I.J. Wireless and Microwave Technologies, **2022, 5, 1-9**
Published Online on October 8, 2022 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijwmt.2022.05.01

Load Balancing Mechanism for Edge-Cloud-Based Priorities Containers

Wael Hadeed
Computer Science Department, College of Computer Science and Mathematics, University of Mosul, Mosul, IRAQ
Email: wael.hadeed@uomosul.edu.iq

Dhuha Basheer Abdullah
Computer Science Department, College of Computer Science and Mathematics, University of Mosul, Mosul, IRAQ
Email: prof.dhuha_basheer@uomosul.edu.iq

Received: 15 April 2022; Revised: 02 May 2022; Accepted: 25 May 2022; Published: 08 October 2022

Abstract: Considering edge devices have limited resources, it's critical to keep track of their current resource usage and device resource allocation algorithms that send containers to edge and cloud nodes according to their priority. To minimize the strain on edge devices and enable the running of mission-critical applications, edge containers may need to be transferred to a cloud platform. In this paper, we suggested a mechanism for prioritizing container balance between the edge and cloud while attempting to assign delay-sensitive containers to edge nodes. We assess the performance of Docker container management systems on resource-constrained computers and offer ways for reducing administration and migration overhead based on the workload type, to bring load balance to the systems. The proposed algorithm gives flexibility to get the best possible ways to achieve load balancing. The Tensorflow's object discovery API was used, accessed with Flask, Python's micro-web framework. Docker container management technology was used in the implementation of this application.

Index Terms: Container, Docker, Edge computing, Load Balance, Resource Management.

1. Introduction

1.1 Overview

The cloud has become increasingly significant for enterprises lately. Companies move a large chunk of their data to the cloud. the cloud platform has a lot of data, the analytics and transfer of that data haven't been fully improved until the arrival of edge computing. Edge computing is an information technology architecture that allows data to be processed computationally away from nodes that are centralized and nearer to the network's local edge. Lower latency, better bandwidth, improved user experiences with the delivery of nearby real-time insights are some of the advantages of edge computing over a traditional centralized approach .[1]

In edge and cloud nodes, containerization is utilized to allow processing resources. Because containers are lightweight, loosely linked, versatile, secure, scalable, and portable, they have become a viable alternative to virtualization [2,3]. Virtual machines and containers both isolate and allocate resources. Multiple containers can work on the same computer thanks to containerization.

The hypervisor in virtualization allows several VMs (virtual machines) to work on same the system. Virtual machines are hypothetical hardware whereas containers virtualize the operating system. Containers, as a result, are lighter than virtual computers. Multi containers can be sharing the kernel of the operating system in containerization. In virtualization, each VM (virtual machine) has its operating system [4]. Containers are smaller and faster to start than virtual machines, allowing multiple containers to run on the same computer.[5,6]

Edge computing reduces latency and saves bandwidth, which is vital for mission-critical domains such as healthcare monitoring and factory automation that demand real-time data-driven decisions. Additionally, edge computing enhances security and privacy. Edge computing reduces the amount of data transmitted through a network. Edge computing also increases privacy by giving consumers more control over their data by allowing them to upload fewer data to the cloud and analyze more data locally . [5]

As a result, edge computing addresses several difficulties by lowering the cost of data access to the cloud and storing data at the edge rather than sharing it. When processing shifts to the edge, the resource provisioning challenge

arises. To establish load balancing, there must be a balance in the processing transmission process between the cloud and the edge [5,7].

The main objective of this research is to give realistic solutions in the harmony between cloud and edge. load Balancing between the cloud and the edge by routing the tasks and relying on the set of metrics serving this field (such as response time, implementation time, resource availability ... etc.). Consideration should be given to making good use of the resources available on the edge and the cloud to reduce overhead. The process of routing tasks and placing them in containers while exploiting the properties of containers and taking advantage of the Docker engine, gave us a flexible way to manage the execution of tasks in general. There is a group of works, part of which was addressed in this research. Previous works focused on the use of virtual machines as well as artificial intelligence techniques to develop solutions to control resource abundance and load balancing. Looking at this work, we see that the future global trend is towards the cloud with the undermining of the use of personal devices, so we can move gradually to the cloud considering that there are some containers that are sensitive to delays so they can be implemented at the edge. [5,7]

The following is the structure of this article: the related works are discussed in the next section. Section 1.3, article's Contributions. Section 2 provides background on the theoretical components of this work, whereas Section 3 describes the methodology. Section 4 describes the experimental results and section 5 describes the paper conclusion.

1.2 Related Work

Many container deployment studies concentrate on the placement of container infrastructure and cost considerations related to resource availability. Different types of resources are frequently required by different containers (CPU, RAM). Thus, there are many authors focused on this area because it contains an abundance information of on these topics. We review some of the research in resource management allocation.

Guo et al. [6] created a dynamic two-way delay analysis model to improve the edge resource problem by employing average container delay to give flexibility in container flow.Optimal use of the peripheral architecture is one of the most important reasons for the success of time-dependent systems .

The authors, Kaur et al. [8], introduce a new architecture for selecting and scheduling tasks at the edge of the network using a container as a service (CoaaS). The proposed algorithm reduces power consumption depending on various constraints such as memory, processor, etc. The process of using lightweight container technology instead of virtual machines has led to reduced redundancy and response time as well as reduced power consumption.

The author ZHOU et al. [9], introduces a project to increase the overall performance of AI (artificial intelligence) model training and inference, the EI (edge intelligence) idea was broadened to incorporate a model that fully utilizes available resources and data throughout the hierarchal of Cloud data centers, Edge nodes, and end devices. According to a six-level classification of EI architectures, The AI models can work in conjunction with Cloud–Edge devices. Moving from "Cloud Intelligence" to "All On-Device" reduces data offloading volume and path length, as well as transmission delay and bandwidth cost while increasing data privacy. As a result, while creating complicated IoT-oriented computing systems, Cloud and Edge resources should be utilized as needed.

Authors Rausch et al. [10] introduced the container scheduling system. This system makes trade-offs between data and computational movement and takes into account computing requirements such as processing layoffs. The results show that the scheduler greatly increases task placement quality when compared to Kubernetes' state-of-the-art scheduler and that the approach for fine-tuning the weights of scheduling constraints aids in fulfilling operational goals significantly.

Smimite and Afdel [11] developed a dynamic method in how RAM is used for the host and virtual machines in the data center to reduce unnecessary power consumption. The results showed that using containers instead of virtual machines saves energy, migration time, which affected the quality of service (QoS), and reduced service level agreement (SLA) violation when the suggested model was compared to alternative strategies. The movable node poses a substantial challenge for systems that rely on edge computing and container movement in the same context. It takes skill to solve a situation of this nature. In order to address this disease, several authors have followed their research paths.

The aforementioned description cites numerous studies on the application of load balancing of container technology, edge computing, and cloud computing in various contexts. As a result, the current study helps to provide solutions for task implementation and routing between the edge and the cloud while also considering energy conservation, resource allocation, and fault tolerance. With the suggested algorithm that relies on real-time to choose the suitable container orientation and failover in case of resource unavailability, this work is also a continuation of the work completed in earlier research.

1.3 Contributions

The following were the main goals that paper attempted to achieve:

- Create a resource allocation model for IoT devices.
- evaluate Docker container management technologies function on machines with constrained resources.
- Propose methods to reduce migration and management overhead depending on the type of workload and the availability of resources.
- Demonstrate the benefits of using edge computing to analyze data faster and more precisely.

2. Background

We'll go through the backdrop of edge computing and its analogues, cloud computing, and containers in this section.

2.1 Edge computing

Edge computing's particular advantages over traditional cloud computing (which relies on a centralized approach) have boosted its popularity among academics and researchers. Client data is handled as close to the originating source as possible on the network's edge in Edge computing distributed information technology architecture [12,15].

Edge computing puts storage and servers close to the data, requiring only a portion of the requirements to process data locally. Data flow normalization and analysis are frequently included in processing, with only the analysis conclusions being forwarded to the main data center [12,16].

2.2 Cloud computing

Cloud suppliers also include an assortment of services for IoT operations, turn out of the cloud is a popular choice for IoT deployments [16]. Although cloud computing provides away more resources and services than traditional data centers, the nearest territorial cloud facility can be hundreds of miles away from where data is gathered, and connections depend on the same fickle internet connectivity that upholds traditional data centers. In practice, cloud computing is used as a replacement for traditional data centers, or as a supplement in some cases. The cloud allows for closer centralized computing to a data source, but not at the edge of the network [17,18].

Edge and cloud computing have distinct advantages and disadvantages. These ideas are related to distributed computing and concentrate on the physical placement of computation and storage resources for the data being created. It's all about where those resources are positioned that make the difference. Fig.1 illustrates edge and cloud computing location [7], [19,22].

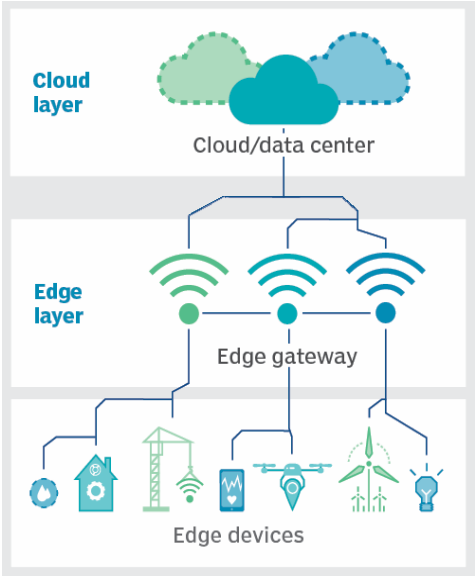

Fig.1. Edge and Cloud computing location.

2.3 Containerization

Containerization is a sort of virtualization in which applications run in discrete user regions called containers but share the same operating system. A container is a completely packaged and portable computing environment that includes encapsulates and isolates everything an application needs to run, including binaries, libraries, configuration files, and dependencies. As a result, the containerized program may be operated in the cloud without requiring any refactoring. Containerization reduces startup time and eliminates the need to set up a separate guest OS for each application. Containerization reduces startup time and eliminates the need to set up a separate guest OS for each application because they all run on the same OS kernel. [23,25]

Containerization is often used to package up the many different micro-services that make up modern apps because of their great efficiency. For creating and containerizing the application, container Engine is an open-source containerization tool. There are various container engines available, including Docker, LXC (Linux Containers), and others. Fig.2 illustrates containerization architecture. [26,30]

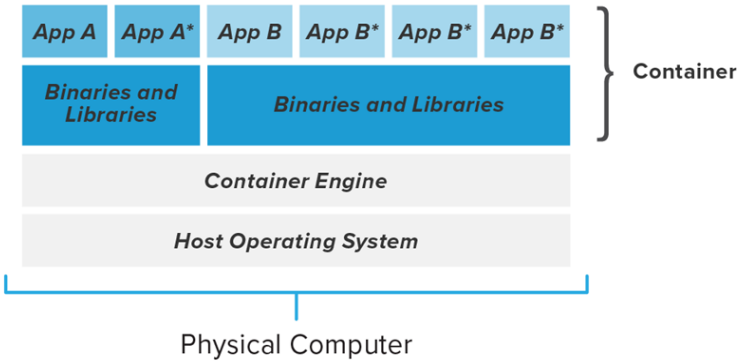

Fig.2. Containerization Architecture.

In this research, a set of tools were used that work in harmony with each other to reach the best possible environment in implementing the proposed algorithm. The tools such as containers, Docker, Flask, Tanserflow, Python micro-web framework are used to rout the containers between the cloud and the edge in this work.

3. Methodology

Since the concept of load balancing between edge nodes and the cloud is relatively new but widely applicable, the performance of containers on edge nodes has been investigated to improve data storage and transmission, to use them to improve applications in the manufacturing industry, medical assistance services, smart home systems, and others. This paper focuses on a specific aspect of advanced computing where software containers are installed on the edge nodes with the use of cloud nodes to reduce application and migration latency as much as possible and achieve load balancing.

The workload is supposed to be a data file. When a request is received by controller with the data file to be processed and the specified delay, the controller passes the request to a container working a data file processing. To rating, the CPU consumption for the required delay, a CPU utilization table, and expected delay is employed. By restricting the used amount of CPU by the container and gauging the length of time required to process different data files for different CPU Utilization, the CPU Utilization table and rated delay are formed. Table 1 represents the container workload which shows the CPU consumption.

We anticipated that by adjusting the number of resources supplied to each task, we might regulate the delay in data file categorization. As the latency decreases, we projected that CPU utilization would rise. The system structure is shown in Fig.3.

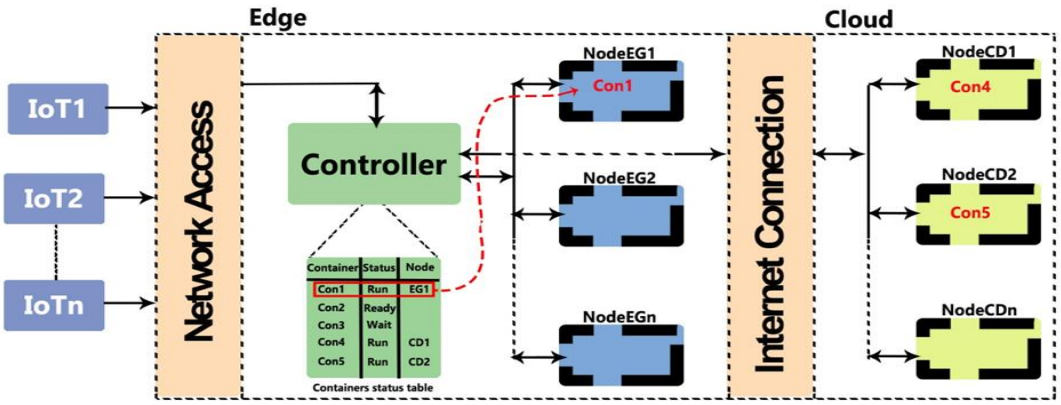

Fig.3. System Structure.

To process the data file, our project used the Object Detection API in Tensorflow. Flask is used to provide the Object Detection API in Tensorflow. Docker is used to containerizing the application.

The tasks consist of five data files with varying types and sizes. As the data file's type and size grow larger, so does the amount of time it takes to process it. We set a restriction on how much CPU the container could use and timed how long it took to analyze different files with different CPU usage. The data were fitted with an exponential model.

The addition of workloads collected from IoT devices to the table maintained in the controller is recommended using an algorithm. Then, the IoT device makes a request to the controller with the appropriate workload and delay, and controller are delivers the request to the container that will take up the task, as shown in Fig.3. The CPU utilization and estimated delay table, displayed in Table 1, is used to rating the CPU consumption for the required delay.

The maintained table is constructed by calculating the amount of CPU utilized by containers and calculating the time required to handle various workloads for various CPU usages. Essentially, the request can be delivered to the appropriate node if the predicted delay is smaller than the required delay.

Table 1. Container Workload Table

Container	Priority	Node	CPU usage	Execution time
Con1	1	Edge node 1	20%	40 msec
Con2	2	--------	15%	35 msec
Con3	3	--------	10%	30 msec
Con4	4	Cloud Node 1	12%	32 msec
Con5	5	Cloud Node 2	20%	40 msec

The controller receives the request from the terminal devices (IoT), specifying the CPU consumption and execution time. The function of the controller is to balance the loads between the edge and cloud nodes. Therefore, it distributes containers to edge and cloud nodes, taking the priority of the container to achieve good load balancing. Fig.4 shows the flowchart for the load balancing algorithm .

Depending on the resources that the container needs as well as the resources that are available in the nodes on the edge and in the cloud, the container is sent to the node that gives good execution of the container. A container that is delay-sensitive has a higher priority to execute on the edge node but if no suitable node is available on the edge it is sent to a node on the cloud.

In addition to the existence of an algorithm to determine the path of the container, there is another issue that takes an essential role in achieving load balancing is the abundance of resources. The abundance of sources gives greater flexibility in achieving load balancing. This can be seen by increasing node CPU usage.

A table is prepared with the list of available nodes, indicating the node place on the edge or the cloud. This list is supplied to the controller. The table also contains the state of each node (used, available), in addition to the possible CPU usage and a name of the container if the node is being used. Table 2 represents an example of the controller's list of nodes.

Table 2. Node location list

Node Name	Status	Container	CPU Available
NodeEG1	Used	Con1	30%
NodeEG2	Available	------	50%
NodeEG3	Available	------	40%
NodeCD1	Used	Con4	20%
NodeCD2	Used	Con5	30%
NodeCD2	Available	------	40%

NodeEG1: Node Edge 1, NodeCD1: Node Cloud 1

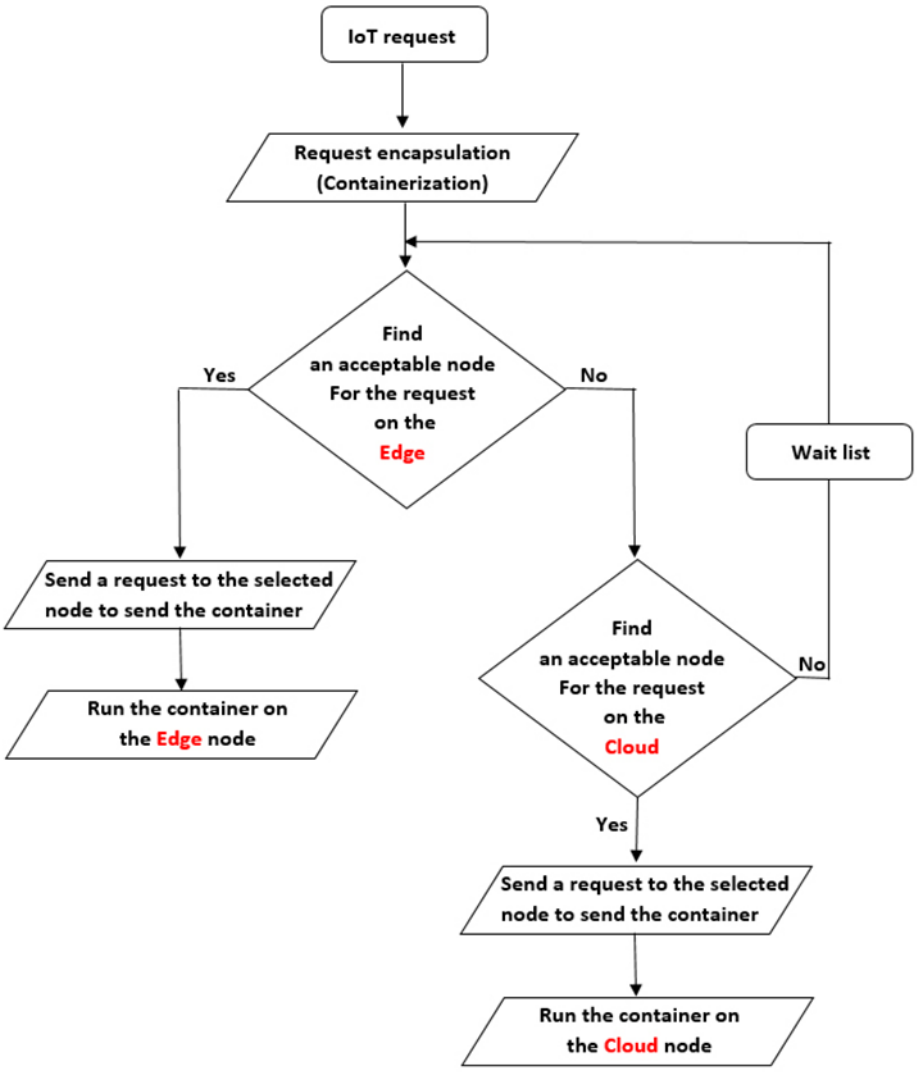

Fig.4. Load balancing flowchart

4. Experimental Results

Depending on the proposed algorithm shown in Fig.4, there are several possibilities. The algorithm depends on giving a chance to get the best load balance. The controller stores the state of all nodes within the system, whether they are on the edge or in the cloud. The status of the contract is updated periodically to ensure that there is no interference in the implementation and sending requests between the terminal devices and the controller on the one hand, and on the other hand between the controller and the nodes.

There are four cases :

Case 1: Edge nodes have enough resources to handle the incoming request. The new request is routed to the edge node with the highest CPU utilization.

Case 2: Edge nodes are unable to handle the incoming request due to a lack of resources. Sending the new request to the cloud node with the highest CPU utilization.

Case 3: If there aren't enough resources on edge and in cloud nodes, the request is forwarded to the waiting queue, where it will be given another chance to be executed.

Case 4: When more than one request is received at the same time, then a comparison will be made between requests based on the highest priority. Thus, allowing execution of the most important request.

Tests were conducted on the virtual environment by setting up containers running different-sized workloads with different CPU ratio allocations. These experiments were carried out under a variety of conditions to test our original hypothesis: as latency lowers, CPU utilization rises.

Initially, we assigned varied CPU percentages to each container with prioritization, such as 20, 15, 10, 12, and 20%, as shown in Table 1. When running files of various sizes, we see that the larger file, it takes the longer process, and the lower the CPU use, the longer it takes. The test bench was then expanded with more CPU utilization to get a

greater correlation in data. Each file was tested at 10, 20, 40, 75, and 90% CPU allocation and the time it took to analyze the file was plotted. There is a direct relationship when increasing the CPU utilization rate and the execution time of containers, as shown in Fig.5.

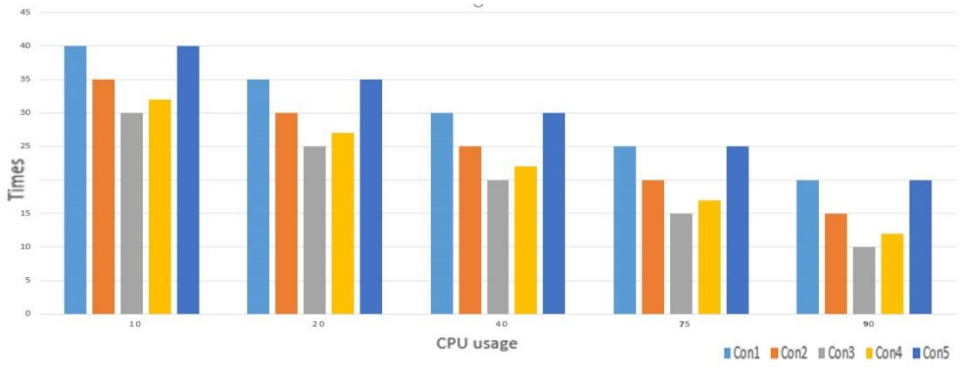

Fig.5. Performance evaluation of CPU usage.

5. Conclusions

The goal of this study was to look into the relationship between resources management allocation and load balancing. Edge nodes are always restricted to resources, but they are closer to the controller. Therefore, the execution of the delay-sensitive container on the edge node is better in terms of time consuming if the required resources from the container are available. Changing the amount of resources in the implementation of containers with the use of the proposed hypothesis, gives flexibility to get the best possible ways to achieve load balancing. This paper focuses on a specific aspect of advanced computing where software containers are installed on the edge nodes with the use of cloud nodes to reduce application and migration latency as much as possible and achieve load balancing. Our study focused on high-processing workloads and used Tensorflow's object discovery API, accessed using Flask, Python's micro-web framework. The experiment was conducted at different times and the results were similar. Docker container management technology was used in the implementation of this application. Also specify the amount of CPU usage when setting up the containers. CPU allocations were also tested, checking for delays by timing requests to the API, and measuring the time the request takes to process.

Acknowledgment

The author is grateful to the Department of Computer Science at the University of Mosul/ Iraq for supporting me in completing this work.

References

[1] F. A. Salaht, F. Desprez, and A. Lebre, “An Overview of Service Placement Problem in Fog and Edge Computing,” ACM Comput. Surv., vol. 53, no. 3, 2020, doi: 10.1145/3391196.

[2] M. Planeta, J. Bierbaum, L. S. D. Antony, T. Hoefler, and H. Härtig, “MigrOS: Transparent live-migration support for containerised RDMA applications,” 2021 USENIX Annu. Tech. Conf., pp. 47–63, 2021.

[3] S. P. Adithela, S. Marru, M. Christie, and M. Pierce, “Django content management system evaluation and integration with apache airavata,” ACM Int. Conf. Proceeding Ser., no. 2, pp. 3–6, 2018, doi: 10.1145/3219104.3229272.

[4] S. Maheshwari, S. Choudhury, I. Seskar, and D. Raychaudhuri, “Traffic-Aware Dynamic Container Migration for Real-Time Support in Mobile Edge Clouds,” Int. Symp. Adv. Networks Telecommun. Syst. ANTS, vol. 2018-Decem, no. December, 2018, doi: 10.1109/ANTS.2018.8710163.

[5] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, “Live Service Migration in Mobile Edge Clouds,” IEEE Wirel. Commun., vol. 25, no. 1, pp. 140–147, 2018, doi: 10.1109/MWC.2017.1700011.

[6] S. Guo, K. Zhang, B. Gong, W. He, and X. Qiu, “A delay-sensitive resource allocation algorithm for container cluster in edge computing environment,” Comput. Commun., vol. 170, no. January, pp. 144–150, 2021, doi: 10.1016/j.comcom.2021.01.020.

[7] D. R. Vasconcelos, R. M. C. Andrade, V. Severino, and J. N. De Souza, “Cloud, Fog, or Mist in IoT? That is the qestion,” ACM Trans. Internet Technol., vol. 19, no. 2, 2019, doi: 10.1145/3309709.

[8] T. Navigation and E. Technologies, “P Ort E Fficiency and T Rade,” no. November, pp. 48–56, 2017.

[9] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, “Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing,” Proc. IEEE, pp. 1–25, 2019, doi: 10.1109/JPROC.2019.2918951.

[10] T. Rausch, A. Rashed, and S. Dustdar, “Optimized container scheduling for data-intensive serverless edge computing,” Futur. Gener. Comput. Syst., vol. 114, pp. 259–271, 2021, doi: 10.1016/j.future.2020.07.017.

[11] O. Smimite and K. Afdel, “Containers Placement and Migration on Cloud System,” Int. J. Comput. Appl., vol. 176, no. 35, pp. 9–18, 2020, doi: 10.5120/ijca2020920493.

[12] S. Ketu and P. K. Mishra, "Cloud, Fog and Mist Computing in IoT: An Indication of Emerging Opportunities," IETE Tech. Rev. (Institution Electron. Telecommun. Eng. India), vol. 0, no. 0, pp. 1–12, 2021, doi: 10.1080/02564602.2021.1898482.
[13] S. Maheshwari, P. Netalkar, and D. Raychaudhuri, "DISCO : Distributed Control Plane Architecture for Resource Sharing in Heterogeneous Mobile Edge Cloud Scenarios," pp. 519–529, 2020.
[14] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez, "Edge-computing video analytics for real-time traffic monitoring in a smart city," Sensors (Switzerland), vol. 19, no. 9, 2019, doi: 10.3390/s19092048.
[15] K. Govindaraj and A. Artemenko, "Container Live Migration for Latency Critical Industrial Applications on Edge Computing," IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA, vol. 2018-Septe, no. iii, pp. 83–90, 2018, doi: 10.1109/ETFA.2018.8502659.
[16] A. Elgazar and K. Harras, "Teddybear: Enabling efficient seamless container migration in user-owned edge platforms," Proc. Int. Conf. Cloud Comput. Technol. Sci. CloudCom, vol. 2019-Decem, no. Section 2, pp. 70–77, 2019, doi: 10.1109/CloudCom.2019.00022.
[17] Z. Benomar, F. Longo, G. Merlino, and A. Puliafito, "Cloud-based Enabling Mechanisms for Container Deployment and Migration at the Network Edge," ACM Trans. Internet Technol., vol. 20, no. 3, 2020, doi: 10.1145/3380955.
[18] S. A. Bello et al., "Cloud computing in construction industry: Use cases, benefits and challenges," Autom. Constr., vol. 122, p. 103441, 2021, doi: 10.1016/j.autcon.2020.103441.
[19] S. B. Melhem, A. Agarwal, N. Goel, and M. Zaman, "A markov-based prediction model for host load detection in live VM migration," Proc. - 2017 IEEE 5th Int. Conf. Futur. Internet Things Cloud, FiCloud 2017, vol. 2017-Janua, pp. 32–38, 2017, doi: 10.1109/FiCloud.2017.37.
[20] S. V. N. Kotikalapudi, "Comparing Live Migration between Linux Containers and Kernel Virtual Machine : Investigation study in terms of parameters," no. February, p. 42, 2017, [Online]. Available: www.bth.se.
[21] A. Dhumal and D. Janakiram, "C-Balancer: A System for Container Profiling and Scheduling," pp. 1–10, 2020, [Online]. Available: http://arxiv.org/abs/2009.08912.
[22] L. Deshpande and K. Liu, "Edge computing embedded platform with container migration," 2017 IEEE SmartWorld Ubiquitous Intell. Comput. Adv. Trust. Comput. Scalable Comput. Commun. Cloud Big Data Comput. Internet People Smart City Innov. SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI 2017 - , pp. 1–6, 2018, doi: 10.1109/UIC-ATC.2017.8397578.
[23] O. Oleghe, "Container Placement and Migration in Edge Computing: Concept and Scheduling Models," IEEE Access, vol. 9, pp. 68028–68043, 2021, doi: 10.1109/ACCESS.2021.3077550.
[24] A. Barbalace, M. L. Karaoui, W. Wang, T. Xing, P. Olivier, and B. Ravindran, "Edge computing: The case for heterogeneous-ISA container migration," VEE 2020 - Proc. 16th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Exec. Environ., pp. 73–87, 2020, doi: 10.1145/3381052.3381321.
[25] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live Service Migration in Mobile Edge Clouds," IEEE Wirel. Commun., vol. 25, no. 1, pp. 140–147, 2018, doi: 10.1109/MWC.2017.1700011.
[26] S. Maheshwari, P. Netalkar, and D. Raychaudhuri, "DISCO : Distributed Control Plane Architecture for Resource Sharing in Heterogeneous Mobile Edge Cloud Scenarios," pp. 519–529, 2020.
[27] Saxena, Shailesh, Mohammad Zubair Khan, and Ravendra Singh. "Green computing: an era of energy saving computing of cloud resources." Int. J. Math. Sci. Comput.(IJMSC) 7.2 (2021): 42-48.
[28] Alakberov, Rashid G. "Clustering Method of Mobile Cloud Computing According to Technical Characteristics of Cloudlets." International Journal of Computer Network & Information Security 14.3.(2022)
[29] Kaur, Amanpreet, et al. "Load balancing optimization based on deep learning approach in cloud environment." International Journal of Information Technology and Computer Science 12.3 (2020): 8-18.
[30] Hadeed, Wael W., and Dhuha Basheer Abdullah. "Real-Time Based Big Data and E-Learning: A Survey and Open Research Issues." AL-Rafidain Journal of Computer Sciences and Mathematics 15.2 (2021): 225-243.

Authors' Profiles

Wael Hadeed Received his M.Sc. degree in Computer Science from the University of Mosul in 2010. He is working as a lecturer at the Department of Computer Science, University of Mosul, Iraq. His current research interests include containerization, Big Data and cloud computing.

Dhuha Basheer Abdullah Dean of Computers and Mathematics College, University of Mosul. She received her Ph.D. degree in computer sciences in 2004 in the specialty of computer architecture and operating systems. She supervised many Master and Ph.D. degree students in Operating System, Computer Architecture, Dataflow Machines, Mobile computing, Real-Time Systems, Distributed Systems, Cloud Computing, and Big Data. She also leads and teaches modules at BSc, MSc, and Ph.D. levels in computer science. Also, she teaches many subjects for Ph.D. and master students.

How to cite this paper: Wael Hadeed, Dhuha Basheer Abdullah, "Load Balancing Mechanism for Edge-Cloud-Based Priorities Containers", International Journal of Wireless and Microwave Technologies(IJWMT), Vol.12, No.5, pp. 1-9, 2022. DOI:10.5815/ijwmt.2022.05.01