Carnegie Mellon

DevOps: Engineering for Deployment and Operations

Microservice Architecture 1

Carnegie Mellon University

Overview

- Definition
- Microservices and DevOps
- Quality attributes

Carnegie Mellon University

Definition

- A microservice architecture is
  - A collection of independently deployable processes
  - Packaged as services
  - Communicating only via messages

Carnegie Mellon University

~2002 Amazon instituted the following design rules - 1

- All teams will henceforth expose their data and functionality through service interfaces.
- Teams must communicate with each other through these interfaces.
- There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.

Carnegie Mellon University

Amazon design rules - 2

- It doesn't matter what technology they[services] use.
- All service interfaces, without exception, must be designed from the ground up to be externalizable.
- Amazon is providing the specifications for the "Microservice Architecture".

Carnegie Mellon University

In Addition

- Amazon has a "two pizza" rule.
- No team should be larger than can be fed with two pizzas (~7 members).
- Each (micro) service is the responsibility of one team
- This means that microservices are small and intra team bandwidth is high

- Large systems are made up of many microservices.
- There may be as many as 140 in a typical Amazon page.

Carnegie Mellon University

Services can have multiple instances

- The elasticity of the cloud will adjust the number of instances of each service to reflect the workload.
- Requests are routed through a load balancer for each service
- This leads to
  - Lots of load balancers
  - Overhead for each request.

Carnegie Mellon University

Digression into Service Oriented Architecture (SOA)

- The definition of microservice architecture sounds a lot like SOA.
- What is the difference?
- Amazon did not use the term “microservice architecture” when they introduced their rules. They said “this is SOA done right”

Carnegie Mellon University

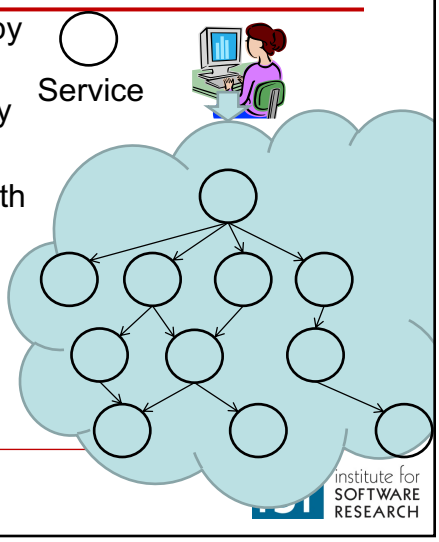
SOA typically has but microservice architecture does not

- Enterprise service bus
- Elaborate protocols for sending messages to services (WDSL*)
- Each service may be under the control of different organization
- Brokers
- etc

Carnegie Mellon University

Micro service architecture

- Each user request is satisfied by some sequence of services.
- Most services are not externally available.
- Each service communicates with other services through service interfaces.
- Service depth may
  - Shallow (large fan out)
  - Deep (small fan out, more dependent services)



Carnegie Mellon University

Microservice architecture supports DevOps processes

- Reduces need for coordination
- Reduces pipeline errors
- Supports continuous deployment

Carnegie Mellon University

How does microservice architecture reduce requirements for coordination?

- Coordination decisions can be made
  - incrementally as system evolves or
  - be built into the architecture.
- Microservice architecture builds most coordination decisions into architecture
- Consequently they only need to be made once for a system, not once per release.

Carnegie Mellon University

Seven Decision Categories

- Architectures can be categorized by means of seven categories
  1. Allocation of functionality
  2. Coordination model
  3. Data model
  4. Management of resources
  5. Mapping among architectural elements
  6. Binding time decisions
  7. Technology choices

Carnegie Mellon University

Design decisions with microservicess

- Microservice architecture either specifies or delegates to the development team five out of the seven categories of design decisions.

  1. Allocation of responsibilities.
  2. ~~Coordination model~~.
  3. Data model.
  4. ~~Management of resources.~~
  5. ~~Mapping among architectural elements.~~
  6. ~~Binding time decisions.~~
  7. ~~Choice of technology~~

Carnegie Mellon University

How do microservices reduce pipeline errors?

- Many integration errors are caused by incompatibilities.
- Each team makes its own technology choices for their services. This reduces errors during integration
  - No problems with inconsistent versions of dependent software
  - No problems with language choices

Carnegie Mellon University

How do microservices support continuous deployment?

- Each team can deploy their service when it is ready without consultation with other teams.
  - Since services are small, this can happen frequently
  - New feature implementation in one service can be deployed without waiting for that feature to be implemented in other services.
  - Requires architectural support – discussed in lectures on deployment

Carnegie Mellon University

Quality attributes

- Availability
- Modifiability
- Performance
- Reusability

Carnegie Mellon University

Availability

- Availability is achieved using
  - Multiple instances
  - Timeouts to recognize failure
  - Stateless instances
- If instances are stateless, when an instance failure occurs a new instance can be created and placed into service

Carnegie Mellon University

Modifiability

- Modifiability depends on the design of the various microservices.
- Low coupling and high cohesion are design principles to achieve modifiability.
- Also, a catalog of microservices must be maintained since there will be many microservices and developers must be able to understand how functionality is distributed among them.

Carnegie Mellon University

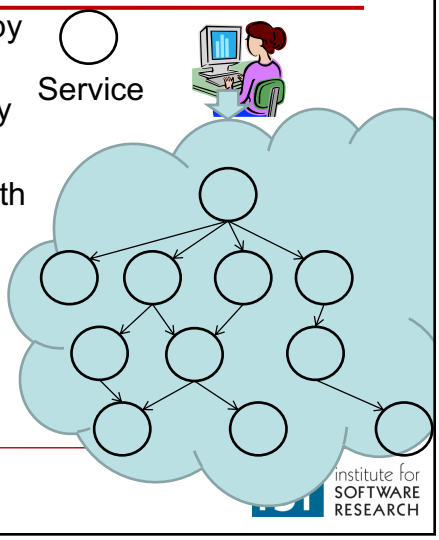
Performance

- Microservices are based on message passing.
- Network communication takes time.
- Important to measure and monitor microservices to determine where time is spent so that performance objectives can be met.
- Pods are means for improving performance

Carnegie Mellon University

Reusability

- Each user request is satisfied by some sequence of services.
- Most services are not externally available.
- Each service communicates with other services through service interfaces.
- Service depth may
  - Shallow (large fan out)
  - Deep (small fan out, more dependent services)



institute for SOFTWARE RESEARCH

Carnegie Mellon University

Distinguish between coarse grain and fine grain reuse

- Large microservices (coarse grained) are built into architecture and reuse is achieved by using these microservices
- Small microservices (fine grained) can be designed for reuse (deep service hierarchy) or reuse can be ignored at this level (shallow service hierarchy)

institute for SOFTWARE RESEARCH

Carnegie Mellon University

Trade off

- Shallow hierarchy
  - higher performance since fewer messages,
  - low reuse since each service is developed by independent team with little coordination
- Deep hierarchy
  - Lower performance since more messages
  - Higher reuse since services can be designed for multiple different clients.

Carnegie Mellon University

Summary

- Microservice architecture consists of collection of independently deployable services
- Microservices supports devops processes by reducing need for communication among teams and making technology choices independent
- Microservice architecture favors modifiability over performance.
- Reuse can be achieved, if desired, by having deep service hierarchy.