Thesis

Progress Report 2

Pieter De Clercq

February 16, 2020

GHENT UNIVERSITY

FACULTY OF SCIENCES

Metadata

The title of my thesis is still provisionary.

- **Title:** Applying Test Case Prioritization to optimize Continuous Integrating
- **Name:** Pieter De Clercq `(01503338)`
- **Study programme:** Master of Science in Computer Science
- **Promotor(s):** prof. dr. Bruno Volckaert, prof. dr. ir. Filip De Turck
- **Supervisors:** Jasper Vaneessen, Dwight Kerkhove

Status

In the previous progress report, I have determined the scope of my masters' thesis. As outlined before, my masters' thesis will research whether or not it is possible to optimize Continuous Integration performance by adopting Test Case Prioritization. In essence, this means that I want to reduce the time occupied by executing regression, integration and unit tests by ranking them according to their failure expectation probability.

First of all, I could not spend as much time to my masters' thesis as I had wanted during December/January, as a result of my busy exam schedule. Since this is my graduation year, I wanted to ensure myself not to fail any exams, a task at which I succeeded. Besides studying however, I was able to find some free moments at which I could progress my thesis.

My first goal was to finish the literature study I had started in the first semester and write a textual report. I have not yet finished this report, however I expect to do so by next week (my textual report currently consists of 21 pages).

Secondly, I had to think of an architecture and start coding an implementation. As to the architecture part, I have devised a simple web-based API as outlined in Figure 1. I have opted for a web API since this allows data to be exposed easily, optionally for other tools to take advantage of in the future. The architecture works as follows: First, the Continuous Integration system `[CI-server]` sends a request to the prioritization server `[PRIO-server]`, indicating that a new commit is ready to be tested.

Subsequently, the commit is pulled from the remote repository by the PRIO-server, after which an order is determined. This analysis is performed asynchronously, since it can span several seconds. The CI-server frequently polls the PRIO-server for status updates regarding the analysis. When this is complete, the CI-server receives an order in which the tests should be executed. The tests are executed in the given order and the outcome of the tests is sent back to the PRIO-server as feedback, to improve subsequent analyses. As for the implementation, I have currently finished a proof-of-concept JUnit runner which allows tests (in Java) to be executed in a self-chosen order. The order can currently be provided using a `YAML`-file, as illustrated in Listing 1.

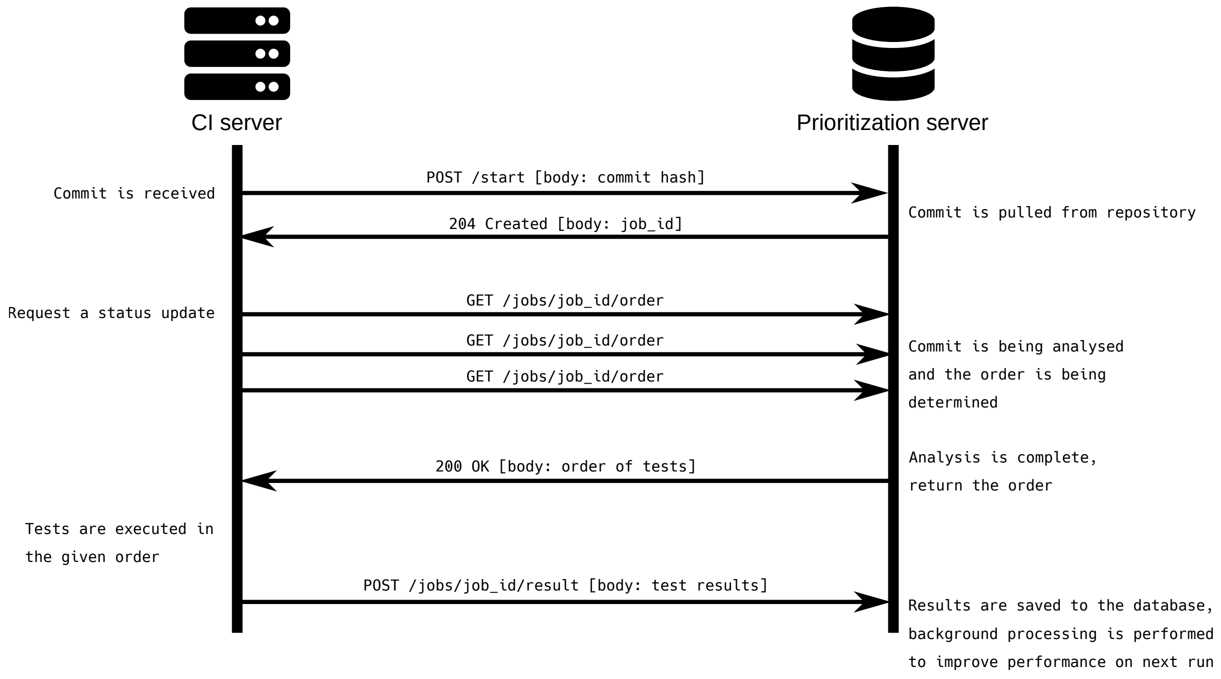
CI server
Prioritization server
Commit is received
POST /start [body: commit hash]
Commit is pulled from repository
204 Created [body: job_id]
Request a status update
GET /jobs/job_id/order
GET /jobs/job_id/order
GET /jobs/job_id/order
Commit is being analysed and the order is being determined
200 OK [body: order of tests]
Analysis is complete, return the order
Tests are executed in the given order
POST /jobs/job_id/result [body: test results]
Results are saved to the database, background processing is performed to improve performance on next run

Figure 1: Architecture design of the implementation

Listing 1: JUnit Order format

```
- {class: "FirstTest", method: "testHello"}
- {class: "SecondTest", method: "testWorld"}
- {class: "SecondTest", method: "testLoremIpsum"}
- {class: "FirstTest", method: "testFoo"}
- {class: "SecondTest", method: "testBar"}
```

Schedule

This semester, I only have to attend 1 course, which means that I can dedicate most, if not all of my time towards writing my masters' thesis. Until now I have mainly focussed on writing the textual report, my first task will be to finish the literature study. Starting from next week, I will start implementing the architecture discussed in the previous section. I do not expect the Prioritization server to be very resource intensive (performance testing is however required), so I will attempt to deploy it on a Raspberry Pi. Afterwards, I will feed some open source projects into this architecture, in order to evaluate its performance. In the previous progress report I had already identified three potential projects that I wanted to use. Finally, I will write a detailed report of both the implementation, the results, and future work and append this to the literature study. Concerning these sections, I have already written some schematic pseudo-sentences containing the basic ideas of these sections.

In a final phase, I will investigate the feasibility to support other languages beside Java, and other frameworks beside JUnit, such as Python.