Performance Dashboard for Continuous Benchmarking of HPC Libraries

Chingun Ariunbat, Maximilian Schik, Walter Alexander Böttcher, Darius Schefer, Jamil Bagga

2021-05-10

1 Introduction

Performance Dashboard for Continuous Benchmarking of HPC Libraries

PSE SS21

2 Goals

2.1 Required

heading NICHT IMPLEMENTIERT **M1**

yet more placeholder

Schnelle Weiterleitung Kurz- zu Lang-URL **M2**
Implementiert durch: FR1

Authentifizieren mit E-Mail oder Facebook **M3**
Implementiert durch: FR2

Rechtlichte Vorgaben werden eingehalten **M4**
Implementiert durch: FR3 FR4

template

2.2 Optional

Authentifizieren mit Github **K1**
Implementiert durch: FR2

Seite mit Betreiberinfo keine entsprechende Anforderung **K2**

template

2.3 Limitation

Keine Wahl Kurz-URL **A1**

template

3 Usage

template

4 Product Environment

template

5 Functional Requirements

Schnelle Weiterleitung NICHT GETESTET **FR1**
Implementiert: M2

template

template NICHT GETESTET **FR2**
Implementiert: M3 K1

template

Auf jeder Seite ist ein Link “Impressum” NICHT GETESTET **FR3**
Implementiert: M4

template

Auf jeder Seite ist ein Link “Datenschutz” NICHT GETESTET **FR4**
Implementiert: M4

template

Daten werden persistent gespeichert NICHT GETESTET **FR5**
Implementiert:

template

6 Nonfunctional Requirements

Modernes Design **NF1**

template

Persistenz **NF2**

template

Erweiterbarkeit **NF3**

template

7 Product Data

Input data: - performance data - **JSON/CSV formats** - problem (descr.) data - algorithm result data (time, storage, accuracy) - git history (of what? probably perf. data) - visualization data - save Templates (graph layouts) **JSON format** - ?: visualization history (per session)

Benchmark Results **PD1**

Generated by: Format: .JSON/.CSV,

Comparisons **PD2**

8 Tests

9 Scenarios

Scenario name: pushAndInspect
Participating actor: Ted: Developer

- Ted pushes his work to a git repository and fires off a benchmarkt test
- Ted opens the web app and selects his last pushed change
- Ted chooses a type of visualization
- The app creates the given type of visualization with the benchmark results from the selected change

Scenario name: visualizeFromTemplate
Participating actor: Greta: User

- Greta opens the web app
- Greta chooses a template for a visualization
- Greta chooses which commit she wants to visualize
- The app creates the given type of visualization with the commit

Scenario name: saveTemplate
Participating actor: Greta: User

- Greta opens the web app
- Greta configures a visualization
- Greta saves her visualization as a template for future use

Scenario name: inspect
Participating actor: Greta: User

- Greta wants to see the latest performance benchmarks for the project
- Greta opens the web app and selects the latest change
- Greta chooses a benchmark to compare

- Greta chooses a type of visualization by selecting which value to plot on the x axis and which value on the y axis
- The app creates the given type of visualization with the benchmark results from the selected change

Scenario name: compareImplementations
Participating actor: Greta: User

- Greta wants to know which implementation is the fastest
- Greta opens the web app and selects a benchmark
- Greta selects commits from different branches containing different implementations
- Greta chooses a type of visualization by selecting which value to plot on the x axis and which value on the y axis.
- The app creates the given type of visualization with the benchmark results from the selected change

Scenario name: pushAndCompare
Participating actor: Ted: Developer

- Ted pushes his work to a git repository, and fires off a benchmark test.
- Benchmark results are fed into the database.
- Ted opens the webapp and selects his last pushed change>
- Ted selects a previous change that he wants to compare to.
- Ted chooses a type of visualization.
- The app creates the given type of visualization with the benchmark results from the selected changes.

Scenario name: badPerformance
Participating actor: Ted: Developer

- Ted pushes his work to a git repository, and fires off a benchmark test.
- Benchmark results are fed into the database.

- Our dashboard-backend realizes that the benchmark data for this change is far worse than usual.

- Ted gets notified that his last pushed change significantly worsened the performance and the related details about that.

Scenario name: impossiblePerformance
Participating actor: Ted: Developer

- Ted pushes his work to a git repository, and fires off a benchmark test.

- Benchmark results are fed into the database.

- Our dashboard-backend realizes that the benchmark data for this change is theoretically impossible.

- Ted gets notified that his last pushed change has improved the performance above the theoretical maximum and the related details about that.

Scenario name: shareVisualization
Participating actor: Greta: User

- Greta found an interesting visualization for something.

- Greta clicks a *share* button next to the visualization.

- Greta gets a link she can share with others that redirects them to the exact same visualization.

Scenario name: visualizeCommitWithoutBenchmark
Participating actor: Greta: User

- Greta opens the web app and wants to visualize benchmarkdata for a specific commit. This commit has no benchmark data attached to it, only the commit before and the commit after.

- Greta can't click on the commit because it is greyed out.

Scenario name: takeVisualizationFromHistory
Participating actor: Greta: User

- Greta opens the webapp and visualizes something. She then visualizes something else. Her previous visualizations are stored in a list somewhere.

- Greta decides to take another look at a previous visualization.
- Greta picks her previous visualization and gets the previous visualization.

Scenario name: postBenchmarkResults
Participating actor: bencharkCI: CI

- The benchmarkCI processes a benchmark and gets some results.
- The benchmarkCI posts the results to the backend of the system using the API supplied by the system.
- The benchmark results are stored in the backend database system.

10 Use Cases

Use case name: Visualize
Participating actors: initiated by User
Entry conditions: configuration is available
Flow of events:

1. Web app fetches the data specified in the configuration from the backend.
2. The web app uses the fetched data to generate a plot according to the specifion.
3. The generated plot is visualized in the web app.

Exit conditions: The plot specified by the configuration gets shown to the User.
Quality requirements: Shouldn't take more than 30 seconds?

Glossary

CI Continuous Integration.

Developer Person working on the project that is to be benchmarked.

template Configuaration of a visualization.