THM | CAMPUS GIESSEN | MNI
TECHNISCHE HOCHSCHULE MITTELHESSEN
Mathematik, Naturwissenschaften und Informatik

Sommersemester 2019

Exposé

Dozent

Prof. Dr. Harald Ritz

Betreuer

M.Sc. Pascal Bormann

Autor

Name	**Matrikelnummer**
Simon Stockhause	5143959

29. Oktober 2019

Inhaltsverzeichnis

1 Problemstellung

In einem System werden Nachrichten ausgetauscht. In Falle eines verteilten Systems spielt dabei die Kommunikation über Prozessgrenzen eine zentrale Rolle. Entscheidende Ereignisse und deren zeitliches Auftreten sind von besonderem Interesse. Diese Ereignisse werden Events genannt. Events bilden einzelne Zeitpunkte ab. Die Intrasystem-Netzwerkkommunikation bedarf Konzepte zur Nachvollziehbarkeit von Events und deren Beziehungen zueinander.

Das System generiert asynchron *Frames* und sendet diese in gleichbleibenden Intervallen über eine Websocketverbindung an einen Client. Frames werden verworfen, sobald neuere Frames generiert wurden, d.h. innerhalb eines Intervals können mehrere Frames entstehen, aber nur eines ist relevant. Der Client kann die kommenden Frames durch Übermittlung von Daten über den Websocket beeinflussen. Die bei diesem Datenaustausch entstehenden Events wie z.B. das Starten einer Framegenerierung, dem Beending einer Framegenerierung, dem Senden eines fertiggestellten Frames und dem Empfangen eines Frames sollen erstellt werden. Die zeitliche Einordnung der Events hat dabei eine zentrale Rolle einzunehmen. Das Konstrukt von Events, die zueinander in Verbindung stehen, nennt sich *Trace*. Dazu stellt sich folgende Frage: *Ist es möglich Events zu generieren und miteinander in Verbindung zu setzen, die es erlauben, einen Stream von Frames als Trace darzustellen.*

2 Erkenntnisinteresse

Aufgrund des streambasierenden Ansatzes des Systems, welcher kontinuierlich Frames an einen Client sendet, sollen die Unterschiede zu Request-Response basierender Kommunikation aufgezeigt werden. Zumdem soll festgestellt werden, welche Daten nötig sind, um eine Kausalordnung zwischen Events herzustellen. Dabei ist die Ordnung der Events in einem asynchronen Umfeld zu betrachten. Hierbei könnten Designunterscheide aufgezeigt werden. Diese wären z.B. das Speichern der Events in verteilten Datenbanken oder das *Piggybacking*, d.h. das Mitführen von Tracemetadaten, welche die Events beinhaltet, über Prozessgrenzen hinweg.

3 Zielsetzung

Es soll eine Bibliothek entwickelt werden, die es ermöglicht Tracingevents zu generieren. Dazu muss ein Datenmodell dieser Events entwickelt werden. Die Events könnten entweder in einer zentralen Stelle(Piggybacking) oder in dezentralen Stellen(verteilte Datenbank) gesammelt und anschließend geordnet und visualisiert werden.

4 Forschungsstand

Es gibt diverse Konzepte und Werkzeuge zur Erhebung von Tracingdaten. Darunter zählen Instrumentalisierungsbibliotheken von z.B.:

- Zipkin
- Jaeger
- Opentracing
- Brown Tracing Framework
- X-Trace

Abgesehen von dem Brown Tracing Framework und dem X-Trace verwenden alle genannten Ansätze das spanbasierte Datenmodell. Die Brown University präsentiert in ihrer Veröffentlichung *Universal Context Propagation for Distributed System Instrumentation* eine Schichten-Architektur zur Übermittlung von Tracingdaten in einem spezifizierten *Baggage Context*. Der Baggage Context wird als Metadata mitgereicht und stellt somit eine Form des Piggybacking dar. Das Paper *End-to-End Tracing Models: Analysis and Unification* beschreibt das spanbasierte Modell als eine Sammlung von *spans*, welche jeweils eine Block von Rechenarbeit darstellt. spans representieren einen Zeitinterval, weshalb sie eine Anfangs- und Endzeit benötigen.[1]

5 Gliederung

- Einleitung
  - Motivation
  - Problemstellung
  - Foschungsstand
  - Thesisübersicht
- Grundwissen / Themeneinstieg / Themenüberblick
  - Verteilte Systeme
    - Überwachung von verteilte Systemen
    - Synchronisation
    - Ordnung von Events
  - Bibliotheksentwicklung
- Problembeschreibung
  - Eventgenerierung
    - Eventkorrelation
    - Synchronisation zwischen Eventgeneratoren
  - Eventübermittlung
- Design
  - Datenmodell
    - Eventmodell
    - Eventgraph
  - Verarbeitsmodell
    - Agenten
    - Collector
- Implementierung
  - Bibliothek: Eventgenerator
  - Eventagent

[1] [Lea14] "End-to-End Tracing Models: Analysis and Unification". 2014.

    - Eventcollector
- Evaluierung
    - Genauigkeit der Eventgenerierung
    - Darstellung der Events
    - Vergleich mit Jaeger
        * Datenmodelle
        * Bereitstellung
        * Ergebnisse
- Fazit

6 Zeitplan

- Gesamtzeitraum: 13.01 - 13.04 (91 Tage)
- Recherche: 13.01 - 20.01 (7 Tage)
- Implementierung: 21.01 - 14.02 (24 Tage)
- Schreiben: 15.02 - 25.03 (39 Tage)
- Abschluss: 16.03 - 13.04 (28 Tage)

Literaturverzeichnis

[Lea14] Jonathan Leavitt. “End-to-End Tracing Models: Analysis and Unification”. In: *Department of Computer Science Brown University* (5. Mai 2014).