Heterogeneity-Aware Task Allocation for Modern HPC Systems

SOWMYA YELLAPRAGADA, University of Utah, USA
JESSICA IMLAU DAGOSTINI, University of California, USA
KEVIN GOTT, Lawrence Berkeley National Laboratory, USA
REBECCA HARTMAN-BAKER, Lawrence Berkeley National Laboratory, USA

Modern supercomputing systems exhibit heterogeneous node configurations, where even seemingly identical hardware exhibits significant performance variations due to memory capacity differences, manufacturing tolerances, and deployment conditions. This heterogeneity impacts the efficiency of scientific applications built on frameworks like AMReX, leading to substantial computational waste on leadership-class systems. We present Performance-Aware and Relation-Aware load balancing algorithms specifically designed for scientific applications, like AMReX on heterogeneous HPC clusters. Our approach uses empirically measured node performance characteristics and a Relative Performance Matrix (R_{ij}) to optimize task distribution across diverse computational resources. Evaluation of NERSC Perlmutter with 14 representative AMReX computational kernels demonstrates 99.9% scheduling efficiency, achieving performance improvements of 4.4-11.5% over traditional methods in moderate heterogeneity scenarios (A100 40GB vs 80GB) and up to 300x improvements in extreme CPU-GPU mixed configurations where homogeneous methods fail to effectively utilize CPU resources. The algorithms handle million-task workloads with $O(nlogn + nm)$ complexity while maintaining practical deployment feasibility.

1 Introduction

Modern high-performance computing environments can exhibit performance heterogeneity that challenges traditional homogeneous scheduling assumptions. In distributed systems deployments across academic clusters, cloud platforms, and specialized testbeds, computational nodes demonstrate substantial performance variations even when utilizing seemingly identical hardware configurations[1]. This heterogeneity manifests in two primary forms: moderate variations of 25-30% among identical hardware due to manufacturing tolerances, memory capacity differences, and deployment conditions, and extreme disparities of 430x in mixed CPU-GPU configurations (as observed on NERSC Perlmutter).

The heterogeneity problem becomes particularly sensitive in scientific computing frameworks like AMReX[2], where traditional load balancing algorithms assume uniform node capabilities and distribute tasks equally without considering specific performance characteristics. This approach leads to significant computational waste, as faster nodes remain idle while slower nodes become bottlenecks, resulting in suboptimal makespan and reduced overall system efficiency. On leadership-class supercomputing systems where computational resources are extremely valuable, such inefficiencies translate directly into wasted energy, reduced scientific throughput, and increased operational costs.

2 Methodology and Algorithm Design

This research introduces two novel heterogeneity-aware load balancing algorithms designed for scientific applications. The approach begins with a comprehensive performance characterization using empirically measured node capabilities, where performance factors P_i (representing the relative performance of node i) are modeled as $P_i = t_{baseline}/t_i$, where

Authors' Contact Information: Sowmya Yellapragada, University of Utah, Salt Lake City, Utah, USA; Jessica Imlau Dagostini, University of California, Santa Cruz, USA; Kevin Gott, Lawrence Berkeley National Laboratory, Berkeley, USA; Rebecca Hartman-Baker, Lawrence Berkeley National Laboratory, Berkeley, USA.

Manuscript submitted to ACM

t_i is the execution time on node i and $t_{baseline}$ is a reference execution time, establishing a baseline for relative node performance. System heterogeneity is quantified using $H = max(p_i)/min(p_i)$, providing a metric to assess the degree of performance variation within the computational environment.

The Performance-Aware Load Balancing algorithm implements a greedy assignment strategy that scales task assignments based on measured node performance characteristics. Unlike homogeneous approaches that distribute tasks uniformly, this algorithm considers the scaled completion times from empirically measured node performance, ensuring that faster nodes receive proportionally larger workloads. The algorithm sorts tasks by size in descending order and assigns each task to the node that minimizes the completion time, calculated as $node_{loads[i]} + task.size/P_i$.

The AMReX benchmark suite[4], which provides representative computational kernels for adaptive mesh refinement applications, is essential for evaluating load-balancing strategies across diverse scientific computing patterns. The Relation-Aware Load Balancing algorithm extends this approach by incorporating a novel Relative Performance Matrix ($R_{ij} = P_j/P_i$) that captures inter-node performance relationships. This algorithm introduces redistribution mechanisms with imbalance penalties and bonus alignment to optimize task distribution further. For each task assignment, the algorithm evaluates a comprehensive score that includes completion time, balance penalties computed from the (R_{ij}) matrix, and redistribution bonuses that encourage optimal load distribution across heterogeneous resources:

$$\text{score}_i = \text{completion\_time} + \alpha \cdot \text{balance\_penalty} - \beta \cdot \text{redistribution\_bonus} + \gamma \cdot \text{capacity\_bonus}$$

where α, β, and γ are tunable parameters that control the balance between load distribution and redistribution efficiency.

3 Implementation and Experimental Design

The algorithms were validated through computational experiments modeling NERSC's Perlmutter system architecture, a supercomputing platform featuring NVIDIA A100 GPUs with varying memory configurations. The experimental test considered 1 million tasks distributed across 6 heterogeneous nodes, modeling A100 GPUs with both 40GB and 80GB memory configurations in a 3x(40GB+80GB) arrangement. Testing was conducted across multiple grid sizes (256^3, 128^3, 64^3, 32^3) to evaluate scalability and performance consistency across different problem scales.

The evaluation utilized 14 representative AMReX computational kernels covering diverse computational patterns: memory-bound operations including mb (daxpy), dbl sum (reduce), scn (scan), and max (reduction); compute-intensive kernels featuring Jacobi variants (jac, jsy, jex, jmp), cb (compute-bound), and gsrb (Gauss-Seidel Red-Black); memory access pattern kernels including aos smp/sha (Array of Structures operations); and specialized workloads such as br (branching) and parser (parsing operations). This comprehensive kernel suite ensures that the algorithms are evaluated across the full spectrum of computational patterns typical in scientific applications.

Performance evaluation metrics included scheduling efficiency, defined as $ideal_{makespan}/actual_{makespan}$, load balance variance across nodes, and scalability across grid resolutions. Algorithmic complexity analysis reveals that while the traditional homogeneous approaches for knapsack algorithm operate at $O(nlogn + nlogm)$, the proposed algorithms maintain the practical complexity of $O(nlogn + n \cdot m)$ for n tasks and m nodes, for performance-aware algorithm, ensuring deployability on large-scale systems.

4 Results and Impact

The experimental results demonstrate notable improvements across both moderate (GPU-only) and extreme (CPU-GPU) heterogeneity scenarios. In moderate heterogeneity environments using A100 40GB versus 80GB configurations

(heterogeneity factors ranging from 1.09x to 1.26x), the proposed algorithms achieve 99.9% scheduling efficiency compared to 88.5-95.6% for traditional homogeneous approaches, translating to consistent 4.4-11.5% performance improvements.

Memory-intensive kernels show particularly strong gains of 12-26% on higher-capacity 80GB nodes due to their superior memory bandwidth characteristics compared to 40GB variants.

In extreme heterogeneity scenarios involving mixed CPU-GPU configurations, the improvements become dramatic. With heterogeneity factors ranging from 50.9x to 430.5x, traditional homogeneous schedulers achieve only 0.3-2.8% efficiency, while both Performance-Aware and Relation-Aware algorithms maintain 99.9% efficiency, representing improvements of up to 300x over conventional methods. These extreme improvements occur because traditional methods ineffectively assign computationally intensive tasks to CPUs, creating severe load imbalances where CPUs become bottlenecks with execution times orders of magnitude longer than GPUs, while our algorithms appropriately distribute work based on actual computational capabilities.

The performance characterization reveals that memory-intensive kernels benefit most from heterogeneity-aware scheduling in systems with varying memory bandwidth capabilities, such as A100 80GB nodes with faster memory performance compared to 40GB variants. Compute-bound kernels show more modest but still significant improvements, indicating that the algorithms are capable of successfully optimizing across diverse computational patterns.

5 Conclusions and Future Directions

Performance heterogeneity represents a fundamental challenge in modern HPC environments that cannot be addressed through traditional homogeneous scheduling assumptions. The proposed heterogeneity-aware algorithms provide solutions for HPC centers, potentially offering significant efficiency gains and improved scientific throughput on current and future heterogeneous supercomputing systems. The heterogeneous algorithms used in this work are available on GitHub[3].

Future research directions include benchmarking against WarpX applications, as WarpX represents a highly scalable, multi-physics plasma simulation code with heterogeneous CPU/GPU execution phases that make it particularly suitable for evaluating heterogeneous load-balancing strategies on real-world systems with over 1000 nodes, and analyzing the impact of spatial locality on communication overhead using Space Filling Curves.

References

[1] Thomas Scogland et al. “Node variability in large-scale power measurements: perspectives from the Green500, Top500 and EEHPCWG”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC ’15. Austin, Texas: Association for Computing Machinery, 2015. ISBN: 9781450337236. DOI: 10.1145/2807591.2807653. URL: https://doi.org/10.1145/2807591.2807653.

[2] Weiqun Zhang et al. “AMReX: a framework for block-structured adaptive mesh refinement”. In: *Journal of Open Source Software* 4.37 (2019), p. 1370. DOI: 10.21105/joss.01370. URL: https://doi.org/10.21105/joss.01370.

[3] Sowmya Yellapragada. *amrex_LB: optimized_algorithms/heterogeneous_lb*. 2025. URL: https://github.com/sowmya8900/amrex_LB/tree/main/src/optimized_algorithms/heterogeneous_lb.

[4] Weiqun Zhang. *amrex-microbenchmarks*. GitHub. 2025. URL: https://github.com/WeiqunZhang/amrex-microbenchmarks.