Improving Service Availability with Rule-Based Adaptation

SSP 2018, Hildesheim

Marc Adolf Reiner Jung & Lars Blümke

8th November 2018
Cloud application rely on adaptation

- Up- and Down-Scaling
  - (De-)Replication
  - Migration
- Reconfiguration

→ Potential information loss during scaling and reconfiguration
Example Scenarios
Example Scenarios

deploy Cart B, configure Database

System Border

Frontend
10.10.1.2

Cart A
10.10.1.3

Cart B
10.10.1.7

Database
10.10.1.4

EasyPay
134.2.5.125
Example Scenarios

System Border

Frontend 10.10.1.2
Cart A 10.10.1.3
Cart B 10.10.1.7
Database 10.10.1.4
EasyPay 134.2.5.125

transfer state
Example Scenarios

System Border

- Frontend
  - 10.10.1.2
- Cart A
  - 10.10.1.3
- Cart B
  - 10.10.1.7
- Database
  - 10.10.1.4
- EasyPay
  - 134.2.5.125

connect Cart B, update state
Example Scenarios

System Border

disconnect Cart A

Frontend 10.10.1.2
Cart A 10.10.1.3
Cart B 10.10.1.7
Database 10.10.1.4
EasyPay 134.2.5.125
Example Scenarios

System Border

Frontend
10.10.1.2

Cart A
10.10.1.3

Cart B
10.10.1.7

Database
10.10.1.4

EasyPay
134.25.125

Reiner Jung 3 / 10
Example Scenarios

System Border

```
undeploy Cart A
EasyPay 134.2.5.125
Cart B 10.10.1.7
Frontend 10.10.1.2
Database 10.10.1.4
```

Reiner Jung 3 / 10
Example Scenarios

System Border

Frontend
10.10.1.2

Cart B
10.10.1.7

Database
10.10.1.4

EasyPay
134.2.5.125

SecuPay
104.9.4.22

configure Cart B to use SecuPay
Example Scenarios

System Border

- Frontend
  - 10.10.1.2

- Cart B
  - 10.10.1.7

- Database
  - 10.10.1.4

- EasyPay
  - 134.2.5.125

- SecuPay
  - 104.9.4.22

Configure Cart B to no longer use EasyPay
Solution

Divide adaptions actions in smaller execution actions
Application
Candidate
Architecture
Selection
Adaptation
Planning
HTTP
Planning
<<service>><<service>>
PAM
CAM
Monitoring
Analysis
Candidate
Architecture
Selection
Planning
Execution
<<service>>
HTTP
CAM
Execute Adaptations
Adaptation Planning

Compute composed adaptation actions

- Difference
  PAM vs. CAM

- Consider data dependencies

- Consider state

\[ 4! = 24 \text{ options, only two safe} \]

Reiner Jung
Generating atomic actions

Migration of B to B'

1. deploy B'
2. start state sync
3. connect B' -> C'
4. connect A -> B'
5. disallow new sessions from A -> B
6. wait for session completion
7. complete sync
8. disconnect A -> B
9. disconnect B -> C
10. undeploy B

Reiner Jung
Generating atomic actions

Migration of B to B'

- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B

C

A

B

B'

C'

Reiner Jung
Generating atomic actions

Migration of B to B'
  deploy B'
  start state sync
  connect B' -> C'
  connect A -> B'
  disallow new sessions from A -> B
  wait for session completion
  complete sync
  disconnect A -> B
  disconnect B -> C
  undeploy B
  start state sync
Generating atomic actions

Migration of B to B'
  deploy B'
  start state sync
  connect B' -> C'
  connect A -> B'
  disallow new sessions from A -> B
  wait for session completion
  complete sync
  disconnect A -> B
  disconnect B -> C
  undeploy B
  connect B' -> C'
  connect A -> B'
Generating atomic actions

Migration of B to B'
- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B
Generating atomic actions

Migration of B to B'

- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B
- complete sync
Generating atomic actions

Migration of B to B'

- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B
Generating atomic actions

Migration of B to B'  
  deploy B'  
  start state sync  
  connect B' -> C'  
  connect A -> B'  
  disallow new sessions from A -> B  
  wait for session completion  
  complete sync  
  disconnect A -> B  
  disconnect B -> C  
  undeploy B
Execution

Service

- Map atomic action to cloud API calls
- Execute actions without violating data constraints
  - Currently sequential execution
  - Future concurrent execution

Mapping

- Depends on cloud API
- Depends on service features
  - State transfer
  - Availability checks
Conclusion

Summary
- Introduced iObserve services for MAPE-K
  - Candidate Selection
  - Planning
  - Execution
- Rule-based adaption with Drools

Code
https://github.com/research-iobserve/iobserve-analysis

Outlook
- Concurrent execution
- Additional atomic actions
- More executors supporting different technologies
- Use of Kieker probes to check availability of services