Continuous Integration in Complex Research Software – Handling Complexity

deRSE24 – Conference for Research Software Engineering

Tobias Huste, Christian Hüser, Norman Ziegner, Simeon Ehrig, Rene Widera
Helmholtz-Zentrum Dresden – Rossendorf

Continuous Integration @ HZDR and HIFIS

General Purpose Runners
- Docker-machine based runners in Openstack
- Different flavors (#CPUs, RAM)

Special Purpose Runners
- Multiple CPU architectures (ARM, Power, AMD, Intel)
- GPUs (NVIDIA, AMD)

Helmholtz Codebase - CI Infrastrucure

HPC Runner in Alpha state
- Test software stacks directly on the HZDR HPC system
- E.g. for runtime tests or continuous performance analysis

Different Operating Systems
- Apple Silicon based MacOS-Runner
- Windows runner ready for early adpoters

Showcase: Alpaka

- Header-only C++17 abstraction library for accelerator development.
- Its aim is to provide performance portability across accelerators through the abstraction (not hiding!) of the underlying levels of parallelism.
- Write code once and execute it on different processors

https://github.com/alpaka-group/alpaka

Showcase: Alpaka

CPU

```
struct InitRandomKernel
{
    template<typename TAcc, typename TExtent, typename TRandEngine>
    ALPAKA_FN_ACC auto operator()(
        TAcc const& acc,
        TExtent const extent,
        TRandEngine* const states,
        std::size_t pitchRand) const -> void
    {
        auto const idx = alpaka::getIdx<alpaka::Grid, alpaka::Threads>(acc);

        if(idx[0] < NUM_Y && idx[1] < NUM_X)
        {
            auto const linearIdx = alpaka::mapIdx<1u>(idx, extent)[0];
            auto const memoryLocationIdx = idx[0] * pitchRand + idx[1];
            TRandEngine engine(42, static_cast<std::uint32_t>(linearIdx));
            states[memoryLocationIdx] = engine;
        }
    }
};
```

GPU

FPGA

Showcase: Alpaka

Example, full test matrix CUDA backend:

- 4 GCC versions, 6 Clang versions
- 10 CUDA SDK versions
- 4 Cmake versions
- 7 Boost versions

Naive approach:

→ 2800 combinations at 6 minutes per job

→ Pipeline run: ~9,5h at 30 jobs in parallel

Supported Compilers

This library uses C++17 (or newer when available).

Accelerator Back-end	gcc 9.5 (Linux)	gcc 10.4 / 11.1 (Linux)	gcc 12.3 (Linux)	gcc 13.1 (Linux)	clang 9 (Linux)	clang 10 / 11 (Linux)	clang 12 (Linux)	clang 13 (Linux)	clang 14 (Linux)
Serial	✅	✅	✅	✅	✅	✅	✅	✅	✅
OpenMP 2.0+ blocks	✅	✅	✅	✅	✅	✅	✅	✅	✅
OpenMP 2.0+ threads	✅	✅	✅	✅	✅	✅	✅	✅	✅
std::thread	✅	✅	✅	✅	✅	✅	✅	✅	✅
TBB	✅	✅	✅	✅	✅	✅	✅	✅	✅
CUDA (nvcc)	✅ (CUDA 11.0 - 12.2)[2]	✅ (CUDA 11.4 - 12.0)[2]	✅ (CUDA 12.0 - 12.2)	❌	✅ (CUDA 11.0-11.2; 11.6 - 12.0)[2]	✅ (CUDA 11.2, 11.6 - 12.0)[2]	✅ (CUDA 11.6 - 12.0)[2]	✅ (CUDA 11.7 - 12.0)	✅ (CUDA 11.8 - 12.0)
CUDA (clang)	-	-	-	❌	❌	❌	❌	❌	✅ (CUDA 11.0 - 11.5)
HIP (clang)	-	-	-	❌	❌	❌	❌	❌	✅ (HIP 5.0 - 5.2)

Challenge

- Bigger Open Source software projects very often hosted on GitHub
- Free CI resources not always enough due to, e.g.
  - Amount of resources
  - Special hardware (CPU architectures, GPUs) required
  - Reaching free limits
- Required resources are locally available, but not usable on GitHub by default

How to combine GitHub with locally available GitLab CI resources?

Idea

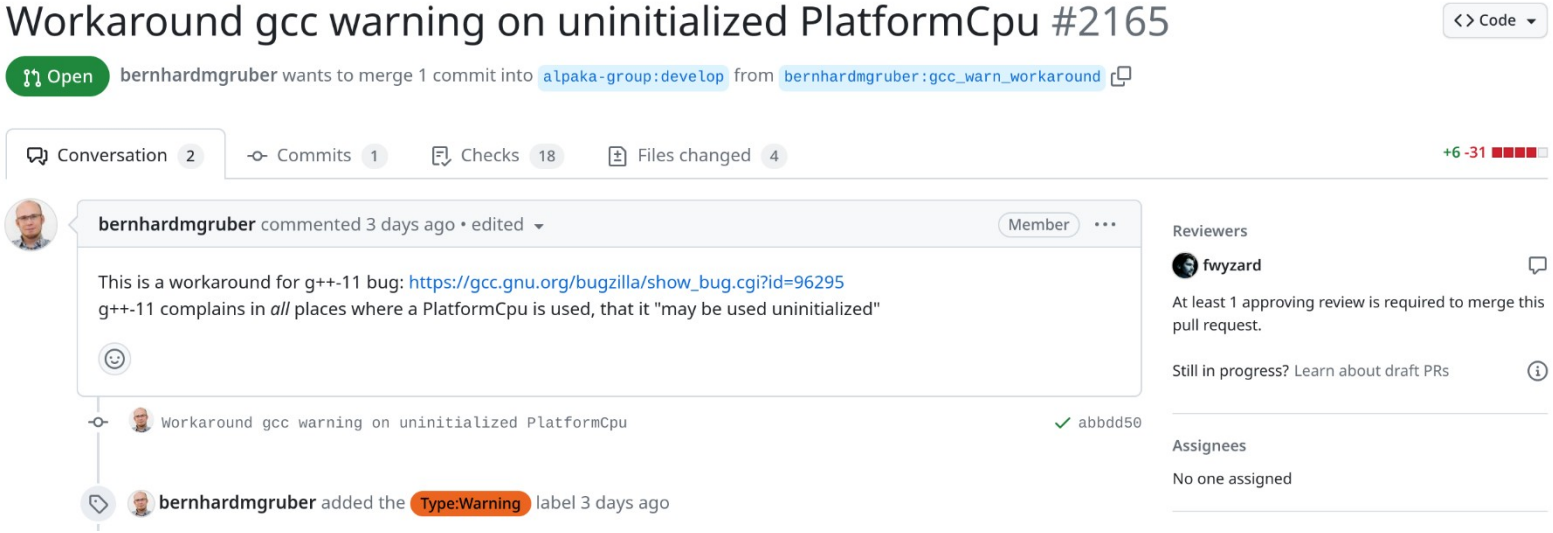
Use GitLab - GitHub CI Integration

- GitLab provides an integration to allow for using GitLab CI in a GitHub project
- Limitations
  - Feature available in the Premium Edition only → cannot use our own instance
  - Did not work for Pull Requests from forks
- **How to proceed?**

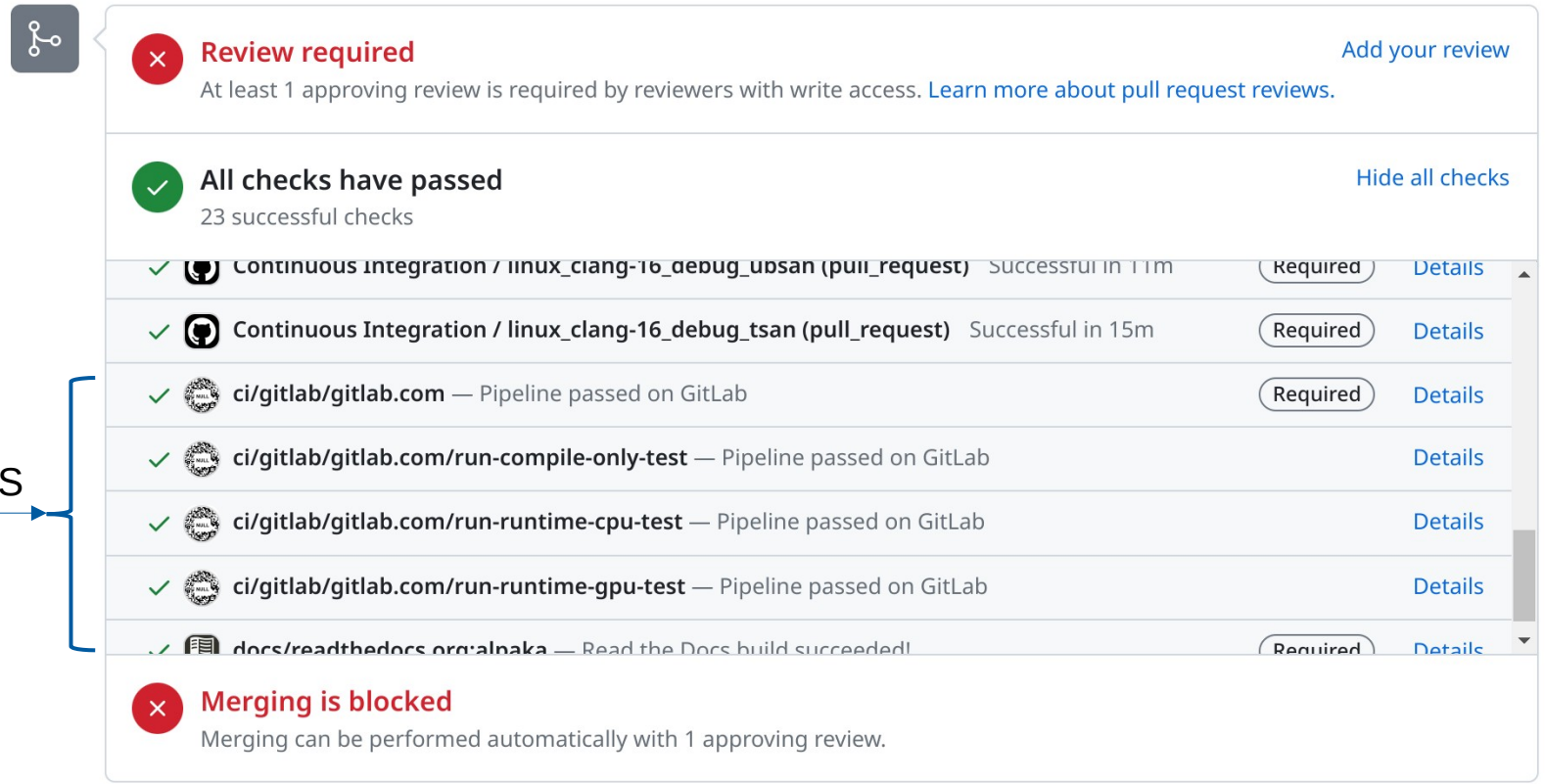
Use gitlab.com and make use of group runners

Requires registration in the GitLab for Open Source program

Workaround gcc warning on uninitialized PlatformCpu #2165



...



Uses CI@HIFIS

DRESDEN concept SCIENCE AND INNOVATION CAMPUS

HZDR

Workaround gcc warning on uninitialized PlatformCpu

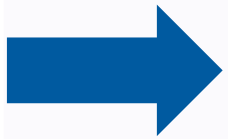
Delete

passed **HZDR Bot** created pipeline for commit abbdd505 finished 2 days ago

For pr-2165/bernhardmgruber/alpaka/gcc_warn_workaround

latest 4 Jobs 0.81 128 minutes 41 seconds, queued for 0 seconds

Pipeline | Needs | Jobs 4 | Tests 0



generator

generate

run-test-jobs

run-compile-only-test
Trigger job

run-runtime-cpu-test
Trigger job

run-runtime-gpu-test
Trigger job

Downstream

run-compile-only...
#1006719087
Child

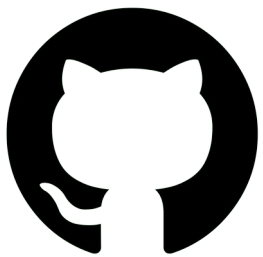
compile_only_job-stage0

linux_clang-cuda14-cuda11.4_compile_only_c...
linux_clang-cuda15-cuda11.0_compile_only_c...
linux_clang-cuda16-cuda11.3_compile_only_c...
linux_hipcc5.0_compile_only_cmake3.24.4_bo...
linux_hipcc5.2_compile_only_cmake3.25.3_bo...
linux_hipcc5.4_compile_only_cmake3.22.6_bo...
linux_hipcc5.4_compile_only_cmake3.23.5_bo...
linux_hipcc5.4_compile_only_cmake3.24.4_bo...
linux_hipcc5.4_compile_only_cmake3.25.3_bo...
linux_hipcc5.5_compile_only_cmake3.22.6_bo...
linux_hipcc5.5_compile_only_cmake3.23.5_bo...
linux_hipcc5.5_compile_only_cmake3.25.3_bo...
linux_hipcc5.5_compile_only_cmake3.25.3_bo...

compile_only_job-stage1

linux_clang-cuda14-cuda11.2_compile_only_c...
linux_clang-cuda14-cuda11.5_compile_only_c...
linux_clang-cuda15-cuda11.1_compile_only_cm.
linux_clang-cuda15-cuda11.4_compile_only_c...
linux_hipcc5.0_compile_only_cmake3.23.5_bo.
linux_hipcc5.1_compile_only_cmake3.25.3_bo..
linux_hipcc5.2_compile_only_cmake3.25.3_bo.
linux_hipcc5.3_compile_only_cmake3.23.5_bo.
linux_hipcc5.3_compile_only_cmake3.25.3_bo.
linux_hipcc5.3_compile_only_cmake3.26.4_bo.
linux_hipcc5.4_compile_only_cmake3.25.3_bo.
linux_hipcc5.5_compile_only_cmake3.25.3_bo.
linux_hipcc5.5_compile_only_cmake3.25.3_bo.

How does it work?



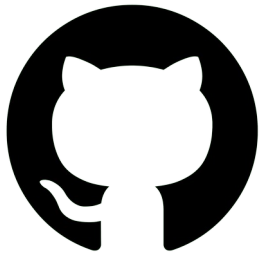
github.com/alpaka-group/alpaka

GitLab native project mirroring

Local branches only

gitlab.com/hzdr/crp/alpaka

How does it work?



github.com/alpaka-group/alpaka

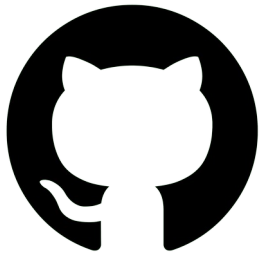
GitLab native project mirroring

Local branches only

gitlab.com/hzdr/crp/alpaka

Fork 1 – e.g. github.com/tobiashuste/alpaka
Fork 2
…
Fork n

How does it work?



github.com/alpaka-group/alpaka

GitLab native project mirroring

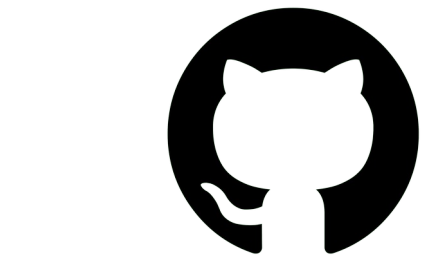
Local branches only

gitlab.com/hzdr/crp/alpaka

Fork 1 – e.g. github.com/tobiashuste/alpaka Fork 2 … Fork n

Local branch 1 – (tobiashuste-alpaka-feature-1) Local branch 2 … Local branch n

How does it work?



GitLab native project mirroring

Local branches only

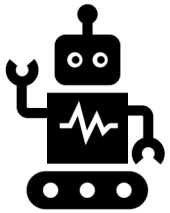
github.com/alpaka-group/alpaka

gitlab.com/hzdr/crp/alpaka

Fork 1 – e.g. github.com/tobiashuste/alpaka
Fork 2
…
Fork n

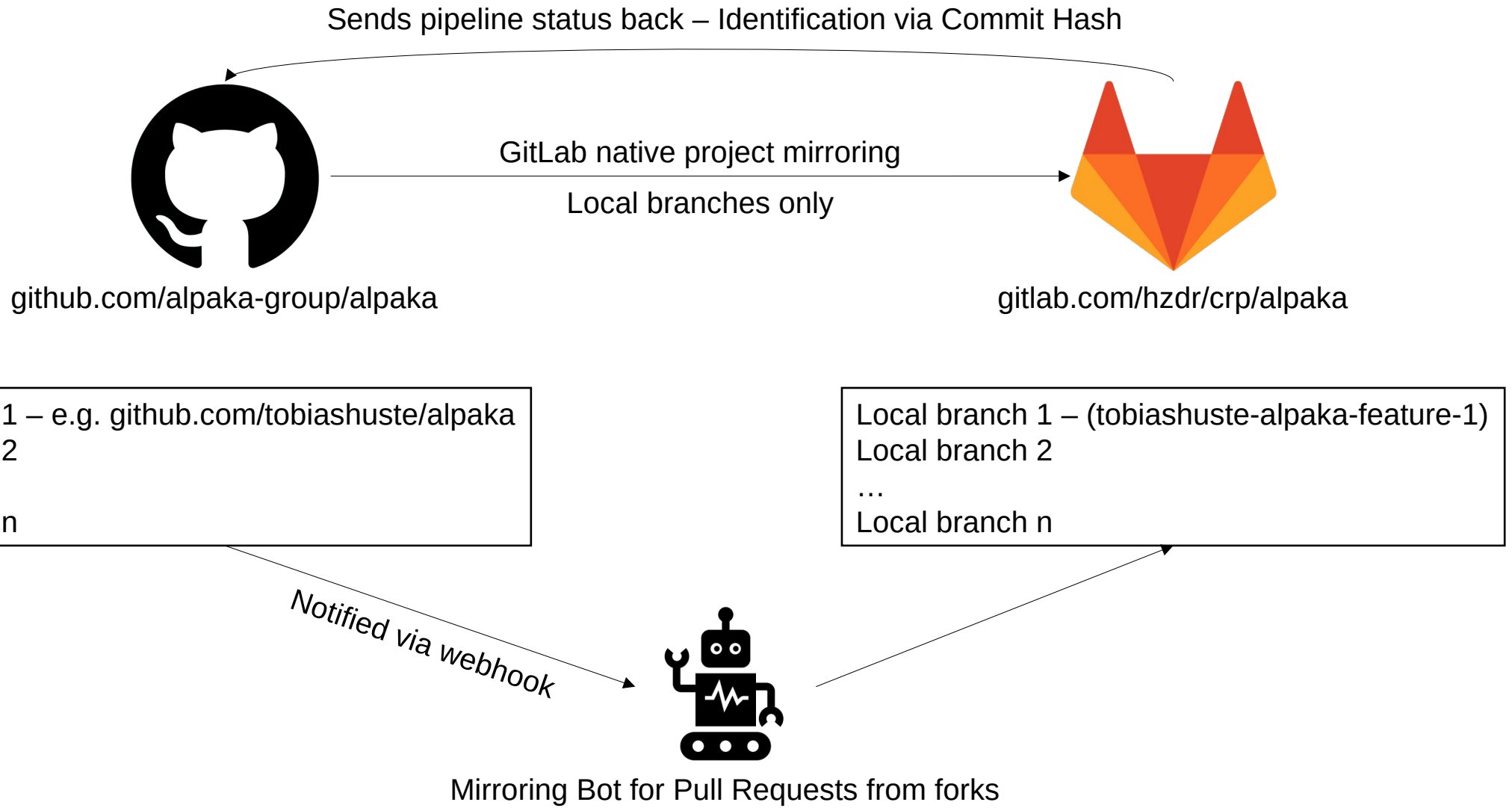
Local branch 1 – (tobiashuste-alpaka-feature-1)
Local branch 2
…
Local branch n

Notified via webhook



Mirroring Bot for Pull Requests from forks

How does it work?



Strategies to Optimize the CI Pipeline

Job Generator

Goals

- Reduce total CI runtime
- Good test coverage
- Maintainability, extensibility, verifiability

Restrictions and Competitions

- Shared runners
- Special runner resources more limited
- Person hours

Use a job generator to implement algorithms for dynamic job configuration

Uses Dynamic Child Pipelines

Strategies to Optimize the CI Pipeline

Pairwise Testing

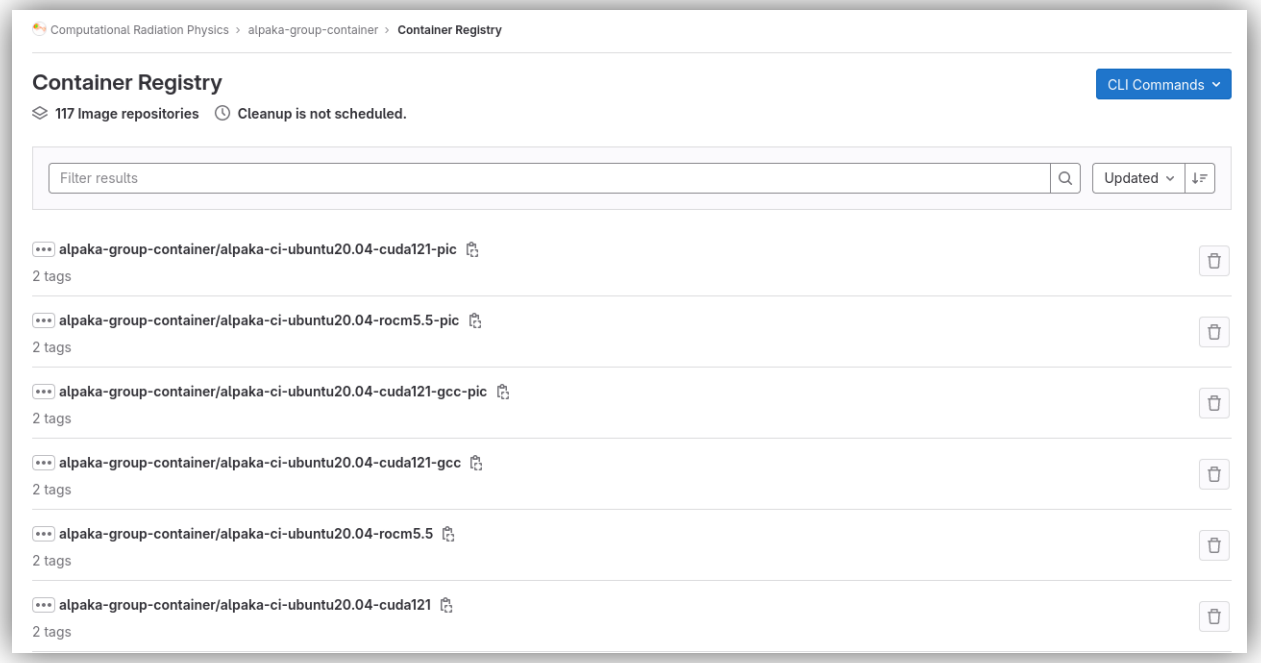
- Algorithm to create a test matrix depending on the test parameters
  - Example: Host compiler, device compiler, boost version, …
- Each combination of two parameter values appears at least in one job
- Generator is able to forbid combinations
  - Alpaka-related combination rules defined in Python library - https://github.com/alpaka-group/alpaka-job-matrix-library
  - Use Python library `allpairspy` - https://pypi.org/project/allpairspy/
- **In practice:** Number of jobs increases logarithmic depending on number of job parameters

Strategies to Optimize the CI Pipeline

Pre-Built Containers

- Provide container images containing all dependencies
- Fast download vs. slow compiling
- Caching of image layers possible
- Can easily be used on local development systems as well

https://codebase.helmholtz.cloud/crp/alpaka-group-container



Now hosted on the Helmholtz Codebase container registry

- Same data center as runners
- No hard storage limitations

Strategies to Optimize the CI Pipeline

Wave Scheduling

Problem

- Running all jobs in same stage utilize all resources
- If a job finished instantly a new job starts

→ no other PR of the same or project can execute CI jobs

- If one job fails, all other running jobs still try to finish

→ wasting CI resources

Solution

- Distribute jobs on waves (stages)
- Release CI resources constantly and start to allocate new resources, if the new stage is starting
- Reorder jobs
  - Try to fail in the first wave, if there is a bug in the code

Workaround gcc warning on uninitialized PlatformCpu

Delete

passed **HZDR Bot** created pipeline for commit abbdd505 finished 2 days ago

For pr-2165/bernhardmgruber/alpaka/gcc_warn_workaround

latest 4 Jobs 0.81 128 minutes 41 seconds, queued for 0 seconds

Pipeline | Needs | Jobs 4 | Tests 0

generator

- generate

run-test-jobs

- run-compile-only-test — Trigger job
- run-runtime-cpu-test — Trigger job
- run-runtime-gpu-test — Trigger job

Downstream

- run-compile-only... #1006719087 — Child

compile_only_job-stage0

- linux_clang-cuda14-cuda11.4_compile_only_c...
- linux_clang-cuda15-cuda11.0_compile_only_c...
- linux_clang-cuda16-cuda11.3_compile_only_c...
- linux_hipcc5.0_compile_only_cmake3.24.4_bo...
- linux_hipcc5.2_compile_only_cmake3.25.3_bo...
- linux_hipcc5.4_compile_only_cmake3.22.6_bo...
- linux_hipcc5.4_compile_only_cmake3.23.5_bo...
- linux_hipcc5.4_compile_only_cmake3.24.4_bo...
- linux_hipcc5.4_compile_only_cmake3.25.3_bo...
- linux_hipcc5.5_compile_only_cmake3.22.6_bo...
- linux_hipcc5.5_compile_only_cmake3.23.5_bo...
- linux_hipcc5.5_compile_only_cmake3.25.3_bo...
- linux_hipcc5.5_compile_only_cmake3.25.3_bo...

compile_only_job-stage1

- linux_clang-cuda14-cuda11.2_compile_only_c...
- linux_clang-cuda14-cuda11.5_compile_only_c...
- linux_clang-cuda15-cuda11.1_compile_only_cm...
- linux_clang-cuda15-cuda11.4_compile_only_c...
- linux_hipcc5.0_compile_only_cmake3.23.5_bo...
- linux_hipcc5.1_compile_only_cmake3.25.3_bo...
- linux_hipcc5.2_compile_only_cmake3.25.3_bo...
- linux_hipcc5.3_compile_only_cmake3.23.5_bo...
- linux_hipcc5.3_compile_only_cmake3.25.3_bo...
- linux_hipcc5.3_compile_only_cmake3.26.4_bo...
- linux_hipcc5.4_compile_only_cmake3.25.3_bo...
- linux_hipcc5.5_compile_only_cmake3.25.3_bo...
- linux_hipcc5.5_compile_only_cmake3.25.3_bo...

Strategies to Optimize the CI Pipeline

Job filter

- During development only run a reasonable subset of the CI pipeline
- Example: no need to run the full pipeline during development if I know I implement a specific CUDA feature
- Implemented via git commit message

```
Add function to filter and reorder CI jobs

This commit message demonstrates how it works. The job filter removes
all jobs whose names do not begin with NVCC or GCC. Then the jobs are
reordered. First all GCC11 are executed, then all GCC8 and then the
rest.

CI_FILTER: ^NVCC|^GCC
CI_REORDER: ^GCC11 ^GCC8
```

Summary

- Alpaka developer team happy with the solution
- Allows them to implement more advanced CI pipeline
- Still, the setup is quite complex
- A lot of effort put into automation

→ **Manual testing would be more time consuming and more error-prone**

→ **More resources, especially GPUs in Openstack will help a lot (stay tuned)**

Further Reading

- HIFIS – https://hifis.net
- Ansible Roles: https://github.com/hifis-net
- Helmholtz Research Software Directory: https://helmholtz.software
- Helmholtz Codebase: https://codebase.helmholtz.cloud
- Alpaka group: https://github.com/alpaka-group