Project Proposal: Alpaka Benchmarks for CMS Pixel Track Reconstruction

Description

This project aims to create a suite of performance benchmarks for the *alpaka*[1] library to evaluate various parallel programming constructs used in the CMS experiment's Pixeltrack[2] reconstruction software at CERN. *Alpaka* enables platform-independent parallel programming across CPUs, GPUs, and FPGAs through a unified API. As scientific software complexity increases, continuous monitoring and optimization of *alpaka's* performance become essential.

The primary goal is writing targeted C++ benchmarks to measure the performance of specific *alpaka* functionalities. The project will begin with simple benchmarks such as memory allocation, arithmetic operations, and random number generation, progressively introducing the student to *alpaka's* diverse features and API. Finally, the standard miniBUDE[3] benchmark will be implemented using *alpaka*, leveraging the existing Babelstream[4] benchmark as a practical reference. While integrating benchmarks into our Continuous Integration (CI) and CTest systems is beyond the primary scope, the benchmarks will be designed with future CI integration in mind.

Deliverables

- **Github issue** with benchmark implementation plan and progress documentation as a
- **Simple benchmark suite** with C++ implementations for memory allocation, arithmetic operations, and random number generation.
- **miniBUDE benchmark** implementation in *alpaka*.
- **Brief user documentation** explaining benchmark setup and usage.
- **Final report** summarizing implemented benchmarks, encountered challenges, and performance observations.

[1] https://github.com/alpaka-group/alpaka
[2] https://github.com/cms-patatrack/pixeltrack-standalone
[3] https://github.com/UoB-HPC/miniBUDE
[4] https://github.com/alpaka-group/alpaka/tree/develop/benchmarks/babelstream

Timeline

Month 1: Introduction and Initial Benchmarks

- Setup development environment with *alpaka* and basic project dependencies.
- Familiarize with *alpaka* API and existing Babelstream benchmark.
- Implement simple benchmarks (memory allocation, arithmetic operations, random number generation).
- Document initial progress and plan benchmarks.

Month 2: Development and Intermediate Complexity

- Enhance and refine initial benchmarks.
- Gain deeper understanding of *alpaka's* parallel constructs and data handling.
- Begin implementation of the miniBUDE benchmark, referencing Babelstream.
- Continuously document code and benchmarking methods.

Month 3: Final Benchmark Implementation and Reporting

- Complete and optimize miniBUDE benchmark.
- Validate and assess performance of implemented benchmarks.
- Finalize user documentation and usage guidelines.
- Compile and submit a final report detailing benchmarks, implementation insights, and performance results.