Application-specific, Dynamic Reservation of 5G Compute and Network Resources by using Reinforcement Learning

Anousheh Gholami*
University of Maryland
College Park, MD, USA

Kunal Rao
NEC Laboratories America, Inc.
Princeton, NJ, USA

Wang-Pin Hsiung
NEC Laboratories America, Inc.
San Jose, CA, USA

Oliver Po
NEC Laboratories America, Inc.
San Jose, CA, USA

Murugan Sankaradas
NEC Laboratories America, Inc.
Princeton, NJ, USA

John S. Baras
University of Maryland
College Park, MD, USA

Srimat Chakradhar
NEC Laboratories America, Inc.
Princeton, NJ, USA

ABSTRACT

5G services and applications explicitly reserve compute and network resources in today's complex and dynamic infrastructure of multi-tiered computing and cellular networking to ensure application-specific service quality metrics, and the infrastructure providers charge the 5G services for the resources reserved. A static, one-time reservation of resources at service deployment typically results in extended periods of under-utilization of reserved resources during the lifetime of the service operation. This is due to a plethora of reasons like changes in content from the IoT sensors (for example, change in number of people in the field of view of a camera) or a change in the environmental conditions around the IoT sensors (for example, time of the day, rain or fog can affect data acquisition by sensors). Under-utilization of a specific resource like compute can also be due to temporary inadequate availability of another resource like the network bandwidth in a dynamic 5G infrastructure. We propose a novel Reinforcement Learning-based online method to dynamically adjust an application's compute and network resource reservations to minimize under-utilization of requested resources, while ensuring acceptable service quality metrics. We observe that a complex application-specific coupling exists between the compute and network usage of an application. Our proposed method learns this coupling during the operation of the service, and dynamically modulates the compute and network resource requests to mimimize under-utilization of reserved resources. Through experimental evaluation using real-world video analytics application, we show that our technique is able to capture complex compute-network coupling relationship in an online manner i.e. while the application is running, and dynamically adapts and saves upto 65% compute and 93% network resources on average (over multiple runs), without significantly impacting application accuracy.

CCS CONCEPTS

• **Networks** → **Cloud computing**; *Network management*; **Cloud computing**; **Network management**.

ACM Reference Format:
Anousheh Gholami, Kunal Rao, Wang-Pin Hsiung, Oliver Po, Murugan Sankaradas, John S. Baras, and Srimat Chakradhar. 2022. Application-specific, Dynamic Reservation of 5G Compute and Network Resources by using Reinforcement Learning. In *ACM SIGCOMM 2022 Workshop on Network-Application Integration (NAI '22), August 22, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3538401.3546598

1 INTRODUCTION

The infrastructure on which applications run has seen a tremendous change over the past decade. From single-tier e.g. on premise or in the cloud, to now multi-tiered with compute capability at the IoT devices, at the edge (MEC) or in the cloud, and these computing tiers are connected with network with varying capacity and guarantees e.g. 5G, LAN/VLAN, MAN, WAN, etc. as shown in Fig. 1. Such a fabric came into existence, because next-generation applications such as autonomous driving, smart manufacturing, remote health, augmented or virtual reality (AR/VR), etc. have very stringent performance requirements that can be met only through such a tiered-fabric.

For applications to achieve specific performance requirements, they need to reserve certain amount of resources such as network and compute, and applications are charged by the infrastructure providers depending on how much resources they reserve. Higher the reservation more will be the cost. Therefore, applications have to reserve only what they need and avoid unnecessary reservation of resources. During the lifetime of an application, neither the infrastructure (compute and network) remains fixed nor the environment in which the application is operating e.g. scene observed by a camera, is fixed. This leads to variation in the amount of network and compute required by the application and one-time, fixed reservation does not work well, as it can get unnecessarily too expensive due to over provisioning of resources. Moreover, for

*Work done as an intern at NEC Laboratories America, Inc.

NAI '22, August 22, 2022, Amsterdam, Netherlands

ACM ISBN 978-1-4503-9395-9/22/08…$15.00
https://doi.org/10.1145/3538401.3546598

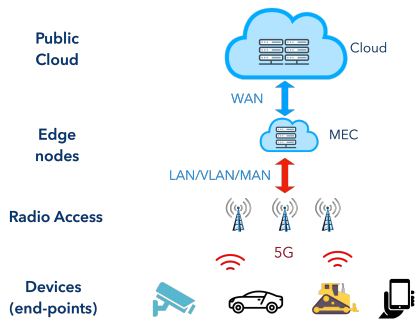


Figure 1: Multi-tiered edge-cloud environment

modern software architectures, which rely on *microservices architecture* for scalability and ease of maintenance of services [1], [9], there is a non-linear coupling relationship between network and compute resources which is studied in [2] in an offline manner. Such offline profiling does provide some guidance regarding resource reservation, but it is not scalable for different applications and for different real-world scenarios.

Given this dynamic variation in resource requirement and non-linear coupling relationships, the problem of optimally and economically deploying microservices-based applications in a heterogeneous and multi-tiered system becomes very challenging. The presence of various kinds of networking capabilities at different tiers (e.g. 5G connectivity between devices and edge servers, MAN between distributed edge resources and WAN between edge and central cloud) and the high variability in the compute (multi-tenancy, heterogeneity, etc.), and changing network (5G NR interference, link congestion, packet loss, etc.) conditions makes this problem even more challenging. Therefore, an online and dynamic solution that takes into account the real-time state of different resources (e.g. available network and compute) and environmental conditions (e.g. scene observed by camera) is necessary to realize an efficient and effective resource orchestration.

Our goal is to optimize the resource requests (network and compute) for different applications automatically. To this end, we incorporate SARSA reinforcement learning (RL) into the resource orchestration framework for microservices-based 5G applications. The main contributions of this paper are as follows:

- We show that the compute and network resources used by an application vary considerably during the lifetime of operation, and the resource reservations made by the application can be dynamically adjusted to minimize under-utilization of reserved resources. This avoids payment for reserved resources that are not utilized.
- We propose a novel application-specific, dynamic reservation of 5G compute and network resources by using reinforcement learning techniques. We automatically capture the application-specific compute and network coupling relationships in a reinforcement learning model, and enable a principled consideration of resource allocation options to significantly reduce under-utilization of reserved network and compute resources.
- We implement a real-world video analytics application and show that our RL-based technique can save upto 65% compute and 93% network on average, without significantly impacting application accuracy.

The paper is organized as follows. We provide the literature review in section 2 and discuss motivation in 3. Section 4 describes the system model. In section 5, we introduce our design and proposed solution. Performance evaluation is presented in section 6. Finally, in section 7, we highlight our conclusions.

2 RELATED WORK

Resource management in cloud environments is a well-known research problem which has already been addressed in the literature [4]. The transition to *microservices architecture* introduces new challenges that have not been addressed. One key challenge is to manage multiple application components deployed across a large number of geographically distributed servers in a dynamic manner. In [5], the allocation policy of microservices to the physical hosts in a cloud datacenter is modelled as a binary quadratic programming problem with the objective of minimizing the interaction cost (inter-node communications). They propose an interaction-aware allocation policy to determine the mapping of microservice requests to hosts. The microservices with frequent interactions are deployed on the same host, resulting in response-time and throughput improvements.

A workload profiling framework [3] for cloud-native applications uses profiling results to deploy an application by using a greedy algorithm with the goal of minimizing interactions. It is shown profiling helps resource allocation methods to reduce the application response time. The problem of application deployment and migration has also been explored in MEC systems as well. Due to the distribution of MEC servers across the geographical area, users' services may need to be migrated as users move. In [11], an MDP formulation is proposed for the dynamic service migration problem in the MEC which captures general cost models and provides a mathematical framework to design optimal service migration policies. Moreover, a new algorithm and a numerical technique for computing the optimal solution is proposed, which is shown to be significantly faster than traditional methods based on the standard value or policy iteration. Authors in [7] design a proactive scheme for placement and migration of an already placed microservice in the MEC setup. In contrast to a conservative policy leading to wasteful resource allocation and a reactive on-demand policy causing high latency, the main objective of this paper is to learn and synthesize the optimal proactive prefetch, deployment and migration schedule by utilizing the user mobility. An RL-based approach based on Dyna-Q algorithm is proposed to solve the problem and it is compared against an on-demand invocation policy and a heuristic algorithm based on an iterative matching process followed by a local search phase in which the solution quality is improved.

While the above works study the response time minimization and dynamic function migration problem for cloud-native applications, they ignore the joint optimization of different resources (compute and network resources, for example) and the impact of

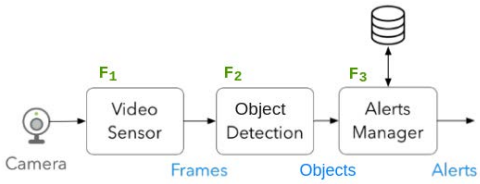


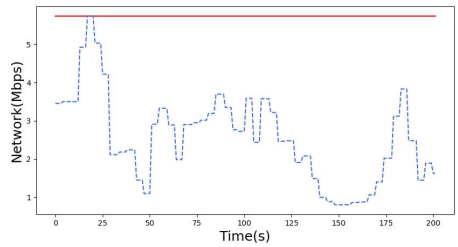
Figure 2: Intelligent transportation systems: object detection application

such joint optimization on the overall performance of an application. Authors in [2] first introduce such coupling functions and illustrate that by considering the coupling relationships into the resource orchestration framework, it is possible to save on network and compute resources without sacrificing the target application's performance. The proposed framework, ROMA, solves the joint problem of function placement, and network and compute resource allocations, all modelled as a mixed integer linear program (MILP). The resource coupling relationships are derived empirically by profiling different applications, and then incorporated into the problem formulation. The performance of ROMA is evaluated on two real-world applications i.e. surveillance (watchlist) and transportation application (person and car detection) and benchmarked against a static resource provisioning framework that ignores resource couplings. The coupling functions are assumed to be linear so that the resulting optimization problem is a MILP or an LP (in the case of only resource allocation).

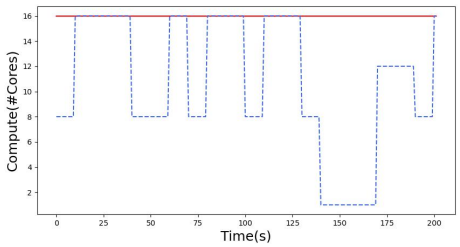
In contrast to prior work, we (a) make no assumptions about the specific coupling between compute and network resource usage (in our experiments, we observed that the coupling relationship is application-specific, and non-linear), (b) learn the application-specific coupling relationship while the application is in operation, and (c) jointly optimize the two resources by using a reinforcement learning approach. The complex coupling relationship is captured in a reinforcement learning model, and it is used to make decisions about network and compute allocations in real-time.

3 MOTIVATION

In this section, we discuss the motivation behind dynamic resource reservation for microservices-based video analytics applications. Fig. 2 shows the object (person) detection analytics pipeline that is part of a larger Intelligent Transportation Systems (ITS) application. Fig. 3a and Fig. 3b show the required compute and network resources for the object detection pipeline for a real-world video that has a varying number of objects (traffic participants like vehicles, pedestrians etc.) in different video frames. We also show the number of cores allocated to the object detection pipeline, and the network bandwidth required to stream the video. We denote the strategy of one-time fixed resource reservation (assuming infrastructure is able to support this) as "static" in the rest of the paper.



(a) Network resource requirement



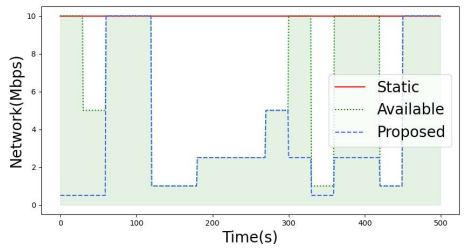
(b) Compute resource requirement

Figure 3: Impact of environment and stream content on resources required

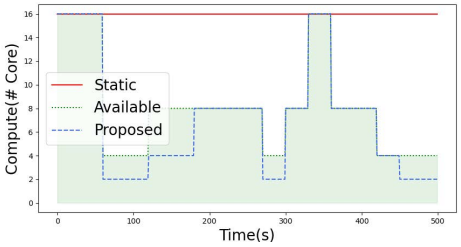
3.1 Impact of environment and stream content on resources required

Fig. 3a shows that the required network bandwidth (in Mbps) for the transportation video stream varies over time. Typically, when the video stream does not have much variation across frames, the network bitrate drops, while if there is significant variation in the video stream from one frame to another, then the bitrate is high. For example, at night time condition, when there is not much activity going on, the bitrate drops, while for the same camera, during the day time when lots of people are walking around, the network bitrate goes high. This is an artifact of the way network cameras encode, compress and stream videos. Such variation in the network bitrate can be leveraged in appropriately reserving network resources. Instead of one time, fixed network reservation, we can adjust the network resources dynamically as the environment and stream content i.e. scene in front of the camera changes.

As the network bandwidth usage varies, we also observe a variation in the compute resources required to process the video stream. Fig. 3b shows the minimum amount of compute resources required to achieve similar accuracy as over-provisioned, fixed amount of compute resources. We observe that there is an opportunity to save on compute, without impacting application accuracy. For example, at about 150 seconds into the video stream, the network bitrate drops and at that time, the amount of required compute also goes down. It does not benefit the application to reserve more compute resources because there is not much content to process. Thus, we can save on compute resources in reaction to changes in the environment and stream content, without impacting application accuracy.



(a) Network resources required vs. resources made available by infrastructure



(b) Compute resources required

Figure 4: Impact of dynamic infrastructure on resources required

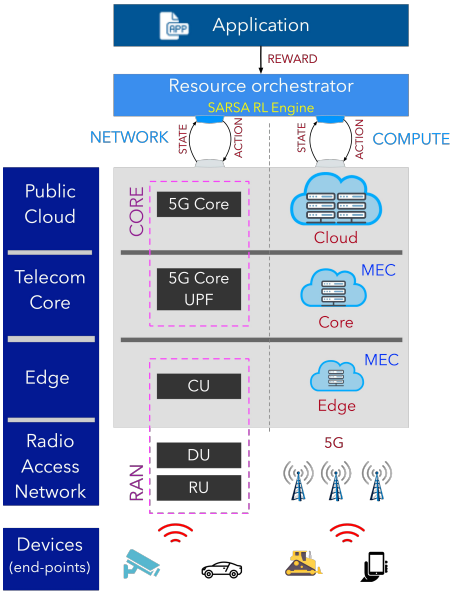


Figure 5: System Design

3.2 Impact of dynamic infrastructure on resources required

In section 3.1, we studied the impact of environment and stream content on the resources required by an application by assuming that the infrastructure can adequately satisfy the resource requests at all times. However, in practice, this may not be the case always. Since the infrastructure is common and is shared across multiple applications, it is not always possible for the infrastructure to satisfy all resource requests. In this section, we study the impact of changes in infrastructure conditions on resource requests by an application.

In Fig. 4a and 4b, we show the case when infrastructure is not abundant for network as well as compute resources. This infrastructure condition is depicted as "Available", which is the maximum network or compute infrastructure available (shown in light green color in Fig. 4a and 4b). Now, the application can only reserve within these infrastructure limits. The amount of resources required by application (within the "Available" resources) is denoted as "Proposed". We observe that changes in infrastructure conditions directly impacts the reservations that an application can make. For example, at around 120 seconds, when the available network from the infrastructure drops and even though application would have desired to have higher network reservation, infrastructure is not able to provide it, then there is no point in reserving high compute, even though the infrastructure can provide it. Here we see that the "Proposed" compute resource goes down. Thus in these scenarios, reserving lower compute than what the infrastructure can provide, will save on the compute resource reservation. This is true vice versa as well i.e. if the compute that the infrastructure can provide drops, even though the application would have desired to be higher, then there is no point in reserving high network because there isn't enough compute available to process the additional content. Thus, we see that infrastructure conditions do impact required resources and we can save on compute and/or network in reaction to changes in infrastructure conditions.

4 SYSTEM MODEL

In this section, we present our system model. Given a multi-tiered infrastructure consisting of computing nodes at different tiers (edge, and central cloud), the reservation of the resources to different applications is realized through slicing. Let $\mathcal{M}$ and $\mathcal{T}$ denote the set of compute nodes and the set of different resource types on each node respectively. Each node $m \in \mathcal{M}$ is specified by $(\boldsymbol{g}_m, tier_m)$, where $\boldsymbol{g}_m = [g_m^t, t \in \mathcal{T}]$ is the vector of available resources, and $tier_m$ denotes the associated tier.

We model an application as a set of functions or microservices and interconnections that represent the data dependency between functions. Let $G = (V, E)$ be the graph representing the application, where V denotes the set of functions in the application and E represents the interconnections (data dependency) between functions. Furthermore, $\mathcal{R}_v = (\omega_v, C_v)$ denotes function v's requirements, where ω_v and $C_v = (core_{min,v}, tier_v)$ are the networking and computing requirements. Our goal is to optimize the resource reservation for different functions and interconnections between them such that the overall application performance is maximized with a minimum amount of total resources.

5 SYSTEM DESIGN AND PROPOSED SOLUTION

Since the demands for different resources fluctuate over time as discussed in Section 3, a dynamic resource allocation approach is necessary to address the adjustments in the resource usage or

placement decisions, taking into account the resource coupling relationships. We propose an RL based orchestration system that automatically derives the resource coupling relationship and selects the best action periodically. We compared Q-learning and SARSA [12] algorithms and found that the learned model by SARSA has better performance. We assume that the available amount of resource t on node m is quantized into L levels, denoted by the set $\mathcal{G}_m^t = \{g_{m,1}^t, ..., g_{m,L}^t\}$. Let $y_{v,m}^t$ denote the amount of resource $t \in \mathcal{T}$ of node $m \in \mathcal{M}$ allocated to function $v \in V$. A valid resource allocation solution must satisfy node capacity constraints given by:

$$y_{v,m}^t \leq g_m^t, \quad \forall t \in \mathcal{T}, v \in V, m \in \mathcal{M} \tag{1}$$

We formulate the resource allocation problem as an episodic RL algorithm, so that the infrastructure nodes' capacities are not violated. Our system design is shown in Fig. 5. We formulate the decision making process as an MDP, denoted by the tuple $< o, a, r >$, which are detailed as follows:

- **State representation** The state of the system at time step i is represented by the tuple $o_i = (\{g_m^{t,i} \in \mathcal{G}_m^t, m \in \mathcal{M}, t \in \mathcal{T}\}, \{y_{v,m}^{t,i}, v \in V, m \in \mathcal{M}, t \in \mathcal{T}\})$. In this paper, we assume that the function placement decisions are given according to heuristic solutions such as [6], and focus on the resource allocation problem. As a result, the size of state space is also reduced.
- **Action representation** An action is a valid resource reservation that determines the amount of resources which an infrastructure node hosting an application microservice consume. We define the action set to include $A = 5|V|$ actions capturing the five possible actions for each function: (a) increase/decrease the allocated network/compute resources, or (b) not change compute and network reservations.
- **Reward function** In RL, the learning agent improves its performance by constantly receiving reward from the environment. To increase the probability of good actions, a positive reward is returned for valid actions. To this end, we define $d_i = \frac{\alpha p_{i+1}}{\sum_{t,v,m} \beta_t y_{v,m}^{t,i+1}} - \frac{\alpha p_i}{\sum_{t,v,m} \beta_t y_{v,m}^{t,i}}$ where p_i is the performance metric of the target application (object detection score defined in Section 6 for an object detection application). The parameters α and β_t's are used for the tradeoff between compute, network and performance metric. d_i has a positive value only if the difference between the fraction of performance over total used resources from step i to $i+1$ is positive. The RL reward function is defined as follows:

$$r_i = r(o_i, a_i, a_{i+1}) = \begin{cases} d_i & \text{if (1)} \\ \text{-H} & \text{ow} \end{cases}$$

where H is a large positive number. In other words, the agent receives a penalty if it reserves resources such that the capacity constraint for infrastructure nodes is violated.

6 EVALUATION

In this section, we present the experimental setup and benchmark the RL resource orchestration solution, against a static resource reservation scheme, which ignores the coupling between resources and environmental changes. We also compare the performance of RL solution with ROMA [2] in an offline setting. Furthermore, we show the effectiveness of our technique in an online setting on a real-world video analytics application.

6.1 Experimental Setup

In our experimental setup, Kubernetes [8] cluster is setup on our MEC servers where intelligent transportation systems (object detection application) run within pods in Kubernetes. Each function runs as a separate pod and multiple replicas of these pods are created, as necessary. We stream videos over 5G from video server using ffmpeg [10] and they are processed in MEC servers on a Kubernetes cluster, within pods.

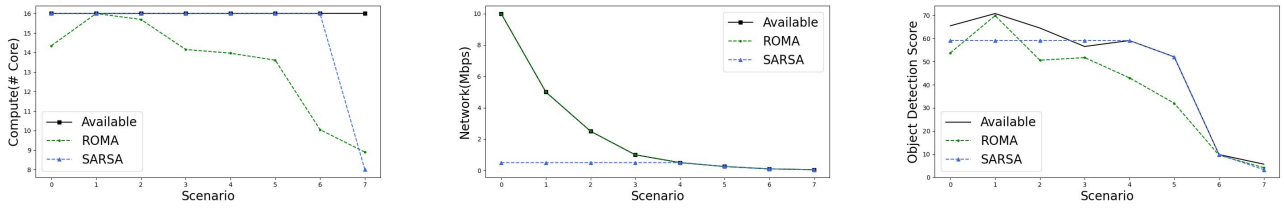
6.2 Numerical Results

We present our results for the ITS use case. The goal is to detect vehicles or pedestrians in the video streams.

6.2.1 Performance comparison: Available, ROMA and SARSA. We implement the proposed RL-based resource reservation approach (denoted as SARSA) for a car detection application and evaluate its capability to capture the non-linear resource coupling relationships and save on resource reservation. For application accuracy metric, we define a weighted object detection score (different from confidence score) as $object\ detection\ score = \sum_{f \in FRAME} w_f \frac{TP_f}{GT_f}$, where f is the frame index, $FRAME$ is the set of processed frames, TP_f is the number of true positive objects and GT_f is the number of objects in ground truth in frame f. For TP classification, we use intersection over union (IoU) metric to measure the overlap between detected and ground truth objects and detections with IoU over 0.5 are considered as TP. We consider two different experiments and in each experiment, we consider several scenarios with given available compute and network resources.

Experiment 1: In this experiment, we consider 8 scenarios in which the available compute is fixed and the available network is reduced gradually (network becomes bottleneck). Fig. 6a, 6b and 6c illustrate the compute and network reservation and the object detection score respectively. We see that across all scenarios, the object detection score (shown in Fig. 6c) for SARSA is comparable or slightly lower than "Available", which is when available compute and network is used. Now, for such good application accuracy, we see that SARSA reserves significantly less network resource (shown in Fig. 6b) than maximum available and when the available network is really low, SARSA also brings down the compute (scenario 7 in Fig. 6a) reservation. Compared to ROMA, SARSA has better accuracy, saves a lot on network but looses slightly on compute. Thus, we see that for similar or slightly lower application accuracy compared to "Available", SARSA is able to save a lot on compute and network resource reservation.

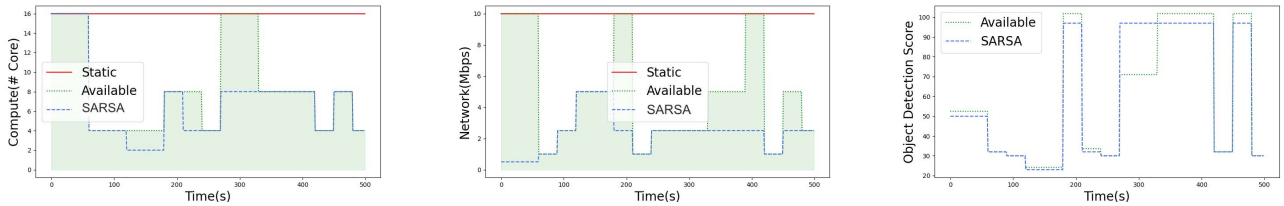
Experiment 2: In this experiment, we consider 4 scenarios in which we fix the available network resources and gradually reduce the available compute resources (compute becomes bottleneck). Fig. 6d, 6e and 6f illustrate the compute and network reservation and the object detection score respectively. Again, we see that SARSA has comparable or slightly lower object detection score (shown in Fig. 6f) than "Available". SARSA achieves this similar application accuracy at significantly lower network reservation (shown in Fig. 6e). The compute reservation by SARSA (and ROMA) is same as



(a) Exp 1: Compute resource reservation (b) Exp 1: Network resource reservation (c) Exp 1: Object detection score

(d) Exp 2: Compute resource reservation (e) Exp 2: Network resource reservation (f) Exp2: Object detection score

Figure 6: Performance of object (car) detection application: SARSA vs. ROMA



(a) Compute resource reservation (b) Network resource reservation (c) Object detection score

Figure 7: Performance of SARSA for object (person) detection application

the "Available". Compared to ROMA, SARSA saves a lot on network resource reservation and most of the time gives better application accuracy, while compute reservation is the same. Thus, we see that for similar or slightly lower application accuracy compared to "Available", SARSA is able to save significant network reservation.

6.2.2 Performance of SARSA in an online setup. In this section, we evaluate performance of SARSA in an online setup where available compute and network varies over time based on a discrete uniform distribution on compute and network space. We run 15 experiments and observed that on average, SARSA reserves upto 93% less network and 65% less compute resources than "Available" compute and network resources. Fig. 7a, 7b, and 7c illustrate the compute and network reservation and the object detection score respectively for a single run. We see that for almost same object detection score (shown in Fig. 7c), SARSA saves on network and compute resource upto 50% and 95%, respectively. Thus, in real-world deployment, we show that our Reinforcement Learning-based online technique is quite effective in capturing the compute and network coupling relationship and is able to significantly reduce network and compute resource reservation.

7 CONCLUSION

In today's deployment of microservices-based 5G applications, infrastructure providers charge based on the compute and network resource reservations. In this paper, we show that compute and network resources used by an application vary depending on the environment and stream content, and also depending on the changes in infrastructure conditions. With this insight, we propose an application-specific, novel online Reinforcement Learning-based technique to dynamically adjust compute and network resources for an application. Our experiments show that for a real-world video analytics application, our technique saves upto 65% compute and 93% network resources on average, without significantly impacting application accuracy.

REFERENCES

[1] Balalaie, A., Heydarnoori, A., and Jamshidi, P. Microservices architecture enables devops: Migration to a cloud-native architecture. *Ieee Software 33*, 3 (2016), 42–52.

[2] Gholami, A., Rao, K., Hsiung, W.-P., Po, O., Sankaradas, M., and Chakradhar, S. Roma: Resource orchestration for microservices-based 5g applications. *arXiv preprint arXiv:2201.11067* (2022).

[3] Han, J., Hong, Y., and Kim, J. Refining microservices placement employing workload profiling over multiple kubernetes clusters. *IEEE Access 8* (2020), 192543–192556.

[4] Jennings, B., and Stadler, R. Resource management in clouds: Survey and

research challenges. *Journal of Network and Systems Management 23*, 3 (2015), 567–619.

[5] Joseph, C. T., and Chandrasekaran, K. Intma: Dynamic interaction-aware resource allocation for containerized microservices in cloud environments. *Journal of Systems Architecture 111* (2020), 101785.

[6] Rao, K., Coviello, G., Hsiung, W.-P., and T. Chakradhar, S. ECO: Edge-Cloud Optimization of 5G applications. In *The 21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid 2021), Melbourne, Victoria, Australia* (2021), pp. 649–659.

[7] Ray, K., Banerjee, A., and Narendra, N. C. Proactive microservice placement and migration for mobile edge computing. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)* (2020), IEEE, pp. 28–41.

[8] Rensin, D. K. *Kubernetes - Scheduling the Future at Cloud Scale.* 1005 Gravenstein Highway North Sebastopol, CA 95472, 2015.

[9] Taibi, D., Lenarduzzi, V., and Pahl, C. Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Computing 4*, 5 (2017), 22–32.

[10] Tomar, S. Converting video formats with ffmpeg. *Linux J. 2006*, 146 (jun 2006), 10.

[11] Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K., and Leung, K. K. Dynamic service migration in mobile edge computing based on markov decision process. *IEEE/ACM Transactions on Networking 27*, 3 (2019), 1272–1288.

[12] Wiering, M., and Schmidhuber, J. Fast online q (λ). *Machine Learning 33*, 1 (1998), 105–115.