# A Toolbox for Load Balancing Development and Analysis in WarpX/AMReX Applications

Jessica Imlau Dagostini
University of California, Santa Cruz
Santa Cruz, California, USA
jessica.dagostini@ucsc.edu

Sowmya Yellapragada
University of Utah
Salt Lake City, Utah, USA
jessica.dagostini@ucsc.edu

Kevin Gott
Lawrence Berkeley National Laboratory, NERSC
Berkeley, California, USA
kngott@lbl.gov

Rebecca Hartman-Baker
Lawrence Berkeley National Laboratory, NERSC
Berkeley, California, USA
rjhartmanbaker@lbl.gov

## Abstract

Efficient load balancing is critical for the scalability of distributed scientific applications. However, there are several challenges for applications to test new balancing strategies, including the need for an easy workflow to validate different algorithms. This work aims to tackle this particular challenge by presenting a toolbox designed to streamline the collection and analysis of load balancing data from *WarpX*, an advanced, fully kinetic particle-in-cell (PIC) code, based on the *AMReX* framework. Our toolbox simplifies the extraction of data from *WarpX* and enables developers to conduct statistical load balancing inferences over real data efficiently. We demonstrate such applicability by conducting a study with a laser-ion acceleration simulation: we collect simulation data and compare six different load balancing approaches, two in-production algorithms, and four under investigation for future use.

## Keywords

Load Balancing, WarpX, AMReX, Distributed Computing, algorithms, software tools

**ACM Reference Format:**
Jessica Imlau Dagostini, Sowmya Yellapragada, Kevin Gott, and Rebecca Hartman-Baker. 2025. A Toolbox for Load Balancing Development and Analysis in WarpX/AMReX Applications. In *Proceedings of SC25*. ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 Introduction

Efficiently balancing workloads is crucial for scaling scientific applications. While achieving maximum hardware utilization is key, testing new load balancing strategies is often complex due to challenges in collecting and analyzing performance data. This work aims to tackle this particular workflow challenge by presenting a toolbox designed to streamline the collection and analysis of load balancing data from WarpX, an advanced, fully kinetic particle-in-cell (PIC) code [2]. This code is based on AMReX, a block-structured adaptive mesh refinement (AMR) software framework primarily used for solving partial differential equations [8]. As part of its functionalities, AMReX offers load balancing algorithms to efficiently distribute AMR grids/boxes to different threads or GPUs based on their computational costs, improving software scalability.

## 2 Toolbox Overview

Our toolbox leverages one of WarpX's "Reduced Diagnostics" to collect load balancing costs. These costs can be either embedded timers or a "heuristic" (a weighted counting of meshes and particles in each box). After defining the input parameters, the toolbox runs a simulation, records costs throughout execution, and parses these data into a CSV file. With the parsed balancing costs, users can run different load balancing algorithms through our toolkit to understand their applicability better and make informed decisions about optimal parallelization strategies.

Currently, this toolbox supports strategies in two main categories: partitioning and communication-approximation algorithms. There are two partitioning algorithms that distribute AMR boxes to workers based solely on each box's cost:

- Knapsack: a greedy algorithm that first orders the boxes by weight, and then assigns the next heaviest box to the lightest rank. AMReX/WarpX supports this strategy [6].
- Karmarkar-Karp [3] (a.k.a. least difference): repeatedly takes the two largest numbers/tuples from a set, replaces them with their difference, and sorts the list. The tuples are formed by repeatedly assigning the largest number to the subset with the smallest sum to minimize the total difference.

*Communication-approximation* algorithms distribute AMR boxes to workers based on their location on the grid, grouping proximate boxes to reduce communication costs. Two such solutions are implemented:

- Space Filling Curve (SFC): Also supported by AMReX/WarpX [6] as the Z-order SFC, this strategy orders the boxes by their proximity in the grid and distributes them by a proportional allowable weight (average). We extend this implementation by adding a Hilbert SFC strategy.
- Painter's Partition: using a similar ordering approach to SFC, this method employs a binary search to find the most efficient partitioning of work. The algorithm searches for a minimum


*SC25, St Louis, MO*

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

"target weight" (or maximum time). If a partition is possible within the target weight, it reduces the target to find an even faster solution; otherwise, it increases the target [5].

The toolbox also includes strategies from a previous study on AMReX load balancing, combining communication-approximation with partitioning [5]. There are two hybrid implementations, pairing the Z-Order SFC algorithm with Knapsack and the Painter's solution with Knapsack. Both use the SFC algorithm for inter-node communication and Knapsack to efficiently distribute the workload locally on each node. We also implement a Painter's solution using the Hilbert SFC strategy.

To demonstrate the applicability of our toolbox, a "Laser Ion Acceleration with a Planar Target" simulation [1, 4, 7] was run and used as a case study. Two different layouts were used, both with a maximum box size of 512. The first domain (2688 x 3712, 48 boxes) was tested on 4 and 8 GPUs, while the second (7488 x 14720, 435 boxes) was run on 24, 48, and 96 GPUs. Both layouts evolved for 5000 steps, with load balancing metrics collected every 100 steps.

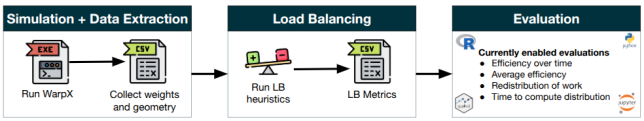


**Figure 1: Workflow applied at our LB toolbox.**

Figure 1 shows the toolbox's workflow. First, simulation characteristics are defined, and load balancing costs are collected. The toolbox then runs the eight strategies using the parsed costs and generates load balancing metrics, which can be analyzed using any state-of-the-art statistical tools. Figure 2 demonstrates the load balancing efficiency over the simulation's execution for each of the six solutions. This is an initial example of a comprehensive overview of the balancing algorithms that our toolbox allows.

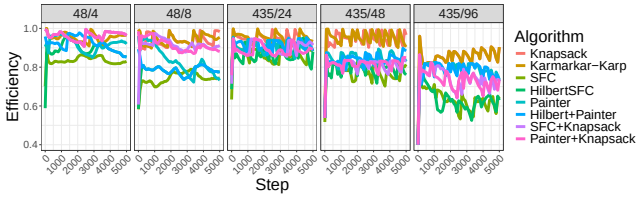


**Figure 2: Load balancing efficiency over the execution of the simulation. Each step represents a phase in the original simulation where load balancing costs were collected.**

From these results, it is clear that some algorithms, like Knapsack and Karmarkar-Karp, have more peak regions than others. The pure SFC solution also shows high peak values. These peaks represent redistributions of work to improve efficiency by at least 10%. The toolbox allows users to easily study how different minimum efficiency values impact redistribution frequency for each algorithm. Using the collected cost data, users can analyze how often each algorithm redistributes work under various efficiency ratios. Figure 3 illustrates these results.

As the percentage to redistribute increases, fewer redistributions occur for almost all scenarios. A follow-up analysis could correlate the redistribution of work with the algorithm's efficiency to

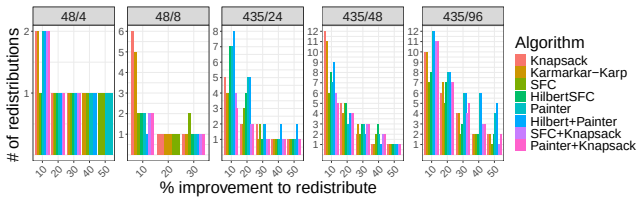


**Figure 3: Bar plot demonstrating how many redistributions can happen with various efficiency ratios.**

identify potential improvements to the application's performance. This is easily addressed in our toolbox, where users can plot such a correlation as depicted by Figure 4. In this figure, points closer to 1 on the x- and y-axes are better. Both partitioning algorithms present points in the best quadrant for most of the experiments, possibly indicating that they offer reasonable balancing solutions.

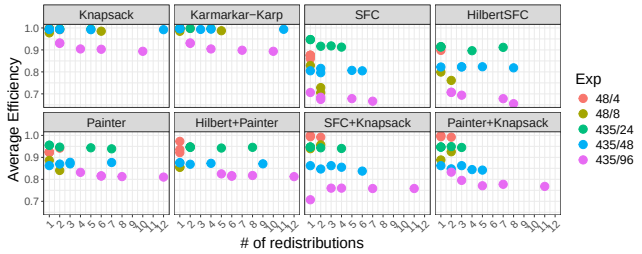


**Figure 4: Scatter plot demonstrating the correlation between the algorithm's efficiency and redistributions of work. Points closer to 1 in both x- and y-axis are better.**

Finally, our toolbox also leverages the analysis of the total execution time of each load balancing algorithm to understand their impact on the overall runtime. These results can be easily plotted and analyzed, as shown in Figure 5.

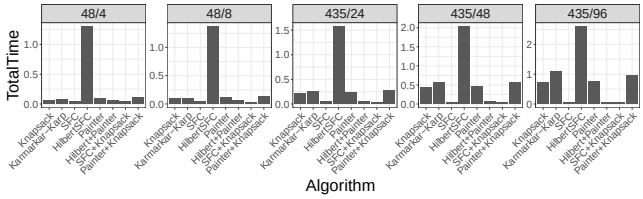


**Figure 5: Bar plot comparing the total execution time spent by each algorithm in the whole execution.**

The Hilbert SFC strategy demonstrates a considerably higher cost than the other solutions in all scenarios. This discrepancy highlights a key area for further investigation, which the toolbox can easily facilitate.

With that, we demonstrate the applicability of our toolbox and how it enables users to focus solely on the load balancing exploration of AMR-based simulations. For future work, we aim to extend these tools to other AMReX-based applications. We also plan to include memory constraints in the algorithms, considering memory limits in each worker when assigning new workloads. Finally, we also plan on using this toolbox to explore energy-efficient load balancing strategies.

## Acknowledgments

The Computing Sciences Summer Program from NERSC and LBL supported this work. Special thanks to Kevin Gott, Rebecca Hartman-Baker, and Sowmya Yellapragada.

## References

[1] S. S. Bulanov, A. Brantov, V. Yu. Bychenkov, V. Chvykov, G. Kalinchenko, T. Matsuoka, P. Rousseau, S. Reed, V. Yanovsky, D. W. Litzenberg, K. Krushelnick, and A. Maksimchuk. 2008. Accelerating monoenergetic protons from ultrathin foils by flat-top laser pulses in the directed-Coulomb-explosion regime. *Phys. Rev. E* 78 (Aug 2008), 026412. Issue 2. doi:10.1103/PhysRevE.78.026412

[2] Luca Fedeli, Axel Huebl, France Boillod-Cerneux, Thomas Clark, Kevin Gott, Conrad Hillairet, Stephan Jaure, Adrien Leblanc, Rémi Lehe, Andrew Myers, Christelle Piechurski, Mitsuhisa Sato, Neïl Zaim, Weiqun Zhang, Jean-Luc Vay, and Henri Vincenti. 2022. Pushing the Frontier in the Design of Laser-Based Electron Accelerators with Groundbreaking Mesh-Refined Particle-In-Cell Simulations on Exascale-Class Supercomputers. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12. doi:10.1109/SC41404.2022.00008

[3] Narendra Karmarkar and Richard M Karp. 1982. *The differencing method of set partitioning*. Computer Science Division (EECS), University of California Berkeley.

[4] Andrea Macchi, Marco Borghesi, and Matteo Passoni. 2013. Ion acceleration by superintense laser-plasma interaction. *Rev. Mod. Phys.* 85 (May 2013), 751–793. Issue 2. doi:10.1103/RevModPhys.85.751

[5] Amitash Nanda, Md Kamal Hossain Chowdhury, Hannah Ross, and Kevin Gott. 2025. Exploring Dynamic Load Balancing Algorithms for Block-Structured Mesh-and-Particle Simulations in AMReX. In *Practice and Experience in Advanced Research Computing 2025: The Power of Collaboration (PEARC '25)*. Association for Computing Machinery, New York, NY, USA, Article 5, 9 pages. doi:10.1145/3708035.3736022

[6] Michael E. Rowan, Kevin N. Gott, Jack Deslippe, Axel Huebl, Maxence Thévenet, Remi Lehe, and Jean-Luc Vay. 2021. In-situ assessment of device-side compute work for dynamic load balancing in a GPU-accelerated PIC code. In *Proceedings of the Platform for Advanced Scientific Computing Conference* (Geneva, Switzerland) *(PASC '21)*. Association for Computing Machinery, New York, NY, USA, Article 10, 11 pages. doi:10.1145/3468267.3470614

[7] SC Wilks, AB Langdon, Thomas E Cowan, Markus Roth, Malay Singh, Stephen Hatchett, MH Key, D Pennington, Andrew Mackinnon, and RA Snavely. 2001. Energetic proton generation in ultra-intense laser–solid interactions. *Physics of plasmas* 8, 2 (2001), 542–549.

[8] Weiqun Zhang, Ann Almgren, Vince Beckner, John Bell, Johannes Blaschke, Cy Chan, Marcus Day, Brian Friesen, Kevin Gott, Daniel Graves, Max Katz, Andrew Myers, Tan Nguyen, Andrew Nonaka, Michele Rosso, Samuel Williams, and Michael Zingale. 2019. AMReX: a framework for block-structured adaptive mesh refinement. *Journal of Open Source Software* 4, 37 (May 2019), 1370. doi:10.21105/joss.01370

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009