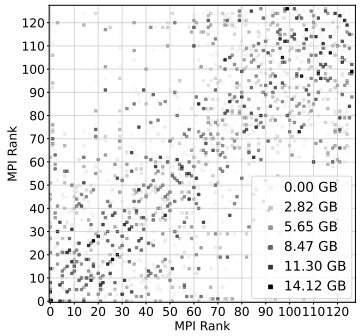

Figure 2. Vlasiator communication topology.

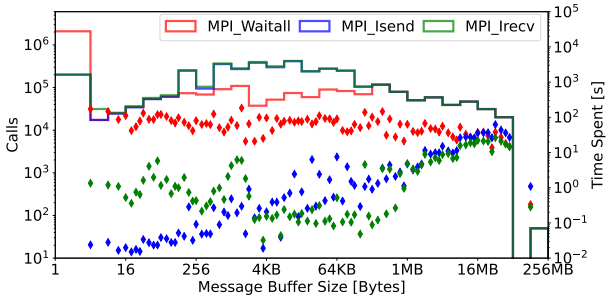

Figure 3. Breakdown of calls (histogram) and time spent (dots) with different MPI message buffer sizes.

also playing a non-negligible part. We note that the costs for these functions are associated with synchronization and process imbalance costs [6]. The Vlasiator communication topology shows several MPI messages of relatively large size, up to 256MB. Possible Vlasiator optimizations require an in-depth analysis of load imbalance sources causing the current Vlasiator simulation to wait at synchronization points. Additional promising optimizations include an increased integration of OpenMP and MPI.

References

[1] Umit V Catalyurek and et al. 2007. Hypergraph-based dynamic load balancing for adaptive scientific computations. In *IPDPS*. IEEE, 11.
[2] Karl Fuerlinger and et al. 2010. Effective performance measurement at petascale using ipm. In *2010 IEEE IPDPS*. IEEE, 373–380.
[3] Ilja Honkonen and et al. 2013. Parallel grid library for rapid and flexible simulation development. *Computer Physics Communications* 184, 4 (2013), 1297–1309.
[4] Minna Palmroth. 2022. Daring to think of the impossible: The story of vlasiator. *Frontiers in Astronomy and Space Sciences* 9 (2022), 210.
[5] Minna Palmroth and et al. 2018. Vlasov methods in space physics and astrophysics. *Living Reviews in Computational Astrophysics* 4, 1 (2018).
[6] Ivy Peng and et al. 2015. The cost of synchronizing imbalanced processes in message passing systems. In *2015 IEEE Cluster*. IEEE, 408.
[7] 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).