OptiRun

A Platform for Optimized Test Execution in Distributed Environments

MASTER'S THESIS JANICKE FALCH JUNE 2016

INTRODUCTION

Our society is becoming increasingly dependent on computers and digital media. The importance of, and demand for, high-quality software has become substantial. Pressure on software vendors to deliver frequent releases of quality software requires efficiency in every stage of the development process. Software testing is an important part of this process, as it serves to provide quality assurance and defect detection as well as ensuring that the test object ~~behaves according to the specifications~~. With test automation, software testing can be performed rapidly, precisely and repeatedly while using minimal resources.

The telecommunications company *Altibox AS* has long wished to incorporate user-level test automation in the testing process of their online streaming service *TV Overalt*, but has failed to make it a priority until now.

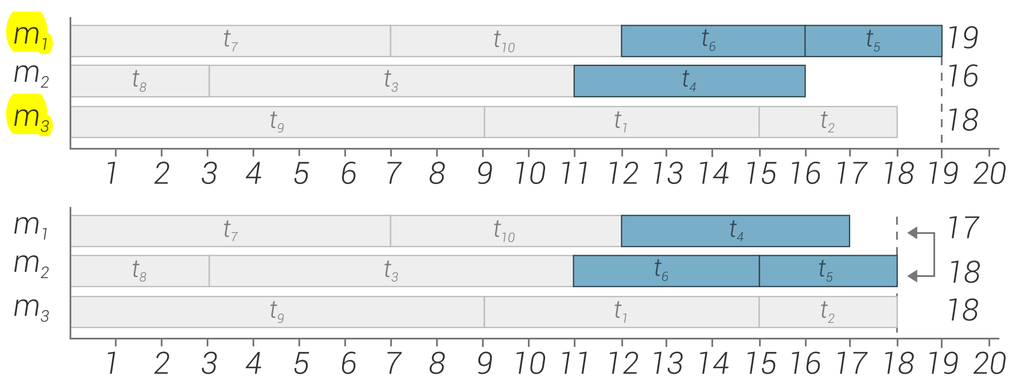
This thesis presents *OptiRun* - a platform where Altibox can run parallel tests in a distributed system. User-level tests generally run slowly, so the tool includes a mechanism for allocating tests to machines in such a way that overall execution time of a test suite is minimized. OptiRun can be operated from a web-based user interface, which has been created as part of the thesis.

METHODOLOGY

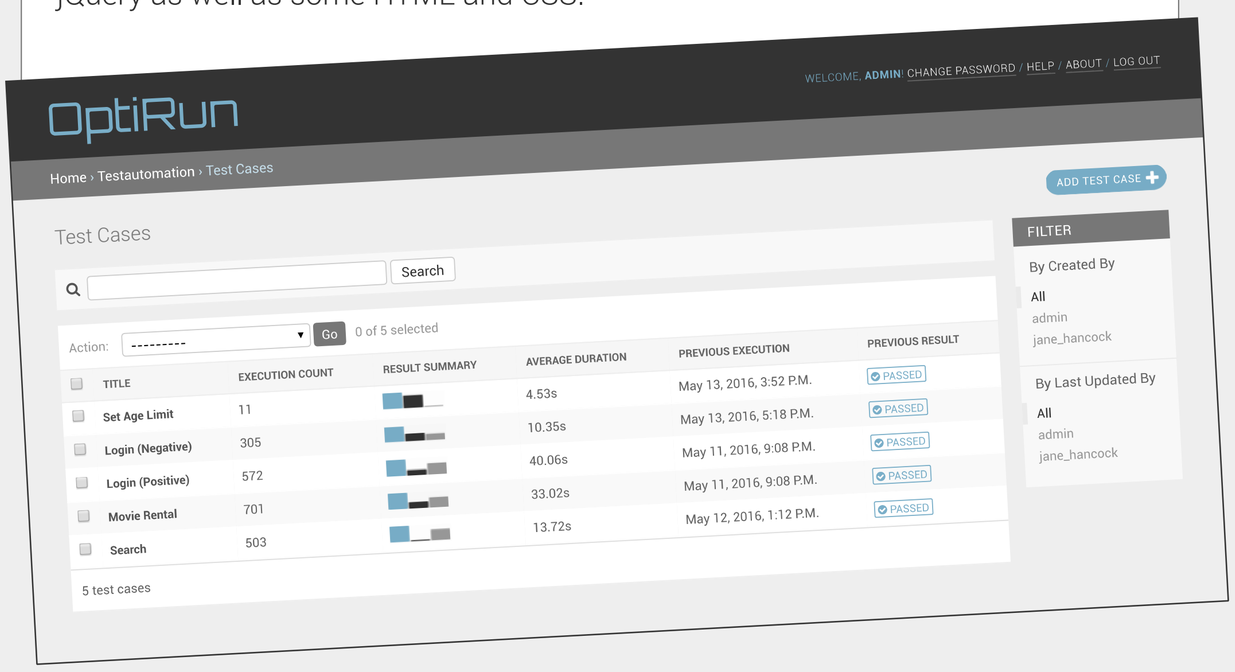
OptiRun consists of two main parts; a controller, which takes care of test allocation, execution and result reporting, and a web application where users can upload and manage test scripts, request test executions, view execution results and report failed test executions to the issue tracking system *JIRA*.

This project was written in the simple, but powerful high-level programming language Python. *Selenium*, a software testing framework for web applications, was used for test execution, and the accompanying *Selenium Grid* was incorporated to allow for remote execution in distributed environments.

To minimize the overall execution time of a collection of tests, they must be carefully allocated to machines in the distributed system. An allocation mechanism which has been named *OptiX* takes care of this. The tests are first strategically sorted before being allocated using a greedy algorithm. After this initial allocation, OptiX attempts to improve the result by identifying two subsets of tests currently allocated to two different machines, that when swapped will reduce the overall execution time. This improvement step is repeated until OptiX can no longer find an improvement or the mechanism times out. The time taken to allocate the tests is also taken into consideration. The two figures below shows the allocation state before and after the conduction of such an improvement. Note that the durations of the tests used in this example are artificially short.



OptiRun's web-based user interface was built on the Python Web framework *Django*, which allows for rapid development and seamless interaction with the rest of the system. It was mainly written in Python, but also includes elements of jQuery as well as some HTML and CSS.



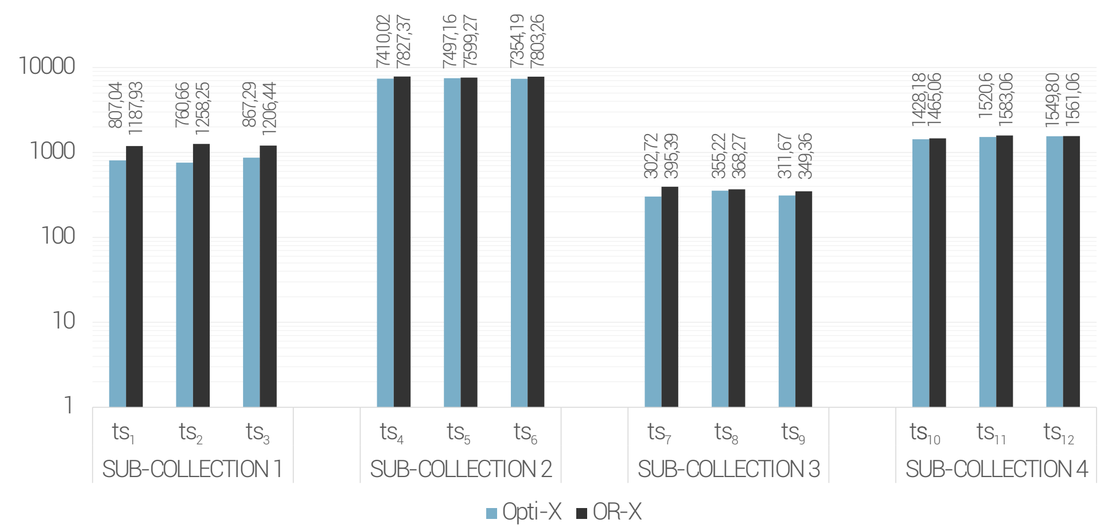
RESULTS

In order to measure and evaluate the performance of the test allocation mechanism OptiX, an alternative mechanism was also implemented. This was done using *OR-tools*, Google's library for combinatorial programming and constraint optimization. This alternative version was named *ORX*.

A collection of random generated test data were created, which formed four sub-collections of similar data, each consisting of three+++ test sets, and representing a distinct combination of test cases and test machines. In each test set, the test cases can be executed on a given number of machines, which means that there is a varying number of combinatorial solutions. Which specific machines each test case could be executed on were determined at random. The durations of the test cases were generated as random values between 30 and 120 seconds. Details about the test collection is displayed in the table below:

	# OF TESTS	# OF MACHINES	# OF MACHINES TESTS ARE EXECUTABLE ON
SUB-COLLECTION 1	1000	100	ts_1: 100, ts_2: 10, ts_3: *Random*
SUB-COLLECTION 2	1000	10	ts_4: 10, ts_5: 5, ts_6: *Random*
SUB-COLLECTION 3	200	50	ts_7: 50, ts_8: 10, ts_9: *Random*
SUB-COLLECTION 4	200	10	ts_{10}:10, ts_{11}:5, ts_{12}:*Random*

All 12 test sets in the collection were run with ORX to create a benchmark, and with OptiX for comparison and evaluation purposes. The results are visualized in the graph below, which shows the combined time used to allocate the tests and to execute them. Note that the y-axis uses a logarithmic scale due to large variation in numbers. As the graph shows, OptiX obtained better results than the benchmark values provided by ORX for all of the 12 test sets.



CONCLUSION

This thesis presents OptiRun: a platform for optimized test execution in distributed environments. OptiRun consists of a controller and a web-based user interface from which the tool can be operated.

OptiX, a mechanism intended for strategically allocating tests to machines in the distributed system, was designed and implemented as part of the thesis. The aim of OptiX is to minimize the overall execution time of test suites. ORX was created as an alternative allocation mechanism. It is built on OR-tools, and was made to be used for benchmarking in the evaluation process of OptiX. An experimental evaluation where the two mechanisms were tested with a collection of test sets were conducted. During the experiments, OptiX provided better results for all of the test sets in the collection.

OptiRun will help Altibox incorporate test automation as a practice in the testing process of their online web service TV Overalt.

IPTV Department
Altibox AS

Department of Electrical Engineering and Computer Science
Faculty of Science and Technology
University of Stavanger