

# **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

<b>heading</b>	NICHT IMPLEMENTIERT	<b>M1</b>
yet more placeholder		
<b>Schnelle Weiterleitung Kurz- zu Lang-URL</b>		<b>M2</b>
Implementiert durch: <a href="#">FR1</a>		
<b>Authentifizieren mit E-Mail oder Facebook</b>		<b>M3</b>
Implementiert durch: <a href="#">FR2</a>		
<b>Rechtliche Vorgaben werden eingehalten</b>		<b>M4</b>
Implementiert durch: <a href="#">FR3</a> <a href="#">FR4</a>		
template		

### 2.2 Optional

<b>Authentifizieren mit Github</b>		<b>K1</b>
Implementiert durch: <a href="#">FR2</a>		
<b>Seite mit Betreiberinfo</b>	keine entsprechende Anforderung	<b>K2</b>
template		

### 2.3 Limitation

<b>Keine Wahl Kurz-URL</b>		<b>A1</b>
template		

### **3 Usage**

template

### **4 Product Environment**

template

## 5 Functional Requirements

<b>Schnelle Weiterleitung</b>	NICHT GETESTET	<b>FR1</b>
Implementiert: M2		
template		
<b>template</b>	NICHT GETESTET	<b>FR2</b>
Implementiert: M3 K1		
template		
<b>Auf jeder Seite ist ein Link "Impressum"</b>	NICHT GETESTET	<b>FR3</b>
Implementiert: M4		
template		
<b>Auf jeder Seite ist ein Link "Datenschutz"</b>	NICHT GETESTET	<b>FR4</b>
Implementiert: M4		
template		
<b>Daten werden persistent gespeichert</b>	NICHT GETESTET	<b>FR5</b>
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 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.

## 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 specification
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?

**Use case name:** Compare commits

**Participating actors:** Server

**Entry conditions:** benchmark results for all selected commits are available

**Flow of events:**

1. The backend compares the performance of all selected commits
2. ???

**Exit conditions:** The server caches the results for later use

**Quality requirements:** wip

**Use case name:** InspectChange

**Participating actors:** initiated by User

**Entry conditions:** benchmark data for commit is available

**Flow of events:**

1. User selects a single change from the history
2. Server generates comparison -> `CompareCommits`

**Exit conditions:** Results are displayed in the web app

**Quality requirements:** If data is present: no more than 30 seconds?

**Use case name:** SaveTemplate

**Participating actors:** User, Server

**Entry conditions:** Web app is currently displaying data

**Flow of events:**

1. User selects the “save template” option
2. User is prompted for name for the [template](#)
3. The Server generates a [JSON](#) file representing the [template](#)

**Exit conditions:** [Template](#) is stored on the server for later use

**Quality requirements:** Requires less than 1kB of storage in the database

**Use case name:** UseTemplate

**Participating actors:** User, Server

**Entry conditions:** A [template](#) is stored in the database

**Flow of events:**

1. The User selects the “open template” option
2. User is shown a list of all saved [templates](#)
3. User selects a [template](#) from the list

**Exit conditions:** The User’s dashboard is set to the selected template

**Quality requirements:** ???

**Use case name:** PostBenchmarkResults

**Participating actors:** [Developer](#), Server

**Entry conditions:** The application was benchmarked

**Flow of events:**

1. The [developer](#) makes a POST request to the server containing the new benchmark data in [JSON](#) format
2. The Server converts the received data into the correct format
3. The Server stores the received data in the database

**Exit conditions:** The received performance data is stored in the database

**Quality requirements:** ???

**Use case name:** ShareVisualization

**Participating actors:** User

**Entry conditions:** A [visualization](#) exists already

**Flow of events:**

1. The User selects the “share visualization” option on the web app dashboard with the [visualization](#) on it
- 2.

**Exit conditions:**

**Quality requirements:**

**Use case name:** ExportVisualization

**Participating actors:** User

**Entry conditions:** A [visualization](#) exists already

**Flow of events:**

1. The User selects the “export visualization” option on the web app dashboard with the [visualization](#) on the dashboard
2. The web app serializes the data as a [JSON](#) file
3. The web app prompts a download of the serialized [JSON](#) file

**Exit conditions:** The User is offered a download of an export of the visualization

**Quality requirements:** This may not take longer than 30 seconds. (Probably.)

**Use case name:** ImportVisualization (probably)

**Participating actors:** User

**Entry conditions:**

**Flow of events:**

1. The user selects the “import visualization” option on the web app
2. The web app prompts the user to upload a JSON File. (or to paste it?)
3. The web app parses the data within the JSON File, and displays it on the web app.

**Exit conditions:**

**Quality requirements:**

**Use case name:** Notify

**Participating actors:** Server, User

**Entry conditions:** A comparison has finished (see compare commits)

**Flow of events:**

1. The server has finished the comparison of the latest commit with benchmark data with its predeccessing commit with benchmark data.
2. The server fires a POST Request to all of its webhook subscribers, containing a notification that the latest change was negatively impacting the performance, a bried description of how it did so, and a link to the web app for immediate review.
3. The user sees this notification on its webhook service (e.g. Discord).

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

**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.

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

**JSON** JavaScript Object Notation.

**template** Configuration of a [visualization](#).

**visualization** (WIP) uhhh... graphs and stuff on one website?.