The Cool and the Cruel of MicroService

Mark Struberg,
RISE GmbH,
Apache Software Foundation,
INSO TU Wien

About me

- Mark Struberg
- 25 years in the industry
- Apache Software Foundation member
- struberg [at] apache.org
- RISE GmbH employee
- TU-Wien / INSO researcher
- Committer / PMC for Apache OpenWebBeans, MyFaces, TomEE, Maven, OpenJPA, BVal, Isis, DeltaSpike, JBoss Arquillian, ...
- Java JCP Expert Group member and spec lead
- MicroProfile Spec Author
- Twitter: @struberg

The Weapon of Choice

- **"If you have a hammer, every problem seems to be a nail"**
- **"Es gibt für jede Schraube den passenden Hammer!"**
- **"Use the right tool for the right job"**
- **Every design decision has pros and cons!**
  - There is no solution which perfcectly fits all your problems
  - Example: centralised vs de-centralised systems, App evolution in waves: HOST -> server/client PCs -> HTML webapps -> AJAX -> native phone apps -> microservices ->?
- **Know your weapons!**
- **Know your problems!**

MicroServices

If MicroServices are the answer

- **... what was the question or problem causing it?**
- **Monoliths**
  - extremely recursive inner dependencies
  - No clear separation of concerns
  - No clear inner design ("take whatever you need")
  - Not easy to scale
  - Hard to roll outs

What is a 'MicroService'?

- https://smartbear.com/learn/api-design/what-are-microservices/

Essentially, microservice architecture is a method of developing software applications as a suite of independently deployable, small, modular services
in which each service runs a unique process and communicates through a well-defined, lightweight mechanism
to serve a business goal.

How big is a MicroService

- **MicroServices are 'small, independent systems'**
  - but how big is 'small'?
  - What is the size of a typical MicroService
- **How big is a JavaEE server in contrast?**
  - Apache TomEE: 35MB
    - https://tomee.apache.org
  - Apache Meecrowave: 9MB
    - https://openwebbeans.apache.org/meecrowave

Independent Services

- **Are MicroServices really independent of each other?**
- **How about versioning?**
- **How to detect if a feature is unused?**
- **Independent Data**
  - A MicroService is self contained - including it's data
- **Independent Programming Language and Frameworks**
  - At least when using REST
  - Not that easy with messaging

Data Consistency and Transactions

- **XA requires fast connections**
  - does not really work over MicroServices
- **Eventual consistency**
- **Compensations**
- **Persistent Messaging**

Netflix does all that?

- **NO, of course not!**

Fallacies of Distributed Computing

- **As postulated by Peter L. Deutsch (Sun Microsystems):**
  - The network is reliable.
  - Latency is zero.
  - Bandwidth is infinite.
  - The network is secure.
  - Topology doesn't change.
  - There is one administrator.
  - Transport cost is zero.
  - The network is homogeneous.

Testing the ball of mud

- **Testing Distributed Applications is no easy task**
- **3 strategies**
  - Massive Integration Testing
  - Mocking the hell out of your project
  - Capture & Replay
  - Traffic Splitter (e.g. istio)

The takeaway?

Trading off Problems

- Problems with a Monolith
  .... can be solved by doing MicroServices
- Problems with MicroServices
  .... can be solved by doing a Monolith
- You just trade off problems
- Different sides of the same coin
- Actually it's not MicroService vs Monolith but Centralised vs Distributed

Useful MicroService tricks

- **Monoliths have the same problems when talking with other systems!**
  - No XA, need to store steps separately or use a state machine (process engine, status in the DB, Compensations, etc)
  - Circuit Breakers
  - Bulkheads
- **Separate high-volume/low consistency areas from important areas**
- **Split your whole problem in distinct parts with their own Database (Conway's Law)**
  - Those parts don't need to be 'micro' though!

Useful MicroService tricks

- **Monoliths have the same problems when talking with other systems!**
  - No XA, need to store steps separately or use a state machine (process engine, status in the DB, Compensations, etc)
  - Circuit Breakers
  - Bulkheads
  - Distributed Log Correlation
- **Separate high-volume/low consistency areas from important areas**
- **Split your whole problem in distinct parts with their own Database (Conway's Law)**
  - Those parts don't need to be 'micro' though!

Application Layering

- **Also works with Monoliths**

JavaEE vs SOA vs MicroService vs ...

- **Is this really a 'vs'?**
- **Or is is more like fitting parts?**

Questions?