SCAD P01 – Cloud Function Behaviour

This is a two-week lab, i.e. SW 1/2, submission before SW 3 lecture.

1. Preparation

1.1 Public FaaS + BaaS provider

In this lab, a commercial FaaS service (or a privately operated FaaS engine) is used to explore how cloud functions behave. Log in to at least one FaaS provider (Cloudflare, Google, Azure, ...) with the credentials received from your lecturer or your own private accounts. Complain if the login does not work.

1.2 Local development tooling

Install any necessary local client (e.g. `wrangler/aws/gcloud/wsk/ibmcloud fn`) from the provider website; some links may only appear after login. Make sure to also have a local development environment for a programming language (recommended: scripting language) of your choice set up.

1.3 Team building

Think about who you want to team up. The standard team size in HS 2021 is four to five. If a lonely student is left, he or she may join one of the teams as fifth/sixth participant; if three are left, they form a team.

1.4 Repository for code, data and documentation

Register at github.zhaw.ch with your student credentials to activate your Github account (in most cases you should have done this already during your studies). Do not confuse it with github.com. Create a single team repository for your team named in the form of `<SCAD>-<whatever>`. Do not forget to add your lecturer (spio) to the repository. Install the appropriate Git client tools (e.g. CLI, Github Desktop or SourceTree) for your operating system. Clone the team repository and introduce a test commit per student to check that it is working.

Use one folder per lab (i.e. P01, P02, ...) to maintain a certain order.

Homework if necessary: Get familiar with the Git workflow to make sure you can be productive as a team with parallel work later.

2. Experiment

2.1 Cloud function

Write a simple cloud function or two following the template offered by the provider. You may use any of the supported programming languages. The function should be slightly more complex than “hello world”. In particular, its runtime should be ideally in the order of a few seconds. It may also have to do some actual processing (E6-E8 below). Test that the function works by invoking it with sample data through the web interface.

2.2 Exploration

Write a test tool which invokes the function 100 times from your computer. Measure the response times as baseline experiment E0 and report on the min, max, mean and median averages.

Then, choose four of the experiment topics to work on. Each experiment examines the influence of a certain factor on the execution times. Is the function behaviour correlating with some of these? Is the deviation becoming less predictable? Check with the lecturer if your choice is valid as there should be a balance between topics.

E1: size of input data
E2: configured memory allocation
E3: function code size [binary decision: will it run or not?]
E4: library size - inclusion of a large library
E5: usage pattern (incl. first time) - including breaks between invocations
E6: provider-internal network calls - e.g. to database
E7: external network calls - e.g. to parse website
E8: processor affinity - check /proc/cpuinfo

Observe the effects such as startup latency and stability/deviation of execution duration over time. Record the statistical results and plot at least four diagrams. Also consider the documented limits by each used provider, such as maximum function size, request payload size or memory allocation.

2.3 Report

Create a report per team (3-4 pages) in which you describe the experiment setup, procedure, results, findings and any interesting observations you may have had. The format of the report (Markdown, PDF) is up to you. Submit the report by committing it into your Git repository and notifying the lecturer about it. Always maintain a README (per repo or per folder) mentioning the team, the task and what has been achieved.