Testing for Configuration Errors in Distributed Systems

PI: Marcelo d'Amorim (damorim@cin.ufpe.br) Universidade Federal de Pernambuco
Phone: +55 (81) 98800-2010 [mobile] Av. Jornalista Anibal Fernandes, S/N
+55 (81) 2126-8430, ext. 4379 [work] Cidade Universitária, PE, Brazil, 50.740-560
Contact: Wei Jin (superbobo@google.com) – **Sponsor**: Sai Zhang (saizhang@google.com)

Abstract

This proposal focuses on the important problem of testing for configuration errors in distributed systems. The goal is to increase system's reliability by preventing these errors to escape to production.

1 Problem

Configurable systems are those that can be adapted from a set of input options, reflected in code in the form of variations. Configurable systems are prevalent and, unfortunately, configuration-related errors are difficult to detect [13,20–22] and diagnose [29,30]. Testing and debugging becomes even worse when the system being configured is a distributed system [25]. Detecting these errors with testing, before they escape, is therefore important as it is hard to troubleshoot error manifestations in complex production environments [10]. Configuration-related errors in distributed systems are unfortunately *not rare*. Some of these errors have been widely publicized in the media given the volume of users or data they affected [1,2,11].

The goal of this project is to improve reliability in distributed systems by finding configuration errors during testing.

Related Work. Literature on distributed debugging is vast [7]. Record and replay solutions, such as Friday [9] and D3S [14], capture nondeterministic events during execution to replay them a posteriori. Tracing solutions [18], such as Pivot Tracing [15], instrument the application to forward tracing metadata to a distributed database for offline analysis. In principle, these solutions could be adapted to track configuration data but they are either heavyweight or intrusive (i.e., they require modification in software). These issues limit the applicability of Record and Replay and Tracing in practice. Log Analysis does not suffer from these limitations as it mines anomalies from the logs that the system already produces [26]. However, Log Analysis needs to cope with the potential for missing relevant data and also to handle massive amounts of data. Considering the configuration dimension, testing and debugging for distributed systems has not been studied in depth [25], in contrast to what has been observed in non-distributed systems [5,24,28–30]. Most testing in distributed systems is done manually and after the fact (i.e. after failures are observed) to augment regression test suites [6]. Model checking distributed systems (,which is essentially systematic testing) has been proposed [12,27] but these existing approaches do not take configurations into account.

2 Approach

We propose an approach that uses automated test generation to find configuration errors in distributed systems. Our approach exercises the system observing anomalies in behavior due to configuration changes. Testing configurable software is a topic on which the group of the PI has focused in the recent past [13,20–22]. Our approach is similar in spirit to MoDist [27], a black-box model checker that runs the system modifying certain aspects during execution such as processor speed and message sequencing, but it takes configuration options into account.

Input/Output. The input of our technique is a configurable distributed system and the output is one of more failing tests and error reports.

Requirements. The technique should be simple to use and highly automated. It should produce high-level reports that enable developers to promptly understand and fix the problems identified. It should run efficiently.

Tasks. 1) Define containerization strategy. 2) Define test strategy. 3) Minimize test sequences.

Considering task 1, it is important to start a distributed system quickly and to run the system in an isolated environment. We plan to use Docker images [3] to jump start the individual nodes of a distributed system.

Considering task 2, our approach is to use lighter weight test methods first and then assess effectiveness and cost of heavier weight testing strategies. More specifically, initially, we plan to use existing test cases [16] and select configurations to run those tests using black-box techniques, such as Combinatorial Interaction Testing (CIT) [17]. It is worth noting that centralized configuration servers such as ZooKeeper's [4] and Spring Cloud Config [23] are becoming popular to manage configurations in distributed systems and should facilitate this task. We also plan to explore gray-box techniques for selecting configurations — we will investigate the effectiveness of mining network interactions to decide what configurations to test next. Existing log analysis tools can help on that [15]. For example, our analysis may discover that a datum in a message triggers a different decision in a node, as observed from execution logs, and that that datum is related to a configuration option. We conjecture that selecting configurations that triggers different interactions increases the ability of testing to find faults. Finally, automated test generation can create arbitrarily long test sequences which hinders reproducibility and fault diagnosis. Task 3 is concerned with the minimization of fault-revealing tests to facilitate debugging.

We will build tools that implement and support our approach. The output of our approach is a failing test, which could reveal a bug or a misconfiguration. In case the developer finds out that the failure corresponds to a bug, the code needs to be fixed. In case of misconfigurations, our approach could be used to prevent errors provided that the developer adds checks to (in)validate certain configurations from being triggered. Alternatively, the approach could be used to diagnose errors as logs reporting error manifestations could be saved and then used for comparison with production runs.

3 Case Studies

We plan to evaluate our approach in some of the following scenarios:

- **Software-Defined Networking (SDN).** SDN is a network architecture that enables network administrators to programmatically control the network behavior through well-defined interfaces such as OpenFlow [8]. Humans can make mistakes in an SDN as in any other configurable software. For example, it is possible that "one of the switches is sending packets to the wrong destination network" [19].
- **Internet of Things (IoT).** IoT deployments are highly configurable and typically depend on services distributed on the network. This is the case for both cloud and non-cloud based IoT setups. Considering the scenario of a smart home, for example, the number of sensors and the configuration options on each sensor can vary (e.g., the max amount of smoke acceptable[1] before signaling an alarm or sending a message to the apartment owner). To note that the PI is involved in a Brazil-US cooperation project[2] related to testing IoT infrastructures and this could create synergies that will help the project to succeed.
- **Microservices.** Microservices is an increasingly-popular approach to organize applications (typically, web apps). It promotes the breakdown of code in small service units. Consider, for example, the scenario where a monolithic application is refactored to use microservices. Refactorings can be buggy and the new channels created for communication (among microservices) can be vulnerable to attacks.

We are also eager to evaluate the techniques we develop in scenarios within Google

4 Data Policy

The results produced with this research will be made available to the public and to the research community. All tools and data sets developed will be available online, and the software will be released under an open source license.

5 Budget

The project will support one PhD student for one year. The total budget for this period is 25,800 USD.

- Student salaries: 22,800 (=1,900*12)
- Conference travel: 3,000

[1] https://diyhacking.com/iot-smoke-alarm-arduino-esp8266-gas-sensor/
[2] Funded by RNP, in Brazil, and NSF, in the US: `https://www.nsf.gov/pubs/2017/nsf17024/nsf17024.jsp`

References

[1] Configuration error brings down the azure cloud platform. `http://www.evolven.com/blog/configuration-error-brings-down-the-azure-cloud-platform.html`.

[2] Dns misconfiguration. `http://www.circleid.com/posts/misconfiguration_brings_down_entire_se_domain_in_sweden`.

[3] Docker. `https://www.docker.com/`.

[4] Apache. Apache zookeeper. `https://zookeeper.apache.org/`.

[5] Farnaz Behrang, Myra B. Cohen, and Alessandro Orso. Users beware: Preference inconsistencies ahead. In *ESEC/FSE*, pages 295–306, 2015.

[6] Ivan Beschastnikh, Yuriy Brun, Michael D. Ernst, and Arvind Krishnamurthy. Inferring models of concurrent systems from logs of their behavior with csight. In *ICSE*, pages 468–479, 2014.

[7] Ivan Beschastnikh, Patty Wang, Yuriy Brun, and Michael Ernst. Debugging distributed systems: Challenges and options for validation and debugging. *ACM Queue*, 14(2):91–110, March/April 2016.

[8] Open Networking Foundation. Onf openflow website. `https://www.opennetworking.org/`.

[9] Dennis Geels, Gautam Altekar, Petros Maniatis, Timothy Roscoe, and Ion Stoica. Friday: Global comprehension for distributed replay. In *NSDI*, Cambridge, MA, 2007. USENIX Association.

[10] Wei Jin and Alessandro Orso. Bugredux: Reproducing field failures for in-house debugging. In *ICSE*, pages 474–484, 2012.

[11] Robert Johnson. More details on todays outage. `https://www.facebook.com/notes/facebook-engineering/more-details-on-todaysoutage/431441338919`.

[12] Charles Killian, James W. Anderson, Ranjit Jhala, and Amin Vahdat. Life, death, and the critical transition: Finding liveness bugs in systems code. In *NSDI*, NSDI'07, pages 18–18, Berkeley, CA, USA, 2007. USENIX Association.

[13] Chang Hwan Peter Kim, Darko Marinov, Sarfraz Khurshid, Don S. Batory, Sabrina Souto, Paulo Barros, and Marcelo d'Amorim. Splat: lightweight dynamic analysis for reducing combinatorics in testing configurable systems. In *FSE*, pages 257–267, 2013.

[14] Xuezheng Liu, Zhenyu Guo, Xi Wang, Feibo Chen, Xiaochen Lian, Jian Tang, Ming Wu, M. Frans Kaashoek, and Zheng Zhang. D3s: Debugging deployed distributed systems. NSDI, February 2008.

[15] Jonathan Mace, Ryan Roelke, and Rodrigo Fonseca. Pivot tracing: Dynamic causal monitoring for distributed systems. In *USENIX*, Denver, CO, 2016. USENIX Association.

[16] Philip Maddox. Testing a distributed system. *Queue*, 13(7):10:10–10:15, July 2015.

[17] Changhai Nie and Hareton Leung. A survey of combinatorial testing. *ACM Comput. Surv.*, 43(2):11:1–11:29, February 2011.

[18] R.R. Sambasivan, R. Fonseca, I. Shafer, and G Ganger. So, you want to trace your distributed system? key design insights from years of practical experience. Technical report, Carnegie Mellon University, 2014.

[19] C. Scott, A. Panda, V. Brajkovic, G. Necula, A. Krishnamurthy, and S. Shenker. Minimizing faulty executions of distributed systems. NSDI '16, 2016.

[20] Sabrina Souto and Marcelo d'Amorim. Time-space efficient regression testing for configurable software. *Journal of Systems and Software*. To Appear.

[21] Sabrina Souto, Marcelo d'Amorim, and Rohit Gheyi. Balancing soundness and efficiency for practical testing of configurable systems. In *ICSE*, pages 632–642, 2017.

[22] Sabrina Souto, Divya Gopinath, Marcelo d'Amorim, Darko Marinov, Sarfraz Khurshid, and Don S. Batory. Faster bug detection for software product lines with incomplete feature models. In *Proceedings of the 19th International Conference on Software Product Line, SPLC 2015, Nashville, TN, USA, July 20-24, 2015*, pages 151–160, 2015.

[23] Spring. Spring cloud config. `http://cloud.spring.io/spring-cloud-config/`.

[24] Aaron Weiss, Arjun Guha, and Yuriy Brun. Tortoise: Interactive system configuration repair. In *Automated Software Engineering*, Urbana, IL, 2017.

[25] Matt Welsh. Volatile and decentralized. `http://matt-welsh.blogspot.com/2013/05/what-i-wish-systems-researchers-would.html`.

[26] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael Jordan. Experience Mining Google's Production Console Logs. In *Proceedings of the 2010 Workshop on Managing Systems via Log Analysis and Machine Learning Techniques*, SLAML'10, pages 5–5, Berkeley, CA, USA, 2010. USENIX Association.

[27] Junfeng Yang, Tisheng Chen, Ming Wu, Zhilei Xu, Xuezheng Liu, Haoxiang Lin, Mao Yang, Fan Long, Lintao Zhang, and Lidong Zhou. Modist: Transparent model checking of unmodified distributed systems. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, pages 213–228, Berkeley, CA, USA, 2009. USENIX Association.

[28] Zuoning Yin, Xiao Ma, Jing Zheng, Yuanyuan Zhou, Lakshmi N. Bairavasundaram, and Shankar Pasupathy. An empirical study on configuration errors in commercial and open source systems. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 159–172, New York, NY, USA, 2011. ACM.

[29] Sai Zhang and Michael D. Ernst. Automated Diagnosis of Software Configuration Errors. In *ICSE*, pages 312–321, 2013.

[30] Sai Zhang and Michael D. Ernst. Which Configuration Option Should I Change? In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 152–163, 2014.