IRIS-HEP Summer Fellow Project Proposal
Benchmarking and Optimizing PyHS3

Applicant: Daryna Shevchuk
Mentor: Giordon Stark
Project duration: 12 weeks
Proposed start date: June 2026

Project Description

In High Energy Physics (HEP), experimental analyses are typically based on likelihood models that describe how well a set of parameters explains observed data. At its core, nearly every statistical inference task relies on evaluating probability density functions (pdfs) at different parameter values. However, evaluating a statistical model is only one part of the workflow. Before a model can be evaluated efficiently, the workspace must be parsed, a computational graph must be constructed, and this graph must be compiled and optimized through algorithmic rewrites. The resulting compiled function can then be evaluated repeatedly during fitting and inference procedures. While compilation introduces an upfront cost, it is typically amortized by the large number of subsequent function evaluations required in realistic analyses. That is why the performance of statistical software is extremely important.

PyHS3 is a pure-Python implementation of the HS3 standard that leverages PyTensor to construct and evaluate statistical models. Rather than directly executing all computations in Python, PyTensor transforms computational graphs and can compile them to optimized backends such as C++ or Numba. As a result, the performance of PyHS3 depends not only on the efficiency of model evaluation itself, but also on graph construction, compilation, optimization, and execution.

Currently, PyHS3 does not yet have a fully established benchmarking infrastructure. A benchmark suite is extremely important in order to understand which code paths are expensive, where unnecessary allocations occur, and whether new changes improve performance or, vice versa, introduce regressions.

Project Goals

The aim of this project is to construct benchmarking and optimization infrastructure for statistical analysis in PyHS3 (Python HEP Statistics Serialization Standard).

In this project, benchmarking is not limited to simply measuring execution time. We aim to examine representative benchmark cases: statistical models that realistically represent actual HEP workloads. We will systematically measure runtime behavior and memory consumption for operations such as pdf evaluation, likelihood calculation, parameter fitting, repeated evaluations, and scaling with model complexity.

Another very important aspect of the project is cross-framework comparison. We plan to compare PyHS3 with already established statistical frameworks such as ROOT, pyhf, zfit, and numba-stats. This comparison will provide two important things: performance comparison and numerical agreement. Numerical agreement is especially important because optimized implementations must still preserve correct and reproducible results across frameworks.

Planned Work and Software Deliverables

For this project we will use profiling and benchmarking tools such as `pyinstrument`, `line_profiler`, `pytest-benchmark`, and `tracemalloc`. The call-stack visualization provided by pyinstrument is particularly useful for understanding how execution time is distributed across the full model evaluation pipeline, from graph construction and compilation to repeated likelihood evaluations. Together, these tools provide complementary views of performance and memory usage and will help identify bottlenecks throughout the PyHS3 workflow.

Benchmarking will be performed separately for the major stages of the PyHS3 execution pipeline: computational graph construction, graph compilation, and model evaluation. Measuring these stages independently allows us to identify whether performance bottlenecks originate from model generation, compilation overhead, or repeated numerical evaluation during fitting and inference.

After the profiling stage, the optimization phase begins. Performance analysis and optimization have several key components. There has already been some initial work done to understand where PyHS3 can be sped up, from how functions are compiled and built to how they are evaluated.

Another direction is the investigation of caching strategies, such as `lru_cache`, to reduce redundant computations and improve the efficiency of likelihood evaluation for serialized workspaces. The project will also investigate optimization techniques employed by established frameworks such as ROOT and evaluate whether similar approaches can be applied within the PyHS3 ecosystem.

Another important direction is the identification of low-hanging optimization opportunities in model construction and graph generation. In some cases, significant performance improvements may come not from low-level numer-

ical optimizations but from simplifying the computational graph before compilation. This can include avoiding the construction of unnecessary graph components, removing parameters with negligible impact on the final likelihood evaluation, or reducing graph complexity through pruning and simplification strategies. Such approaches may decrease compilation costs and improve evaluation efficiency while preserving numerical correctness.

Expected software deliverables include:

- A reproducible benchmarking suite and representative HEP benchmark cases.
- Cross-framework comparisons with ROOT, pyhf, zfit, and numba-stats.
- Profiling reports and optimization improvements in PyHS3.
- Documentation of benchmarking workflows, results, and usage instructions.

Proposed Timeline

Weeks 1–2: Literature Review and Familiarization

- Study the HS3 specification, the current PyHS3 architecture, and related statistical frameworks including ROOT/RooFit, pyhf, zfit, and numba-stats.
- Identify representative benchmark cases and select appropriate profiling and benchmarking tools.

Weeks 3–4: Benchmark Infrastructure Development

- Develop the initial benchmarking suite for PyHS3 and implement representative benchmark cases.
- Validate the correctness, reproducibility, and reliability of benchmark outputs.

Weeks 5–6: Cross-Framework Benchmarking

- Compare PyHS3 against ROOT/RooFit, pyhf, zfit, and numba-stats.
- Analyze runtime behavior, memory usage, and numerical agreement.

Weeks 7–8: Profiling and Bottleneck Analysis

- Perform runtime and memory profiling using benchmarking tools.
- Identify expensive code paths and repeated computations.

Weeks 9–10: Optimization Work

- Implement profiling-driven optimizations in PyHS3, prioritizing the most impactful bottlenecks identified during benchmarking and profiling. Depending on findings, optimization efforts may extend into later stages of the project.
- Re-run benchmarks, assess performance gains, and analyze the impact of the implemented optimizations.

Weeks 11–12: Finalization and Reporting

- Finalize the benchmark suite, benchmarking workflows, and optimization evaluation.
- Prepare documentation, benchmark reports, the final project report, and the presentation.

Expected Outcome

As the result of this project, PyHS3 will be improved by adding reproducible benchmarking infrastructure capable of evaluating runtime performance and memory usage across representative HEP statistical workloads. The project is also expected to provide profiling-driven optimization improvements, cross-framework benchmark comparisons, and documentation that can support future development and performance regression tracking in PyHS3. The project will also verify numerical correctness and reproducibility of PyHS3 results across frameworks, ensuring that performance improvements do not compromise statistical validity.

References

- PyHS3: link
- HS3: link
- ROOT: link
- pyhf: link
- zfit: link
- PyTensor: link