Extending Search-Based Software Testing Techniques to Big Data Applications

Erik M. Fredericks and Reihaneh H. Hariri
Oakland University
Rochester, MI, USA
{fredericks, rhosseinzadehha}@oakland.edu

ABSTRACT

Massive datasets are quickly becoming a concern for many industries. For example, many web-based applications must be able to handle petabytes worth of transactions on a daily basis, and moreover, be able to quickly and efficiently act upon data that exists in each transaction. As a result, providing testing capabilities for such applications becomes a challenge of scale. We argue that existing approaches, such as automated test suite generation, may not necessarily scale without assistance. To this end, we discuss open issues and possible solutions specific to testing big data applications.

CCS Concepts

•Software and its engineering → Software testing and debugging; Search-based software engineering; *Software system structures;*

Keywords

big data, search-based software testing, test suite generation

1. OVERVIEW

Many techniques are currently being developed for generating datasets of massive scale (i.e., big data) for use in validating applications [1]. However, there is little published research in performing testing on applications that already interact with big data [9]. Moreover, even fewer publications explore how search-based software testing (SBST) techniques can be used to optimize testing strategies [6, 8]. As such, research needs to be performed in testing big data applications to determine both the feasibility and applicability of existing testing techniques to such applications. For example, consider a nationwide healthcare network that centralizes medical records for all patients. Such a system can deals with an enormous amount of data as well as an amalgam of heterogeneous systems and devices. This system can enable a patient to visit their primary care physician, receive a prescription for treatment with a specialist in another state, and then enable that specialist to instantly retrieve the entirety of the patient's medical history. As such, specialized applications will require development to handle the dataset, including optimizations for querying and retrieving specific data. However, such applications may not be effectively tested by existing strategies, given the wide range of values that may manifest. As such, this position paper specifically argues for an examination on how big data can impact existing testing strategies, focusing on *automated test suite generation.*

Traditionally, software testing has been considered an ideal field for application of search-based heuristics, such as genetic algorithms [7]. Notable systems include EvoSuite [5] and Nighthawk [2] for automated generation of test suites and instantiation of unit tests, respectively. Given the optimization problems that typically comprise a software testing strategy (e.g., test suite generation, test case prioritization and selection, etc.), search-based heuristics have been shown to quickly and efficiently come to an optimal solution. However, many industries are moving towards the big data paradigm, where petabytes of data must be considered at run time. As such, a strategy such as test suite generation may be cost-prohibitive, given the enormous number of combinations of test parameters and values that can exist in such a system.

We argue that existing techniques must be augmented to support the big data paradigm. Increasing the complexity and scale of applications is constantly a threat to validity for many techniques [1], and as such, big data presents an optimal platform for extending current techniques. To combat the big data problem, many systems have started using the MapReduce paradigm [3] to efficiently sift through and retrieve results in a large dataset. It makes sense then, that testing techniques must evolve alongside traditional application logic. While each aspect of software testing will be impacted by big data, this paper highlights test suite generation for discussion.

2. ISSUES AND POSSIBLE SOLUTIONS

This section discusses issues that can affect test suite generation with respect to big data, including possible SBST-specific solutions.

Test suite generation. Test suites typically comprise a set of test cases that are to be executed as specified by a test plan in order to provide a measure of test coverage. Moreover, test suites are generally intended to validate a system under a specific *operating context*, or set of circumstances that differentiate one context from another. For instance,

SBST16, May 16-17, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4166-0/16/05...$15.00
DOI: http://dx.doi.org/10.1145/2897010.2897014

several test suites can be derived for a GUI-based application, where each test suite corresponds to a particular window active within the application. Test suite generation has been often studied within the context of SBST as an ideal candidate for optimization. For instance, EvoSuite [5] is a framework that uses evolutionary search to generate test suites, as well as to suggest possible oracles, for a specified Java program.

Impact of Big Data. Test suites are intended to provide a measure of coverage with respect to validation activities. As such, a typical application may have a manageable set of operating contexts in which it operates, or in which it can be configured to operate. An application that uses big data, however, may be faced with a completely unmanageable set of operating contexts. Consider, for example, a medical records network (MRN). Given the enormity of its scale, there are innumerable configurations in which the MRN can exist. As such, deriving test suites for each instantiation of the MRN is a non-trivial task. However, if we consider testing an application that interfaces with the MRN (e.g., a phone application that retrieves a specific patient's medical records), test suite derivation becomes more manageable. The number of datasets that must be searched upon, however, remains very large. As such, test suite generation must consider the numerous instantiations of patient records, including textual data, binary large object (BLOB) data (e.g., images, scans, etc.), and even data that has been incorrectly or improperly entered.

Applications of SBST. In the face of big data, SBST remains an optimal choice for generating test suites. Furthermore, if we consider a MapReduce-style approach [3] to software testing, SBST techniques can also be applied to both the Map and Reduce phases (or ReReduce, as necessary). Here, each coverage criterion could specify and be mapped to a particular operating context. Moreover, the reducing phase could also benefit from a searching heuristic in that a minimal set of test suites would be desirable to manage the expected number of instantiated operating contexts. Recently, SBST and Hadoop have been combined to automatically generate test suites using a parallelized genetic algorithm [4]. Furthermore, automated test generation has also been recently explored with respect to relational database schemas [8].

For instance, it may be possible to determine a measure of code coverage, even when faced with such an enormous set of possible configurations, by searching for a *representative set* of test suites that reliably cover all possible conditions. For example, an MRN-affiliated application may have test suites that focus on specific aspects of the dataset. Specifically, test suites could be optimized for textual data, BLOB data, and other types of possible edge cases. Moreover, these disparate sets of test suites could then be intersected to generate additional test suites that validate combinations of parameters that can impact a system (lending further applicability to combinatorial testing). As such, an evolutionary operation heuristic that considers multiple criteria, such as a weighted fitness function or multi-objective optimization algorithm, could be applied to examine the solution space of possible test suites, where optimization factors could be represented by coverage and a minimal, representative set of test suites. Figure 1 demonstrates an example of test suite coverage using operating contexts. In this figure, there exist n possible operating contexts represented by the MRN, where an operating context could be considered a particular use case for an application that uses the overall dataset. As such, an evolutionary algorithm could search for a minimally-representative test suite that covers *each operating context* within the possible space of solutions. Moreover, reduction of test suites may be possible via joining test suites together. For example, Test Suite 2 provides a subset of its test cases to help cover Operating Context n.

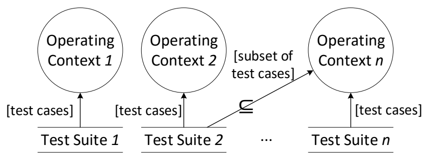

Figure 1: Sample test suite coverage.

Position. The authors posit that SBST techniques can enhance testing techniques for big data applications, specifically with respect to automated test suite generation. Ideally, SBST can effectively reduce the enormous search space presented by big data.

3. ACKNOWLEDGMENTS

This work has been supported by Oakland University. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Oakland University or other research sponsors.

4. REFERENCES

[1] A. Alexandrov, C. Brücke, and V. Markl. Issues in big data testing and benchmarking. In *Proceedings of the Sixth International Workshop on Testing Database Systems*, DBTest '13, pages 1–5, 2013.

[2] J. H. Andrews, T. Menzies, and F. C. Li. Genetic algorithms for randomized unit testing. *IEEE Trans. on Software Engineering*, 37(1):80–94, January 2011.

[3] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[4] L. Di Geronimo, F. Ferrucci, A. Murolo, and F. Sarro. A parallel genetic algorithm based on hadoop mapreduce for the automatic generation of junit test suites. In *Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, ICST '12, pages 785–793, 2012.

[5] G. Fraser and A. Arcuri. Evosuite: automatic test suite generation for object-oriented software. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, pages 416–419, Szeged, Hungary, 2011. ACM.

[6] H. Hemmati, A. Arcuri, and L. Briand. Achieving scalable model-based testing through test case diversity. *ACM Transactions on Software Engineering Methodologies*, 22(1):6:1–6:42, 2013.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, Cambridge, MA, USA, 1992.

[8] P. McMinn, C. J. Wright, and G. M. Kapfhammer. The effectiveness of test coverage criteria for relational database schema integrity constraints. *ACM Transactions on Software Engineering and Methodology*, 25(1):8:1–8:49, 2015.

[9] S. Nachiyappan and S. Justus. Getting ready for bigdata testing: A practitioner's perception. In *Fourth International Conference on Computing, Communications and Networking Technologies*, ICCCNT 2013, pages 1–5, 2013.