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

Generierung und Ordnung von Events in verteilten Systemen mit asynchroner Kommmunikation

Bachlorthesis
Studiengang Informatik

vorgelegt von
Simon Stockhause

Mai 2020

Referent der Arbeit: Prof. Dr. Harald Ritz
Korreferent der Arbeit: M.Sc. Pascal Bormann

Zusammenfassung

Inhaltsverzeichnis

Abbildungsverzeichnis

1 | Einleitung

1.1 Motivation

Die heutigen Bedürfnisse der Anwender ein stets erreichbaren, fehlerfreien und ??schnellen?? Service zur Verfügung zu haben, stellt hohe Erwartung an Unternehmen. Um den Ansprüchen der Nutzer gerecht zu werden, müssen Systeme gewährleisten, dass ein gewisser Grad von Beobachtbarkeit des Systems erreicht wird. Die Beobachtbarkeit sorgt für die nötige reaktionsfähigkeit der Entwickler und Operatoren, um möglicherweise auftretende Komplikationen, die die Benutzerbedürfnisse beeinträchtigen, schnell, präzise und langfristig beheben zu können.

Die Komplexität des Gesamtsystems, welches aus vielen kleinen Komponenten bestehen kann, ist eine große Herausforderung für Entwickler und Operatoren. Die enorme Skalierbarkeit einzelner Komponenten und die aussgezeichnete Resourcennutzung der Hardware löst zwar viele Probleme der Vergangenheit, wie zum Beispiel Überbelastung einzelner Knoten, Ausfall von Komponenten und Latenzprobleme. Allerdings schafft diese Umstellung neue Schwierigkeiten, die es zu bewältigen gilt.

Die Instrumentalisierungsbibliothek *Traktor* soll Events innerhalb eines Systems generieren und ordnen, sodass während der Entwicklungsphase eines Systems Fehlerquellen lokalisiert werden können. Bei der Entwicklung der Tracingbibliothek sollen Standards ermittelt, analysiert und umgesetzt werden. Es sollen Erfahrungswerte in der Domain der Beobachtbatkeit von Systemen, durch die Entwicklung einer Instrumentalisierungsbibliothek gewonnen werden. Datenmodelle sind zu entwickeln.

Glossarbeispiel latex Glossarbeispiel Frames per Second (FPS)

1.2 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 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. Das System generiert Frames innerhalb von 16ms. Die Antwortzeiten betragen ca. 100ms. Der Client kann asynchron Einfluss auf die zu generierenden Frames durch Übermittlung von Daten nehmen. 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. Die Eventgenerierung muss sich mit den zeitlichen Rahmenbedingungen vereinbaren lassen. Das Konstrukt von Events ,die zueinander in Verbindung stehen, soll untersucht werden. Dazu stellt sich folgende Frage: *Ist es möglich Events in einem verteilten System zu generieren und miteinander in Verbindung zu setzen, die es erlauben, einen Stream von Frames als eine Anordnung von Events, die kausal miteinandern verbunden sind, darzustellen*

1.3 Foschungsstand

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.[1]

1.4 Thesisübersicht

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

2 | Themenüberblick

2.1 Verteilte Systeme

Verteilte Systeme nach Tanenbaum:

> Ein verteiltes System ist eine Kollektion unabhängiger Computer, die den Benutzern als ein Einzelcomputer erscheinen[1]

2.1.1 Eigenschaften eines verteilten Systems

- Knoten
- Einzelne Knoten führen einzelne Komponten des Systems aus
- Einzelne Komponenten sind für einzelne Aufgaben zuständig

2.1.2 Überwachung von verteilten Systemen

- Infrastruktur
  - CPU
  - GPU
  - Speicher
- Netzwerk
  - Anzahl Packete
  - Latenz zwischen Komponenten bzw. Client
  - Übertragungsanomalien
- Anwendung
  - Logs

[1] [TS06] *Distributed Systems: Principles and Paradigms (2nd Edition)*. 2006.

  - Metriken
  - Graphen etc.

Quelle beziehen: Verteilte Systeme Zitate

2.1.3 Synchronisation

-
-
-

2.1.4 Ordnung von events

- Bezug auf Lamports Eventordnung[2]

2.2 Bibliotheksentwicklung

-
-
-

2.3 Distributed Tracing

Verteilte Systeme wurden bereits definiert. Allerdings lohnt es sich eine alternative Definition zu betrachten.

> Ein verteiltes System ist ein System, mit dem ich nicht arbeiten kann, weil irgendein Rechner abgestürzt ist, von dem ich nicht einmal weiß, dass es ihn überhaupt gibt.[3]

Diese Definition lässt sich so auffassen, dass Lamport im Jahr 1987 die Problematik der Fehlersuche während der Laufzeit, des Debuggings während der Entwicklungsphase und des organisatorischen Aufwands im Allgemeinen, also damit der grundsätzlich hohen Unübersichtlichkeit und Komplexität von verteilten Systemen, beschreibt. Diese Probleme und Herausforderungen, mit denen man in der heutigen Zeit zusammenstößt, eventuell sogar noch intensiver, als zur damaligen Zeit, können durch distributed tracing angegangen werden.

[2] [Lam78] „Time, Clocks, and the Ordering of Events in a Distributed System“. 1978.
[3] [Lam87] *Distributed Systems Definition by Lamport*. 1987.

- Beschreibe, was distributed tracing werkzeuge bezwecken
- Beschreibe, was blackbox bezweckt und warum es nicht das gleiche ist
- beschreibe, was metriken sind: bezug zu überwachung
- beschreibe was logs sind: bezug zu überwachung
- beschreibe was traces sind
- beschreibe ziele von distributed tracing
- beschreibe nicht-ziele von distributed tracing
-

3 | Problembeschreibung

3.1 Eventgenerierung

3.1.1 Eventkorrelation

3.1.2 Synchronistion von Eventgeneratoren

3.2 Eventübermittlung

4 | Design

4.1 Anforderungsanalyse

4.1.1 Anforderungen

4.1.1.1 Funktionale Anforderungen

4.1.1.2 Nicht-Funktionale Anforderungen

4.2 Datenmodell

4.2.1 Eventmodell

4.2.2 Eventgraph

4.3 Verarbeitungsmodell

4.3.1 Agenten

4.3.2 Collectoren

5 | Implementierung

5.1 Bilbiothek: Traktor

5.2 Traktor Agent

5.3 Traktor Registry

6 | Evaluierung

6.1 Genauigkeit der Eventgenerierung

6.1.1 Uhren und Zeit

6.2 Darstellung der Events

6.3 Vergleich mit Jaeger

6.3.1 Datenmodelle

6.3.2 Bereitstellung

6.3.3 Ergebnisse

7 | Fazit

7.1 Ausblick

7.2 ZitatTest

[MRF15] [Sam+16] [MF18] [Bar+04] [Rey+06] [ACF12] [NCK19] [Bey+16] [Kal+17] [Bar+03] [Red] [SCR18] [Sig+10] [Ste01] [Fon+07] [Wat17] [Ope19]

Glossar

latex Is a mark up language specially suited for scientific documents. 1

Abkürzungsverzeichnis

FPS Frames per Second. 1

Literatur

[ACF12] Mona Attariyan, Michael Chow und Jason Flinn. „X-Ray: Automating Root-Cause Diagnosis of Performance Anomalies in Production Software“. In: *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*. OSDI'12. USA: USENIX Association, 2012, S. 307–320. ISBN: 9781931971966.

[Bar+03] P Barham, R Isaacs, R Moertier und D Narayanan. „Magpie: online modelling and performance-aware systems“. In: *Proceedings of USENIX HotOS IX* (2003).

[Bar+04] Paul Barham, Austin Donnelly, Rebecca Isaacs und Richard Mortier. „Using Magpie for Request Extraction and Workload Modelling“. In: *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*. OSDI'04. USA: USENIX Association, 2004, S. 18.

[Bey+16] B Beyer, C Jones, J Petoff und N R Murphy. *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Incorporated, 2016. ISBN: 9781491929124. URL: https://books.google.de/books?id=81UrjwEACAAJ.

[Fon+07] R Fonseca, G Porter, R Katz, S Shenker und I Stocia. „X-Trace: A Pervasive Network Tracing Framework“. In: *Proceedings of USENIX NSDI* (2007).

[Kal+17] J Kaldor, J Mace, M Bejda, E Gao, W Kuropatwa, J O'Neill, K Win Ong, B Schaller, P Shan, B Viscomi, V Venkataraman, K Veeraraghavan und Y Jiun Song. „Canopy: An End-to-End Performance Tracing And Analysis System“. In: *SOPS 2017* (2017).

[Lam78] Leslie Lamport. „Time, Clocks, and the Ordering of Events in a Distributed System“. In: *Communications of the ACM* 21.7 (Juli 1978), S. 558–565. ISSN: 15577317. DOI: 10.1145/359545.359563.

[Lam87] Leslie Lamport. *Distributed Systems Definition by Lamport*. 1987. URL: https://lamport.azurewebsites.net/pubs/distributed-system.txt (besucht am 29. 02. 2020).

[Lea14] Jonathan Leavitt. „End-to-End Tracing Models: Analysis and Unification“. In: *D* (2014). URL: http://cs.brown.edu/%7B~%7Drfonseca/pubs/leavitt.pdf.

[MF18] Jonathan Mace und Rodrigo Fonseca. „Universal Context Propagation for Distributed System Instrumentation“. In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18. New York, NY, USA: Association for Computing Machinery, 2018. ISBN: 9781450355841. DOI: 10.1145/3190508.3190526. URL: https://doi.org/10.1145/3190508.3190526.

[MRF15] Jonathan Mace, Ryan Roelke und Rodrigo Fonseca. „Pivot Tracing: Dynamic Causal Monitoring for Distributed Systems“. In: *Proceedings of the 25th Symposium on Operating Systems Principles*. SOSP '15. New York, NY, USA: Association for Computing Machinery, 2015, S. 378–393. ISBN: 9781450338349. DOI: 10.1145/2815400.2815415. URL: https://doi.org/10.1145/2815400.2815415.

[NCK19] S Nedelkoski, J Cardoso und O Kao. „Anomaly Detection and Classification using Distributed Tracing and Deep Learning“. In: *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 2019, S. 241–250. DOI: 10.1109/CCGRID.2019.00038.

[Ope19] OpenTracting. *opentracing trace overview figure*. 2019. URL: https://opentracing.io/docs/overview/.

[Red] Inc Red Hat. *Was ist IT-Automatisierung?* URL: https://www.redhat.com/de/topics/automation/whats-it-automation.

[Rey+06] Patrick Reynolds, Charles Killian, Janet L Wiener, Jeffrey C Mogul, Mehul A Shah und Amin Vahdat. „Pip: Detecting the Unexpected in Distributed Systems“. In: *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3*. NSDI'06. USA: USENIX Association, 2006, S. 9.

[Sam+16] Raja R Sambasivan, Ilari Shafer, Jonathan Mace, Benjamin H Sigelman, Rodrigo Fonseca und Gregory R Ganger. „Principled Workflow-Centric Tracing of Distributed Systems“. In: *Proceedings of the Seventh ACM Symposium on Cloud Computing*. SoCC '16. New York, NY, USA: Association for Computing Machinery, 2016, S. 401–414. ISBN: 9781450345255. DOI: 10.1145/2987550.2987568. URL: https://doi.org/10.1145/2987550.2987568.

[SCR18] MARIO SCROCCA. „Towards observability with (RDF) trace stream processing“. Diss. Politecnico Di Milano, 2018. URL: http://hdl.handle.net/10589/144741.

[Sig+10] Benjamin Sigelman, Luiz André Barroso, Mike Burrows, Pat Stephenson, Manoj Plakal, Donald Beaver, Saul Jaspan und Chandan Shanbhag. „Dapper, a Large-Scale Distributed Systems Tracing Infrastructure“. In: *Google Techical Report dapper-2010-1* (2010).

[Ste01] William Stearns. *Ngrep and regular expressions to the rescue*. http://www.stearns.org/. 2001. URL: http://www.stearns.org/doc/ngrep-intro.current.html.

[TS06] Andrew S Tanenbaum und Maarten van Steen. *Distributed Systems: Principles and Paradigms (2nd Edition)*. USA: Prentice-Hall, Inc., 2006. ISBN: 0132392275.

[Wat17] Matt Watson. *8 Key Application Performance Metrics & How to Measure Them*. 2017. URL: https://stackify.com/application-performance-metrics/.