Application-level Chaos Engineering

Long Zhang <longz@kth.se>, KTH
Main advisor: Martin Monperrus

Abstract

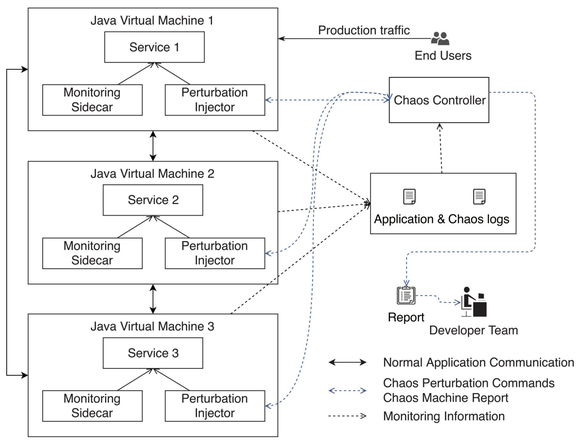
Chaos engineering is a new scientific method within software engineering that consists in specifying and evaluating resilience hypotheses by 1) injecting faults in a production system, 2) observing the impact of such faults, and 3) building new knowledge about the strengths and weaknesses of the resilience of the system. Chaos engineering can be applied at different levels such as network level and infrastructure level. In order to provide more concrete and application-specific insights for developers, our research work focuses on using application-level chaos engineering to address the following challenges: **C1-How to evaluate different aspects of resilience, C2-How to automate the chaos experiments, and C3-How to improve the efficiency of the chaos engineering experiments.**

ChaosMachine

Building confidence in system behavior through EXPERIMENTS in RUNTIME

Java Virtual Machine — Java Virtual Machine — Javaagent, Java byte-code, ASM

In order to address C1, we propose to use different perturbation models at the application level [1,2]. This is because: application-level perturbation models are closer to the application's source code, which helps developers to locate the improvement target, and such models simulate more concrete failure scenarios for this specific application.



- Input
  - Arbitrary software in Java
  - Hypotheses
- Architecture
  - Monitoring sidecars
  - Perturbation injectors
  - Chaos controller
- Output
  - Resilience report

Perturbation model: try-catch block short-circuit testing

- A corresponding exception at the beginning
- The whole try block is made invalid

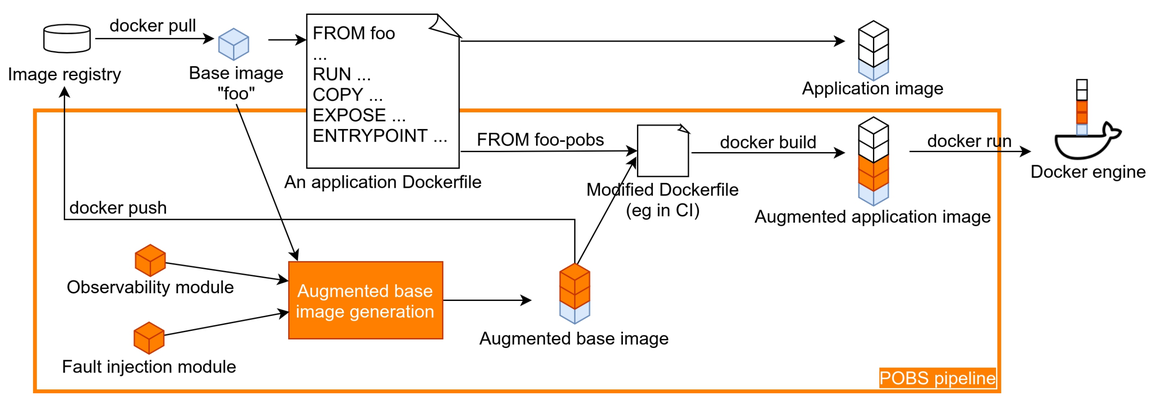
Hypotheses

- RH (Resilience hypothesis)
- OH (Observability hypothesis)
- DH (Debug hypothesis)
- SH (Silence hypothesis)

Evaluation

- 3 large-scale and well-known Java applications totaling 630k lines of code

POBS

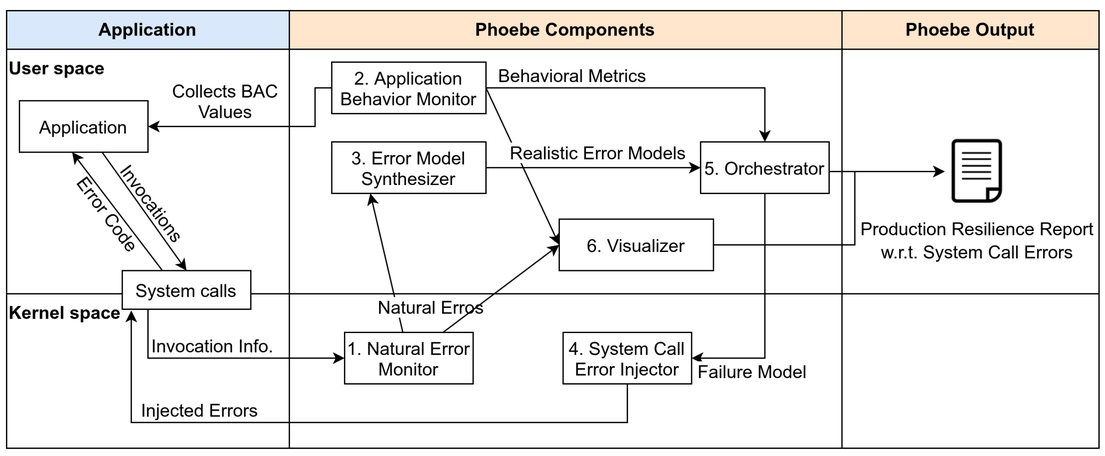


In order to address C2, we propose to design orchestration frameworks to connect monitoring, injection, and analysis. For most of the existing chaos engineering tools, there are several limitations: 1) steady state and hypotheses have to be manually defined, 2) the installation of a tool may be complicated, and 3) the setup of an experiment may be tedious.

Thus we propose a technique called POBS (imProved OBServability) to statically analyze and transform Docker configuration files of Java applications in order to inject observability capabilities [3].

For example, POBS allows developers to observe the JVM memory or CPU usage of their application **with minimal effort: a single line change in the Docker configuration**.

Phoebe



In order to address C3, we present a novel fault injection framework for system call invocation errors, called Phoebe [4]. Phoebe is unique as follows. First, Phoebe enables developers to have full observability of system call invocations. Second, **Phoebe generates error models that are realistic in the sense that they mimic errors that naturally happen in production.** Third, Phoebe is able to automatically conduct experiments to systematically assess the reliability of applications with respect to system call invocation errors in production.

References

1. Zhang et al, *A Chaos Engineering System for Live Analysis and Falsification of Exception-Handling in the JVM,* IEEE TSE, 2019.
2. Zhang et al, *TripleAgent: Monitoring, Perturbation and Failure-Obliviousness for Automated Resilience Improvement in Java Applications,* IEEE 30th International Symposium on Software Reliability Engineering (ISSRE), 2019.
3. Zhang et al, *Automatic Observability for Dockerized Java Applications,* arXiv preprint:1912.06914, 2019.
4. Zhang et al, *Maximizing Error Injection Realism for Chaos Engineering with System Calls,* IEEE TDSC, 2021.

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM