Distributed Systems (2020W)

Homework 04 - Distribute an application with Serverless Computing (FaaS)

Sashko Ristov

November 2020

DEADLINE: Monday, 16.11.2020, 08 am.

The aim of this homework is to offload the workload using serverless computing (Function-as-a-Service - FaaS). In particular, you need to develop, deploy, distribute, run, and evaluate a serverless application using the AWS Lambda service[1].

1 Given codes

All codes are given on the following link[2].

1.1 Fibonacci.java

Provided is the class *Fibonacci.java*, which contains the source code of the Fibonacci calculation used in previous homeworks (Homework 02 and Homework 03). You will need to adapt it, deploy it and run it as a serverless function.

1.2 Provided tools

Provided is our jFaaS tool, which allows you to invoke functions on multiple FaaS systems (AWS Lambda, IBM Cloud Functions, Azure Functions, Alibaba Function Compute and Google Cloud Functions). Follow the README file for how to invoke functions in a portable/transparent way using the Gateway class.

2 Run a sequential function on AWS Lambda

2.1 Development task - Develop a serverless function

You need to develop a single lambda function `LambdaFibonacci()` in JAVA that calculates the Fibonacci numbers for a given input. You can use numbers

[1] https://aws.amazon.com/lambda/
[2] https://github.com/sashkoristov/PSDS2020W/tree/main/H04/Resources

from 1 to 35 to avoid function duration limitation. Make sure to convert the input and output to JSON format (key-value pair).

2.2 Deployment task

Crate a jar file from your project and deploy it as

- the function `LambdaFibonacci128MB()` with 128 MB.
- the function `LambdaFibonacci2GB()` with 2 GB.

You can use the AWS management console to deploy the functions.

2.3 Development task - Develop a Java invoker

You need to develop a Java application that will invoke your serverless functions synchronously. The application should receive three parameters: Function name, its input, and the number of sequential repetitions. For this purpose, we propose to use our jFaaS tool, which is given in the same GitHub link. The expected duration of `LambdaFibonacci128MB()`, which calculates Fibonacci numbers from 1 to 35 is around 20 s. For testing, you can use a smaller input array.

2.4 Evaluation task - Evaluate cold-warm start and memory impact

Invoke the function `LambdaFibonacci128MB()` five times immediately one after the other time. Visualize the execution time of each execution. What do you observe considering cold vs. warm starts?

Repeat the same with the function `LambdaFibonacci2GB()`. Do you observe any improvement in execution time compared to `LambdaFibonacci128MB()`?

Inspect the execution of your function and determine how much memory both functions used. Discuss the used memory vs. performance of both deployed functions.

3 Scale and distribute the problem size

After you learned how to use the AWS Lambda service, let's use horizontal scaling to distribute the work and to speed up the execution.

3.1 Development task

Adapt your invoker application that will loop from 1 to 35 and will invoke the function `LambdaFibonacci128MB()` for each iteration. Use threading to invoke each function `LambdaFibonacci128MB()` concurrently and synchronously.

3.2 Evaluation task - Evaluate distribution of the work

Run five repetitions in your invoker (calculate five time the same Fibonacci numbers from 1 to 35). Visualize the execution time of each execution. What do you observe considering cold vs. warm start?

Compare the costs for running a single "fat" `LambdaFibonacci128MB()`, `LambdaFibonacci2GB()`, and the distributed "thin" `LambdaFibonacci128MB()` functions through your invoker. What is the cost - performance ratio?

4 Upload

In order to pass the task, you have to upload all .java files used in your solution, as well as a text file containing a description and discussion of the deployment and evaluation tasks.