UMEÅ UNIVERSITY
Department of Computing Science

January 23, 2020

Evaluation of distributed tracing frameworks for microservice observability.

Project Plan

Version 1.0

Name	Kristoffer Gilliusson
Supervisor	Johan Tordsson
Examiner	Henrik Björklund

Contents

1 Introduction

This is the project plan for the Master Thesis in ”Evaluation of distributed tracing frameworks for microservice observability”. Most part of the introduction is taken directly from the project specification since it's the same content.

1.1 Background

A current trend in software is the transition from large monolithic applications (silos) to smaller, loosely coupled microservices. Microservices are commonly implemented using container technology and orchestrated with tools such as Kubernetes. The decentralized nature of microservices allows developer teams to work more independent of each other and thus improves velocity of software development. However, this decentralization also has drawbacks, as it becomes more difficult to debug microservices, as well as to diagnose any performance issues. To mitigate this, distributed tracing is an approach to monitor and debug microservice applications. It is based on modification of (HTTP) headers of the network messages exchanged between microservices, enabling full audit of the invocation patterns. In effect, cascading invocations across many microservices can be correlated, commonly back to a single call to a front-end service. Performance bottlenecks can also be detected as the invocation graph commonly is augmented with service response times. This functionality and complexity comes at a cost: application code must be modified to include tracing identifiers, and the handling of these identifiers (for each network message) in the software stack adds overhead as well as introduces additional latency.

1.2 Goal

This thesis will investigate how complex it is to use distributed tracing, as well as quantify the overhead such frameworks add to microservice execution. The maturity of distributed tracing implementations will also be assessed.

2 Objectives

To simplify, the objectives that this thesis should investigate is :

- To understand almost everything there is to know about distributed tracing.
- Fully understand a couple of distributed tracing frameworks.
- Learn and understand how a container (maybe Docker) works.
- Learn and understand how a container orchestrator (Kubernetes) works.

3 Literature and references

The field of distributed tracing is quite new so extensive reading must be done before starting this thesis. These are some ground literature that should be read before starting and more will come as the project ages.

- The original paper on Dapper which came from Google. Most tracing frameworks can be traced back to this paper.[1]
- An explanation on how to set up a tracing framework from a medium blogpost.[2]

4 Communication

This section will explain what kind of communication methods that are to be used in this Master Thesis.

4.1 Documents

All the documents such as the report, PDFs including project plan, project specification, etc. will be available in a private git repository. (Ask for invitation). The only thing that won't be included is the .tex files which will be stored on Overleaf.

4.2 Project diary

A project diary will be written each week to check how, time wise the project is going. The diary will contain information about what has been done each week, what problems occurred and what to do next week.

4.3 Meetings

There will be a meeting with the supervisor each Friday to go through what has been done that week. Also if any serious questions arises there is always someone to ask since the supervisor works in the same building.

5 Time plan

1. Project plan, refined specification, create project log: 1 week
2. Initial studies of tracing and associated tools in the CNCF landscape: 1 week
3. Study and select tracing frameworks: 1 week
    (a) CNCF Cloud Native Interactive Landscape
4. Testbed: 7 weeks
    (a) Kubernetes: 2 week
        i. Managed: AKS/EKS/GKE
        ii. Own installation (on VMs): kube-adm or the similar
    (b) Tracing tools (installation and configuration): 3 weeks
        i. Documentation of usability and project vitality (see research questions...)
    (c) Applications/benchmarks installation (pick one or a few): 2 weeks
        i. TeaStore [3]
        ii. HotROD (Jaeger) (likely best candidate...)
        iii. TrainTicket
        iv. SockShop
        v. HipsterShop
5. Conduct experiments: 6 weeks
    (a) Set up monitoring to gather relevant data: 2 week
    (b) Run all tests (many times for statistically significant data): 4 weeks
6. Final report writing: 4 weeks
7. Plan presentation and read other students report: 1 week

6 Risks

As with most Master Theses the biggest risk is that a proper result will be hard to get. The minimum goal is to at least get one framework going and to be able to run tests on at least one, which will ensure that some result can be achieved. This will make it possible to finish the thesis but without comparison of different frameworks. Some other questions that could be described as a risk is, is it enough to simulate an application? Is it enough to say for sure that it's a satisfactory replacement for a real world application? The risks will be investigated early in the thesis such that if necessary, the thesis specification can be altered early to ensure that the thesis will be finished.

References

[1] Benjamin H. Sigelman and Luiz André Barroso and Mike Burrows and Pat Stephenson and Manoj Plakal and Donald Beaver and Saul Jaspan and Chandan Shanbhag. Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. Technical report, Google, Inc., 2010. https://research.google.com/archive/papers/dapper-2010-1.pdf.

[2] Uzziah Eyee. Microservices Observability with Distributed Tracing. https://medium.com/swlh/microservices-observability-with-distributed-tracing-32ae467bb72a, June 2019.

[3] Jóakim von Kistowski, Simon Eismann, Norbert Schmitt, André Bauer, Johannes Grohmann, and Samuel Kounev. TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research. In *Proceedings of the 26th IEEE International Symposium on the Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS '18, September 2018.