IEEE World Forum on Internet of Things

\Orchestrating a brighter world

NEC

BigDataStack
Holistic stack for big data applications and operations

# Reinforcement Learning based Orchestration for Elastic Services

Mauricio Fadel Argerich <mauricio.fadel@neclab.eu>

Bin Cheng

Jonathan Fürst

15-18 April 2019 – Limerick, Ireland

# Agenda

1. Introduction
2. Programming model
3. Dynamic Orchestration
4. Experimental Evaluation
5. Takeaways & Future work

# Introduction

- Edge computing reduces network stress for operators and improves service responsiveness by allocating computation closer to data producers and consumers

- Challenges
  - Hardware is constraint
  - Hardware is heterogeneous
  - No Cloud-like elasticity features (scale out, scale up, etc.)

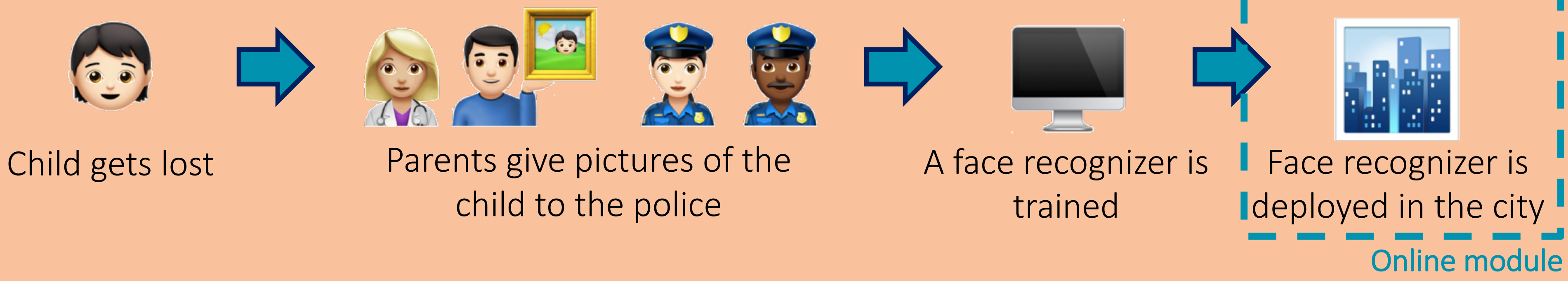
- When deployed, services do not meet their Service Level Objectives (SLOs)

- We have already introduced Elastic Services that dynamically adapt to the current execution context to better comply with their SLOs
  - But how do services decide when and how to adapt?

- In this work, we propose Reinforcement Learning (RL) based approach so edge services can adapt and achieve their SLOs

# Motivation Use Case: The Lost Child Service

## Overall working




## Online module


### Requirements

- To process at least 1 FPS in order to find child even if it appears briefly in image
- To analyse image with precision as high as possible

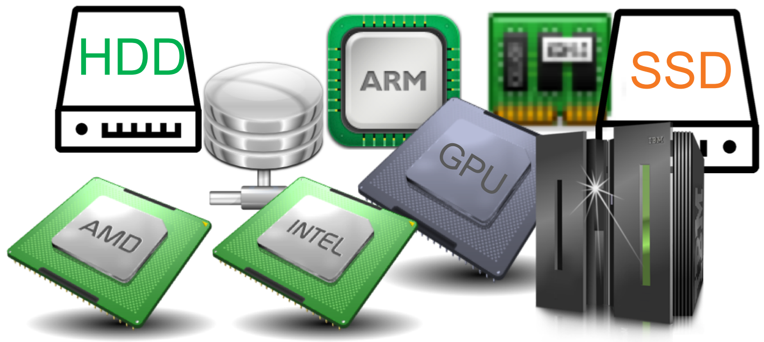
# Problem



## The online module must be deployed on the Edge

- Sending video from several cameras to the cloud to be analyzed is not feasible

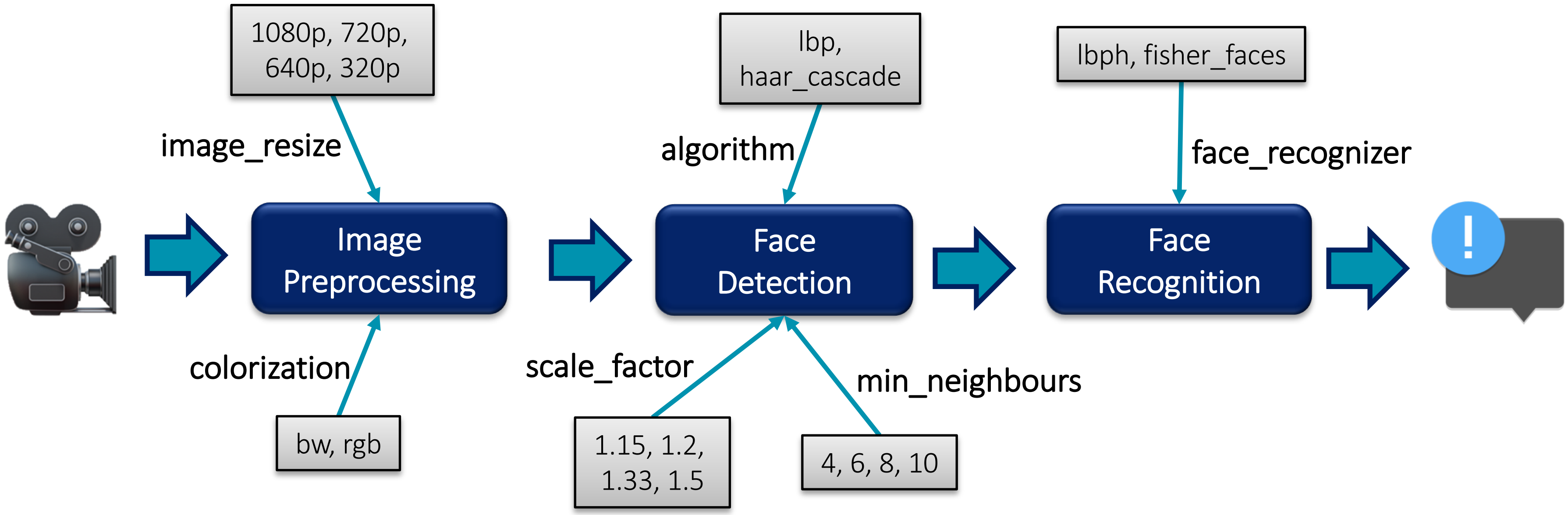
When service is deployed, requirements are not met




Hardware in the Edge is highly heterogenous

Inputs are highly variable

# Adaptation Knobs




- Modifying a parameter affects the service performance in different dimensions
  - Common trade-off: accuracy v. end-to-end latency
- Optimal values vary depending on:
  - Input (image resolution, FPS, image quality, number of faces in image, etc.)
  - Execution context (processing node hardware, shared resources status)
  - Goals (fastest response, highest accuracy, end to end latency < 1s, precision > 0.8, etc.)
- Number of alternative behaviors grows exponentially; even for this simple service:

$$4^3 \cdot 2^3 = 512 \text{ different configurations!}$$

# Programming model

- Services are broken down into small processing functions → **operators**
  - These operators are parameterized to change their internal execution during runtime
- Once operators and their implementations are provided, service topology is defined
- The goal is to achieve and ensure the required QoS continuously, by making orchestration decisions adapted to the current input and context
- We use FogFlow as our edge execution framework
- FogFlow dynamically orchestrates elastic services over cloud and edges, in order to reduce internal bandwidth consumption and offer low latency. It is context-driven, taking orchestration decisions on different contexts:
  - System
  - Data
  - Usage

# Dynamic Orchestration

- In order to generalize our approach over services with different numbers and types of requirements, we model the problem as a constrained optimization problem

- SLOs or requirements → constraints
  - to process documents with end-to-end latency less or equal than 1s
  - to run at a cost of less or equal than $10 per hour

- Service goal → objective
  - Precision
  - Accuracy
  - Battery efficiency

# Optimization Problem

$$\max_{\theta} O(\theta)$$
$$subject\ to\ c_i(\theta) \leq C_i, i = 1, \dots, N$$

$\theta$: is the configuration of adaptation knobs used for all of the operators

O($\theta$): represents the objective of the service,, which is determined by the configuration of parameters used

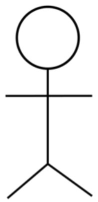
$c_i(\theta)$: is a constraint to the service (such as latency), also determined by $\theta$

$C_i$: is a constraint target(e.g., 1s)

N: is the total number of constraints

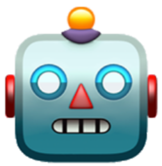
# Steps to define an Elastic Service





Developer

1. Objective of Service
   Max. precision, min. latency, min. cost
2. Service Requirements and metrics
   end-to-end latency must be less than 1 second, related metric: latency
   at least 1 frame should be processed per second, related metric: latency
3. Operators and Adaptation knobs
   image resolution to be analyzed: [1080p, 720p, 640p, 320p]
   classifier to be used: [LBP, Haar Cascade, CNN]



Elastic Service

4. Finds best configuration for adaptation knobs through its **Dynamic Orchestration**

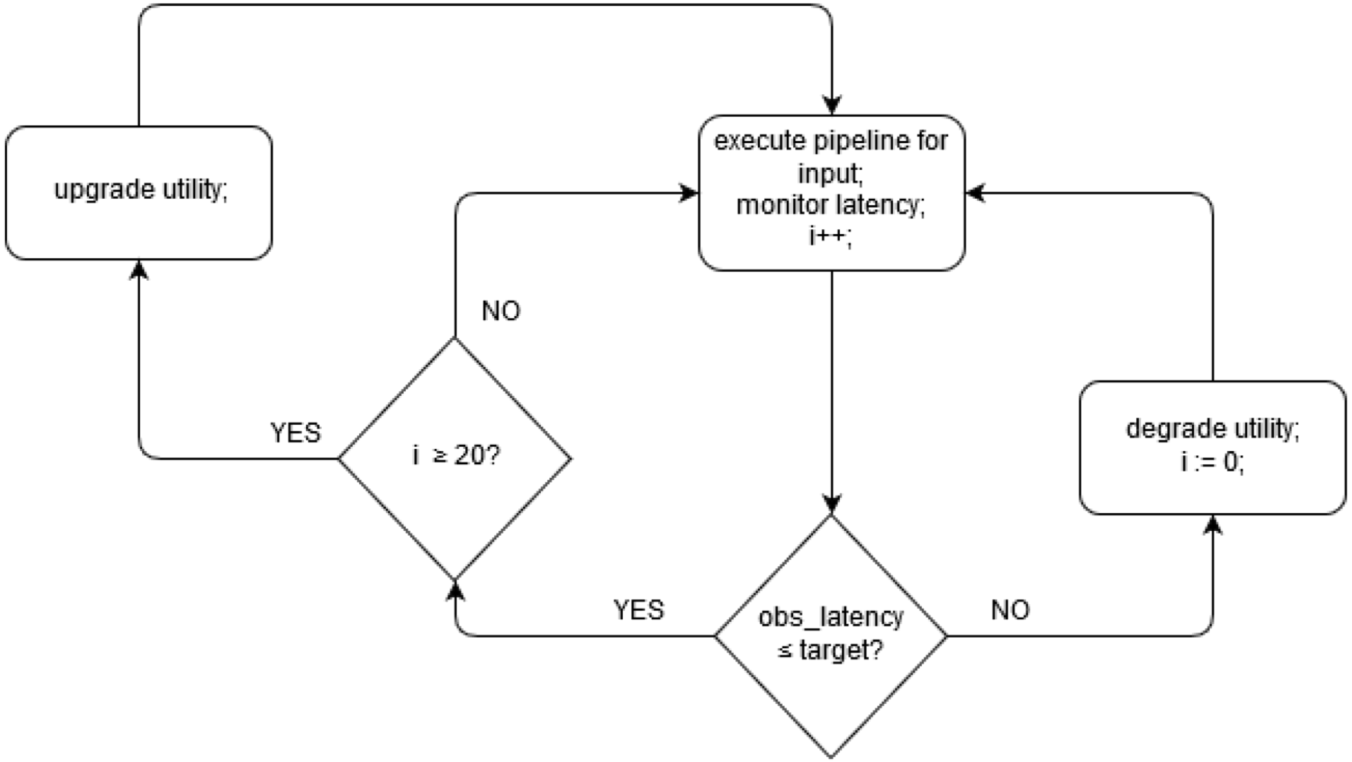
**Requirements for Dynamic Orchestration**

- **Rapid response:** it must adjust the service behavior rapidly to keep up with changes during runtime
- **Low overhead: it** must not create a considerable overhead for the system

# Heuristics based Orchestration



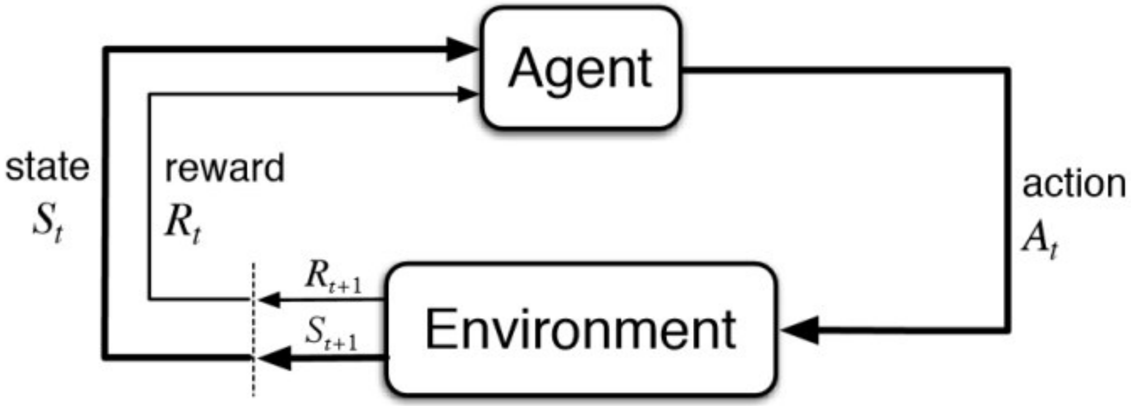
- Exploits common trade-off: latency v. precision
  - Disadvantage: requirements and service performance must exhibit trade-off
- Simple logic: start with "best performance", if requirements are not met, reduce performance until requirements are met
- We also include a mechanism for trying upgrading performance if it seems possible



# Reinforcement Learning 101



- RL enables an agent to learn in an interactive environment by trial and error
- RL is often considered to be in between supervised and unsupervised learning
- It models the problem as a Markov Decision Process (MDP)
  - A MDP is a discrete time stochastic decision control process, which consists of finite environment states S, a set of possible actions A(s) in each state, a real valued reward function R(s) and a transition model P(s', s | a)



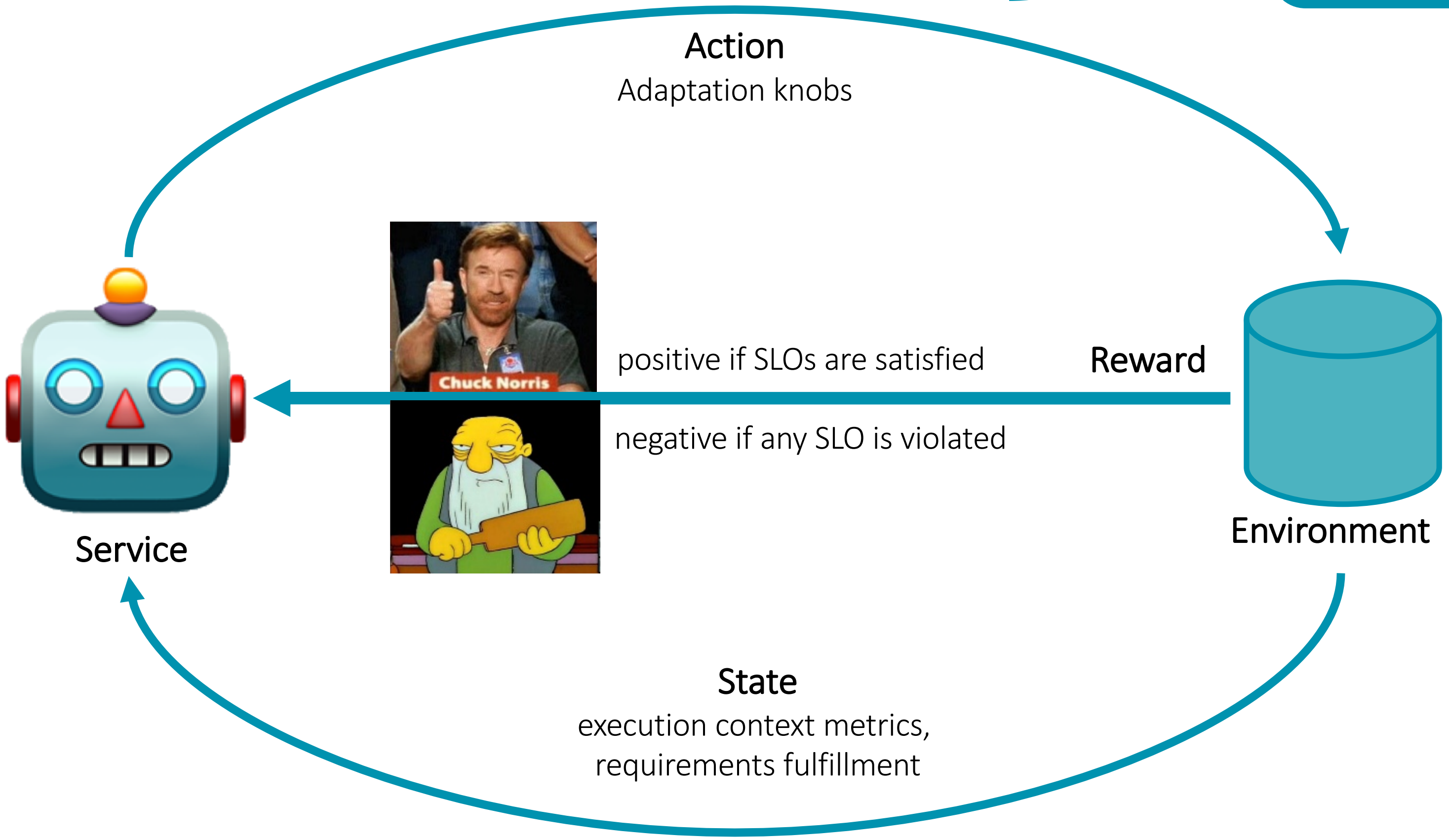

- Elements of RL
  - **Environment**: World in which the agent operates
  - **State**: Current situation of the agent
  - **Reward**: Feedback from the environment
  - **Policy**: Method to map agent's state to actions

Image from: https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

# Reinforcement Learning for Dynamic Orchestration

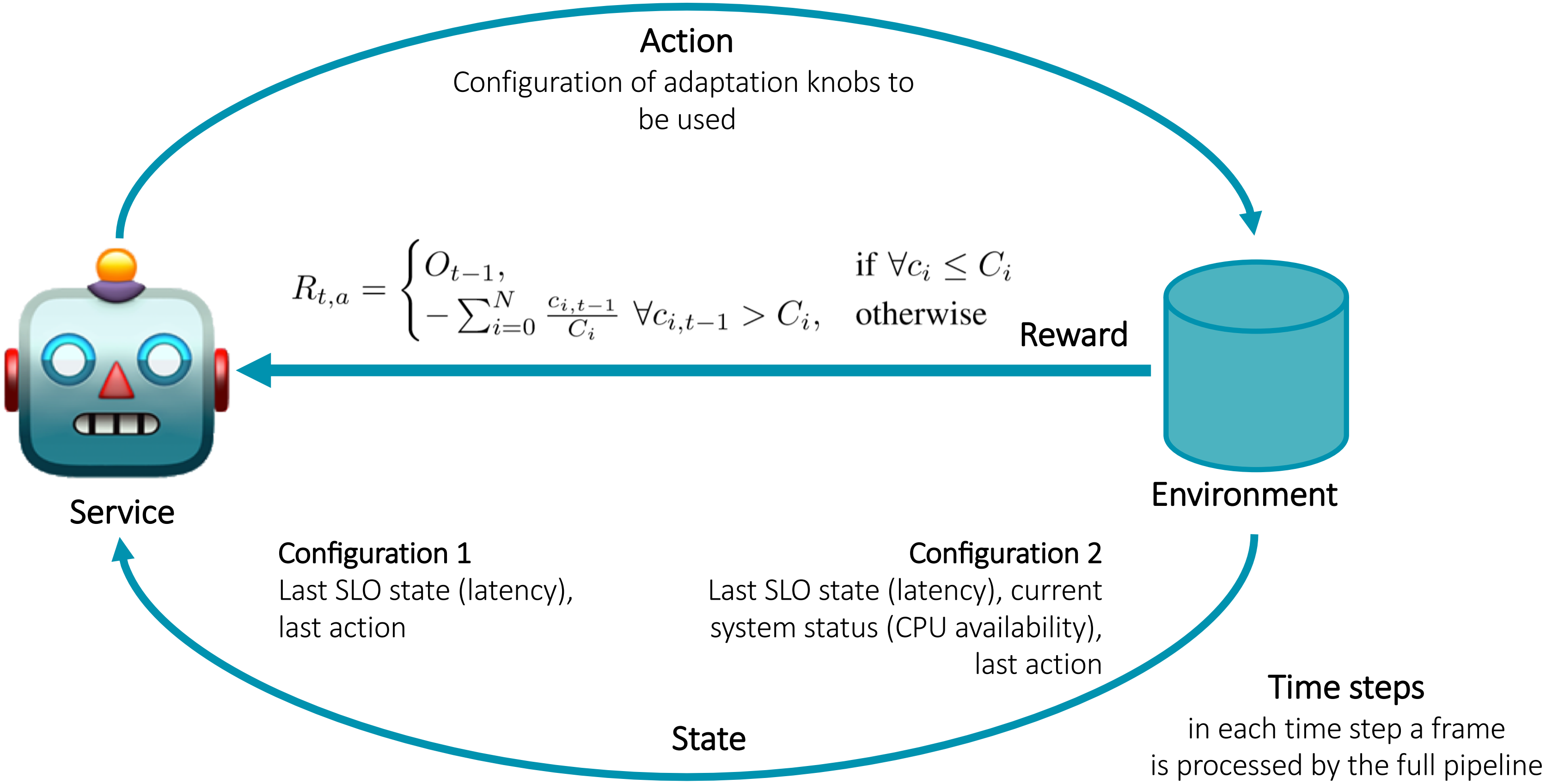
- RL is highly adaptable to different environments and goals, and learns ”on the go”
  - Good for highly heterogenous contexts
  - Good for different services with different requirements
  - Good for cases in which no previous information is available

**These are the characteristics of our problem!**



# Reinforcement Learning for Dynamic Orchestration

- We have implemented and evaluated two configurations that use different information




$$R_{t,a} = \begin{cases} O_{t-1}, & \text{if } \forall c_i \leq C_i \\ -\sum_{i=0}^{N} \frac{c_{i,t-1}}{C_i} \ \forall c_{i,t-1} > C_i, & \text{otherwise} \end{cases}$$

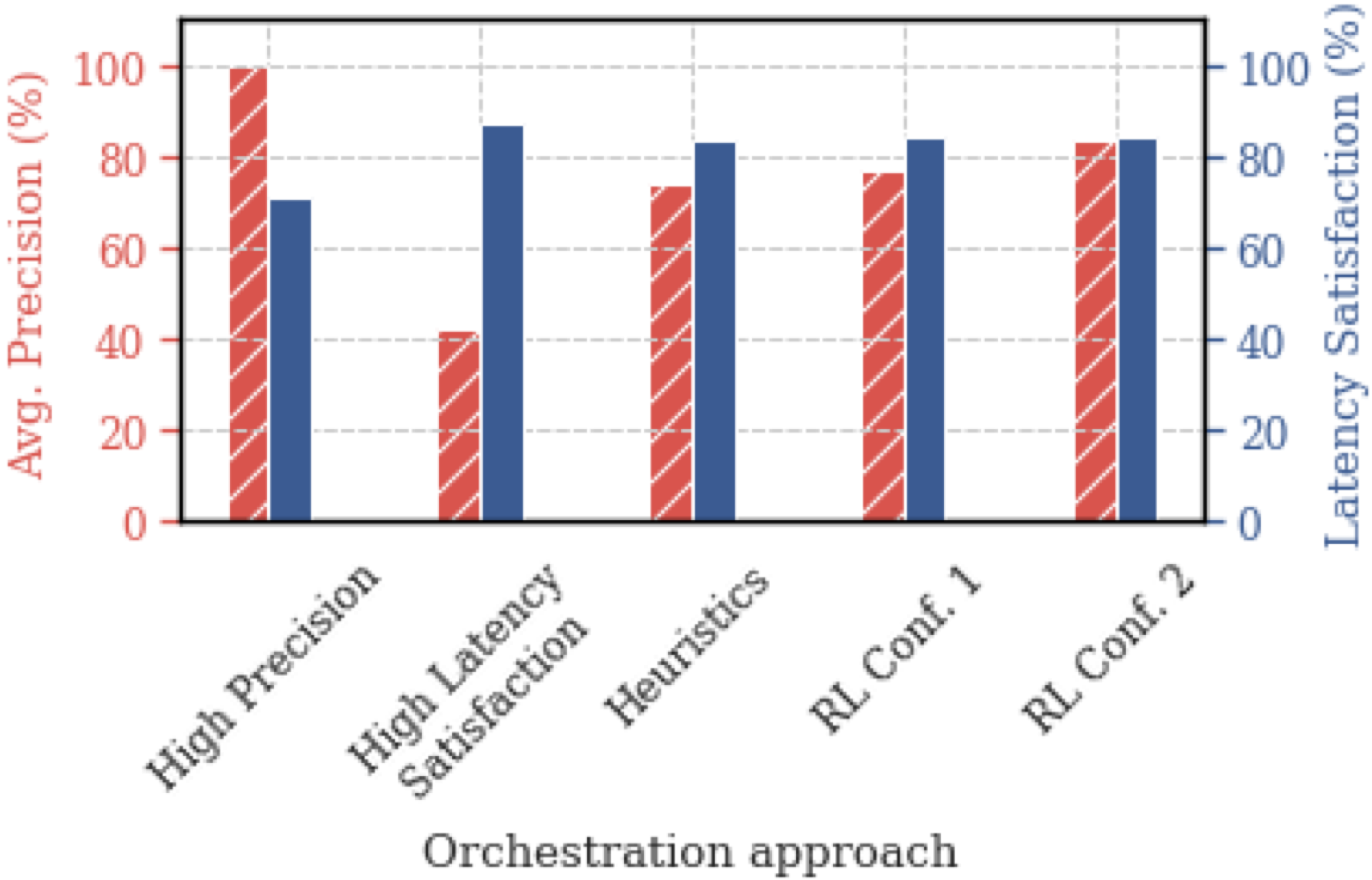
# Experimental Evaluation

- Simulation of Lost Child service (static and elastic) on Raspberry Pi 3B+


- Service runs in shared environment
- CPU availability varies in each step with probability p=0.1 between 0.3 and 1
- Using the dataset Faces94, images with 6, 12, 24, 48, 96 and 192 faces were composed

- Different datasets
  - Fixed input
  - Variable input
  - Full day input
  - Random input

# Results

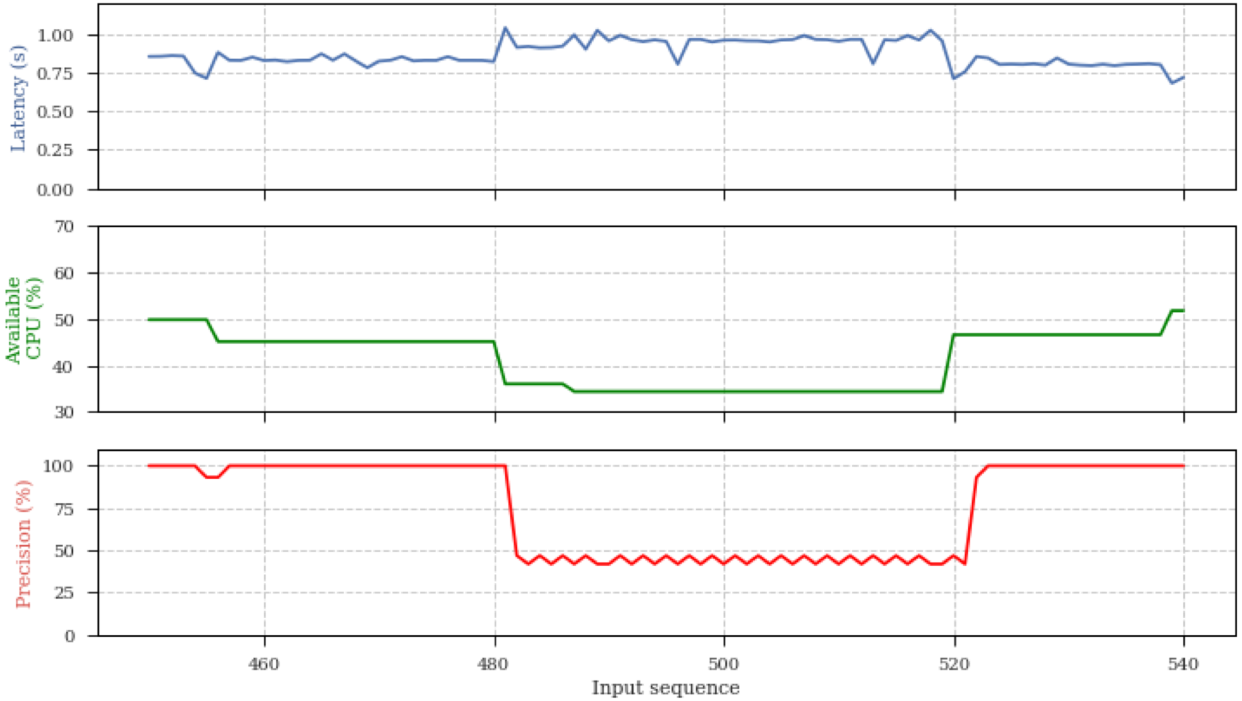


Avg. Precision (%)
100
80
60
40
20
0
Latency Satisfaction (%)
100
80
60
40
20
0
High Precision
High Latency Satisfaction
Heuristics
RL Conf. 1
RL Conf. 2
Orchestration approach

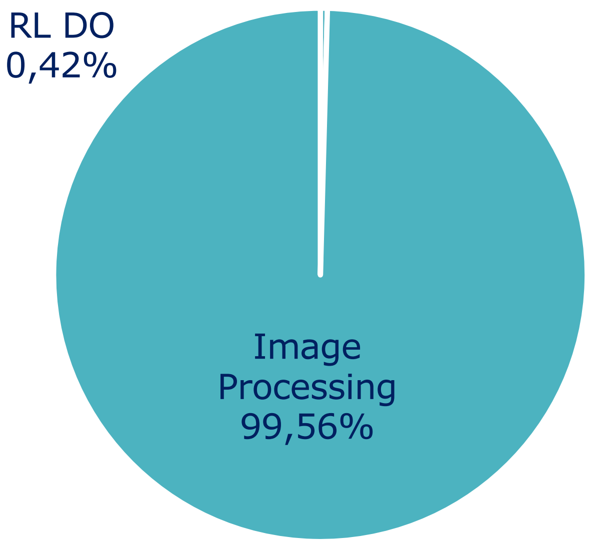
# Results (2)



- How well does the RL based Dynamic Orchestration perform according to its requirements?
- It reacts quickly to changes, avoiding requirements violations and taking advantage of resources when they're available
- It needs little processing power/time




**Low overhead**



# Takeaways and Future work

- Edge services need to adapt in order to meet their SLOs

- Elastic services simplify development of adaptive edge services

- Dynamic Orchestration achieves better performance than static services

- RL is an effective approach to orchestrate elastic services
  - 10-25% higher precision than heuristics
  - 25% less execution time than heuristics

- In the future, we want to improve RL based orchestration
  - Support large number of adaptation knobs (and values for each knob)
  - By further testing it with different services and environments

# Thank you for your attention!

Mauricio Fadel Argerich <mauricio.fadel@neclab.eu>

\Orchestrating a brighter world

NEC

BigDataStack
Holistic stack for big data applications and operations

The research leading to these results has received funding from the European Community's Horizon 2020 research and innovation programme under grant agreement nº 779747