



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

My thesis is focused on “**Serverless Computing**”.

It is a development paradigm according to 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**”.

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.

FaaS platforms impose a **Limit** on the number of function instance runnable at the **Same Time** called **Concurrency Limit**.

- Providers apply that restriction differently.

A delay is observed when a new function instance is started by the provider; this is called **Cold Start**.

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.

- The fulfillment of **Non-Functional Requirements** concerning the **Quality of Service** (QoS) levels that should be guaranteed for any **generic** serverless workflow.
- 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, Multiple-Implementation Serverless Applications**.

Solutions concerning QoS fulfillment **already exist**.

- Despite there are some exceptions, they are unaware of both the current status of FaaS platforms and user traffic.
- Some solutions provide no support for generic workflow.
- No support for Multi-Provider and Multiple-Implementations serverless applications.
 - No one provides an analytical model to evaluate their performance.
- Many solutions rely on QoS-aware scheduling algorithms while others rely on the formulation and solving of optimization problems.

Goal # 1

To guarantee the **Satisfaction of QoS Levels** for Multi-Provider and Multiple-Implementations serverless applications.

It was necessary to develop:

- 1 An **Analytical Model** to evaluate applications performance.
 - Supporting generic workflow (branch, loop, parallel)
 - Providing support for Multi-Provider, Multiple-Implementation Serverless Applications
 - 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 rapidly resolve the aforementioned optimization problem.
 - Based on **Ant Colony Optimization** algorithm (ACO) family.

Goal # 2

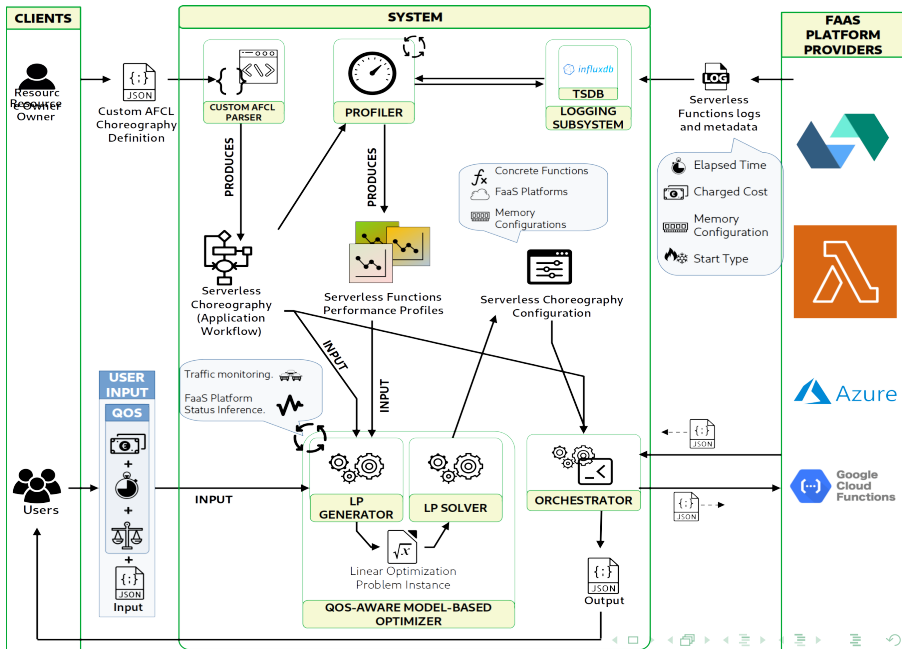
The **Orchestration** for Multi-Provider and Multiple-Implementations serverless applications.

To achieve it, I had to build:

- 1 A **Software Framework**
 - Users/Serverless Application Management (CRUD operations, profiling tasks, orchestration task, etc.)
 - User-Traffic/QoS/FaaS-Status Aware.
- 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).

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.



Serverless applications are abstracted to a
Weakly Connected Weighted Directed Graph.

- Each **vertex** models an **abstract serverless function**.
 - 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**.

The model provides algorithms and equations to evaluate the performance of a **serverless application** under a given configuration.

Aforementioned algorithm depends on:

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

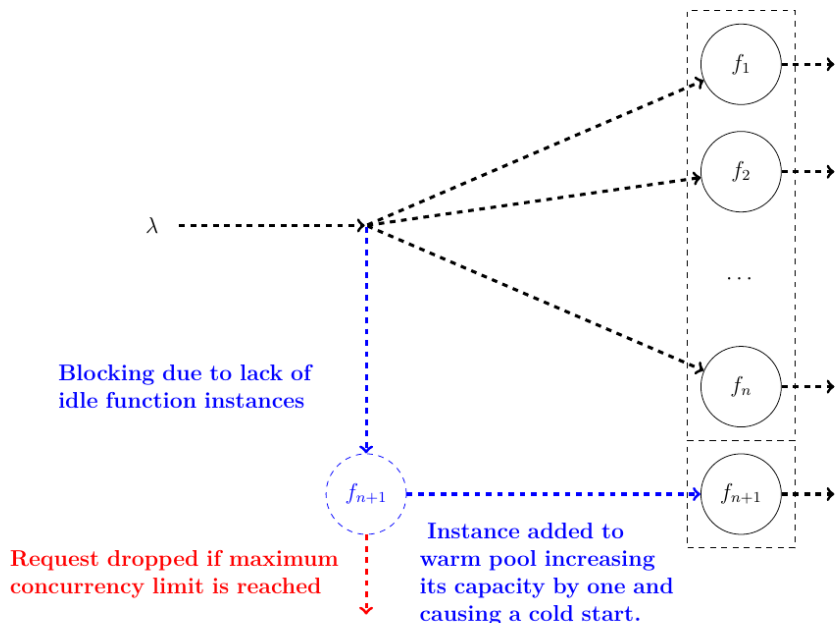
To evaluate the performance of a concrete function is required:

- An 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**)

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.

To compute the cold start probability the **Erlang-B** formula is used.



- 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**.

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

$$\begin{aligned}
 \mathbf{x}_{\mathcal{C}} &\stackrel{\text{def}}{=} \left\{ x_{\phi_1}, \dots, x_{\phi_k} \right\} \\
 &\in \left\{ \left\{ \bigcup_{j=1}^{|\mathbf{F}_{\phi_1}|} f_{\phi_{1j}} \times \mathbf{M}_{f_{\phi_{1j}}} \right\} \times \dots \times \left\{ \bigcup_{j=1}^{|\mathbf{F}_{\phi_k}|} f_{\phi_{kj}} \times \mathbf{M}_{f_{\phi_{kj}}} \right\} \right\} \\
 &= \bigtimes_{i=1}^k \left\{ \bigcup_{j=1}^{|\mathbf{F}_{\phi_i}|} f_{\phi_{ij}} \times \mathbf{M}_{f_{\phi_{ij}}} \right\} \\
 &\subseteq \bigtimes_{i=1}^k \left\{ \mathbf{F}_{\phi_i} \times \mathbb{N} \right\} = \mathbf{X}_{\mathcal{C}} \tag{1}
 \end{aligned}$$

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

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

$$\begin{aligned} & \sum_{\phi_i \in \mathcal{F}_{\mathcal{E}}(C) \setminus \mathcal{F}_{\mathcal{E}}(\tilde{\mathcal{P}})} \sum_{j=1}^{|\mathbf{F}_{\phi_i} \times \mathbb{N}|} y_{\phi_{i_j}} \cdot rt_{\phi_{i_j}}(t) + \\ & + \sum_{\phi_h \in \delta_C} \sum_{j=1}^{|\mathbf{F}_{\phi_h} \times \mathbb{N}|} y_{\phi_{h_j}} \cdot rt_{\phi_{h_j}}(t) \leq RT \end{aligned} \quad \forall \delta_C \in \Delta_C \quad (5.12)$$

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

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

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

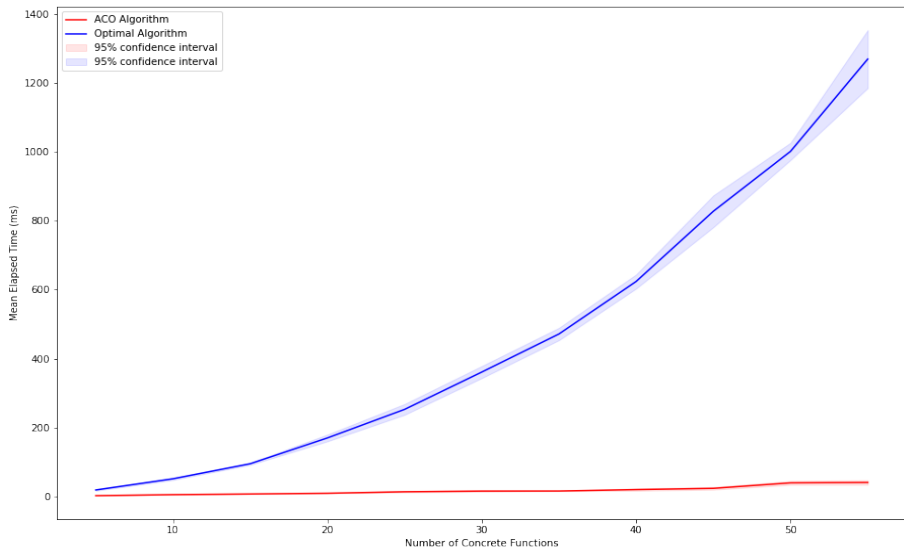
I develop a custom heuristic algorithm based on **Ant Colony Optimization** (ACO); it is called **Pre-provisioned Colony Optimization Algorithm with Lazy Pheromone Update**.

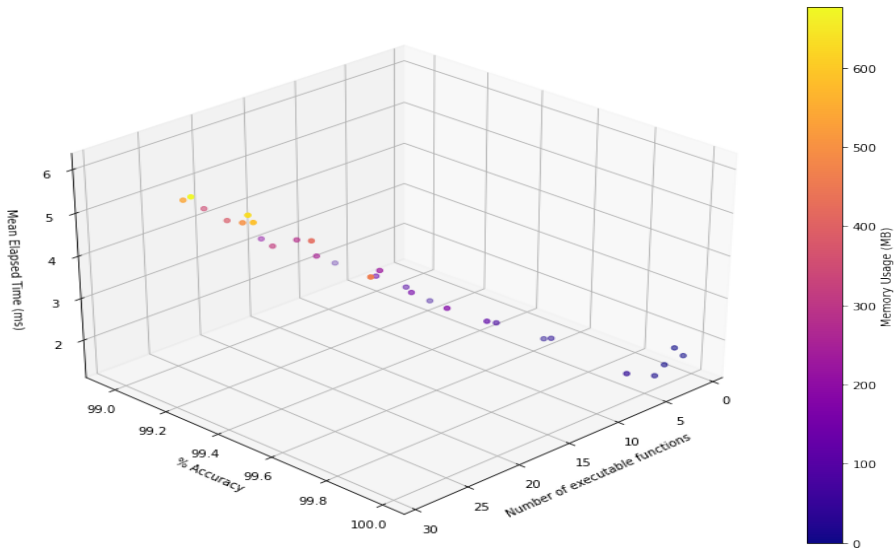
- It is based on a set of computational agents, called **artificial ants**, which **iteratively** construct a solution.
- At each iteration, each agent moves from a solution to another, applying a series of stochastic **local** decisions whose policy is based on following parameters:
 - **Attractiveness**.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





- 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?