JMH Reader

Giordano Tinella

Advanced Verification and Validation (2024)

Introduction

The goal

The goal of this project is to create an environment that allows users to upload JSON files generated by JMH and visualize the summarized data for each benchmark added to the system. This will enable users to easily review all the information related to specific commits.

Java Microbenchmark Harness (JMH)

JMH (Java Microbenchmark Harness) is a framework developed by the OpenJDK community for creating, running, and analyzing microbenchmarks in Java. It is specifically designed to measure the performance of small code snippets or components within the Java Virtual Machine (JVM).

Instructions

Working with JMH

First, you need to generate the JSON files for a specific benchmark. To do this, you can use the following GitHub repository: https://github.com/eclipse/eclipse-collections. I recommend using version 10.0.0. Once you have cloned the repository, set up the environment by following these steps:

Install dependencies:

- mvn install -DskipTests -am -pl eclipse-collections,eclipse-collections-api,eclipse-collections-forkjoin,eclipse-collections-testutils
- mvn install -DskipTests -am -f jmh-scala-tests/pom.xml

Build jmh-tests:

- mvn package -DskipTests -am -f jmh-tests/pom.xml
- list the available benchmarks: java -jar jmh-tests/target/microbenchmarks.jar -l

Generate the JSON file:

- java -jar jmh-tests/target/microbenchmarks.jar org.eclipse.collections.impl.jmh.LongIntMapTest.put -rf json -wi 1 -r 1 -f 1 -i 1 -bm ss
- The previous command will generate a JSON file named “jmh-result”

* The JSON generation can be customized to meet your needs (the benchmark used for this example is “LongIntMapTest”, but you can use any benchmark you prefer). This repository includes a PDF named “parameters” that lists all the available options for generating the JSON.

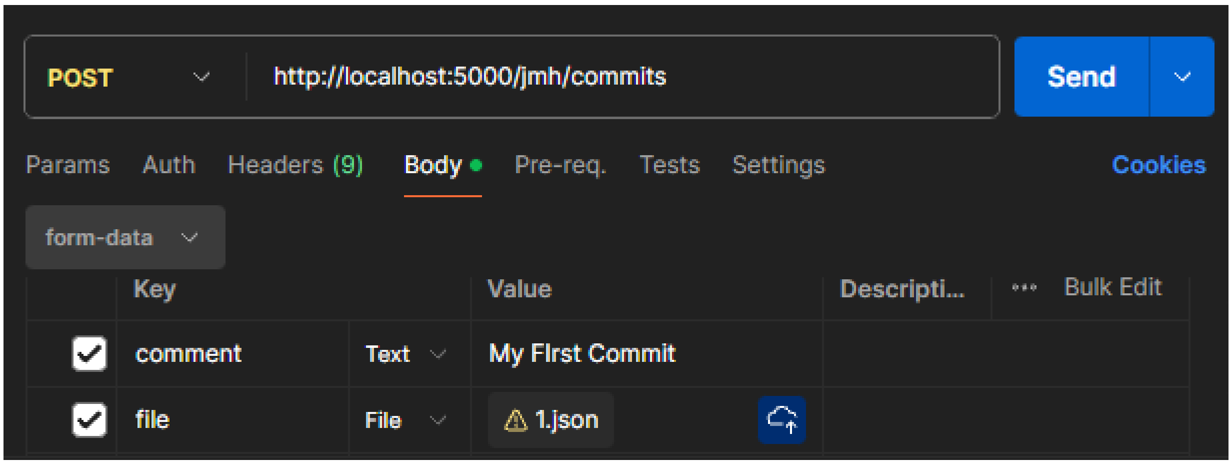
Working with JMH-Reader

Run the app:

- Clone this repository
- Run the command "docker compose up -d" in the root directory
- Wait a few minutes for the app to start up

Add a commit:

- DO an http request POST on "http://localhost:5000/jmh/commits", the payload should contain:
    - comment (a comment for the commit)
    - file (the json file generated using JMH)



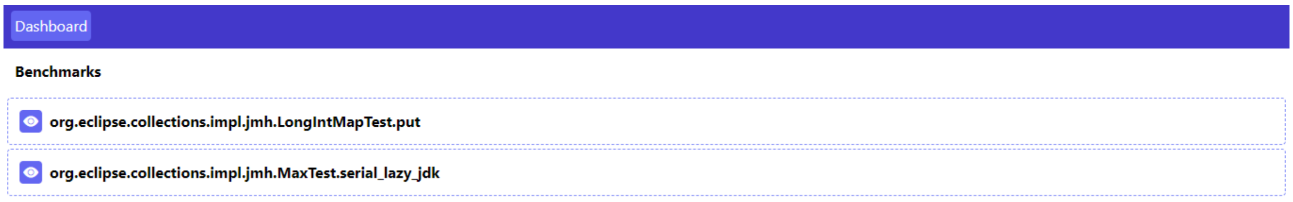
	Key		Value	Descripti...
✓	comment	Text	My FIrst Commit	
✓	file	File	1.json	

Visualize the dashboard:

- Go to "http://localhost:5001"

Into The APP

Dashboard



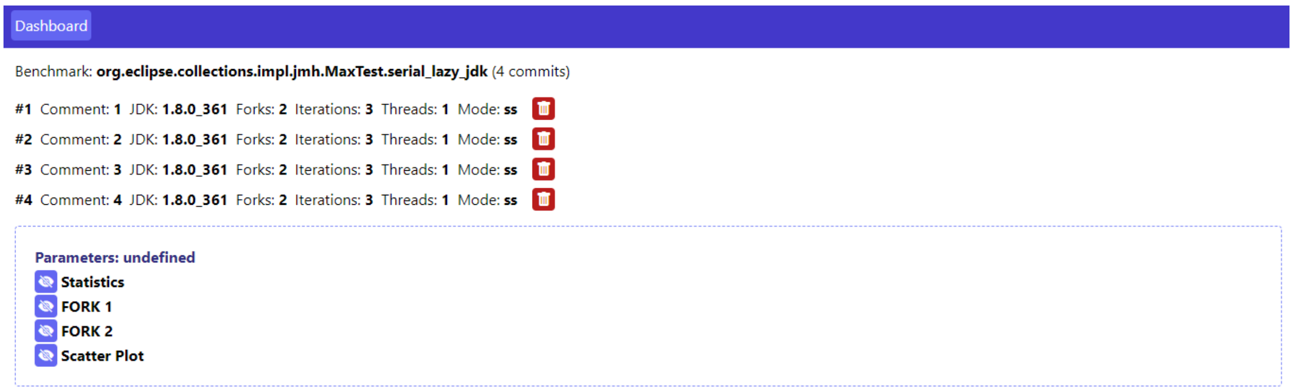
Dashboard

Benchmarks

org.eclipse.collections.impl.jmh.LongIntMapTest.put

org.eclipse.collections.impl.jmh.MaxTest.serial_lazy_jdk

Benchmark



Dashboard

Benchmark: **org.eclipse.collections.impl.jmh.MaxTest.serial_lazy_jdk** (4 commits)

#1 Comment: **1** JDK: **1.8.0_361** Forks: **2** Iterations: **3** Threads: **1** Mode: **ss**

#2 Comment: **2** JDK: **1.8.0_361** Forks: **2** Iterations: **3** Threads: **1** Mode: **ss**

#3 Comment: **3** JDK: **1.8.0_361** Forks: **2** Iterations: **3** Threads: **1** Mode: **ss**

#4 Comment: **4** JDK: **1.8.0_361** Forks: **2** Iterations: **3** Threads: **1** Mode: **ss**

Parameters: undefined

Statistics

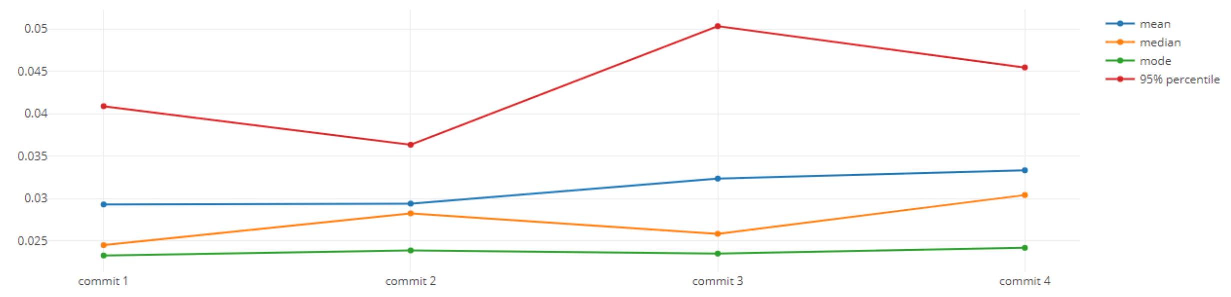
FORK 1

FORK 2

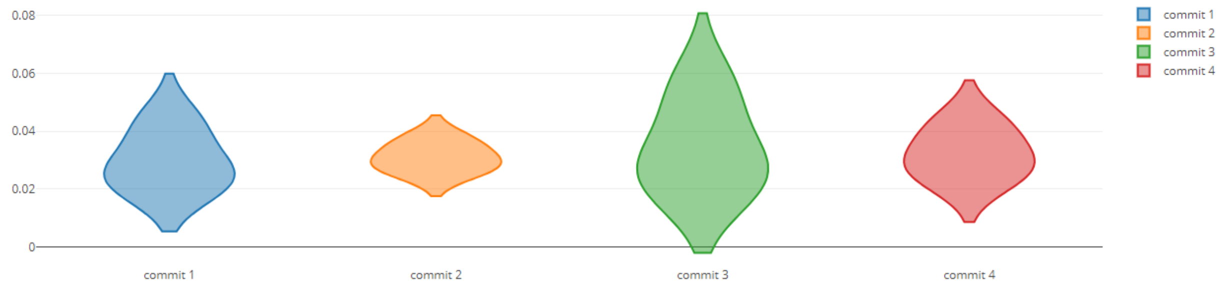
Scatter Plot

The data:

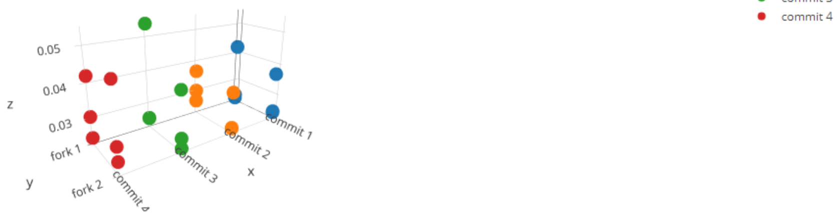
Statistics



FORK 1



Scatter Plot


z
fork 1
fork 2
y
commit 1
commit 2
commit 3
commit 4
x
● commit 1
● commit 2
● commit 3
● commit 4