Computers, Materials & Continua
DOI: 10.32604/cmc.2023.032474
Article

Tech Science Press

Vertical Pod Autoscaling in Kubernetes for Elastic Container Collaborative Framework

Mushtaq Niazi[1], Sagheer Abbas[1], Abdel-Hamid Soliman[2], Tahir Alyas[3], Shazia Asif[4] and Tauqeer Faiz[5,*]

[1]School of Computer Science, National College of Business Administration and Economics, Lahore, 54000, Pakistan
[2]School of Digital, Technologies and Art, Staffordshire University, ST42DF, UK
[3]Department of Computer Science, Lahore Garrison University, Lahore, 54000, Pakistan
[4]Higher Colleges of Technology, Abu Dhabi, UAE
[5]Department of Enterprise Computing, Skyline University College, Sharjah, UAE
*Corresponding Author: Tauqeer Faiz. Email: tauqeer.khan@skylineuniversity.ac.ae
Received: 19 May 2022; Accepted: 21 June 2022

Abstract: Kubernetes is an open-source container management tool which automates container deployment, container load balancing and container(de)scaling, including Horizontal Pod Autoscaler (HPA), Vertical Pod Autoscaler (VPA). HPA enables flawless operation, interactively scaling the number of resource units, or pods, without downtime. Default Resource Metrics, such as CPU and memory use of host machines and pods, are monitored by Kubernetes. Cloud Computing has emerged as a platform for individuals beside the corporate sector. It provides cost-effective infrastructure, platform and software services in a shared environment. On the other hand, the emergence of industry 4.0 brought new challenges for the adaptability and infusion of cloud computing. As the global work environment is adapting constituents of industry 4.0 in terms of robotics, artificial intelligence and IoT devices, it is becoming eminent that one emerging challenge is collaborative schematics. Provision of such autonomous mechanism that can develop, manage and operationalize digital resources like CoBots to perform tasks in a distributed and collaborative cloud environment for optimized utilization of resources, ensuring schedule completion. Collaborative schematics are also linked with Bigdata management produced by large scale industry 4.0 setups. Different use cases and simulation results showed a significant improvement in Pod CPU utilization, latency, and throughput over Kubernetes environment.

Keywords: Autoscaling; query optimization; pods; kubernetes; container; orchestration

1 Introduction

Emerging technologies are reshaping the digital ecosystem at a very fast pace. New platforms, tools, and hardware rapidly take the digital scenario to the next level. This swift pace, versatility

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

of digital entities and increasing demand need careful attention towards autonomous and dynamic interfacing to reduce the complexity. Cloud computing plays a central role and allied technologies to address the need for autonomy and evolving interfacing. Cloud technology is getting complex and requires significant skill to configure and deploy professional solutions. On the other side widespread internet of things (IoT) has created another challenge in terms of data versatility and volume that is also transforming into big data. The current digital ecosystem is taking advantage of artificial intelligence, but it also requires to adhere the challenges brought by Industry 4.0 and robotics [1].

The emergence of internet and World Wide Web gradually transformed into a more productive, distributed and cost effective platform i.e., cloud computing. It is a platform that provides IT resources on demand through internet connectivity. It has become phenomenal due to the, cost-effectiveness and availability of services globally. Instead of bearing the infrastructural costs and engaging a physical data center, cloud computing has provided a more straightforward managed services solution with a pay-as-you-use concept accompanied by easy to use services and professional Management of cloud structure. Many international players provide cloud computing including IBM, Microsoft, Amazon, Google and Oracle [2].

These categories provide various services and on-demand configurable computation capacity, storage facilities, and high-tech network; therefore, cloud platform is becoming a highly suitable platform for individuals, small groups, small-to-medium enterprise (SME)s, and large size organizations. Though such versatile configurations made cloud management significantly complex, distributed computing reaches a new level. In cloud computing, infrastructure and application are managed simultaneously using virtualization technique that makes it possible to generate multiple virtual machines to provide network, compute and storage to a huge number of users without compromising quality and security [3].

Software as a service (SaaS) is owned and managed remotely by the service provider, allowing us to connect and use cloud based apps over the internet. The service provider manages all parts of the applications, such as data centre, networking, and operating system. End user only focus on the usage of the application. Off-the-shelf applications are access over the internet like google apps, spreadsheet. SaaS allow user to run existing online applications, accessible from any computer via internet, facilitate collaborative working [4].

Platform as a service (PaaS) gives us a development framework. PaaS allows user to create their on-cloud applications. PaaS helps us to build, test, deploy, and manage applications e.g., google apps engine. Infrastructure as a Service (IaaS) enables users to run applications as they want on their own cloud hardware. IaaS allows existing applications to be run on cloud suppliers' hardware. Existing applications can be moved from the data centre to the cloud environment. The fundamentals unit of the cloud is a server. The server can be physical or virtual. Physical servers are individual computers, in contrast, virtual servers' instants are software-controlled spices for physical services [5].

These cloned virtual entities are virtual machines which are generated from a common hardware resource but maintain complete isolation. Virtualization has make this possible to reduce redundant hardware, configurations and maintenance while extending better management in a secure distributed environment. It also became possible to configure versatile computing environments and platforms on various virtual machines and link them together transparently. Virtualization allows many users to share one physical server. Virtualization is making feasible with hypervisor, a software that run on physical hosts or servers. There are two type of hypervisor one can be directly installed on the top of the physical servers, also called bare metals hypervisor, more secure and low latency. Second type of

hypervisor is based on the hosts operating system (OS) are called hosted OS. These are less frequent, mainly used for user virtualizations like Oracle virtual box, VMware Workstation [6].

Developer initiate by accessing the Docker Hub, cloud repository of Docker Hub containers. Containers can be easily paused, resumed and removed. Container generally perform a single operation like PostgreSQL database, new javascript (JS) application, and network simultaneously to potentially scale. Unlike the VMs in containers resources are directly shared to host. This allows users to run many Docker's containers using the less disk space [7].

The utilization of these facilities needs professional tools like Kubernetes. Kubernetes [8] is a platform for working with containers and help us with the deployment, scaling and monitoring of containers. Kubernetes have controller nodes and worker nodes. Nodes is physical or virtual and can contain containers. If something goes wrong with the node container will not be available. In Kubernetes, we launch containers on more than one node. A Group of nodes called a cluster would be managed by our master node. Master nodes consist of application programming interface (API) server, controller, schedulers and key-value store [9].

IoT in assembly unit is the solution family that incorporates the ability to value chains from analytics, cognitive computing, and operational performance. In the automotive sector, this approach has three advantages. Finding and solving problems before casting delays helps to plan out and get 100 percent productivity out of their equipment [10]. Second of all it allows operations as well as cognitive processes, so plans can achieve optimum quality and yield from raw material and components of production. Finally, it allows managers to prepare the management of resources to enhance worker skills and provide a stable working atmosphere. A new way of connected manufacturing that revolves objects, processes and people was characterized by advancements in these three fields [11].

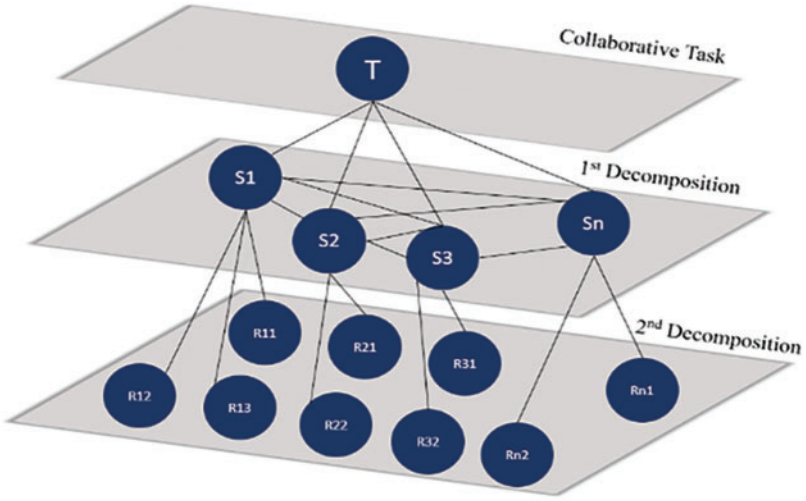

Figure 1: Collaborative task distribution model

Third is supply chain management beyond the factory the artificial intelligence (AI) can anticipate in the supply chains with predictive algorithms. AI is essential to reach the cutting edge of industrial automation design.

Big data is divided into structured, unstructured, and semi-structured data. Structured data refers mostly to traditional data sources saved and organized in a database. Unstructured has no defined data models and clear storage format and is difficult to search. The line between semi-structured and

structured data has always been vague. It is a type of structured data but lacks a strict data model structure, which helps analyze it easily [12].

Machine-related data are facts and figures from sensors, weblogs, financial systems, medical instruments, Global Positioning System (GPS) data, data usage statistics, satellite images, scientific data, and radar data. The 'API server' provides an application programme interface through which Kubernetes provides its functionality as shown in Fig. 2. The 'controller' is a master component that monitors the cluster's status via the 'API server' and transforms the current state to the desired state. The 'scheduler' is the control plane component in charge of scheduling pods in several nodes. The 'etcd' database is a key-value based database that maintains all Kubernetes cluster configuration information. Users communicate with the master node's 'API server' via the command-line utility 'Kubectl.' Practitioners can give configuration-related information in the form of configuration files, such as yet another markup language (YAML) and javascript object notation (JSON) files, while deploying Kubernetes. These files include important information about the Kubernetes installation's network, central processing unit (CPU), and memory settings [13].

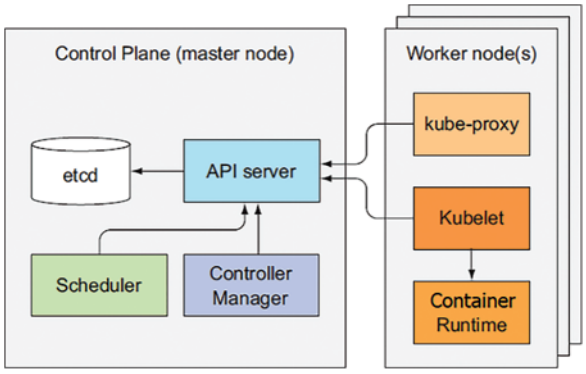

Figure 2: Kubernetes architecture

Data from different sources are routed to one system, and for velocity, we require faster massive data processing. Similarly, variety refers to structured, semi-structured, and unstructured data, while veracity means accuracy and trustworthiness. All this data will benefit the relevant sector by performing its analytics, known as the value of data [14].

2 Related Work

For scientific workflows, infrastructure automation techniques reduce complications related to repeatable infrastructures working and participating in scientific experiments that are reproduced able. The business that had already proposed virtualization can incorporate database servers into fewer domain controllers and benefit from multidimensional feature space, power, and requirements needed. Optimization use is expected to increase efficiency and system responsiveness for consumers without increasing server utility functionality. Virtualization allows us by using containers technology which is much elasticity for capacity management in a server. Due to the overhead delay in scaling containers, main module on various computers takes longer than that on a dedicated processor [15]. Throughout this studies, we had did multiple scenarios to evaluate the CPU load on single and numerous data centres.

A new term in software architecture patterns is the micro-service approach, which is getting famous due to its elasticity, granular approach and loosely linked services. An API Gateway system is a scaling system in Kubernetes and Prometheus that adjust things. Varley the number of Instances that occurs during the course of action. It can enhance the use of system resources while ensuring the application high availability and quality of service. The Auto Scaling System can effectively guide the number of service cases according to the API Gateway stack. When its load exceeds the specified procedure, more instances are created with dynamism to balance the workload [16].

Distributed technology is a term that adds new application development stacks to asset virtualization. Due to this sense of expanding the load on the data centres, the facilities are not efficiently manipulated. They explained what Docker and container are, allowing us better to understand the technology of Docker swarm and Kubernetes. Docker is a phase of the program that allows you to produce, test, and quickly send applications. Docker combines computing into institutional units that are deemed holders that do everything the product required to develop, such as code, library resources, configuration, system and systems [17]. Finally, with the help of the Docker containers and Kubernetes, the author demonstrates how the services are accessed by the node in a cluster and provides the distinction between them.

Kubernetes is an innovation compartment for managers created in Google Experiment to monitor container-based applications in various circumstances, such as physical, virtual, and cloud structures. It is open-source and helps to create and supervise the use of container shipping. Kubernetes enables healing to enhance the quality and comprehensive through disappointment recovery actions. Since our primary objective is to build configurations with Kubernetes for middleware applications to enable high availability, we investigate the consistency obtained by Kubernetes underneath the default settings in this document. We have carried out several studies that demonstrate that the outage of the service can be substantially greater than anticipated [18].

Through malfunction following the decision, Kubernetes makes healing and internal communications also test them. For these kinds of programs, relative to their sensitivity to errors arising from external causes, Kubernetes responds relatively strongly. An intelligent development device based on Industry 4.0 in such a sunroof light source development system [19]. The implemented application begins with the process of transformation to convert the current manufacturing environment to become competent by installing real-time data capture systems, specifically radio frequency identification (RFID) systems, to convert the organizations of the entire systems into smart objects in the process flow and become wirelessly highly contagious to achieve an intelligent manufacturing ability to track and control the entire structure smartly. Manual working is not acceptable to apply the mechanism of industry 4.0 implementation. Machine learning is one example of a computational method that substitutes data-based assessment for human work requiring subjective judgment [20].

Industry 4.0 the paradigm means a reduction in the sensitivity of industrial processes, an increase in the machine's long career, and an improvement in the standard of products. The data from the entire production process is the fourth industrial revolution's fundamental principle of this 4th industrial revolution. Techniques that work on many data collected from the field by many sensors are usually automatically extracted [21]. Sensors and data search seem to be an imminent part of the booming Internet of Things process. The Internet of Things (IoT) will allow physical objects to function as intelligent pervasive computing that could recognize events and changes in their external environment and, without any human interference, accumulate, methodology and communicate the information gathered [22].

In cloud computing, another problem is the complexity of collaborative schemata considering the distributed task allocation. The design structure collaboration is different from functional collaboration and collaborative task assignment. Similarly, it requires an action performing schema with a feedback structure schema to carry on the collaborative schematics in the cloud service platform. Various frameworks for decomposing collaborative schemas as shown in Fig. 1 and re-structuring such schemas are an important research area to look into in applications like CoBots [23].

3 Proposed Methodology

The proposed model consists of four layers of multiple components i.e., Structure, Management, Execution and Analytics. Structuring and management is linked to the execution layer where individual team members shall perform each task, this layer is cloned using virtualization, therefore, it shall be executed on multiple instances. Considering the problem statement, it is required to deal with the dynamic nature of collaborative schema; therefore, on cloud platform, segregation of relevant structuring is vital to formulate cloud structure for adaptable, collaborative schema including cognitive correlates with ensuring the learning phenomenon. Similarly, collaborative tasking needs a management module that shall cater to multiple task scheduling, allocation and sub-tasking to achieve individual task goals. This module shall also manage the taskers/team members, it is notable that these team members may be devices, bots, or soft agents to generalize this model for multiple scenarios where collaborative tasking is involved. Finally, the proposed model provides a complete layer for analytics to manage neural networks modifications and suitable configurations for learning.

The use case we will be using for collaborative robot (CoBots), which are becoming an important constituent of industry 4.0 and industrial robotics. CoBots shall develop a working ecosystem and collaboratively perform tasks to attain maximum productivity and coherence, essential in industrial robotics. In CoBots, it is required to deploy a collaborative schema to execute tasks in parallel, sequential and hierarchical modes. As in human tasking there is a need to provide task swaps, time management and skill development in terms of knowledge sharing and exchange. Beyond this use case, the proposed model is generalizable for IoT devices and machine to machine collaboration management in the cloud environment.

The Proposed structure as shown in Fig. 3 module contains a control layer consisting of project, configuration and task linked with a communication controller to manage various modes of communication. Further linkage is with Tools/Applications for project mapping to activate Dockers for respective service activation. Control layer provides configuration to project module in the management layer. The Controller manages control layer to formulate cloud structure consisting of API, etcd, Kubernets and a scheduler while the controller layer provides segregated information to monitor where projects, tasks, swapper, time adjustments in case of swapping are available. Multiple projects and tasks shall generate patterns and clusters of various project configurations. The structure layer is supported by cognitive correlations that include memory structure to cater to the cloud's knowledge base. The constituents of cognitive Memory contain long term memory for archiving short term memory for instance, management, while semantic are available to provide association and understanding of the content. Cognitive Memory and monitor are linked with the learner in an analytical module, which provides a neural network for machine learning required for semantics, preferences-as-memory (PAM), and pattern/cluster make.

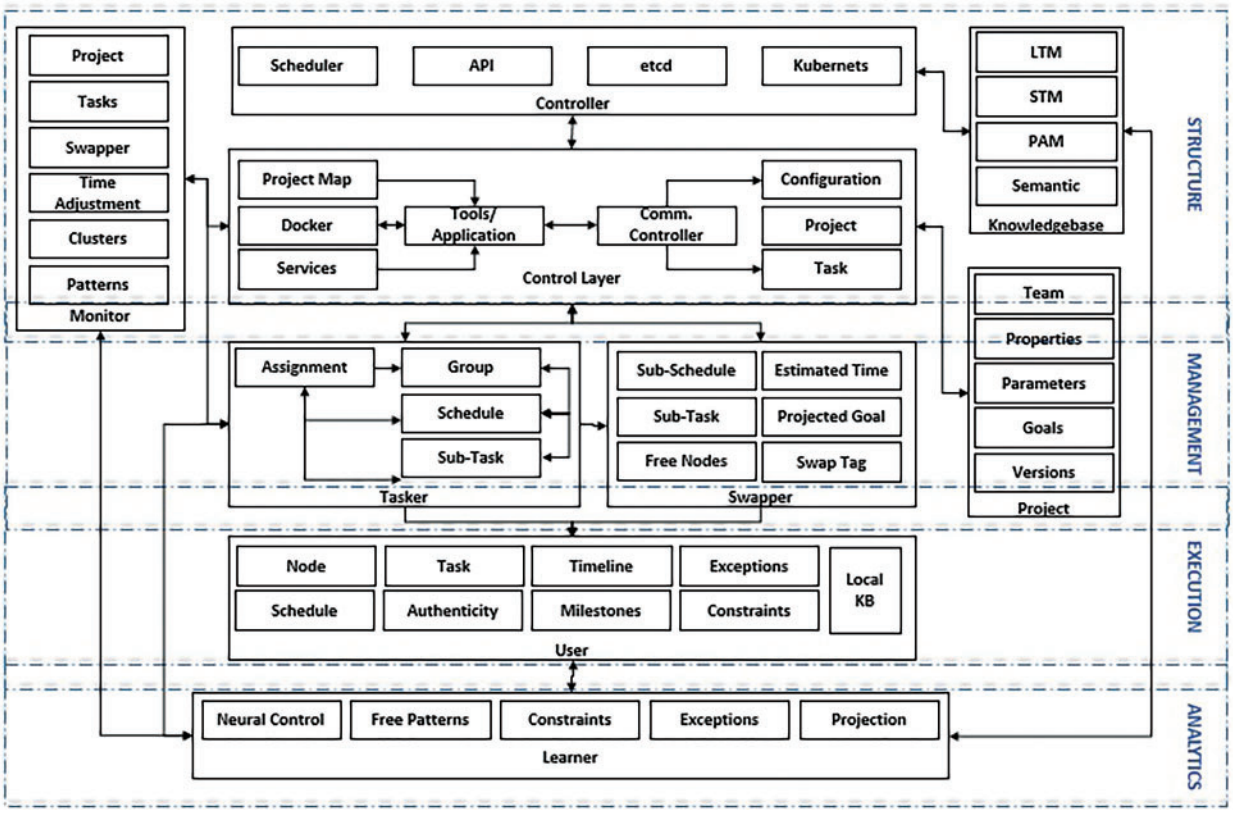

Figure 3: Proposed system model

The management layer is divided into three main components i.e., tasker, swapper and project. Tasker and swapper are receiving information from the control module of the structure layer, while the project is an overlapping module between the structure and management layer containing the team, properties, parameters, goals and versioning. Management layer distributes tasks to execution layer to multiple virtual users or CoBots. Tasker is splitting into assignment with a schedule containing the estimated time for completion while it also contains sub-tasking and may create groups to perform assignment, accordingly team members will get allocations. A complex routine in the collaborative schema is swapping of the tasks/assignments. Swapper provides free team members/nodes and shall develop a sub-schedule in case of swapping, here we treat swapping as a sub-task to the target swap node/member. Every instance of swap contains swap tag to make source and target linked along with the projected goal of the swap and estimated time, which in ideal conditions shall not disturb the primary task schedule but in case of additional time required, that will be adjusted by control layer. Tasker is linked with the monitor module, memory module, and control module in the structure layer and supported by learner module from the analytics layer.

The execution layer contains user modules that work in the environment developed by the structure layer and perform operations developed and configured by the management layer. As mentioned earlier, the execution layer and user module will work virtually in multiple users in terms of CoBots or IoT devices which will act as team members/nodes in a project. The management layer handles the collaborative schema, but we also provide a local knowledge base to enable every member to have its own repository to store the experience and skill it is learning while performing the tasks.

The user module contains a task, affiliated nodes/members, and a schedule. For inter-exchange of tasks and communication, authenticity modulation is also available to ensure an encapsulated environment for each node which is shareable only if authorized or having consensus between two nodes/members. The user module will make milestones and timelines of those milestones for each task and record exceptions and constraints that may occur during the execution of the tasks. Therefore, if the skillset required to perform a certain task is not available or insufficient, it will first consult the learning module or communicate with other nodes to share their local knowledge base if it gets authorization. In this manner the collaborative schema will also cater to implicit collaboration at knowledge level and explicit collaboration at functional level. In case of such constraints, a swapping option is available to hand over or share the task with another free node.

The final layer of the proposed module is analytics which is getting data from all modules to formulate constraints and exceptions for the structure layer to develop clusters and patterns, which will be used to develop more adaptable configurations for future projects and collaborative schemas. This layer will also provide projections for the skill threshold level of local knowledge bases and the structure knowledge base. All these components are encapsulated in a learner module provided with a neural control which will provide machine learning to perform analytics.

4 Results & Discussion

Case 1:

In a physical machines that runs on Intel(R) Core (TM) i9–12900F @ 5.10 GHz ∗ 16 Intel® Iris® Xe Graphics eligible, We set up a Kubernetes cluster with five, one master and four worker nodes. Each cluster node is a virtual machine running Ubuntu 20.04.4 LTS, Docker version 20.10.13, and Kubernetes version 1.23.5. In terms of computational power, the master has four core processors and sixteen gigabytes of random access memory (RAM), while each of the worker nodes has two core processors and two gigabytes of RAM. Furthermore, the load generator is Gatling open-source version 3.7.6, On each worker node, an HTTP request is delivered to our application via a configured NodePort [24].

Our program is built to use a lot of CPU resources. In other words, it requires CPU resources until it sends back a response to the sender after successfully receiving an HTTP request. Each replica CPU request and limit are 100 and 200 m, respectively. The number of copies varies from 4 (average of 1 replica per node) to 24 (average of 6 replicas per node).

Every experiment is conducted for 300 s. Gatling average inbound request rate is around 1800 requests/s for the first 100 s, then 600 requests/s for the next 100 s, resulting in a total of 240,000 requests. The terms we use to describe these two times are high traffic period (HTP) and low traffic period [25] . The remaining simulation period is spent analyzing the decline in metrics when no requests are arriving. We put Kubernetes HPA to the test in seven distinct tests in this study. To assure accuracy, each experiment is performed ten times.

In Fig. 4, Y-axis describe about the percentage of the CPU usage at the specified moment The horizontal axis shows progression over time in last 24 h. The average CPU usage increases between 8:00 to 12:00 h of execution time because of the high rate of requests. after that, it decreases as the number of requests is balanced on other nodes.

Fig. 5 Percentages of disk space used were about 55% across the nodes. On X-axis there is timeline for which we have executed.

In Fig. 6 vertical axis of the graph shows the bandwidth used in megabytes per second. The horizontal axis shows progression over time in last 24 h. Read Throughput increases between 8:00 to 12:00 h as the number of request from user side increases upto to 15 Mega Bytes per second.

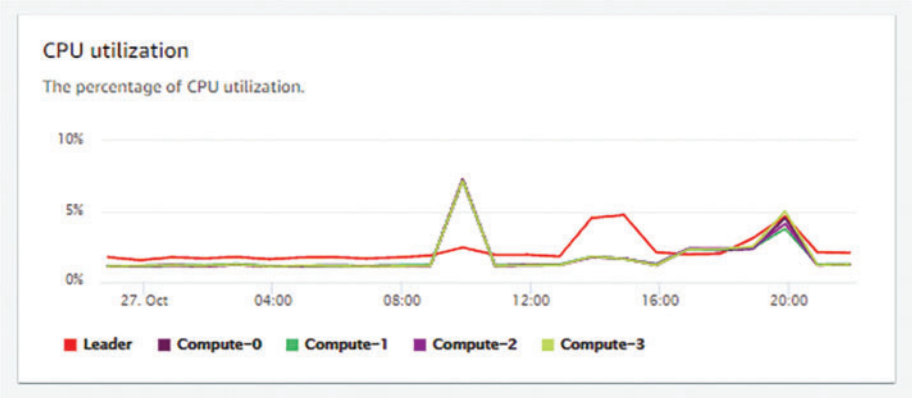

Figure 4: CPU utilization

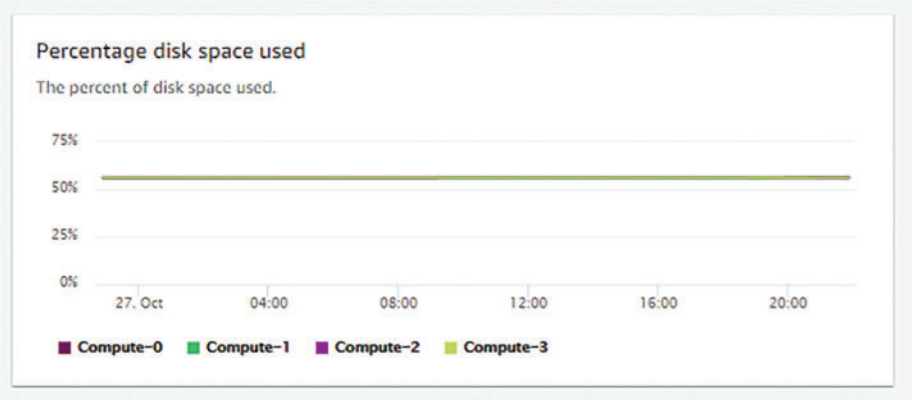

Figure 5: Storage performance

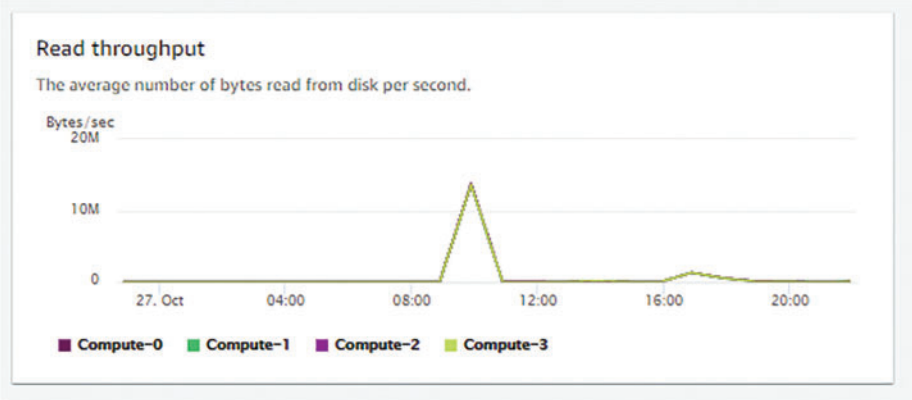

Figure 6: Worker node throughput

In Fig. 7 vertical axis of the graph shows, read latency for disk read I/O operations in seconds. The horizontal axis shows progression over time in last 24 h. Read Latency of Compute 0, 1, 2 is 10 s while Compute 3 Read Latency is 5 s Average for disk read I/O operations.

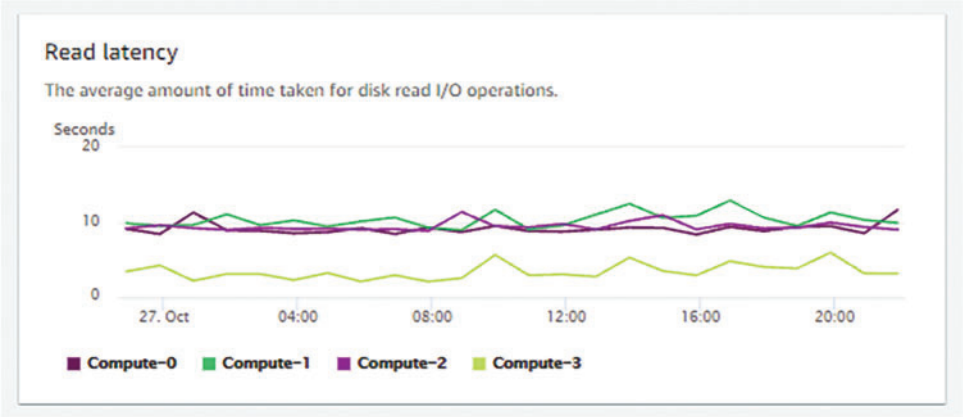

Figure 7: Worker node latency

In Fig. 8 The vertical axis of the graph shows the write throughput for the average number of bytes written to disk per second. The horizontal axis shows progression over time in last 24 h. Write throughput increases between 8:00 to 12:00 h as the number of request from user side increases up to to 11 Mega Bytes per second.

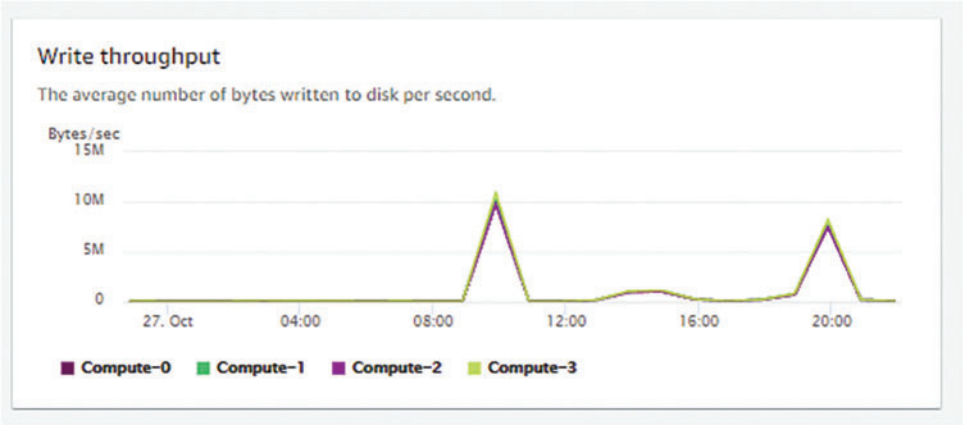

Figure 8: Worker node latency

Case 2:

We have a nextcloud application that is deployed to amazon web services (AWS) cloud compute instances. We have four amazon elastic compute cloud (EC2) instances with three replicas in three different availability zone. Each ec2 instance type is m5.xlarge, vCPU 4, Memory 16GB, Instance Storage type is amazon elastic block store (EBS), Network Bandwidth up to 10 Gbps, EBS Bandwidth up to 4750 Mbps. Nextcloud application is deployed to these EC2 instaces. We have configured a web based load balancer infront of these EC2 instances to manage the web traffic load.

In Fig. 9 The vertical axis of the graph shows the CPU utilization percentage. The horizontal axis shows progression over time in last 60 days. Maximum CPU utilization increased upto 45.6% as the number of request from user side increases.

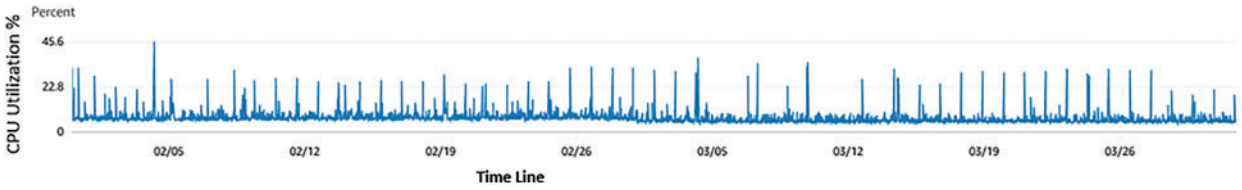

Figure 9: Compute I maximum CPU usage

In Fig. 10 The vertical axis of the graph shows the CPU utilization percentage. The horizontal axis shows progression over time in last 60 days. Average CPU Utilization remained upto 5%–6% during the application processes in last 60 days.

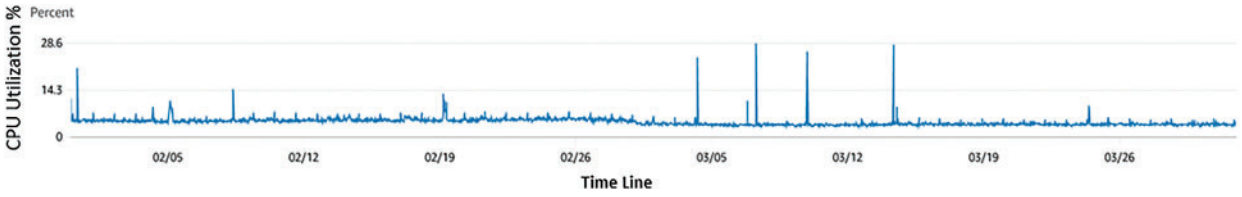

Figure 10: Compute I average CPU usage

In Fig. 11 The vertical axis of the graph shows the Memory usage percentage. The horizontal axis shows progression over time in last 60 days. Maximum Memory used reahed upto 33.3% during this time interval of 60 days.

Figure 11: Compute I max memory usage (c)

In Fig. 12 The vertical axis of the graph shows the Memory usage percentage. The horizontal axis shows progression over time in last 60 days. Average Memory usage remained 17%–18%.

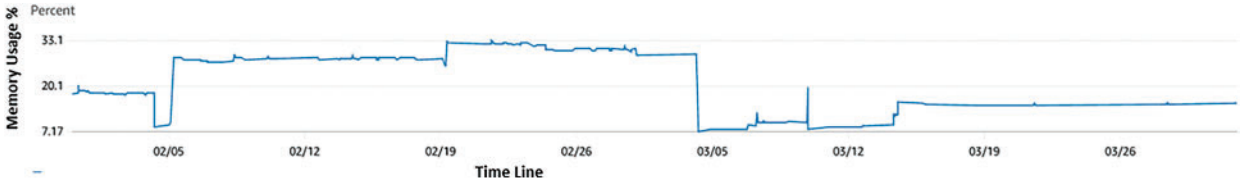

Figure 12: Compute I average memory usage (d)

In Fig. 13 The vertical axis of the graph shows the CPU utilization percentage. The horizontal axis shows progression over time in last 60 days. Maximum CPU Utilization increased upto 46.3% as the number of request from user side increases.

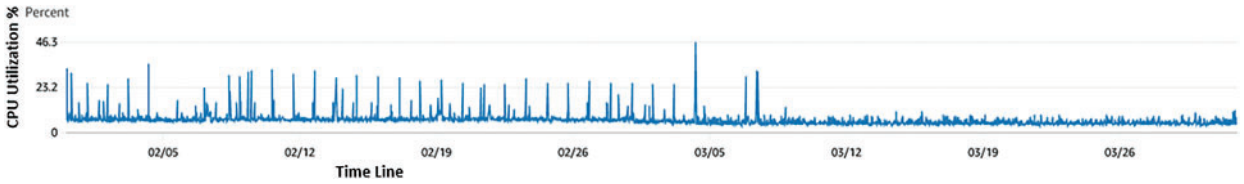

Figure 13: Compute II Maximum CPU usage (a)

In Fig. 14 The vertical axis of the graph shows the CPU utilization percentage. The horizontal axis shows progression over time in last 60 days. Average CPU Utilization remained upto 5%–5% during the application processes in last 60 days.

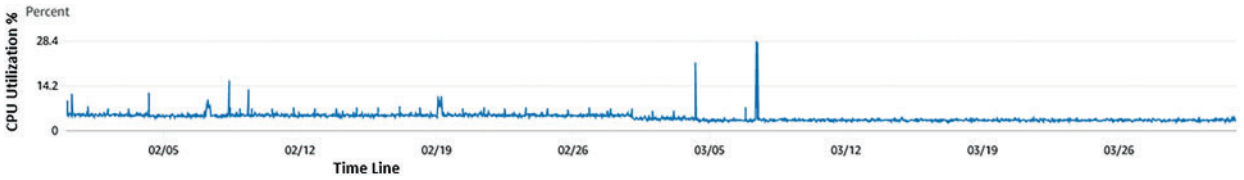

Figure 14: Compute II average CPU usage (b)

In Fig. 15 The vertical axis of the graph shows the Memory usage percentage. The horizontal axis shows progression over time in last 60 days. Maximum Memory used reahed upto 33.8% during this time interval of 60 days.

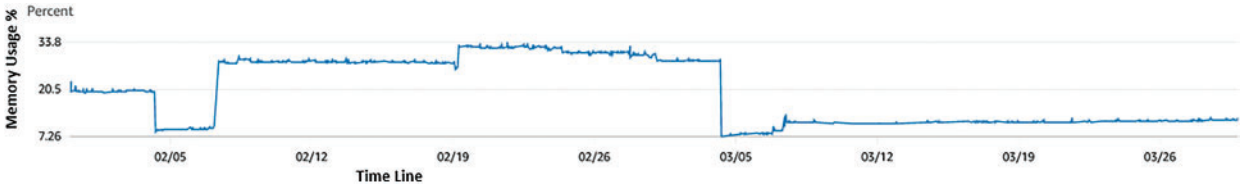

Figure 15: Compute II maximum memory usage (c)

In Fig. 16 The vertical axis of the graph shows the Memory usage percentage. The horizontal axis shows progression over time in last 60 days. Average Memory usage remained 9%–10%.

ComputeIII

Figure 16: Compute II average memory usage (d)

In Fig. 17 The vertical axis of the graph shows the CPU utilization percentage. The horizontal axis shows progression over time in last 60 days. Maximum CPU Utilization increased upto 45.6% as the number of request from user side increases.

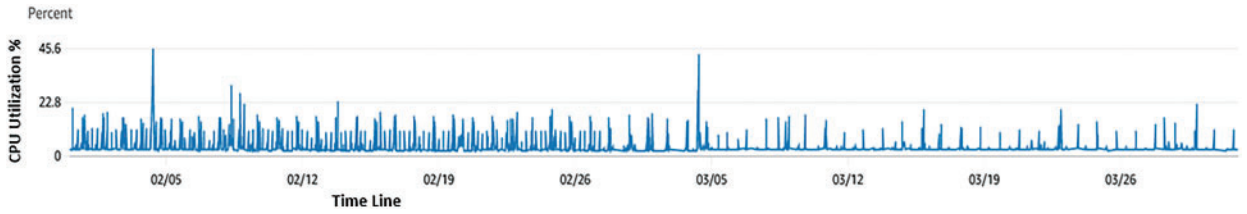

Figure 17: Compute III maximum CPU usage (a)

In Fig. 18 The vertical axis of the graph shows the CPU utilization percentage. The horizontal axis shows progression over time in last 60 days. Average CPU Utilization remained upto 5%–6% during the application processes in last 60 days.

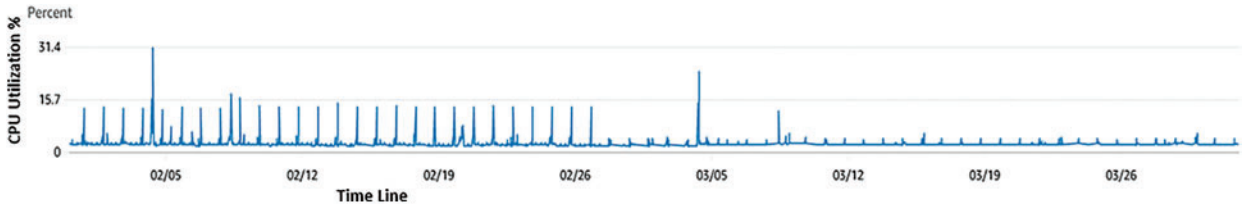

Figure 18: Compute III average CPU usage (b)

In Fig. 19 The vertical axis of the graph shows the Memory usage percentage. The horizontal axis shows progression over time in last 60 days. Maximum Memory used reahed upto 33.3% during this time interval of 60 days.

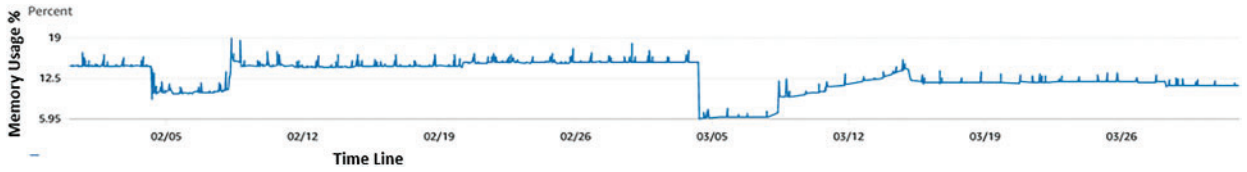

Figure 19: Compute III maximum memory usage (c)

In Fig. 20 The vertical axis of the graph shows the Memory usage percentage. The horizontal axis shows progression over time in last 60 days. Average Memory usage remained 17%–18%.

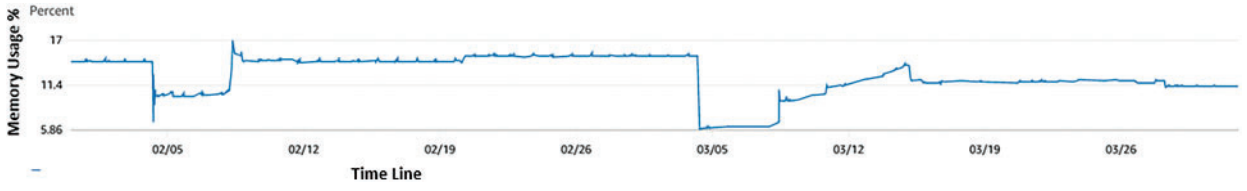

Figure 20: Compute III average memory usage (d)

In Fig. 21 The vertical axis of the graph shows the CPU utilization percentage. The horizontal axis shows progression over time in last 60 days. Maximum CPU Utilization increased upto 45.6% as the number of request from user side increases.

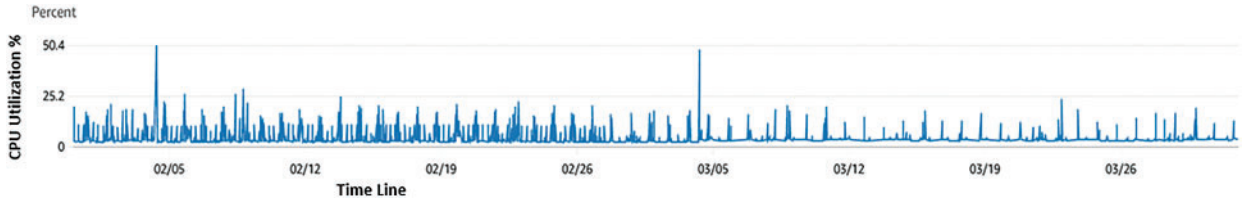

Figure 21: Compute IV maximum CPU usage (a)

In Fig. 22 The vertical axis of the graph shows the CPU utilization percentage. The horizontal axis shows progression over time in last 60 days. Average CPU Utilization remained upto 5%–6% during the application processes in last 60 days.

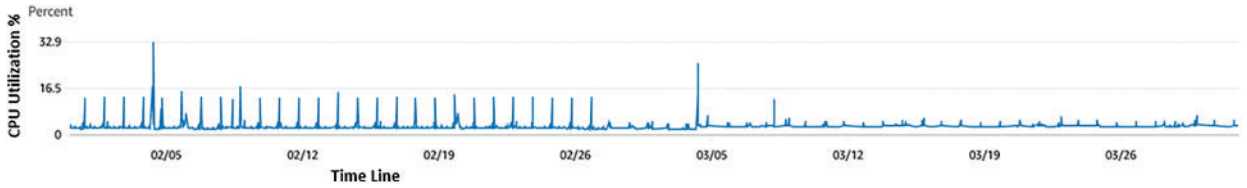

Figure 22: Compute IV average CPU usage (b)

In Fig. 23 The vertical axis of the graph shows the Memory usage percentage. The horizontal axis shows progression over time in last 60 days. Maximum Memory used reahed upto 33.3% during this time interval of 60 days.

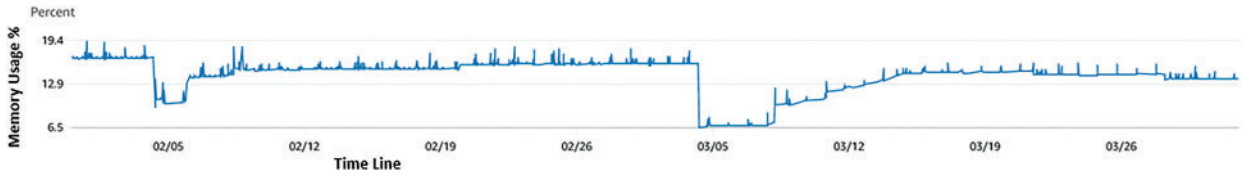

Figure 23: Compute IV maximum memory usage (c)

In Fig. 24 The vertical axis of the graph shows the Memory usage percentage. The horizontal axis shows progression over time in last 60 days. Average Memory usage remained 17%–18%. The complete summary of all the cases as shown in Tab. 1

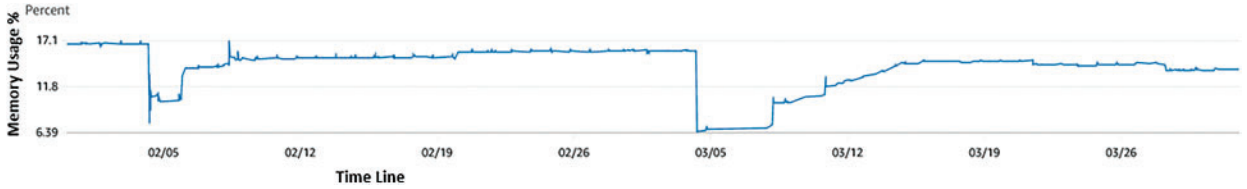

Figure 24: Compute IV average memory usage (d)

Table 1: Comparison of both cases

Iteration	Use case-1					Use case-2			
	CPU Use	Storage utilization	Write throughput	Read throughput	Latency	CPU utilization		Memory utilization	
	Max	Avg	Max	Avg	Read	Max	Avg	Max	Avg
Compute-I	3.0%	55.0%	0–11MB/s	0–15Mb/s	10s	45.6%	6.0%	33.3%	18.0%
Compute-II	2.5%	55.0%	0–11MB/s	0–15MB/s	10s	**46.3%**	**5.0%**	**33.8%**	**10.0%**
Compute-III	3.0%	55.0%	0–11MB/s	0–15MB/s	10s	45.6%	6.0%	33.3%	18.0%
Compute-IV	**7.0%**	**55.0%**	**0–11MB/s**	**0–15MB/s**	**05s**	45.6%	6.0%	33.3%	18.0%

5 Conclusion

In containers, environment provide fine-grained resource sharing, lightweight performance isolation, and quick and flexible deployment. Containers are self-contained, stand-alone containers that bundle software and its dependencies. The scheduler optimizes container placement at the start, to enhance system utilization and minimize cost. The autoscaler allows the current demand for resources to be met while underutilized or idle nodes are shut down. The rescheduler allows changes to the initial placement of containers at runtime to eliminate fragmentation and combine loads for improved resource efficiency. The Kubernetes Vertical Pod Autoscaler modified your pods' CPU and memory reservations to help you "right-size" your applications. Compute results showed that this change can increase cluster resource usage while freeing up CPU and Memory for other pods.

Acknowledgement: Thanks to our teachers and families who provided moral support.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding this study.

References

[1] N. Iqbal, S. Abbas, M. A. Khan, T. Alyas, A. Fatima *et al.,* "An RGB image cipher using chaotic systems, 15-puzzle problem and DNA computing," *IEEE Access*, vol. 7, pp. 174051–174071, 2019.

[2] T. Alyas, I. Javed, A. Namoun, A. Tufail, S. Alshmrany *et al.,* "Live migration of virtual machines using a mamdani fuzzy inference system," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 3019–3033, 2022.

[3] M. I. Sarwar, M. W. Iqbal, T. Alyas, A. Namoun, A. Alrehali *et al.,* "Data vaults for blockchain-empowered accounting information systems," *IEEE Access*, vol. 9, pp. 117306–117324, 2021.

[4] F. Rossi, V. Cardellini, F. Lo Presti and M. Nardelli, "Geo-distributed efficient deployment of containers with kubernetes," *Computer Communication*, vol. 159, no. 3, pp. 161–174, 2020.

[5] M. Tao, K. Ota and M. Dong, "Ontology-based data semantic management and application in IoT-and cloud-enabled smart homes," *Future Generation Computer System*, vol. 76, pp. 528–539, 2017.

[6] K. N. Vhatkar and G. P. Bhole, "Optimal container resource allocation in cloud architecture: A new hybrid model," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 1–13, 2019.

[7] G. M. Diouf, H. Elbiaze and W. Jaafar, "On byzantine fault tolerance in multi-master kubernetes clusters," *Future Generation Computer System*, vol. 109, pp. 407–419, 2020.

[8] L. Heilig, E. Ruiz and S. Vob, "Modeling and solving cloud service purchasing in multi-cloud environments," *Expert System Application*, vol. 147, no. 20, pp. 540–555. 2020.

[9] S. Taherizadeh and M. Grobelnik, "Key influencing factors of the kubernetes auto-scaler for computing-intensive microservice-native cloud-based applications," *Advance Engineering Software*, vol. 140, no. 19, pp. 1–11, 2020.
[10] H. Zeng, B. Wang, W. Deng and W. Zhang, "Measurement and evaluation for docker container networking," in *Int. Conf. on Cyber-Enabled Distributed Computing and Knowledge Discovery*, China, vol. 18, pp. 105–108, 2017.
[11] P. Dziurzanski, S. Zhao, M. Przewozniczek, M. Komarnicki and L. S. Indrusiak, "Scalable distributed evolutionary algorithm orchestration using docker containers," *Journal of Computer Science*, vol. 40, pp. 1–10, 2020.
[12] A. Dua, S. Randive, A. Agarwal and N. Kumar, "Efficient load balancing to serve heterogeneous requests in clustered systems using kubernetes," in *IEEE 17th Annual Consumer Communication Network Conf. CCNC 2020*, USA, no. 5, pp. 1–2, 2020.
[13] M. Wang, D. Zhang and B. Wu, "A cluster autoscaler based on multiple node types in kubernetes," in *Proc. of IEEE 4th Information Technology,Networking Electronic and Automation Control Conf. ITNEC 2020*, China, no. 5, pp. 575–579, 2020.
[14] M. Benedictis and A. Lioy, "Integrity verification of docker containers for a lightweight cloud environment," *Future Generation Computer System*, vol. 97, pp. 236–246, 2019.
[15] Y. Cui, S. Kara and K. C. Chan, "Manufacturing big data ecosystem: A systematic literature review," *Robotics and Computer-Integrated Manufacturing*, vol. 62, no. 19, pp. 101861–101870, 2020.
[16] C. T. Joseph and K. Chandrasekaran, "IntMA: Dynamic interaction-aware resource allocation for containerized microservices in cloud environments," *Journal of Systems Architecture*, vol. 111, no. 4, pp. 1–15, 2020.
[17] N. Tabassum, T. Alyas, M. Hamid, M. Saleem, S. Malik *et al.,* "Semantic analysis of urdu English tweets empowered by machine learning," *Intelligent Automation & Soft Computing*, vol. 30, no. 1, pp. 175–186. 2021.
[18] M. Malawski, A. Gajek, A. Zima, B. Balis and K. Figiela, "Serverless execution of scientific workflows: Experiments with HyperFlow, AWS lambda and google cloud functions," *Future Generation Computer System*, vol. 110, pp. 502–514, 2020.
[19] A. M. Fernández, D. Gutiérrez, A. Troncoso and F. Martínez, "Automated deployment of a spark cluster with machine learning algorithm integration," *Big Data Research*, vol. 19–20, pp. 1–15, 2020.
[20] S. Yin, D. Chen and J. Le, "Deep neural network based on translation model for diabetes knowledge graph," in *Fifth Int. Conf. on Advanced Cloud and Big Data (CBD)*, Shanghai, China, pp. 318–323, 2017.
[21] A. Botta, W. Donato, V. Persico and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future Generation Computer System*, vol. 56, pp. 684–700, 2016.
[22] F. Alhaidari, S. H. Almotiri, M. A. Ghamdi, M. A. Khan, A. Rehman *et al.,* "Intelligent software-defined network for cognitive routing optimization using deep extreme learning machine approach," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 1269–1285, 2021.
[23] A. H. Khan, M. A. Khan, S. Abbas, S. Y. Siddiqui, M. A. Saeed *et al.,* "Simulation, modeling, and optimization of intelligent kidney disease predication empowered with computational intelligence approaches," *Computers, Materials & Continua*, vol. 67, no. 2, pp. 1399–1412, 2021.
[24] N. Tabassum, A. Ditta, T. Alyas, S. Abbas, H. Alquhayz *et al.,* "Prediction of cloud ranking in a hyperconverged cloud ecosystem using machine learning," *Computers, Materials & Continua*, vol. 67, no. 3, pp. 3129–3141, 2021.
[25] Q. Khan, S. Abbas, M. A. Khan, A. Fatima, S. Alanazi *et al.,* "Modelling intelligent driving behaviour using machine learning," *Computers, Materials & Continua*, vol. 68, no. 3, pp. 3061–3077, 2021.