Performance Dashboard for Continuous Benchmarking of HPC Libraries

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

2021-05-13

Contents

1 Introduction **3**

2 Goals **3**

2.1 Required . . . 3

2.2 Optional . . . 3

2.3 Limitation . . . 3

3 Usage **3**

4 Product Environment **4**

5 Functional Requirements **5**

6 Nonfunctional Requirements **5**

7 Product Data **5**

8 User Interface **6**

9 Tests **7**

10 Scenarios **8**

11 Use Cases **12**

Glossary **19**

1 Introduction

Performance Dashboard for Continuous Benchmarking of HPC Libraries

PSE SS21

2 Goals

2.1 Required

Criteria-Template **C1**
Implemented By: FR1

template

2.2 Optional

Optional-Criteria-Template no according requirement **OC1**

template

2.3 Limitation

Non-Criteria-Template **NC1**

template

3 Usage

template

4 Product Environment

template

5 Functional Requirements

functionality template with “smthn in quotes” NOT TESTED **FR1**
Implements: C1

template

6 Nonfunctional Requirements

non-functionality template **NF1**

template

7 Product Data

Benchmark Results (Name in progress) **PD1**

Format: JSON/CSV
Description:

- saved on server
- algorithm result data (time, storage, accuracy, convergence(?))

Git Histories **PD2**

Format: ??? (WIP)

Templates **PD3**

Format: JSON (?)

8 User Interface

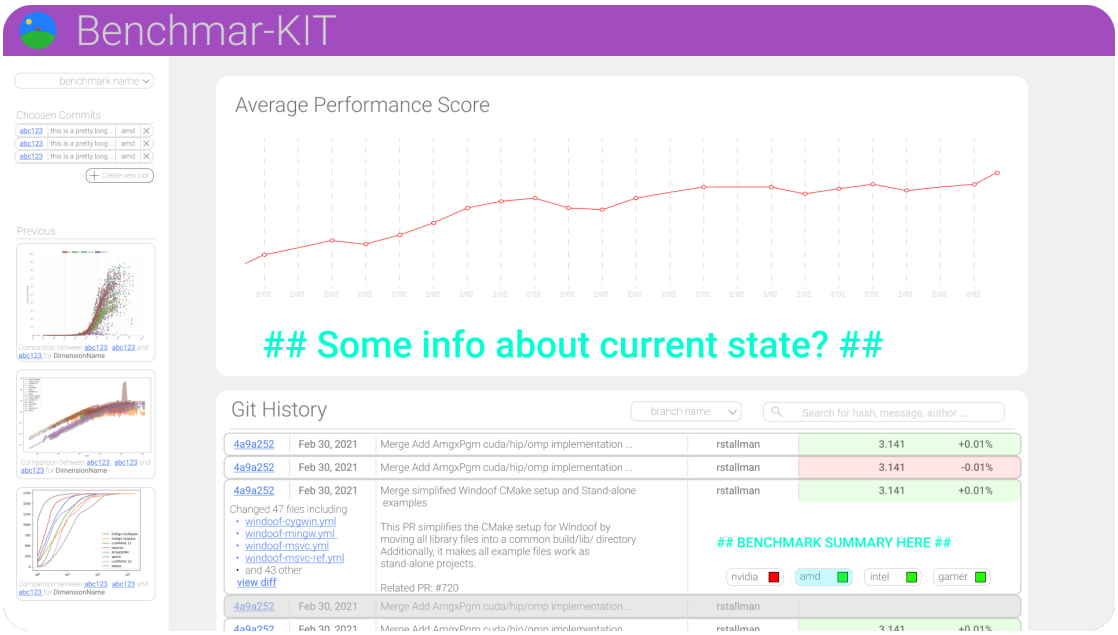

Figure 1: Main page

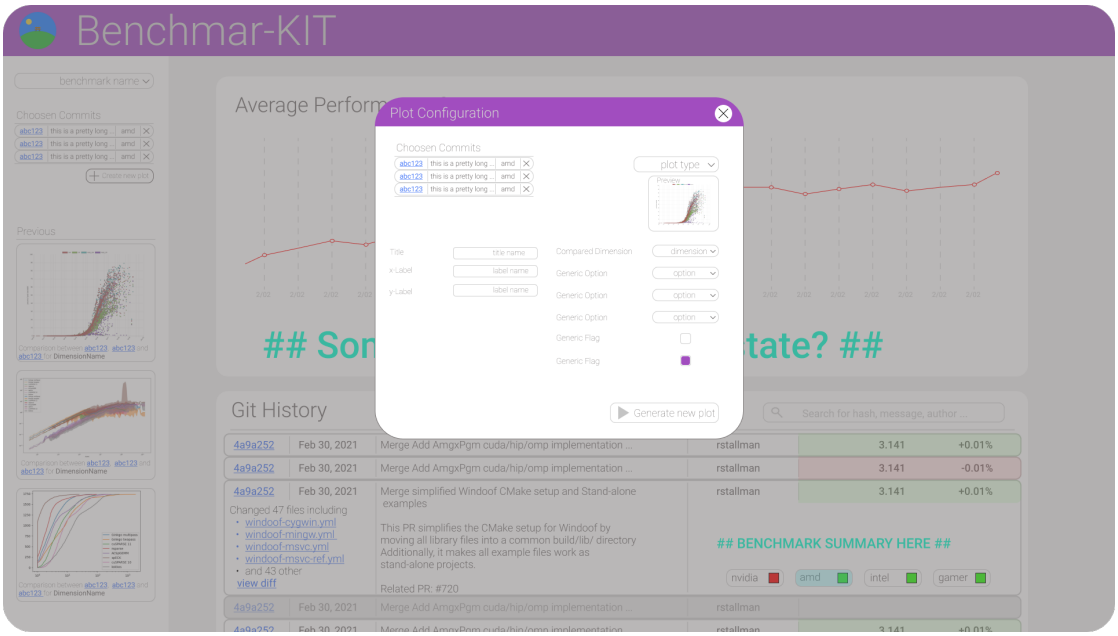

Figure 2: Configuration Popup

9 Tests

Test Template Name **T1**

Tests:

something that should be tested

10 Scenarios

Scenario name: pushAndInspect
Participating actor: Ted: Developer

- Ted pushes his work to a git repository and fires off a benchmark 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 web app 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 benchmark data 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 web app 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.

11 Use Cases

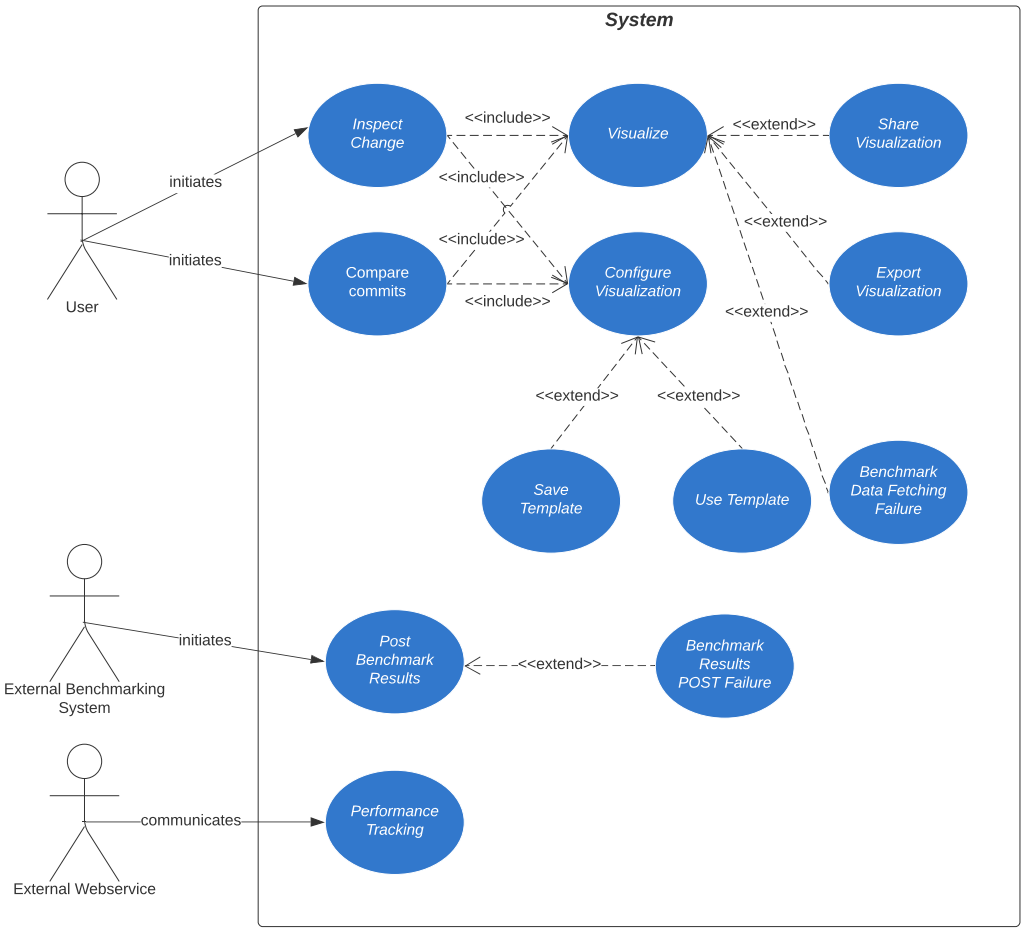

Figure 3: Use case diagram

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

1. The web app sends a request to the backend containing the configuration.
2. The backend fetches the specified data from a databank.
3. The backend does the calculations specified in the configuration (mean, median, standard deviation).
4. The backend sends the data back to the webapp.
5. The webapp takes the data and generates the plot specified in the configuration.
6. The user gets redirected to a new site where he can inspect the generated plot.

Exit conditions: The plot specified by the configuration gets shown to the User.
Quality requirements: Shouldn't take more than 10 seconds

Use case name: Configure Visualization
Participating actors: initated by User
Entry conditions: User selected the "Create New Plot" option
Flow of events:

1. A popup appears.
2. The user chooses a plot type.
3. The user chooses between certain options that are specific to the plot type.

Exit conditions: A configuration gets created.
Quality requirements: The available options should be understandable without any previous knowledge or otherwise described by a short text.

Use case name: Inspect Change
Participating actors: initated by User
Entry conditions: benchmark data for commit is available
Flow of events:

1. The user selects a single commit.
2. The user initiates the `Configure Visualization` use case by selecting the “Create New Plot” option.
3. Once the user is satisfied with his configuration, he initiates the `Visualize` use case by selecting the “Create New Plot” option in the popup.

Exit conditions: Visualization is displayed to the user
Quality requirements: ???

Use case name: Compare commits
Participating actors: intiated by User
Entry conditions: benchmark results for all selected commits are available
Flow of events:

1. The user selects multiple commits.
2. The user initiates the `Configure Visualization` use case by selecting the “Create New Plot” option.
3. Once the user is satisfied with his configuration, he initiates the `Visualize` use case by selecting the "Generate New Plot" option in the popup.

Exit conditions: Visualization is displayed to the user
Quality requirements: ???

Use case name: Share Visualization
Participating actors: initiated by User
Entry conditions: A visualization has been generated
Flow of events:

1. The user selects the “Share Visualization” option.
2. A link gets displayed.
3. The link redirects any visitors to the same visualization.

Exit conditions: A link is shown which redirects to the visualization
Quality requirements: ???

Use case name: Export Visualization
Participating actors: initiated by User
Entry conditions: A visualization has been generated
Flow of events:

1. The user selects the “Export Visualization” option.
2. A popup appears.
3. The user chooses a filetype for the export.
4. The user confirmes and downloads the visualization in the choosen file format.

Exit conditions: The User is offered a download of an export of the visualization
Quality requirements: Support for the filetypes png, pdf and pgf (what is the preferred latex format?)

Use case name: Save Template
Participating actors: initiated by User, (maybe web browser as well? the cookies get stored on the web browser)
Entry conditions: The user is in the `Configure Visualization` use case
Flow of events: The `Save Template` use case extends the `Configure Visualization` use case.

1. The User selects the “save template” option.
2. The User enters a name for the template.
3. The webapp stores the template locally (cookies).

Exit conditions: Template is stored on the locally (might add global templates for later?)
Quality requirements: Requires less than 1kB of memory

Use case name: Use Template
Participating actors: initiated by User (maybe web browser as well? the cookies get stored on the web browser)
Entry conditions: The user is in the `Configure Visualization` use case and a template is available locally
Flow of events: The `Use Template` use case extends the `Configure Visualization` use case.

1. The User selects the "use template" option.
2. User is shown a list of all available templates.
3. User selects a template from the list.
4. The current configuration options get set to the values specified in the template.

Exit conditions: The template is applied to the current configuration.
Quality requirements: ???

Use case name: Post Benchmark Results
Participating actors: initiated by External Benchmarking System
Entry conditions: The external benchmarking system ran the benchmarks
Flow of events:

1. The benchmarking system makes a POST request to the backend containing the new benchmark data in JSON format.
2. The backend converts the received data into the correct format.
3. The backend stores the received data in a database.

Exit conditions: The received performance data is stored in a database
Quality requirements: ???

Use case name: Performance Tracking
Participating actors: communicates with External Webservice
Entry conditions: New benchmark data has been posted to the backend
Flow of events:

1. The backend evaluates the performance of the new benchmark data.
2. The backend compares the performance of the new benchmark with the performance of the corresponding benchmark of the last commit.
3. The backend relays the results to a configured number of hooks.
4. The hooks contact their external webservices according to how they have been configured.

Exit conditions: The server fires a POST Request to all webhook subscribers
Quality requirements:

Use case name: Benchmark Results POST Failure
Participating actors: return error code to External Benchmarking System
Entry conditions:
Flow of events: This use case extends the `Post Benchmark Results` use case if an error occurs.

1. The backend identifies the error.
2. The backend creates a response with the correct error code.

Exit conditions:
Quality requirements:

Use case name: Benchmark Data Fetching Failure
Participating actors: displays error to User
Entry conditions: This use case extends the `Visualization` use case if an error occurs.
Flow of events:

1. The backend identifies the error.
2. The backend creates a response with the correct error code.
3. The webapp displays an error message.

Exit conditions:
Quality requirements:

Use case name: UseCaseTemplate
Participating actors: Actors
Entry conditions: Entry cond.
Flow of events:

1. Flow 1
2. Flow 2

Exit conditions: Exit cond.
Quality requirements: Quality Requirements(?)

Glossary

CI Continuous Integration.

configuration A complete description of a visualization. It contains all the necessary information except the benchmark data.

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

JSON JavaScript Object Notation.

template A partial configuration of a visualization, It contains preconfigured values, but leaves others blank for the user to costumize.

visualization A graphical representation of benchmark data.