TOR VERGATA

UNIVERSITÀ DEGLI STUDI DI ROMA

Macroarea di Ingegneria

A QoS-Aware Broker for Multi-Provider Serverless Applications

Andrea Graziani
m. 0273395

Supervisor: Valeria Cardellini
Tutor: Gabriele Russo Russo

May 23, 2022

Serverless Computing: Overview

My thesis is focused on "**serverless computing**".

It is a development paradigm according to which:

- The provider takes care of all aspect of server management.
- Small-granularity billing pricing model: **pay-as-you-go**.
- Cloud application are abstracted as a group of so-called **serverless functions**, which are computation units implementing a business functionality.

Serverless Computing: Overview

A serverless function is executed inside a containerized environment: the so-called **function instance**.

The FaaS platform **automatically** scales the number of function instances.

FaaS platforms impose a **limit** on the number of function instance runnable at the **same time** called **concurrency limit**.

A delay is observed when a new function instance is started by the provider: this event is called **cold start**.

Serverless Computing: Overview

To invoke a **serverless function**, users have to specify a so-called **serverless function configuration**.

- Generally, the **amount of memory allocated** to a serverless function.

Configuration parameters **significantly** affect the **cost** and **response time** of serverless functions.

Serverless Computing: Problems

- The fulfillment of **non-functional requirements** concerning the **quality of service** (QoS) levels that should be guaranteed for **multi-provider serverless applications**.
- The lack of support for application whose functions are hosted on **multiple providers**.
- The lack of support for serverless function implementations abstraction, that is, for the so-called **concrete functions**.
- The fulfillment of **functional requirements** concerning the **orchestration** of multi-provider applications.

State of Art

Solutions concerning QoS fulfillment **already exist**.

- Many solutions rely on **QoS-aware scheduling algorithms** while others rely on the formulation and **solving of optimization problems**.
  - Generally, proposed solution do **not** support Multi-Provider serverless applications.

- Despite there are some exceptions, current solution are unaware of both the current status of FaaS platforms and user traffic.

Thesis Goals

Goal # 1

To guarantee the **satisfaction of QoS levels** for multi-provider serverless applications.

It was necessary to develop:

1. An **analytical model** to evaluate aforementioned class of applications.
2. A **methodological way** to find the "best" configuration to satisfy QoS constraints.
    - By solving an **optimization problem** (LP).
3. A custom **heuristic algorithm** to rapidly resolve the aforementioned optimization problem.

Thesis Goals

Goal # 2

The **orchestration** for multi-provider serverless applications.

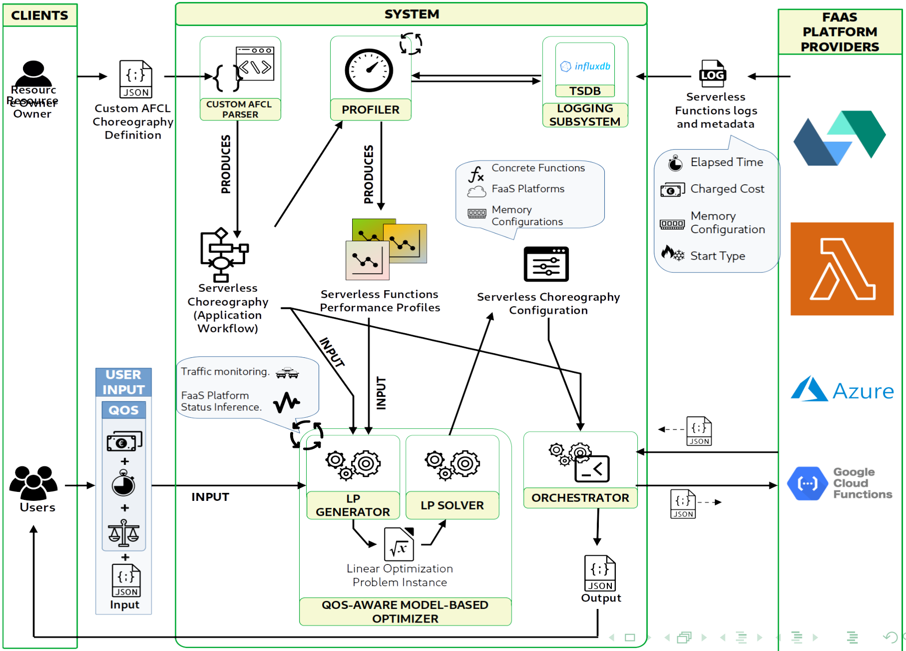
To achieve it, I had to build:

1. A **software framework**
   - Users/Serverless application management (CRUD operations, profiling tasks, orchestration task, etc.)
2. An extension to an already existent **representation scheme** to define a serverless application workflow.
   - Based on an existing language called **abstract function choreography language** (AFCL).

Prototype

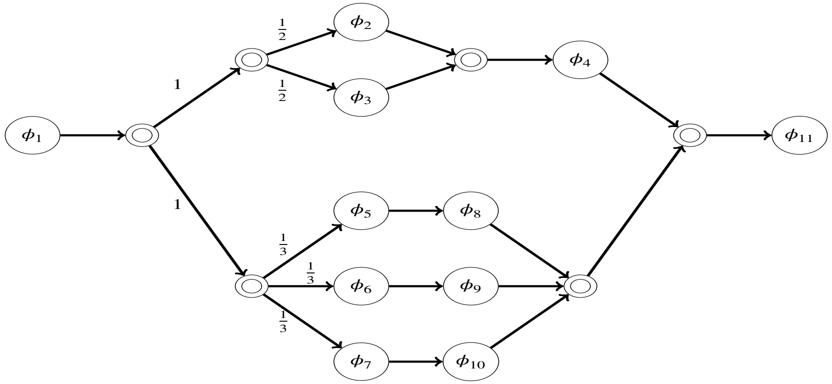
Main features of our software framework:

- **Client-server architecture**.
- **Cloud-native** application.
- Includes a set of **adapters** to interact with following FaaS providers:
  - AWS Lambda.
  - Apache OpenWhisk.
- **REST** architectural style.


CLIENTS
Resource Owner
Custom AFCL Choreography Definition
JSON
SYSTEM
CUSTOM AFCL PARSER
PROFILER
influxdb
TSDB
LOGGING SUBSYSTEM
LOG
Serverless Functions logs and metadata
FAAS PLATFORM PROVIDERS
PRODUCES
PRODUCES
Concrete Functions
FaaS Platforms
Memory Configurations
Elapsed Time
Charged Cost
Memory Configuration
Start Type
Serverless Choreography (Application Workflow)
Serverless Functions Performance Profiles
Serverless Choreography Configuration
Azure
Google Cloud Functions
INPUT
INPUT
USER INPUT
QOS
Traffic monitoring.
FaaS Platform Status Inference.
Users
INPUT
Input
LP GENERATOR
LP SOLVER
ORCHESTRATOR
Linear Optimization Problem Instance
QOS-AWARE MODEL-BASED OPTIMIZER
Output

System Model

Serverless applications are abstracted to a **weakly connected weighted directed graph**.



System Model

- Estimations of **average response time** and **cost** under **any** possible configuration of all concrete functions is done using **exponential moving average** approach.
- An estimation of the **probability** according to which a request follows a cold start is required.
  - This is done using **Erlang-B** formula by modeling FaaS platform providers by **sets** of $M/G/K(t)/K(t)$ queueing systems.
    - $K(t)$: the number of function instances at time t.

Optimization Problem

- To achieve our goal consisting in finding the best configuration to guarantee QoS constraints, we have to solve an **optimization problem**.
  - It is based on **multi-dimensional multi-choice knapsack problem formulation** (MMKP).

$$max \quad \sum_{i=1}^{|\mathcal{F}_\mathcal{E}(\mathcal{C})|} \sum_{j=1}^{|\mathbf{F}_{\phi_i} \times \mathbb{N}|} y_{\phi_{i_j}} \cdot p_{\phi_{i_j}}(t) \tag{5.10}$$

$$\text{subject to} \quad \sum_{i=1}^{|\mathcal{F}_\mathcal{E}(\mathcal{C})|} \sum_{j=1}^{|\mathbf{F}_{\phi_i} \times \mathbb{N}|} y_{\phi_{i_j}} \cdot c_{\phi_{i_j}}(t) \leq C \tag{5.11}$$

$$\sum_{\phi_i \in \mathcal{F}_\mathcal{E}(\mathcal{C}) \setminus \mathcal{F}_\mathcal{E}(\widetilde{\mathcal{P}})} \sum_{j=1}^{|\mathbf{F}_{\phi_i} \times \mathbb{N}|} y_{\phi_{i_j}} \cdot rt_{\phi_{i_j}}(t) + \\ + \sum_{\phi_h \in \delta_\mathcal{C}} \sum_{j=1}^{|\mathbf{F}_{\phi_h} \times \mathbb{N}|} y_{\phi_{h_j}} \cdot rt_{\phi_{h_j}}(t) \leq RT \qquad \forall \delta_\mathcal{C} \in \Delta_\mathcal{C} \tag{5.12}$$

$$\sum_{i=1}^{|\mathcal{F}_\mathcal{E}(\mathcal{C})|} \sum_{j=1}^{|\mathbf{F}_{\phi_i} \times \mathbb{N}|} y_{\phi_{i_j}} \cdot a_{(\phi_{i_j}, \omega_P^{(l)})} \leq l - R(\mathbf{Q}_{\omega_P^{(l)}}, t) \qquad \forall \omega_P^{(l)} \in \widetilde{\mathbf{S}}_\mathcal{C} \tag{5.13}$$

$$\sum_{j=1}^{|\mathbf{F}_{\phi_i} \times \mathbb{N}|} y_{\phi_{i_j}} = 1 \qquad \forall i \in \mathbb{N} \cap [1, |\mathcal{F}_\mathcal{E}|] \tag{5.14}$$

$$y_{\phi_{i_j}} \in \{0, 1\} \qquad \begin{array}{r} \forall i \in \mathbb{N} \cap [1, |\mathcal{F}_\mathcal{E}|] \\ \forall j \in \mathbb{N} \cap [1, |\mathbf{F}_{\phi_i} \times \mathbb{N}|] \end{array}$$

Heuristic Algorithm

I develop a custom heuristic algorithm based on **ant colony optimization** (ACO); it is called **pre-provisioned colony optimization algorithm with lazy pheromone update**.

- It is based on a set of computational agents, called **artificial ants**, which **iteratively** construct a solution.
- At each iteration, each agent moves from a solution to another, applying a series of stochastic **local** decisions whose policy is based on following parameters:
  - **Attractiveness**.
  - **Pheromone trail**.

Heuristic Algorithm

Pheromone trails are updated during **every iteration**.

Pheromone trails are used to decide which solutions should be preferred during **next iterations**.

- A **lazy approach** for pheromone trails update is used.
- A **pre-provisioning tactic** is used to anticipates data needs assuring a **lower latency**.

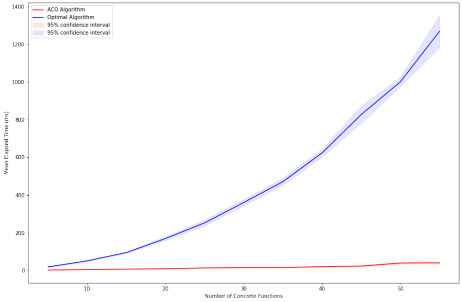
Validation Test

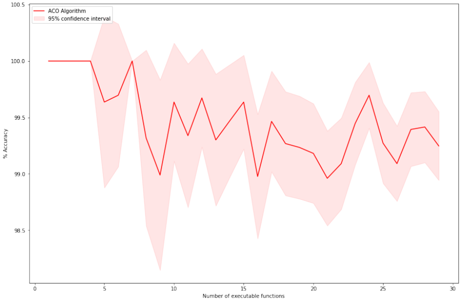
We validate our model through several experiments using an image-processing serverless application.

- Firstly, we check the respect of user specified QoS constraints in a static way thorough several **sequential invocations**.
- Then, we test the model in a dynamically way thorough **several concurrent and parallel invocations**.
  - That experiment aimed to **run out of capacity** on one FaaS provider and so force our prototype to schedule the concrete function on the other.

Heuristic Algorithm Evaluation

- We compared performance of our heuristic algorithm with that of optimal algorithm through several experiments.





Conclusions

1. I presented an analytical model to evaluate the performance of multi-provider serverless application
2. I defined an optimization problems formulation to address the problem of finding a suitable configuration in order to meet user defined QoS constraints
3. I developed a heuristic algorithm to rapidly solve it.
4. I validate proposed solution through test and experimental evaluations using my prototype.

Thanks for your attention!

Questions?