**UMEÅ UNIVERSITY**
**Department of Computing Science**

**February 4, 2019**

# Improving Operations with Distributed Tracing

Project Plan

version 1.0

|  |  |
|---|---|
| **Name** | Niclas Nyström |
| **Netrounds Supervisor** | Joakim Söderberg |
| **University Supervisor** | Jerry Eriksson |
| **Examinator** | Henrik Björklund |

# Contents

# 1 Introduction

This is a project plan for my master thesis as the final examination towards my masters degree in computer science at Umeå University in the spring semester of 2019. The project is done in cooperation with *Netrounds* which specializes in network performance testing and quality assurance.

# 2 Problem Specification

## 2.1 Background

Monitoring in software development has since long remained the same without evolving much in terms of tools and standards. Logging output and saving it onto files are the standard for most applications and programs nowadays and has been so since back in the days when systems only consisted of a couple of servers in the same network.

But as the complexity of software systems increases for every day the difficulties of logging and monitoring the health- and performance of systems increases as well. As the current paradigm shift moving steadily towards distributed- and cloud-based systems modern programs faces increasing difficulties of maintaining the same level of monitoring ability as programs expands and consists of multiple servers spanning over several networks.

This results in challenges like how to keep and store metrics and logs from all components in the system, how and when to alert if anomalies are detected in the systems and how to distinguish the components as a whole and as individual components when monitoring them.

Solving the issue requires programs to have fine-grained control over all of their components in the system and having the ability to control and monitor each component from a high level.

Hence introducing **Open Tracing** API (**distributed tracing**) that can grant end-end visibility in systems by recording the journey of requests from one point to another. New open source frameworks that uses distributed tracing, like *Jaeger* and *Istio*, provides solutions for visualizing and analyzing cross-process requests between servers and networks and can be used for detecting and diagnosing anomalies and problems in systems and make distributed profiling.

More specific distributed tracing encapsulates a request/transaction into *traces* which can be seen as *directed acyclic graphs* (DAG) where each node is a component that the request passes through when travelling from A to B. The nodes in this context are called *spans* and contains info as specified from the system. The info can be logs and metrics that later can easily be collected by a single method from point A [1][2].

As new techiques and architecture becomes popular due to the shift towards distributed- and cloud-based systems such as microservices and event-driven architecture the question emerges whether distributed tracing can be used for these architectures as well. Originally distributed tracing was intended for systems that propagates requests from A to B by passing through a number of components inbetween. The new techniques which often emphasizes on loosely-couple components for easier understanding and increasing modularity may not follow that type of structure but instead follows a bus implementation that components/services subscribes and unsubscribes to.

## 2.2 Goals

The goal of this project is to implement and investigate how distributed tracing can be used to improve the operations of companies that deals with large distributed systems.

More specifically there are two goals to this project: The first goal is to implement a program that uses distributed tracing into a distributed environment, like a Kubernetes cluster where distributed tracing is a sidecar in a Kubernetes pod, and test how effective it can detect anomalies like latency issues and component errors in the cluster and investigate how large distributed systems can benefit from it.

The second goal is to investigate whether distributed tracing can be applied to modern architectures such as microservices and event-driven architecture. This project will focus primarily on event-driven bus architecture. This goal is of more hypothetical nature but may also contain a barebone implementation if it's possible and within the scope of this project.

# 3 Objectives

The main objectives of this project are as follows:

- Study of how (and if) distributed tracing can be applied to modern distributed- and cloud-based architecture patterns like *event-driven architecture.*
- Implement distributed tracing to investigate how it can handle and detect errors in large and complex systems. The program will be deployed as a sidecar in a *Kubernetes* cluster and tested accordingly.
- Study of how distributed tracing can help large and complex distributed systems to detect, monitor and analyze individual components in systems in order to detect anomalies such as high latancies.

### 3.1 Critera for Success

The criteria of success for this project is when it's determined whether distributed tracing can be applied to architectures like *event-driven architecture* or not. Furthermore the project will hinge on implementing a program with distributed tracing that can be deployed as a sub-container in a *Kubernetes* cluster in the objective of testing how it can improve operations in big and complex distributed- and cloud-based systems by monitoring and detecting errors or anomalies like big latencies or component failures.

### 3.2 Risks

Due of the experimental nature of this project there are few risks to this project since the scope of the experiments can always be modified in order to cope with the situation. There are close to no risks when it comes to the implementation since it exists frameworks and documentation to lean on.

The hypothetical part of combining distributed tracing with alternative architectures such as the event-driven architecture does not have any risks since every possible outcome is an answer to the question.

## 4 Method

This sections describes preliminary the methods, communication and tools used that will be used during the project. All (non-confidential) documentation will be stored in the following Github repository: Link.

### 4.1 Tools & Technologies

This project will use the following tools and technologies:

- AWS EC2
- Kubernetes
- Open-Source Frameworks that is based on the *Open Tracing API* a.k.a Distribued Tracing such as:
  - Jaeger/Zipkin
  - Istio

Note that the list is tentative and can be modified during the project as needed.

### 4.2 Workplace

All experiments will take place in the Netrounds office as they will provide with a desk and necessary equipment and tools in order to perform the experiments. It's also possible that some parts will be done from home.

### 4.3 Weekly report

Weekly diary (logs) will be provided as markdown documents in the linked repository. All logs will contain what I have accomplished, what problems that I've encountered for that given week and also what I will do the following week.

### 4.4 Documentation

All of the documentation regarding the master thesis (project plan, thesis etc) that are not related to *Netrounds* will be available in the repository.

### 4.5 Meetings

Meetings with the company supervisor, Joakim Söderberg, can easily be had when needed due to daily working in the office of Netrounds (and available in Slack and Skype).

Meetings with the university supervisor, Jerry Eriksson, will be had when needed and mostly through online platforms like Skype. It's also possible that some meetings will take place at the university.

## 5 Tentative Timeline

This section contains a timeplans for the activities, such as university- and project deadlines, that as of now are planned to take place during the project. Every activity will contain rough time estimates of how long it will take to implement. In the development timeline there is two weeks of reserve time at the end if some activities does not follow the timeline.

The subsections below divides the time plan by the following types of deadlines: hard-, report- and development deadlines.

## 5.1 Hard Deadlines

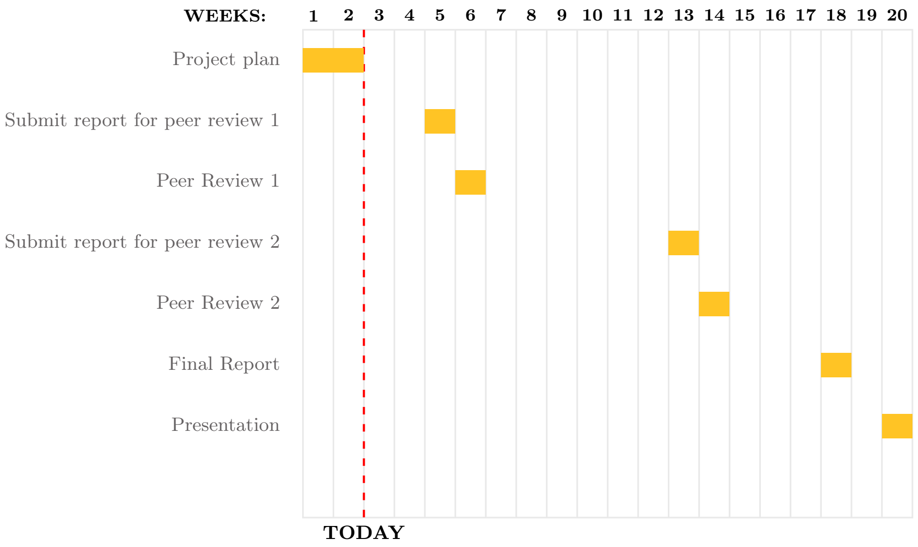

Figure 1: Timeline (20 weeks) for the hard deadlines from the university.

## 5.2 Report Deadlines

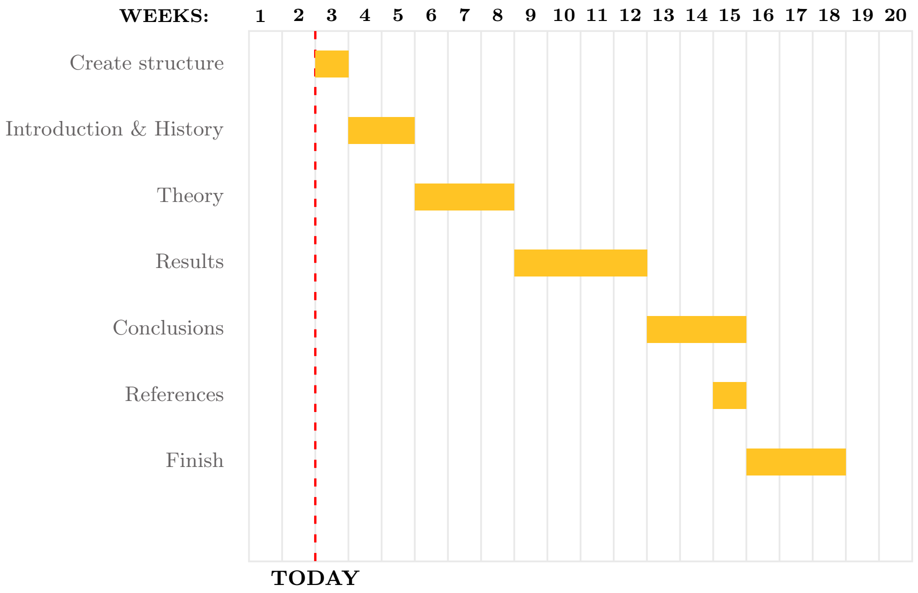

Figure 2: Timeline (20 weeks) for the final report deadlines (tentative).

## 5.3 Development Deadlines

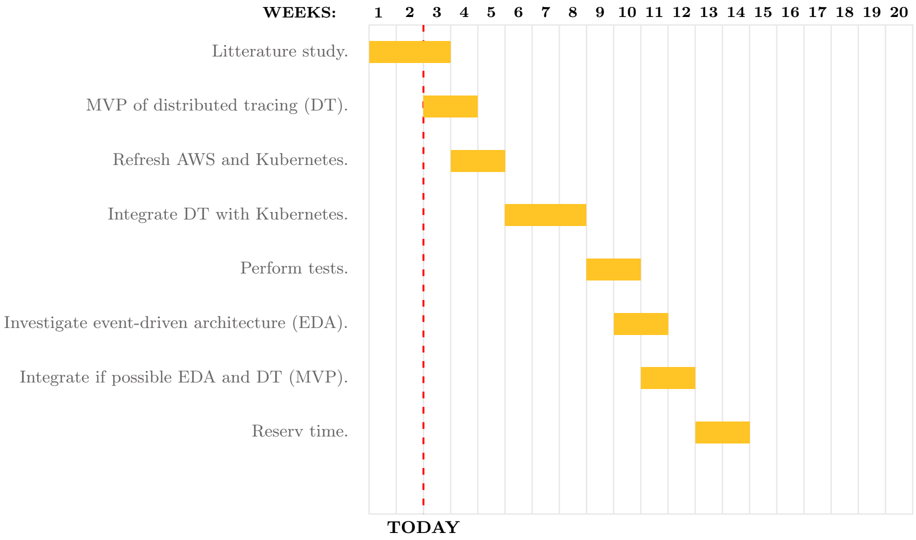

Figure 3: Timeline (20 weeks) for the development deadlines (tentative). MVP = *Minimum Viable Product*, EDA = *Event-driven architecture*, DT = *Distributed tracing*

# 6 Literature

The following literature will be used in the project:

- Open Tracing: Emerging Distributed Tracing Standard [2]
- Istio [3]
- Jaeger [4]

Note that the list above is the whitepaper of distributed tracing and documentation of the frameworks that may be used.

# References

[1] Sematext. *Open Tracing: Emerging Distributed Tracing Standard.*

[2] Open Tracing API,
`https://opentracing.io/`

[3] Istio,
`https://istio.io/`

[4] Jaeger,
`https://www.jaegertracing.io/`