Project Proposal: CI/CD improvements for Alpaka library, and related projects

Modern High-Energy Physics experiments are presented with significant computational challenges, both in terms of data volume and processing power. As HPC centers attempt to increase their computational power, the energy consumption limits of this architecture have necessitated the introduction of GPU-based accelerators to provide the majority of the computing. Programming techniques for GPUs and other massively parallel accelerators are very different from those of traditional CPUs, requiring a significant understanding of the hardware architecture to achieve good performance. HEP experiments have developed millions of lines of code, and in order to use these new computational accelerator facilities, they would need to rewrite large amounts of their codebases.

In order to be able to run on all the different hardware architectures, from various vendors, this task would have to be repeated multiple times in each architecture's preferred language. That's the reason why the alpaka[1] library was developed in the first place. It is platform-independent and supports concurrent and cooperative runs on multiple devices, such as the host's CPU with different instruction sets like x86, ARM, or RISC-V as well as attached accelerators, for instance, CUDA , AMD, or Intel GPUs. It allows the programmer to write a function once and execute it on different accelerators.

This project aims to optimize the test coverage and runtime of alpaka. Testing alpaka with all possible software dependencies would result in about 2,500,000 different combinations of test jobs. If each job would take 4 minutes on average, testing a single commit on GitHub would take months. To reduce the number of jobs and also save time for manually adding new software dependency combinations, we implemented an open source CI job generator[2]. The generator reduces the number of jobs to about 170, but has several problems and limitations. The biggest problem is to check whether all expected test pairs are generated. Therefore, as a result of the project, we want to finalize the work on the new open source version of the generator, migrate Alpaka to the new generator[3], and verify that the CI works as expected.

[1] https://github.com/alpaka-group/alpaka
[2] https://github.com/alpaka-group/alpaka/blob/develop/script/job_generator/job_generator.py
[3] https://github.com/alpaka-group/bashi

Deliverables

- **Design Document** in form of a Github issue in bashi generator repository
- **Updated list of filter rules** up to the reference implementation
- **Testing system** for 100% filter rule coverage
- **User documentation** describing the new features
- **Deployment on PyPI** and integrating into alpaka Github repository
- **Performance analysis** assessing acceptable increase in CI runtime
- **Report** detailing the implementation details, test results, and further improvement ideas

Timeline

Month 1: Learning

- Understand *alpaka* software development workflow with CI/CD on Github and Gitlab
- Set up Python development environment for bashi CI development
- Set up an example Github C++ project for CI testing
- Write the design document describing plans for implementing missing features

Month 2: Implement missing features in bashi

- Implement the filter rules list
- Implement tests for the filter rules list
- Write user documentation as the implementation progresses
- Deploy on pypi

Month 3: Alpaka integration, performance analysis and optimization

- Integrate into alpaka CI
- Test the performance of the generator and the CI runners
- Write a report detailing the development process, code changes, improvements, and performance analysis results