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

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

My thesis is focused on "**Serverless Computing**".

The Goal

The fulfillment of **non-functional requirements** concerning the **quality of service** (QoS) levels that should be guaranteed for any **generic** serverless workflow supporting:

- **Multiple FaaS Providers**
- **Multiple Serverless Functions Implementations.**

Serverless Computing: Overview

What's meant by serverless computing?

It is an execution model in which:

- Administration tasks (provisioning, monitoring, scaling, etc.) are directly managed by the provider.

- Small-granularity billing pricing model: **pay-as-you-go**.

- Cloud application are abstracted as a group of so-called "**Serverless Functions**".

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

What is a serverless function?

A computation unit implementing a business functionality.

- **Stateless**
- **Event-Driven**
- **Short-Lived**

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

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

Serverless Computing: Overview

Concurrency Level

The limitation, imposed by FaaS platforms, on the **number** of function instance runnable at the **same time**.

Providers apply that restriction differently:

- **Global (Per-Account) Concurrency Model**.
  - AWS Lambda.
  - IBM Cloud Functions.

- **Local (Per-Function) Concurrency Model**.
  - Google Cloud Functions.

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

To invoke a **serverless application**, users have to specify a configuration for **all** its functions.

Configuration parameters **significantly** affect the **cost** and **response time** of serverless functions.
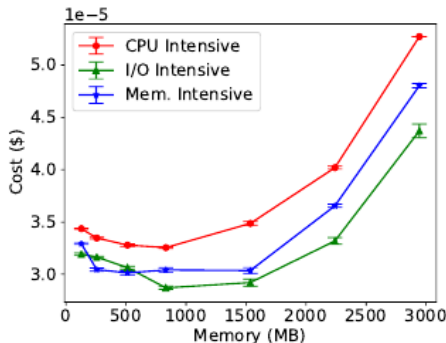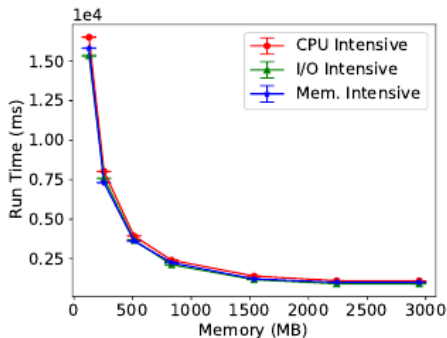
What current problems or limitations of the serverless computing paradigm I tried to solve?

- The fulfillment of quality of service (QoS) levels.
- No support for application whose functions are hosted on multiple providers.
- No support for application whose functions have more than one implementation (**Concrete Functions**).
- Vendor lock-in issues.

Solutions **already** exist, but:

- They are unaware of the current status of FaaS platforms.

- No support for Multi-Provider and Multiple-Implementations serverless applications.
  - Lack of an analytical model to evaluate their performance.

- Many do not support generic workload applications.

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

My contributions respect to the current state of art:

1. An **Analytical Model** to evaluate applications performance.
   - Supports generic workflow (branch, loop, parallel)
   - Multi-Provider support.
   - Support for Multi-Implementation for functions.
   - FaaS-Status-Aware.

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 resolve the aforementioned optimization problem.
   - Based on **Ant Colony Optimization** algorithm (ACO) family.

4. A **QoS-Aware Software Framework** for the orchestration for multi-provider, multiple-implementations serverless applications.

5. 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).

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

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.

The Analytical Model
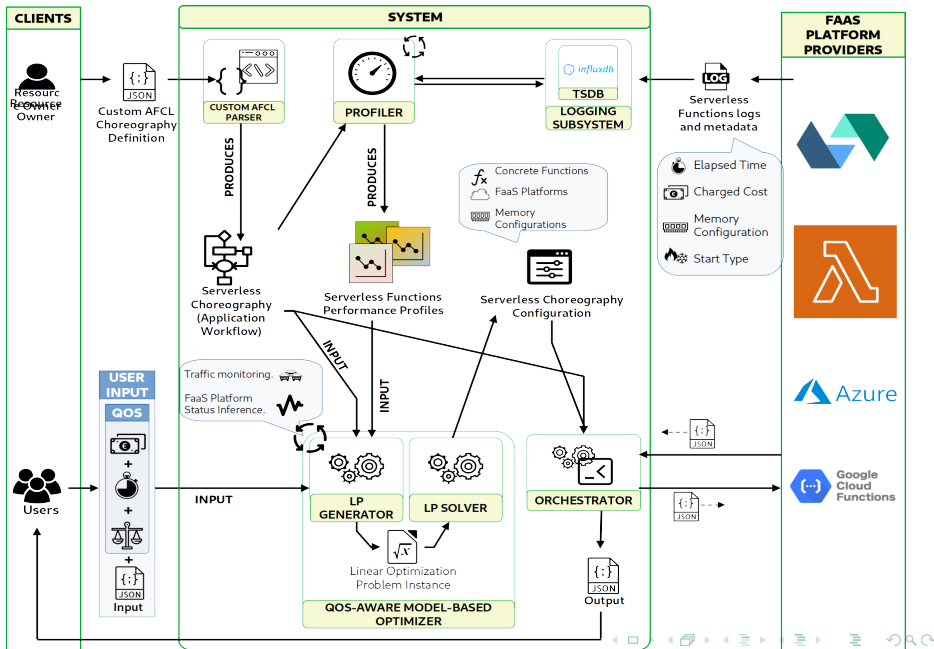
> Serverless applications are abstracted to a
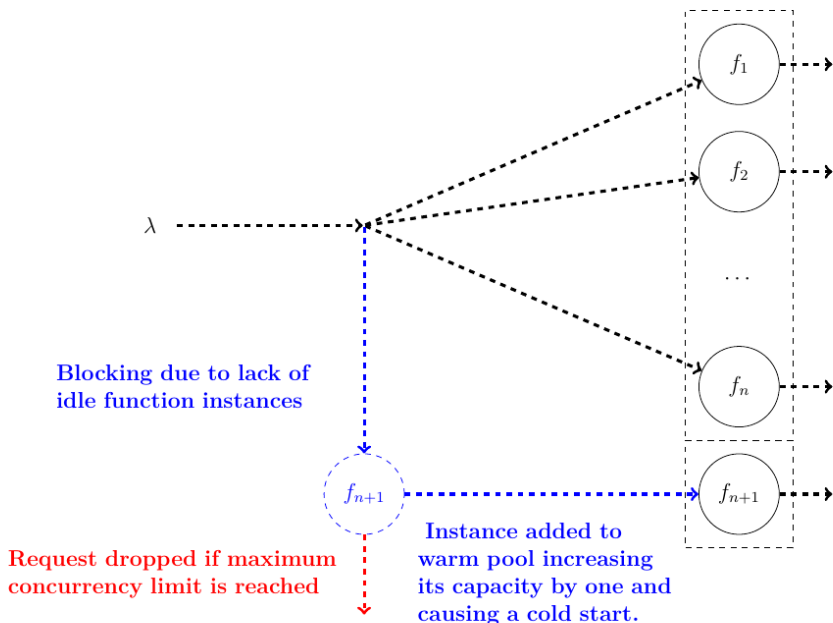> **Weakly Connected Weighted Directed Graph**.

- Each **vertex** models an **abstract serverless function**, that is a computational unit required by business logic.
  - To each abstract function corresponds a set containing all concrete function implementing it.

- Each **edge** represents the calling relationship between two abstract functions.
  - Edge weights represent the so-called **transition probability**.

To evaluate the performance of a **Concrete Function**:

- Estimations about the **Average Response Time** (**Cost**) in case of cold (warm) start are required.
  - To compute aforementioned estimations, **Exponential Moving Average** based approach is adopted.

- An estimation of the probability according to which a request follows a cold start (**Cold Start Probability**) is required.

The Analytical Model

To compute **Cold Start Probability**:

- Any FaaS platform provider is modeled by **set** of $M/G/K(t)_{C_{max}}/K(t)_{C_{max}}$ queueing systems.
  - $K(t)$: the number of function instances at time t.
  - C_{max}: the concurrency limit.

- Aforementioned set depends on the concurrency model adopted by providers.

- **Erlang-B** formula is used.

Warm Pool having capacity n

λ

f_1

f_2

$\ldots$

f_n

f_{n+1}

f_{n+1}

Blocking due to lack of idle function instances

Request dropped if maximum concurrency limit is reached

Instance added to warm pool increasing its capacity by one and causing a cold start.

I formalized algorithms and equations to evaluate the performance of both a **serverless application** and of an **abstract function**.

Aforementioned algorithm depends on:

- The topological properties of the graph representing the application.
- The actual value of transition probabilities of all edges.

- 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**.

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Formally, the configuration vector $\mathbf{x}_{\mathscr{C}}$ we want to find is such that:

$$\mathbf{x}_{\mathscr{C}} \overset{\text{def}}{=} \left\{ x_{\phi_1}, \ldots, x_{\phi_k} \right\}$$

$$\in \left\{ \left\{ \bigcup_{j=1}^{|\mathbf{F}_{\phi_1}|} f_{\phi_{1_j}} \times \mathbf{M}_{f_{\phi_{1_j}}} \right\} \times \ldots \times \left\{ \bigcup_{j=1}^{|\mathbf{F}_{\phi_k}|} f_{\phi_{k_j}} \times \mathbf{M}_{f_{\phi_{k_j}}} \right\} \right\}$$

$$= \bigtimes_{i=1}^{k} \left\{ \bigcup_{j=1}^{|\mathbf{F}_{\phi_i}|} f_{\phi_{i_j}} \times \mathbf{M}_{f_{\phi_{i_j}}} \right\}$$

$$\subseteq \bigtimes_{i=1}^{k} \left\{ \mathbf{F}_{\phi_i} \times \mathbb{N} \right\} = \mathbf{X}_{\mathscr{C}} \tag{1}$$

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

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

$$\sum_{\phi_i \in \mathcal{F}_{\mathcal{E}}(\mathcal{C}) \setminus \mathcal{F}_{\mathcal{E}}(\widetilde{\mathcal{P}})} \sum_{j=1}^{|\mathbf{F}_{\phi_i} \times \mathbb{N}|} y_{\phi_{i_j}} \cdot rt_{\phi_{i_j}}(t) +$$

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

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

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

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

The 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**.

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**.

Conclusions
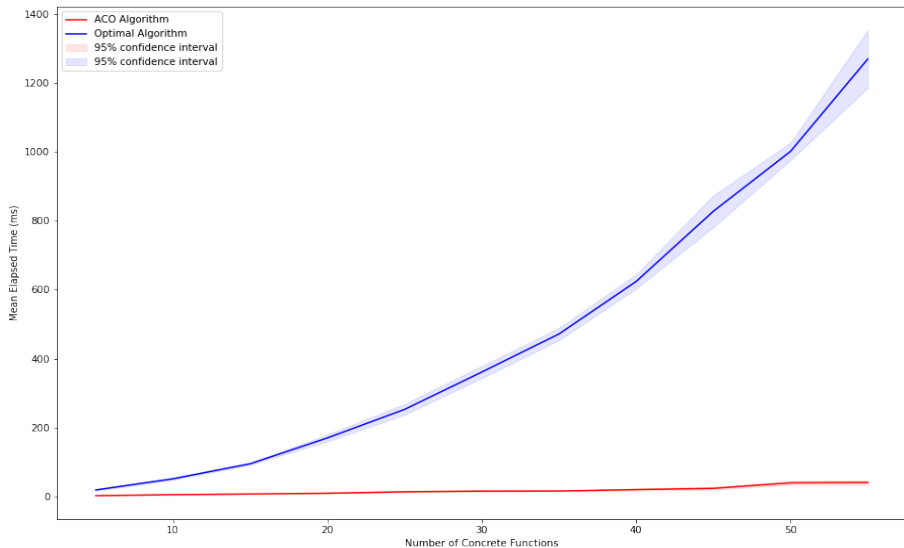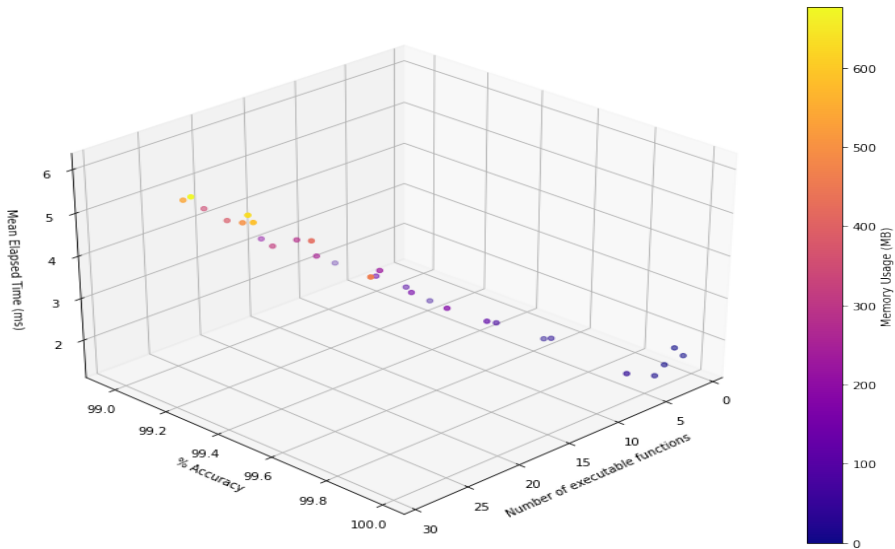
- I presented an analytical model to compute the performance of a generic serverless workflows having multiple concrete functions hosted on multiple FaaS providers.

- I defined an optimization problems formulation to address the problem of finding a suitable configuration in order to meet user defined QoS constraints

- We developed a heuristic algorithm to rapidly solve it.

- We verified the validity of both the proposed algorithms and the analytical model through test and experimental evaluations developing a prototype supporting AWS and OpenWhisk.

Thanks for your attention!
Questions?