3 typische Stolperfälle bei der Microservice-Integration und wie man sie verhindert

@felixlmueller
Who I am?

Felix Müller
REST, SOAP,
Cloud, Saas,
Microservices, SCS,
FaaS, Serverless,
...
...
Distributed systems
Distributed systems

Communication is complex

Asynchronous communication

Distributed Transactions
Communication is complex
Failure will happen. Accept it!

But keep it local! Be resilient.
Let’s start with a simple example
Live hacking

https://github.com/flowing/flowing-retail/blob/master/rest/java/payment/src/main/java/io/flowing/retail/payment/port/resthacks/PaymentRestHacksControllerV1.java
Let’s start with a simple example
Circuit Breaker

Photo by CITYEDV, available under Creative Commons CC0 1.0 license.
Live hacking

https://github.com/flowing/flowing-retail/blob/master/rest/java/payment/src/main/java/io/flowing/retail/payment/port/resthacks/PaymentRestHackControllerV2.java
Fail fast is important
Fail fast is important but not enough!
Photo by Tookapic, available under Creative Commons CC0 1.0 license.
Translation: "There was an error while sending your boarding pass"
Current situation
Current situation

User → Web-UI → Check-in → Output Mgmt → Barcode Generator
Current situation

User

Web-UI

Check-in

Output Mgmt

Barcode Generator

Current situation
Internal Server Error - Read

The server encountered an internal error or misconfiguration and was unable to complete your request.

Reference #3.1d079ccc.1519892932.9c55d68
Current situation – the bad part
Current situation – the bad part

User -> Web-UI

Check-in

Output Mgmt

Barcode Generator

Flash symbol indicates a problem or fault.
Current situation – the bad part

User

Stateful Retry

Web-UI

Check-in

Output Mgmt

Barcode Generator
Translation: It's your problem now.
We are having some technical difficulties and cannot present you your boarding pass right away.

But we do actively retry ourselves, so lean back, relax and we will send it on time.

...just made this up...
Possible situation – much better!
Possible situation – much better!
Possible situation – much better!

User → Web-UI → Check-in → Stateful Retry

Output Mgmt → Barcode Generator → The failure never leaves this scope!
Persist thing (Entity, Document, Actor, ...)

Typical concerns

DIY = effort, accidental complexity

Scheduling, Versioning, operating, visibility, scalability, ...

State machine or workflow engine

Complete, proprietary, heavyweight, slow, developer adverse
Workflow engines, state machines

It is relevant in modern architectures
Silicon valley has recognized

Workflow engines, state machines
There are lightweight open source options for workflow engines, state machines.
Workflow engines, state machines

also at scale

AWS Step Functions

Camunda

jBPM

Conductor

Activiti

NetfliX OSS

UBER

CADENCE

zeebe by Camunda
Workflow engines, state machines for today's demo
Live hacking


Now you have a state machine!
Client
has to implement
Retry

Service Provider
has to implement
Idempotence
It is a business problem anyway!

We are processing your payment. Do not leave this page.
And for god sake – do not reload!
We are currently processing your request. Don’t worry, it will happen safely – even if you lose connection. Feel free to reload this page any time!

It is a business problem anyway!
Distributed systems introduce complexity you have to tackle!
Distributed systems
It is impossible to differentiate certain failure scenarios.

Independent of communication style!
Distributed systems introduce complexity you have to tackle!
Distributed systems introduce complexity you have to tackle!

Do it reliably
Workflows live within service boundaries
Workflows live within service boundaries

User → Web-UI → Check-in → Output Mgmt, Barcode Generator

- Generate 3D Barcode
- Send boarding pass
  - Including stateful retry
Manifold architecture options

https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91
Manifold architecture options

https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91
Manifold architecture options

https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91
Manifold architecture options

https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91
A synchronous response is possible in the happy case, otherwise it is switched to asynchronous processing.
„The customer wants a synchronous response“
Synchronous communication is the crystal meth of distributed programming.

Todd Montgomery and Martin Thompson in “How did we end up here” at GoTo Chicago 2015
Asynchronous communication
Asynchronous communication

You need to monitor timeouts
Remember...

User → Web-UI → Check-in → Output Mgmt → Barcode Generator

The failure never leaves this scope!
Workflow...

- Send generate barcode command
- Wait for barcode
- Resend generate barcode command
- Send boarding pass
- Wait for mail confirmation
- Resend boarding pass

Easy to handle time
Workflow...
BPMN
Business Process Model and Notation
ISO Standard
Executable and mature

Widespread

Easy to understand*

*(for Biz, Dev, and ops)
Biz Dev Ops

- Understand and discuss business processes
- Leverage state machine & workflow engine
- Operate with visibility and context
- Evaluate optimizations in-sync with implementation
- Living documentation
- Visibility in testing

improve communication
Proper operations

Visibility + Context
Compare to e.g. Step Functions
Compare to e.g. Step Functions
Imagine more complex stuff
Client

has to implement

Timeout, Retry

Service Provider

has to implement

Idempotence
Who uses a message bus?
Who has no problems operating a message bus?

Dead messages | No context | Inaccessible payload | Hard to redeliver | Home-grown message hospitals | ...
Manifold architecture options

https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91
Manifold architecture options

https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91
Manifold architecture options

https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91
Distributed Transactions
Life beyond Distributed Transactions: an Apostate’s Opinion
Position Paper

Pat Helland
Amazon.Com
705 Fifth Ave South
Seattle, WA 98104
USA
PHelland@Amazon.com

ABSTRACT

The positions expressed in this paper are personal opinions and do not in any way reflect the positions of my employer Amazon.com.

Many decades of work have been invested in the area of distributed transactions using protocols such as 2PC. Pat Helland argues that the need for these protocols has been over exaggerated. Instead, applications are built using different techniques which do not provide the same transactional guarantees but still meet the needs of their businesses.
Distributed transactions using compensation * aka Saga pattern

Compensation
Eventual consistency

Temporarily inconsistent state

But only temporarily!
Live hacking

Client

has to implement

Timeout, Retry, Compensation

Service Provider

has to offer

Compensation

has to implement

Idempotency
Client

has to implement
Timeout, Retry, Compensation

Don't forget about state

Service Provider

has to offer
Compensation, Idempotency

Communication is complex
Challenges of asynchronicity
Distributed Transactions
Be aware of complexity of distributed systems

Know strategies and tools to handle it

e.g. Circuit breaker (*Hystrix*)
Workflow engine for stateful retry, waiting, timeout and compensation (*Camunda*)
Thank you

&

Come and see me @Camunda booth

felix.mueller@camunda.com
@felixlmueller
https://github.com/felix-mueller