Abstract

BACKGROUND: The number of users of web services is increasing annually and, therefore, all developers aim to develop a high reliable software. However, no system can withstand any load, so site reliability engineers prepare for potential service failures by implementing various reliability practices to minimize associated risks. Feedback control system is a system that can give feedback about its current status so that engineers can react to it in advance the risks of a fall.

OBJECTIVE: However, for feedback control systems developers face a problem that each reliability practice is configured in its own way and combinations of these settings can give different results in the reliability of the system. This paper outlines a web-service for reproducible load testing, visualization, and configuration of feedback control systems

METHODS: The web-services was developed by using Kotlin and Spring Boot framework, and Kafka broker message was used for communication. Yandex tank was chosen as a high load generator for the ability to use several cores for load generation and a convenient API. Additionally, by using Angular framework I developed a UI interface for a more convenient and visual use of the service.

RESULTS: The proposed service can handled services for configuration, executed load testing scenarios and provided real-time results as graphs.

CONTRIBUTION AND APPLICABILITY: This service can be used in testing systems under various load scenarios with different configuration variants. These practices help to find the most effective combination of parameters that ensures optimal system performance.

Chapter 1

Introduction

Chapter 2

Literature Review

To further implementation of the service it is required to choose appropriate tools and approaches for development. This chapter is organized in such way:

- Section 1 reviews modern load generation tools
- Section 2 gives a brief overview of reliability patterns configuration
- Section 3 outlines the tools and technologies for implementing service

I Load Generation Tool

[1] presents a comprehensive comparison of the following modern load generation tools:

- HP LoadRunner
- Apache JMetr
- MicroFocus (Borland) Silk Performer

- Microsoft Visual Studio
- 1C KIP
- Yandex.Tank
- Pure Load
- Tsung
- Gatling
- Grinder

Yandex tank is the most suitable tool because it is open-source and configured using a config file. The author advises using Jmeter, but he did not cover the point that Yandex tank, unlike Jmeter, can use different engines to generate loads. And, for example, with the Phantom engine[2], it can generate a load of more than 100,000 requests per second. Also, Yandex Tank has a ready-made http web server [3], which will be taken as the basis for the load generation microservice

II Reliability patterns

Buyer et al. [4] examines strategies for high load withstand. They present several approaches, such as rate-limit, retry strategy, circuit-breaker and load-balancer. However, the authors did not highlight best practices for configuring these approaches.

Circuit-breaker have several tune parameters [4], such as sleep window, error percentage threshold, etc. At the same time, load balancer [10] have

multiple strategies, such as round robin, ratio, etc. Retry strategy can be configured by maximum number and interval of retries [11]. These approaches parameters produce different throughput in different combination, and it is crucial to find the most suitable for the task.

III Service implementation

First, Spring framework is the main choice of the web services implementation. [8] explains that it is more preferable due to its scalability and high-quality documentation

Second, service is divided into microservices due to machine high load during load test [9]. And in case of system throttling, configuration service must be stable and do not depend on the state of the load generation service.

Third, service based on event-driven architecture [6] because of the constant configuration changes and the need for real-time awareness. According to this architecture, it is required to use stream of event. Kafka tool is used to implement it due to high throughput and fault tolerance [7].

Kotlin + Spring Docker + K8s