K-RAF: A Kubernetes-based Resource Augmentation Framework for Edge Devices

Youngwoo Jang[§*], Jiseob Byeon[§*], Soonbeom Kwon[§*]
Illyoung Choi[⋆], Dukyun Nam[§], Byungchul Tak[§], Gap-Joo Na[◇], Young-Kyoon Suh[§†]

[§]Kyungpook National University, Daegu, Republic of Korea
[⋆]University of Arizona, Tucson, AZ, USA
[◇]Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

ABSTRACT

Internet of Things (IoT) (or edge) devices are typically resource-constrained in terms of CPU, memory, and storage. Thus, it is viable for the devices to request resource provisioning to an edge server in the presence of growing data and heavy computation, as the edge server provides better accessibility than cloud servers. Consequently, the edge devices often perform computation and storage provisioning to the edge servers in large-scale data operations. However, the conventional methods for provisioning edge devices take into little consideration the characteristics of resources that jobs executed at the devices rely on. In particular, fully migrating computation jobs from the device to the server may waste valuable resources of the server without considering the computation and I/O characteristics of the jobs, thereby making the devices' resources idle. To overcome these limitations, we propose a novel Kubernetes-based resource augmentation framework, termed *K-RAF*, for provisioning edge devices with limited capabilities and accelerating the devices' job processing. Our experiment demonstrates that utilizing GPU acceleration, on average, K-RAF can run tasks 306 times faster than local computation on an edge device. Also, we show that utilizing the task distribution between an edge device and K-RAF can offer an average speedup of about 40% compared to K-RAF alone.

KEYWORDS

Edge devices, Private cloud, Resource augmentation, Kubernetes

ACM Reference Format:
Youngwoo Jang, Jiseob Byeon, Soonbeom Kwon, Illyoung Choi, Dukyun Nam, Byungchul Tak, Gap-Joo Na, and Young-Kyoon Suh. 2024. K-RAF: A Kubernetes-based Resource Augmentation Framework for Edge Devices. In *The 33rd International Symposium on High-Performance Parallel and Distributed Computing (HPDC '24), June 3–7, 2024, Pisa, Italy.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3625549.3658826

[*]Student author
[†]Corresponding author

HPDC '24, June 3–7, 2024, Pisa, Italy

ACM ISBN 979-8-4007-0413-0/24/06
https://doi.org/10.1145/3625549.3658826

1 INTRODUCTION

In the recent IoT environments, there is a growing need to perform both I/O-intensive tasks (e.g., sensor data collection and retrieval and multimedia data annotation) and compute-intensive operations (e.g., model training and inference, object recognition, video summarizing) at the edge. However, it is tough to process such large jobs promptly due to the limited computation and storage capacity of edge devices. Consequently, edge devices frequently depend on edge servers for resource provisioning, which offer greater accessibility compared to cloud servers. As a result, whenever extensive data operations are required, the edge devices often delegate the computation and I/O tasks to the edge servers.

However, the conventional works to provision such edge devices expose several limitations. First, existing provisioning approaches [1, 12, 14] based on edge servers seldom reflect the characteristics of resources that applications heavily rely on, particularly for compute-intensive tasks that demand significant computational resources from edge devices. Thus, provisioning compute resources at the same performance level becomes less effective than explicitly tailoring compute resources for high-level computational purposes. In addition, the advent of high-performance IoT devices [11] has rendered the traditional complete computation at edge servers no longer optimal. The conventional approach, which does not consider diverse environments, has not been quite effective in efficiently utilizing the computational and I/O resources of both edge servers and edge devices.

To address these concerns, in this paper, we propose a novel resource augmentation framework, termed *K-RAF*, which assists edge devices to overcome their limited capabilities by provisioning virtualized computation and storage resources in a Kubernetes environment. In particular, our framework and the edge devices can collaborate to quickly finish given tasks by fully exploiting available computation resources on both sides. In addition, this collaboration can reduce the load on edge servers and save computation resources, enabling computation provisioning for more edge devices.

2 THE PROPOSED FRAMEWORK: K-RAF

Figure 1 illustrates the overall architecture of the K-RAF framework that we have implemented. In the architecture, we deploy data access and application pods sharing a single PVC (Persistent Volume Claim) in the Kubernetes environment [7]. Within this structure, the application pod is responsible for managing the resources in the virtual environment where the application operates. The pod ensures consistency by running the same application as the one already operational on the device when provisioning resources.

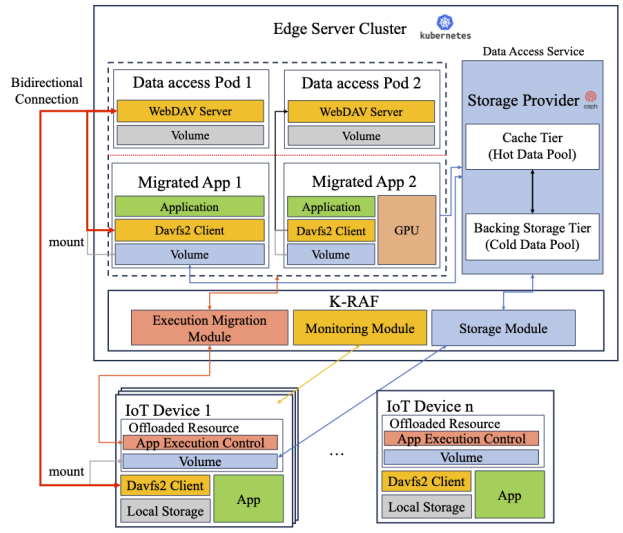


Figure 1: Overall architecture of our proposed framework

Concurrently, the data access pod is mounted on the PVC to store data generated by both the device and application pods. An internal davfs2 server within the pod facilitates the transmission of these data to various endpoints. To accelerate I/O intensive tasks, we apply a cache tiering technique [2] using rook-ceph [9], in which we define a cache tier, a storage resource mapped to a solid-state drive, and a backing storage tier, a storage resource mapped to a hard disk drive. We decide the promotion or demotion of the used tier through I/O traces during the execution of the application using the storage resource. This design reflects our storage provisioning approach reflecting an application's I/O characteristics to improve an application's IOPS. Moreover, to optimize the execution of tasks that require substantial computational resources, we employ GPUs utilized as virtual acceleration resources within the Kubernetes ecosystem. The GPUs are made available to the application pods that need computation acceleration. The allocation of the GPUs is quantified in terms of VRAM units and is determined at the time of pod initialization. In this architecture, computationally intensive applications, especially those associated with AI models, are poised to benefit significantly from this enhanced processing capability acceleration.

Also, we have implemented a bidirectional connection structure, as illustrated in Figure 1, for task migration and data sharing between an edge device and its application pod. To ensure accessibility in various environments, we utilize the WebDAV [13] protocol to establish the connection between the endpoints as it meets our concurrency and data integrity requirements. Upon running the WebDAV server in the data access pod, edge devices and application pods connect by mounting via davfs2 [4]. This enables efficient task allocation and data sharing between devices and application pods using the secure WebDAV protocol, which supports concurrency and data integrity.

3 PRELIMINARY EVALUATION

To test the validity of the proposed K-RAF framework, we evaluate its performance on compute-bound and I/O-bound tasks as follows.

3.1 Compute-bound Tasks

We investigated the performance of image inference tasks using Yolo [6], which is a compute-bound application, in various execution contexts. We aimed to establish a baseline for efficient resource distribution between an edge device and the corresponding edge server. For our measurement, we considered four types of execution environments: 1) a pod with the RTX 3080 GPU [10] (denoted as 'K-RAF GPU') at the server, 2) environments with jobs distributed across Raspberry Pi and pod (denoted as 'K-RAF device') at the server, 3) a pod without the GPU (denoted as 'K-RAF') at the server, and 4) a Raspberry Pi model 4 (denoted as 'Edge device'). We executed image inference tasks across expanding datasets, varying from 1 GB to 4 GB in each environment.

As shown in Figure 2(a), on average, K-RAF outperformed the edge device by about 36.8x, while K-RAF GPU achieved a remarkable speedup of about 306x compared to the edge device. In particular, on average, K-RAF GPU outpaced K-RAF by a factor of 8.3, highlighting the critical role of GPU acceleration in executing compute-intensive tasks within virtualized environments. Furthermore, compared to K-RAF, the K-RAF device yielded about an average speedup of 4%. This outcome highlights the necessity of exploring task partitioning with edge devices. By doing so, we can effectively utilize edge server resources by tapping into the idle capacities of edge devices, all while ensuring computational performance is upheld.

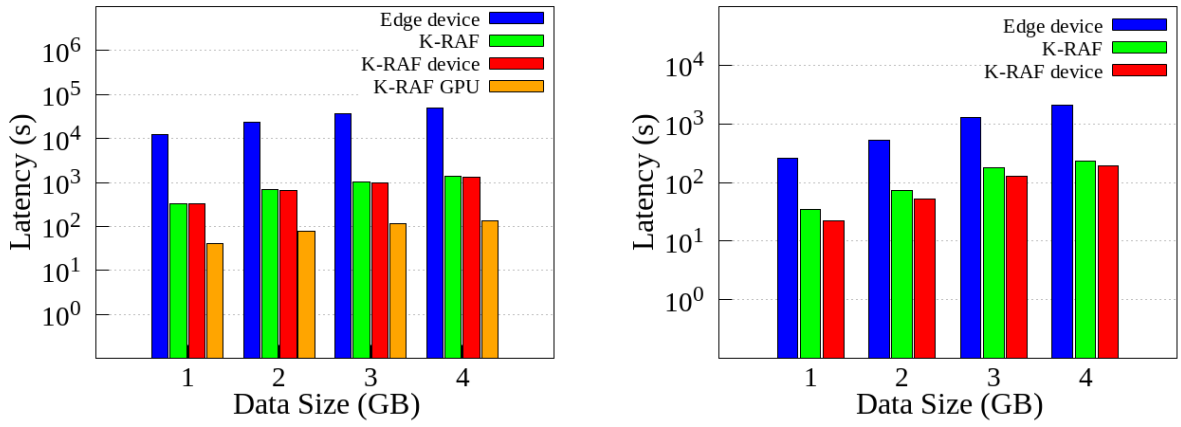
3.2 I/O-bound Tasks

In this section, we explore a potential opportunity for mitigating I/O overhead in the collaboration between an edge device and the edge server for efficient task completion. More specifically, we measure the performance of ElasticSearch [3], an I/O-bound application, under various execution scenarios to identify potential opportunities for decreasing I/O overhead and enhancing task efficiency. To this end, we compare the performance of the K-RAF, K-RAF device, and Edge device used in Section 3.1. (K-RAF GPU is not considered as it is out of focus in this experiment.) To realize an I/O bound task, we loaded data into Elasticsearch for image datasets ranging from 1 GB to 4 GB in each environment.

Figure 2(b) illustrates the loading results. The average latency when loading data in K-RAF was 95.4 seconds, while the average latency in edge devices was 692 seconds, resulting in approximately a 7.25x performance improvement for the operation in K-RAF. However, the average latency of the tasks in K-RAF device was 67.3 seconds, about 1.4x faster than in K-RAF. These results show that offloading storage resources can improve the performance of IO-bound applications. It also suggests that task partitioning using edge devices can improve not only the efficient use of resources but also the performance of applications in terms of I/O efficiency.

4 RELATED WORK

Google Distributed Cloud Edge [5] uses Google's high-performance cloud infrastructure [8] to achieve flexible scalability of computing



(a) Average latency to execute a compute-bound task

(b) Average latency to execute a IO-bound task

Figure 2: Performance comparison on compute-bound and IO-bound tasks

and storage resources. But since K-RAF edge computing collaborates with local hardware, it is less susceptible to network delay than Google Distributed Cloud Edge due to the shorter physical distance.

5 CONCLUSION AND FUTURE WORK

To overcome the limitations of conventional provisioning methods, this paper presented the Kubernetes-based Resource Augmentation Framework (K-RAF) that can efficiently allocate computation and storage resources on the edge server for resource-constrained IoT edge devices. Our experiments revealed that K-RAF successfully enhanced the performance of compute resource provisioning through GPU virtualization and the WebDAV protocol and achieved efficient task distribution and data sharing between edge devices and their application pods. Also, K-RAF could improve the performance of I/O-intensive tasks, such as sensor data collection, image processing, and media streaming, as well as compute-intensive tasks, such as model training and inference, object recognition and video summarizing, by leveraging the characteristics of the resources employed by the application. We see that this promising approach can heighten the efficiency of task execution and save resources on the edge server, optimizing the overall system sustainability.

Our preliminary evaluation results show the practicality of K-RAF in improving system performance and resource utilization in IoT environments by using IoT edge devices as computing resources when offloading storage-intensive workloads. Our work is ongoing in three directions. First, we are currently devising a differential offloading policy, meaning offloading different resources considering edge devices' computational capacity and storage capacity. Second, we are adding a PVC-resizing feature readjusting excessive PVC based on current usage. Lastly, we are implementing an adaptive workload distribution algorithm based on the data from multiple execution environments. That is, we're developing a feature that dynamically distributes the workloads between a device and its application pod to maximize resource utilization and reduce latency by adapting to the execution context in real time. Moreover, to further attest to the validity of K-RAF, we are currently evaluating the performance of the WebDAV protocol and conducting a quantitative analysis of the framework with the influence of bare metal excluded. Again, our ultimate goal is to efficiently provision edge resources to maximize performance while reducing the latency in completing a given task.

NOTES

This work was supported by the National Research Foundation of Korea (NRF) grants (4199990214394, NRF-2021R1I1A3056669, and NRF-2018R1A6A1A03025109) funded by the Korean government (MSIT and MoE) and by partial support from an Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2021-0-00546, Building a Digital Open Lab as open innovation platform). All the materials used in this paper are publicly available at https://github.com/lab-paper-code/K-RAF.

REFERENCES

[1] Ale, L., et al. Delay-aware and Energy-efficient Computation Offloading in Mobile-Edge Computing using Deep Reinforcement Learning. *IEEE Transactions on Cognitive Communications and Networking 7*, 3 (2021), 881–892.
[2] Ceph Authors and Contributors. Cache Tiering. https://docs.ceph.com/en/latest/rados/operations/cache-tiering/, viewed on April 17, 2024.
[3] elastic. Elasticsearch. https://github.com/elastic/elasticsearch/, viewed on April 17, 2024.
[4] Free Software Foundation, Inc. davfs2 - Summary. https://savannah.nongnu.org/projects/davfs2, viewed on April 17, 2024.
[5] Google. Google Distributed Cloud Edge. https://cloud.google.com/distributed-cloud/edge/latest/docs/overview, viewed on April 17, 2024.
[6] Jocher, G., et al. ultralytics/yolov5: v7. 0-yolov5 SOTA Realtime Instance Segmentation. *Zenodo* (2022).
[7] Kubernetes. Kubernetes. https://kubernetes.io/docs/, viewed on April 17, 2024.
[8] Mauch, V., Kunze, M., and Hillenbrand, M. High Performance Cloud Computing. *Future Generation Computer Systems 29*, 6 (2013), 1408–1416.
[9] Rook-Ceph. Rook-ceph. https://rook.io/, viewed on April 17, 2024.
[10] The Kubernetes Authors. Schedule GPUs. https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/, viewed on April 17, 2024.
[11] Valladares, S., et al. Performance Evaluation of the NVIDIA Jetson Nano through a Real-time Machine Learning Application. In *Proceedings of the 4th Int'l Conf. on Intelligent Human Systems Integration* (2021), Springer, pp. 343–349.
[12] Wang, T., Zhou, J., Liu, A., Bhuiyan, M. Z. A., Wang, G., and Jia, W. Fog-based computing and storage offloading for data synchronization in iot. *IEEE Internet of Things Journal 6*, 3 (2018), 4272–4282.
[13] WebDAV. Webdav resources. http://www.webdav.org/, viewed on April 17, 2024.
[14] Zhang, Y., Chen, X., Chen, Y., Li, Z., and Huang, J. Cost efficient scheduling for delay-sensitive tasks in edge computing system. In *2018 IEEE International Conference on Services Computing (SCC)* (2018), IEEE, pp. 73–80.