Chaos Engineering

13.11.2018 Dennis Pietruck

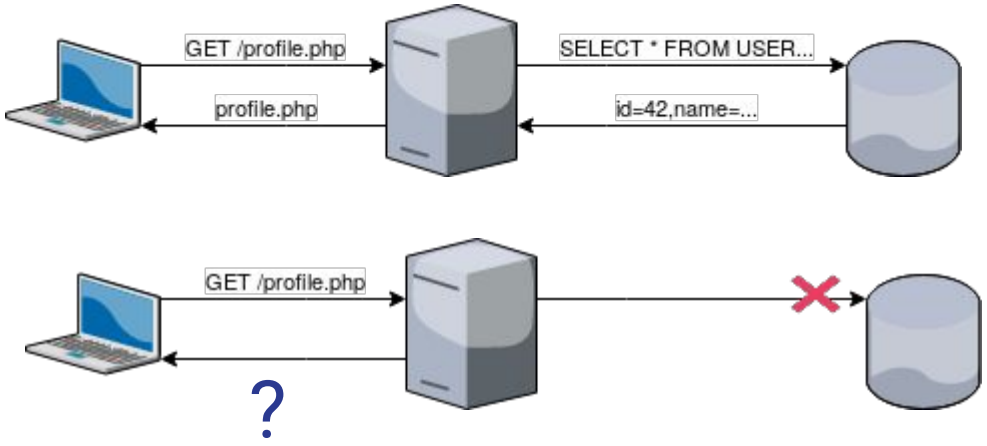
Agenda

- Motivation
- Grundlagen
- Forschung & Konferenzen

Motivation

Fehler in einem Monolith

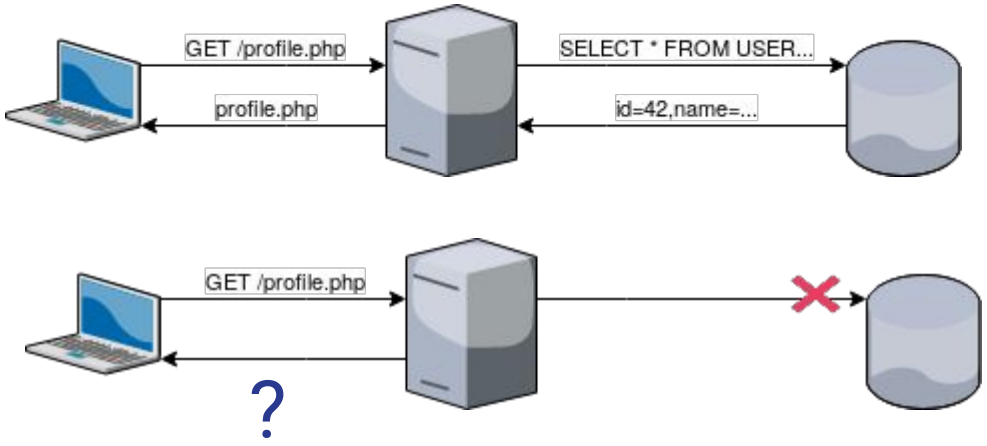
Was sieht der Client?



Fehler in einem Monolith

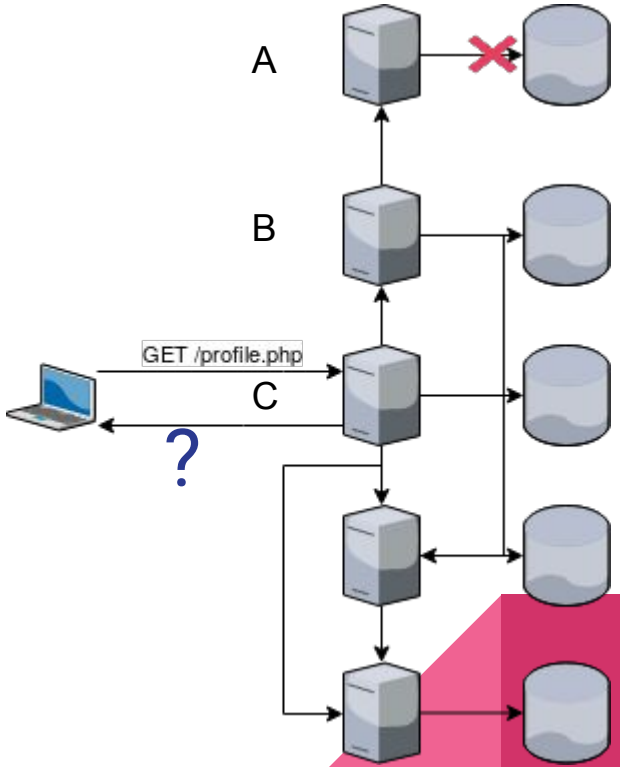
Was sieht der Client?

- Internal Server Error - 500
- Timeout



Fehler in einem verteilten System

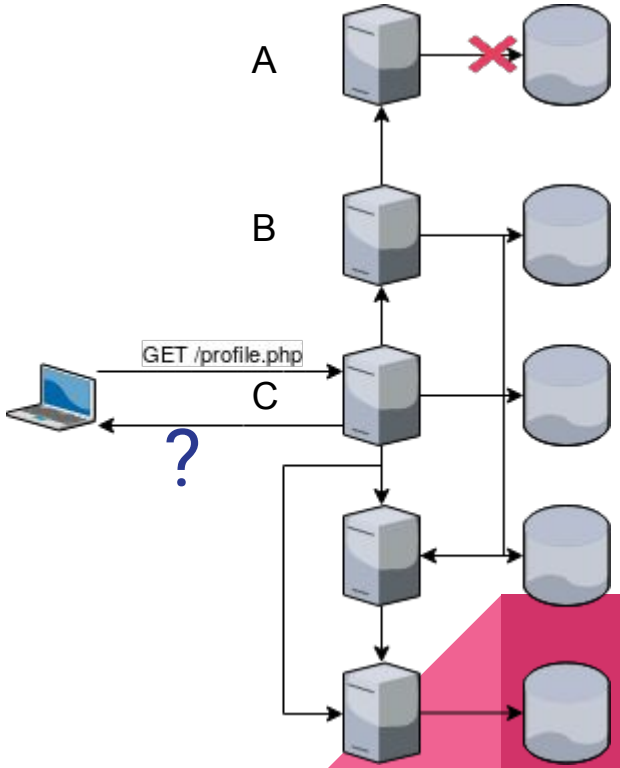
Was sieht der Client?



Fehler in einem verteilten System

Was sieht der Client?

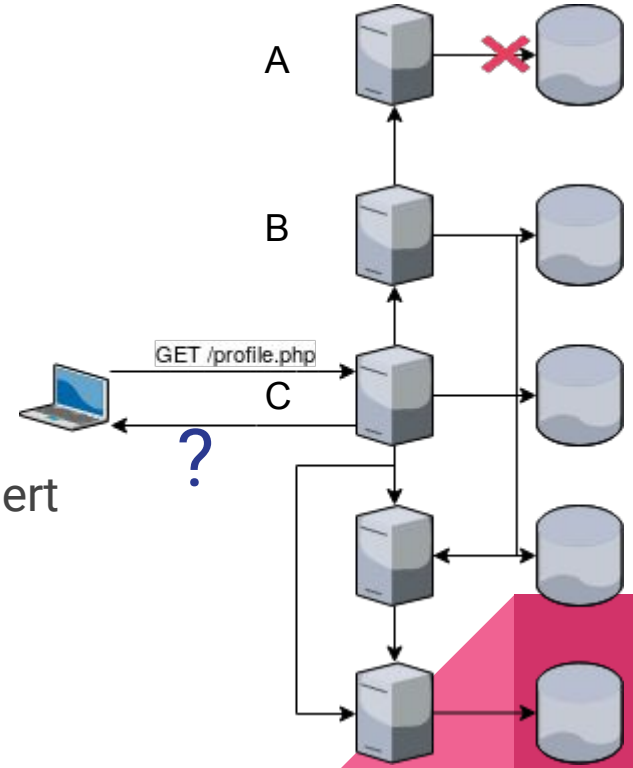
- Internal Server Error - 500
- Timeout
- OK - 200 (mit fehlenden Daten)
  - nach langem Warten



Fehler in einem verteilten System

Wie kann sich dieser Fehler auswirken?

- Keine Fehlerbehandlung
  - A kann auf DB Fehler nicht reagieren
  - B kann auf Fehler in A nicht reagieren
  - C kann auf Fehler in B nicht reagieren
- A hat Retries nicht konfiguriert
- A und B haben Timeout nicht konfiguriert



Verteilte Systeme heute

Josh Evans - A Netflix Guide to Microservices
https://de.slideshare.net/JoshEvans2/mastering-chaos-a-netflix-guide-to-microservices

Grundlagen

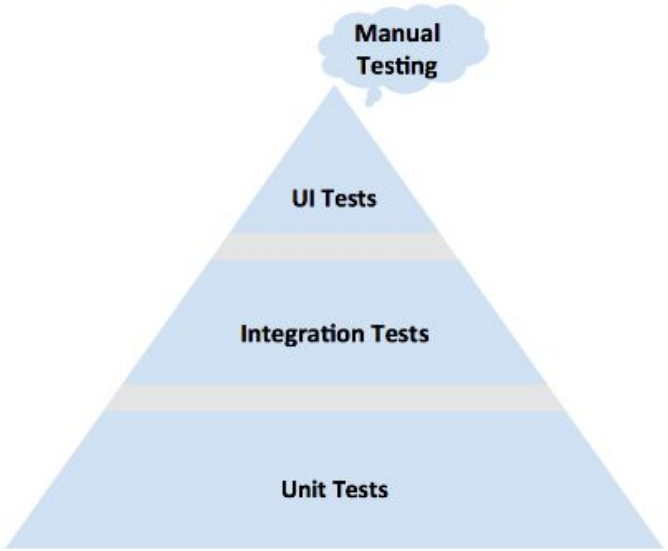
Was ist Chaos Engineering?

Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system's capability to withstand turbulent conditions in production.

principlesofchaos.org

Testing ≠ Chaos Engineering

Entspricht das Systemverhalten der Spezifikation?



https://www.a1qa.com/blog/interview-with-daniel-knott-upside-down-testing

Was passiert wenn…

- Latenz erhöht wird?
- Services abstürzen?
- Netzwerkkomponenten ausfallen?
- Regionen nicht erreichbar sind?
- …

Ursprünge von Chaos Engineering (1/2)

- 2010
  Netflix entwickelt Chaos Monkey
  für den Umzug nach AWS
- 2011
  Simian Army erweitert Chaos Monkey
- 2012
  Chaos Monkey auf github

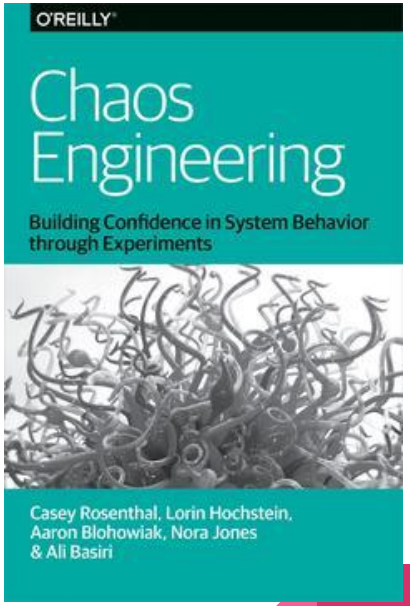
Chaos Monkey

Simian Army

Ursprünge von Chaos Engineering (1/2)

- 2016
  Gremlin Inc. bietet “Failure as a service”
- 2017
  Chaos Engineering Book

Gremlin



Vorgehen

Voraussetzung: Monitoring und Tracing

1. Messbaren stabilen Zustand definieren
2. Annehmen, dass dieser Zustand weiter besteht
3. Fehler simulieren
4. Annahme überprüfen
5. Maßnahmen treffen

Monitoring

Whitebox Monitoring (RED-Principle)

Blackbox Monitoring (USE-Principle)

Tracing

- Zeigt Kommunikation der Services
- Zeigt Latenzen
- Unterstützt Debugging

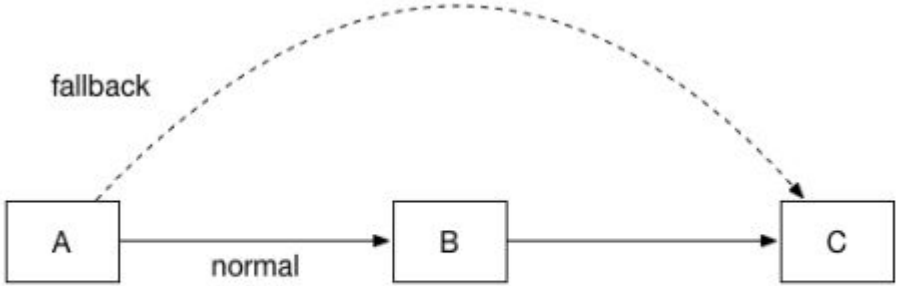
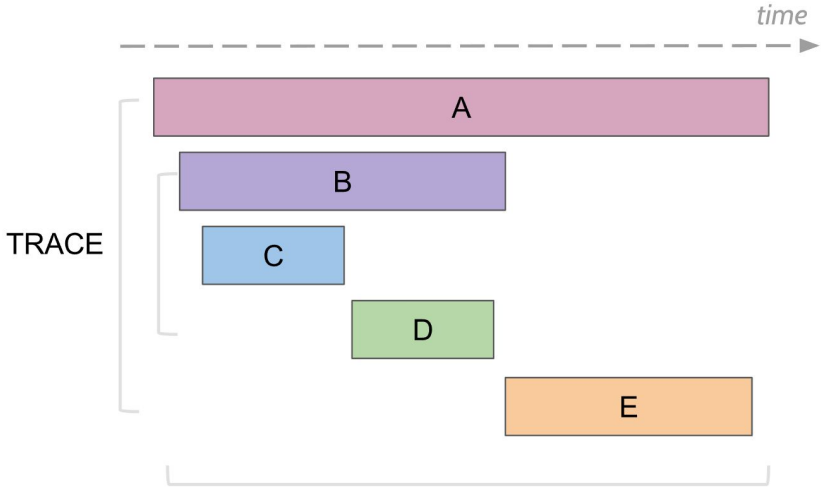


Figure 1. Unexpected fallback behavior



jaegertracing.io/docs/1.7/architecture/

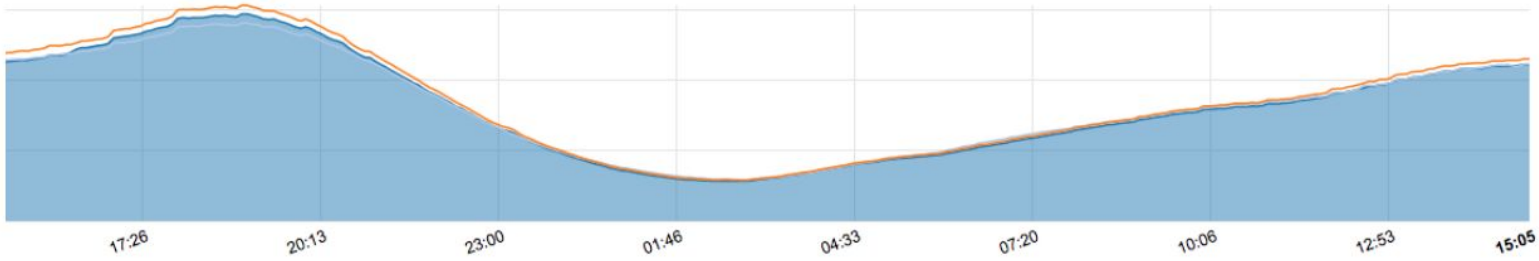
[1]

A. Blohowiak, A. Basiri, L. Hochstein, und C. Rosenthal, „A Platform for Automating Chaos Experiments“

Stabilen Zustand definieren

Metriken finden die zeigen, dass das System noch funktioniert

Bei Netflix: “Stream Starts per Second”



Streams per Second im stabilen Zustand

A. Basiri *u. a.*, „Chaos Engineering“, *IEEE Software*, Bd. 33, Nr. 3, S. 35–41, Mai 2016.

Fehler simulieren

- Nur für ausgewählte Komponenten
- Für einen kleinen Anteil der Benutzer

Tools:

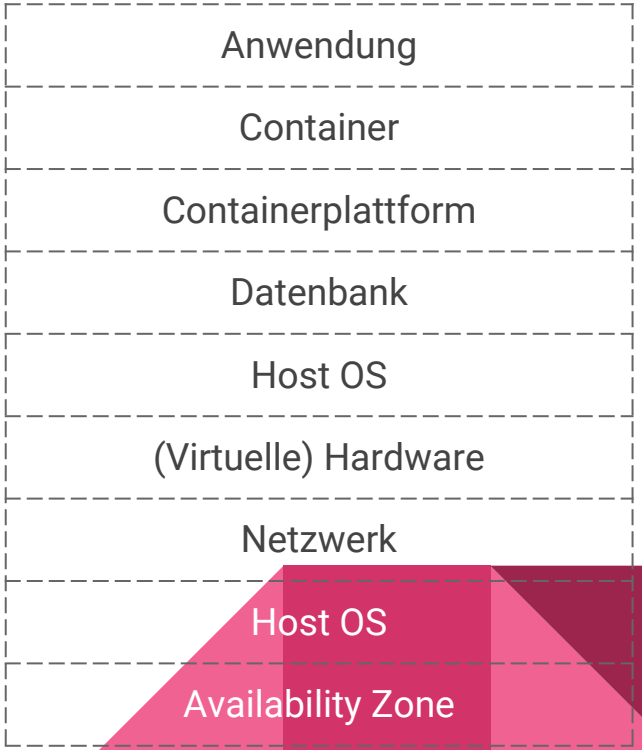
Chaos Toolkit

Chaos Monkey

Toxi Proxy

Pumba

Mögliche Fehlerquellen:



Forschungsthemen

Forschungsthemen

- Tooling
  - Fehlersimulation
  - Automatisierung
    - der Experimente
    - der Auswertung
- Anpassung an kleinere Organisationen
- Chaos Engineering in nicht technischen Bereichen

Forschungsthemen

Paper

- The Business Case for Chaos Engineering
  H. Tucker, L. Hochstein, N. Jones, A. Basiri, und C. Rosenthal
- A Platform for Automating Chaos Experiments
  A. Blohowiak, A. Basiri, L. Hochstein, und C. Rosenthal
- Why is random testing effective for partition tolerance bugs?
  R. Majumdar und F. Niksic

Konferenzen

- IEEE International Conference on Cloud Computing
- IEEE International Symposium on Software Reliability Engineering

The Business Case for Chaos Engineering

- Quantitative Vorteile
  - Return on Investment
    Voraussetzung: bestehendes Monitoring & Incident Management
    Kosten-Nutzen Verhältnis für Chaos Engineering

- Qualitative Vorteile
  - Förderung zur Entwicklung von Widerstandsfähigen Anwendungen
  - Widerstandsfähigkeit als Priorität

Automating Failure Testing Research at Internet Scale

- Chaos Automation Platform
- Konfiguration eines Experiments über ein Dashboard
- Automatische Ausführung des Experiments
- Auswahl von Metriken über ein Dashboard

Weitere Entwicklung:

- Automatische Konfiguration von Tests
- Automatische Auswertung der Tests

Quellen

A. Basiri *u. a.*, „Chaos Engineering“, *IEEE Software*, Bd. 33, Nr. 3, S. 35–41, Mai 2016.

„Principles of Chaos Engineering“. [Online]. Verfügbar unter: https://principlesofchaos.org/. [Zugegriffen: 10-Nov-2018].

A. Blohowiak, A. Basiri, L. Hochstein, und C. Rosenthal, „A Platform for Automating Chaos Experiments“, in *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2016, S. 5–8.

H. Tucker, L. Hochstein, N. Jones, A. Basiri, und C. Rosenthal, „The Business Case for Chaos Engineering“, *IEEE Cloud Computing*, Bd. 5, Nr. 3, S. 45–54, Mai 2018.

R. Majumdar und F. Niksic, „Why is random testing effective for partition tolerance bugs?“, *Proceedings of the ACM on Programming Languages*, Bd. 2, Nr. POPL, S. 1–24, Dez. 2017.