DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

Pacific Northwest
NATIONAL LABORATORY

PNNL-38155

Benchmark Tracking System for Performance Monitoring

August 2025

Braedon Billingsley
Joseph Cottam

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

PNNL-38155

Benchmark Tracking System for Performance Monitoring

August 2025

Braedon Billingsley
Joseph Cottam

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Benchmark Tracking System for Performance Monitoring

Braedon Billingsley

Abstract

Benchmarking is essential for high-performance software development, particularly for monitoring performance across code iterations. This project focused on enhancing the benchmarking process for Lamellar, an asynchronous runtime for High-Performance Computing (HPC) systems developed at Pacific Northwest National Laboratory. Prior to this work, benchmark results were difficult to track and compare across code versions, presenting significant challenges in identifying performance regressions and long-term trends. The primary objective was to establish a systematic, reproducible approach for measuring performance and detecting regressions following code commits. Our methodology involved three key components: standardizing benchmark outputs, implementing data versioning, and developing analysis tools. We standardized the benchmark output format to JSON Line records containing specific fields (execution time, hardware specifications, and environmental variables). To address data management challenges, we evaluated several options and eventually chose a git repository dedicated to benchmark data. We developed a suite of Python tools that processed benchmark results, enriched them with metadata, and facilitated search in the repository. The resulting system enables more efficient filtering and comparison of performance metrics across commit histories, hardware configurations, and benchmark variants through a unified query interface. Our implementation reduces computational overhead by first checking for existing results through configuration matching before initiating new benchmark runs, thereby conserving resources. The system has been validated by Lamellar developers. It organizes results by benchmark type and build configurations for efficient retrieval. Future developments include a planned Large Language Model interface for predicting benchmark performance, incorporating the criterion package for statistical analysis, which will enable automated detection of statistically significant performance changes, and integration with continuous integration pipelines. Despite these enhancements being reserved for future work, this project has successfully provided the Lamellar development team with a framework for maintaining consistent performance standards and identifying optimization opportunities across workloads and hardware environments.

Introduction

Software performance optimization relies on consistent benchmarking to guide development decisions. In high-performance computing (HPC) environments, where applications run across distributed systems, even minor performance regressions can substantially impact computational efficiency.

Lamellar is an HPC runtime built using Rust, aiming to offer a productive and safe development environment. It is a modern, performance-oriented runtime using Partitioned Global Address Space (PGAS) programming and asynchronous tasking program models [1]. While Lamellar has demonstrated promising results for distributed memory parallelism, its ongoing development faces a common challenge in HPC software engineering: maintaining consistent performance across an evolving codebase.

This project focused on tracking performance across code revisions, hardware configurations, and varied workloads (as opposed to micro-benchmarks focused on hotspot identification). Several approaches exist for addressing this challenge, including continuous integration performance testing and automated regression detection systems, but implementing these for HPC environments presents unique difficulties.

This project addresses needs within the Lamellar development workflow to systematically capture, store, and analyze benchmark results across its development lifecycle. By creating benchmarking infrastructure that connects performance metrics directly to code revisions, we enable data-driven optimization decisions and provide early detection of potential performance regressions.

The remainder of this report details our technical approach to designing and implementing this benchmarking system, the specific contributions made during the internship period, and recommendations for future enhancements to further support the Lamellar development team.

Progress

Problem Analysis and Solution Design

The initial phase of this project involved a review of Lamellar's existing benchmarking workflow, and the challenges faced by developers. Through discussions with the Lamellar development team, we identified requirements for an effective benchmarking system.

Results needed to be directly tied to specific git commits in both the Lamellar and Lamellar Benchmarks repositories. The system needed to support comparison of results across different hardware configurations while including comprehensive metadata about the execution environment. Robust query capabilities were essential for filtering and analyzing results across multiple revisions, all while integrating smoothly with existing development workflows.

After identifying these requirements, we evaluated several potential approaches.

- Data Version Control (DVC) [2]
- Git Large File Storage (LFS) [4]
- Criterion [3]
- LakeFS [5]
- Dolt [6]
- Git [7]

DVC initially seemed promising for its ability to version large datasets alongside code. However, testing revealed that DVC introduced unnecessary complexity for our use case, particularly in how it managed data across repository branches. Git LFS, while suitable, was ruled out due to licensing costs. Criterion, though a major benchmarking framework in the Rust ecosystem, wasn't selected due to its limited support for benchmarking across multiple environments, which was a key requirement for our project. Criterion excels at statistical rigor for function-level microbenchmarks but was not designed for comparing results across different hardware configurations with environmental metadata. Other options like Dolt (versioned relational databases) and LakeFS (data lake management) were beyond our use case requirements.

Based on this analysis, we decided to use a git repository for simplicity. We designed a solution with three core components: a standardized benchmark result format, a dedicated git repository for result storage, and a suite of Python tools for data processing and analysis. This approach balanced simplicity with the flexibility needed to support complex queries across benchmark results.

Implementation of Benchmark Result Standardization

The first technical contribution involved standardizing the output format for Lamellar benchmarks. We implemented a JSON Lines-based format where each benchmark execution generates a structured record containing performance metrics such as timing measurements, throughput rates, and data transfer statistics, along with benchmark-specific configuration parameters that define the test conditions. The format also captures comprehensive system information including hardware specifications, operating system details, and processor characteristics, as well as execution metadata such as run timestamps and dates for reproducibility. Additionally, each record includes version control data with git repository state information including commit hashes, messages, and dates for the benchmarking codebase, along with software version information to ensure experimental reproducibility across different environments and code versions.

This standardization required modifications to the existing benchmark code to capture system information and format the output consistently. The structured format enabled programmatic analysis while remaining human-readable for debugging purposes. By enforcing consistent

output fields, we ensured that results remained comparable across different execution environments and code versions.

Development of Data Management Infrastructure

For storing and versioning benchmark results, we created a dedicated git repository with a carefully designed structure to facilitate efficient querying. The repository organization separates results by benchmark type, build type, and benchmark commit, enabling retrieval without loading the entire dataset. This approach provides complete history preservation through git's versioning capabilities, distributed access for team members across multiple locations, simple integration with existing git-based development workflows, and the ability to opt out of tracking benchmarks.

To manage this repository, we developed a Python-based data export tool that processes raw benchmark outputs, validates their format, enriches them with additional metadata, and exports them to the appropriate location in the data repository. This automation saves time and minimizes the potential for human error in the data management process while ensuring consistent organization of benchmark results.

Query and Analysis Tool Development

The most significant technical contribution was the development of a flexible query interface for benchmark results. This Python tool enables Lamellar developers to filter benchmark results by git commit range, hardware configuration, benchmark parameters, and compare performance metrics across code revisions to identify regressions. With future development it will generate statistical summaries of performance trends and export filtered results for further analysis or visualization. Figure 1 demonstrates the complete data processing pipeline, illustrating how benchmark execution flows through filtering and metadata enrichment stages before being stored in the git repository, where the query tools can search and retrieve results (Figure 1).

The query interface features a configuration-based approach where users can specify complex queries through JSON files. The tool first checks for matching existing results before suggesting new benchmark commands to run, significantly reducing computational overhead by eliminating redundant executions.

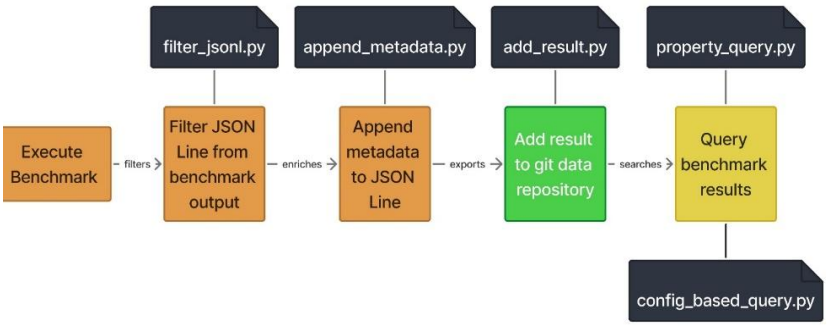

Figure 1 describes the data transformation process and python scripts responsible for each stage.

Integration with Development Workflow

To ensure adoption, we integrated the benchmarking infrastructure with Lamellar's existing development practices. This integration included providing documentation of the benchmarking workflow for developers, example queries for common performance analysis scenarios, and helper scripts to simplify the most frequent benchmark operations. Figure 2 illustrates the complete system architecture, showing how benchmark data flows from code commits through benchmark execution to the git-based storage system, where it can later be processed by both statistical testing tools for regression detection and Python analysis tools for generating performance insights (Figure 2).

The system was designed to support both interactive use during development and potential future integration with continuous integration pipelines. While automated performance regression detection was not implemented during this internship, the infrastructure provides the foundation for such capabilities, as demonstrated in the workflow diagram. The adoption of git as the storage mechanism allowed developers to leverage their existing knowledge of version control rather than learning an entirely new system.

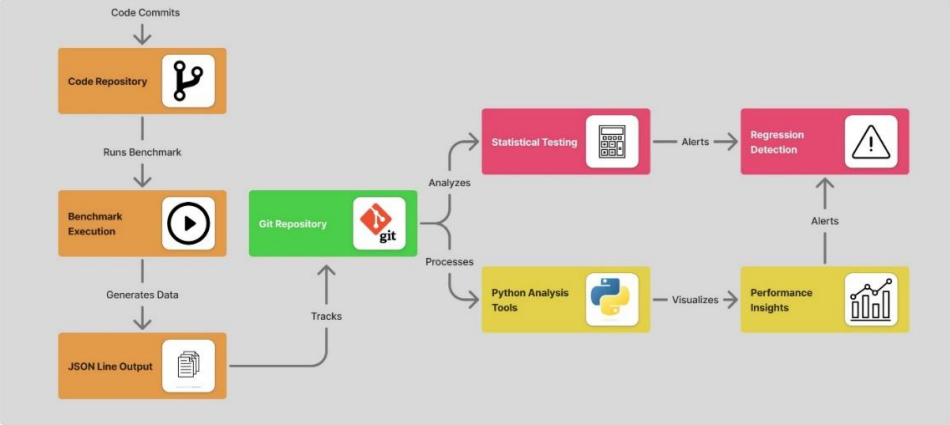

Figure 2 illustrates the complete workflow of the benchmarking system, from execution through storage to analysis and visualization.

Impact and Evaluation

The benchmarking infrastructure is expected to demonstrate value for the Lamellar team by providing visibility into performance changes across code revisions. It is intended to reduce the time required to record and analyze benchmark results by automated processing and supporting systematic comparison of performance across hardware configurations and software revisions. It establishes a foundation for data-driven optimization decisions.

Lamellar developers have validated the approach and identified opportunities for future enhancement. The design ensures that the system can evolve alongside Lamellar's development, accommodating new benchmarks and analysis requirements as they are developed. Performance data collected through this system will inform optimization.

Future Work

The benchmarking infrastructure established during this internship provides a solid foundation for several planned enhancements. Future development will focus on implementing automated performance insights generation through statistical analysis of benchmark trends. While Criterion wasn't suitable for our current multi-environment requirements, its compatibility with git and statistical rigor remain appealing aspects. If Criterion develops more general results-tracking capabilities to support benchmarking across different hardware configurations, we may incorporate it in the future.

A particularly promising direction is the development of a Large-Language-Model (LLM) based tool that could generate benchmark commands and predict expected performance results based on natural language prompts, allowing developers to anticipate performance impacts without executing full benchmark suites. Integration with continuous integration pipelines will enable automated benchmark execution upon new commits to the benchmark repository, coupled with statistical testing frameworks for regression detection. These automated systems will provide real-time performance monitoring and alert developers to potential performance regressions immediately upon code submission, transforming the current manual workflow into a fully automated performance validation system that integrates with Lamellar's development cycle.

Impact on Laboratory or National Missions

This benchmarking infrastructure project supports the Department of Energy's mission to advance scientific discovery through advancing high-performance computing capabilities. By enabling systematic performance tracking of Lamellar, our work enhances PNNL's distributed computing framework as it supports scientific computing applications. The improved performance visibility provided by our benchmarking system accelerates the development of efficient computational tools. The infrastructure creates capabilities for data-driven decision making in software optimization, allowing researchers to more efficiently use the laboratory's computational resources for scientific discovery. This work was partially supported through the High-Performance Data Analytics (HPDA) program at PNNL, which supports fundamental research and development in high-performance computing to enable scientific discovery.

Conclusions

This project successfully addressed a critical gap in the Lamellar development workflow by establishing benchmarking infrastructure that transforms ad-hoc performance testing into a structured approach. By standardizing benchmark output formats, implementing a version-controlled data repository, and developing query tools, we've created a system that links

performance results to code commits, enables meaningful cross-configuration comparisons, and eliminates redundant benchmark executions by first checking if they already exist. These improvements enhance the team's ability to maintain high-performance standards throughout the development lifecycle.

The modular approach employed ensures scalability as Lamellar continues to evolve, accommodating new benchmark types and analysis requirements without architectural changes. While the planned LLM-based performance prediction system remains for future implementation, the data collection and organization mechanisms established provide the necessary foundation for this advancement. This benchmarking infrastructure will play a crucial role in ensuring that performance remains a top consideration throughout Lamellar's development process, supporting PNNL's broader mission of advancing scientific discovery through high-performance computing.

References

[1] Friese R.D., R. Gioiosa, J. Cottam, E. Mutlu, G. Roek, P. Thomadakis, and M. Raugas. 2024. "Lamellar: A Rust-based Asynchronous Tasking and PGAS Runtime for High Performance Computing," *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Atlanta, GA, USA, 2024, pp. 1236-1251, doi: 10.1109/SCW63240.2024.00165.

[2] Iterative, *DVC: Data Version Control - Git for Data & Models* (2020) DOI:10.5281/zenodo.012345.

[3] Heisler, B. Criterion 0.7.0: *Statistics-driven Benchmarking Library for Rust.* https://docs.rs/criterion/latest/criterion/ (accessed 08-14-2025).

[4] GitHub Inc. Git Large File Storage (LFS): *An Open Source Git Extension for Versioning Large Files*. https://git-lfs.com/ (accessed 08-14-2025).

[5] Treeverse Ltd. LakeFS: *Open Source Data Version Control for Data Lakes*. https://lakefs.io/ (accessed 08-14-2025).

[6] DoltHub. Dolt: *SQL Database with Git-style Versioning*. https://www.dolthub.com/ (accessed 08-14-2025).

[7] Git Project. Git: *Distributed Version Control System*. https://git-scm.com/ (accessed 08-14-2025).

[8] SchedMD LLC. SLURM: *Simple Linux Utility for Resource Management*. https://www.schedmd.com/ (accessed 08-14-2025).

Appendix

Participants

Name	Institution	Project Role
Joseph Cottam	Pacific Northwest National Laboratory	Data Scientist; Mentor
Community College Internship Program (CCI)	U.S Department of Energy (DOE), Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS)	This work was supported in part by DOE and WDTS under the CCI program.

Scientific Facilities

Research Computing (RC):

- Junction cluster at PNNL
- gitlab.pnnl.gov

Notable Outcomes

Gold Experience Research Symposium:

- Presented a PowerPoint presentation to fellow interns, laboratory staff, and guests.

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354

1-888-375-PNNL (7665)

www.pnnl.gov