1.– 4. September 2014
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Trau, schau, wem!

Absicherung gegenüber externen Diensten mit Hystrix

Alexander Schwartz

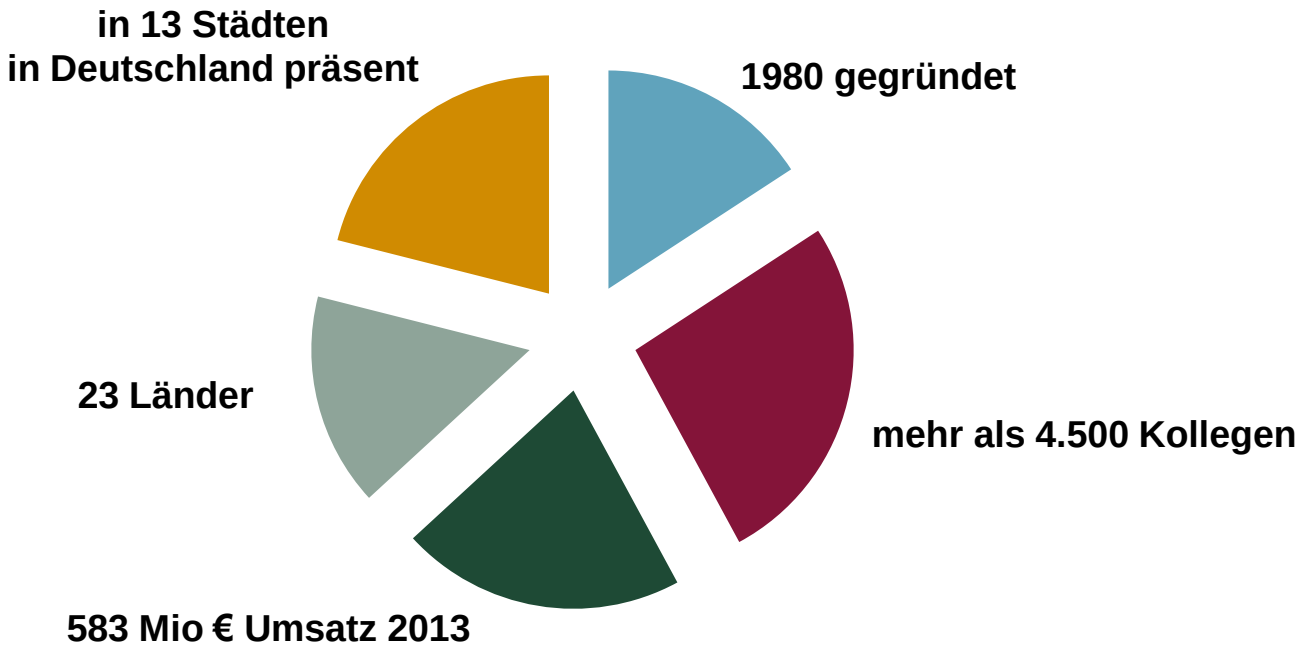
msg systems ag

AGENDA

1. Anwendungen mit externen Diensten
2. Einbau von Hystrix Schritt für Schritt
3. Anwendungsüberwachung mit Hystrix
4. Hystrix – für jede Anwendung geeignet?
5. Zusammenfassung

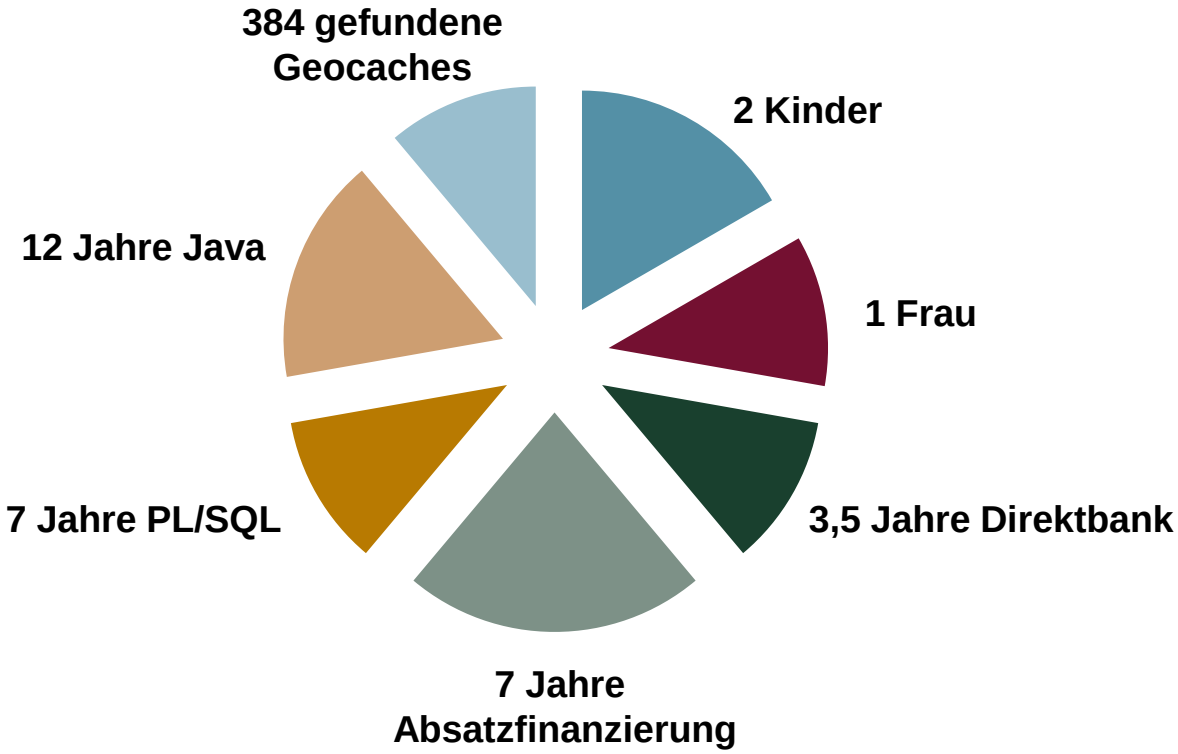
Mein Sponsor und Arbeitgeber

msg systems ag



Alexander Schwartz

Principal IT Consultant im GB Travel und Logistics

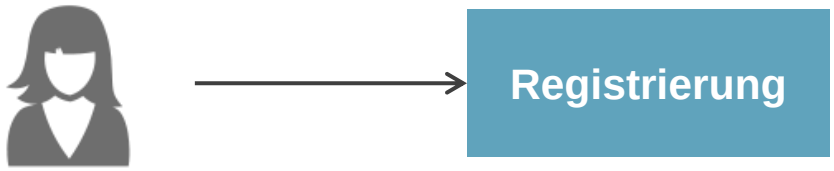


AGENDA

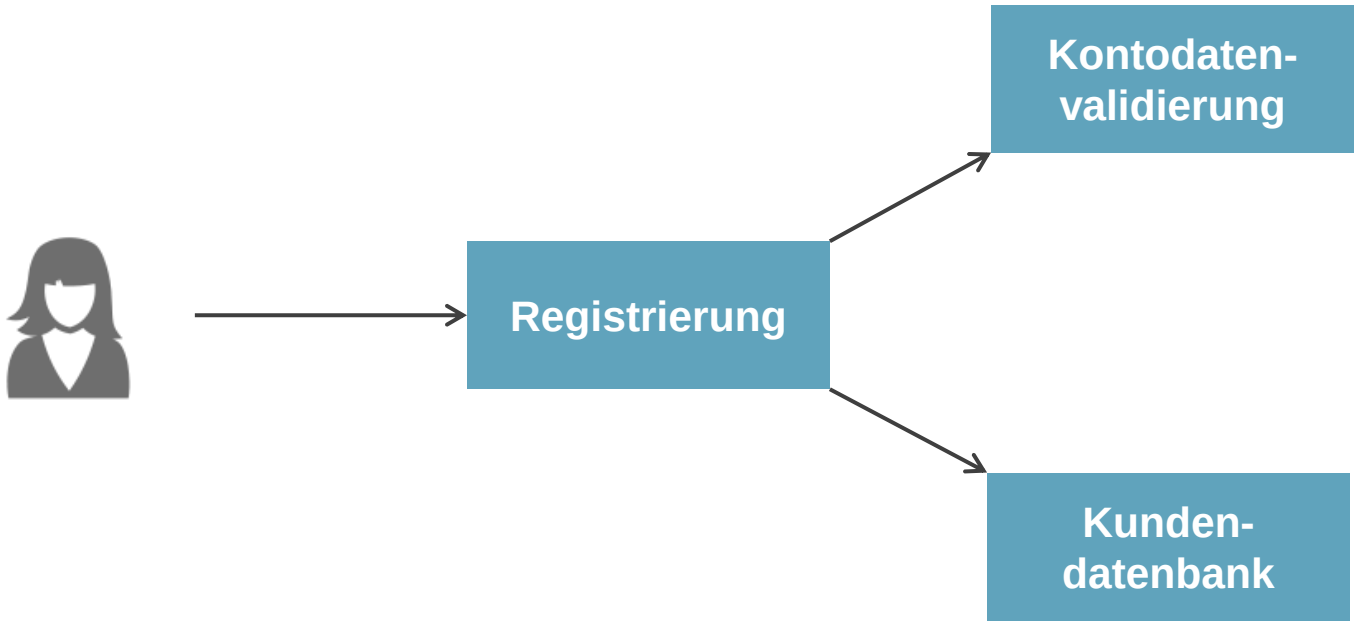
1. **Anwendungen mit externen Diensten**
2. **Einbau von Hystrix Schritt für Schritt**
3. **Anwendungsüberwachung mit Hystrix**
4. **Hystrix – für jede Anwendung geeignet?**
5. **Zusammenfassung**

Anwendungen mit externen Diensten

Idealisierte Anwendungen haben keine Abhängigkeiten



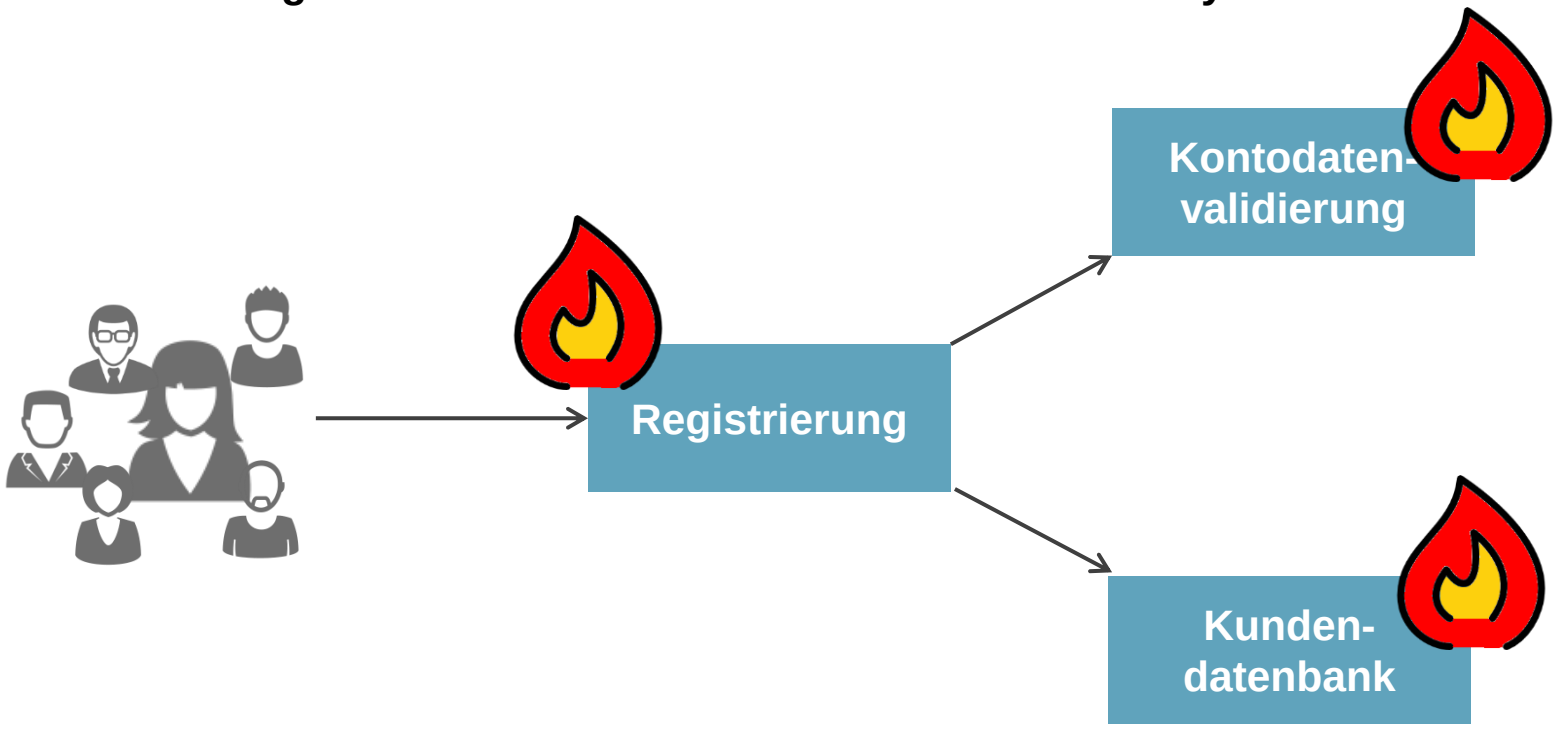
Echte Anwendungen haben Abhängigkeiten



Ein Dominoeffekt tritt auf, wenn eine Komponente ausfällt

- Ein Nutzer kann sich nur registrieren, wenn alle drei Komponenten funktionieren.
- (Performance-)Probleme können sich fortpflanzen und verstärken.

Domino: Eine langsame Kundendatenbank destabilisiert das System



- Eine langsame Kundendatenbank führt zur Überlastung der Kontovalidierung.
- Die aufgestauten Nutzer führen zu einer Überlastung der Registrierung.
- Das System kann nicht ohne Hilfe in einen stabilen Zustand gelangen.

Probleme in einer Komponente sollen sich nicht fortpflanzen

Möglichkeit 1: Asynchrone Verarbeitung

- Warteschlangen: Die Kunden werden nach der Registrierung asynchron in der Kundendatenbank angelegt.

Aber: Funktioniert nicht bei synchroner Rückgabe von Werten wie bei der Kontovalidierung.

Möglichkeit 2: Robuste synchrone Verarbeitung

- Beschränkung der Antwortzeit (Timeout)
- Sicherungen beim Abweichen vom Normbereich (Circuit Breaker)
- Abschottung der Komponenten untereinander (Bulkhead)

Technische Implementierung: Netflix Hystrix

AGENDA

1. Anwendungen mit externen Diensten
2. **Einbau von Hystrix Schritt für Schritt**
3. Anwendungsüberwachung mit Hystrix
4. Hystrix – für jede Anwendung geeignet?
5. Zusammenfassung

Netflix Hystrix: Kapselung von Schnittstellen

- Java-Bibliothek: kann beliebige Schnittstellen kapseln
- Implementiert Patterns für robuste Anbindung von Schnittstellen
- Open Source seit 2011
- Hystrix @ Github: https://github.com/Netflix/Hystrix
- Hystrix Tutorial: https://github.com/ahus1/hystrix-examples

Einbau von Hystrix Schritt für Schritt

Ausgangsbasis: Aufruf des externen Services ohne Absicherung

```
if (!IBANValidator.isValid(account)) {
    throw new ValidationException("invalid");
}
```

Einbau von Hystrix Schritt für Schritt

Schritt 1: Kapselung des Aufrufs als HystrixCommand

```
private static class IBANValidatorCommand extends
                                    HystrixCommand<Boolean> {
  private Account account;

  protected IBANValidatorCommand(Account account) {
     super(Setter.withGroupKey(HystrixCommandGroupKey.Factory
                                                  .asKey("iban")));
     this.account = account;
  }

  @Override
  protected Boolean run() throws Exception {
     return IBANValidator.isValid(account);
  }

}
```

Einbau von Hystrix Schritt für Schritt

Schritt 2: Anpassen des ursprünglichen Codes

```
if (!new IBANValidatorCommand(account).execute()) {
    throw new ValidationException("invalid");
}
```

IBANValidatorCommand wirft HystrixRuntimeExceptions.
Die Exceptions des gekapselten Aufrufs werden in HystrixRuntimeExceptions eingepackt.

Schritt 2b: Entpacken der ursprünglichen Exceptions (bei Bedarf)

```
try {
  if (!new IBANValidatorCommand(account).execute()) {
    throw new ValidationException("invalid");
  }
} catch (HystrixRuntimeException e) {
  if (e.getCause() instanceof MyException) {
    throw (MyException) e.getCause();
  }
  throw e;
}
```

Einbau von Hystrix Schritt für Schritt

Standardverhalten von Hystrix implementiert alle drei Patterns

- Zeitlimit von 1.000 ms für die Antwort (Timeout)
- Maximal 10 parallele Anfragen (Bulkhead)
- Abschaltung, wenn mehr als 50% der Aufrufe fehlschlagen (Circuit Breaker)
- Nach einer Abschaltung Wiederanlaufversuch alle 5 Sekunden (Circuit Breaker)

Die Standardwerte und die individuellen Werte auf Kommandoebene sind konfigurierbar.

Schritt 3: Hinzufügen von Fallbacks

```
private static class IBANValidatorCommand ... {

 /* ... */

 @Override
 protected Boolean getFallback() {
   return IBANFallback.isCheckDigitValid(account);
 }

}
```

Schritt 4: Konfiguration zur Laufzeit

- Hystrix wird standardmäßig über Archaius konfiguriert.
- Archaius kann die Werte aus einer Datenbank, einer Datei, etc. auslesen.
- Änderungen werden ohne Neustart aktiv.

(Mehr Informationen: https://github.com/Netflix/Hystrix/wiki/Configuration)

```
# Java Start-Parameter
-Darchaius.configurationSource.additionalUrls=file:///.../archaius.properties
-Darchaius.fixedDelayPollingScheduler.delayMills=1000
-Darchaius.fixedDelayPollingScheduler.initialDelayMills=1000
```

```
# archaius.properties (spezifische Werte für ein HystrixCommand)
hystrix.command.IBANValidatorCommand.execution.isolation.thread.timeoutInMilliseconds=1000
hystrix.command.IBANValidatorCommand.circuitBreaker.errorThresholdPercentage=50
hystrix.command.IBANValidatorCommand.circuitBreaker.requestVolumeThreshold=20
hystrix.command.IBANValidatorCommand.circuitBreaker.sleepWindowInMilliseconds=5000
```

Einbau von Hystrix Schritt für Schritt

Option: Verwendung von Futures

```
Future<Boolean> futureResult =
         new IBANValidatorCommand(account).queue();

/* ... do something in between ... */

// see if the call has completed in the meantime
if(futureResult.isDone()) {
  /* ... */
}

// retrieve result - wait if necessary
Boolean result = futureResult.get();
```

Einbau von Hystrix Schritt für Schritt

Option: Verwendung von Observables

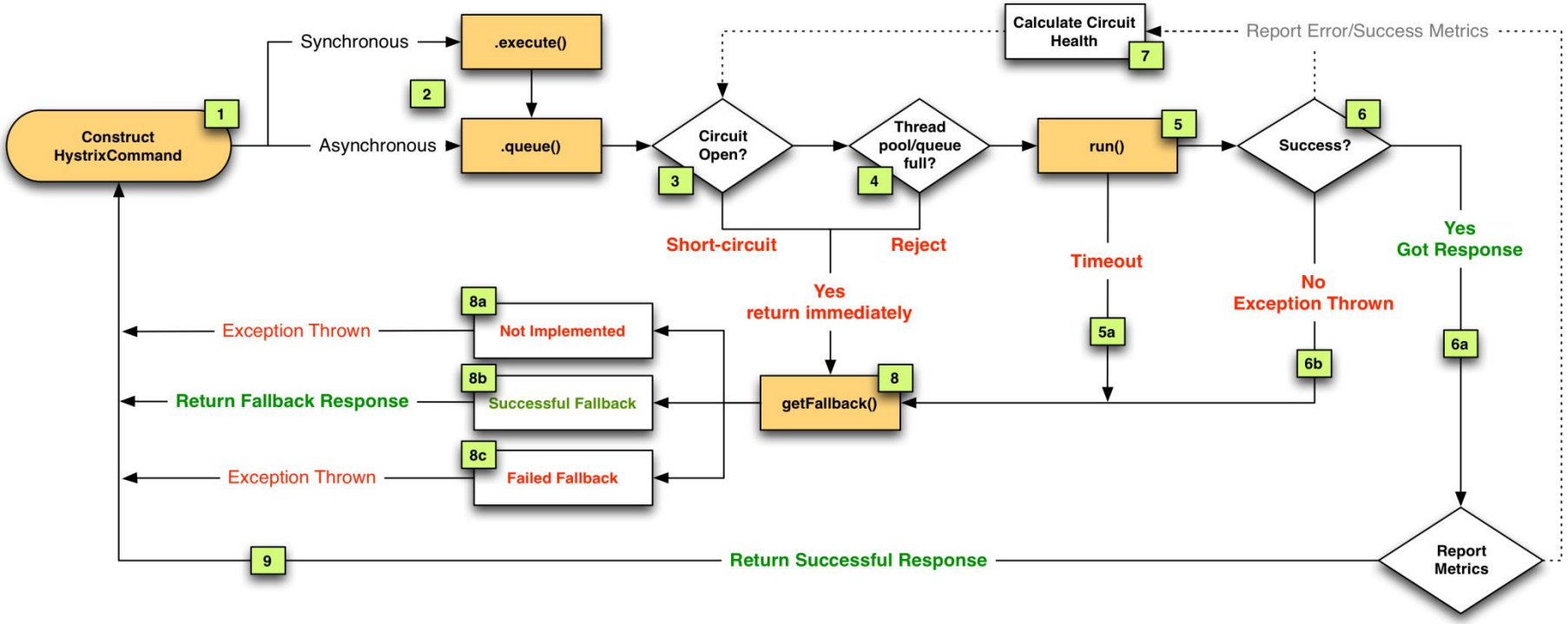
```
Observable<Boolean> result =
                new IBANValidatorCommand(account).observe();

result.subscribe(new Action1<Boolean>() {

    @Override
    public void call(Boolean b) {
         // do something once the response is ready
    }

});
```

Interner schematischer Ablauf eines HystrixCommands



Quelle: https://github.com/Netflix/Hystrix/wiki/How-it-Works

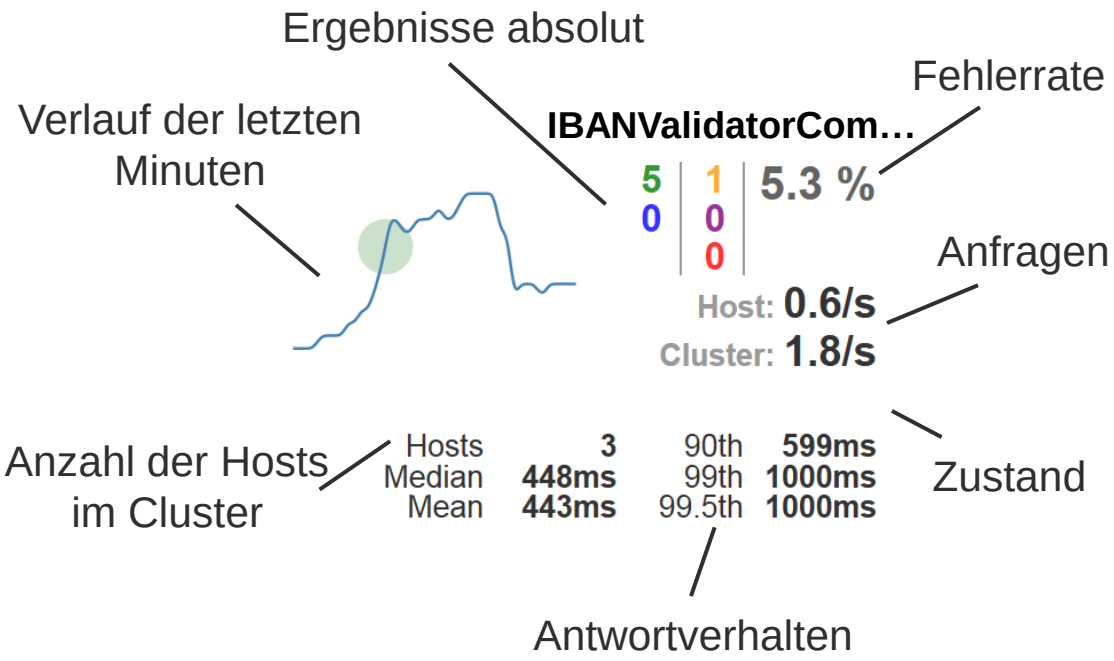
AGENDA

1. Anwendungen mit externen Diensten
2. Einbau von Hystrix Schritt für Schritt
3. **Anwendungsüberwachung mit Hystrix**
4. Hystrix – für jede Anwendung geeignet?
5. Zusammenfassung

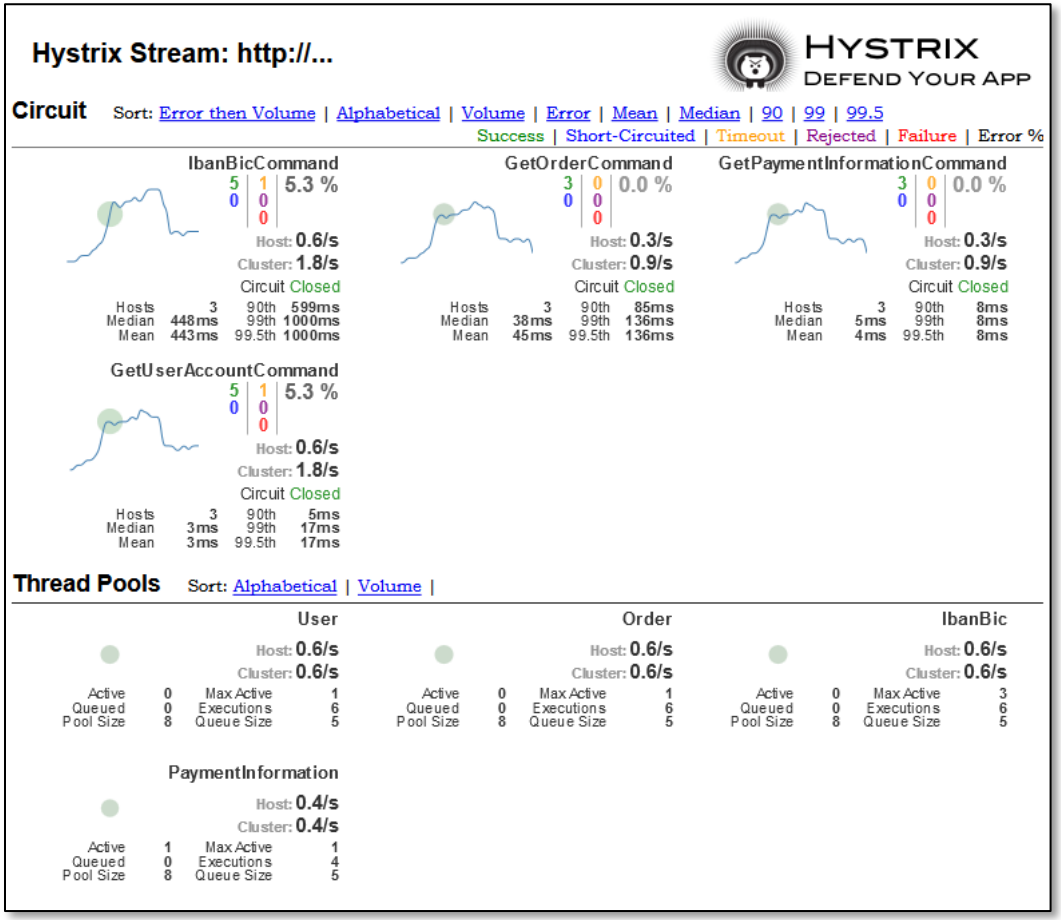
Langzeit- und Echtzeit-Überwachung

- Überwachung liefert mir wichtige Informationen:
  - Funktioniert die Anbindung wie gewünscht?
  - Hat sich der Normbereich verändert?
- Alle Parameter und Kennzahlen sind verfügbar per API oder Konnektoren für Yammer Metrics, Netflix Servo, JMX, …
- Proof of Concept mit Zabbix und Riemann im Tutorial
- Hystrix Dashboard ist Teil der Hystrix Distribution:
  Browser-Seite mit Commands und Thread Pools pro Anwendung im Cluster

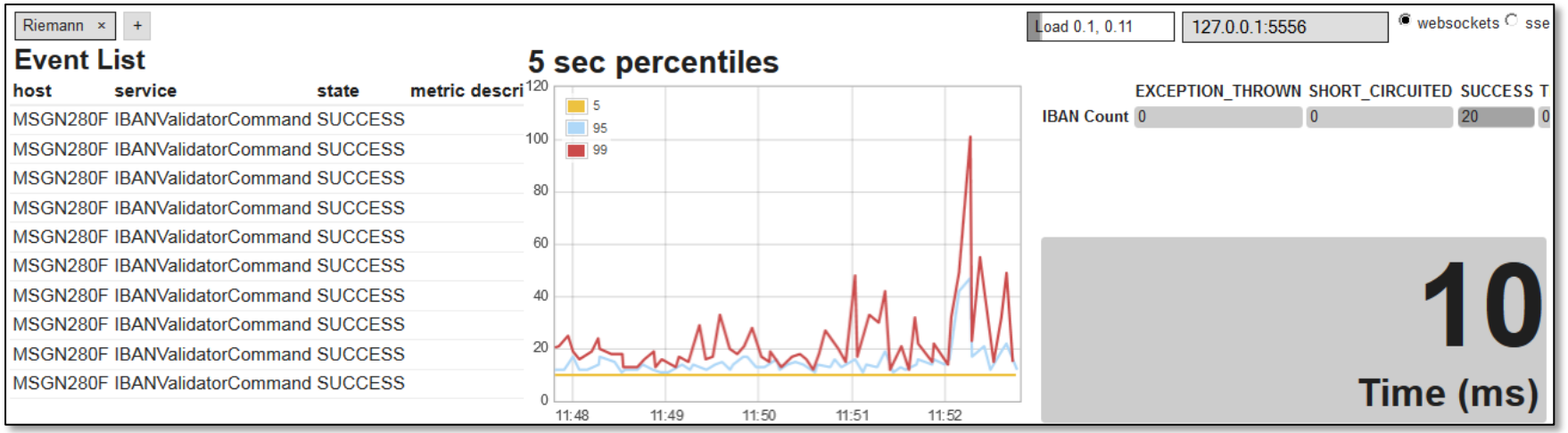
Echtzeitinformationen werden im Hystrix Dashboard angezeigt



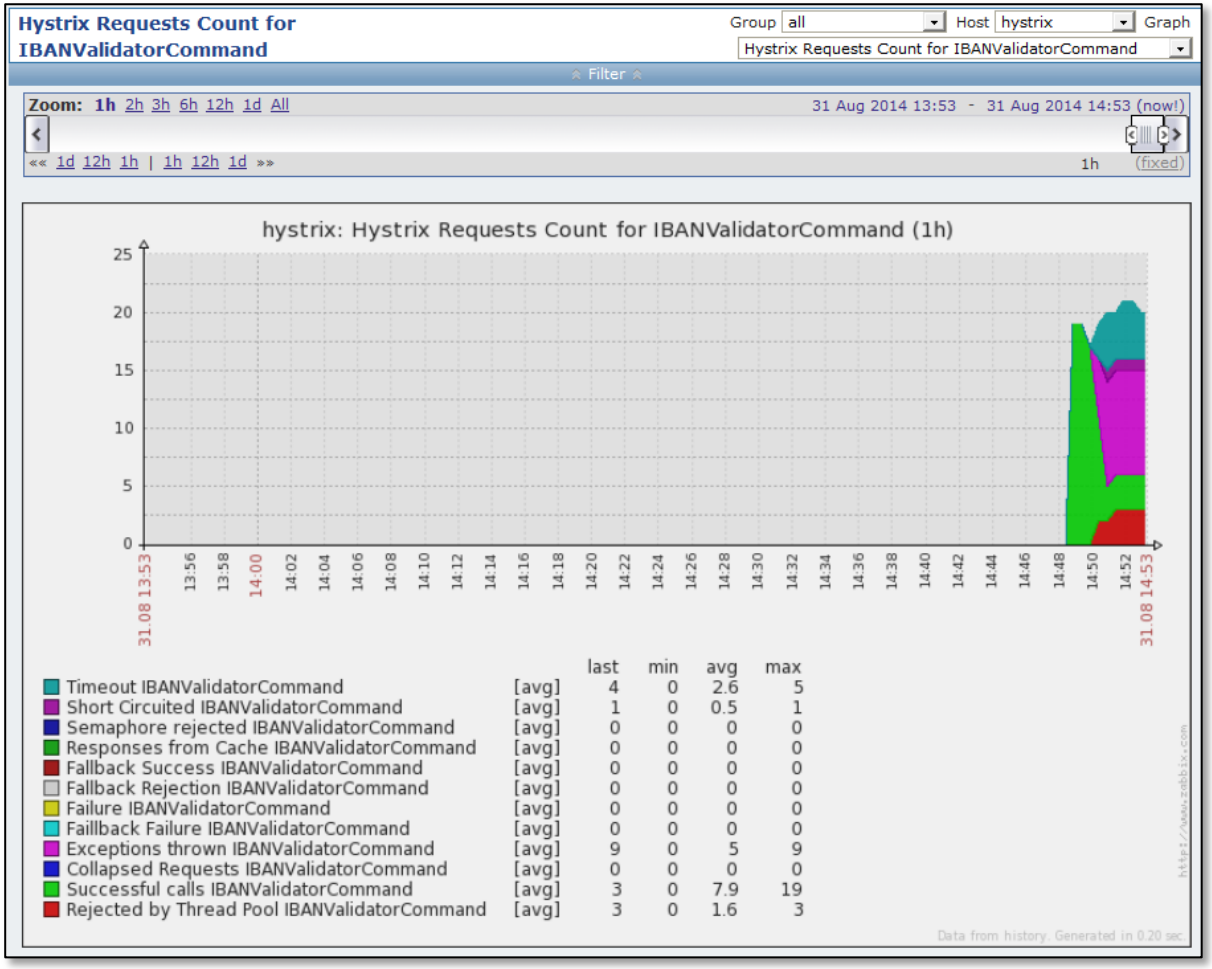
Echtzeitinformationen werden im Hystrix Dashboard angezeigt



Echtzeitinformationen werden im Riemann Dashboard angezeigt



Langzeitinformationen werden in Zabbix angezeigt



AGENDA

1. Anwendungen mit externen Diensten
2. Einbau von Hystrix Schritt für Schritt
3. Anwendungsüberwachung mit Hystrix
4. **Hystrix – für jede Anwendung geeignet?**
5. Zusammenfassung

Hystrix – für jede Anwendung geeignet?

Rezept für die Einführung von Hystrix

1. Bewusstsein schaffen: Komponenten können ausfallen, Fehler dürfen sich nicht fortpflanzen.
2. Vermessung des Normalzustands.
3. Identifikation der geeigneten Schnittstellen (synchron und idempotent).
4. Akzeptieren, dass zu langsame Antworten als Fehler gewertet werden.
5. Aufbau Monitoring **und** Einbau und Aktivierung von Hystrix in die Anwendung.

Hystrix – für jede Anwendung geeignet?

Alternative: Rezept II für die Einführung von Hystrix

1. Identifikation der geeigneten Schnittstellen (synchron und idempotent).
2. Einbau von Hystrix mit neutralisierter/deaktivierter Funktion.
3. Aufbau Monitoring und Vermessung des Normalzustands.
4. Akzeptieren, dass zu langsame Antworten als Fehler gewertet werden.
5. Hystrix scharfschalten.

Hystrix – für jede Anwendung geeignet?

Mögliche Herausforderungen rund um Hystrix

1. **Hystrix ist Open-Source-Software**
   Für Korrekturen und neue Funktionalität ist Mitarbeit am Code explizit gewünscht.

2. **Gebaut für Netflix-Infrastruktur**
   Durch die Nutzung bei Netflix bewährt. Eigene Anwendungsfälle sollten ausgiebig getestet werden.

3. **Entkopplung basiert auf Threads**
   Kein Problem in Apache Tomcat, ggf. in anderen Application Servern.
   Eingeschränkte Nutzung ohne Threads ist möglich.
   JEE7 JSR 236 (Concurrency Utilities for Java EE) noch nicht unterstützt.

4. **Exceptions des Service werden als HystrixRuntimeExceptions verpackt**
   Wenn der eigene Code die Exceptions des Service erwartet, so müssen die HystrixRuntimeExceptions entpackt werden.

AGENDA

1. Anwendungen mit externen Diensten
2. Einbau von Hystrix Schritt für Schritt
3. Anwendungsüberwachung mit Hystrix
4. Hystrix – für jede Anwendung geeignet?
5. Zusammenfassung

Zusammenfassung

Hystrix ermöglicht widerstandsfähige Anwendungen

- Mit Hystrix steht eine erprobte technische Lösung für bewährte Patterns zur Verfügung.
- Ein Umbau bestehender Anwendungen ist auch nachträglich möglich.
- APIs mit bestehenden Implementierungen für die Integration in ein bestehendes Monitoring existieren.

Literatur: Release It! – Design and Deploy Production-Ready Software (M. Nygard)

Hystrix @ Github: https://github.com/Netflix/Hystrix

Hystrix @ Heise Developer: http://heise.de/-2176465

Hystrix Examples: http://ahus1.github.io/hystrix-examples/

@ahus1de

Vielen Dank für Ihre Aufmerksamkeit

Alexander Schwartz

Geschäftsbereich Travel & Logistics
Principal IT Consultant

Telefon: +49 171 5625767
alexander.schwartz@msg-systems.com

www.msg-systems.com

.consulting .solutions .partnership

.msg