Programming Abstractions for Composing Serverless Applications

Motivation

“Why is there no cloud button?”
UC Berkeley Professor [1]

“[…] but if I can move one layer up where I’m just writing business logic and the code gets split up appropriately, that’s real magic.”
Serverless is eating the stack [2]

“We don’t yet have the Rails of serverless”
Comment on Serverless is eating the stack [2]

→ We need programming model abstractions to build and compose larger Serverless applications.

Background

What is Serverless?
A cloud-native platform [3]
• for short-running, stateless computation
• end even-driven applications
• which scales up and down instantly and automatically
• and charges for actual usage at millisecond granularity

Why Serverless?

Research Question
How can we map (existing) single-machine code into applications composed of scalable cloud functions?

Related Work

[1] PyWren: run Python code in AWS Lambda
[5] Apache OpenWhisk Composer: new programming model for composing cloud functions
[6] AWS Step Functions: serverless orchestration for modern applications as visual state machine

Serverless AppCode (SAC) Prototype

Codebase

Split application into small deployment units

Challenges

• Notion of side-effects
• Integration of 3rd party services

Future Work

• Extend prototype and test with use cases
• Optimize deployment at runtime based on live monitoring data

References

[1] Occupy the cloud: distributed computing for the 99%, E. Jonas et al., Symposium on Cloud Computing (SoCC), 2017
[2] Serverless is eating the stack. URL: https://read.acloud.guru/serverless-is-eating-the-stack-and-people-are-freaking-out-and-they-should-be-431a9e0db482

Recent Publications

• Estimating Cloud Application Performance Based on Micro-Benchmark Profiling, IEEE CLOUD’18
• Performance testing in the cloud. How bad is it really?, preprint, 2nd
• A Cloud Benchmark Suite Combining Micro and Applications Benchmarks, QUDOS’18@ACM/SPEC ICPE

https://github.com/sealuzh/cloud-workbench