CMPSC 441 Final Project

An Empirical Study of Distributed Test Suite Execution

Michael Camara

Gary Miller

Herbie Torrance

Colton McCurdy

Honor Code Pledge: This work is ours unless otherwise cited

4-11-16

Abstract

Testing software is important for ensuring correctness of any system, but it is often the case that test suites take a prohibitive amount of time to complete. We will analyze a distributed approach for executing test cases, with the goal of achieving improvements in overall response time. We will develop a novel system for accomplishing this task, in addition to analyzing an existing system called Test Load Balancer. We will collect and analyze data from both systems so that we can make inferences as to the trade-offs of distributed test suite execution.

1 Overview

In order to assess the effectiveness of distributed test suite execution, we plan to build a novel system and compare it with an existing system called Test Load Balancer. Our system will utilize Java RMI and a FileZilla FTP server to allow communication between a client, a registry, and multiple servers, each located on separate nodes. The node with the registry, hereafter known as the delegator, will host the FTP server and maintain a local directory of resources that will be accessible via this server. The client will communicate with the

delegator via Java RMI and send a list of `.class` test files, as well as all resources that serve as dependencies for the collection of tests. The delegator will store the resources in its local filesystem, accessible via the FTP server, and then begin to send the test files to servers for further processing, hereafter known as test servers. Ideally, the delegator will utilize a load sharing approach to distribute these test files evenly to all connected test servers, waiting until a test server finishes running a test before giving it a new one. The test servers will each utilize a custom class loader that targets the appropriate directories in the FTP server, and then use JUnit to execute the passed test file. This will allow the test servers to only access the files they need to run their respective tests from the FTP server, without needing to copy every file to their local filesystems. The result from running each test will be sent back to the delegator, and then back to the client, thereby completing the distributed test suite execution process.

Work in distributing Java test suites has been attempted by already existing systems, including a tool named Test Load Balancer (TLB) that we found on GitHub (*https://github.com/test-load-balancer/tlb*). This tool is able to distribute JUnit tests to multiple nodes. These nodes can be multiple threads on a local machine, multiple virtual machines, or multiple physical machines. It is capable of evenly distributing tests to these nodes by looking at the results from previously run tests. It looks at this data so that it is able to best prune, order and distribute the tests in future runs. The results from the previously run tests are stored on the server, which is referenced upon calling the load balancer. This is, of course, a brief overview and introduction to TLB tool.

In this project we plan to compare the runtime of running test suites locally, through the TLB tool and finally against Michael's test load sharer. Based upon these results, we will be able to inform readers of our final project the best way to run test suites in a distributed fashion. We will take advantage of existing systems with adequate test suites rather than writing a system to test. There are a number of Java projects on GitHub with adequate test suites. We will run a number—large enough to gather adequate results—of these projects' test suites against the aforementioned systems and compare the runtimes.

A significant portion of our novel system has already been implemented; but if we encounter any significant difficulties then we do have a backup plan to pursue. This plan

consists of taking advantage of the many features of the TLB tool, its ability to run a test suite in a distributed fashion on multiple threads on a local machine, multiple virtual machines, or multiple physical machines. We would compare the runtimes of these forms of distribution of the tests to that of running the test suite locally.

2 Work Distribution

- Michael Camara - Michael will implement a system for test suite distribution. Additionally, Michael will write about this system in our final report.
- Colton McCurdy - Colton will run the existing TLB tool in the multiple accepted configurations to collect and analyze data about runtimes. Colton will produce visualizations of these results using the R programming language.
- Herbie Torrance - Herbie will run the test suites of the existing projects on GitHub, locally and analyze those results, as well as results obtained from running our novel system.
- Gary Miller - Gary will read and write about the TLB tool and any other tools that he finds will contribute to this project. Gary will also write about test suite distribution in general.