Continuous Performance Monitoring for Lattice QCD

Simon Bürger, Antonin Portelli

December 12th, 2024

Our code base

- *Grid*: data parallel C++ container classes mapping efficiently to SIMD architectures including GPUs
  `https://github.com/paboyle/Grid`
- *Hadrons*: Grid-based workflow management system for lattice field theory simulations
  `https://github.com/aportelli/Hadrons`

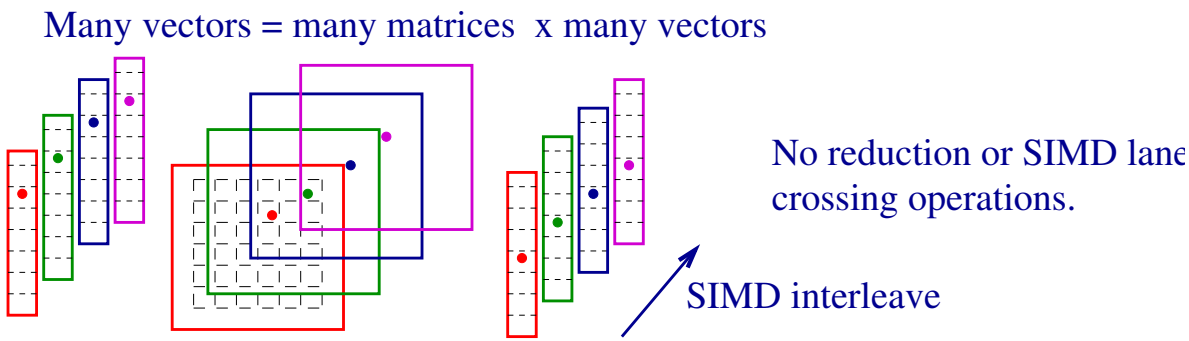


Figure: Grid architechture, from [hep-lat/1512.03487]

Tursa Computing cluster

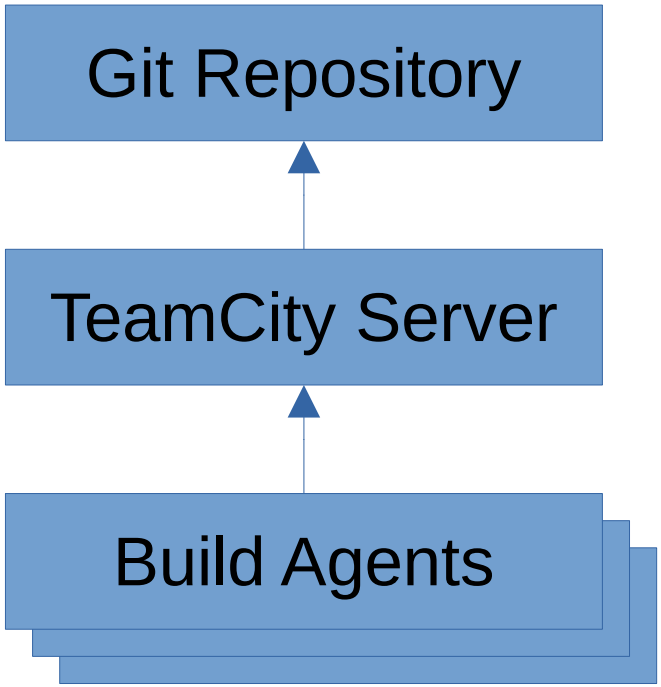
- 178 compute nodes, each equipped with
  - two *AMD EPYC* processors
  - four *NVIDIA Ampere A100* accelerators
- interconnect
  - NVLink on each node, running at 600 GB/s
  - Four HDR-200 infiniband interfaces per node
- jobs are run via Slurm scheduler from dedicated login nodes

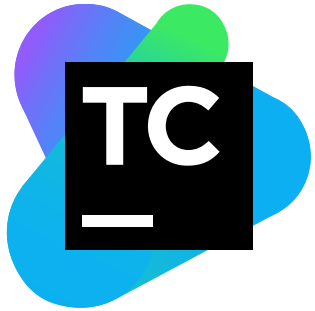
DiRAC

Goals

- **Continuous integration ("CI")**: Notice breaking changes as soon as possible, avoid infamous "works on my machine"
- **Continuous deployment ("CD")**: Improve reproducibility and simplify user experience
- **Performance monitoring**: Detect performance degredations caused by any part of the system

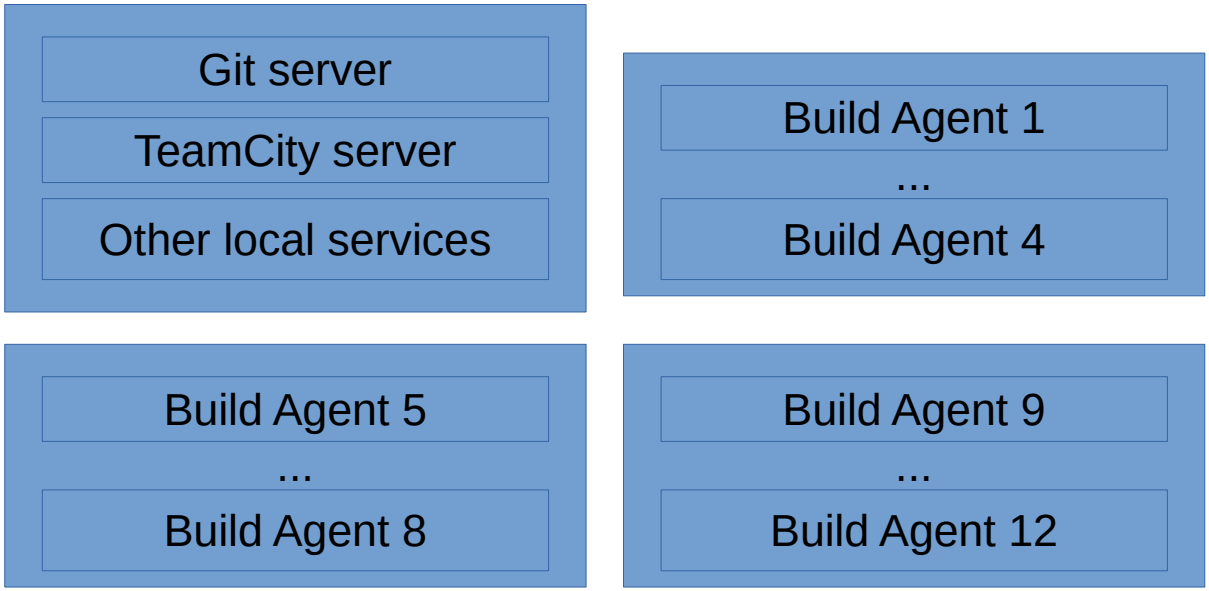
Jetbrains TeamCity Architecture





- Scalable to arbitrary number of build servers
- Communication via https

Dedicated CICD hardware



- Four AMD EPYC servers, same environment as login nodes
- Head server: docker containers, externally visible
- Agents: run on bare metal, not externally visible

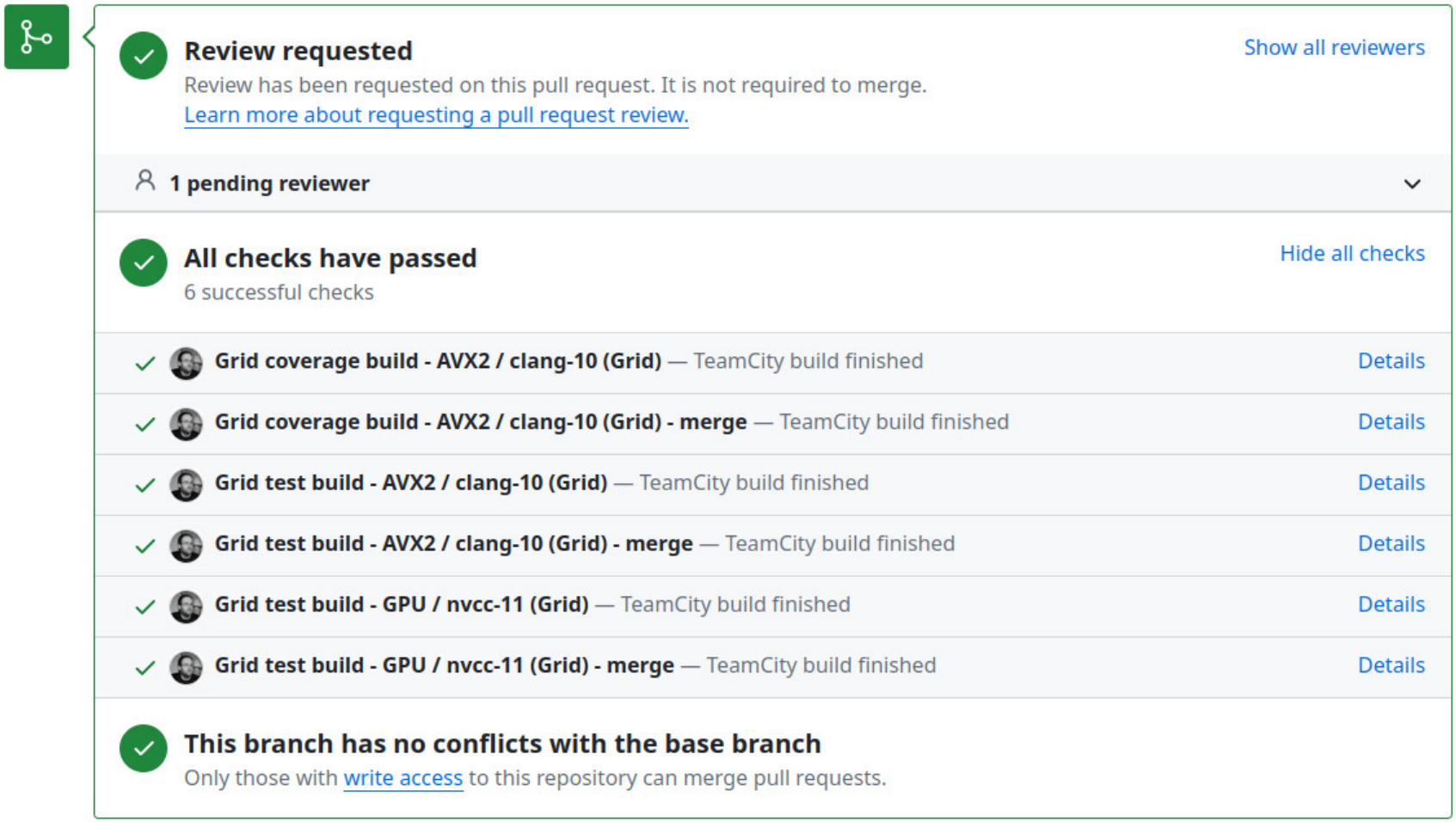
Triggers for the CI system

- Each commit, including pending pull requests
  $\longrightarrow$ build Grid and Hadrons and run unittests
- Once per day:
  $\longrightarrow$ deploy new production binaries if there were any changes
  $\longrightarrow$ run benchmarks, using latest production binaries
- Benchmarking run regardless of code changes, thus monitoring the runtime environment as well.

Integration into the HPC cluster Tursa

- Problem: Build servers do not have GPUs
- Solution: GPU-based unittests and benchmarking jobs are submitted via SLURM to the computing cluster
- Finished jobs report back to TeamCity via a REST API
- Build agents are not waiting for SLURM queue

What does it look like? On a GitHub pull request:



TC

Projects Changes Agents 8 Queue 0

Log in...

Search...

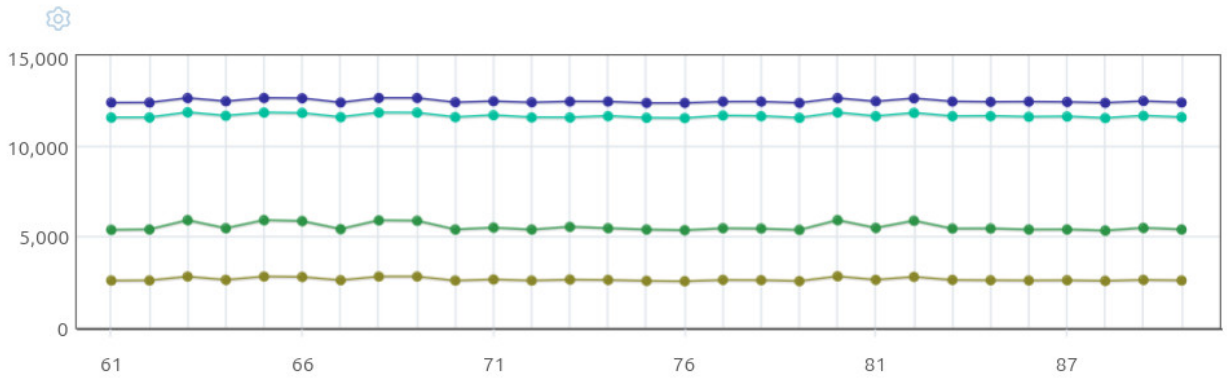
All Projects
Favorite Builds

PROJECTS
- Lattice
  - Benchmarks
    - TursaCpu
    - TursaGpu
      - Benchmark_Grid
        - **4 x NVIDIA A100-40**
        - 4 x NVIDIA A100-80
        - 8 x NVIDIA A100-40
        - 8 x NVIDIA A100-80
  - Grid
  - Hadrons

Lattice / Benchmarks / TursaGpu / Benchmark_Grid

4 x NVIDIA A100-40

Overview Change Log Problems **Statistics** Build Chains Flaky Tests 0



GFlops per nodes All N

- Comparison Point
- Wilson Fermion
- Staggered Fermion
- DomainWall Fermion

Average

Show failed

Range Month

Stability

- All build artifacts are backed up to a cloud storage provider
- Configuration of TeamCity itself is tracked in a git repository

Hardening

- Build servers are not publicly visible, communication to main server is "one-way"
- Build agents run as dedicated user with limited permissions on the cluster, e.g., no access to research data

Future outlook

- Solution is scalable to multiple HPC clusters

This CI/CD solution was funded by the STFC DiRAC Facility

Questions?

for a live demo, come talk to me later, or visit
`https://ci.dev.dirac.ed.ac.uk`
(choose "Log in as guest")

DiRAC

THE UNIVERSITY of EDINBURGH