Josef Spillner <josef.spillner@zhaw.ch>
Service Prototyping Lab (blog.zhaw.ch/icclab)

Dec 21, 2017 | Vienna Software Seminar
Miniaturisation

meetup.com/FloridaGardeners

japanese.alibaba.com

TINY
ToCS Vol.4

ICCLAB SPLAB
Prospects of Serverless Applications

Industry: DEPOT meta-model
- Deployment model → control plane APIs, gateways, limits
- Execution model → isolation, state, memory/time limits
- Programming model → function signatures, implementations
- Orchestration model → Step Functions, Composer...
- Tariff model → pay-per-call, -per-load, ...

Academia:
catching up...

Example → ...
Our Function-as-a-Service Research

**Academia claims**
- 1st control plane analysis
- 1st HPC/SC analysis
- 1st function in production at uni
- 1st tutorial at conference

**Global claims**
- 1st code transformation tools
- 1st function marketplace
- 1st multi-provider migration

**2nd runtime**
(after OpenLambda †)
Function-as-a-Service Delivery

But: What about software service delivery?
• sensu stricto: serving the clients
  • according to message exchange patterns
• sensu lato: entire end-to-end pipeline from idea to dev to ops to usage
Function-as-a-Service Delivery

Mostly solved problems
• Deployment via SLFW, Composeless or provider-specific tools (awscli, wsk, gcloud, ...)
• Aggregate-monitored execution

Open issues
• Real-time insight
• Debugging support
  • Google Stackdriver? AWS X-Ray? RLY?
• Automated code transformation and fitting

Proposal →
“DevOps-style Tracing, Profiling and Autotuning of Cloud Functions“
Tracing

Using Snafu’s python3-tracing executor *

* contributed by L. Fernández-Garcés & Bernard Jollans @ KTH
Profiling

Application topology and behaviour
• precise, via execution traces
• heuristic, via e.g. Peddycord's algorithm

Insights through visualisation
• flame graphs (B. Gregg USENIX '13/'17)
Autotuning

Computers are dumb but fast + programmable!

Rules (disclaimer: not thought through)
• timeout reached → redeploy with different decomposition granularity, use
  worm functions (state handover), [when applicable]
  redeploy with more memory
• out of memory → implement map-reduce schemes, recluster local private
  functions, redeploy with more memory
• data latency issues → use cache, narrow gap (edge, copy)
• dependency service unavailable → notify

Automation + constructive developer notification
DevOps Perspective

- Application engineer
- Function
- Deployed Function
- Function Instance
- Application
- Application user

- Insight
- Visualisation
- Debugging/Profiling Tools
- Feedback
- Autotuning
- Improved experience
DevOps Perspective

- Triggers
- Function metrics: invocation count + time, throttling, faults, variables
- M Monitor
- E Execute
- K K-rules
- A Analyse
- P Plan

- Stacktraces
- Profiles
- Anomalies over time

- Recombination of function clusters
- Memory reconfiguration

- Hot deployment
- Canary testing
- Step/Breakpoint execution

- Faas Debugging, Profiling and Autotuning Catalogue
- Management functions

Triggers
Rapid Service Prototyping

1) Programming
   using FaaSification (decomposition): shallow, medium, deep

2) Provisioning
   through commercial clouds
   big 4
   through community clouds
   GuiFi
   through ad-hoc networks (device, meshes)

3) Autotuning

4) Service Delivery

5) ...

6) Profit!
Closing

Cloud functions are
- the sincerest form of microservices ("stateless nanoservices")
- great for education, link to programming
- practically free for small to medium usage (but you pay for state)

Negatives
- manual programming and debugging still tedious at scale
- rapid service prototyping still cumbersome

Upcoming Events:

Feb 2018: Serverless @ Swiss Python Summit
May 2018: Serverless @ DevOpsDays Zurich-Winterthur (CfP still open)
Dec 2018: Serverless symposium @ 11th IEEE/ACM UCC/BDCAT Zurich

https://blog.zhaw.ch/icclab/tag/faas/