Concurrent Programming and the Microservice Architecture Style

Maximilian Irro

Seminar für DiplomandInnen
4.12.2017

Timeline of the Microservice concept

- 2012 (?) first mention of the term “Microservice”
- 2014 Whitepaper by Fowler and Lewis
- 2015 Academia gets interested
- 2016, 2017 Explosion of publications

Definitions

Definitions

"A **microservice** (MS) is a cohesive, independent process interacting via messages"

Definitions

"A **microservice** (MS) is a cohesive, independent process interacting via messages"

"A **microservice architecture** (MSA) is a distributed application where all its components are microservices"

[Dragoni, Nicola, et al. "Microservices: yesterday, today, and tomorrow." Present and Ulterior Software Engineering. Springer, Cham, 2017. 195-216.]

Characteristics of Microservices (1)

- Services as building blocks
- Focus on business capability
- Multiple executable artifacts
- Communication via messages
- Lightweight communication (e.g. REST, “dumb” message broker)
- No-shared state, only accessible via API
- High cohesion, loose coupling
- etc. . .

Characteristics of Microservices (2)

Microservices

- Services are building blocks
- Identity (PID, network-address), state, behavior
- Communication via messages
- Interfaces, design by contract
- Encapsulation, data-hiding
- High cohesion, loose coupling
- ...

Characteristics of Microservices (2)

Microservices

- Services are building blocks
- Identity (PID, network-address), state, behavior
- Communication via messages
- Interfaces, design by contract
- Encapsulation, data-hiding
- High cohesion, loose coupling
- ...

Objects

- Services are building blocks
- Identity (memory-address), state, behavior
- Communication via messages
- Interfaces, design by contract
- Encapsulation, data-hiding
- High cohesion, loose coupling
- ...

Characteristics of Microservices (cont.)

- MSA consists of multiple processes
- Message passing is inter-process communication
- Application is a distributed system

Are MS just Distributed Objects?

- Distributed Objects based on idea of
  - Putting objects into processes
  - Transparent in-process/remote communication

Are MS just Distributed Objects?

- Distributed Objects based on idea of
  - Putting objects into processes
  - Transparent in-process/remote communication
- MS make remote method invocation (e.g. REST call) explicit
  - Different API granularity

[Waldo, Jim, et al. "A Note on Distributed Computing." International Workshop on Mobile Object Systems. Springer, Berlin, Heidelberg, 1996.]

Concurrency and Microservices

- Threads vs Processes

Concurrency and Microservices

- Threads vs Processes
- Concurrency Models often look similar to Distributed Systems Architecture

Concurrency and Microservices

- Threads vs Processes
- Concurrency Models often look similar to Distributed Systems Architecture
- "[...] expect to see a return of distributed programming with increasingly fine-grained distributed systems that will make systems look more and more like classic concurrent system"

[Jan Stenberg. „Concurrent and Distributed Programming in the Future". 2017. https://www.infoq.com/news/2017/03/distributed-programming-qcon]

Special Interest Aspects of MSA

- Geographically distributed code
- Explicit parallelization (via deployment)
- Fault tolerance, resilience, self-healing
- Scalability
- Independent replacement and upgradability
- Decentralized data management (no shared memory)
- Polyglot programming, polyglot persistence

Roadmap

- Conceptual similarities between MS and OO, SOA, etc.
- Identify concurrency model(s) resembling MSA
- Practically demonstrate and compare with prototypical scenario
- Argue if “sufficiently concurrent” programming languages need MSA?

Actors (Akka implementation)

- Objects encapsulate state and behavior
- (Asynchronous) message passing, immutable messages
- Isolated mutable state, no global state
- Distributed by default
- Fault tolerance (supervision hierarchy)