Review Article

International Journal of Advanced Technology and Engineering Exploration, Vol 11(116)
ISSN (Print): 2394-5443 ISSN (Online): 2394-7454
http://dx.doi.org/10.19101/IJATEE.2023.10102431

A systematic review on recent methods of scheduling and load balancing for containers in distributed environments

Neelima Gogineni[1,2*] and Saravanan Madderi Sivalingam[3]
Research Scholar, Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences Chennai – 602105, Tamil Nadu, India[1]
Assistant Professor, Gokaraju Rangaraju Institute of Engineering & Technology, Hyderabad. Telangana, India[2]
Professor, Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai – 602105, Tamil Nadu, India[3]

Received: 16-October-2023; Revised: 03-July-2024; Accepted: 07-July-2024

Abstract
The concept of containers is widely used in cloud computing environments due to perceived benefits linked to packaging and deploying software efficiently. Due to the unprecedented usage of containers, it became indispensable to manage them efficiently. In other words, container scheduling and load-balancing strategies have assumed significance in distributed environments. In this paper, a systematic review is presented that covers ninety-five peer-reviewed research articles from 2012 to 2023, providing a comprehensive understanding of the current state of research in this field. This review synthesizes new insights and knowledge, shedding light on the challenges in container load balancing, various existing methods of container placement, container orchestration, scheduling, load balancing, cluster management, dynamic resource allocation, and micro-services. This paper also investigates load balancing for containers in fog or edge computing environments connected to cloud computing. The study thus synthesizes new knowledge on diversified aspects of containers and their load balancing in cloud environments, paving the way for further investigation and the realization of more optimal container scheduling and load-balancing approaches.

Keywords
Containerization, Scheduling, Load balancing, Cloud computing, Container placement, Container orchestration.

1.Introduction
Containerization has emerged as a crucial element of software deployment in distributed environments, owing to its user-friendly nature. The rise of cloud computing, fog computing, and edge computing has made container management a necessity. However, scheduling containerized applications in distributed environments presents a host of challenges. One such challenge is optimizing the scheduling process to achieve container load balance. Docker containers, a popular choice in cloud computing, often face issues with the default spread strategy. This paper explores various container scheduling approaches and underscores the need to enhance dynamic resource allocation in the presence of heterogeneous computing resources.

The practical implications of this research are significant, as it can lead to more efficient and effective container scheduling and load balancing in real-world cloud computing environments, thereby enhancing their performance and scalability.

Many researchers contributed to scheduling and load balancing for containers. Singh et al. [1] stated that containerization with cloud computing addresses data application needs. Docker Swarm with microservice architecture offers efficient load balancing and service discovery for scalability and fault tolerance. This approach shows potential for enhancing data application management, with potential for future improvements in computational complexity and multi-cloud scenarios. Wan et al. [2] optimized application deployment in cloud data centers using microservices and Docker containers. The aim is to minimize deployment and operation costs while meeting service delay requirements. Comparison with existing

*Author for correspondence

strategies in Docker Swarm using real data traces demonstrates cost savings and flexibility. However, their methodology is so complex and needs further improvement. The research on load balancing of containers is relatively new, motivating the present study. There are many challenges, such as a lack of depth in the literature, required datasets for simulation study, and a lack of support for simulators. To review the literature findings, this paper followed a systematic review approach. To conduct a systematic review, the article selection process was based on the preferred reporting items for systematic reviews and meta-analyses (PRISMA) model, focusing on load balancing and scheduling for containers.

Our contributions to this paper are as follows.

1. The article selection process was systematically reviewed based on the PRISMA model on load balancing and container scheduling.
2. The findings highlight various aspects of containers and their efficiency in distributed environments.
3. The study also focused on essential gaps in the latest research articles leading to specific findings.

The remainder of the paper is structured as follows: Section 2 presents our methodology for systematically reviewing container load balancing. Section 3 reviews the literature and answers questions posed in our methods. Section 4 presents research gaps found in the survey, while section 5 concludes our work and gives directions for future work.

2.Research methodology

Our research methodology is based on a systematic literature review that illuminates the diversified aspects of containers. This includes container technology, the distributed environments it operates in, scheduling and load balance for containers, and more. It also investigates the difference between containers and virtual machines (VMs), focusing on the utility of edge and fog computing environments in the context of increased usage of microservices and containers.

2.1Research questions (RQ)

Synthesis of literature in this paper from 2012 to 2023 is expected to derive new helpful knowledge. In other words, this research probes into the literature to answer specific research questions. These questions unearth valuable insights latent in the literature articles. Our research includes more recent works in the given research area. The research questions are as follows:

RQ1: What are the challenges involved in scheduling for containers?

RQ2: What are the existing methods for container load balancing and scheduling?

RQ3: What is the main difference between scheduling and load balancing for VM vs. containers?

RQ4: What are the existing approaches for dynamic resource allocation in containerized environments?

RQ5: What is the necessity of fog or edge computing, and how is container load balancing done in fog computing?

RQ6: What is the significance of container orchestration and Docker cluster management and their utility in managing containers?

RQ7: What is the importance of security for micro-services and Docker containers in distributed environments?

These research questions probe into the literature to derive helpful knowledge. Each question is answered with a review of relevant literature. Section 3.1 throws light on RQ1 and reviews the challenges involved in container scheduling. RQ2 is responded to in Section 3.2 as it focuses on existing methods of container load balancing and scheduling. Section 3.3 throws light on RQ3 to ascertain the main difference between scheduling and load balancing for VMs vs. containers. RQ4 is answered in Section 3.4, which covers existing approaches for dynamic resource allocation in containerized environments. Section 3.5 answered RQ5, reflecting the necessity of fog or edge computing and how container load balancing is done in fog computing. RQ6 is responded to in section 3.6, which investigates the significance of container orchestration and Docker cluster management and their utility in managing containers. Section 3.7 answers RQ7, showing the importance of security for micro-services and Docker containers in distributed environments.

2.2Research process

Our research process and article selection procedure are based on the PRISMA model, which is widely used for selecting articles in systematic reviews. 150 articles were collected from different reputed sources and the Scopus database. Then, articles were subjected to a filtering process that applies inclusion and exclusion criteria, excludes duplicates, and removes irrelevant articles. Finally, 99 research articles were selected for the review and analysis. As presented in *Figure 1*, the article selection process involves a systematic approach to improving the quality of the literature review research outcomes.

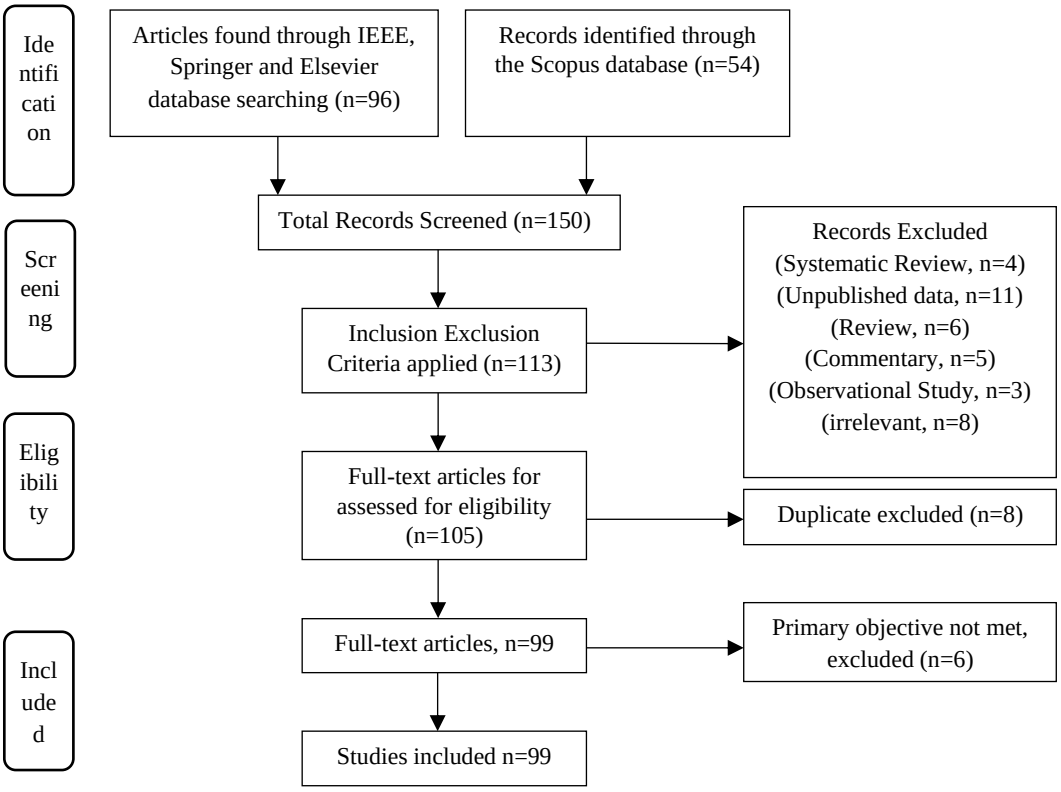

Figure 1 PRISMA model showing the article selection process

2.3Data sources

Research articles were collected from reputed and peer-reviewed data sources. The article section is governed by the PRISMA approach presented in *Figure 1*. A systematic approach is used as part of the article selection process. The digital libraries used to obtain research articles are Google Scholar, Springer, Elsevier, IEEE Xplore, Hindawi, Wiley, ACM, and MDPI.

2.4Search process

Article search has different phrases, as provided in *Table 1*. The search process exploits digital libraries and their associated search applications to gather the most relevant articles that have been peer-reviewed. The search process uses different phrases to provide articles on various aspects of containers and load balancing. However, a systematic review cannot include all the articles available through the search process. Towards this end, the criteria required to improve the quality of the article selection process were used.

2.5Criteria to improve the articles selection process

Criteria for article inclusion and exclusion are defined and presented in Table 2. These criteria ensured that the researcher selected high-quality articles suitable for the systematic review. Finally, the filtered articles have coverage from 2012 to 2023. The criteria used for the article selection process enable decision-making in filtering all the acquired research articles.

Table 1 Shows different phrases for searching

S.No.	Search Phrase	Description
1	Load balancing for containers	This phrase is used to find load-balancing methods for containers.
2	Scheduling for containers	This phrase is used to find scheduling methods for containers.
3	Dynamic resource allocation for containers	This phrase is used to find articles on dynamic resource allocation for containers.
4	Container orchestration	This phrase is used to find articles on container orchestration dynamics
5	Load balancing for VMs vs. containers	This phrase is used to find load-balancing methods for containers and VMs.
6	Security for Microservices and containers	This phrase is used to find articles linked to micro-services and containers.

Table 2 Criteria used for the article selection process

S.No.	Inclusion Criteria	Exclusion Criteria	Justification
1	English	Other languages	English is the globally understood language
2	Containerization in a software context	Containers concept in transportation, etc.	Containerization linked to software deployment is relevant.
3	Peer-reviewed	Non-peer reviewed	Peer-reviewed articles reflect high-quality research
4	Container orchestration in a software context	Container orchestration in a non-software context	Container orchestration linked to software deployment and management is relevant
5	Container scheduling and load balancing in a software context	Container scheduling or load balancing in a non-software context	Container scheduling and load balancing linked to software deployment and management are relevant

2.6Research bibliographic findings

This section presents the bibliographic findings of the research articles obtained through selection. It includes details of the articles by year and publisher. *Table 3* shows how the initially collected articles in stage 1 are reduced to 99 in stage 2.

Table 3 Shows statistics about the article collection across years

Stage	Load balancing for containers research articles			
Stage1	150			
Stage2	Article categorization by year			
	Until 2011	2012-2015	2016-2019	2020-2023
	0	6	50	43

As presented in *Figure 2*, there is no representation for 2013. However, the article covers the period from 2012 to 2013. The highest number of articles used in 2020 is 21.

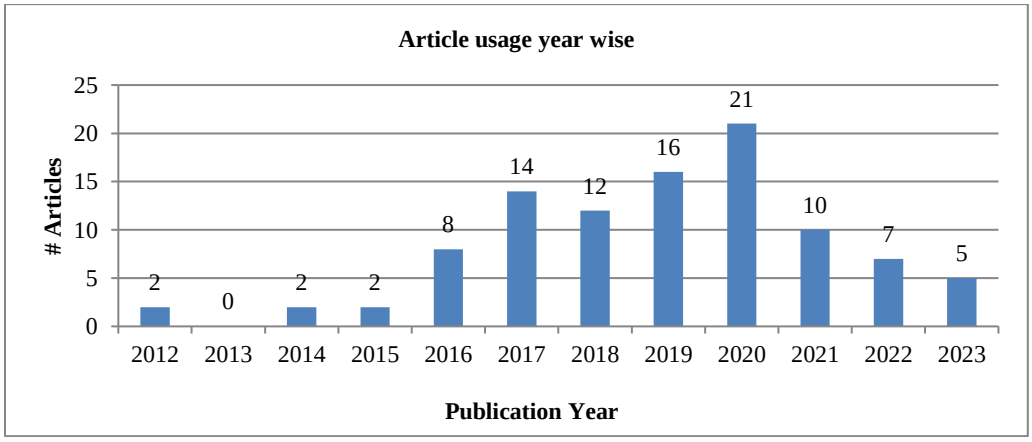

Figure 2 Shows articles used in the current research publication year-wise

As presented in *Table 4*, journal articles and conference papers distributed across publishers are provided.

Table 4 Publisher-wise article distribution dynamics

Publisher	# Articles	
	Journal	Conference
IEEE	29	13
Elsevier	12	-
Springer	18	-
MDPI	4	-
Others	16	-
ACM	4	-
Hindawi	3	-

As presented in *Figure 3*, journal articles and conference papers distributed across publishers are provided. Thirteen IEEE conference papers were used in the research, while only journal articles from all publishers were used.

Bubble graph representation

As shown in *Figure 4*, the article distribution is provided year-wise. As shown in *Figure 5*, the article distribution is provided publication-wise.

Table 5 presents the publisher-wise article references. This will help you quickly choose articles publisher-wise.

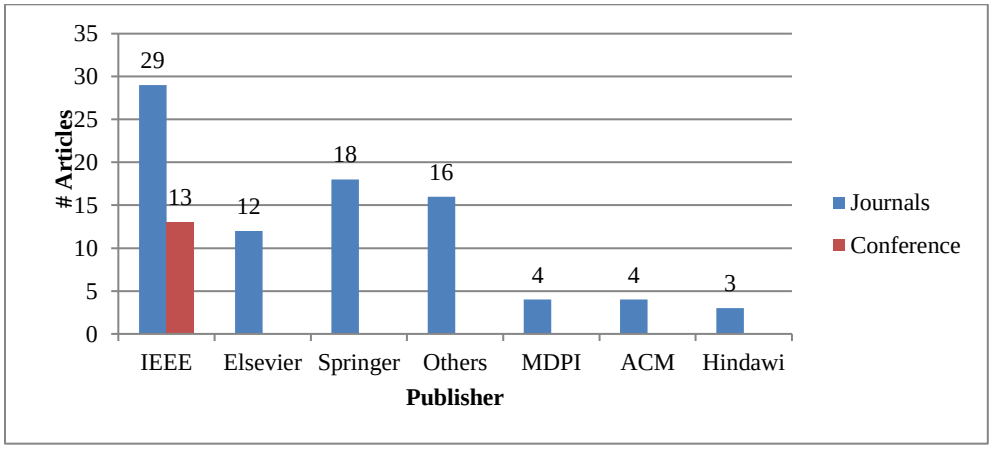

Figure 3 Publisher-wise journal or conference distribution

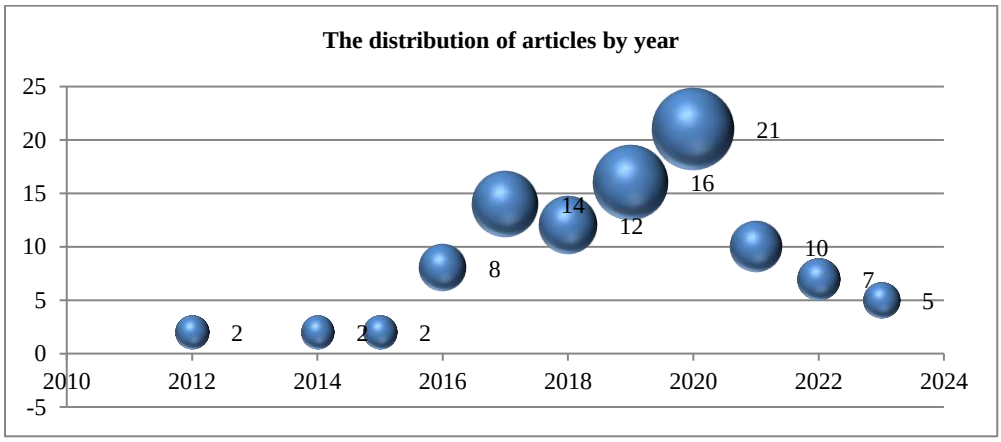

Figure 4 The distribution of articles by year

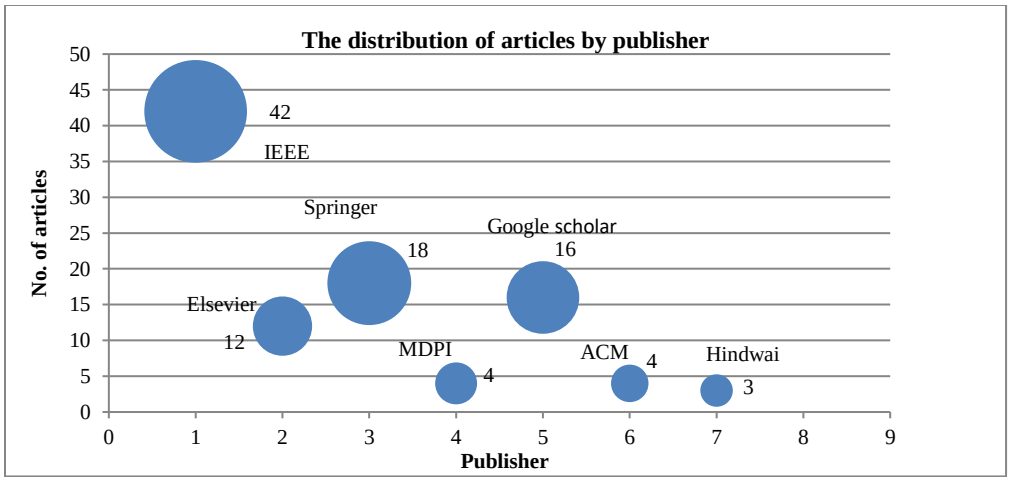

Figure 5 Publication wise articles

Table 5 Publisher-wise article references

Publisher	Number of the papers
IEEE	29
Elsevier	12
Springer	18
MDPI	4
Google Scholar	16
Hindawi	3
IEEE Conference	13
ACM	4

3.Literature review

This section reviews the literature on various aspects of containers and their load balancing in distributed environments. It throws light on the different research questions provided in section 2.

3.1Challenges in scheduling for containers

Scheduling in the context of containerized applications in distributed environments has many challenges. Martin et al. [1] stated that containerization with cloud computing addresses data application needs. Docker Swarm with microservice architecture offers efficient load balancing and service discovery for scalability and fault tolerance. This approach shows potential for enhancing data application management, with potential for future improvements in computational complexity and multi-cloud scenarios. Wan et al. [2] optimized application deployment in cloud data centers using microservices and Docker containers. The aim is to minimize deployment and operation costs while meeting service delay requirements. Comparison with existing strategies in Docker Swarm using real data traces demonstrates cost savings and flexibility. They recommended reducing computational complexity and considering additional optimization goals, such as minimizing response time and improving energy efficiency. Fazio et al. [3] opined that micro-services-based cloud applications enable independent deployment but come with remote call costs and synchronization overhead. This is one of the significant challenges. Containerization has its benefits and also challenges. Acharya and Suthar [4] suggested that Docker containers are commonly used in cloud computing; however, the default spread strategy poses challenges for their placement. They also mentioned several other container scheduling approaches. Future work will aim to explore memory utilization, resource reservation, and heterogeneous machines, and compare these with other scheduling strategies. Watada et al. [5] explored and stated that containerization offers lightweight, efficient computing but faces networking, security, and tooling challenges. However, VMs excel in isolation and portability. Rabiu et al. [6] investigated and found that microservices break monolithic software into independent containers. Load balancing and auto-scaling in cloud-based microservices address challenges for a better quality of service. Piraghaj et al. [7] discussed cloud and edge computing supporting deep learning and the internet of things (IoT). However, their findings reveal security and overhead challenges in such environments. Application programming interfaces and microservices handle tasks, with Kubernetes and Istio facilitating flexible deep-learning applications. No et al. [8] discussed optimizing data processing using container technology, edge computing, and machine learning (ML). However, their study is incomplete without mentioning challenges in container scheduling, such as template design, governance, standards, and security policies. *Table 6* shows a summary of the findings and *Table 7* show a Significant Performances.

In cloud computing environments, container scheduling presents several challenges. These challenges include the dynamic nature of workloads, diverse resource requirements, varying resource availability, and the need to fulfil consumer requests. Containers running microservices must handle many requests, tolerate network failures, and ensure scalability, availability, and high performance.

3.2Container load balancing and scheduling

Scheduling and load balancing for containers are two closely related activities. Sureshkumar and Rajesh [9] introduced an energy-efficient Docker container and job scaling model, focusing on load balancing and energy optimization. Azab [10] investigated Docker containers and their utility in multi-task and high-performance computing (HPC). Li et al. [11] proposed a particle swarm optimization-based Docker container scheduling algorithm within Swarm, improving resource utilization and performance. Kaewkasi and Chuenmuneewong [12] explore ant colony optimization (ACO) to enhance Docker container scheduling in Swarm kit, improving application performance by approximately 15%. Qian et al. [13] investigated the integration of OpenStack and Docker for cloud-edge computing demands a load-balancing Docker scheduling approach to address resource imbalance. It offers better resource utilization and scalability for cloud systems.

Li and Fang [14] studied Docker and its management tool Swarm and found that they face load-balancing challenges. A new multi-algorithm collaboration

scheduling strategy (MCSS) outperforms existing Swarm strategies, enhancing cluster load balancing. Future work includes refining message delivery and exploring multi-container scheduling with Docker Compose. Ahmad et al. [15] studied a container technology vital for cloud services that lack a comprehensive scheduling technique survey. The paper classifies, analyses, and outlines future research directions for container scheduling techniques. Guo and Yao [16] said that docker microservices pose challenges for VM scheduling. Container scheduling strategy based on neighborhood division (CSBND) strategy optimizes performance with load balancing. Aruna and Pradeep [17] proposed the ACO-based optimization algorithm for container load balancing. Liu et al. [18] focused on Docker container resource scheduling as a research hotspot with NP-complete optimization criteria. The multiport algorithm improves performance, considering central processing unit (CPU), memory usage, network image transmission time, container-node association, and clustering. Swarm's Spread, Binpack, and random algorithms have shortcomings like storage space debris and centralized allocation. In upcoming container fault tolerance and explore recent research results' application.

According to Patra et al. [19], cloud computing enables on-demand resource provision, necessitating efficient load balancing among multiple servers. Experimental results show equitable workload distribution. It also differentiates between VM-based and container-based clouds, highlighting the latter's advantages. The proposed system model and resource allocation algorithm utilize a minimum spanning tree. Further, it contributes to load-balancing research in containerized clouds, with future enhancements planned. Cérin et al. [20] focused on the scheduling strategy for Docker Swarm, focusing on service level agreements (SLA) classes. It aims to optimize container scheduling for organizations with private infrastructure. Unlike existing methods, it dynamically allocates CPU cores based on user SLA and machine load. Emulation testing shows promise for further development. It also mentions other container scheduler strategies and proposes dynamic machine consolidation for energy efficiency in the future. Zhang et al. [21] discussed an adaptive scheduler that considers energy, image, and network costs, outperforming Docker Swarm. The approach, employing integer linear programming, offers cost savings and easy integration into container orchestration frameworks—a novel algorithm that addresses various costs, contributing to containerized application scheduling in the cloud. In the Future, they will extend workload representation and reconstruct the scheduler in GO or Python for open-source frameworks.

Marathe et al. [22] stated that distributed computing adds a new layer to virtualized assets. The goal of load balancing among nodes using Docker, containers, Docker Swarm, and Kubernetes technologies. The paper evaluates host provisioning in Open Stack and the impact of cloud computing. Kubernetes outperforms Docker Swarm, and even though Kubernetes installation is complex, it is worth the effort. Bernstein [23] explores the growing significance of container technology in cloud computing infrastructure. Most cloud systems use hypervisors. Zhang et al. [24] introduce the "Container-VM-PM" architecture to optimize resource allocation, focusing on placement strategies for all three entities. The proposed approach improves resource utilization by considering physical servers outperforming the existing plan. In the Future, they will explore container consolidation and load balancing under this architecture, testing additional algorithms.

Zhiyong and Xiaolan [25] studied container technology and found that it replaces traditional VMS in cloud computing. This paper discusses container cloud task scheduling algorithms, platforms, and improvements, emphasizing efficiency and future research areas. Adhikari and Srirama [26] present an energy-efficient container-based scheduling strategy for the IoT and non-IoT tasks. The proposed method outperforms existing strategies across various metrics. Future research will explore trade-offs in container-based scheduling and develop a dynamic container-based cloud environment for IoT applications. Menouer [27] discussed a multi-criteria Kubernetes container scheduling strategy. Kingston caregiver stress scale (KCSS) aims to optimize container scheduling based on six critical criteria, improving performance under various scenarios compared to single-criteria strategies. Future work includes fault tolerance, consolidation, and application in intelligent buildings. Yang et al. [28] present Yun, a high-performance container management service based on OpenStack, integrating Docker. Yun enhances resource utilization, load balancing, and overall performance. Future work aims to expand container management functions. Smimite and Afdel [29] stated that an efficient data center involves virtualization and workload consolidation. A hybrid ACO and first-fit decreasing algorithm minimizes power consumption.

In distributed environments, various methods are used to solve the problems of container scheduling and load balancing. These methods encompass heuristic approaches, including traditional methods, bioinspired optimizations, and learning-based approaches. Learning-based approaches leverage artificial intelligence (AI) and related technologies to comprehend runtime situations and make well-informed decisions during the scheduling process. Additionally, load balancing facilitates resource optimization, leading to improved infrastructure efficiency.

3.3Scheduling and load balancing for VMs vs. containers

Scheduling and load balancing can encompass larger contexts in distributed computing environments. Mattia and Beraldi [30] proposed a peer-to-peer function-as-a framework based on Docker containers that facilitates decentralized scheduling and load balancing algorithms in Edge and Fog Computing, enhancing accurate environment testing. Li and Xia [31] proposed a hybrid cloud-based auto-scaling platform using Docker to adjust web application resources based on demand efficiently. Patra et al. [32] observed that container-based virtualization enhances cloud computing efficiency. Bio-inspired models like grey wolf optimization with simulated annealing improve load balancing. Yang et al. [33] focused on cloud computing resources and scheduling optimization at virtual and physical resource layers, proposing future research directions. Wang et al. [34] discussed high-performance Docker integration in OpenStack via the container management service "Yun." Yun optimizes deployment, throughput, and performance, introduces resource-aware scheduling, and ensures OpenStack compatibility. Future expansion is planned. Kim et al. [35] used real-time traffic reporting and the minor least traffic load balancing (LTLB) algorithm for load balancing. Docker's flexibility is explored, and the minor traffic (LT) algorithm shows potential for network balancing. A proactive auto-scaling scheme is proposed to reduce user wait times and cloud resource expenses, potentially outperforming reactive scaling.

Talukder et al. [36] observed that SDN faces challenges in energy management and load balancing—dual threshold load balancing (DTLB) method for SDN, optimizing process and resource allocation. DTLB improves energy efficiency, throughput, and response time compared to traditional models. Elmagzoub et al. [37] stated that cloud computing offers flexible resource access and load balancing, vital for efficient, scalable, and high-quality cloud services. Song et al. [38] focused on cloud computing enhancements with large-scale simulations. Then, we intend to develop a VM-based federate migration scheme for efficient high-level architecture (HLA) systems in cloud simulation platforms. Afzal and Kavitha [39] discussed cloud computing load balancing issues, reviewed techniques, and suggested improving quality of service (QoS) metrics and algorithm complexity. Naik et al. [40] discussed the superiority of container-based virtualization over traditional VMs in cloud computing, focusing on load balancing. The honeybee mating algorithm (HBMA) outperforms other methods in load variance and makespan. Polikarpova et al. [41] revealed challenges in verifying complex container libraries like EiffelBase2 and presented insights. It highlights the need for predictability, flexible tools, and how verification promotes good design. Lee et al. [42] proposed a load-balancing scheme for multi-agent-based distributed simulations, optimizing resource distribution by monitoring agent workload. It significantly reduces simulation time. Further, they can include reducing overhead and testing in diverse domains.

Scheduling and load balancing are concepts related to improving cloud infrastructure performance and enhancing customer satisfaction. There are differences between scheduling and load balancing for VMs and containers. Fundamentally, a VMs is different from a container. A VMs can handle several tasks, while a container is a self-contained software package that needs to be deployed.

3.4Dynamic resource allocation in containerized environments

Dynamic resource allocation is crucial in VMs and containers in distributed computing environments. Tao et al. [43] proposed a fuzzy logic-based scheduling algorithm for global container resource allocation, enhancing cluster performance. Zhiyong and Xiaolan [44] observed that Docker drives container cloud growth; resource scheduling affects cluster performance. Kubernetes needs enhanced load balancing. Li et al. [45] proposed an optimized artificial bee colony algorithm for containerized microservice scheduling, considering component relevance and improving cluster load balancing and service response time. Mao et al. [46] showed that virtualization in cloud computing enhanced services, but VMs faced boot volume and provisioning issues. Docker containers solved these, and we propose Dynamic and resource-aware placement scheme

(DRAPS), a dynamic resource-aware placement scheme for Docker container clusters, enhancing performance in heterogeneous clusters by optimizing resource usage. Tihfon et al. [47] explored the application deployment challenges in cloud computing, advocating Docker for rapid deployment and optimization. Docker's advantages include resource efficiency and application portability, making it an ideal choice. The work in the future is to implement and scale the platform using Amazon EC2 for performance management and thorough evaluation.

Long et al. [48] advocate field programmable gate array (FPGA) resource virtualization using Docker and Kubernetes, enhancing utilization, scalability, and accessibility for local and remote applications. The approach improves FPGA resource allocation and isolation, ensuring efficient use. Imdoukh et al. [49] agree that containers are revolutionizing cloud apps with resource efficiency. A ML-based auto-scaler ensures timely scaling to handle dynamic workloads. Mondesire et al. [50] focused on adapting traditional HPC for cloud-based interactive simulations using load-balancing techniques via virtualization and containers. El et al. [51] observed that cloud computing's importance continues to grow, but dynamic scalability models for containerized services are lacking. A queuing theory-based model is proposed to optimize QoS and resource utilization while minimizing costs. Simulation results support its effectiveness. As stated by Celesti et al. [52], the IoT cloud connects smart devices to the cloud via the internet, offering a service of loT with container virtualization for better QoS. Overhead from concurrent containers is acceptable for enhanced IoT as a Service (IoTaaS) provisioning. In the coming years, CPU, memory, and network usage analysis, distributed IoTaaS orchestration, and security with Blockchain technology.

Xiaojing and Govardhan [53] focused on efficient VM sizes for hosting containers to minimize resource wastage and improve energy consumption. Thongthavorn and Rattanatamrong [54] focused on parallel multi-container migration in a wide area network (WAN), utilizing feedback control to optimize migration times. Vaishali et al. [55] investigate the efficient container selection for big data computing, focusing on security, integrity, and data privacy. The proposed guided container selection (GCS) Employs deep neural learning (DNL) to optimize container selection based on dynamic data involvement, promoting efficient data processing. Further work involves real-world application and prioritizing cost-effective containers for improved performance. Hofer and Sehr [56] evaluated dual-frequency precipitation radar (DPR). Precipitation algorithms using synthetic data based on tropical rainfall measuring mission (TRMM) and precipitation radar (PR) data. It was found that Ku-band precipitation radar (KuPR) and DPR estimates matched actual values for total precipitation amounts, while Ka-band precipitation radar (KaPR) underestimated precipitation rates, mainly during intense precipitation events in the Tropics. The underestimation was attributed to an attenuation correction problem. Future tasks include algorithm improvements and over-land evaluations. Ma et al. [57] addressed the HPC load imbalance via container-based virtualization, dynamically adjusting CPU allocation to message-passing interface (MPI) processes for improved performance. Kanchanadevi et al. [58] studied and found that container as a service (CaaS) replaces VMs, focusing on efficient CPU and memory utilization in cloud environments. ACO performs well in minimizing active physical machines and VMs. Souza et al. [59] investigated edge computing, which reduces communication costs but faces hardware limitations in IoT devices, impacting intelligent city apps. Osmotic computing balances workloads, overcoming these limitations. An Osmotic execution framework deploys smart city apps across edge and cloud, focusing on load balancing. Evaluation with a real intelligent parking app demonstrates efficient resource utilization and low latency.

Various approaches exist for dynamic resource allocation in containerized environments. It's crucial to allocate resources appropriately and dynamically in distributed environments with deployed containers. Methods for resource allocation include heuristic approaches and learning-based approaches. The former relies on specific heuristics or procedures, while the latter is based on learning and acquiring knowledge during runtime.

3.5Container load balancing in a fog computing environment

Fog computing is a computing infrastructure between cloud and data sources, making the computation faster. Baburao et al. [60] discussed fog computing, which bridges IoT and cloud, alleviating network congestion. Particle swarm optimization-based embedded dynamic random-access memory (EDRAM) enhances load balancing for improved quality of experience (QoE). Pérez et al. [61] specified two critical

challenges in Docker container scheduling for cloud-fog-IoT networks. Li et al. [62] First, integrating intelligent container schedulers into platforms like Docker Swarm, Google Kubernetes, and Apache Mesos. Second, creating intelligent container schedulers for various network interfaces: cloud-to-fog, fog-to-IoT, and cloud-to-fog. The goal is to optimize microservices allocation, improving QoS parameters such as latency, load balance, energy consumption, and runtime. Choudhury et al. [63] stated that mobile edge cloud (MEC) faces resource assignment and load balancing challenges. ShareOn, a dynamic container migration framework, improves system response time by 15-22%. Baburao et al. [64] stated that handling missing data in coevolving time series is challenging. It develops dynamic matrix factorization techniques, addressing efficiency and effectiveness, with experiments demonstrating their performance. Singh et al. [65] observed that rapid IoT growth strains cloud data centers. Edge computing alleviates latency issues, with energy-efficient container-as-a-service proposed for QoS improvement.

Ma et al. [66] studied the significance of edge computing which addresses IoT's delay-sensitive needs. Container migration enhances load balancing for efficient resource utilization in edge networks. Li et al. [67] observed that edge computing addresses IoT's geographic load imbalance. Container tech aids dynamic load balancing via container migration with an improved ant colony algorithm. Singh et al. [68] envisaged the growing IoT in fog computing introduces latency and resource issues. A secure, energy-efficient load-balancing solution was proposed, significantly improving efficiency. Alqahtani et al. [69] observed that fog computing brings IoT services to the cloud-fog edge, reducing data center load and latency. Load balancing ensures reliability and performance. Dlamini and Vilakati [70] investigated small cell base stations (SBSs) with computing capabilities to aid 5G networks. Multi-access edge computing (MEC) enables low latency and bandwidth. The green-based edge network management (GENM) algorithm ensures energy savings, quality of service, and load balancing. Dynamic management, auto-scaling, and tuning drivers are employed. GENM outperforms benchmarks. Chen and Liu [71] The IoT architecture handles data generation but faces bandwidth and connectivity constraints. A 3.5-tier edge computing AIoT (ECAIoT) architecture improves response times, scalability, and cost-effectiveness, facilitating IoT applications. Experimental results confirm better performance and scalability.

Unlike cloud infrastructure, fog, and edge computing resources are located close to users. This means that they can enhance task execution performance with lower latency. By utilizing fog and edge computing in addition to cloud infrastructure, we can improve task performance in distributed environments.

3.6Orchestration and cluster management

3.6.1Container orchestration

Container orchestration is the automatic process of ensuring the life cycle of containers. It enables portability, scalability, and efficiency. Khan [72] proposed a twelve-factor app that enhances agility; containers aid efficiency in evolving cloud apps, enabling microservices for scalability and reliability. Hussein et al. [73] focused on container technology, which encapsulates tasks and libraries at the OS level, and (CaaS) is rising. This architecture employs scheduling heuristics and meta-heuristic algorithms, showing superior performance in cloud environments. Future research will explore workflow containerization processes and consider additional computing resources. Zhong and Buyya [74] observed that containers are widely used in cluster management systems. A cost-efficient strategy combines task packing, auto-scaling, and rescheduling for heterogeneous workloads to optimize dynamic clusters. Experiments demonstrate cost savings. Littley et al. [75] discussed the build operate lease-transfer (BOLT) design and proposed a hyper-converged registry system where all servers share caching, storage, and processing tasks, significantly improving performance, scalability, and fault recovery. Louati et al. [76] worked on the LXCloud-CR, a decentralized Checkpoint-Restart model for IaaS container-based virtualization, enhancing reliability and reducing overheads.

Louati et al. [77] focused on LXCloudFT, a fault-tolerant Cloud system, adding LXCloud-Rep for decentralized, versioned, and GC-enhanced container replication, enhancing availability for web apps. Martin et al. [78] cloud technology is vital in modern information technology optimization, addressing budget constraints. Automation, orchestration, and intelligent measures optimize IT processes and reduce operational costs. The cloud journey undergoes centralization, consumerization, and federation and embraces multi-cloud strategies. Cloud orchestration platforms mitigate growing complexities. Wan et al. [79] exploited Docker and Spark in a distributed

computing paradigm to realize a prototype application. Kaur et al. [80] introduced an inkjet-printed information technology outsourcing (ITO) layer between silver and Zink to improve semiconductor/contact integration, significantly reducing contact resistance. Muthakshi and Mahesh [81] stated that container virtualization emerges with multitenancy and virtualization advancements. A load balancing with a power optimization algorithm enhances performance. Container load balancing has specific practical issues, as envisaged by Queiroz et al. [82]. Rizvi et al. [83] deal with deploying diverse service-based apps on Cloud platforms, introducing mobile service containers for independent service mobility. Two mobility techniques, encapsulation into agents and adding mobility components, are compared, with the latter showing greater efficiency. In the future, this work will also include developing a decision module for mobile service containers. Gamal et al. [84] observed that container technology revolutionized cloud computing with its benefits. Challenges include security, performance optimization, and adapting to varied workloads.

3.6.2Docker cluster management

Docker cluster is the means of managing Dockers efficiently. Peinl et al. [85] found that Docker simplifies running composite applications in the cloud but lacks multi-host management. Various tools emerged to address this, mapping to case study requirements. The solution targets a broad user spectrum between individual developers and large cloud providers. Actual cloud environments with elasticity require automation and fulfilment of specified requirements. Some requirements remain unfulfilled, and closer cooperation within the Docker ecosystem is needed, possibly in an embracing ecosystem project involving Mesos proponents. Tihfon et al. [86] proposed a Docker and Amazon web services (AWS) based PaaS cloud infrastructure for efficiently deploying distributed applications. Pahl and Lee [87] investigated container technology's relevance for edge clouds and lightweight virtualization solutions in multi-cloud architectures. Zheng et al. [88] proposed a self-defined Kubernetes scheduler to improve load balancing and cluster efficiency compared to native Kubernetes. Hu et al. [89] focused on OS containers gaining popularity for cloud computing, but existing scheduling focuses on single objectives. A new algorithm balances load, dependencies, and multi-resource guarantees on heterogeneous clusters. The algorithm, implemented as MODscheduler, outperforms existing orchestration platform schedulers in simulations. Future work can include integrating the algorithm into Google Kubernetes and accommodating time-critical applications.

Mao et al. [46] studied virtualization in cloud computing to improve services, but VMs faced boot volume and provisioning issues. Docker containers solved these, and we propose DRAPS, a dynamic resource-aware placement scheme for Docker container clusters, enhancing performance in heterogeneous clusters by optimizing resource usage. Nguyen and Kim [90] investigated Kubernetes, a popular container orchestration platform that faces challenges in load balancing, leader distribution, and consistency maintenance. Experimental results emphasize leader distribution's importance for performance. Wang et al. [91] noticed containerization's growth in cloud tech challenges efficient scheduling. The new CNN-BiGRU-Attention prediction model enhances accuracy, achieving 37.4% better predictions and 21.7% improved scheduling efficiency, addressing container cloud issues. Various techniques tackle scheduling. Hanafy et al. [92] observed that traditional app deployment lacked resource efficiency. Containers boosted density but consumed more energy. It may also suggest efficient container and host selection. De and Solis [93] present a cloud computing architecture using containers and a Proportional integral derivative (PID) controller-based algorithm to allocate processing resources for QoS, particularly response time efficiently. The evaluation shows promising results. In the future, they will include PID parameter optimization and testing with other orchestrators.

Container orchestration and Docker cluster management are crucial aspects of container scheduling and load balancing. These mechanisms help the infrastructure manage the lifecycle of containers, allocate resources effectively, and improve communications. Container orchestration optimizes container functionality, leading to better performance in distributed infrastructures.

3.7Microservices and docker container security

Microservices emerged as software components, while Docker has software packaged for ease of deployment and usage. In either of them, security plays a crucial role in the efficient functioning of the applications. Manu et al. [94] observed that Docker establishes multilateral security for the cloud. It addresses challenges, emphasizes privacy, and advocates collaboration. Aksakalli et al. [95] said that micro-service architecture offers benefits but presents deployment complexities. An automated approach for

efficient micro-service deployment to limited resources is proposed. Rajasekar and Palanichamy [96] developed a dynamic scheduling and resource provisioning strategy for workflow as a service (WaaS) platforms by utilizing containers to enhance task execution, guarantee scalability, and assure cost-effectiveness. Potential areas of advancement include resolving privacy and security concerns, improving fault tolerance strategies, and streamlining container and energy management. Jiang and Wu [97] investigated a multi-priority queue-based load balancing method for microservice chains to reduce latency, particularly for long chains. As Gamal et al. [84] observed, container technology revolutionized cloud computing with its benefits. Challenges include security, performance optimization, and adapting to varied workloads.

Microservices running in distributed environments like cloud infrastructure and docker containers need attack protection. In other words, security measures must be in place to ensure the smooth functioning of these services. Security vulnerabilities in such environments can lead to a decline in service performance or render services unusable. Therefore, providing security for microservices and Docker containers in distributed environments is essential.

Table 6 Summary of findings

Reference	Technique	Algorithm	Advantage	Limitation
[1]	Load balancing technique	Clustering algorithm	Energy efficiency, cost reduction, and efficient utilization of data centers	Workload prediction-based estimation of container service placement is yet to be investigated,
[3]	ACO Technique	ACO-based algorithm	Faster convergence	Heuristic parameters are now fixed and need to be improved to be dynamic in the future.
[5]	Load balancing	Scheduling algorithm	Portability, modularity, ease of use	Service displacement management is yet to be improved.
[6]	Load balancing	-	Bandwidth-independent load balancing	To be evaluated with Docker and real-world environments
[7]	Load balancing	BigLB	Continuous integration and reduction of average deployment time.	Cost efficiency and computational complexity are yet to be investigated.
[10]	Container scheduling	Optimization algorithm	Efficiency in cloud resource management.	Resource management has scope for further research
[17]	Load balancing	Particle swarm optimization (PSO)	Faster and more accessible to migrate	Data security and data analytics are to be explored in the future.
[12]	Container placement strategy	Container Scheduling Algorithm	Less CPU utilization	Memory utilization is yet to be explored
[13]	Load balancing and auto-scaling techniques	Load balancing algorithm	Improved response time	Future work focuses on developing hybrid approaches for container placement optimization.
[14]	Load scheduling	Energy-Aware scaling algorithm	Energy efficiency and docker container performance	Multiple containers running in multiple server machines is the work to be done in the future
[15]	Many-Task Computing	Container launching procedure	Socker runs containers securely	The present methodology needs to be improved with better optimization
[21]	Auto-scaling technique	Scaling algorithm	Scalability	It needs to be extended for more types of workflow applications
[29]	load balancing	Kubernetes scheduling algorithm	Fairness and resource conservation	This will be explored in the future in different domains.

Reference	Technique	Algorithm	Advantage	Limitation
[30]	Containerization	Container Scheduling Algorithm	Faster booting and lightweight virtualization	Tracing and debugging issues are the work yet to be done
[39]	Docker	Container Scheduling Algorithm	Lightweight, ease of deployment and migration	Random and binpack strategies are to be explored in the future.
[43]	Containerization	PSO	Optimized scheduling and energy efficiency	PSO has limitations and needs to enhance it.
[45]	Edge and Fog Computing	Load balancing algorithms	Speed and energy efficiency	Service management and placement are yet to be improved
[58]	Container Migration	Container pre-selection for migration and utilization regions (UR) of a node	Resource optimization	To be explored with software defined networking (SDN) based systems.
[68]	Load Balancing	Load balancing algorithms	Energy efficiency	Quality of services needs to be incorporated in the future
[70]	Load Balancing	ACO	Improved robustness	Tracking of under-utilized hosts is to be done in future
[71]	Load Balancing	ACO	Adaptable dynamism in scheduling	Cost-effectiveness and energy efficiency are to be explored in the future.
[76]	Hybrid approach	Genetic algorithm (GA) and PSO	Better resource utilization	Needs enhancement for handling dynamic containers
[77]	Load Balancing	Load balancing in Fog	Cost and energy efficiency	Performance and security parameters are to be explored.
[81]	DNL	GCS algorithm	Container selection optimality	Scheduling needs further improvement.
[89]	Load balancing	Grey Wolf optimization algorithm	Better makespan and versatility	To be extended for containerized service optimization further

Table 7 Significant performances

References	Technique	Algorithm	Significant Performances
[30]	Cloud computing	Resource scheduling optimization algorithm	Balanced load, efficiency in rendering service, and improved QoS.
[35]	Docker container	EC placement and task assignment (EPTA) algorithm	Deployment efficiency and ease of management.
[40]	Docker integration	Optimal NUMA node selection	Improved throughput and asynchronous messaging
[44]	Cloud Task Scheduling	Task scheduling	Better performance under strict deadline constraints
[46]	Container scheduling	Computing of the CPU utilization rate	Improved global performance
[48]	Cloud Computing	Orchestration	Timely migration and performance enhancement
[53]	ML	Planner algorithm	Reduced cost and scalability.
[56]	Automated deployments for microservices	Optimization algorithms.	Scalability, reliability and decoupling
[57]	Cloud Services	-	Scalability, reliability, resource conservation

References	Technique	Algorithm	Significant Performances
[58]	Dynamic Container Migration	Container Pre-Selection for Migration	High computational capability, efficiency in balancing load

4. Discussion

Most research literature has made many contributions to container load balancing. Aruna and Pradeep [17] proposed the ACO-based optimization algorithm for container load balancing. Ahmad et al. [15] observed that container-based virtualization enhances cloud computing efficiency. Bio-inspired models like grey wolf optimization with simulated annealing improve load balancing. Acharya and Suthar [4] observed that Docker containers are widely used in cloud computing, but their placement faces issues with the default spread strategy. Several other container scheduling approaches are also mentioned, and future work aims to explore memory utilization, resource reservation, and heterogeneous machines and compare them with other scheduling strategies. Choudhury et al. [63] stated that MEC faces resource assignment and load balancing challenges. ShareOn, a dynamic container migration framework, improves system response time by 15-22%. No et al. [8] discuss optimizing data processing using container technology, edge computing, and ML. However, their study is incomplete without mentioning challenges in scheduling containers, such as template design, governance, standards, and security policies. From these research insights, different problems are conceived. First, there is a need for a greedy heuristic approach for improving load balancing for containers. Second, it is essential to be involved in further research to leverage bio-inspired methods to reap their benefits in load balancing. Third, there is a need for learning-based approaches using AI to improve contain load balancing mechanisms.

Table 8 Summary of findings

Reference	Approach	Algorithm	Advantages	Disadvantages
[23]	VM	Randomized Algorithm	Scheduling performance increased	Parameter tuning is desired
[32]	Multi-objective	Container Consolidation	The success rate is high	A pluggable schedule is yet desired.
[59]	ML	ML-based algorithm	Load balancing is efficient	A multi-user environment is to be used in the future.
[90]	Long short-term memory (LSTM)	Edge network management	Energy-efficient and quality in scheduling	To be improved with further optimization.
[95]	CNN-BiGRU	Predictive algorithm	Improves scheduling latency	Load balancing efficiency to be improved.

The methodology section of this research includes various research questions answered in the literature review. The research questions address challenges in scheduling containers, current methods for container load balancing and scheduling, the role of fog and edge computing in container load balancing, the importance of container orchestration and Docker cluster management, and the significance of security for Microservices and Docker containers in distributed environments. The literature review provides information that addresses all of these research questions. Optimizing container scheduling and load balancing is essential for delivering benefits to service providers and users.

4.1Limitations of the study

The present study has certain limitations. First, the research questions provided and investigated are limited to limited aspects of the study. Their scope can be elaborated. Second, the research papers used in this study are limited to a few publishers. Data from more publishers can be elaborated. Third, the study can be improved by focusing on datasets and distributed computing environments.

A complete list of abbreviations is listed in *Appendix I.*

5.Conclusion and future work

In this paper, a systematic review is conducted that encompasses 99 peer-reviewed research articles published between 2012 and 2023. The review illuminates the field by synthesizing new insights and advancing knowledge regarding container technologies in cloud environments. Specifically, the paper addresses a range of critical issues including

challenges in container load balancing, and explores the diverse methodologies employed in container placement and orchestration. Additionally, it explores into the strategies for scheduling, load balancing, and cluster management. The review also examines dynamic resource allocation and the implementation of microservices, providing a comprehensive overview of current practices and identifying areas ripe for future research.

Some of the future suggestions are as under:

- Future research will focus on developing advanced algorithms to dynamically optimize resource allocation across various platforms, including fog, edge, and cloud computing [98].
- In the future, various heuristic and machine learning models can be developed and tested with the aim of enhancing the efficiency of load balancing techniques [99].

Acknowledgment
None.

Conflicts of interest
The authors have no conflicts of interest to declare.

Dataset availability
None.

Author's contribution statement
Neelima Gogineni: Conceptualization, investigation, writing–review and editing. **Saravanan Madderi Sivalingam**: Conceptualization, investigation, and supervision.

References
[1] Singh N, Hamid Y, Juneja S, Srivastava G, Dhiman G, Gadekallu TR, et al. Load balancing and service discovery using docker swarm for microservice based big data applications. Journal of Cloud Computing. 2023; 12(1):4.
[2] Wan X, Guan X, Wang T, Bai G, Choi BY. Application deployment using microservice and docker containers: framework and optimization. Journal of Network and Computer Applications. 2018; 119:97-109.
[3] Fazio M, Celesti A, Ranjan R, Liu C, Chen L, Villari M. Open issues in scheduling microservices in the cloud. IEEE Cloud Computing. 2016; 3(5):81-8.
[4] Acharya J, Suthar AC. Container scheduling algorithm in docker based cloud. Webology. 2022; 19(2):627-37.
[5] Watada J, Roy A, Kadikar R, Pham H, Xu B. Emerging trends, techniques and open issues of containerization: a review. IEEE Access. 2019; 7:152443-72.
[6] Rabiu S, Yong CH, Mohamad SM. A cloud-based container microservices: a review on load-balancing and auto-scaling issues. International Journal of Data Science. 2022; 3(2):80-92.
[7] Piraghaj SF, Dastjerdi AV, Calheiros RN, Buyya R. Efficient virtual machine sizing for hosting containers as a service (services 2015). In world congress on services 2015 (pp. 31-8). IEEE.
[8] No JK, Gedung PB, Indonesia M. Utilization docker swarms in a container technology system to problem solving load balance. Journal of Theoretical and Applied Information Technology. 2022; 100(18):5201-8.
[9] Sureshkumar M, Rajesh P. Optimizing the docker container usage based on load scheduling. In 2nd international conference on computing and communications technologies (ICCCT) 2017 (pp. 165-8). IEEE.
[10] Azab A. Enabling docker containers for high-performance and many-task computing. In IEEE international conference on cloud engineering (IC2E) 2017(pp. 279-85). IEEE.
[11] Li L, Chen J, Yan W. A particle swarm optimization-based container scheduling algorithm of docker platform. In proceedings of the 4th international conference on communication and information processing 2018 (pp. 12-7).
[12] Kaewkasi C, Chuenmuneewong K. Improvement of container scheduling for docker using ant colony optimization. In 9th international conference on knowledge and smart technology (KST) 2017 (pp. 254-9). IEEE.
[13] Qian J, Wang Y, Wang X, Zhang P, Wang X. Load balancing scheduling mechanism for openstack and docker integration. Journal of Cloud Computing. 2023; 12(1):67.
[14] Li Q, Fang Y. Multi-algorithm collaboration scheduling strategy for docker container. In international conference on computer systems, electronics and control (ICCSEC) 2017 (pp. 1367-71). IEEE.
[15] Ahmad I, Alfailakawi MG, Almutawa A, Alsalman L. Container scheduling techniques: a survey and assessment. Journal of King Saud University-Computer and Information Sciences. 2022; 34(7):3934-47.
[16] Guo Y, Yao W. A container scheduling strategy based on neighborhood division in micro service. In NOMS IEEE/IFIP network operations and management symposium 2018 (pp. 1-6). IEEE.
[17] Aruna K, Pradeep G. Ant colony optimization-based light weight container (ACO-LWC) algorithm for efficient load balancing. Intelligent Automation & Soft Computing. 2022; 34(1):201-19.
[18] Liu B, Li P, Lin W, Shu N, Li Y, Chang V. A new container scheduling algorithm based on multi-objective optimization. Soft Computing. 2018; 22:7741-52.
[19] Patra MK, Patel D, Sahoo B, Turuk AK. A randomized algorithm for load balancing in containerized cloud. In 10th international conference on cloud computing, data science & engineering (confluence) 2020(pp. 410-4). IEEE.

[20] Cérin C, Menouer T, Saad W, Abdallah WB. A new docker swarm scheduling strategy. In 7th international symposium on cloud and service computing (SC2) 2017 (pp. 112-7). IEEE.
[21] Zhang D, Yan BH, Feng Z, Zhang C, Wang YX. Container oriented job scheduling using linear programming model. In international conference on information management (ICIM) 2017 (pp. 174-80). IEEE.
[22] Marathe N, Gandhi A, Shah JM. Docker swarm and Kubernetes in cloud computing environment. In 3rd international conference on trends in electronics and informatics (ICOEI) 2019 (pp. 179-84). IEEE.
[23] Bernstein D. Containers and cloud: from LXC to docker to Kubernetes. IEEE Cloud Computing. 2014; 1(3):81-4.
[24] Zhang R, Zhong AM, Dong B, Tian F, Li R. Container-VM-PM architecture: a novel architecture for docker container placement. In cloud computing-CLOUD: 11th international conference, held as part of the services conference federation, SCF 2018, Seattle, WA, USA, Proceedings 2018 (pp. 128-40). Springer International Publishing.
[25] Zhiyong C, Xiaolan X. Overview of container cloud task scheduling. In proceedings of the 2020 artificial intelligence and complex systems conference 2020 (pp. 50-5). ACM.
[26] Adhikari M, Srirama SN. Multi-objective accelerated particle swarm optimization with a container-based scheduling for internet-of-things in cloud environment. Journal of Network and Computer Applications. 2019; 137:35-61.
[27] Menouer T. KCSS: Kubernetes container scheduling strategy. The Journal of Supercomputing. 2021; 77(5):4267-93.
[28] Yang S, Wang X, An L, Zhang G. Yun: a high-performance container management service based on OpenStack. In fourth international conference on data science in cyberspace (DSC) 2019 (pp. 202-9). IEEE.
[29] Smimite O, Afdel K. Hybrid solution for container placement and load balancing based on ACO and bin packing. International Journal of Advanced Computer Science and Applications. 2020; 11(11).
[30] Mattia GP, Beraldi R. P2PFaaS: a framework for FaaS peer-to-peer scheduling and load balancing in fog and edge computing. SoftwareX. 2023; 21:101290.
[31] Li Y, Xia Y. Auto-scaling web applications in hybrid cloud based on docker. In 5th international conference on computer science and network technology (ICCSNT) 2016 (pp. 75-9). IEEE.
[32] Patra MK, Misra S, Sahoo B, Turuk AK. GWO-based simulated annealing approach for load balancing in cloud for hosting container as a service. Applied Sciences. 2022; 12(21):11115.
[33] Lin J, Cui D, Peng Z, Li Q, He J, Guo M. Virtualized resource scheduling in cloud computing environments: an review. In conference on telecommunications, optics and computer science (TOCS) 2020 (pp. 303-8). IEEE.
[34] Yang S, Wang X, Wang X, An L, Zhang G. High-performance docker integration scheme based on OpenStack. World Wide Web. 2020; 23(4):2593-632.
[35] Kim WY, Lee JS, Huh EN. Study on proactive auto scaling for instance through the prediction of network traffic on the container environment. In proceedings of the 11th international conference on ubiquitous information management and communication 2017 (pp. 1-8). ACM.
[36] Talukder A, Abedin SF, Munir MS, Hong CS. Dual threshold load balancing in SDN environment using process migration. In international conference on information networking (ICOIN) 2018 (pp. 791-6). IEEE.
[37] Elmagzoub MA, Syed D, Shaikh A, Islam N, Alghamdi A, Rizwan S. A survey of swarm intelligence based load balancing techniques in cloud computing environment. Electronics. 2021; 10(21):1-46.
[38] Song X, Ma Y, Teng D. A load balancing scheme using federate migration based on virtual machines for cloud simulations. Mathematical Problems in Engineering. 2015; 2015(1):506432.
[39] Afzal S, Kavitha G. Load balancing in cloud computing–a hierarchical taxonomical classification. Journal of Cloud Computing. 2019; 8(1):1-24.
[40] Naik KD, Sahoo RR, Kuana SK. A bio inspired approach for load balancing in container as a service cloud computing model. International Research Journal on Advanced Science Hub. 2023; 5(5):426-33.
[41] Polikarpova N, Tschannen J, Furia CA. A fully verified container library. In international symposium on formal methods 2015 (pp. 414-34). Cham: Springer International Publishing.
[42] Lee YJ, Park GY, Song HK, Youn HY. A load balancing scheme for distributed simulation based on multi-agent system. In 36th annual computer software and applications conference workshops 2012 (pp. 613-8). IEEE.
[43] Tao Y, Wang X, Xu X, Chen Y. Dynamic resource allocation algorithm for container-based service computing. In 13th international symposium on autonomous decentralized system (ISADS) 2017 (pp. 61-7). IEEE.
[44] Zhiyong C, Xiaolan X. An improved container cloud resource scheduling strategy. In proceedings of the 4th international conference on intelligent information processing 2019 (pp. 383-7).
[45] Li P, Song J, Xu H, Dong L, Zhou Y. Resource scheduling optimisation algorithm for containerised microservice architecture in cloud computing. International Journal of High Performance Systems Architecture. 2018; 8(1-2):51-8.
[46] Mao Y, Oak J, Pompili A, Beer D, Han T, Hu P. Draps: dynamic and resource-aware placement scheme for docker containers in a heterogeneous cluster. In 36th international performance computing and communications conference (IPCCC) 2017 (pp. 1-8). IEEE.
[47] Tihfon GM, Kim J, Kim KJ. A new virtualized environment for application deployment based on

docker and AWS. In information science and applications (ICISA) 2016 (pp. 1339-49). Springer Singapore.
[48] Long X, Liu B, Jiang F, Zhang Q, Zhi X. FPGA virtualization deployment based on docker container technology. In 5th international conference on mechanical, control and computer engineering (ICMCCE) 2020 (pp. 473-6). IEEE.
[49] Imdoukh M, Ahmad I, Alfailakawi MG. Machine learning-based auto-scaling for containerized applications. Neural Computing and Applications. 2020; 32(13):9745-60.
[50] Mondesire SC, Angelopoulou A, Sirigampola S, Goldiez B. Combining virtualization and containerization to support interactive games and simulations on the cloud. Simulation Modelling Practice and Theory. 2019; 93:233-44.
[51] El KS, El MI, Salah K, Hanini M. Dynamic scalability model for containerized cloud services. Arabian Journal for Science and Engineering. 2020; 45(12):10693-708.
[52] Celesti A, Mulfari D, Galletta A, Fazio M, Carnevale L, Villari M. A study on container virtualization for guarantee quality of service in cloud-of-things. Future Generation Computer Systems. 2019; 99:356-64.
[53] Xiaojing XI, Govardhan SS. A service mesh-based load balancing and task scheduling system for deep learning applications. In 20th international symposium on cluster, cloud and internet computing (CCGRID) 2020 (pp. 843-9). IEEE.
[54] Thongthavorn W, Rattanatamrong P. Multi-container application migration with load balanced and adaptive parallel TCP. In international conference on high performance computing & simulation (HPCS) 2019 (pp. 55-62). IEEE.
[55] Vaishali KR, Rammohan SR, Natrayan L, Usha D, Niveditha VR. Guided container selection for data streaming through neural learning in cloud. International Journal of System Assurance Engineering and Management. 2021:1-7.
[56] Hofer F, Sehr MA, Iannopollo A, Ugalde I, Sangiovanni-vincentelli A, Russo B. Industrial control via application containers: migrating from bare-metal to IAAS. *In international conference on cloud computing technology and science (CloudCom)*, Sydney, NSW, Australia, 2019 (pp. 62-9). *IEEE.*
[57] Ma H, Wang L, Tak BC, Wang L, Tang C. Auto-tuning performance of MPI parallel programs using resource management in container-based virtual cloud. In 9th international conference on cloud computing (CLOUD) 2016 (pp. 545-52). IEEE.
[58] Kanchanadevi P, Kumar VA, Kumar GA. Optimal resource allocation and load balancing for a container as a service in a cloud computing. Journal of Critical Reviews. 2020; 7(4):900-2.
[59] Souza A, Wen Z, Cacho N, Romanovsky A, James P, Ranjan R. Using osmotic services composition for dynamic load balancing of smart city applications. In 11th conference on service-oriented computing and applications (SOCA) 2018 (pp. 145-52). IEEE.
[60] Baburao D, Pavankumar T, Prabhu CS. Load balancing in the fog nodes using particle swarm optimization-based enhanced dynamic resource allocation method. Applied Nanoscience. 2023; 13(2):1045-54.
[61] Pérez DPR, García-galán S, Muñoz-expósito JE, Marchewka A, Ruiz-reyes N. Smart containers schedulers for microservices provision in cloud-fog-IoT networks. Challenges and Opportunities. Sensors. 2020; 20(6):1-21.
[62] Li X, Jiang Y, Ding Y, Wei D, Ma X, Li W. Application research of docker based on mesos application container cluster. In international conference on computer vision, image and deep learning (CVIDL) 2020 (pp. 476-9). IEEE.
[63] Choudhury S, Maheshwari S, Seskar I, Raychaudhuri D. Shareon: shared resource dynamic container migration framework for real-time support in mobile edge clouds. IEEE Access. 2022; 10:66045-60.
[64] Baburao D, Pavankumar T, Prabhu CS. Survey on service migration, load optimization and load balancing in fog computing environment. In 5th international conference for convergence in technology (I2CT) 2019 (pp. 1-5). IEEE.
[65] Singh A, Aujla GS, Bali RS. Container-based load balancing for energy efficiency in software-defined edge computing environment. Sustainable Computing: Informatics and Systems. 2021; 30:100463.
[66] Ma Z, Shao S, Guo S, Wang Z, Qi F, Xiong A. Container migration mechanism for load balancing in edge network under power internet of things. IEEE Access. 2020; 8:118405-16.
[67] Li K, Chang C, Yun K, Zhang J. Research on container migration mechanism of power edge computing on load balancing. In 6th international conference on cloud computing and big data analytics (ICCCBDA) 2021 (pp. 386-90). IEEE.
[68] Singh J, Singh P, Amhoud EM, Hedabou M. Energy-efficient and secure load balancing technique for SDN-enabled fog computing. Sustainability. 2022; 14(19):1-22.
[69] Alqahtani F, Amoon M, Nasr AA. Reliable scheduling and load balancing for requests in cloud-fog computing. Peer-to-Peer Networking and Applications. 2021; 14(4):1905-16.
[70] Dlamini T, Vilakati S. LSTM-based traffic load balancing and resource allocation for an edge system. Wireless Communications and Mobile Computing. 2020;2020(1):8825396.
[71] Chen CH, Liu CT. A 3.5-tier container-based edge computing architecture. Computers & Electrical Engineering. 2021; 93:107227.
[72] Khan A. Key characteristics of a container orchestration platform to enable a modern application. IEEE Cloud Computing. 2017; 4(5):42-8.
[73] Hussein MK, Mousa MH, Alqarni MA. A placement architecture for a container as a service (CaaS) in a cloud environment. Journal of Cloud Computing. 2019; 8:1-5.
[74] Zhong Z, Buyya R. A cost-efficient container orchestration strategy in Kubernetes-based cloud

computing infrastructures with heterogeneous resources. ACM Transactions on Internet Technology (TOIT). 2020; 20(2):1-24.
[75] Littley M, Anwar A, Fayyaz H, Fayyaz Z, Tarasov V, Rupprecht L, et al. Bolt: towards a scalable docker registry via hyperconvergence. In 12th international conference on cloud computing (CLOUD) 2019 (pp. 358-66). IEEE.
[76] Louati T, Abbes H, Cérin C, Jemni M. Lxcloud-cr: towards linux containers distributed hash table based checkpoint-restart. Journal of Parallel and Distributed Computing. 2018; 111:187-205.
[77] Louati T, Abbes H, Cérin C. LXCloudFT: towards high availability, fault tolerant cloud system based Linux containers. Journal of Parallel and Distributed Computing. 2018; 122:51-69.
[78] Martin JP, Kandasamy A, Chandrasekaran K. Exploring the support for high performance applications in the container runtime environment. Human-centric Computing and Information Sciences. 2018; 8:1-5.
[79] Wan F, Wu X, Zhang Q. Chain-oriented load balancing in microservice system. In world conference on computing and communication technologies (WCCCT) 2020 (pp. 10-4). IEEE.
[80] Kaur K, Guillemin F, Sailhan F. Container placement and migration strategies for cloud, fog, and edge data centers: a survey. International Journal of Network Management. 2022; 32(6):e2212.
[81] Muthakshi S, Mahesh K. Container selection processing implementing extensive neural learning in cloud services. Materials Today: Proceedings. 2023; 80:1868-71.
[82] Queiroz R, Cruz T, Mendes J, Sousa P, Simões P. Container-based virtualization for real-time industrial systems-a systematic review. ACM Computing Surveys. 2023; 56(3):1-38.
[83] Rizvi H, Rahman HU. Load balancing on cloud computing. International Journal of Creative Research Thoughts. 2023; 11(4):927-31.
[84] Gamal M, Rizk R, Mahdi H, Elhady B. Bio-inspired load balancing algorithm in cloud computing. In proceedings of the international conference on advanced intelligent systems and informatics 2018 (pp. 579-89). Springer International Publishing.
[85] Peinl R, Holzschuher F, Pfitzer F. Docker cluster management for the cloud-survey results and own solution. Journal of Grid Computing. 2016; 14(2):265-82.
[86] Tihfon GM, Park S, Kim J, Kim YM. An efficient multi-task PaaS cloud infrastructure based on docker and AWS ECS for application deployment. Cluster Computing. 2016; 19:1585-97.
[87] Pahl C, Lee B. Containers and clusters for edge cloud architectures-a technology review. In 3rd international conference on future internet of things and cloud 2015 (pp. 379-86). IEEE.
[88] Zheng G, Fu Y, Wu T. Research on docker cluster scheduling based on self-define kubernetes scheduler. In journal of physics: conference series 2021 (pp. 1-6). IOP Publishing.
[89] Hu Y, De LC, Zhao Z. Multi-objective container deployment on heterogeneous clusters. In 19th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID) 2019 (pp. 592-9). IEEE.
[90] Nguyen N, Kim T. Toward highly scalable load balancing in kubernetes clusters. IEEE Communications Magazine. 2020; 58(7):78-83.
[91] Wang L, Guo S, Zhang P, Yue H, Li Y, Wang C, et al. An efficient load prediction-driven scheduling strategy model in container cloud. International Journal of Intelligent Systems. 2023; 2023(1):5959223.
[92] Hanafy WA, Mohamed AE, Salem SA. Novel selection policies for container-based cloud deployment models. In 13th international computer engineering conference (ICENCO) 2017 (pp. 237-42). IEEE.
[93] De AMC, Solis P. An algorithm based on response time and traffic demands to scale containers on a cloud computing system. In 15th international symposium on network computing and applications (NCA) 2016 (pp. 343-50). IEEE.
[94] Manu AR, Patel JK, Akhtar S, Agrawal VK, Murthy KB. Docker container security via heuristics-based multilateral security-conceptual and pragmatic study. In international conference on circuit, power and computing technologies (ICCPCT) 2016 (pp. 1-14). IEEE.
[95] Aksakalli IK, Celik T, Can AB, Tekinerdogan B. Systematic approach for generation of feasible deployment alternatives for microservices. IEEE Access. 2021; 9:29505-29.
[96] Rajasekar P, Palanichamy Y. Scheduling multiple scientific workflows using containers on IaaS cloud. Journal of Ambient Intelligence and Humanized Computing. 2021; 12:7621-36.
[97] Jiang C, Wu P. A fine-grained horizontal scaling method for container-based cloud. Scientific Programming. 2021; 2021(1):6397786.
[98] Casalicchio E, Iannucci S. The state-of-the-art in container technologies: application, orchestration and security. Concurrency and Computation: Practice and Experience. 2020; 32(17):e5668.
[99] Kuramoto K, Furuichi M, Kakuda K. Efficient load-balancing scheme for multi-agent simulation systems. Computer Modeling in Engineering & Sciences. 2015; 106(3):169-86.

Mrs. Neelima Gogineni is an Assistant Professor in the Department of Computer Science & Engineering at Gokaraju Rangaraju College of Engineering and Technology, Hyderabad, Telangana, India. She earned her B.Tech in 2005 and her M.Tech in 2012. Currently, she is pursuing a Ph.D. from Saveetha Institute of Medical and Technical Sciences, Chennai, Tamil Nadu. Mrs. Gogineni has over 18 years of teaching experience, having worked at TKR College of Engineering and Technology, Malla Reddy

Institute of Technology, and currently at Gokaraju Rangaraju Institute of Engineering and Technology. She is a two-time recipient of the Best Faculty Award from TKR College of Engineering and Technology and Malla Reddy Institute of Engineering Technology. Her research interests include Machine Learning, Cloud Computing, the Internet of Things (IoT), Artificial Intelligence (AI), and Deep Learning. Throughout her career, she has taught a wide range of subjects, including C & Data Structures, Object-Oriented Programming through Java, Database Management Systems, Computer Networks, Digital Communications and Computer Networks, Computer Graphics, Software Testing and Maintenance, and Cloud Computing.
Email: yangalaneni.neelima@gmail.com

Saravanan Madderi Sivalingam is a Professor in the Department of Computer Science and Engineering at Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, India. He is an accomplished researcher and academic in the field of Computer Science and Engineering. He has authored two books, edited four papers, published 153 papers in refereed international journals, presented 26 papers at refereed conference proceedings, and holds 16 patents. Dr. Saravanan has also contributed to 21 major invited talks and technical reports, and has earned a Cloud Foundation Certificate and a Data Analytics IBM Cognos Certificate. Since 2017, he has been serving as a Professor at Haramaya University in East Africa, within the School of Computing, for two years. Dr. Saravanan is a member of IEEE, ISTE, and IET, and serves as a Student Branch Councillor for IEEE and as an Innovation Ambassador of the Institution's Innovation Council (IIC) at SIMATS. Dr. Saravanan's expertise encompasses Artificial Intelligence, Data Science, and Cloud-based Technologies. He is recognized for his innovative work in developing a cost-effective cabinet dyeing process using process mining techniques, for which he developed a new algorithm called "LinkRuleMiner." Dr. Saravanan leads a dynamic research group focused on advancing AI-based products. His lab's groundbreaking research has been published in leading peer-reviewed journals. He has received top faculty and researcher awards from national and international societies. Dr. Saravanan is actively involved in mentoring graduate students and collaborating with industry partners to bridge the gap between academia and practical applications..
Email: saranenadu@mail.com

Appendix I

S. No.	Abbreviation	Description
1	ACO	Ant Colony Optimization
2	AI	Artificial Intelligence
3	AWS	Amazon Web Services
4	BOLT	Build Operate Lease-Transfer
5	CaaS	Container as a Service
6	CPU	Central Processing Unit
7	DRAPS	Dynamic and resource-aware placement scheme
8	DNL	Deep Neural Learning
9	DTLB	Dual Threshold Load Balancing
10	ECAIoT	Edge Computing AioT
11	EDRAM	Embedded Dynamic Random-Access Memory
12	FPGA	Field Programmable Gate Array
13	GENM	Green-based Edge Network Management
14	GA	Genetic Algorithm
15	HMBA	Honeybee Mating Algorithm
16	HPC	High-Performance Computing
17	HLA	High-Level Architecture
18	IoTaaS	Internet of Things as a Service
19	IoT	Internet of Things
20	ITO	Information Technology Outsourcing
21	KaPR	Ka-Band Precipitation Radar
22	KuPR	Ku-Band Precipitation Radar
23	LSTM	Long Short-Term Memory
24	LTLB	Least Traffic Load Balancing
25	MCSS	Multi-Algorithm Collaboration Scheduling Strategy
26	MEC	Multi-access Edge Computing
27	MPI	Message-Passing Interface
28	ML	Machine Learning
29	PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
30	QoE	Quality of Experience
31	QoS	Quality of Service
32	SBSs	Small Cell Base Stations
33	SDN	Software Defined Networking
34	SLA	Service Level Agreement
35	UR	Utilization Regions
36	VM	Virtual Machine
37	WAN	Wide Area Network