Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Distributed tracing in tests

Szántó Tamás III. évf, (BSc) mérnökinformatikus szakos hallgató
Konzulens: Micskei Zoltán egyetemi docens, PhD, MIT
Horányi Gergő, Prezi
Rendszertervezés szakirány
Önálló laboratórium összefoglaló
2016/17. II. félév

With distributed systems debugging, monitoring and testing can be really challenging tasks. Distributed tracing can help with these tasks by logging the communication process with timing data between the components. Zipkin is an open-source distributed tracing system developed by Twitter. It specialised to help resolve latency problems in micro service architectures.

The main task of the Project Laboratory is to investigate how can Zipkin support the testing. One part of this is how can be used the provided tracing data and the other is how well can work Zipkin as a part of a testing architecture.

First task was creating a test system with four microservices with different programming languages and frameworks. After the system worked with Zipkin on my computer I moved it to a distributed environment (Amazon Web Services).

After I had a system the next task was to crate integration and performance tests which use tracing data provided by Zipkin api. This first attempt for tests was really specialised for my example system, but it showed Zipkin can help with testing tasks.

In order to generalise my tests and rethink the performance tests evaluation process I did a research about test frameworks. I created a test framework that can work with any distributed system that uses http requests for communication. The idea behind the new performance test framework was to select a group of traces and then define constrains for these traces. So these tests can help to find fault traces. To run the tests automatically I added Travis CI to the project. The test scripts run on Travis CI but the tested system runs on AWS.

As an other approach to use tracing data in tests I created a basic structural analysis framework. Simple structural requirements can be defined for specific traces and then a Python script evaluates these based on the tracing data.

As conclusion Zipkin and overall the distributed tracing can be really helpful to analyse, monitor and test the all of the components of a system, while it performs a specific task. The main feature of the performance tests with Zipkin is the opportunity to check each components performance as part of the architecture.

For the future it could be interesting to try out similar tests with Zipkin on larger, more complex systems. And also the structural analysis seems to be an interesting topic about distributed tracing and tests.