Chapter 6
Towards Migrating Genetic Algorithms for Test Data Generation to the Cloud

Sergio Di Martino
University of Naples Federico II, Italy

Filomena Ferrucci
University of Salerno, Italy

Valerio Maggio
University of Naples Federico II, Italy

Federica Sarro
University of Salerno, Italy

ABSTRACT
Search-Based Software Testing is a well-established research area, whose goal is to apply meta-heuristic approaches, like Genetic Algorithms, to address optimization problems in the testing domain. Even if many interesting results have been achieved in this field, the heavy computational resources required by these approaches are limiting their practical application in the industrial domain. In this chapter, the authors propose the migration of Search-Based Software Testing techniques to the Cloud aiming to improve their performance and scalability. Moreover, they show how the use of the MapReduce paradigm can support the parallelization of Genetic Algorithms for test data generation and their migration in the Cloud, thus relieving software company from the management and maintenance of the overall IT infrastructure and developers from handling the communication and synchronization of parallel tasks. Some preliminary results are reported, gathered by a proof-of-concept developed on the Google’s Cloud Infrastructure.

DOI: 10.4018/978-1-4666-2536-5.ch006

Copyright © 2013, IGI Global.Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
INTRODUCTION

The software testing encompasses a range of different activities that are critical for software quality assurance. For each activity, depending on the testing objective, specific test cases need to be devised to check the system (Bertolino, 2007). Since an exhaustive enumeration of software inputs is unfeasible for any reasonable-sized system (McMinn, 2004), a careful selection of test data must be performed to obtain a high testing effectiveness. This task is often difficult, time-consuming, and error-prone. Thus, the need to decrease time and costs of software testing, while increasing its effectiveness has motivated the research for advanced techniques able to automatically generate test data. This is an active research area and a number of different approaches has been proposed and investigated in the literature (e.g., (Ali, 2010; Bertolino, 2007; De Millo, 1991; Miller, 1976)).

Among them, Search-Based techniques (Harman, 2007) are promising approaches to increase testing quality, by automatically generating relevant test data (Harman, 2001). Search-Based techniques include a variety of meta-heuristics, such as Local Search (i.e., Hill Climbing, Tabu Search, Simulated Annealing, etc.), Evolutionary Algorithms (i.e., Genetic Algorithms, Evolution Strategies, Genetic Programming, etc...), Ant Colony Optimization, or Particle Swarm Optimization. All these meta-heuristics search for a suitable solution in a typically large input space guided by a fitness function which expresses the goals and leads the exploration into potentially promising areas of the search space. Thus, using these approaches, test data generation is treated as a search or optimization problem whose goal is to find the most appropriate input data conforming to some adequacy criteria (i.e., test goals/objectives), such as maximizing the code coverage. Thus, moving from conventional manual test data definition to Search-Based test data generation essentially consists in defining a suitable fitness function to determine how good a test input is.

The generic nature of these metaheuristics let them to be fruitful for different testing goals and issues, simply by redefining the solution representation and the fitness function. Thus, in the last decades there has been an explosion of researches on the use of Search-Based techniques for software testing that have addressed a range of testing problems, giving rise to a very active research field, known as Search-Based Software Testing (SBST). These techniques have been used for structural testing (both static and dynamic) functional testing (both for generating test data and testing the conformance of the implementation to its specification), non-functional testing (e.g., testing for robustness, stress testing, security testing, gray-box testing (e.g., assertion testing and exception condition testing), state–based testing, mutation testing), regression testing interaction testing, integration testing, test case prioritization, and so on (McMinn, 2004; Ali, 2010). Despite all such efforts, so far these investigations have produced limited impact in industry (Bertolino, 2007). Maybe the main reason for that lies in the fact that few attempts have been made to improve performance of these techniques and make them more scalable. On the other hand, while several empirical studies have been carried out showing that Search-Based testing can outperform other automated testing approaches (e.g., random testing), little attention has been deserved to the scalability and effectiveness issues for real world applications (Ali, 2010). One of the few experimentation with complete open source applications reported that there are still many challenges to be addressed to make existing Search-Based test data generation tools robust and with a significant level of code coverage that can be useful for industrial use (Lakhotia, 2009).

The use of Cloud Computing can provide a significant impulse in this direction. Indeed, this model is based on the provisioning of configurable computing resources in a distributed environment allowing for an on-demand resource allocation from a virtual unlimited resources and infra-