Proactive Methods in Dealing with Configuration Problems

Yuanyuan Zhou

University of California, San Diego Center for Networked Systems
Department of Computer Science and Engineering
University of California, San Diego
9500 Gilman Drive, MC 0440
La Jolla, CA 92093-0440

1 Summary

Misconfiguration is an important and challenging issue. In the first stages of this project, we conduct two comprehensive characteristics studies on misconfigurations in both commercial systems and open-source software. The studies confirm the dominance and severity of configuration errors, and provide insights on the types of misconfigurations to focus on as well as the types of solutions that are needed. In the second stages of this project, we explore using proactive methods to deal with configuration errors. We design and implement a tool, SPEX, to expose misconfiguration vulnerabilities and test software resilience to misconfigurations. Additionally, we build a tool, EnCore, which leverages the correlations of configuration and environment, and learns configuration rules to detect misconfigurations.

In this report, we summarize a latest solution we've been exploring towards proactively handle misconfiguration: a configuration validation framework to allow cloud system developers/operators to easily and systematically validate configurations.

2 Personnel

The principal investigator of this project is Yuanyuan Zhou. Students involved in the project include Tianyin Xu, Jiaqi Zhang, Xinxin Jin and Peng Huang.

3 Introduction

Cloud-scale systems are controlled by an ocean of configuration artifacts that control software and hardware stack across different components. These configuration artifacts evolve frequently as software and/or environment changes. Ensuring that these configurations are set correctly is challenging and a wrong configuration setting can bring down the entire service.

Such misconfiguration-induced incidents are common even in major public cloud services such as AWS, Microsoft Azure and Google Cloud, causing significant SLA loss [1, 2, 3].

A number of work in the past few years has been proposed to detect, troubleshoot and fix misconfiguration. While useful, these solutions often attack misconfiguration reactively or incur overhead that is not affordable to production systems (*e.g.,* instrumentation, record and replay). Preventing configuration errors with low overhead is a desirable gaol for production-level systems.

Towards this end, one way is to to validate configurations continuously as the system and configurations are updated. Validating configuration can not only catch the issue in early stage but also provide useful information in how to fix the issue. Additionally, when a configuration problem is resolved, the problem-solving experience can be expressed in new validation logic to be used later.

We observe that practitioners in production cloud environment carry out configuration validations in some forms, *e.g.*, configuration reviews, validation scripts, and checking logic in system code. However, these practices are often inefficient and ad-hoc. There are repeated manual efforts invested to look for similar errors. The validation code is scattered in different code region with lots of redundancy, making it hard to reuse and maintain these code. These ad-hoc practices in turn make practitioners reluctant to put more validation in place and cause repeated misconfiguration-induced incidents in production.

4 Systematic Configuration Validation

Our goal is to build a framework to allow practitioners in cloud-scale systems to easily, systematically and efficiently validate configuration. In interacting with the practitioners, we find that unlike end users, these practitioners are trained and dedicated to operate the systems. Therefore, they have the expertise and experiences to know what should or should not be set for certain configurations. By providing the right tool to them, we believe it will help make configuration validation an ordinary part of the system development and operation.

The challenges lying in building such a configuration validation framework is how to allow operators to express desired properties easily, how to minimize manual efforts so operators have incentives to use the validation, how to run validation efficiently on a large volume of configuration data.

At the core of our proposed validation framework is a simple, declarative validation language to describe various validation requirements that are independent of the underlying configuration representations. The language allows both static checking (*e.g.*, type, range) and dynamic checking (*e.g.*, if an endpoint is reachable from a deployed node). The framework provides an interactive console for operators to perform some quick checking as well as services that run specified validation automatically and figure out when to perform revalidation as the environment and configuration changes.

Given the volume of configurations in cloud systems, writing all validation requirements from scratch can be time-consuming. Therefore, our framework also contains an inference component to automatically infer and generate requirements as a basis for developers/operators to work on.

5 Preliminary Result

We preliminarily evaluate our configuration validation framework inside a major cloud service provider as as well as two open-source cloud systems, OpenStack and CloudStack. Our framework demonstrates its clear advantages over prior validation practices. Specifically, we rewrite the existing configuration validation code used in the commericial cloud systems in our new language with more than a 10x reduction in terms of lines of code. The new concise validation code is more declarative and easier to read. A similar level of reduction and increased readability is seen rewriting the validation for open-source cloud systems. Our inference component in the framework can infer thousands of requirements with high accuracy when evaluated in the commercial cloud system. With the inferred requirements, we validate the latest configuration snapshot in the commercial system and report 43 violations, 32 of which are true configuration errors. With requirements written by experts, reported 8 configuration errors, all of which are confirmed.

6 Publications

The project contributed to three major publications and one retracted publication:

- "An Empirical Study on Configuration Errors in Commercial and Open Source Systems", Zuoning Yin, Xiao Ma, Jing Zheng, Yuanyuan Zhou, Lakshmi N Bairavasundaram, and Shankar Pasupathy, Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (SOSP '11), October 2011.

- "Do Not Blame Users for Misconfigurations", Tianyin Xu, Jiaqi Zhang, Peng Huang, Jing Zheng, Tianwei Sheng, Ding Yuan, Yuanyuan Zhou, and Pasupathy, Shankar, Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP '13), November 2013.

- "EnCore: Exploiting System Environment and Correlation Information for Misconfiguration Detection", Jiaqi Zhang, Lakshminarayanan Renganarayana, Xiaolan Zhang, Niyu Ge, Vasanth Bala, Tianyin Xu, and Yuanyuan Zhou, Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14), March 2014.

- "Why Does a Cloud-Scale Service Fail Despite Fault-Tolerance?", Peng Huang, Xinxin Jin, William J. Bolosky, and Yuanyuan Zhou. OSDI '14 (Retracted due to data sensitivity).

Reference

[1] *Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region.* `http://aws.amazon.com/message/65648`. April 2011.

[2] *Details of the December 28th, 2012 Windows Azure Storage Disruption in US South.* `http://blogs.msdn.com/b/windowsazure/archive/2013/01/16/details-of-the-december-28th-2012-windows-azure-storage-disruption-in-us-south.aspx`. December 2012.

[3] *Today's outage for several Google services.* `http://googleblog.blogspot.com/2014/01/todays-outage-for-several-google.html`. January 2014.