Supporting Microservice Evolution

Adalberto R. Sampaio Jr, Harshavardhan Kadiyala, Bo Hu, John Steinbacher, Tony Erwin, Nelson Rosa, Ivan Beschastnikh, Julia Rubin
From monoliths to microservices

- Microservices (µServices)
  - Fast and easy to deploy
  - Can be scaled independently
  - Multilingual and multi-technology
  - Loose dependencies (REST)

https://martinfowler.com/articles/microservices.html
From monoliths to microservices

- Microservices ($\mu$Services)
  - Fast and easy to deploy
  - Can be scaled independently
  - Multilingual and multi-technology
  - Loose dependencies (REST)

Leading to Constant evolution
Evolution-related challenges

- Upgrades break inter-services compatibility
  - Different teams work on different services
- Tracking down failure root causes complicated
  - Many moving parts that keep changing
- Deployment configuration must also evolve
  - Changes to performance/dependencies impact config
Existing tool support

- **Academia**: app-bisect [1], GRU [2], Gremlin [3], Formal methods [4]

**Wanted**: program analysis to support change

[4] Panda et al., *Verification in the Age of Microservices*, HotOS 2017
Existing tool support

- **Academia**: app-bisect [1], GRU [2]

Idea: model $\mu$App over time

Wanted: program analysis to support change

[8] https://www.influxdata.com/
1. Construct a model of the \( \mu \text{App} \) and its configuration

2. Update model as \( \mu \text{App} \) or configuration change

3. Analyze sequence of models (past, present, future)

4. Use the analysis to support developers
Proposed evolutionary (meta) model
Example model instance

- **Application**
  - ToDo

- **ApplicationVersion**
  - version 1
  - IBM Bluemix
  - version 2

- **Location**
  - EastCoast: vm1
  - WestCoast: vm2

- **Host**
  - vm1
  - vm2

- **ServiceReplica**
  - Frontend.2.A
  - Frontend.1.A
  - Frontend.2.B

- **ServiceVersion**
  - frontend.1
  - frontend.2

**Supporting microservice evolution**
Model analysis

- Sequence of models over time allows for rich analyses
  - **Retrospective**: Study inter-services messages to recommend service refactorings
  - **Prospective**: Explore and instantiate new deployment configurations to optimize resources usage
Ongoing/future work

- **Assessing developer needs**: which tasks are the most pressing?
- **Model representation**: many modeling formalisms, which one is the best for the task?
- **Defining analysis**: Build on existing model analysis work
- **Extensibility**: Can we allow the model to change? (Cannot foresee future information we may need to integrate)
- **Social factors**: interplay between technical and social dependencies
Microservices require new approaches

- Loose coupling
- Rapid deployment
- Multi-lingual
- Multiple teams

Vision: Generate an evolutionary model from dynamic observations (logs) for retrospective and prospective analyses

Find us at tomorrow’s poster session!

https://github.com/adalrsjr1/k8s-monitoring