AI-based Resource Management in Beyond 5G Cloud Native Environment

Abderrahmane Boudi *, Miloud Bagaa *, Poyhonen, Petteri§, Tarik Taleb*¶, Hannu Flinck§
* Aalto University, Finland, {abderrahmane.boudi, miloud.bagaa, tarik.taleb}@aalto.fi
§ Nokia Bell Labs, FI-02610 Espoo, Finland {petteri.poyhonen, hannu.flinck}@nokia-bell-labs.com
¶University of Oulu, 90570 Oulu, Finland; Sejong University, Seoul, Korea.

***Abstract*—5G system and beyond targets a gigantic number of emerging applications and services that will create an extra overhead on the network traffic. Moreover, these industrial verticals have aggressive, contentious, and conflicting requirements that make the network have an arduous mission for achieving the desired objectives. It is expected to get requirements with close to zero time latency, high data rate, and network reliability. Fortunately, a ray of hope comes shining the way of telecom providers with the new progress and achievements in machine learning, cloud computing, micro-services, and the ETSI Zero-touch network and Service Management (ZSM) era. For this reason there is a colossal impetus from industries and academia towards applying these techniques by creating a new concept called Cognitive Cloud Native environment that can cohabit and adapt according to the network and resource state, and perceived Key Performance Indicators (KPIs). In this paper, we pursue the aforementioned concept by providing a unified hierarchical closed-loop network and service management framework that can meet the desired objectives. Also, we have proposed a cloud-native simulator that accurately mimics cloud-native environments, and enables us to quickly evaluate new frameworks and ideas. The simulation results demonstrate the efficiency of our simulator for parroting the real testbeds in various metrics.**

I. Introduction

Hitherto, there is an impressive growth of network data traffic and progressive digital transformation of industry and society. This stresses the network capacity and pushes it to the limits by creating issues related to network and resource capacity, security, resiliency, reliability, and privacy treads. From another side, the 5G system and beyond targets an immense number of emerging verticals and applications that will stress even more the network traffic. In contrast to the 4G LTE system, 5G envisioned close to zero time latency, higher data rate with ten times more, five-nines network availability, and a more dense network with 10 - 100 more connected users and devices [1]. It is expected to observe more challenges in upcoming years related to the expected exponential increase of traffic and ever-growing network complexity and heterogeneity. On another side, the KPIs of different services and applications become more demanding, which makes their satisfaction an extra overhead on the network and telecom providers.

Fortunately, a ray of hope comes shining telecom providers' way with the new progress and achievements in Artificial Intelligence (AI), cloud computing, and micro-services era. The new achievements enable the development of a cognitive network that can cohabit and adapt according to the network and vertical changes. Shifting to the micro-service concept by telecom (*i.e.*, 5G service-based architecture [2]) creates a new concept dubbed Cloud Native Environment (CNE) that can adapt according to the network and vertical needs by favoring horizontal than vertical scaling. Moreover, the service granularity created thanks to the micro-service concept, leads to:

- **Increase the elasticity and reduce the Operating Expenses (OPEX):** This concept gives high freedom of adaptation according to the network and resources changes, and thus an atomic service provisioning that reduces the cost and increases the flexibility. It gives the possibility of deploying services using the right amount of replicates and resources to reduce the OPEX and enhance the elasticity. The resources would be used according to their needs which prevents the over-provisioning of services;
- **Increase the resiliency and the reliability:** This approach enhances the network resiliency and reliability by enabling the deployment of multiple micro-service replicates. This gives the network the capacity to maintain the service level even with failures in some replicates. Moreover, the cognitive network can automatically adapt to network failures by deploying more replicates, which does not consume too many resources, if needed. From another side, in the presence of attacks, legitimate services could be easily shifted and moved towards safe locations to ensure the service continuity and prevent the attacks from spreading. Meanwhile, different mitigation mechanisms would be applied in a locked-down environment for alleviating the attacks;
- **Ensure the expected KPIs and enhance the Quality of Experience (QoE):** This approach enables the deployment of services with fine granularity for handling few objects or users, which increases the likelihood for deploying them on the proximity of users and hence increased bandwidth, and reduced latency will be perceived. Moreover, the use of micro-services enables the smooth services relocation across Multi-access Edge Computing (MEC) and clouds, with limited computational resource and network bandwidth utilization, following user and object motion, and hence ensures the targeted KPIs and enhance the perceived QoE.

Besides the micro-service concept, technology enablers and

concepts, such as MEC, Software-Defined Networking (SDN), Network Functions Virtualization (NFV), and ETSI ZSM [3] will play crucial roles in the 5G system and beyond to mitigate the challenges above and achieve the desired objectives and KPIs. Thanks to these techniques and technologies, the elasticity, resiliency, reliability and QoE will be increased, the OPEX will be decreased, and the KPIs ensured. In 5G system and beyond, with the new move towards the cloud-native approach, there is a growing enthusiasm for using AI techniques and ETSI ZSM concept that will play a deeper and deeper role in the orchestration, control, and user planes. The CNE will be mainly executed on top of distributed data centers characterized by their resources and computational heterogeneity, and its geographical distribution could include private/public clouds and MECs. The CNE will leverage both AI and ZSM to attenuate the intensity of resource heterogeneity and geographical distribution by: First, managing, controlling, and orchestrating distributed resources and services across those clouds and MECs. Second, running services over a continuum of geographically distributed resources by pushing those services near the end-users while hiding the resources' heterogeneity. Third, cloud-native services that enable smooth service relocation among different sites in a seamless fashion.

In this paper, we present an Artificial Intelligence based Resource aware Orchestration (AIRO) framework in CNE. AIRO framework leverages ZSM concept, cloud-native approach, and Machine Learning (ML) techniques for efficiently managing network and computation resources. The contributions of this paper are manifold:

- A unified AIRO framework that is aligned with ETSI ZSM vision and leverages ML techniques for enabling closed-loop automation and autonomous CNE.
- A monitoring and management agent deployed alongside the master node at each cluster for creating a single management domain. The latter receives the high-level controls and commands generated from the End-to-End (E2E) management domain using domain integration fabric.
- A simulation platform that accurately mimics K8s microservices clouds, and thus enables to evaluate the scalability of AIRO framework and any other similar framework to be proposed in the future.

The remaining of the paper is organized as follow: Section II offers a background study and summarizes different related works presented in the literature. While Section III presents the AIRO framework functionalities and summarizes our cloud-native simulator, as well as the monitoring and management agent. Meanwhile, Section V shows the performance evaluations and the comparison between the real deployment and simulation performances. Finally, the paper is concluded in Section VI.

II. Background and related work

A. Background

ZSM can be considered as a natural extension of "system of systems" concept by integrating AI techniques for enabling self-orchestration and management. ZSM defines a unified management and orchestration framework for managing functions, services and resources in an autonomous and harmonized way. The managed resources include physical resources, such as Physical Network Functions (PNFs), Virtual Network Functions (VNFs) and/or cloud resources (*e.g.* "X-as-a-Service" (XaaS) resources). ZSM has modular architecture consisting of self-contained and loosely coupled services that communicate using intent-based interfaces. Such interfaces expose high-level abstraction in order to hide complexity, technology and vendor specific details. This helps for ensuring the flexibility and extendibility, and hence managing services (horizontal/vertical control scaling) and capabilities at different subsystems in a transparent manner. Moreover, this ZSM framework ensures both scalability and resiliency by adopting a decentralized approach and by giving more freedom to the local subsystem management for taking local decisions. ZSM ensures the self-management capability by leveraging AI techniques for enabling a fully automated system that prevents human intervention. ZSM architecture consists of two management parts: i) Management domains; ii) E2E service management domain. While there is a multiple entities from the former, there is only one entity from the latter.

Each management domain is considered as a loosely coupled entity, whereby its functions internally communicate using internal domain integration fabric. This fabric, besides ensuring the communication between management functions, it enables the registration, discovery and invocation of management services within the management domain. Conceptually, a management domain could be formed based on functional similarities (*e.g.*, Transport, Access Network, Core Network, etc.) or relationships such as services using the same set of infrastructure resources, types of infrastructure resources, technology and/or ownership. While the E2E service management domain, which is also a loosely coupled entity, ensures the cohesion and the unified management and interaction between the different management domains, thus it helps enable E2E services that meet the desired KPIs [4]. An external domain integration fabric is used to ensure the communication between various management domains, including E2E service management domain, the registration, the communication and the discovery functions. Internal and external domain integration fabrics use different service patterns including registration/discovery, synchronous (request-response) and asynchronous (publish-subscribe) interactions.

Thanks to AI techniques, closed-loop management and orchestration is ensured at the management domain level, as well as at the E2E service management domain level. As part of the closed-loops, the following functions are employed: Data collection, control, orchestration, intelligence, and analytics. In fact, the domain data collection service monitors all the entities, functions and services. Then accordingly, it provides feedback to the other functions (control, orchestration, intelligence, and analytics) in order to support the closed-loop automation. The domain control service allows to steer the states of managed entities and coordinate with the orchestration functions. While the orchestration functions enable the instantiation and maintenance of domain-level network services, including creation, modification and termination of the services, and they

allow the automation of corresponding workflows. Meanwhile, intelligence functions are responsible for driving the intelligent closed-loop automation by leveraging various AI techniques including Machine Learning (supervised, unsupervised and reinforcement learning). Finally, based on the feedback of data collection functions, the analytics functions provide domain specific predictions which enhance further the intelligent closed-loop automation.

Kubernetes is a container orchestration system that is used to manage containerized workloads and services. A Kubernetes cluster consists of (at least) one master and multiple worker nodes. The nodes run PODs which are the smallest deployable and managed units in the cluster. A POD typically contains multiple containers that provide a service and are co-located and co-scheduled. The Kubernetes cluster maintains a (logically) centralized storage which is used to store the cluster's state information and its configuration and that is also used for service discovery. One of the Kubernetes' system components is the scheduler and its main responsibility is to find the most suitable worker node where to run a POD. Therefore, it is responsible for service placement which is a very important topic in 5G networks [5]. The scheduler and other controllers are bound to work only for one cluster where the maximum number of nodes, preferably, does not go beyond 5000 nodes. Also, the scheduler policy is fixed, it is manually set by the operator. In 5G and beyond systems, the deployments are heading toward multi-cluster setups with a high-level of automation. The scheduling mechanisms would need to cope with the high number of possible configurations that comes from the increased number of nodes and the increasing users requirements. Following the ZSM framework, implementing hierarchical AI driven systems would alleviates some of the issues induced by multi-cluster setups.

B. Related Work

A lot of efforts have been put into applying the cloud-native principles in the 5G mobile system. In [6], the authors introduced Next Generation Platform as a Service (NGPaaS) project. The main idea behind NGPaaS is to remove the barrier between the telecommunication and IT industries. In one scenario, the telecommunication industry will enable Content Providers to place VNFs into the networks that connect their data centers and their customers. Other works have applied the cloud-native principles to 5G systems [7], [8]. In [8], the authors provided a proof of concept for 5G network slicing. They proposed to replace large monolithic network entities with Service Function Chains (SFCs) that are made up of several lightweight VNFs. The SFCs would represent an abstraction of a network function or capability, such as network authentication or packet scheduling. While [7] proposes to shift Virtual Machine (VM) based VNFs toward container based ones to achieve more lightweight virtualization. This approach, called native-cloud VNF, is meant to provide 5GaaS.

In current 5G systems, the exhibited scale and heterogeneity of resources and users demands results in further complex systems with a huge space of possible settings. Indeed, in a scenario of task scheduling that have multi-dimensional resource requirements, coupled with a telco-grade system, the scheduling problem would prove to be too complex. Using traditional ad-hoc scheduling mechanisms may underperform in such complex systems. Therefore, ML techniques are being widely adopted to replace heuristic approaches. Due to the large set of parameters, the use of AI techniques is becoming of utmost importance. With ML techniques, it is possible for the scheduling framework to automatically understand both of the workload and the environment [9]. A careful design of the ML algorithm would ensure a robust scheduling in heterogeneous changing environments while following the performance guidelines set by the operator. [10] introduced a Deep Learning (DL) scheduler that is specifically aimed for DL clusters where the main objective is the average job completion time. It is worth noticing that DL scheduler deals only with the resource allocation problem, which means that the resources locations are irrelevant in such scenarios. Another interesting use case for AI techniques, is smart monitoring. Given the sensitivity of Ultra-Reliable Low-Latency Communication (URLLC) type applications and the high number of devices in massive Machine Type Communications (mMTC), the monitoring demands of 5G networks are quite pervasive [11]. In this context, ML methods can greatly reduce the size of transferred monitoring data by using ML-based methods for intelligence extraction close to the source of the data.

One pillar of 5G networks is edge computing. It consists in pushing the computing resources toward the edge of the network where they are closer to the end-user. The closer the edge cluster to the end-user is, the better QoE perceived by the end-user becomes. It is also clear that with edge computing the resources would be distributed in a large geographical area. Considering these properties, the VM placement problem would become fundamental to the good functioning of the network. [5] have provided a comprehensive list of several solutions of the VM placement problem. They categorized the solutions in regards to energy consumption minimization, cost optimization, Quality of Service (QoS), resource usage, reliability, and load balancing. In [12], the authors have provided a comprehensive view about the interplay between Deep Learning and the edge. They show how DL techniques are leveraged to optimize the edge and how the edge can be optimized to better support DL based services. Finally, authors in [13] have leveraged machine learning techniques for enhancing the NFV Management and Orchestration (MANO) platform. A bottom-up micro-functionality approach has been presented that leverages reinforcement learning and federated learning techniques. In contrast to the mentioned solution, AIRO framework leverages deep reinforcement learning techniques for providing closed-loop automation and autonomous CNE.

III. Artificial Intelligence based Resource aware Orchestration

In this section, we present our AIRO framework that aims to leverage ML techniques for orchestrating network and computing resources in CNE. The AIRO framework consists

of two parts, which are E2E service management domain, and single service management domains. Finally, we conclude this section by presenting our cloud-native simulator that accurately mimics real deployment of cloud-native cluster (*i.e.*, K8s).

A. AIRO Framework: General Overview and E2E Service Management Domain

Figure 1 depicts a general overview of AIRO framework targeted by this paper to create Cognitive Cloud Native. The AIRO framework leverages both ZSM and ML for creating self-orchestrated and self-optimized CNEs that are able to cohabit and adapt according to the network state and industrial verticals' KPIs. AIRO mainly consists of two planes, which are i) The orchestration and management planes that present E2E service management domain; ii) The management domain that presents the user plane. The latter consists of a set of clouds and edges, whereby different vertical micro-services and applications run. For instance, K8s and K3s can host services at the cloud and edge level, respectively. While the communication components within the edge and cloud present the internal domain integration fabric, the "Monitoring Bus" and "Command & Control Bus" present external domain integration fabric.

In AIRO framework, we have two resource orchestration and management levels that aim to optimize the deployment of the services. The first level consists of E2E service management domain that has the global vision and information about various verticals, CNE clusters states, network and computational resources, as well as running services. AIRO framework has two scheduling levels; the first for E2E service management domain and the second for CNE cluster.

In the first scheduling level, different services and POD intents would be generated and queued up to serve various verticals. Then, according to the vertical state (*e.g.*, user location), clusters states, and network and computational resources, the AI based Orchestrator (*i.e.*, the first scheduling level) decides at which time and on which CNE cluster the PODs and services should be created. In order to take the right decisions, the AI based Orchestrator would employ various AI techniques including deep learning and deep reinforcement learning techniques, such as value-based (*e.g.*, DQN, DDQN, and Duel-DDQN) and policy-based (*e.g.*, A2C, A3C, DDPG, and PPO) approaches. The latter offers the ability to learn from the environment, and then accordingly offer the ability to cohabit and adapt according to the KPIs and resources state. The AI based Orchestrator requires two inputs. While the first input consists of the target KPIs per vertical and network slice, the second one consists of the aggregate logging network state that includes information about resources in different clusters and perceived KPIs. Based on the observation that in a closed-loop management system, the AI based Orchestrator deals with continuous time series data. Thus, there is a need for data pre-processing that includes data cleaning, integration, smoothing and reduction. For this purpose, the "Monitoring Aggregator" entity has been suggested. In order to remove the noise in this module, different data pre-processing strategies and methods would be applied including dimensionality reduction, discrete Fourier transform, numerosity reduction, and discrete wavelet transform.

Meanwhile in the second scheduling level, the PODs will be instantiated at the single management domain (cloud-native cluster) from the received POD intents. In each single management domain, there is a Monitoring Agent (MA) that is responsible to gather local information related to network and computation resources and applications, such as QoE, E2E delay and bandwidth, to ensure the desired KPIs. In each cloud-native cluster (*i.e.*, K8s or K3s), the required services (*i.e.*, POD intents) are deployed as PODs and exposed them to the outside world. Besides the master node, there is a management agent that exposes REpresentational State Transfer (REST) API and is able to apply different strategies for selecting the appropriate worker node for each POD deployment. Therefore, according to the general guidelines received from the E2E service management domain (*i.e.*, AI based Orchestrator) through the REST API, the cloud-native management agent will deploy PODs in various worker nodes to meet the desired KPIs while preventing service over-provisioning and resources under utilization.

B. AIRO Framework: Single Service Management Domain

In this section, we detail the scheduling process at each single management domain (*i.e.*, K8s and K3s cluster). The latter also offers close-loop self-management automation platform to deploy received POD intents. In vanilla K8s and K3s cluster, the PODs are scheduled and deployed in a predefined order and using two cycles, which are the scheduling and the binding cycles. While the appropriate worker nodes are selected for hosting different PODs during the scheduling cycle, the PODs are actually deployed during the binding cycle. In the scheduling process, the PODs that fail to be scheduled should get back to the initial queue, and then considered in the next epochs. The deployment of the POD would be aborted after a given threshold of scheduling failures. We have enhanced the single management domain to ensure the close-loop self-management automation by adding two new modules, which are i) MA, and ii) management agent. The former gathers logging information within that cluster and reports them to the latter and to "Monitoring Aggregator" modules. Meanwhile, the management agent leverages the outputs of the MA for updating the internal strategies of PODs scheduling to reduce and prevent deployment failures and to ensure the satisfaction of desired KPIs. Similar to E2E service management domain, AI techniques including deep learning and deep reinforcement learning techniques would be explored in the future for providing the required strategies. Taking the decisions at the cluster level helps AIRO framework to be a hierarchical, an extensible and a scalable platform by lightweighting the overhead on the E2E service management domain.

Figure 2 depicts a detailed communication diagram between the two developed modules and vanilla K8s and K3s clusters. With the help of these new communication links, it becomes

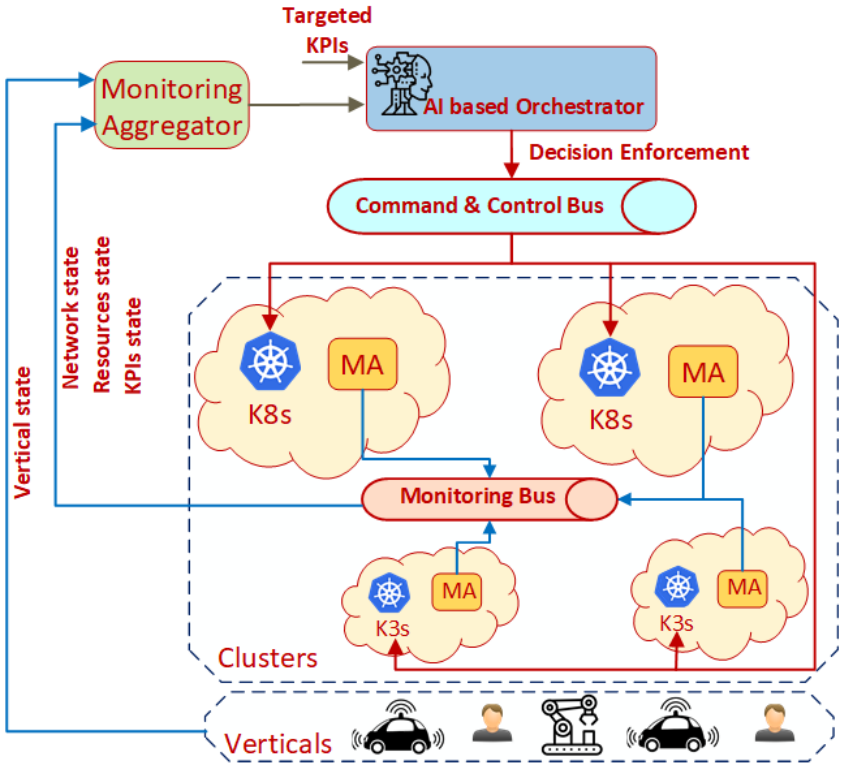


Figure 1: Main idea of AIRO framework.

possible to exploit AI based techniques together with ML to assist the K8s cluster's workload scheduling. In such system, both K8s and external entity have specific well-defined roles based on their interactions that are carried out via the specified Application Programming Interfaces (APIs). These roles and the respective APIs are depicted in Figure 2. The external entity (from the K8s cluster's point of view) called AI/ML logic provides guidance to the K8s scheduler on how to evaluate candidate nodes (workers). This guidance consists of mathematical formulas and the related KPIs based on which a rank value (see Scoring in [14]) for each candidate node is calculated. In this way, the AI/ML logic is continuously maintaining up-to-date guidance information in the K8s scheduler so that at any moment, a new workload scheduling can be done accurately.

This approach does not slow down the scheduling cycle by introducing external entities in the scheduling loop, since any communication (synchronous or asynchronous) to fetch external information for the scheduler's decision making phase may delay the execution of the internal scheduling functionalities. The feedback for the AI/ML logic is achieved in two steps; i) the results of the scheduler's decision making are communicated back (Report) and ii) the cluster's conditions are actively monitored (Observe). This feedback is then used to further refine the AI/ML logic's "knowledge" affecting to the related ML among other things. All this is then used to update the guidance to the scheduler.

Additionally, this kind of system enables a single AI/ML logic to govern multiple clusters that are operating autonomously, *i.e.*, there are no federation or co-operation

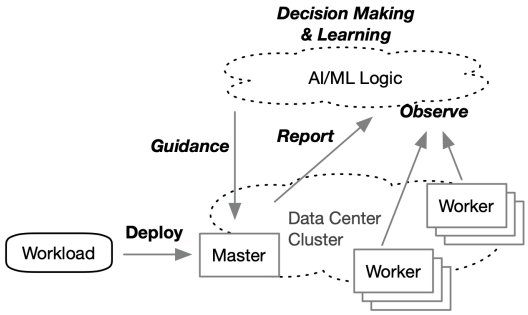


Figure 2: Communication system overview.

between clusters. However, even if clusters operate autonomously, certain management actions could be implemented in a distributed fashion resulting in a federated management operation on top of the clusters at the AI/ML logic level. For instance, federated scheduling can be achieved by having an external entity in the scheduling loop, even if the scheduling process becomes slower than the autonomous approach, the advantages of such distributed operation could far outweigh the slower execution times.

Cluster operators are interested in driving the cluster towards some equilibrium of multiple objectives. These objectives represent operator's incentives and goals, and one of the main incentives is to reduce costs and increase revenue. In what follows we shall present some objectives that the operators may chase:

- **Utilization and load distribution among nodes:** Distributing the load among the nodes consists in keeping a low utilization across all nodes. In fact the euclidean

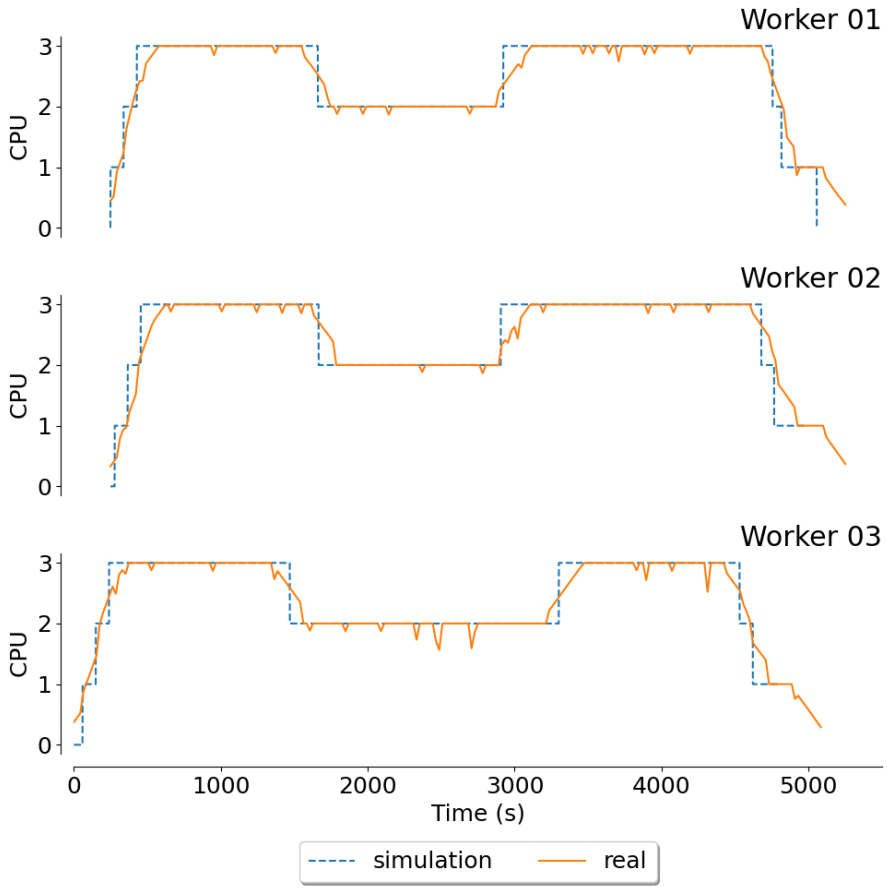


Figure 3: CPU consumption per node.

distance of the percentage of resource utilization among nodes should be reduced as much as possible. This helps maximizing resource utilization while mitigating and reducing points of congestion. The accurate load distribution ensures a good distribution of network traffic which may result in a positive impact on the QoS and QoE.

- **Reduce operation costs:** In a scenario where nodes have different running costs, it may be useful to shift load from nodes with high running costs to nodes with smaller running costs.
- **Reduce energy consumption:** Likewise, this objective may help the operator in choosing nodes with small energy footprints. It can also clear some nodes from all the PODs so they can be turned off.
- **Service satisfaction:** In this instance, the objective is to maximize the service satisfaction. In other words, regardless of the cluster and node status, the best node would be the one that satisfies all the preferences of the service.
- **Selecting healthy nodes:** We may consider the nodes that have less probability of failure. Such a probability can be calculated from the current load exerted on the node and its history of failures. This objective will help increase the reliability of deployed services.

C. Cloud-native Simulator

This section addresses the challenges faced in the deployment of AIRO framework. One of the main challenges is how to support ML for the new clusters in such way that a newly created cluster has sufficient knowledge to carry out intelligent decisions making without needing from scratch training of the ML. One way would require us to build a set of pre-trained agents that can be injected into a new system. The injected agents will help the system to quickly converge and to take the right decisions in a short period. In order to create such pre-trained agents for a system following the proposed framework (Section III), a simulator was developed to provide training to ML agents in pre-configured cluster environments.

Given the nature of deep reinforcement learning algorithms where the agent learns from scratch, using real environments as a playground for the agent would result in multiple issues. Real environments evolve in real time while simulated ones can be accelerated. Moreover, multiple instances can run in parallel which will greatly reduce the time needed to train the agent. Real environments can be irremediably harmed due to a bad behavior of the agent. Using simulated environments would help the agent to learn in widely different configurations, while real environments are constrained with available resources; in other words, simulation is cost effective. Last but not least, simulated environments are energy efficient, which represent a growing concern in the field of Reinforcement

Learning. The simulated environment is meant to jump start the agent learning, it is not meant to replace the real environments. Indeed, after finding a decent policy, the agent needs to continue learning once it is deployed in a real environment. On the downside of using simulated environments is the fact that the simulation has to be sufficiently accurate. When simulated and real environments greatly differ, the agent will, in the best case, underperform when deployed and it can even be useless or harm the real system in the worst case scenario.

Given the reasons mentioned above, in what follows, we shall introduce our simulation testbed of a Kubernetes cluster. In our case study, there are four main components that defines a Kubernetes cluster: a) a set of PODs, b) a set of Nodes, c) a scheduling mechanism, and d) a statistics/reporter module. A POD can be characterized by its actual resources consumption, its requested/claimed resources, and the upper limits that it cannot go beyond. It can also specify some requirements and preferences on which kind of node it should be deployed on. A Node is characterized mainly by how much resources it contains and it can specify other characteristics such as type of node, location, or cost. The node should also keep track of the PODs that are deployed on top of it. The scheduling mechanism responsibility is to place new PODs into a node. Finally, the statistics/reporter module is used to extract the status of the cluster. It will act as a monitoring system that will help gather all the necessary information needed by Reinforcement Learning (RL) agents. The cluster also needs to keep track of deployed PODs, especially when there are multiple replicas of the same POD running in different nodes.

The lifetime of a POD is very short compared to the lifetime of the Kubernetes cluster. Therefore, the Kubernetes cluster will witness many arrivals and departures of PODs. Nodes can also be added and deleted dynamically from a Kubernetes cluster. It is clear then that the simulator should provide a sense of the passing of time. In order to do this, we opted to implement a discrete time simulator with which the lifecycle of the different components of the Kubernetes cluster would be managed. The time module is used to model the arrival and departure processes of the PODs and to model the resources consumption changes during the lifetime of a POD. It can also be used to generate random events, such as the failure of a POD or a node, and it can also be used to dynamically change the load exerted on the cluster.

With this simple design, the simulator can easily be expanded to support new use cases. In order to alleviate the problem of the difference between the real and simulated environments, this simulator will be updated to reflect the insights gained when deploying the learned agent in real testbeds.

IV. Cloud native environment use-cases and scenarios

In densely crowded environments, a large number of Internet connectivity demands originate from a small geographical area. Under such circumstances, it is not always possible to satisfy such demands. Usually, to alleviate this problem, users are grouped into clusters that share computational and storage resources [15]. The purpose of such clustering is to satisfy the demands of the users inside that cluster. Therefore, the number of users inside a cluster and the number of clusters varies greatly. Given this high variability in network topology, having independent and self-managed clusters is of utmost importance. In this regard, the AIRO framework would permit the hierarchical management of such clusters.

Many verticals will benefit from the AIRO framework thanks to the intertwining between the flexibility of micro-services and the smartness provided by the artificial intelligence. Self-driving vehicles are next-to-come, multiple car manufacturers around the globe are pursuing this vein. Such vehicles need a reliable network with stringent latency requirements. This is why the interplay between vehicles and Roadside Units (RSUs) plays a vital role in the functioning of these vehicles. With AIRO framework, multiple AI managed clusters can collaborate to satisfy the requirements of self-driving vehicles. Indeed, with the help of the E2E service management domain, the AI agents can collaborate, for instance, on the placement and relocation of VNFs problem [16].

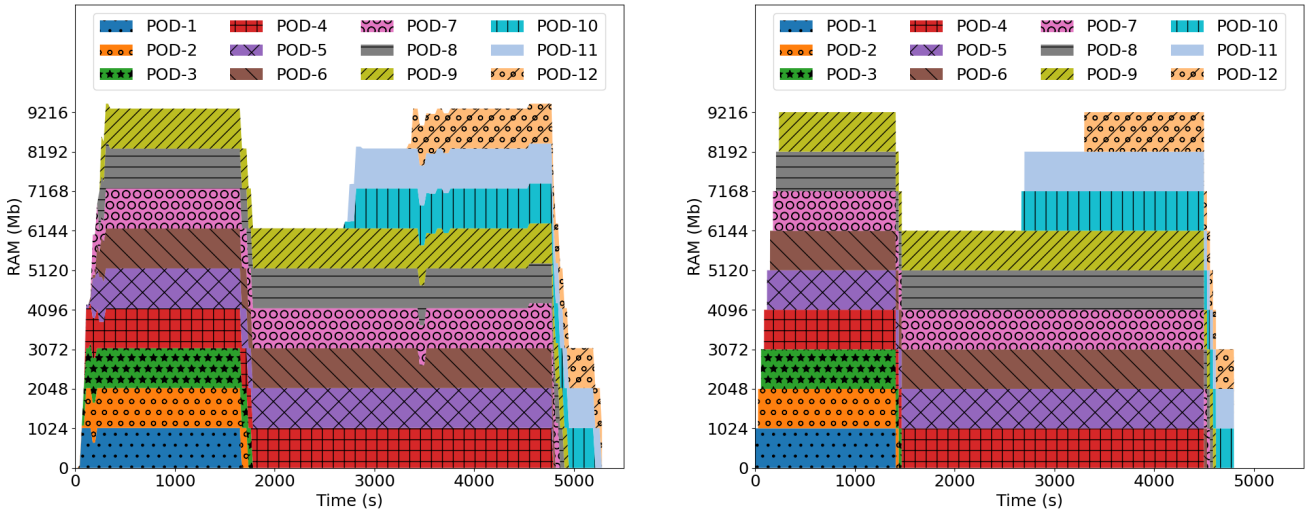
From another side, object detection, and recognition would play a crucial role in future use-cases and applications varying from self-driving cars to augmented reality. Unfortunately, object detection and recognition is a time and resource-consuming process. The problem heavily lies in the number of images that should be treated to track moving objects. AIRO framework will play a crucial role in enabling these applications by splitting the jobs smartly into different micro-services and running them at the appropriate locations for achieving the desired QoE.

Last but not least, the AIRO framework will play a pivotal role in enabling smart home applications. Soon, it is expected to observe a massive amount of Internet of Things (IoT) devices used everywhere for managing people surroundings [17]. These IoT devices would generate tremendous uplink traffic that should be treated efficiently and close to their source generation. Moreover, placing the services closer to the users would have a positive impact on the perceived QoE. By leveraging ML techniques, the AIRO framework is able to identify and deploy the required micr-oservices at the clouds that are close to the user.

V. Performance evaluation

Using a simulator instead of using a real deployment would greatly reduce the time needed by the agent to learn a good policy. However, the simulator should show a similar behavior to the real deployment. In what follows, a performance evaluation is carried out to show how close our cloud-native simulator is to the real deployment.

The experiment testbed comprises three worker nodes and one master node. Each node has 4 CPU and $4Gib$ of memory. Prometheus is used to monitor the load exerted on the cluster. In order to study the cluster's behavior, dummy PODs are launched. These dummy PODs do random calculation and write/read random values to/from the memory. Therefore, PODs do not only reserve, but actually consume computation



(a) Real testbed

(b) Simulated testbed

Figure 4: Memory consumption per POD.

and memory resources. The PODs run a containerized version of stress-ng tool, which is used to stress test computers.

The main idea behind this performance evaluation study is to deploy the dummy PODs into K8s and study how resource consumption evolves with respect to time. In this experiment, all PODs are configured to use 1 CPU and and $1Gib$ of RAM. In order to reduce the effect due to system and Prometheus PODs, 1 CPU and $1Gib$ of RAM are reserved in each worker node. Consequently, nine PODs can be fitted into the testbed. The experiment scenario is scripted as follows:

1) $POD_{1..9}$ are launched with a $30s$ interval between each two launches
2) After 20 minutes, three PODs are terminated, one from each worker node.
3) After 20 minutes, POD_{10} and POD_{11} are launched.
4) After 10 minutes, POD_{12} is launched.
5) After 20 minutes, the remaining six pods from $POD_{1..9}$ are terminated.
6) After 5 minutes, all remaining PODs (*i.e.*, $POD_{10..12}$) are terminated.

Figure 3 shows CPU consumption per node. It is clear that the evolution of CPU consumption in the real and the simulated testbeds is quite similar. There are, though, two main differences between the two testbeds. The real testbed shows small variation of CPU consumption through time and it also shows small delays when PODs are launched and somewhat longer delays when they are terminated. While in the simulated testbed, the CPU consumption is stable and the changes are instantaneous. These same conclusions hold true for memory utilization. With the exception that the variation in memory utilization is less pronounced than the variation in CPU utilization.

Figure 4 shows memory consumption for each POD. From Figure 4(a) and Figure 4(b), it is clear that the real and simulated testbeds are almost identical. The only notable difference between the two is that the real testbed can show random behavior such as it is shown by POD_6 and POD_9 between $3500s$ and $4000s$. Likewise, CPU utilization of PODs in both testbed is quite similar. Similar to what was shown at the node level, at the pod level, the real testbed shows some noisy behavior compared to the simulated testbed.

This experiment shows that when it comes to CPU and memory utilization, the real and the simulated testbeds are quite similar. Therefore, the small variation in resources consumption should not have an impact on the scheduling decision. Also, it is indeed quite straightforward to add noise in resource consumption for each POD.

VI. Conclusion

Leveraging ZSM concept, cloud-native approach and ML solutions, AIRO framework has been proposed. The AIRO framework offers a solution for telco-grade system resource management and service satisfaction enforcement. It proposes to alleviate the scheduling and placement problem in very large system where the number of possible configurations is huge. In order to evaluate and quicken development of the AIRO framework, a simulation platform has been introduced. This simulation platform mimics the behavior of a real deployment of a cloud-native cluster. Evaluation results have shown that the behavior of a cluster can be approximated with a small degree of uncertainty which is due to the stochastic behavior of computing systems.

Acknowledgment

The research leading to these results received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°101016509 (project CHARITY). The paper reflects only the authors' views. The Commission is not responsible for any use that may be made of the information it contains. The research work is also partially funded by the Academy of Finland 6Genesis project under Grant No. 318927, and by the Academy of Finland CSN project under Grant No. 311654.

REFERENCES

[1] 5G-PPP Architecture WG, "View on 5G Architecture," 5G-PPP, Tech. Rep., July 2017, v2.0.

[2] 3GPP, "System Architecture for the 5G System," 3GPP, Technical Specification (TS) 23.501, Jun 2018, v15.2.0.

[3] ETSI GS ZSM 002, "Zero-touch network and Service Management (ZSM): Reference Architecture," ETSI, Tech. Rep., Aug 2019, v1.1.1.

[4] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa, and A. Ksentini, "Network Slicing-Based Customization of 5G Mobile Services," *IEEE Network*, vol. 33, no. 5, pp. 134–141, 2019.

[5] A. Laghrissi and T. Taleb, "A Survey on the Placement of Virtual Resources and Virtual Network Functions," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1409–1434, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8556457/

[6] A. Mimidis, E. Ollora, J. Soler, S. Bessem, L. Roullet, S. Van Rossem, S. Pinneterre, M. Paolino, D. Raho, X. Du, J. Chesterfield, M. Flouris, L. Mariani, O. Riganelli, M. Mobilio, A. Ramos, I. Labrador, A. Broadbent, P. Veitch, and M. Zembra, "The Next Generation Platform as a Service Cloudifying Service Deployments in Telco-Operators Infrastructure," in *Proceedings of the 25th International Conference on Telecommunications (ICT 2018)*. IEEE, 2018, pp. 399–404.

[7] S. Imadali and A. Bousselmi, "Cloud Native 5G Virtual Network Functions: Design Principles and Use Cases," in *2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2)*. IEEE Computer Society, Nov 2018, pp. 91–96. [Online]. Available: https://ieeexplore.ieee.org/document/8567377/

[8] S. Sharma, R. Miller, and A. Francini, "A Cloud-Native Approach to 5G Network Slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 120–127, Aug 2017.

[9] R. Yang, X. Ouyang, Y. Chen, P. Townend, and J. Xu, "Intelligent Resource Scheduling at Scale: A Machine Learning Perspective," in *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. IEEE, Mar 2018, pp. 132–141. [Online]. Available: https://ieeexplore.ieee.org/document/8359158/

[10] Y. Peng, Y. Bao, Y. Chen, C. Wu, C. Meng, and W. Lin, "DL2: A Deep Learning-driven Scheduler for Deep Learning Clusters," 2019. [Online]. Available: http://arxiv.org/abs/1909.06040

[11] Y. Tseng, G. Aravinthan, B. Berde, S. Imadaliz, D. Houatra, and H. Qiu, "Re-Think Monitoring Services for 5G Network: Challenges and Perspectives," in *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, Jun 2019, pp. 34–39. [Online]. Available: https://ieeexplore.ieee.org/document/8854046/

[12] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8976180/

[13] D. M. Manias and A. Shami, "The Need for Advanced Intelligence in NFV Management and Orchestration," *IEEE Network*, to appear.

[14] Kubernetes.io, *Kubernetes Documentation v1.18 - Kubernetes Scheduler*, 2020 (accessed May 6, 2020). [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/

[15] M. Aloqaily, I. Al Ridhawi, H. B. Salameh, and Y. Jararweh, "Data and Service Management in Densely Crowded Environments: Challenges, Opportunities, and Recent Developments," *IEEE Communications Magazine*, vol. 57, no. 4, pp. 81–87, 2019.

[16] B. E. Mada, M. Bagaa, T. Taleb, and H. Flinck, "Latency-aware Service Placement and Live Migrations in 5G and Beyond Mobile Systems," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[17] A. Zarca, M. Bagaa, J. B. Bernabe, T. Taleb, and A. Skarmeta, "Semantic-Aware Security Orchestration in SDN/NFV-Enabled IoT Systems," *Sensors*, vol. 20, p. 3622, 06 2020.

Abderrahmane Boudi received the M.Sc. and Ph.D. degrees from the Higher National School of Computer Science, Algiers, Algeria, in 2013 and 2020 respectively. He was a visiting student of MOSA!C Lab/AALTO in 2017 and he has worked there as a research assistant in 2020. His research interests include congestion control in computer networks, intelligent feedback control systems, SDN, and 5G and beyond networks.

Miloud Bagaa received the bachelors, masters, and Ph.D. degrees from the University of Science and Technology, Houari Boumediene Algiers, Algeria, in 2005, 2008, and 2014, respectively. He is currently a senior researcher with the Communications and Networking Department, Aalto University. His research interests include wireless sensor networks, the Internet of Things, 5G wireless communication, security, and networking modeling.

Petteri Pöyhönen received the B.Sc., M.Sc. and Ph.D. degrees in computer science from the University of Central Lancashire (United Kingdom) in 1995, from the University of Kuopio (Finland) in 1996, and from the University of Helsinki (Finland) in 2012 respectively. He joined Nokia in 1996 and continues his work in Nokia Bell Labs as a senior research specialist in Espoo, Finland. He has been participating in a number of EU-projects. His research interests include 5G and beyond mobile networks, multi-access edge computing, and cloud computing.

Tarik Taleb received the B.E. degree (with distinction) in information engineering in 2001, and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2003, and 2005, respectively. He is currently a Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. He is the founder and the Director of the MOSA!C Lab. He is the Guest Editor-in-Chief for the IEEE JSAC series on network Softwarization and enablers.

Hannu Flinck received the M.Sc. and Lic.Tech. degrees in computer science and communication systems from Aalto University (formerly, Helsinki University of Technology) in 1986 and 1993, respectively. He was with the Nokia Research Center and the Technology and Innovation Unit of Nokia Networks in various positions. He is a research manager with Nokia Bell Labs, Espoo, Finland. He has been actively participating in a number of EU research projects. His current research interests include mobile edge computing, SDN, and content delivery in mobile networks, particularly in 5G networks.

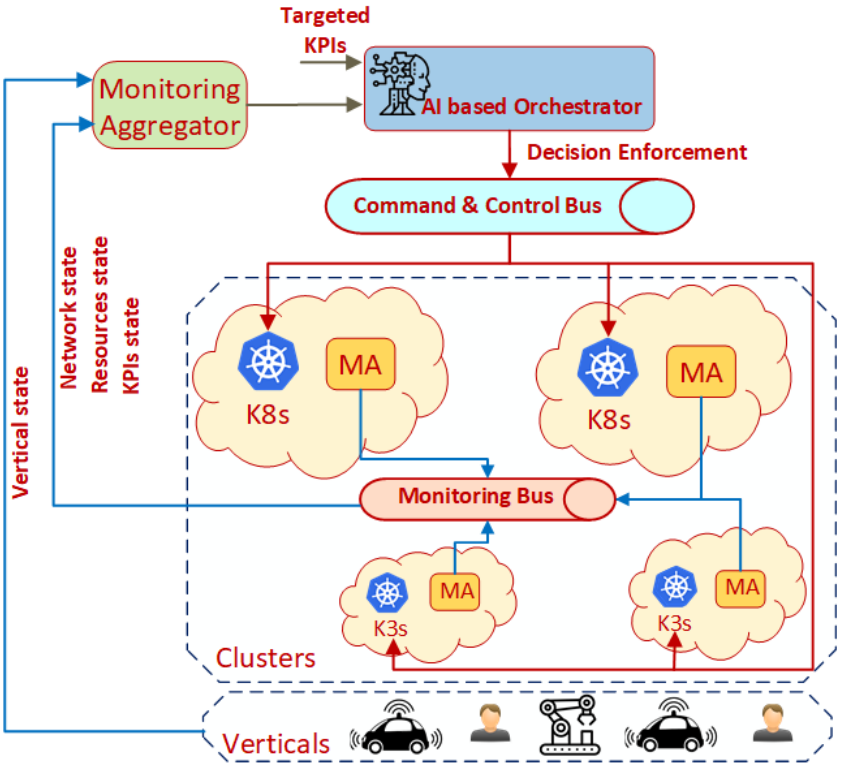


Figure 1: Main idea of AIRO framework.

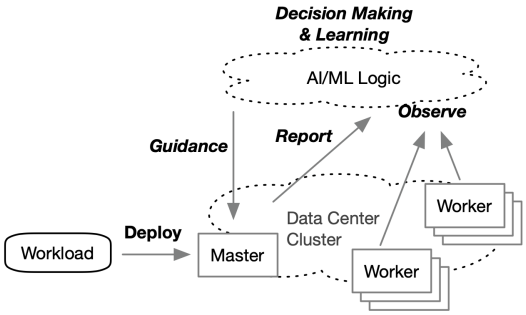


Figure 2: Communication system overview.

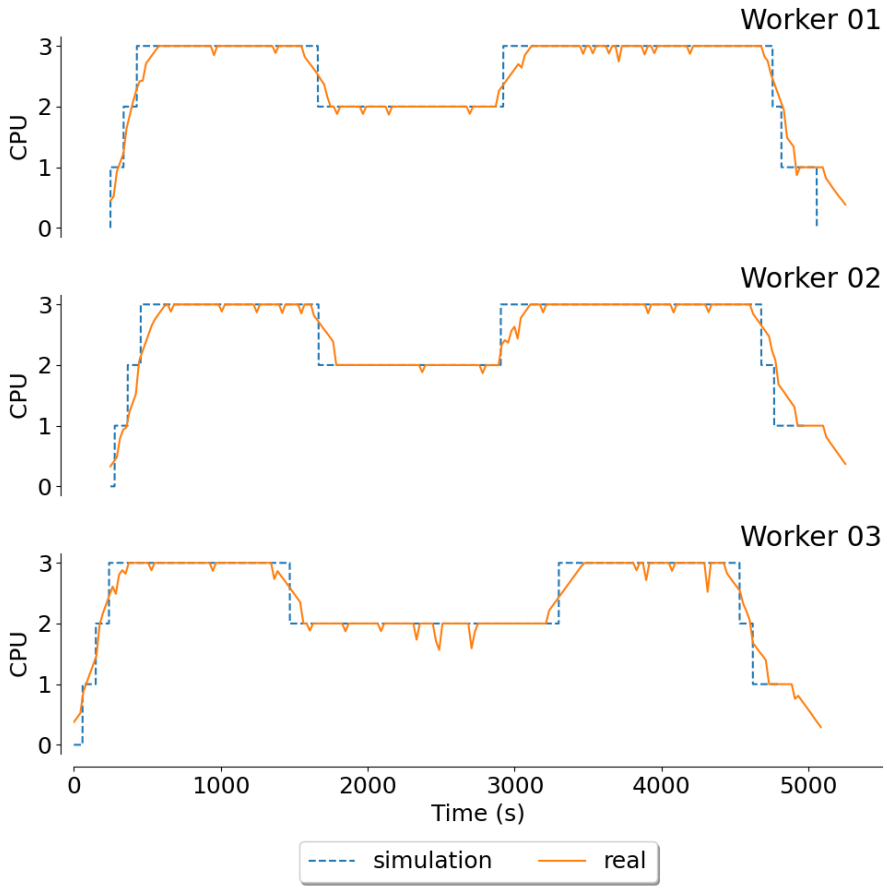
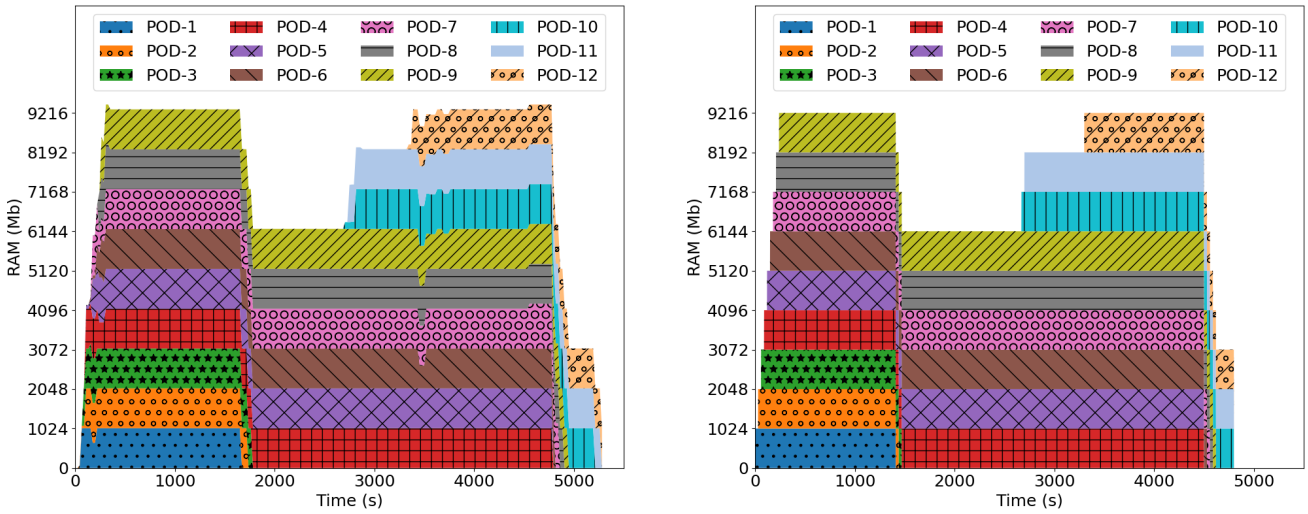


Figure 3: CPU consumption per node.



(a) Real testbed

(b) Simulated testbed

Figure 4: Memory consumption per POD.