Extending Search based Software Testing techniques to Big Data Applications

A M Chandrashekar

Assistant Professor,
Dept of Computer Science and Engineering
Shri Jayachamarajendra College of Engineering (SJCE),
JSS S & T University
Mysuru, Karnataka

Bhavana S

2nd Sem, M.Tech,
Dept of Computer Science and Engineering
Shri Jayachamarajendra College of Engineering (SJCE),
JSS S & T University
Mysuru, Karnataka

Abstract: **Large amount of datasets are at faster rate becoming more important issues in the field of organizations and industrial areas. For example applications like web-based must handle Exabyte of worth data transactions per day and they need to act quickly and efficiently on these data transactions. Hence to provide test-cases for these applications becomes a challenge. In existing system the given automated test-cases may work suitably but not efficiently without the help of guidelines and support. Thus we provide an open subject and possible solutions to test the applications of big data.**

Keywords: **MapReduce, SBST (Search-based software testing), Test suite generation, Automated software testing tools**.

I. INTRODUCTION

Many techniques are being developed to handle and generate big data that are used in many applications [1]. However there are few number of research published on performing testing on applications that efficiently interact with big data and also few how SBST (search based software testing) methods can be used to optimize testing strategies [10]. Thus we need to perform research to test big data applications to determine applicability and feasibility of existing testing techniques.

For example, consider a nationwide healthcare network that centralizes medical records for all patients. Such a system can deal with an enormous amount of data as well as an combination 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 entire patient's medical history [14].

These special applications needs extra care to handle the dataset including storing, quering and retrieving of the required data. However, these kinds applications are not effectively tested by existing methods, for the wide range of values that are evident. Thus, this paper argues to test on how big data can impact existing testing methods, focusing on automated test case generations. Software testing provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation [13]. Test techniques include the process of executing a program or application with the intent of finding software bugs, and verifying that the software product is fit for use.

Traditionally, software testing has been considered an ideal field for application of search-based heuristics, such as genetic algorithms [9]. The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on nature. The algorithm repeatedly modifies a population of individual solutions. They provide instantiation for unit tests, respectively [12]. For the optimization problems given that typically contains a software testing strategy (e.g., test suite generation, test case prioritization and selection, etc.), search-based heuristics are used to get an optimal solution quickly and efficiently. However, many industries are moving towards the big data paradigm, where petabytes of data must be considered at run time [20]. 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.

II. APPLICATION TESTING

Application for testing tools : Accord*ingly* there are at least 50 testing tools available in market today [15]. These include both paid and open source tools. Moreover, some tools are purpose specific e.g. UI testing, Functional Testing, DB Testing, Load Testing, Performance, Security Testing and Link validation testing etc. However, some tools are strong and provide the facility of testing several major aspects of an application. The general concept of 'Application Testing' is its functional testing. So, our focus will be on functional testing tools[6].

Once the AUT(Automated software testing tools) is ready for testing, the practical phase of testing cycle starts in which testers actually execute the test cases on AUT [21]. Keep in mind that here the testing cycle is regardless of Testing Levels (Unit, Module, Integration, System and User Acceptance) and Testing Environments (Dev, QA, Client's Replica, Live).

Application Testing Methodologies: there are only 3 universally accepted methodologies.

Black box: In black-box testing, the AUT is validated against its requirements considering the inputs and expected outputs, regardless of how the inputs are transformed into outputs. Testers are least concerned with internal structure or code that implements the business logic of the application[16]. There are four primary techniques to design test cases for black box testing.

- BVA (Boundary value Analysis)
- EP (Equivalence Partitioning)
- Decision Tables
- State Transition Tables (and diagrams)

White Box: Primary focus of this methodology is to validate, how the business logic of application is implemented by code. Internal structure of the application is tested and the techniques available to do so are;

- Code Coverage
- Path Coverage

Grey Box: Practically speaking, this is a mixture of black box and white box. In this methodology, mainly the tester tests the application as in black box. But, for some business critical or vulnerable modules of application; testing is done as white box.

We argue that existing techniques must be made of greater value to support the big data paradigm [18]. Increasing the complexity and scale of applications is constantly a threat to validity for many techniques and as such, big data presents an optimal platform for extending current techniques[3]. To combat the big data problem, many systems have started using the MapReduce paradigm to efficiently sift through and retrieve results in a large dataset. It makes sense then, that testing techniques must evolve alongside traditional application logic [22]. While each aspect of software testing will be impacted by big data, this paper highlights test suite generation for discussion.

MapReduce: It is a parallel programming framework. It's neither a database nor a direct competitor to databases [19].It is a complementary to existing technologies. There are a lot of tasks that can be done in a MapReduce environment that can also be done in a relational database. MapReduce consists of two primary processes that a programmer builds the *"map"* step and the *"reduce*" step. These steps get passed to the MapReduce framework, which then runs the programs in parallel on a set of worker nodes as shown in Fig 1[11].

Each MapReduce worker runs the same code against its portion of the data. However, the workers do not interact or even have knowledge of each other [25]. For Example, If there is a steady stream of web logs coming in, it might be handed out in chunks to the various worker nodes. A simple method would be a round robin procedure where data is passed to nodes sequentially over and over. In some cases, Some sort of hashing is also common [17]. In this case, records are passed to workers based on a formula so that similar records get sent to the same worker. For example, hashing on customer ID will send all records for a given customer to the same worker.

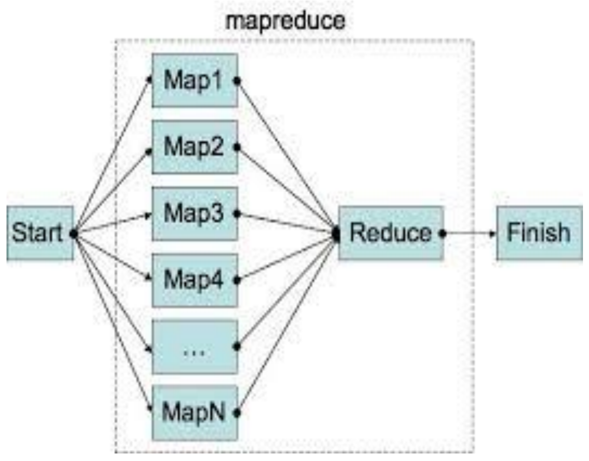

Fig 1. MapReduce

Organizations are finding that it's vital to quickly analyze the huge amounts of data they are generating to make better decisions. MapReduce is a tool that's helping those organizations handle the unstructured and semi-structured sources that are not easy to analyze with traditional tools. Most enterprises deal with multiple types of data in addition to relational data from a database [24]. These include text, machine-generated data like web logs or sensor data, images, and so forth. Organizations need to process all that data quickly and efficiently to derive meaningful insights.

With MapReduce, computational processing can occur on data stored in a file system without loading it into a database. Loading big chunks of text into a "blob"(binary large object) field in a database is possible, but it really isn't the best use of the database or the best way to handle such data[11]. A database can even provide input data to a MapReduce process, just as a MapReduce process can provide input to a database.

MapReduce is not a database, so it has no built-in security, no indexing, no query or process optimizer, no historical perspective in terms of other jobs that have been run, and no knowledge of other data that exists [26].

III. PROBLEM AND SOLUTIONS

It deals with the problems that can affect test suite generation related to big data, and also includes possible and specific SBST (Search-based software testing) solutions. The solutions proposed are:

Test suite generation: It is a collection of set of test cases that has to be executed as specified by a test plan. This provides a measurement of test coverage [2]. Moreover, they are generally made to validate a system under specific operating context, or set of circumstances that differentiate one context from another. For that time being , for any GUI-based application several test suites can be derived, where each test suite corresponds to a particular window active within the application. Test suite generation has been often studied as an ideal candidate for optimization within the context of SBST. For instance, EvoSuite[5] is a framework that uses evolutionary search to generate test suites, as well as to suggest possible solutions, for a specified Java programs.

Managing of Big data: According to the literature survey done from the beginning of recorded time (1990) until 2003,5 billion gigabytes of data was created. In 2011, the same amount was created every two days. In 2013, the same amount of data was created every 10 minutes. In 2015, same or more data was generated every 10 minutes. Advances in communications, computation, and storage have created huge collections of data, having information of value to business, science, government and society.

Explosion of new and powerful data sources like Facebook, Twitter, LinkedIn, Youtube etc., contributes immensely to Bigdata & research. Advance Analytics will be of great impact. To stay competitive, it is imperative that organizations aggressively pursue capturing and analyzing these new data sources to gain the insights that they offer [23]. Ignoring the big data will put an organization at risk and cause it to fall behind the competition. Analytic professionals have a lot of work to do indeed. It won’t be easy to incorporate big data alongside all the other data that has been used for analysis for years. Risks that are involved in big data is that costs escalate too fast as too much big data is captured before an organization knows what to do with it. The biggest challenge with big data may not be the analytics we do with, but the extract, transform, and load (ETL) processes to be build to get it ready for analysis.

Impact of Big Data: Test suites are used to provide a measurement of coverage regarding the validation activities. Hence, 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 may face problems and complexity with completely unmanageable set of operating context that uses big data. 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 MRN is interfaced with the application that has to be tested (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.

SBST (Search-based software testing) is a performance testing generally executed to determine how a system or sub-system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

*Applications of SBST***:** When dealing with the big data, SBST is always an optimal solution or a choice for generating test suites. Furthermore, if we consider a MapReduce-style approach to software testing, SBST techniques can also be applied to both the Map and Reduce phases (or ReReduce, as necessary).MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.A MapReduce program is composed of a Map() procedure performs filtering and sorting and a Reduce() method performs a summary operation [8].

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 [27]. Recently, SBST and Hadoop have been combined to automatically generate test suites using a parallelized genetic algorithm. Hadoop is an open-source software framework used for distributed storage and processing of big data sets using the MapRedce programming model. Furthermore, automated test generation has also been recently explored with respect to relational database schemas.

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.

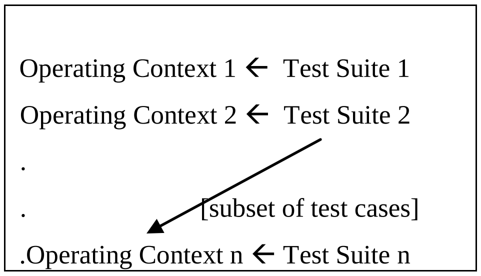
Operating Context 1 ← Test Suite 1
Operating Context 2 ← Test Suite 2
.
. [subset of test cases]
.Operating Context n ← Test Suite n

Fig 2. Sample test suite coverage

Figure 2 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.

IV. CONCLUSION

Big data is a term for data sets that are so large or complex that traditional data processing application software is inadequate to deal with them. It exceeds the reach of commonly used hardware environments and software tools to capture, manage, and process it within a tolerable elapsed time for its user population Challenges include data capture, storage, analysis,duration, transfer, visualization querying, updating and information privacy. Thus here we conclude that SBST is the optimal choice for generating test suites. They combine with MapReduce and Hadoop to give the best results and Hadoop is the best known implementation of the MapReduce framework. The given evolutionary algorithm search the minimal representing test cases that covers all operating context within the solution space. Application Testing is a vast subject and the primary activity of any software tester.Application Testing involves strategies, phenomena, approaches, tools, technologies and guidelines.

REFERENCES

[1]. A. Alexandrov, C. Br¨ucke, 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.

[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.

[7]. Web source ‘Test Case Design Techniques’.Posted In Basics of Software testing, Testing best practices, Testing Concepts, Types of testing April 17, 2017".

[8]. P. McMinn, C. J. Wright, and G. M. Kapfhammer. The e_ectiveness 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]. J. H. Holland. Adaptation in Natural and Arti_cial Systems. MIT Press, Cambridge, MA, USA, 1992.

[10]. S. Nachiyappan and S. Justus. Getting ready for bigdata testing: A practitioner's perception. In Fourth International.

[11]. Bill Franks, Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with advanced analytics, John Wiley & sons, 2012.

[12]. A. M. Chandrashekhar and K. Raghuveer, “Confederation of FCM Clustering, ANN and SVM Techniques of Data mining to Implement Hybrid NIDS Using Corrected KDD Cup Dataset”, Communication and Signal Processing (ICCSP) IEEE International Conference,2014, Page 672-676.

[13]. AM. Chandrashekhar and K. Raghuveer , “Improvising Intrusion detection precision of ANN based NIDS by incorporating various data Normalization Technique – A Performance Appraisal”, IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014.

[14]. A. M Chandrashekhar A M and K. Raghuveer, “Diverse and Conglomerate modi-operandi for Anomaly Intrusion Detection Systems”, International Journal of Computer Application (IJCA) Special Issue on “Network Security and Cryptography (NSC)”, 2011.

[15]. A. M. Chandrashekhar and K. Raghuveer, “Amalgamation of K-means clustering algorithem with standard MLP and SVM based neural networks to implement network intrusion detection system”, Advanced Computing, Networking, and Informatics –Volume 2(June 2014), Volume 28 of the series Smart Inovation, Systems and Technologies pp 273-283.

[16]. A. M. Chandrashekhar and K. Raghuveer, “Fusion of Multiple Data Mining Techniques for Effective Network Intrusion Detection – A Contemporary Approach”, Proceedings of Fifth International Conference on Security of Information and Networks (SIN 2012), 2012, Page 178-182.

[17]. A. M. Chandrashekhar, Jagadish Revapgol, Vinayaka Pattanashetti, “Big Data Security Issues in Networking”, International Journal of Scientific Research in Science, Engineering and Technology (*IJSRSET),* Volume 2, Issue 1, JAN-2016.

[18]. Puneeth L Sankadal, A. M Chandrashekhar, Prashanth Chillabatte, “Network Security situation awareness system” International Journal of Advanced Research in Information and Communication Engineering(IJARICE), Volume 3, Issue 5, May 2015.

[19]. P. Koushik, A.M.Chandrashekhar, Jagadeesh Takkalakaki, “Information security threats, awareness and cognizance” International Journal for Technical research in Engineering(IJTRE), Volume 2, Issue 9, May 2015.

[20]. A.M.Chandrashekhar, Yadunandan Huded, H S Sachin Kumar, “Advances in Information security risk practices” International Journal of Advanced Research in data mining and Cloud computing (IJARDC), Volume 3, Issue 5, May 2015.

[21]. A. M. Chandrashekhar,Muktha G, Anjana D, “Cyberstalking and Cyberbullying: Effects and prevention measures”, Imperial Journal of Interdisciplinary Research (IJIR), Volume 2, Issue 2, JAN-2016.

[22]. A.M.Chandrashekhar, Syed Tahseen Ahmed, Rahul N, “Analysis of Security Threats to Database Storage Systems” International Journal of Advanced Research in data mining and Cloud computing (IJARDC), Volume 3, Issue 5, May 2015.

[23]. A.M.Chandrashekhar, K.K. Sowmyashree, RS Sheethal, “Pyramidal aggregation on Communication security” International Journal of Advanced Research in Computer Science and Applications (IJARCSA), Volume 3, Issue 5, May 2015.

[24]. A.M.Chandrashekhar, Rahil kumar Gupta, Shivaraj H. P, “Role of information security awareness in success of an organization” International Journal of Research(IJR), Volume 2, Issue 6, May 2015.

[25]. A.M.Chandrashekhar, Huda Mirza Saifuddin, Spoorthi B.S, “Exploration of the ingredients of original security” International

Journal of Advanced Research in Computer Science and Applications(IJARCSA), Volume 3, Issue 5, May 2015.

[26]. A.M.Chandrasekhar, Ngaveni Bhavi, Pushpanjali M K, “Hierarchical Group Communication Security”, International journal of Advanced research in Computer science and Applications (IJARCSA), Volume 4, Issue 1,Feb-2016.

[27]. A. M Chandrashekhar A M and K. Raghuveer, “Hard Clustering Vs. Soft Clustering: A Close Contest for Attaining Supremacy in Hybrid NIDS Development”, Proceedings of International Conference on Communication and Computing (ICCC - 2014), Elsevier science and Technology Publications.