UMEÅ UNIVERSITY **February 8, 2019**
Department of Computing Science

Improving Operations with Distributed Tracing

Project Plan

version 2.0

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.

1.1 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 between 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.

Traditionally end-end metrics, e.g. latency, helped systems to debug the performance of internal services. The rise of modern architectures however has resulted in the skewing of these metrics since it represents the system as a whole and does not describe the state of individual services. Distributed tracing solves that by breaking down the metrics on a per service base and traces the journey of the operations within the system which opens the door for in-depth profiling in form of *Root-cause analysis*.

1.2 Problem Specification

This idea originated from the need of being able to analyze performance issues in large and complex distributed systems. Distributed tracing can grant systems the ability to pin-point issues to individual services and can additionally provide data to perform profiling such that it becomes possible to answer why issues appeared and from where it originated from (root-cause analysis).

1.2.1 Research Questions

The main question of this project is as follows:

- *How can distributed systems improve it's operations by tracing requests in order to gather necessary data to perform effective root-cause analysis?*

As distributed systems strives to decrease the complexity of the system as a whole the consequence are that individual components/services gets more complex as a result of several factors such as heterogeneous and unreliability. As a result it gets harder to observe the system beyond the system as a whole and monitoring individual services are hard and expensive. *Improving operations* means in this case how we can give the system the means to effectively test and debug operations on a fine-grained level in order to achieve observability and efficiency. Whether it's a slow service causing high end-end latency or some component that aren't behaving properly due to the state of other components the goal of this project is to investigate how distributed tracing can help distributed system to become more fault-tolerant and scalable due to increased observability and availability.

1.3 Netrounds

Netrounds are as of now in the process of transitioning their system from a monolith solution to an architecture that is event-driven and consists of microservices/kubernetes. Their vision is to have their core-system converted into an event-driven architecture using Kafka as the message bus and tools like Druid for time-series storage.

This architecture is as of now under development and is close for first deployment and the third goal (and partially the second goal as well) is connected to how they can benefit of adding distributed tracing into that architecture.

The plan is to design a new design from the existing design but with distributed tracing added to it and test if it can provide some value to their system (pros/-cons).

1.4 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 three goals to this project: The <u>first</u> 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 <u>second</u> 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 a event-driven bus architecture. This goal is of more hypothetical nature but may also contain a minimal implementation if it's possible and within the scope of this project.

The <u>third</u> goal is connected to *Netrounds* and will consist of implementing distributed tracing to their platform in kubernetes form and perform tests to that architecture in order to analyze whether distributed tracing is beneficial to their system.

1.5 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.
- (Netrounds-1) Study of how distributed tracing can be beneficial to Netrounds existing architecture by creating a design with distributed tracing in mind, deploying the design and perform tests to evaluate pros and cons.
- (Netrounds-2) Deploy the design in the context of Netrounds and perform tests to evaluate pros and cons and provide recommendations.

Note: (Netrounds-x) is objectives specific to the company that I work together with during the project and the rest of the objectives are of a more general nature.

1.6 Critera for Success

The criteria of success for this project is when the implementation goals are achieved and enough data have been collected in order to make conclusions in regards to the objectives and goals that aren't specific to Netrounds. The specific goals and objectives does not determine the success of the project but rather it's beneficiary to the company.

1.7 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.

The only risks that exists are the goals that are linked with the company but those does not effect the success of the thesis.

2 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.

2.1 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.

2.2 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.

2.3 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.

2.4 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.

2.5 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.

3 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.

3.1 Hard Deadlines

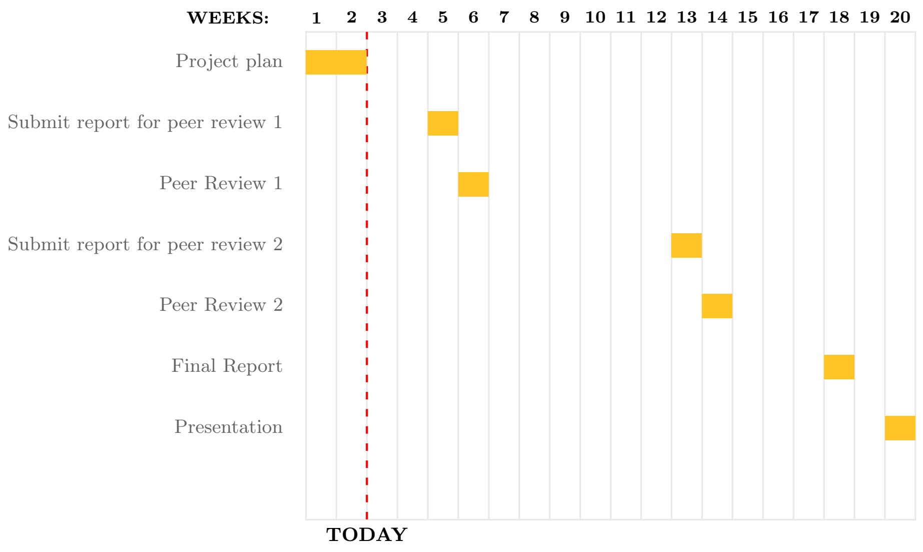

Figure 1: Timeline (20 weeks) for the hard deadlines from the university.

3.2 Report Deadlines

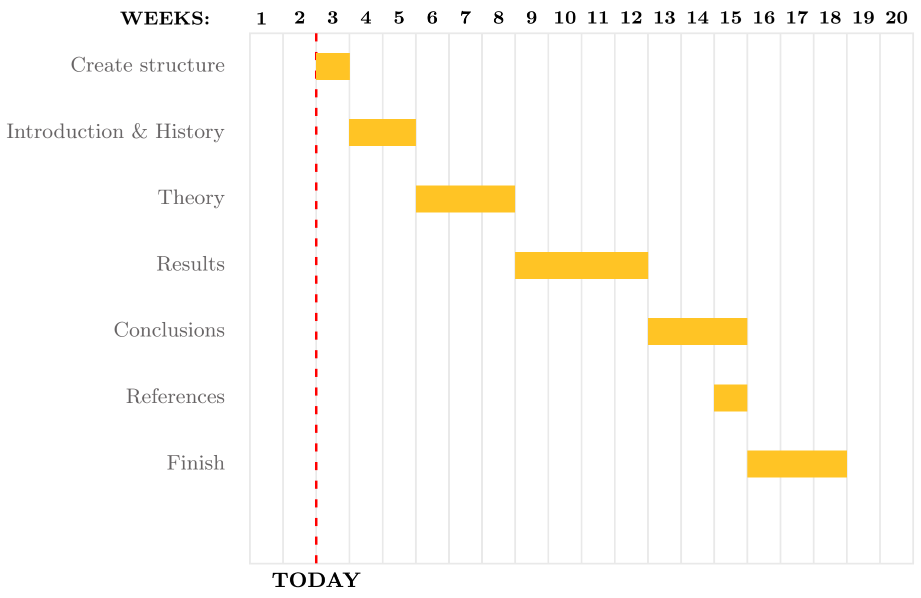

Figure 2: Timeline (20 weeks) for the final report deadlines (tentative).

3.3 Development Deadlines

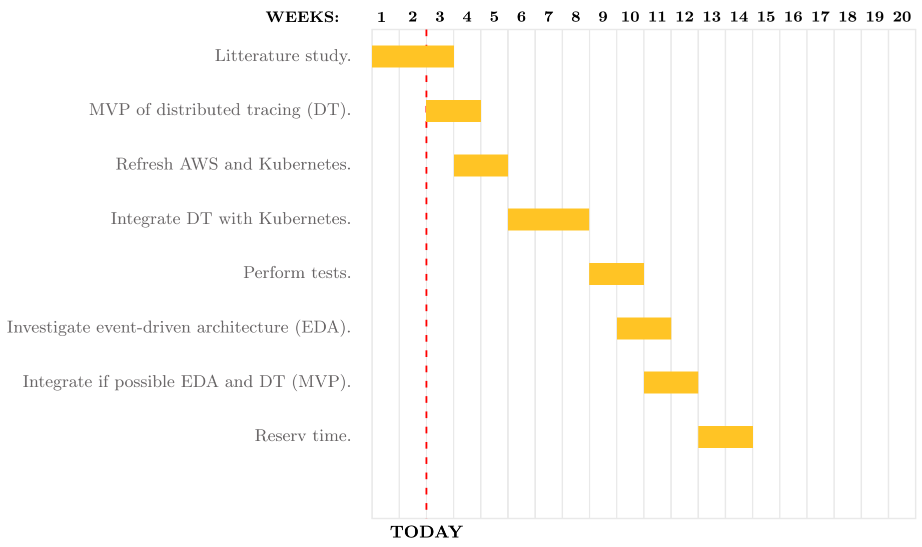

Figure 3: Timeline (20 weeks) for the development deadlines (tentative). MVP = *Minimum Viable Product*, EDA = *Event-driven architecture*, DT = *Distributed tracing*

4 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/`