Distributed Systems (2020W) - Proseminar

Homework 06 - Build an AFCL file for your project

Sashko Ristov

November 2020

DEADLINE: Monday, 30.11.2020, 08 am.

In the previous Homework 05, you were able to build a scalable serverless workflow application (Function Choreography - FC) in our AFCL (Abstract Function Choreography Language) using our FC Editor. After building the FC, you were able to run a simple FC using our *xAFCL* enactment engine.

In this homework, you need to apply the gained knowledge from Hoomework 05 and develop an FC for your project, also in AFCL.

1 Given codes / tools

The following GitHub repository[1], provides several additional resources for this task:

1.1 The schema for AFCL

Provided is a schema (both in YAML and JSON) that you can use to validate your FC (for Section 2.3.2).

1.2 *xAFCL* Enactment Engine

Provided is our *xAFCL* tool.

2 Build and test the FC for your project

2.1 Development task

Build a draft version of the FC for your project following these steps:

1. Check the project that is assigned to your team.

[1] https://github.com/sashkoristov/PSDS2020W/tree/main/H06/

2. Explain in a report which compound functions you will use, the dependencies between them, and the level of parallelism (parallel sections and parallel loops). You will use this part in your final report for the project.

3. Think of your functions and their dataIns and dataOuts. How will you specify the data-flow and data distribution?

4. Analyse and build the assigned FC in AFCL using the FC Editor.

5. Save the visual representation of your draft FC (save as YAML and XML).

2.2 Evaluation task

We recommend that you develop a single serverless function `allInsOuts` in the programming language that you will use in your project and, for now, use it as a placeholder for all functions in the FC (to later replace it with the actual implementation in Homework 07). `allInsOuts` should receive and return all key/value pairs that appear across the FC. Also, you can use the same ARN for each function.

Note: Contact us (Discord, email, or open hours) in case you have questions regarding the *xAFCL* enactment engine. We will give you a feedback as soon as possible.

2.3 Hints

2.3.1 Hints for the FC

Follow these recommendations to minimize the mistakes in your FC:

- General recommendations:
  - Do not forget to fill the `resource` attribute in the `properties` of each base function. Do not fill the `properties` from `dataOuts` or `dataIns`.
  - Use the same value for the key `name` of `dataIns` as the keys in the JSON that you pass to each base function
- Regarding the control flow:
  - Follow the mandatory rules in the schema
  - Mandatory use both `then` and `else` branches for each `if` compound function.
- Regarding the data flow:
  - Use the sources for `dataOuts` from the immediate predecessors (either at the same level, or from the outer function). If you need to make a source from an earlier `dataOuts`, then use passing for that `dataOut` through each function of the flow until the predecessor.

  - use BLOCK(N) for each dataIns in the parallelFor that is a collection. *N* has to be a number (cannot be a variable - dataOuts of a function). This will allow you to send *N* elements of the collection to each iteration. Probably you will use BLOCK(1) and send a single element to each iteration.
  - use REPLICATE(*) for each dataIns in the parallelFor that is a primitive (tested are a number and a string). This will allow you to send the same dataIns to each iteration.
  - dataOuts of a parallelFor, whose source is dataOuts of an inner function, has to be of a type collection. Similar for dataOuts of if and parallel.

2.3.2 Validate your FC

Validate your FC before running it with *xAFCL*:

- Validate your FC with the FC editor
- Validate your FC with the schema. Proposed steps:
  - Convert YAML to JSON[2]
  - Use the JSON schema validator[3]

3 Upload

In order to pass the task, you have to upload the code of your test function(s), the test input .JSON file, the .yaml and .xml files of your draft FC, as well as a text file containing a description and discussion of the development and evaluation tasks.

[2] https://www.json2yaml.com/

[3] https://www.jsonschemavalidator.net/