Moving towards Continuous Benchmarking (CB)

9th JLESC Workshop
April 15th-17th, 2019 | Knoxville, TN

Anzt, Chen, Cojean, Dongarra, Flegar, Nayak, Quintana-Orti, Tsai, Wang
A Healthy Software Development Cycle
A Healthy Software Development Cycle

- Version control system for tracking changes and coordinating collaborative development.
A Healthy Software Development Cycle

• Version control system for tracking changes and coordinating collaborative development.
• CI Build system continuously checks the compilation success on different architectures.

*Continuous Integration (CI)*
A Healthy Software Development Cycle

- Version control system for tracking changes and coordinating collaborative development.
- CI Build system continuously checks the compilation success on different architectures.
- Unit tests ensure functionality and validity of all building blocks.

**Continuous Integration (CI)**
A Healthy Software Development Cycle

- Version control system for tracking changes and coordinating collaborative development.
- CI Build system continuously checks the compilation success on different architectures.
- Unit tests ensure functionality and validity of all building blocks.
- Goal: Automated benchmark runs every time new code is added.

Continuous Integration (CI)
A Healthy Software Development Cycle

- Version control system for tracking changes and coordinating collaborative development.
- CI Build system continuously checks the compilation success on different architectures.
- Unit tests ensure functionality and validity of all building blocks.
- Goal: Automated benchmark runs every time new code is added.

**Continuous Integration (CI)**
A Healthy Software Development Cycle

- Version control system for tracking changes and coordinating collaborative development.
- CI Build system continuously checks the compilation success on different architectures.
- Unit tests ensure functionality and validity of all building blocks.
- Goal: Automated benchmark runs every time new code is added.

**Security Thrust!**
- External code is run on HPC system
- Possible point of attack

**Continuous Integration (CI)**

**Developer**

**Source Code Repository**

**CMake**

**CI Build**

**Cl Test**

**CI Benchmark Tests**

**Schedule in Batch System**

**HPC System**
A Healthy Software Development Cycle

- Source Code Repository
- CI Build
- CI Test
- CI Benchmark Tests
- Trusted Reviewer
  - Code Review
  - Merge into Master Branch
  - Manual code reviews by trusted reviewers who take responsibility for executing code on the HPC system.

- HPC System

**Continuous Integration (CI)**
A Healthy Software Development Cycle

Continuous Integration (CI)

• Manual code reviews by trusted reviewers who take responsibility for executing code on the HPC system.
• How to archive the results and make them accessible?
A Healthy Software Development Cycle

Continuous Integration (CI)

- Source Code Repository
- CI Build
- CI Test
- CI Benchmark Tests
- Performance Data Repository

- Trust Reviewer
  - Code Review
  - Merge into Master Branch

- Store data in separate repository.
- Use JSON data exchange format.

Developer

Push

HPC System

Schedule in Batch System
A Healthy Software Development Cycle

Continuous Integration (CI)

- Source Code Repository
- CMake
- googletest
- gtest
- CI Build
- CI Test
- CI Benchmark Tests
- Schedule in Batch System
- HPC System
- Performance Data Repository
- Code Review
- Merge into Master Branch
- Trusted Reviewer

- Store data in separate repository.
- Use JSON data exchange format.
- Cloning a huge repository is expensive.
- "Interest in my experiments, only."
A Healthy Software Development Cycle

- Store data in separate repository.
- Use JSON data exchange format.
- Cloning a huge repository is expensive.
- "Interest in my experiments, only."
- Interactive web-based Ginkgo Performance Explorer (GPE).
A Healthy Software Development Cycle

Continuous Integration (CI)

1. Source Code Repository
2. CI Build
3. CI Test
4. Trusted Reviewer
   - Code Review
   - Merge into Master Branch
5. CI Benchmark Tests
6. Schedule in Batch System
7. Performance Data Repository
8. Web-Application
9. Ginkgo Performance Explorer
10. Users
11. Developer
12. Push
Ginkgo Performance Explorer (GPE)

Data Selection Tab

Transformation Script Editor

Data and Plot Viewer
1. Select Data in Git repository.

Ginkgo Performance Explorer (GPE)

Data Selection Tab

Transformation Script Editor

Data and Plot Viewer
Ginkgo Performance Explorer (GPE)

1. Select Data in Git repository.

2. Write JSONata script to visualize data (examples are provided).
Ginkgo Performance Explorer (GPE)

Data Selection Tab

1. Select Data in Git repository.

Transformation Script Editor

2. Write JSONata script to visualize data (examples are provided).

Data and Plot Viewer

3. Analyze data visually.
Ginkgo Performance Explorer (GPE)
Ginkgo Performance Explorer (GPE)
Ginkgo Performance Explorer (GPE)

Performance profile on V100_saturn(cuda)

Maximum slowdown factor over fastest

Dolan & More: Benchmarking optimization software with performance profiles
Ginkgo’s Performance Evaluation Framework

Continuous Benchmarking Benefits

• Archiving performance data along with execution parameters ensures full benchmark reproducibility.
• Comparing the performance results over the code lifetime identifies performance degradations.
• Ease of use: the setup allows to launch benchmark with few clicks.
Ginkgo’s Performance Evaluation Framework

Continuous Benchmarking Benefits

• Archiving performance data along with execution parameters ensures full benchmark reproducibility.
• Comparing the performance results over the code lifetime identifies performance degradations.
• Ease of use: the setup allows to launch benchmark with few clicks.

Ginkgo Performance Explorer (GPE) Benefits

• The design of GPE efficiently realizes the analysis as web service, removing the need for downloading performance data to local disk or installing additional software.
• External developers without access to HPC systems can test and engineer their codes on HPC resources.
• Extensibility: Option to compare performance with other software libraries.

Learn More about Ginkgo

- Open-source C++ framework for sparse linear algebra.
- Sparse linear solvers, preconditioners, SpMV etc.
- Generic algorithm implementation:
  + reference kernels for checking correctness;
  + architecture-specific highly optimized kernels.
- Focused on GPU accelerators (i.e. NVIDIA GPUs).
- Software quality and sustainability efforts guided by xSDK community policies:
  - open source
  - BSD
  - Doxygen
  - CMake
  - googletest
  - GitLab CI

https://xsdk.info/

https://ginkgo-project.github.io/

https://bssw.io/