Chaos Experimentation based on Established Risk Analysis Methods: Experiences from an Industrial Case Study

Extended Abstract

Dominik Kesim[1]
[1] University of Stuttgart, Germany
Dominik.Kesim@gmail.com

Modern distributed systems more and more frequently adapt the microservice architectural style to design cloud-based systems [1]. Assessing the resilience of such systems can be done by leveraging fault injections on the application- and network-level also known as resilience benchmarking. Chaos engineering is a new, yet, evolving discipline that forces a change in the perspective of how systems are developed with respect to their resilience. Whereas fault injection and testing approaches are said to be binary, i.e., an assertion is made about the system that is then verified to be true or false, chaos engineering is claimed to be exploratory [2]. The key idea is to apply empirical experimentation in order to learn how a system behaves under turbulent conditions by intentionally injecting failures.

Large distributed systems contain many components that are communicating and even perhaps interacting with each other. Such dependencies increase the number of potential failures enormously [2]. Peter Deutsch's "Seven fallacies of distributed computing" [3] are a good example of neglected failures that may occur during the operation of distributed systems in production. From a chaos engineering perspective technologies such as Kubernetes also increase the complexity by increasing the number of potential failures, e.g., Pods can fail, node memory space is drained.

In the scope of an industrial case study conducted as part of my Bachelor's thesis [4], our work provides means to identify risks and hazards by applying hazard analysis methods known from engineering safety-critical systems to the domain of chaos engineering. The case study involved five stages. The first stage focused on collecting information about the investigated system by conducting three interviews with the system developers and architect. In the second stage the information were processed in order to reconstruct the architecture of the system. The result was a detailed architecture description comprising different architectural views, i.e., sequence and component diagrams to illustrate the structure and communication of individual architectural components. The third stage comprised the identification of risks by applying three different hazard analysis methods, namely i) Fault Tree Analysis as a top-down approach to identify root causes, ii) Failure Mode and Effects Analysis as a component-based inspection of different failure modes, iii) and Computational Hazard and Operations as a means to analyze the system's communication paths. In the fourth stage a dedicated number of chaos engineering experiments were implemented in Chaostoolkit [4] based on the identified risks. As Chaostoolkit provides, among others, a Kubernetes-driver, the experiments were specifically designed to target the Kubernetes platform. In total four experiments have been derived from the findings of

the hazard analysis. In the fifth and last stage the derived experiments have been executed and analyzed by applying non-parametric statistical hypothesis tests to the observations.
In this talk, I will summarize the methods, results, and lessons learned from the case study.

References

[1] A. van Hoorn, A. Aleti, T. F. Düllmann, T. Pitakrat. “ORCAS: Efficient Resilience Benchmarking of Microservice Architectures.”2018 IEEE, International Symposium on Software Reliability Engineering Workshops (ISSREW)
[2] A. Basiri, N. Behnam, R. de Rooij, L. Hochstein, L. Kosewski, J. Reynolds, C. Rosenthal. “Chaos Engineering.” IEEE Software 33.3 (2016)
[3] Rotem-Gal-Oz, A. (2006). Fallacies of distributed computing explained. URL http://www. rgoarchitects. com/Files/fallacies. pdf, 20.
[4] Dominik Kesim. Assessing Resilience of Software Systems by the Application of Chaos Engineering — A Case Study (2019). Bachelor’s Thesis, University of Stuttgart. Submission date Sep 1, 2019. The thesis will be available under the following URL for the reviewers right after submission: https://drive.google.com/drive/folders/1o7Ii96QLH73c9gJ_UbEBbvWfhiRPQI71?usp=sharing
[5] https://chaostoolkit.org/