Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.043627

REVIEW

Cloud Datacenter Selection Using Service Broker Policies: A Survey

Salam Al-E'mari[1], Yousef Sanjalawe[2,*], Ahmad Al-Daraiseh[3], Mohammad Bany Taha[4] and Mohammad Aladaileh[2]

[1]Department of Information Security, Faculty of IT, University of Petra, Amman, 11196, Jordan

[2]Department of Cybersecurity, School of IT, American University of Madaba (AUM), Amman, 11821, Jordan

[3]Department of Computer Science, School of IT, American University of Madaba (AUM), Amman, 11821, Jordan

[4]Department of Data Science and Artificial Intelligence, School of IT, American University of Madaba (AUM), Amman, 11821, Jordan

*Corresponding Author: Yousef Sanjalawe. Email: y.sanjalawe@aum.edu.jo

Received: 07 July 2023 Accepted: 15 September 2023 Published: 30 December 2023

ABSTRACT

Amid the landscape of Cloud Computing (CC), the Cloud Datacenter (DC) stands as a conglomerate of physical servers, whose performance can be hindered by bottlenecks within the realm of proliferating CC services. A linchpin in CC's performance, the Cloud Service Broker (CSB), orchestrates DC selection. Failure to adroitly route user requests with suitable DCs transforms the CSB into a bottleneck, endangering service quality. To tackle this, deploying an efficient CSB policy becomes imperative, optimizing DC selection to meet stringent Quality-of-Service (QoS) demands. Amidst numerous CSB policies, their implementation grapples with challenges like costs and availability. This article undertakes a holistic review of diverse CSB policies, concurrently surveying the predicaments confronted by current policies. The foremost objective is to pinpoint research gaps and remedies to invigorate future policy development. Additionally, it extensively clarifies various DC selection methodologies employed in CC, enriching practitioners and researchers alike. Employing synthetic analysis, the article systematically assesses and compares myriad DC selection techniques. These analytical insights equip decision-makers with a pragmatic framework to discern the apt technique for their needs. In summation, this discourse resoundingly underscores the paramount importance of adept CSB policies in DC selection, highlighting the imperative role of efficient CSB policies in optimizing CC performance. By emphasizing the significance of these policies and their modeling implications, the article contributes to both the general modeling discourse and its practical applications in the CC domain.

KEYWORDS

Cloud computing; cloud service broker; datacenter selection; quality-of-service; user request

1 Introduction

CC has gained significant popularity as a technology that enables users to conveniently access a shared pool of computing resources. These resources, which include networks, servers, storage, applications, and services, are configurable and available on a pay-per-usage basis. The National

Institute of Standards and Technology provides a definition for CC, describing it as a model that facilitates on-demand access to diverse computing resources. With CC, these resources can be quickly allocated and released with minimal need for management involvement or direct interaction with the service provider [1]. The concept of CC was first introduced in 1997 by Dr. Chellapa at Texas University and has evolved through various stages, including grid computing, utility computing, application service provider, and Software as a Service (SaaS) [2]. As depicted in Fig. 1, The architecture of cloud computing encompasses the structural framework and components necessary for delivering on-demand computing resources via the internet. It involves a diverse set of hardware and software elements that collaborate to offer scalable and flexible services to users. This architecture comprises layers such as infrastructure, platform, and software, supported by virtualization and networking technologies. A comprehensive understanding of cloud computing architecture empowers organizations to leverage cloud services efficiently, enabling them to take advantage of cost-efficiency, scalability, and accessibility in their computing operations.

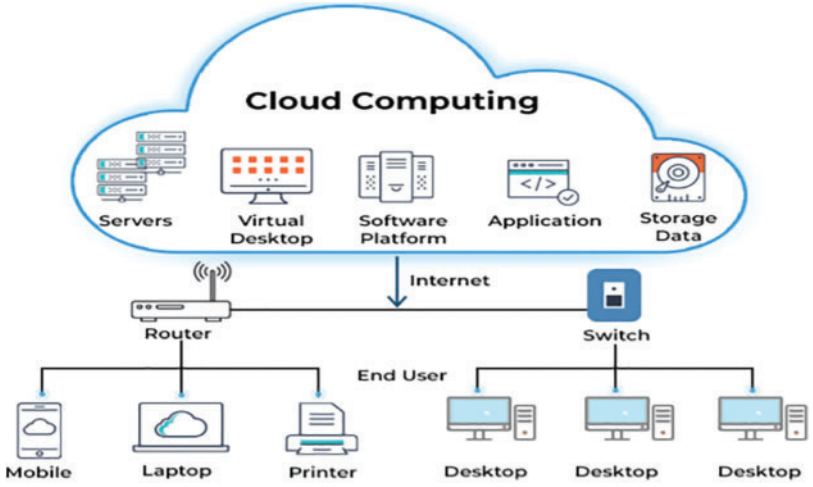

Figure 1: Architecture of CC

CC encompasses three primary service models, as depicted in Fig. 2. These models include Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS) [3,4]. IaaS grants client's access to robust and up-to-date computer infrastructures via the internet [5]. SaaS refers to internet-based software provision, allowing users to select web-based applications without purchasing the complete software package. On the other hand, PaaS provides users with a platform to develop and operate services without the burden of maintaining the infrastructure typically associated with launching an application [6]. CC is specifically designed for remotely managed and scalable IT resources, employing decentralized IT resources and virtualization technology [7]. The services provided through CC can be intricate and may involve service level agreements (SLAs) to ensure system functionality. Nonetheless, SLAs can be breached due to dynamic attributes, hardware and software malfunctions, and workloads. Challenges may arise during SLA negotiations and service performance, leading to inefficient resource utilization, service execution delays, and subpar performance [8,9].

It is extremely important to efficiently utilize CC's available resources. One such resource is DC, which is selected by a CSB. The CSB manages the routing between the DC and the user in the CC environment. Essentially, a CSB executes a DC selection policy [10]. Online services offered by CC providers are accessed through web browsers, while the applications and required data are stored

on storage area network devices. Hundreds of DCs host different servers and applications used by users of the cloud [4,7]. There are various DCs and CC resources available for use by users, who are primarily concerned with how to use these resources, services, and applications to meet their needs rather than the actual location or maintenance of the DCs and resources [11,12]. With CC, applications are run on servers hosted in DCs rather than on the local laptop or desktop computer, which may reduce the need for large processing power and storage on personal devices [13]. A DC is a centralized container that includes both virtual and physical resources for managing, storing, and distributing data and information organized around a particular subject or business [1,14–18]. For example, the National Climatic DC is a well-known DC [19] that stores the world's largest archive of weather information. Private DCs may be located within institutional or specialized facilities. As shown in Fig. 3, CC revolutionizes the access and utilization of computing resources by offering a variety of features. These features include efficient resource allocation, on-demand self-service, high availability, pay-as-you-go pricing, robust security measures, and economical benefits. Through cloud computing, users have the flexibility to allocate and manage resources according to their specific requirements, ensuring optimal utilization and cost-effectiveness. The on-demand self-service capability allows users to quickly provision and scale resources without human intervention. Cloud services prioritize availability, ensuring that resources are consistently accessible and reliable. The pay-as-you-go pricing model enables users to pay only for the resources they consume, offering flexibility and cost savings. Security is a top priority for cloud providers, who implement robust measures to safeguard data and systems.

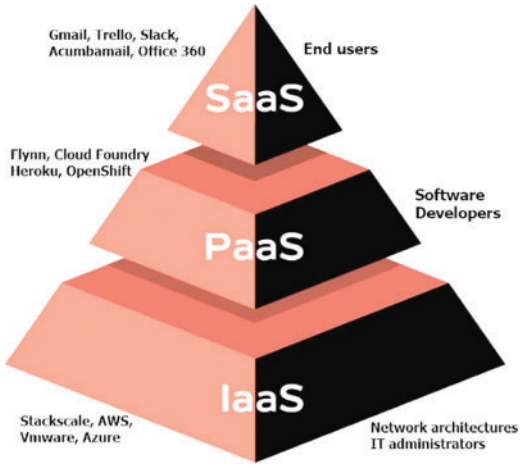

Figure 2: Models of CC

The main architecture of a DC consists of a Load Balancer (LB), Physical Machines (PMs), and Virtual Machines (VMs), as shown in Fig. 4. When a user requests a specific service, the CSB policy is responsible for selecting the best DC to provide the service efficiently. The DC includes several PMs, each of which hosts several VMs. A Load Balancing Server (LBS) is used to route incoming user requests to the appropriate PMs [10,20]. When a user request is received, it is placed in a queue waiting until a VM becomes available to process the request. If the global queue is full, the user request may be dropped immediately. Before being selected, the VMs must be provisioned and the LBS must be informed of the available resources. The LBS then routes the user request to the appropriate VM [20]. Once the user request is routed to the desired service, the corresponding subtasks are sent directly from the queue to the appropriate VM [21]. This architecture is widely used in CC research (e.g., [21,22]).

The LBS has a different function from that of a CSB. While the CSB is responsible for selecting the appropriate DC to provide the service to a specific user request, the LBS is responsible for selecting the most appropriate VM to process the request efficiently. In other words, the CSB policy has a wider scope than the LBS.

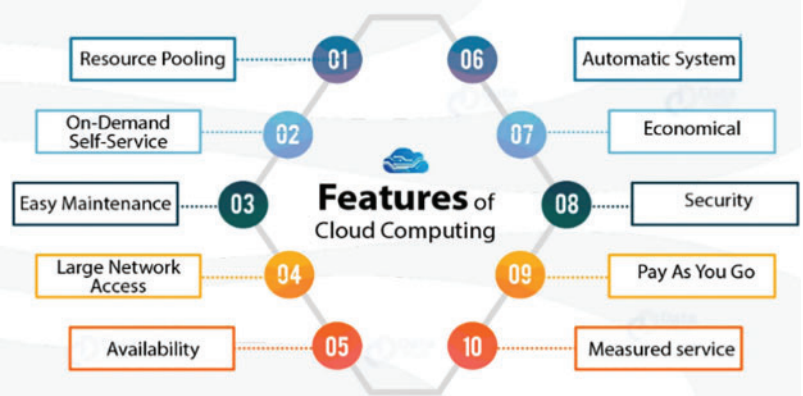

Figure 3: Features of CC

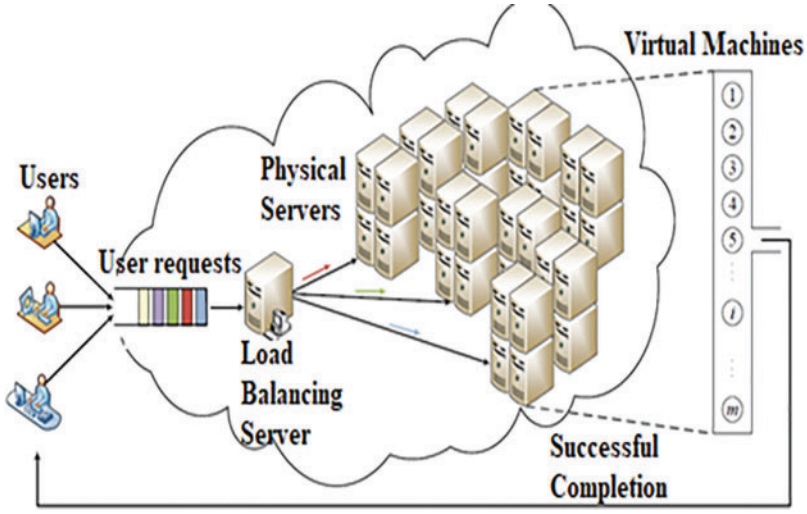

Figure 4: Architecture of the cloud's DC

For instance, as depicted in Fig. 5, the geographical distribution of Amazon's cloud data centers has transformed the global delivery of cloud services. By strategically dispersing their data centers across different locations, Amazon ensures that their customers benefit from fast access, increased reliability, and enhanced performance. This distribution enables businesses and individuals to effectively utilize Amazon's cloud services from multiple regions, facilitating efficient storage, processing, and distribution of data. With a worldwide presence, Amazon's geo-distributed cloud data centers offer seamless scalability, robust disaster recovery capabilities, and compliance with data sovereignty regulations. This expansion of infrastructure has positioned Amazon as a leading provider in the cloud computing industry, catering to the diverse requirements of customers worldwide [23].

Figure 5: Ge-distribution of cloud DC of Amazon

To ensure high levels of QoS, efficient resource utilization, and high performance at minimal cost for CC services, it is important to select the optimal DC among the available DCs [24]. Online services often use mapping nodes to identify the IP addresses that belong to users and manage DC selection. Alternatively, the process of DC selection can be outsourced to third parties [25,26], or cloud service providers (CSPs) [27]. While many previous works have attempted to propose new or enhanced CSB policies for efficient DC selection, they still face various challenges [24,27,28]. Therefore, the main objectives of this research are to:

- Investigate existing CSB policies for DC selection.
- Propose a new classification taxonomy for CSB policies.
- Clarify the strengths and weaknesses of CSB policies in each category.
- Outline areas for future research to improve CSB policies.

1.1 Background of Cloud Data Center, Isolation and Virtualization

This section provides an introduction to cloud computing and virtualized data centers. Cloud refers to an internet-based IT infrastructure where data processing and storage services are performed at the data center level. This entails a shift from local computer-based IT operations to distributed installations managed by third-party IT services. The cloud industry is experiencing significant growth, with the establishment of large-scale data centers that can accommodate over a million servers to meet increasing customer demands. Cloud architecture relies on consolidated or geographically dispersed data centers, each consisting of servers organized into racks. Virtualization techniques are employed to partition a server into virtual components such as VMs and/or containers.

Virtualization is a key technology that has played a significant role in the development of cloud computing. It enables resource sharing within cloud systems, allowing multiple applications to run on different platforms. This technology offers the capability to divide hardware resources by emulating either partial or complete machines, resulting in the creation of isolated execution environments. In contrast, traditional architectures support only one natively installed operating system on each physical machine to ensure stability and consistency [29]. Native environments generally offer faster performance compared to virtualized ones [30]. In a virtualized environment, CPU utilization can see an increase of 40% to 60% [31]. However, resource allocation in virtualized environments tends to align more closely with the actual needs of specific tasks, unlike traditional architectures where an entire host is dedicated to running the same tasks [29].

Within the realm of cloud computing, there are various forms and levels of virtualization, such as network virtualization, storage virtualization, and more. For our focus here, we concentrate on

hardware and operating system level virtualization. As illustrated in Fig. 6, this domain encompasses two primary types of virtualizations: VM-based virtualization and container-based virtualization:

1. VM-based virtualization: VMs are crucial components within the cloud infrastructure [29]. They are virtual representations of computer devices created using specialized software and have the ability to run applications. VM-based virtualization is the predominant method used in cloud environments, where physical resources are allocated virtually at the hardware level through a hypervisor [32]. A hypervisor, also known as a Virtual Machine Monitor (VMM), facilitates the sharing of a single physical machine by multiple operating systems through the creation and management of virtual machines. Acting as a management layer, the hypervisor oversees all instantiated VMs, each running an independent operating system [33]. There are two primary types of hypervisors: type 1 and type 2. The first type, known as a "bare metal" or native hypervisor, runs directly on the underlying hardware. The second type, called a hosted hypervisor, operates as a software layer within another operating system. Above the hardware, there is a guest operating system. Compared to type 2 hypervisors, bare metal hypervisors offer better performance since they eliminate the need for an additional software layer between the host hardware and VMs [28]. Type 1 hypervisors are generally considered more secure and efficient in terms of resource usage but require specific configurations on the host machine to be utilized [28]. Well-known hypervisors like VMware ESX, KVM, Xen, and Hyper-V belong to the bare metal category [34].
2. Container-based virtualization: We now turn our attention to lightweight operating system level virtualization, which revolves around container technology. Containers are a notable innovation in the realm of cloud computing [35]. They package self-contained segments of applications, along with their dependencies, making them ready for deployment [36]. Within this containerized environment, a container engine oversees multiple isolated user spaces or container instances [30] that coexist on the same machine, sharing the underlying operating system kernel, including the root file system, libraries, and common files [37]. This approach eliminates the need for duplicating code and brings forth several advantages. Containerization has gained significant popularity among providers, leading to the availability of container services from various providers such as Google Container Engine, Docker Datacenter, and Amazon Elastic Container Service (ECS). The emergence of containers has also introduced a new cloud service model called Container-as-a-Service (CaaS), which enables applications to break free from dependencies on PaaS and promotes independence [33]. Prominent providers offering CaaS include Google and Amazon [38]. Containers can either host an entire application or execute individual components of a distributed application within separate containers, which are then deployed on physical machines (PMs). Docker, the most widely used container implementation, is an open-source management tool that automates application deployment [39]. An alternative container technology called Rocket is also gaining traction as a secure, interoperable, and open container solution, competing with Docker [34].
3. Isolation and performance discussion: Both VM-based and container-based virtualization methods provide isolation, establishing separate spaces within the machine. While a hypervisor introduces an extra layer of virtualization, containers offer isolated environments at the operating system level. Containers achieve efficient resource usage and lightweight operation by isolating and managing processes at the core level of the operating system. This isolation is accomplished through the use of Namespaces in the Linux operating system, with Linux control groups (CGroups) assisting in resource management, such as memory, CPU, and block

I/O [34]. However, attaining the same level of isolation as VMs proves more challenging with containers.

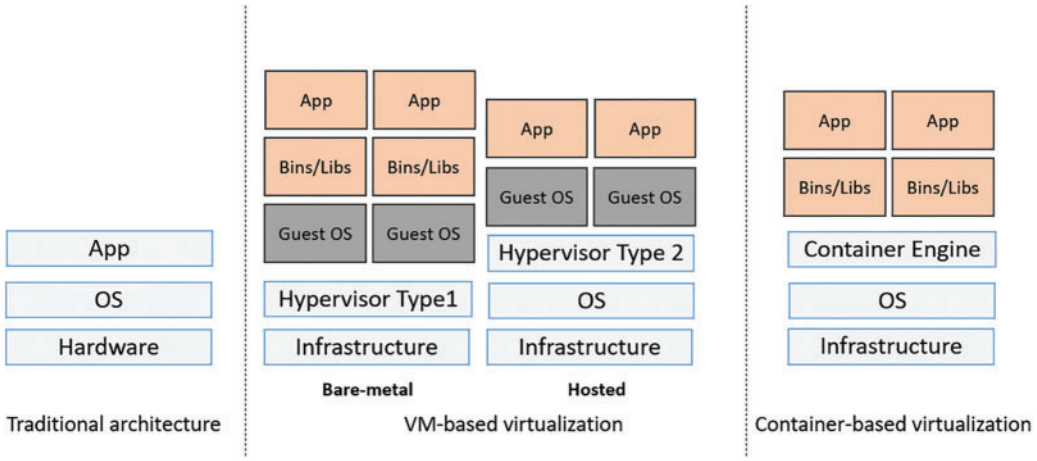

Figure 6: Virtualization architecture

Multiple studies [29–31,37–39] have conducted comparisons between the performance of VM-based and container-based virtualization methods using various metrics. For instance, in a study by [40], it is demonstrated that Docker containers outperform VMs across multiple metrics, including throughput and average response time. Another study [30] compared Docker containers with KVM virtual machines, conducting comprehensive experiments on memory, CPU, and I/O performance. The experiments encompass different parameters such as the number of VMs and containers running on machines, file copying, and web server configuration. The results indicate that Docker exhibits superior resource efficiency and faster performance compared to KVM, even on identical hardware [30]. While the CPU usage difference is not significant, KVM consumes additional memory resources, leading to a 3.6 to 4.6 times difference in memory usage for the operating system alone, even without active operations [30]. A comparison conducted in [27] between traditional architecture, virtualization, containerization, and containerization through virtualization within the context of game-based interactive simulation applications. The findings, summarized in Fig. 7, highlight some important points. As we move higher up the virtualization stack (the narrower right side in Fig. 7), there is an observed uptick in CPU and memory usage. This can be attributed to the necessary overhead required for supporting each technique and accommodating additional operating systems [29].

The primary goal of DC consolidation is to optimize resource utilization and reduce active components by strategically placing or migrating objects, enhancing energy efficiency and resource utilization in cloud infrastructures. Applications requiring computing resources, like memory and CPU, are allocated to VMs or containers, which are then placed on VMs, or PMs based on the underlying virtualization technology, with similar strategies employed for data storage placement [41]. During migration, object placement is adjusted to enhance DC configuration. Cloud DC consolidation maximizes idle resource usage through object organization, encompassing VMs, containers, software, or data. Consolidation can occur at various levels, such as VM, container, software, or data, aiming to optimize resource utilization, application response time, and overall placement performance while reducing costs and power consumption [42–47]. Hybridizations of consolidation scenarios, like container and VM consolidation, further enhance these objectives [38,47]. Fig. 8 shows examples of consolidation.

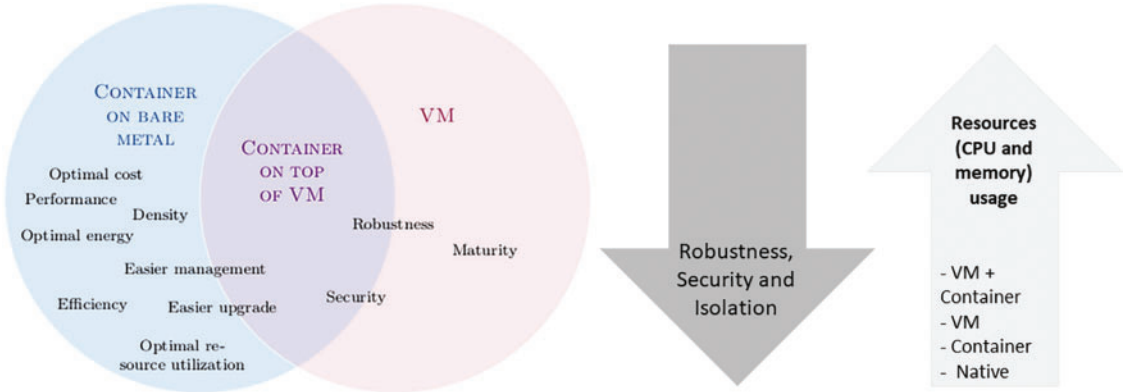

Figure 7: Main advantages of each virtualization architecture

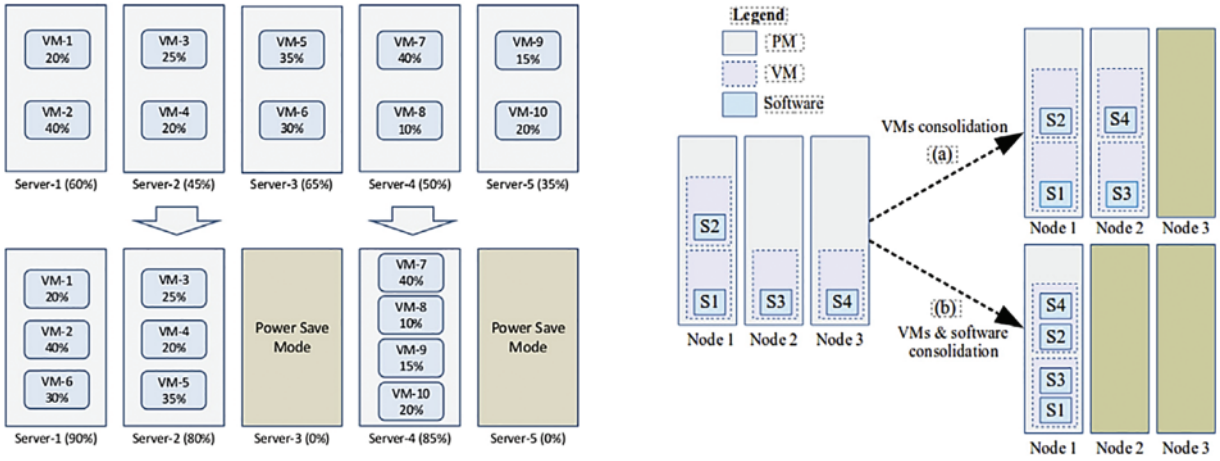

Figure 8: Examples of consolidation

As summarized in Fig. 9, consolidation in cloud data centers is an important practice that aims to optimize resource utilization, improve energy efficiency, and enhance system performance. It involves grouping and organizing various elements such as VMs, containers, software, and data within the data center [46]. There are several reasons for consolidation, including power management, load balancing, and fault tolerance. Power management focuses on minimizing energy consumption, while load balancing ensures even distribution of workloads for efficient resource usage. Fault tolerance aims to enhance system resilience and the ability to handle failures.

Communication optimization is another key aspect of consolidation, where the goal is to manage and reduce communication costs within the data center. By consolidating resources, communication overhead can be minimized, leading to improved efficiency and cost savings. Consolidation also helps in streamlining maintenance processes within the system, making it easier to manage and maintain resources. In addition to these reasons, consolidation in cloud data centers addresses various needs such as performance improvement, scalability, efficiency, reliability, and availability. These needs drive the consolidation process and contribute to enhancing the quality of service and resource utilization in the data center. Fig. 10 summarizes the main requirements for consolidation.

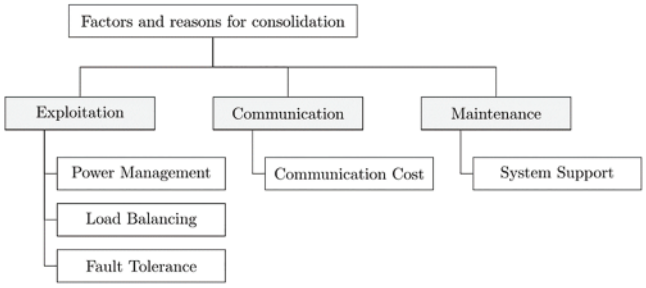

Figure 9: Main reasons for consolidation

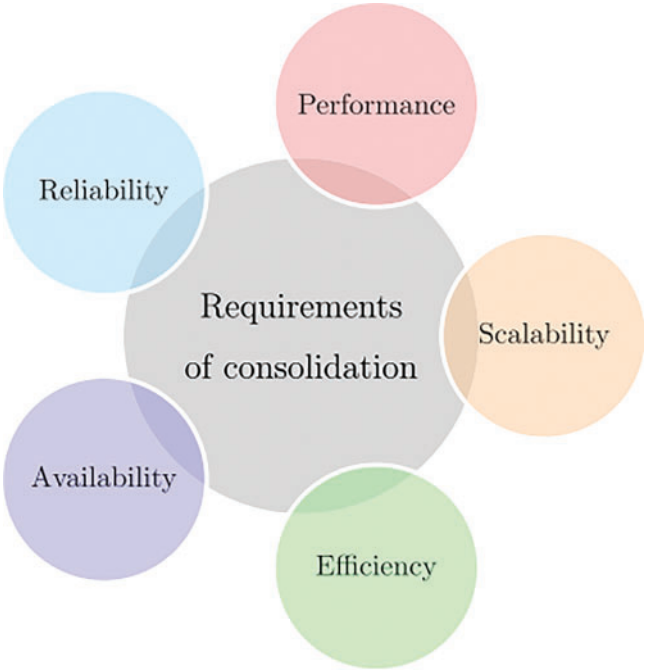

Figure 10: Requirements for consolidation

Consolidation in cloud environments brings benefits for both clients and providers, as it enables low prices, optimized quality of service, and profitability (See Fig. 11). Cloud providers aim to configure resources effectively to meet client requirements while ensuring DC utilization and VM performance. SaaS and PaaS providers minimize costs by reducing the number of VMs and may lease VM instances during peak demand to maximize profit [45–48]. Consolidation allows providers to balance financial objectives and customer satisfaction by optimizing resource utilization and reducing expenses.

1.2 The Need for a CSB Policy

As previously discussed, the CSB policy plays a crucial role in efficiently handling user requests by selecting suitable DCs. Its objective is to ensure uninterrupted processing of user requests by distributing service instances across relevant DCs in the event of a failure in any specific DC. In essence, the CSB directs user requests to the most suitable DC situated in various regions worldwide [12]. Once

the ideal DC is determined for a user's request, a specific VM is chosen by the load balancer within the selected DC to execute the request [49]. The response is then redirected back to the user, who is seeking a service with high QoS. User satisfaction is determined by the provided QoS which depends on the selected DC and its ability to provide services efficiently and effectively. Therefore, proper DC selection enhances the scalability and efficiency of cloud services, enabling priority for users' requests that require immediate execution over other demands, facilitated by appropriate scheduling algorithms. Other objectives of an efficient CSB policy include minimizing energy consumption, maximizing profit for CSPs, and meeting QoS requirements. CloudSwitch [50] and RightScale [51] are well-known CSBs that were established to offer Amazon EC2 and management platform services, respectively. More recently, Dell and VMWare have partnered to offer a brokerage infrastructure for cloud services [52].

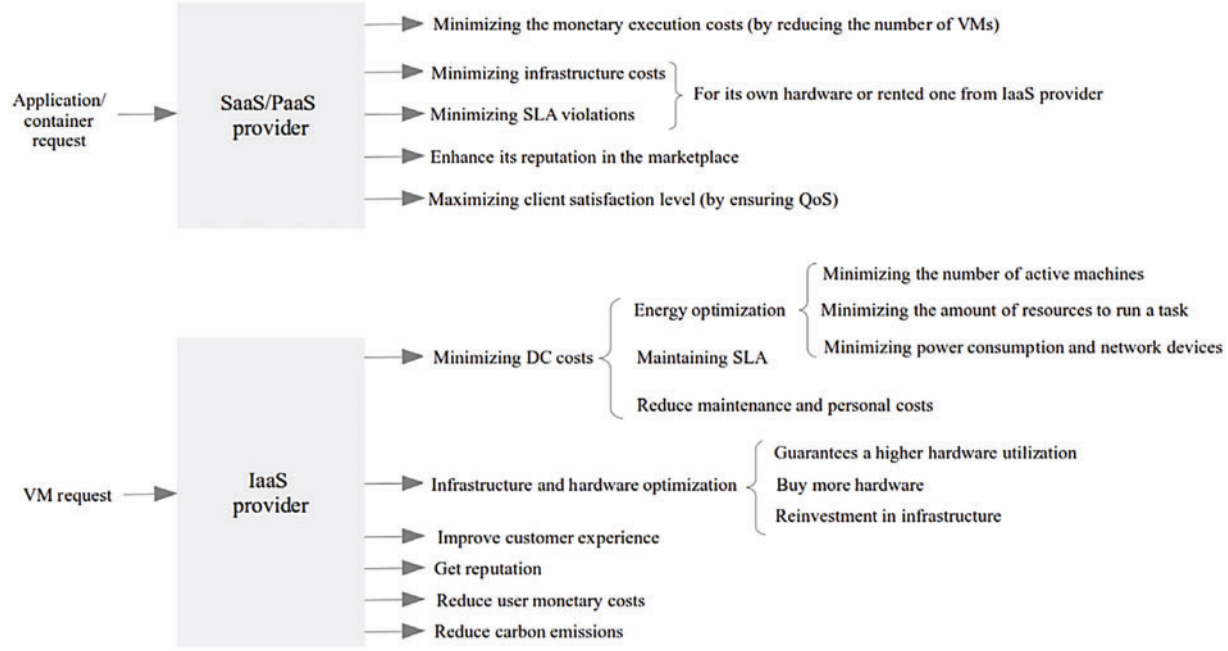

Figure 11: Main benefits of consolidation service from CSP perspectives

In a broad sense, the Cloud environment involves four essential actors that interact with each other in various ways to fulfill the primary objective of Cloud Computing. These actors are depicted in Fig. 12. The first actor is the cloud user, who possesses the capability to access and utilize services provided by a CSP. The second actor is the CSP, an organization that owns and offers services to interested users. Acting as an intermediary, the third actor is the CSB, responsible for managing the interactive relationship between the cloud user and the CSP. The CSB can either extend or directly provide services from the CSP to the user. Finally, the Cloud carrier serves as another intermediary actor tasked with transporting Cloud Computing services from CSPs to cloud users. This can be achieved through the involvement of a CSB or by directly connecting to the user via a network connection. Although the roles of the CSB and Cloud carrier share similarities, there is a crucial distinction between them: the CSB possesses the ability to modify the service before delivering it to cloud users, while the carrier does not possess this capability.

The role of a CSB is of utmost importance in directing user requests towards the most appropriate DCs situated in diverse regions across the globe [52]. Once the suitable DCs are identified, the requests

are executed by specific VMs selected by a load balancer within the chosen DC. The resulting response is then routed back to the user, who seeks a service with a high-QoS. User satisfaction is gauged based on the achieved QoS level, which is influenced by the chosen DC and its effectiveness in delivering services efficiently. The process of DC selection facilitated by the CSB policy is visualized in Fig. 13.

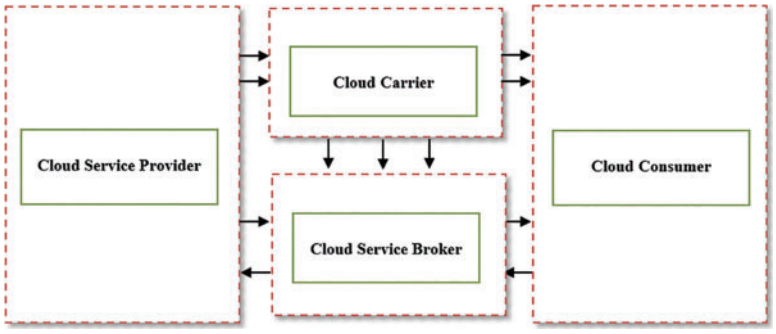

Figure 12: Main actors in the CC

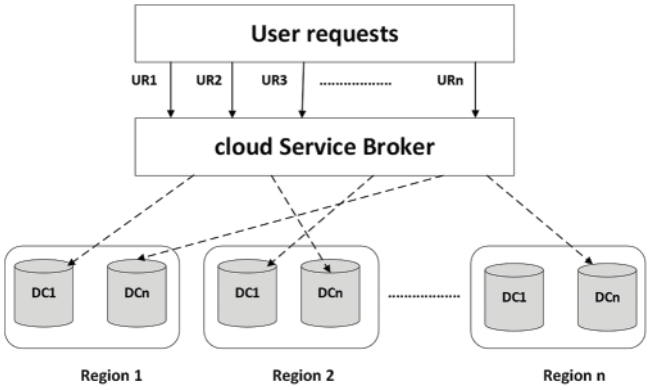

Figure 13: Selecting DC through CSB policy

As mentioned earlier, a CSB serves as an intermediary element that holds a pivotal role in facilitating the seamless integration, effective management, and seamless interaction between cloud service providers and users or organizations seeking cloud services. It acts as an essential link bridging the diverse range of cloud services furnished by different providers with the end-users who are in need of utilizing these services. A fundamental duty of the CSB lies in aiding the selection of the most fitting data center for user requests within a cloud-based environment. In the realm of CC, there exists a multitude of cloud service providers extending a variety of services encompassing IaaS, PaaS, and SaaS [53]. These providers operate data centers situated across varying geographic locations, each possessing distinct capabilities, levels of performance, and pricing structures. This intricacy can present a challenge for cloud consumers, impeding their ability to make well-informed decisions regarding the optimal data center to cater to their precise needs.

Here is an elucidation of how a CSB facilitates the selection of the most suitable DC for user requests [54]:

- Aggregation and Comparative Analysis: The CSB aggregates a wealth of information pertaining to the diverse array of cloud services and data centers proffered by different providers. It

collects data concerning factors such as geographical placement, latency, compliance adherence, Service-Level Agreements (SLAs), pricing models, and performance metrics.

- Evaluation of User Requirements: Upon receiving a cloud service request from a user or organization, the CSB conducts a comprehensive assessment of the particular stipulations and limitations associated with the request. This encompasses factors like desired performance levels, requirements for data residency, compliance prerequisites, and budgetary constraints.
- Matching and Provision of Recommendations: Drawing from the user's specific requirements and the data amassed from various cloud providers, the CSB employs algorithms and decision-making protocols to align the user's needs with the most appropriate data center alternatives. Factors considered include proximity to the user, distribution of workloads, security protocols, and the availability of resources.
- Real-Time Monitoring: Post-recommendation, the CSB continues to actively monitor the performance and operational status of the chosen data center in real-time. Should any disruptions or issues arise, the CSB can trigger automatic failover or migration to an alternative data center, thus ensuring uninterrupted service delivery.
- Facilitation of Dynamic Scaling: In instances where the demands of a user's workload undergo fluctuations, the CSB can contribute to the seamless adjustment of resources by suggesting data centers capable of accommodating heightened load or provisioning additional resources as required.
- Optimization of Costs**: The CSB is equipped to scrutinize the cost structures of diverse data centers and cloud providers, thereby orchestrating expense optimization. It factors in elements such as pricing models, costs associated with data transfers, and efficiency in resource utilization to furnish recommendations that align with cost-effective solutions.
- Enhanced Flexibility and Choice**: The Cloud Service Broker extends users a broader array of options encompassing cloud service providers and data center selections. By doing so, it mitigates the risk of vendor lock-in, empowering users to seamlessly transition between providers or data centers when the need arises.

In essence, a CSB plays a pivotal role in guiding the selection of the most fitting DC for user requests in a cloud environment. By consolidating, analyzing, and comparing data from multiple cloud providers, the CSB streamlines decision-making for cloud consumers, thus elevating the efficiency and efficacy of their utilization of cloud services.

1.3 Datacenter Selection Techniques

In the ever-evolving realm of CC, the decision regarding which DC to choose assumes a position of utmost significance. It plays a pivotal role in shaping the effectiveness, dependability, and operational efficiency of cloud-based services. With organizations progressively adopting cloud technologies to cater to a wide array of user needs, the selection of the right data center transforms into a pivotal choice, exerting direct influence over user experiences and the outcomes of operations.

Against this backdrop, the acquisition of a comprehensive comprehension of data center selection techniques becomes imperative. This discourse delves deeply into a spectrum of strategies employed for the selection of DC within the landscape of CC environments. By embarking on an exploration of these methodologies and unearthing their merits and constraints, readers stand to attain invaluable insights into the intricacies underpinning the process of making informed decisions. These decisions,

in turn, facilitate the optimization of service delivery and the judicious utilization of resources. Table 1 summarizes the most commonly used techniques for DC selection in CC [55]:

Table 1: Comparison among most commonly used techniques for DC selection in CC

Technique	Description	Advantages	Drawbacks
Proximity-driven selection	This approach involves opting for the nearest data center to the user's location, aiming to reduce latency and enhance response times.	Minimizes latency, boosts user experience, and is straightforward to implement.	Might lead to uneven workload distribution, overlooking factors like resource availability and cost.
Performance-centric selection	This technique factors in historical performance metrics of data centers, including response time, throughput, and availability, to make the optimal choice.	Enhances application performance, ensures consistent user experience, and reduces downtime risks.	May not consider sudden workload fluctuations or resource spikes, potentially resulting in resource congestion.
Cost-driven selection	This strategy prioritizes lowering operational expenses by selecting data centers with lower prices, data transfer costs, and resource charges.	Optimizes cost-efficiency, allows budget control, and suits resource-intensive workloads.	Could compromise performance or reliability for cost savings, and comparing intricate cost structures can be challenging.
Resource-based selection	This method evaluates data centers based on available resources like computing power, memory, storage, and network bandwidth.	Guarantees resource availability, maximizes resource utilization, avoids contentions, and supports scalability.	Might overlook other aspects such as performance or location, potentially impacting user satisfaction.
Hybrid approach	This technique integrates multiple criteria—like proximity, performance, cost, and resource availability—to make well-balanced decisions.	Offers a comprehensive perspective, considers diverse factors, and adapts to various workloads.	Complex decision-making can lead to higher computational demands, and striking a balance among multiple criteria requires careful tuning.

(Continued)

Table 1 (continued)

Technique	Description	Advantages	Drawbacks
Dynamic selection with machine learning	This advanced method employs machine learning algorithms to predict data center performance, leveraging historical data and current conditions.	Adapts to evolving workload patterns, enhances accuracy over time, and handles intricate decision scenarios.	Demands substantial historical data for training, initial setup can be resource-intensive, and model precision might be impacted by abrupt changes.

1.4 Research Motivations

In a cloud computing environment, the process of choosing a data center involves allocating user requests effectively among the available data centers to optimize resource utilization and fulfill the needs of both users and CSPs. In this review, several CSB policies are evaluated. The CSB aims to route user requests to different DCs efficiently to ensure efficient resource utilization and high levels of QoS, so it is important to explore issues and challenges in DC selection and work on addressing them. As cloud services become increasingly popular, data storage is rapidly expanding to meet the high volume of usage. CSPs are adding more DCs with large storage and processing capabilities to serve the large volume of users' requests from around the world, and it is important to recognize the role of the CSB in this context. Therefore, relevant articles have been analyzed and summarized systematically to highlight issues and directions for future research.

1.5 Article Organization

This article provides a review of CSB policies and is organized as follows: In Section 2, we compare our survey with earlier surveys of CSB policies and DC selection techniques in the CC environment. Section 3 explains the survey method used to gather relevant research articles, including search criteria, sources of articles, and quality evaluation. In Section 4, we summarize the problem of DC selection in the cloud. Section 5 discusses the importance of the CSB. In Section 6, we review various CSB policies. Section 7 summarizes the research issues and challenges faced by existing CSB policies. Section 8 summarizes the research trends. Finally, in Section 9, we offer concluding remarks.

2 Related Surveys

In this section, we compare the review presented in this article with existing reviews that have surveyed similar topics. The comparison is based on the topics covered by each review, which include: an overview of CC, CC deployment models, CC service models, an overview of CSB, categories of CSB, CSB policies, research issues and directions, and research trends.

A survey of cloud brokerage techniques was proposed in [56]. It describes the concept of a third-party entity between the CC user and CSP and defines the role of the cloud broker. A brief survey of CSB technologies was conducted in [54], which summarizes future opportunities in this field. In [57], the main features of CSBs were compared, including prioritizing CSPs to serve users. The article covers the types of CSBs and their roles, as well as existing frameworks, impacts, and advantages of

CSBs. A brief survey of CC was conducted in [1], which covers CC architecture, characteristics, and deployment models. However, a more detailed survey of existing CSB algorithms is found in [8].

Additionally, a survey of different CSB selection techniques was conducted [58,59]. This survey defined CC, covered CSB selection algorithms and quantitatively compared them, and briefly discussed the CloudAnalyst simulator. The CC services and brokerage were surveyed and compared with different solutions in [60], while a comparison between traditional CSPs and CSBs was conducted in [61]. This survey identified certain parameters that should be considered when choosing between traditional CSPs and CSBs. Another survey aimed to gather and investigate all previous research on cloud service composition was conducted in [62]. Table 1 provides a comparison amongst prior related reviews based on topics covered. Table 2 shows that there is no comprehensive survey of CSB policies that covers the importance of these policies, their issues, research directions, and trends. This demonstrates that the proposed survey is more comprehensive than other prior related surveys in this field.

Table 2: Comparison with related surveys

Ref.	CC overview	Deployment models	Service models	CSB overview	CSB categories	CSBs policies	Research challenges	Future directions
[8]	Yes	Yes	Yes	Yes	No	Yes	No	No
[62]	Yes	No	Yes	Yes	No	No	No	No
[57]	Yes	No	No	Yes	No	No	No	No
[58]	Yes	No	No	Yes	No	No	No	No
[1]	Yes	Yes	Yes	No	No	No	No	No
[59]	No	No	No	Yes	No	Yes	No	No
[60]	Yes	Yes	Yes	Yes	No	Yes	No	No
[61]	Yes	Yes	Yes	Yes	No	No	No	No
[62]	Yes	Yes	Yes	Yes	No	No	No	No
Proposed	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

3 Survey Technique

3.1 Mapping Research Questions to Objectives

Different methodological survey standards, such as those found in [63–67], have been followed in this article. To focus on research related to CSB policies and DC selection in the CC environment, the survey method involves identifying relevant references, extracting the results, and analyzing the challenges. The sources of related references are presented in this section, along with the selection criteria for references, the evaluation of quality, and the evaluation results. Table 3 lists the research questions that were used to explore CSB policies and determine the current challenges related to CSB policies and DC selection in the CC environment.

3.2 Quality Assessment

A quality evaluation process was applied to the retrieved articles based on inclusion and exclusion criteria [64,65] listed in Table 4. A total of 781 articles were retrieved based on search queries containing the keywords. Initially, all articles with irrelevant titles were excluded, and the remaining articles were fully reviewed. Only 119 articles were analyzed in this survey.

Table 3: Mapping between research questions and objectives

Research question	Objective (s)
1. What is the importance of CSB policies and DCs selection in the CC environment? 2. Why are CSB policies required in the CC environment? 3. Do the current CSB policies fulfill DCs selection essentials QoS requirements for the Cloud's users? 4. Do the current CSB policies fulfill DCs selection essentials interests for the CSPs? 5. What is the current status of DCs selection in the CC environment? 6. What are the new techniques that are required to improve DCs selection process in the CC environment? 7. How does one minimize response time, processing time, and availability within the lowest cost? 8. How does one maximize CSPs' profit within the lowest power consumption? 9. Which QoS requirements of CSB policies are most significant for the performance of the CCenvironment?	• To identify several CSB policy references published over time in different resources. • To analyze and evaluate different CSB policies based on evaluation metrics. • To identify issues and challenges in current CSB policies to ensure users' QoS requirements and CSPs' requirements.
10. How does one develop CSB policies that fulfill the Cloud's user QoS needs? 11. How does one develop CSB policies that fulfill the CSPs' interests and needs? 12. How does one evaluate the current CSB policies through available simulation tools? 13. How does one develop new CSB policies through available simulation tools?	• To identify research articles from different CSB policy categories to reveal research gaps. • To define DCs selection criteria and to develop CSB policies that address a huge number of users' requests efficiently. • To identify future research directions and trends.

4 Cloud Datacenter Selection Problem

According to a Cisco survey from 2018, Cloud DC traffic was 6.8 Zettabytes in 2016 and is expected to reach 20.6 Zettabytes in 2021, representing a growth rate of 27% or a threefold increase from 2016 to 2021 [68]. This information is depicted in Fig. 14.

Table 4: Inclusion and exclusion criteria

Inclusion	Clearly describes CSB policies and DCs selection.
	Presents emerging ideas.
	Develops or modifies CSB policies in the CC environment.
	Resource scheduling perspective.
	Published in CC area.
	Written in the English language.
	Published in reputable resources.
	Written by academic researchers.
Exclusion	Does not focus on CSB policies and DCs selection in the CC environment.
	Contains common issues and references.
	Is it not a full research paper?
	Focuses on service broker in other paradigms (i.e., outside CC environment).

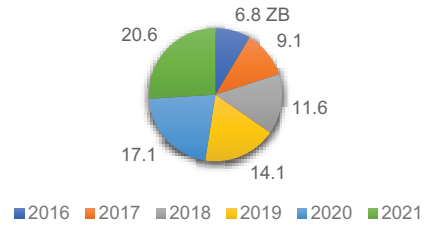

Figure 14: DCs traffics (in Zettabytes) from 2015–2020[1]

As mentioned earlier, CC environments consist of multiple DCs that interact with each other in an ad-hoc manner to fulfill user requests. The QoS is employed as a metric to gauge user satisfaction with the provided services. In order to direct user requests to the most suitable DC, the selection of Cloud DCs is determined by optimizing various objectives, including processing time, response time, availability, total cost, power consumption, and profit, within a specified timeframe.

In this context, consider a scenario where a set of user requests, denoted as N users' requests $= \{UR_1, UR_2, \ldots, UR_n\}$, needs to be directed to a corresponding set of available cloud data centers, represented as $DC = \{DC_1, DC_2, \ldots, DC_n\}$. The goal is to establish an optimal configuration, denoted as x ($DC \times UR \rightarrow x$), where the matching of user requests to data centers is determined, while concurrently optimizing specified objectives defined as $O = \{O_1, O_2, \ldots, O_n\}$. This optimization process is carried out within a predefined time frame. In essence, the aim is to efficiently allocate user requests to suitable cloud data centers in a way that maximizes the achievement of predefined objectives. The term "fitness of x" alludes to the overall suitability or appropriateness of the assignment configuration, with the primary objective being the optimization of the defined objectives. These objectives can encompass a variety of factors such as resource utilization, latency, cost, energy efficiency, or any other relevant performance metrics.

[1] ZB = Zettabyte

The process involves considering various potential combinations of user requests and available data centers, seeking the arrangement that best aligns with the specified objectives. This optimization task is subject to a time constraint, implying that the desired configuration must be achieved within a specific timeframe to ensure timely and effective service delivery. Overall, the objective of this scenario is to create an optimized mapping of user requests to available cloud data centers, taking into account specific goals or objectives, while adhering to a defined time limit. This process aims to enhance the overall performance and efficiency of the cloud computing environment in delivering services to users.

This process, known as DC selection in the CC environment, is depicted in Fig. 15. It is important to note that the objectives often include both the QoS concerns of users and the interests of service providers. The CSB is responsible for ensuring that these objectives are met. In some cases, a single DC may be able to serve multiple user requests.

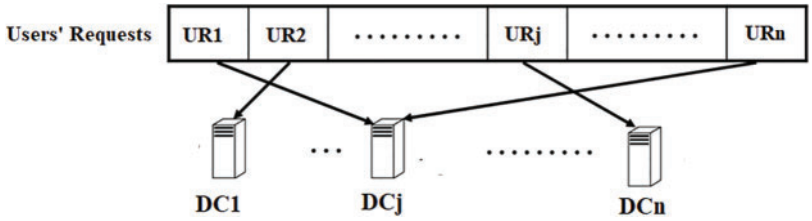

Figure 15: Cloud DC selection problem: Routing N users' requests to N DCs

5 Cloud Service Broker

In a CC environment, there are three main actors: Cloud providers, who offer Cloud resources and services to users; Cloud users, who utilize these resources; and a CSB, which acts as an intermediary [69–77] by bundling and managing the connected parties in CC environment [8], simplifying the execution of user requests, and selecting the optimal deal between the Cloud user and CSPs with the maximum profit [70,71]. According to [78], the CSB may take on various roles, such as service intermediator, service aggregator, or service arbitrage, and is responsible for meeting the QoS requirements of SaaS [8]. It acts as a facilitator for the CC's services to meet the needs of users. The proper CSBs can be identified through Cloud Exchange, and they may work with CC coordinators to allocate resources to meet these requirements.

The CSB can be viewed from two perspectives: as a scheduler and as a company. With the increasing number of private and public CSPs entering the CC market and offering a range of cloud services with different characteristics, it can be difficult for users to choose the best option for their needs. As a scheduler, the CSB helps users select the most appropriate CSPs and assists CSPs with efficient resource allocation by converting the heterogeneous CC into a commodity-like service [79]. An efficient CSB policy is necessary to optimize the selection of DCs to minimize the total cost of VM deployment and to ensure a satisfactory level of QoS [80–82]. In some cases, the CSB may function as a third-party company offering a variety of CC services, but unlike CSPs, it does not have its private cloud resources and instead rents them from CSPs. Regardless of its nature, the CSB should allocate user requests to the most suitable DC based on the users' QoS requirements and the interests of CSPs [83]. When a user requests a CC service, the request is routed to the CSB, which uses an Intelligent Discovery Component to route it to the most relevant CC service [84]. The CSB is responsible for controlling the routing between the DC and the Cloud user and implementing a selection policy at the DC level [10].

6 Review of Existing CSB Policies

To ensure optimal performance within the CC environment, it is crucial to assign new user requests to the most suitable DC based on factors such as response time, processing time, and total cost. This responsibility lies with the CSB, making the development of effective CSB policies a significant area of research. Improper DC selection can result in overload, negatively impacting CC performance, particularly when there is a surge in user requests. Such situations can lead to a decline in quality of service, user satisfaction, and even rejection of new user requests due to overwhelming DC capacity [85,86]. Hence, assigning new user requests to the most appropriate DC with the lowest response time, processing time, and total cost is essential for enhanced CC performance. Researchers have focused on proposing CSB policies to address this issue. These policies can be categorized primarily into two types, as depicted in Fig. 16: (i) CSB policies based on service providers' interests, and (ii) CSB policies based on users' concerns regarding QoS. The existing literature on CSB policies is extensively surveyed and reviewed in the following sections.

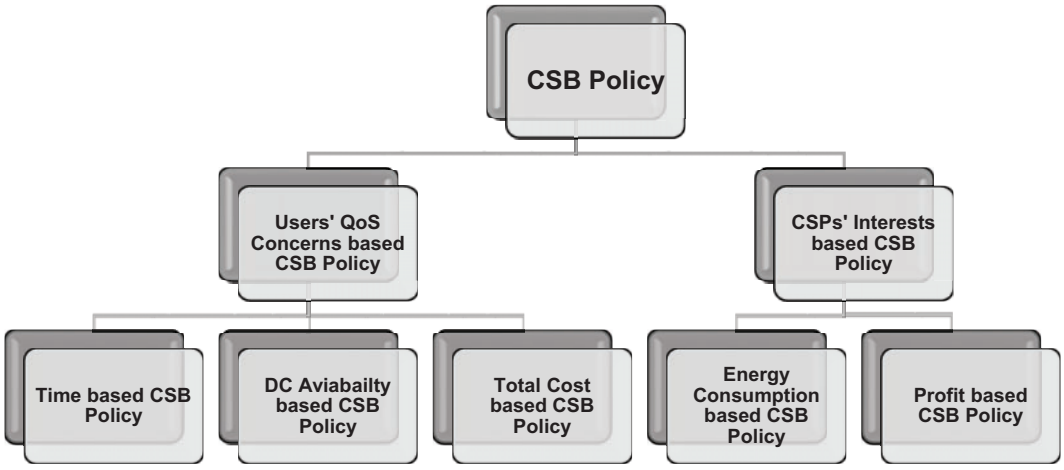

Figure 16: Taxonomy of CSB policies used for DC selection

6.1 CSB Policies Based on the Interests of Service Providers

Several existing CSB policies have been proposed based on the interests of CSPs (i.e., energy consumption and profit) to allocate resources in the CC environment efficiently. The goal is to ensure that CSPs can effectively meet the demands of their users while also maximizing their profits and minimizing energy consumption.

6.1.1 Energy Consumption-Based CSB Policy

There are several studies in the literature that propose CSB policies to minimize energy consumption. However, only a few of these studies consider the fact that the CSB itself can have a significant impact on energy consumption. One such CSB policy, proposed in [87], measures the energy consumption required to execute user requests, taking into account the processing time of the requests and the power consumed by the VMs that processed them. This policy routes user requests to DCs that minimize energy consumption. The results of this approach show that it requires less power compared to other methods [87]. In addition, Beloglazov [88] proposed a CSB policy to optimize energy consumption while minimizing the dependence of energy management on the CSB. Another

CSB policy, proposed in [89], ranks DCs based on their energy and performance features. This policy has two main steps: (i) an introduction step, which uses a sophisticated algorithm to prioritize DCs based on energy consumption and performance features, and (ii) a running step, which combines another feature of the DC called low time. At the end of this step, the overall DC parameter is calculated and the DCs are ranked based on this value. User requests are then routed to the DC with the highest rank. If the DC is at peak workload, the user request is routed to the next DC, and so on. The simulation results of this policy show that the power consumption required to execute requests is significantly reduced.

Several other CSB policies were proposed to optimize resource utilization and minimize energy consumption. For example, Cong et al. [90] introduced a new resource utilization technique that efficiently allocates DCs based on optimizing the number of active DCs. A heuristics approach proposed in [91] combines the bin-packing technique to address the problem of scheduling independent requests. The Cloud-SEnergy technique proposed in [92] aims to select the most energy-efficient DC in the CC environment, using a bin-packing method to identify the most efficient service plans. The results of this technique show that it outperforms other techniques, such as DC-Cloud and Base Cloud, in terms of energy consumption ratio, the number of examined services, and run-time. In [93], a different CSB policy is proposed, which takes into account both energy consumption and profit considerations. This policy utilizes a resource allocation technique known as MCD (Minimum Cost Dominating set) to minimize the energy cost for CSPs. The authors demonstrated that the problem is NP-hard, and to address it, they employed an approximation algorithm based on LP (Linear Programming) and a greedy algorithm to ensure consistent performance. The results of this policy exhibit superior performance in terms of energy consumption, resource utilization, and profit maximization. However, it is important to note that this work has certain limitations. Firstly, it does not consider changes in user demand over time. Additionally, it fails to account for the fact that each CSP has its own heterogeneous data centers (DCs) with varying capabilities and prices.

6.1.2 Profit-Based CSB Policy

In the CC environment, organizations need to make a profit to maintain normal operations. Therefore, increasing the profit of CSPs has become an important issue and has been investigated from various perspectives in different studies. Users' requests may have various requirements, including security and privacy constraints, cost, and more. It is the responsibility of the CSB to improve the CSP's profit by routing users' requests (i.e., service requests) to the most appropriate CC resources (i.e., DCs) based on the needs of the users and/or CSPs [94]. A hybrid CSB policy was proposed in [95] to route a Cloud's service to specific DCs or public CC resources. The profit of the CSB depends mainly on two main factors: (i) the cost of the brokerage service, and (ii) the cost of resource utilization. If the service is executed by public DCs, the profit of the CSP is the fee of resource provisioning, and the profit of the CSB is the fee of the brokering service. Otherwise, the CSB profit is the sum of both. To increase the total profit of the CSB, a greedy technique is used to maximize the in-house execution of all user requests (i.e., service requests). However, this greedy technique is not suitable when there is a limited number of resources and specific QoS needs, so using a heuristic CSB algorithm may address this issue.

Virtual Cloud Brokers (VCBs) were proposed in [96] to achieve a high level of profit by offering multiple instances of VMs from different CSPs at a lower price than traditional CSPs for a longer time (i.e., VM outsourcing). This allows for efficient management of available resources to execute user requests. To address the issue of resource allocation, a profit maximization model was proposed concerning resource constraints and fast optimization methods. In addition, a CSB pricing policy was

proposed in [96] to maximize the profit of the CSB. The CSB first sets the VM selling price based on the VM demand. If the VM demand is increased, the price is increased to minimize the request delay. The performance of this technique is significantly improved when considering the loss caused by missing VMs. Another CSB policy, proposed in [97], aims to minimize the cost for CC users. The CSB rents multiple DCs from CSPs at a specific price and provides them to CC users on demand at a price that is lower than what the CSPs offer to the same users [97]. The CSB also uses a shorter billing method compared to CSPs. By using this policy, the CSB can minimize the cost for CC users. Furthermore, specific CSB configurations and pricing methods are defined in [97] to ensure profit maximization while reducing costs for CC users. The profit of a CSB is directly influenced by various factors, including the number of user requests, the price of VMs, the scale of the CSB, and more.

A new CSB framework was implemented in [98] to provide an efficient brokering policy that ensures finding the most appropriate DC for executing user requests with high performance and profit. This technique works based on the Round-Robin algorithm and follows the following steps: (i) formulating SLA requirements, (ii) mapping SLA requirements of users to CC resources, and (iii) deploying resource allocation. As a result, the CSP with the lowest cost is selected to increase overall profit by executing more requests [98]. The results of this policy show that it increases the execution rate of user requests at a low cost, which can be beneficial for both users and brokers. Many previous works focus on enhancing the profit of the CSB and/or CSP based on a specific setting. However, a major challenge for a CSB to maximize profit is determining the number of DCs needed to execute user requests and how to provide an efficient price for each DC.

Since various factors can impact the profitability of the CSB and CSP, it is important to consider a comprehensive and accurate analysis of these parameters when configuring and pricing the CSB. Many CSB policies based on energy consumption can improve energy efficiency, but they may serve a lower number of user requests and do not consider users' QoS needs. Additionally, CSB policies based on profit may suffer from the same challenges, as well as potentially overwhelming the DCs significantly.

6.2 CSB Policies Based on Users' QoS Concerns

Since the CSB operates in a dynamic paradigm, where the QoS requirements negotiated with CSPs can fluctuate, it requires more efficient techniques to enhance the QoS level provided to Cloud users. To this end, various existing CSB policies have been proposed in the literature to reduce the processing time, increase the response time, enhance the DC availability, and decrease the total cost. Selecting the optimal DC based on QoS and cost considerations is a challenge when executing user requests in the cloud. Several optimized, modified, or hybridized CSB policies have been proposed to select the optimal DC based on user QoS interests. These policies are categorized as follows.

6.2.1 Time-Based CSB Policy

A new DC selection method based on the user's priorities is presented for use in a federated CC environment in [99]. It works based on the user's priorities and uses a matrix to select the most appropriate DC. When a user's request is initiated, if there is no available DC to execute the request, the request is placed in a waiting queue [100]. A central entity is responsible for creating VMs to execute the user request after reading the waiting queue and selecting the optimal DC to execute the request. One challenge of this proposed method is that it is a centralized method, which means, it shifts the load from an underutilized physical machine to another physical machine and shuts off the less used physical machine to minimize power consumption. The simulation results show that this CSB policy

achieves high levels of performance and acceptable response time at a low cost. Another DC selection technique based on the k-nearest-neighbor method is proposed in [101]. The DCs are represented in n-dimensional space with defined features, and a tuple of identical features is defined as thresholds for these attributes. When a new user request is received, it is attached to the threshold tuple. If there is only one DC in the closest region, the DC in this region will be chosen to serve the user request. Otherwise, if there are multiple DCs in the closest region, one of them will be selected randomly to execute the request. The distance between users' requests and DCs must be calculated first, and then the DC with the minimum distance would be selected. The simulation results show that this proposed DC selection technique outperforms a random-based policy in terms of average response and processing times.

Three CSB policies were compared in [101]: The Closest DC Policy (CDCP), the Optimized Response Time Policy, and the Dynamically Reconfigurable CSB with Load Balancing (DRCSB). In CDCP, the DC with the least proximity to the user is selected. When multiple DCs have the same proximity, a DC is selected randomly to balance the load. The selection process depends on the closest region listed in the proximity list, and DCs within the same region are selected randomly to execute users' requests [102]. However, this technique has several challenges: (i) DCs are selected randomly if there are multiple DCs in the same region, (ii) it may select a DC with high cost, and (iii) the results may vary even with the same configurations, making it difficult for developers to interpret or use the results [103]. The DRCSB is an extension of the CDCP, where the number of VMs is increased or decreased based on the number of users' requests [100]. This policy works based on DC load and uses dynamic load balancing to ensure the efficient execution of users' requests. The dynamic nature of configuring the DCs also improves the overall performance of the CC environment. On the other hand, the ORTP focuses on minimizing response time by calculating and optimizing the expected response time for each DC, enabling DCs to process user requests with minimal response and processing time [73,104]. When employing the ORTP policy, the CSB identifies the nearest DC based on latency, similar to the CDCP. Subsequently, the response time is computed for each DC. If the estimated response time is the lowest for the closest DC, it is selected. However, if the estimated response time is not the lowest, there is a 50:50 chance of either selecting the closest DC or the DC with the lowest response time [105]. The ORTP policy outperforms the CDCP as it takes into account factors such as latency, DC load, and bandwidth, resulting in improved performance.

The DRCSB policy for deploying applications in a CC environment scales the number of VMs based on the current load and the minimum processing time ever achieved [13]. However, it tends to have worse performance in terms of response time compared to other policies such as ORTP [106]. On the other hand, a Round-Robin algorithm is commonly used for job scheduling and resource allocation in CC environments, distributing the load evenly among available resources [107]. The Round-Robin algorithm works by randomly selecting a VM and allocating the first user request to it. The next user request is then circularly allocated to the next VM. Once a user request has been successfully allocated to a specific VM, the selected VM is moved to the end of the list [108]. This process is repeated until all user requests have been allocated to a VM. A new Round-Robin-based technique for selecting a DC was proposed using the CloudAnalyst simulator and showed improvements in response time compared to other policies [109]. However, this technique can also have challenges such as power consumption and underutilization issues.

A static CSB policy was proposed to improve the load time and request time of a DC and reduce the cost of VMs and data delivery [110]. However, this policy also increases the overall cost in a CC environment. To address this issue, a Round-Robin-based CSB policy was used to distribute the workload among multiple available DCs within the same region [109]. Additionally, an enhanced Service Proximity CSB policy was proposed in [111] to avoid overloading the closest DC and violating

the service-level agreement by rerouting users' requests (partially or completely) to the closest DCs. This improves the response time by reducing the risk of service-level agreement violations. The experimental results showed that this policy improved the response time of the Service Proximity Service Broker by 17% [111].

6.2.2 Total Cost Based-CSB Policy

A new cost-effective DC selection policy was proposed in [112] to select the DC with the lowest cost when all DCs are busy. However, this policy does not consider the change in workload cost or electricity prices, which can lead to routing many workloads to a DC with high electricity prices and increase the overall operational cost. This policy also does not optimize the response time or workload and does not take the user request size into account. Improved CSB policies proposed in [112,113] may suffer from underutilization of resources because they always select the same DC, leaving other DCs never being selected. A new CSB adapted from a service proximity-based routing algorithm was proposed in [114] that uses cost-efficient selection to choose the most appropriate DC in terms of cost but does not consider performance or availability parameters. It significantly improves the total cost. Additionally, a cost-effective DC selection technique for a federated CC environment was proposed in [114], in which the DC is selected based on a matrix of values containing information about the required cost for each DC to execute the user's requests. The findings show that this policy reduces the total cost and improves performance.

In [113], two DC selection algorithms were proposed and compared to a service proximity-based DC selection algorithm. The results showed that the overall cost of the first policy was better than the second one and the service proximity-based policy, but the service proximity-based policy had the best performance in terms of processing time. Specifically, the first policy had the largest processing time compared to the service proximity-based policy and the second policy but cost less than the other two policies. Therefore, the first policy provides cost-effective DC selection, but has a higher processing time than the closest DC selection policy, while the second policy has a lower response time compared to the first policy.

In [8], a new CSB policy was proposed based on two main parameters: cost and network delay. This policy combines these two parameters to provide a lower cost with an acceptable response time. Comparing the existing DC selection algorithms with the proposed policy in two different scenarios showed that the proposed policy provides the best response time with the lowest cost. Specifically, the results showed that the response time was improved by 8%, the total cost was improved by 30%, and the processing time was improved by 10% compared to other existing policies. Additionally, a static CSB policy was proposed in [115] to reduce the DC-DC transmission cost, the cost of VMs, and the request time. It combines the benefits of a weighted Round-Robin-based CSB policy and a service proximity CSB policy. In the weighted Round-Robin policy, the DC selection is based on the processing space of each DC, while in the service proximity service broker policy, the nearest region is selected based on the maximum possible bandwidth from the user base to the DC with the minimum network delay. This proposed policy reduces the number of links and decreases the average request service time. The simulation results showed that the proposed static CSB has a lower total cost compared to the service proximity service broker policy and the weighted Round-Robin CSB policy. However, the main drawback of this policy is that it needs to further improve the average time required by the DC to respond to users' requests (i.e., response time).

In [116], a load-aware brokering policy was proposed to sort user requests into two categories: static and dynamic. Static policies are not typically suitable for a CC environment due to the frequent

changes in user needs, while dynamic policies are better suited for CC environments as they can effectively manage changes in workloads over time. The proposed policy distributes all users' requests among all available DCs and uses a load-balancing technique to select the appropriate VM. Simulation results showed that this policy reduces the total cost and processing time but increases operational costs. In [117], another static DC selection technique was proposed that selects the DC with the lowest cost, only considering the cost of VMs. This technique addresses the issue of random DC selection by reducing the cost but increasing the overall processing time. If the two most cost-effective DCs are selected, it reduces the processing time but increases the cost. Another static DC selection solution was proposed in [118] that performs DC selection based on Priority-based Round-Robin (RR). It arranges the DCs based on their processing time and assigns user requests evenly between all the DCs within a region. This approach improves resource usage and total cost.

A dynamic CSB policy was proposed in [119] that selects the DC based on two different scenarios: (i) different DCs may have the same hardware configurations but different numbers of VMs, allowing a balanced weight to be assigned to the DC based on the number of VMs included, significantly reducing the processing time, and (ii) the selection of the DC is based on two matrices: (a) performance *vs*. availability and (b) cost *vs*. location. Here, performance refers to the number of tasks performed per time unit, availability refers to the number of days available for adding new VMs in a year, cost refers to the cost per VM, and distance refers to network latency. The values of the matrices are calculated for each region and a list of DCs is created for each matrix. The DC is then selected from the intersected list (i.e., the intersection between the two lists) based on the value of the matrix. This solution provides good results in terms of the total cost.

In [72], Kapgate combined the advantages of the weighted Round-Robin CSB policy [90] and the Service Proximity service broker policy [95] to propose a new CSB policy. First, the nearest region is selected and the number of DCs in that region is calculated. If only one DC is selected, the request is sent to that DC and the result is calculated directly. However, if multiple DCs are selected, the Kapgate algorithm calculates the DC processing capacity and the processing requirements for the incoming user requests. It then calculates the number of times the DC is selected, and the results are generated. The proposed policy improves the number of connections and the average time of DC requests. Experiments show that the proposed policy has a lower cost than the weighted Round-Robin policy and the Service Proximity service broker policy. In [86], a new CSB policy based on the Differential Evolution algorithm was proposed. It uses a search method to find the optimal candidate (i.e., solution) from a solution space and the Differential Evolution algorithm to enhance performance by minimizing the value of the selected optimization parameters (i.e., total cost). The evaluation results showed that this policy outperformed some existing policies, performing better than the closest DC policy and the optimized response time policy in terms of total cost and outperforming the weight-based DC selection policy in terms of processing time. Additionally, in [19], a cost-effective CSB policy based on DC efficiency was proposed, selecting the DC with the best efficiency ratio (i.e., DC efficiency over cost). While this policy is logically valid, it was not implemented and compared to other existing CSB policies.

6.2.3 DC Availability-Based CSB Policy

In [85], a DC availability based CSB policy, called a variable CSB policy, was introduced. This technique is a heuristic-based approach that aims to reduce response time by considering the user's request size, available bandwidth, and delay. It reduces the overloading of DCs by transferring the user request to a DC with high availability and modifies the service proximity CSB policy by considering the available bandwidth rather than the delay factor to improve response time based on the user's

request size. The simulation results showed that the proposed CSB policy significantly improves DC availability, enhancing processing time and response time. However, the main challenges of this algorithm are: (i) ignoring power consumption and (ii) ignoring the overall cost, which may be preferred and a critical factor in certain cases over performance and speed. Similarly, in [18], a Genetic algorithm was used to address the multi-cloud CSB problem and increase DC (i.e., service) availability.

A new CSB policy was introduced in [120] to find available DCs for the Cloud's users. It uses two distinct policies simultaneously with a special algorithm for each, each algorithm selects the available DC based on availability and total cost. The regional list is a list of regions in which DCs are found, and the DC list contains the cost and load for current and expected resource allocation. The proposed policy continually selects the DC with the highest availability and lowest cost and updates the list if there is an available DC with a lower cost. The results showed that the proposed policy increased DC availability, and decreased processing time, response time, and total cost. Additionally, in [121], a Round-Robin algorithm was used to distribute a load of users' requests between different DCs within the same cloud region, resulting in efficient resource utilization and availability.

In [122], the authors addressed the problem of inefficient application deployment on fog computing infrastructure, which results in resource and bandwidth wastage, increased power consumption, and degraded QoS. To tackle this issue and enhance application reliability, they propose a multi-objective optimization algorithm named MOCSA. The objective of MOCSA is to minimize both power consumption and total latency between application components. Through simulations, the results demonstrate that the proposed MOCSA algorithm outperforms several existing algorithms in terms of average overall latency and average total power consumption. This highlights its effectiveness in improving performance and addressing the challenges associated with application deployment on fog computing infrastructure. Further, in [123], the authors addressed the challenge of virtual machine placement, aiming to strike a balance between conflicting objectives of stakeholders, users, and providers. This involves optimizing power consumption and resource wastage from the providers' perspective while enhancing Quality of Service (QoS) and network bandwidth utilization from the users' perspective. To tackle this multi-objective problem, the authors propose a hybrid multi-objective genetic-based optimization solution. Through simulations, the results demonstrate that the proposed algorithm outperforms existing state-of-the-art algorithms in terms of power consumption, resource wastage, network data transfer rate, and the number of active servers. Additionally, the algorithm exhibits promising scalability potential in larger search spaces.

In [124], a solution was presented for addressing the scientific workflow scheduling problem in heterogeneous cloud data centers. The scheduling problem is formulated as a bi-objective optimization task with the objectives of minimizing the makespan (achieving quick response time) and maximizing system reliability. To enhance the reliability of the system, a centralized log and a scheduling failure factor (SFF) were introduced. The problem was then formulated as a bi-objective optimization problem and solved using a proposed hybrid bi-objective discrete cuckoo search algorithm (HDCSA). The performance of HDCSA was evaluated in various scenarios, demonstrating its superiority over existing methods in terms of makespan, SFF, speedup, efficiency, and system reliability. On average, HDCSA achieved improvements of 22.11%, 12.97%, 11.81%, 12.18%, and 12.42% in these respective metrics.

A comprehensive review of existing CSB policies was presented in the previous sections, with a summary of their strengths and weaknesses shown in Table 5. Many of the existing CSB policies have common issues, such as under-utilization of resources and power consumption, and do not consider the size of the next user request when making DC selection assumptions, which may lead

to an inaccurate selection. Additionally, most existing CSB policies only consider a single objective when selecting DCs, meaning that other objectives may not always be optimal, resulting in a trade-off between objectives. Currently, there are no CSB policies that consider all objectives when selecting a DC, which may be acceptable to Cloud users but not to CSPs and vice versa. Therefore, the following section addresses research issues that deserve further attention to address these challenges.

Table 5: Summary of the existing CSB policies

Category	Criteria	Strengths	Weaknesses/Limitations
CSPs' interests based CSB Policy	Energy consumption based CSB policy	Energy consumption has been reduced.	The majority of CSB policies do not consider the users' QoS needs. The number of served users' requests needs more enhancement.
	Profit-based CSB policy	CSP's profit has significantly increased.	The majority of CSB policies do not consider the users' QoS needs. The number of served users' requests needs more enhancement. DCs are overloaded significantly.
Users' QoS concerns based CSB Policy	Time-based CSB policy	Response time and processing time are minimized effectively. The number of users' requests is increased. The overall performance of the CC environment is enhanced.	The majority of CSB policies degrade DC availability and efficiency. The DC selection assumption does not consider the next UR size.
	Total cost-based CSB Policy	The total cost is minimized effectively.	The majority of CSB policies do not consider the DC availability factor. The DC selection assumption does not consider the next UR size. Sometimes, the processing time and response time need more enhancement. The number of users requests is not always increased. Under-utilization of recourses and power-consuming policies.

(Continued)

Table 5 (continued)

Category	Criteria	Strengths	Weaknesses/Limitations
	DC availability-based CSB Policy	DC's availability has been significantly increased.	The majority of CSB policies are based on a centralized method, and it does not consider the DC load and processing power. The DC selection assumption does not consider the next UR size. The overall cost is not efficient enough. Sometimes, the processing time and response time need more enhancement.

The challenges and limitations of current cloud DC selection techniques can be summarized as:

- Lack of standardization: The absence of a universally accepted method for choosing cloud DCs has led to a lack of standardization within the industry.
- Opacity of information: The opacity of information surrounding data centers presents a challenge as many CSPs do not openly share comprehensive details, impeding users' ability to assess a DC's suitability. Moreover, reliance on CSP data for accuracy can be uncertain, potentially not reflecting actual conditions within the DC.
- Complexity: The process of selecting a DC is complex and consumes a significant amount of time as it involves evaluating various factors such as network infrastructure, security, reliability, and compliance requirements.
- The evolving nature of CC: The CC landscape is continuously evolving, with new technologies and services being frequently introduced, making it difficult to stay up-to-date and make informed decisions about DC selection.

7 Real-World Examples and Case Studies

In the dynamic realm of CC, the translation of theoretical concepts into tangible real-world applications is where true innovation and effectiveness come to light. While the theoretical underpinnings of CSB policies and DC selection techniques are vital, their true potential is unveiled when put into practical use. To bridge the gap between theory and reality, we delve into the realm of CSB policies and DC selection techniques, exploring detailed examples and case studies that demonstrate their real-world impact.

In this section, we embark on a journey that uncovers how CSB policies and DC selection techniques have been successfully applied in practice. These real-world instances, accompanied by verifiable references, shed light on how these methodologies contribute to optimizing cloud services, enhancing user experiences, and influencing operational outcomes. By delving into these practical cases, we gain a nuanced understanding of the diverse contexts in which these strategies thrive, and we witness their transformative influence on organizations and end-users.

- Proximity-Based Selection in Content Delivery

Netflix, a prominent online streaming service, leverages a proximity-based CSB policy to optimize content delivery. By collaborating with various Internet Service Providers (ISPs) and deploying Open Connect Appliances (OCAs) in close proximity to end-users, Netflix strategically selects data centers for streaming. This approach substantially reduces latency and guarantees uninterrupted playback experiences for subscribers [125].

- Performance-Centric Selection for High-Traffic Applications

Amazon Web Services' employs a performance centric CSB policy via its Elastic Load Balancing service. ELB intelligently distributes incoming application traffic across multiple Amazon EC2 instances based on real-time performance metrics. Continual monitoring of instances and routing traffic to those demonstrating optimal response times ensures elevated availability and responsive applications [126].

- Cost-Optimized Selection for Startups

DigitalOcean, a notable cloud infrastructure provider, extends diverse Droplet plans tailored to varying resource requirements. Startups and small enterprises can select Droplet plans based on budget considerations and resource prerequisites. This CSB policy empowers financially mindful organizations to optimize cloud expenditures while preserving performance standards [127].

- Hybrid Approach for Enterprise Workloads

Microsoft Azure introduces a hybrid approach by facilitating seamless integration between on-premises DCs and Azure's cloud services. Enterprises can implement a CSB policy that dynamically selects an apt data center, considering aspects such as workload demand, data sensitivity, and regulatory compliance. This strategy equips enterprises with the flexibility to harmonize performance, security, and cost-effectiveness [128].

- Machine Learning-Enhanced Dynamic Selection

Google Cloud harnesses machine learning for dynamic DC selection within its Global Load Balancing service. Through real-time analysis of user location, server health, and network conditions, Google Cloud optimally distributes incoming traffic to data centers demonstrating superior performance. This method efficiently curtails latency and maximizes resource utilization [129].

In the realm of cloud DC selection, the optimization of performance metrics presents a multi-dimensional challenge. While crucial benchmarks like availability, latency, throughput, resource utilization, and energy efficiency guide DC performance assessment, the intricate interplay among these metrics and the consequences of favoring specific parameters demand thorough exploration. This section provides a comprehensive analysis of how trade-offs and prioritization decisions associated with these metrics influence strategies for selecting DCs across diverse usage scenarios.

- Navigating Trade-Offs and Priorities: Balancing Performance Metrics

Selecting an ideal DC requires a nuanced understanding of the intricate balance among performance metrics. However, the elevation of one metric over others often necessitates careful consideration of trade-offs that may impact overall performance and user satisfaction. For instance, pursuing exceptionally low latency might lead to escalated resource usage or compromised energy efficiency. Similarly, an emphasis on resource optimization may inadvertently affect the availability of services

during periods of high demand. Recognizing these trade-offs is pivotal for making informed choices tailored to specific objectives.

- Impact of Prioritization: Guiding DC Selection Approaches

In the dynamic landscape of CC, varying user needs and the distinct goals of CSPs introduce diverse priorities. The ramifications of favoring specific performance metrics ripple through the DC selection process. For instance, a latency-critical application like real-time video streaming might emphasize low latency to ensure uninterrupted user experiences. In contrast, a data-intensive analytical workload could prioritize resource allocation for optimal processing efficiency.

Implementing CSB policies in real-world contexts presents multifaceted challenges that require innovative solutions. To demonstrate the practical feasibility and effectiveness of our proposed method, we address these challenges by harnessing the capabilities of Hadoop clusters, Apache CloudStack, and serverless computing. By strategically integrating these technologies, we navigate the complexities encountered during CSB policy implementation.

Navigating Real-World Challenges:

- Scalability and Adaptability: Handling diverse user requests at scale while accommodating fluctuating workloads demands robust scalability and adaptability. Furthermore, the ever-changing nature of cloud environments necessitates a flexible approach.
- Optimal Resource Allocation: Efficiently allocating resources, such as VMs and containers, is a key concern. Balancing resource utilization, adhering to QoS criteria, and considering factors like workload distribution and server capacities require careful management.
- Streamlined Management: Orchestrating interactions between multiple CSPs and users, ensuring seamless integration, and facilitating data transfer calls for streamlined management strategies.

Leveraging Technological Solutions:

- Enhanced Scalability and Flexibility with Hadoop Clusters: Leveraging Hadoop clusters empowers us to harness scalable and distributed computing frameworks. The utilization of Hadoop's MapReduce model enables efficient processing of vast datasets, while its intrinsic flexibility facilitates dynamic resource scaling to accommodate changing demands.
- Efficient Resource Allocation with Apache CloudStack: By capitalizing on Apache CloudStack's capabilities, we simplify the allocation, provisioning, and management of cloud resources. The sophisticated resource allocation features of CloudStack ensure optimal usage, aligning with QoS requisites and automating resource management tasks.
- Simplified Agility with Serverless Computing: Serverless computing abstracts infrastructure complexities, allowing us to focus solely on code deployment and execution. The adoption of serverless platforms streamlines application development, reduces operational intricacies, and offers cost-efficient solutions.

Demonstrating Practicality and Effectiveness:

By employing these technologies, we substantiate the practical feasibility and efficacy of our recommended CSB policy approach. Through concrete implementation and empirical evaluation, we showcase how our method adeptly addresses scalability challenges, optimizes resource utilization, and simplifies management complexities. Real-world scenarios, supported by Hadoop clusters, Apache

CloudStack, and serverless computing, tangibly exhibit the positive outcomes of our CSB policy recommendations, enriching DC selection and enhancing overall cloud service delivery.

In summation, the strategic amalgamation of Hadoop clusters, Apache CloudStack, and serverless computing provides practical remedies for real-world CSB policy implementation challenges. Through these technology-driven solutions, we not only surmount obstacles but also substantiate the workability and impact of our proposed method in practical settings. This synthesis between theoretical concepts and tangible application serves to advance CSB policy implementation within the dynamic realm of cloud computing.

8 Research Issues and Directions

Improving the performance of CC through the use of efficient CSB policies has been a focus of research. While many researchers proposed valuable CSB policies, the field of CSB policies and Cloud DC selection is still challenging, with many issues and challenges yet to be examined and have a potential for further development in this area. In this section, various issues and potential research directions are discussed.

- *Real-Time DC Selection*

CC is a dynamic environment where users' requests and Cloud DCs can change significantly. Therefore, a CSB policy should be able to adapt to real-time changes. Many existing CSB policies are centralized, meaning they find the best offline solution for the Cloud DC selection problem. While these centralized CSB policies use global information in the DC selection process, they often struggle to respond to real-time changes in online mode [130–135].

To date, real-time optimization techniques remain a challenge for DC selection due to the repetition-based and dynamic nature of DCs. One way to address this issue is to use online CSB policies based on metaheuristic optimization with a small population size. However, while Cloud DC selection can be impacted by the large volume, velocity, and variety of users' requests during runtime and the evolutionary phase, more effective and efficient use of problem information is needed [107]. In addition, machine learning techniques or data mining algorithms could be used to analyze the data to predict the needs of the Cloud users, the differences between DCs, and other factors to guide the search toward more accurate and efficient DC selection techniques in the CC paradigm.

As investigation approach, conducting empirical studies and simulations to assess the impact of real-time DC selection on user experiences and resource utilization, while the possible advantages are improving user satisfaction due to reduced latency, enhancing application responsiveness, and better utilization of available resources.

- *Heterogeneous DCs*

There have been numerous studies on the selection of Cloud DCs and policies for CSBs. However, these studies have focused on homogenous DCs. In reality, the needs of Cloud users are constantly changing, requiring DCs with high levels of QoS [135–139]. Therefore, there is a need for an effective and efficient CSB policy for heterogeneous DCs, which are often found in CC environments. However, integrating different types of DCs can be challenging due to security, deployment, and resource management issues. Even private DCs, which are under the control of a single entity, can be difficult to manage when they are of different types and run on different virtualization platforms.

As investigation approach is exploring the impact of selecting from a diverse range of heterogeneous data centers, which may vary in terms of architecture, performance capabilities, energy

efficiency, and geographic location, and the possible advantages are resource optimization, cost efficiency, performance enhancement, enhanced resilience.

- *Adaptive Dynamic DCs Selection*

There is a need for an adaptive CSB policy to handle the diverse needs of different users in a CC environment. This policy should be able to adjust the selection of DCs based on changing user and CSP needs. One potential solution is to use micro metaheuristic optimization, but this can be challenging due to the dynamic and complex nature of user requests and DCs. Additionally, the elasticity of DCs requires CSP policies to be able to adapt in real-time [140–145]. Therefore, using adaptive metaheuristic algorithms to manage CSB policies is an important research area to ensure efficient and effective DC selection. To meet real-time requirements, it may also be necessary to implement parallel techniques. The metaheuristic algorithms used in this context should be able to dynamically adjust their parameters and operations to meet the changing QoS needs of cloud users. Overall, adaptive management of CSB policies based on metaheuristic algorithms can support high performance and real-time DC selection in a CC environment.

As investigation approach, developing algorithms that dynamically adjust DC selection based on changing workloads, network conditions, and performance metrics, and the possible advantages are optimized resource allocation, reduced over-provisioning, and adaptable response to varying demands, leading to cost savings and improved QoS.

- *Large-Scale DCs Selection*

The rapid growth of CC has led to an increase in the number of users, services, and resources in the cloud, as well as the complexity of workflows. In the future, current CSB policies may struggle to handle the large volume of user requests in a CC environment that involves managing and selecting many DCs on the Internet. As a result, research on developing CSB policies for large-scale DC selection in a CC environment is emerging as a new trend. However, this is a challenging task due to the large search space and the presence of numerous local optima in this optimization problem. Metaheuristic algorithms based CSB policies are a promising approach for large-scale DC selection in a CC environment because they can provide optimal solutions in a non-deterministic time frame. However, there is still an open challenge in efficiently implementing time efficient large-scale CSB policies based on metaheuristic algorithms. One potential solution is to use partitioning techniques to divide large-scale Cloud DCs into smaller units and select these DCs using coevolution [146,147]. Another approach is to incorporate the features of available DCs in the CC environment into the design of the algorithm.

As the investigation approach is employing simulation studies to analyze how large-scale DC selection impacts data center distribution, load balancing, and network traffic patterns, and the possible advantages are efficient utilization of numerous data centers, enhanced fault tolerance, and effective handling of vast amounts of user requests.

- *Multiobjective-Based DCs Selection*

DC selection in CC often involves multiple conflicting objectives, as the needs of CSPs and cloud users can be very different. For example, a CSP may be focused on reducing costs through more efficient CSB policies, while cloud users demand high levels of QoS. Different cloud users may also have diverse QoS needs, such as low processing times, low costs, high availability, and so on [148–153]. As a result, the multiobjective nature of DC selection is becoming increasingly important in the future of CC. Most current CSB policies for DC selection only consider a single optimization objective, which

may not be sufficient for industrial applications in the future. The trend in CC will likely shift toward treating DC selection as a multiobjective problem and using CSB policies based on metaheuristics to provide efficient DC selection.

As investigation approach is developing decision frameworks that consider multiple conflicting objectives, such as latency, energy efficiency, cost, and reliability, and the possible advantages are enhanced decision-making by accommodating diverse stakeholder requirements and balancing competing goals for improved overall performance.

- *Distributed-Based DCs Selection*

One issue with CSB policies in a CC environment is that Cloud DCs are often located in different regions and may need to handle requests from users located anywhere around the world [154–156]. This means that it is important to have distributed CSB policies to handle the distribution of user requests. It may not be effective to select DCs in Europe for Asian users or to select DCs in Asia for American users, so having a distributed approach to DC selection is necessary. This is not only important for selecting DCs across different regions, but also for selecting different types of DCs within the same region.

As investigation approach is implementing distributed algorithms that enable autonomous decision-making by individual DCs while ensuring global optimization, and the possible advantages are reduced centralized control overhead, faster response times, and increased scalability as DCs collaborate in optimizing resource allocation.

9 Conclusion

In conclusion, optimizing DC selection remains a significant challenge within cloud computing environments. This article has synthesized prior research on DC selection, a pivotal aspect of cloud computing. Through our analysis, we introduced a taxonomy of CSB policies categorized into QoS requirements-based and CSP interests-based approaches. We systematically compared these policies, highlighting their strengths and limitations in a summarized table across various metrics including availability, latency, throughput, resource utilization, and energy efficiency. To advance this field, future research should focus on evaluating recent CSB policies using simulators and novel metrics within each category. Addressing issues such as real-time, adaptive, large-scale, and multi-objective DC selection, along with distributed approaches, will further enrich our understanding. As cloud computing, big data, IoT, and related technologies continue evolving, the importance of refining CSB policies to meet user QoS needs and CSP interests becomes paramount. This investigation serves as a valuable resource for researchers identifying gaps in DC selection within cloud computing. It presents a comprehensive overview of existing CSB policies and paves the way for future implementations utilizing diverse techniques including Hadoop clusters, Apache CloudStack or OpenStack, and cloud-native technologies like serverless computing and microservices. By refining CSB policies, we ensure efficient DC selection, contributing to the continued evolution of cloud computing landscapes.

Acknowledgement: I express my gratitude to University of Petra and American University of Madaba in Jordan for administrative and technical support.

Funding Statement: This research is funded by University of Petra, Jordan, Amman.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Y. Sanjalawe and S. Ammari; data collection: A. Aldarasieh, M. Bany Taha, and M. Aladaileh;

analysis and interpretation of results: Y. Sanjalawe, S. Ammari, A. Aldaraiseh, A. Aldarasieh, M. Bany Taha, M. Aladaileh; draft manuscript preparation: Y. Sanjalawe and S. Ammari. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: As this paper is a survey, it does not contain any original datasets generated or analyzed during the current study. All information and citations are sourced from previously published works, which are referenced accordingly in the manuscript.

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

References

1. Ibrahim, I. (2021). Task scheduling algorithms in cloud computing: A review. *Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(4),* 1041–1053.
2. Arif, M. (2018). A history of cloud computing. http://www.computerweekly.com/feature/A-history-of-cloud-computing
3. Ayachi M., Nacer H., Slimani H. (2021). Cooperative game approach to form overlapping cloud federation based on inter-cloud architecture. *Cluster Computing, 24(2),* 1551–1577.
4. Kang, Y., Pan, L., Liu, S. (2022). Job scheduling for big data analytical applications in clouds: A taxonomy study. *Future Generation Computer Systems, 1(3),* 12–29.
5. Sanjalawe, Y., Anbar, M., Al-E'mari, S., Abdullah, R., Hasbullah, A. et al. (2021). Cloud data center selection using a modified differential evolution. *Computers, Materials & Continua, 69(3),* 3179–3204. https://doi.org/10.32604/cmc.2021.018546
6. Alkhatib, A., Alsabbagh, A., Maraqa, R., Alzubi, S. (2021). Load balancing techniques in cloud computing: Extensive review. *Advances in Science, Technology and Engineering Systems Journal, 6(2),* 860–870.
7. Saxena D., Singh A. (2022). OFP-TM: An online VM failure prediction and tolerance model towards high availability of cloud computing environments. *The Journal of Supercomputing, 78(6),* 8003–8024.
8. Chen, Y., Shi, T., Ma, H., Chen, G. (2022). Automatically design heuristics for multi-objective location-aware service brokering in multi-cloud. *2022 IEEE International Conference on Services Computing (SCC)*, pp. 206–214. Barcelona, Spain, IEEE.
9. Wagle, S. (2015). Cloud service optimization method for multi-cloud brokering. *2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pp. 132–139. Bangalore, India.
10. Patel, H., Patel, R. (2015). Cloud analyst: An insight of service broker policy. *International Journal of Advanced Research in Computer and Communication Engineering, 4(1),* 122–127.
11. Ma L., Su W., Wu B., Jiang X. (2021). Real-time data backup in Geo-distributed data center networks against progressive disaster. *Photonic Network Communications, 41(3),* 211–221.
12. Daradkeh T., Agarwal A. (2021). Cloud workload and data center analytical modeling and optimization using deep machine learning. *Network, 2(4),* 643–669.
13. Parida, S., Pati, B., Nayak, S., Panigrahi, C. (2022). eMRA: An efficient multi-optimization-based resource allocation technique for infrastructure cloud. *Journal of Ambient Intelligence and Humanized Computing, 1(23),* 1–19.
14. Xue, S. (2017). Reconstruction of records storage model on the cloud datacenter. *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, pp. 310–315. San Diego, CA, USA, IEEE.
15. Dinita, I., Winckles, A., Wilson, G. (2016). A software approach to improving cloud computing datacenter energy efficiency and enhancing security through Botnet detection. *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pp. 816–819. Poitiers, France.

16. Martini, B., Gharbaoui, M., Cerutti, I., Castoldi, P. (2016). Network and datacenter resource orchestration strategies for mobile virtual networks over telco clouds. *2016 18th International Conference on Transparent Optical Networks (ICTON)*, pp. 1–4. Trento, Italy, IEEE.
17. Yadav, V., Malik, P., Kumar, A., Sahoo, G. (2015). Energy efficient data center in cloud computing. *Cloud Computing in Emerging Markets (CCEM), 2015 IEEE International Conference*, pp. 59–67. Bangalore, India, IEEE.
18. Amato A., Venticinque S. (2016). Multiobjective optimization for brokering of multicloud service composition. *ACM Transactions on Internet Technology, 16(2),* 13–25.
19. Kaspar, F., Müller-Westermeier, G., Penda, E., Mächel, H., Zimmermann, H. et al. (2013). Monitoring of climate change in Germany-data, products and services of Germany's National Climate Data Centre. *Advances in Science and Research, 10(1),* 99–106.
20. Khabbaz, M., Assi, C. (2015). Modelling and analysis of a novel deadline-aware scheduling scheme for cloud computing data centers. *IEEE Transactions on Cloud Computing, 1(23),* 73–98.
21. Padhy S., Chou J. (2021). MIRAGE: A consolidation aware migration avoidance genetic job scheduling algorithm for virtualized data centers. *Journal of Parallel and Distributed Computing, 154(12),* 106–118.
22. Bruneo, D. (2014). A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems, 25(3),* 560–569.
23. Rashid, A., Chaturvedi, A. (2019). Cloud computing characteristics and services: A brief review. *International Journal of Computer Sciences and Engineering, 7(2),* 421–426.
24. Sharkh, M., Kanso, A., Shami, A., Öhlén, P. (2016). Building a cloud on earth: A study of cloud computing data center simulators. *Computer Networks, 108(31),* 78–96.
25. Sharma, T., Singh, M., Selvan, S., Krah, D. (2022). Energy-efficient resource allocation and migration in private cloud data center. *Wireless Communications and Mobile Computing, 1(11),* 187–201.
26. Oracle, O. C. I. (2021). DynDNS. http://dyn.com/dns/
27. Wendell P., Jiang M., Freedman J., Rexford J. (2010). Donar: Decentralized server selection for cloud services. *ACM SIGCOMM Computer Communication Review, 40(4),* 231–242.
28. Gharehpasha, S., Masdari, M., Jafarian, A. (2021). Power efficient virtual machine placement in cloud data centers with a discrete and chaotic hybrid optimization algorithm. *Cluster Computing, 24(2),* 1293–1315.
29. Mondesire, S. C., Angelopoulou, A., Sirigampola, S., Goldiez, B. (2018). Combining virtualization and containerization to support interactive games and simulations on the cloud. *Simulation Modelling Practice and Theory, 93(3),* 233–244.
30. Chae, M., Lee, H., Lee, K. (2019). A performance comparison of linux containers and virtual machines using docker and KVM. *Cluster Computing, 22(1),* 1765–1775.
31. Mavridis, I., Karatza, H. (2019). Combining containers and virtual machines to enhance isolation and extend functionality on cloud computing. *Future Generation Computer System, 94(1),* 674–696.
32. Watada, J., Roy, A., Kadikar, R., Pham, H., Xu, B. (2019). Emerging trends, techniques and open issues of containerization: A review. *IEEE Access, 7,* 152443–152472.
33. Hussein, M. K., Mousa, M. H., Alqarni, M. A. (2019). A placement architecture for a container as a service (CaaS) in a cloud environment. *Journal of Cloud Computing, 8,* 1–15.
34. Al-Dhuraibi, Y., Paraiso, F., Djarallah, N., Merle, P. (2017). Elasticity in cloud computing: State of the art and research challenges. *IEEE Transactions on Services Computing, 11(2),* 430–447.
35. Piraghaj, S. F., Dastjerdi, A. V., Calheiros, R. N., Buyya, R. (2015). A framework and algorithm for energy efficient container consolidation in cloud data centers. *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pp. 368–375. Sydney, NSW, Australia, IEEE.
36. Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., Peterson, L. (2007). Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, pp. 275–287. Lisbon, Portugal.

37. Mavridis, I., Karatza, H. (2017). Performance and overhead study of containers running on top of virtual machines. *2017 IEEE 19th Conference on Business Informatics (CBI)*, vol. 2, pp. 32–38. Thessaloniki, Greece, IEEE.
38. Shi, T., Ma, H., Chen, G. (2018). Energy-aware container consolidation based on PSO in cloud data centers. *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. Rio de Janeiro, Brazil, IEEE.
39. Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal, 239(2),* 2.
40. Kavitha, B., Varalakshmi, P. (2018). Performance analysis of virtual machines and docker containers. *Data Science Analytics and Applications*, pp. 99–113. Chennai, India, Springer Singapore.
41. Boukhelef, D., Boukhobza, J., Boukhalfa, K., Ouarnoughi, H., Lemarchand, L. (2019). Optimizing the cost of DBaaS object placement in hybrid storage systems. *Future Generation Computer Systems, 93,* 176–187.
42. Tchana, A., de Palma, N., Safieddine, I., Hagimont, D. (2016). Software consolidation as an efficient energy and cost saving solution. *Future Generation Computer Systems, 58(5),* 1–12.
43. Ferreto, T. C., Netto, M. A., Calheiros, R. N., de Rose, C. A. (2011). Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems, 27(8),* 1027–1034.
44. Khan, A. A., Zakarya, M., Khan, R., Rahman, I. U., Khan, M. (2020). An energy, performance efficient resource consolidation scheme for heterogeneous cloud datacenters. *Journal of Network and Computer Applications, 150(2),* 102497.
45. Liu, J., Wang, S., Zhou, A., Xu, J., Yang, F. (2020). SLA-driven container consolidation with usage prediction for green cloud computing. *Frontiers of Computer Science, 14(1),* 42–52.
46. Shi, T., Ma, H., Chen, G. (2018). Multi-objective container consolidation in cloud data centers. *AI 2018: Advances in Artificial Intelligence*, pp. 783–795. Wellington, New Zealand, Springer International Publishing.
47. Liu, M., Pan, L., Liu, S. (2023). Cost optimization for cloud storage from user perspectives: Recent advances, taxonomy, and survey. *ACM Computing Surveys, 55(13),* 1–37.
48. Sen, A., Garg, A., Verma, A., Nayak, T. (2011). CloudBridge: On integrated hardware-software consolidation. *ACM SIGMETRICS Performance Evaluation Review, 39(2),* 14–25.
49. Saxena, D., Gupta, I., Kumar, J., Singh, A., Wen, X. (2021). A secure and multiobjective virtual machine placement framework for cloud data center. *IEEE Systems Journal, 11(1),* 12–32.
50. Elijorde, F., Lee, J. (2015). Cloudswitch: A state-aware monitoring strategy towards energy-efficient and performance-aware cloud data centers. *KSII Transactions on Internet and Information Systems, 9(12),* 4759–4775.
51. Bhardwaj A., Krishna C. (2021). Virtualization in cloud computing: Moving from hypervisor to containerization—A survey. *Arabian Journal for Science and Engineering, 46(9),* 8585–8601.
52. Agarwal, A., Sorathiya, P., Vaishnav, S., Desai, K., Mears, L. (2022). Design-as-a-service (DaaS) framework for enabling innovations in small and medium-sized enterprises (SMEs). *Journal of Mechanical Design, 1(11),* 1–12.
53. Chauhan, S. S., Pilli, E. S., Joshi, R. C., Singh, G., Govil, M. C. (2019). Brokering in interconnected cloud computing environments: A survey. *Journal of Parallel and Distributed Computing, 133(3),* 193–209.
54. Jyoti, A., Shrimali, M., Tiwari, S., Singh, H. P. (2020). Cloud computing using load balancing and service broker policy for IT service: A taxonomy and survey. *Journal of Ambient Intelligence and Humanized Computing, 11(11),* 4785–4814.
55. Duc, T. L., Leiva, R. G., Casari, P., Östberg, P. O. (2019). Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey. *ACM Computing Surveys, 52(5),* 1–39.
56. Kaur, M., Kaur, J., Vashist, S. (2014). A survey of techniques in cloud brokerage. *International Journal of Advanced Research in Computer Science and Software Engineering, 11(21),* 410–419.

57. Geetha, V., Hayat, M., Thamizharasan, M. (2014). A survey on needs and issues of cloud broker for cloud environment. *International Journal of Development Research, 4(5),* 1035–1040.
58. Barker, A., Varghese, B., Thai, L. (2015). Cloud services brokerage: A survey and research roadmap. *2015 IEEE 8th International Conference on Cloud Computing*, pp. 1029–1032. New York, NY, USA, IEEE.
59. Kumar P., Kumar R. (2019). Issues and challenges of load balancing techniques in cloud computing: A survey. *ACM Computing Surveys, 51(6),* 1–35.
60. Li X., Pan L., Liu S. (2022). A survey of resource provisioning problem in cloud brokers. *Journal of Network and Computer Applications, 1(2),* 111–131.
61. Fowley, F., Pahl, C., Jamshidi, P., Fang, D., Liu, X. (2016). A classification and comparison framework for cloud service brokerage architectures. *IEEE Transactions on Cloud Computing, 11(21),* 712–734.
62. Jula A., Sundararajan E., Othman Z. (2014). Cloud computing service composition: A systematic literature review. *Expert Systems with Applications, 41(8),* 3809–3824.
63. Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele University Technical Report TR/SE-0401*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=29890a936639862f45cb9a987dd599dce9759bf5
64. Kitchenham, B., Brereton, O., Budgen, D., Turner, M., Bailey, J. et al. (2009). Systematic literature reviews in software engineering—A systematic literature review. *Information and Software Technology, 51(1),* 7–15.
65. Javadpour, A., Wang, G., Rezaei, S. (2020). Resource management in a peer to peer cloud network for IoT. *Wireless Personal Communications, 115(3),* 2471–2488.
66. Kupiainen, E., Mäntylä, M., Itkonen, J. (2015). Using metrics in agile and lean software development—A systematic literature review of industrial studies. *Information and Software Technology, 62(2),* 143–163.
67. Charband, Y., Navimipour, N. (2016). Online knowledge sharing mechanisms: A systematic review of the state-of-the-art literature and recommendations for future research. *Information Systems Frontiers, 18(6),* 1131–1151.
68. Cisco (2021). Cisco global cloud index: Forecast and methodology. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html#_Toc503317520
69. Chiregi, M., Navimipour, N. (2016). A new method for trust and reputation evaluation in the cloud environments using the recommendations of opinion leaders' entities and removing the effect of troll entities. *Computers in Human Behavior, 60(11),* 280–292.
70. Kumrai, T., Ota, K., Dong, M., Kishigami, J., Sung, K. (2017). Multiobjective optimization in cloud brokering systems for connected internet of things. *IEEE Internet of Things Journal, 4(2),* 404–413.
71. Razaq, A., Tianfield, H., Barrie, P., Yue, H. (2016). Service broker based on cloud service description language. *2016 15th International Symposium on Parallel and Distributed Computing (ISPDC)*, pp. 196–201. Fuzhou, China, IEEE.
72. Park, J., Yeom, K. (2015). Virtual cloud bank: Cloud service broker for intermediating services based on semantic analysis models. *2015 IEEE 12th International Conference on Ubiquitous Intelligence and Computing and 2015 IEEE 12th International Conference on Autonomic and Trusted Computing and 2015 IEEE 15th International Conference on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pp. 1022–1029. Beijing, China.
73. Lee, C., Bohn, R., Michel, M. (2019). The NIST cloud federation reference architecture 5, NIST Special Publication 500-332. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-332.pdf
74. Paulsson, V., Emeakaroha, V., Morrison, J., Lynn, T. (2020). Cloud service brokerage: Exploring characteristics and benefits of B2B cloud marketplaces. *Measuring the Business Value of Cloud Computing, 1(11),* 57–71.
75. Vakili M., Jahangiri N., Sharifi M. (2019). Cloud service selection using cloud service brokers: Approaches and challenges. *Frontiers of Computer Science, 13(3),* 599–617.

76. Kurdi, H., Alsalamah, S., Alatawi, A., Alfaraj, S., Altoaimy, L. et al. (2019). Healthybroker: A trustworthy blockchain-based multi-cloud broker for patient-centered ehealth services. *Electronics, 8(6),* 602–621.
77. Torkura, K., Sukmana, M., Strauss, T., Graupner, H., Cheng, F. et al. (2018). CSBAuditor: Proactive security risk analysis for cloud storage broker systems. *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pp. 1–10. Cambridge, MA, USA, IEEE.
78. Gartner (2020). Gartner says cloud consumers need brokerages to unlock the potential of cloud services. http://www.gartner.com/newsroom/id/1064712
79. Hentschel, R., Strahringer (2020). A broker-based framework for the recommendation of cloud services. *Conference on e-Business, e-Services and e-Society*, pp. 409–415. Cham, Swansea, UK, Springer.
80. Jaikar, A., Noh, S. Y. (2015). Cost and performance effective data center selection system for scientific federated cloud. *Peer-to-Peer Networking and Applications, 8(5),* 896–902.
81. Li, C. L., Tang, J. H., Luo, Y. L. (2018). Distributed QoS-aware scheduling optimization for resource-intensive mobile application in hybrid cloud. *Cluster Computing, 21(2),* 1331–1348.
82. Larumbe, F., Sanso, B. (2016). Green cloud broker: On-line dynamic virtual machine placement across multiple cloud providers. *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, pp. 119–125. Hamburg, Germany, IEEE.
83. Mahalle, S., Tayde, S., Kaveri, R. (2015). Implementing service broker policies in cloud computing environment. *2015 International Conference on Communication Networks (ICCN)*, pp. 186–190. Gwalior, India.
84. Manikandan, R., Kousalya, G. (2016). A framework for an intelligent broker model of cloud services selection. *Asian Journal of Information Technology, 15(11)*, pp. 1776–1784.
85. Manasrah, A., Smadi, T., ALmomani, A. (2017). A variable service broker routing policy for data center selection in cloud analyst. *Journal of King Saud University-Computer and Information Sciences, 29(3),* 365–377.
86. Manasrah, A., Gupta, B. (2018). An optimized service broker routing policy based on differential evolution algorithm in fog/cloud environment. *Cluster Computing, 15(7),* 1–15.
87. Kumar, S., Parthiban, L. (2016). A novel approach for submission of tasks to a data center in a virtualized cloud computing environment. *International Journal of Advanced Computer Science and Applications, 7(8),* 238–242.
88. Beloglazov, A. (2013). *Energy-efficient management of virtual machines in data centers for cloud computing (Ph.D. Dissertation)*. The University of Melbourne, Australia.
89. Karamollahi, A., Chalechale, A., Ahmadi, M. (2018). Energy consumption improvement and cost saving by cloud broker in cloud datacenters. *International Arab Journal of Information Technology, 15(3),* 405–411.
90. Cong, P., Zhang, Z., Zhou, J., Liu, X., Liu, Y. et al. (2021). Customer adaptive resource provisioning for long-term cloud profit maximization under constrained budget. *IEEE Transactions on Parallel and Distributed Systems, 33(6),* 1373–1392.
91. Baker, T., Aldawsari, B., Asim, M., Tawfik, H., Maamar, Z. et al. (2018). Cloud-SEnergy: A bin-packing based multi-cloud service broker for energy efficient composition and execution of data-intensive applications. *Sustainable Computing: Informatics and Systems, 19(21),* 242–252.
92. Qiu, C., Shen, H., Chen, L. (2018). Towards green cloud computing: Demand allocation and pricing policies for cloud service brokerage. *IEEE Transactions on Big Data, 23(31),*238–251.
93. Mei, J., Li, K., Tong, Z., Li, Q., Li, K. (2019). Profit maximization for cloud brokers in cloud computing. *IEEE Transactions on Parallel and Distributed Systems, 30(1),* 190–203.
94. Gu H., Li X., Liu M., Wang S. (2021). Scheduling method with adaptive learning for microservice workflows with hybrid resource provisioning. *International Journal of Machine Learning and Cybernetics, 12(10),* 3037–3048.

95. Gąsior, J., Seredyński, F. (2021). An Automata-based profit optimization of cloud brokers in IaaS environment. *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pp. 723–725. Chicago, IL, USA, IEEE.
96. Sathish, A., Dsouza, D., Ballal, K., Archana, M., Singh, T. et al. (2021). Advanced mechanism to achieve QoS and profit maximization of brokers in cloud computing. *EAI Endorsed Transactions on Cloud Systems, 7(20),* 11–37.
97. Verma, A., Bhattacharya, P., Bodkhe, U., Saraswat, D., Tanwar, S., Dev, K., (2023). FedRec: Trusted rank-based recommender scheme for service provisioning in federated cloud environment. *Digital Communications and Networks, 9(1),* 33–46.
98. Motwani, A., Chaturvedi, R., Shrivastava, A. (2016). Profit based data center service broker policy for cloud resource provisioning. *International Journal of Electrical, Electronics, and Computer Engineering, 5(1),* 54–60.
99. Lin, W., Wu, W., He, L. (2019). An on-line virtual machine consolidation strategy for dual improvement in performance and energy conservation of server clusters in cloud data centers. *IEEE Transactions on Services Computing, 15(2),* 766–777.
100. Al Sukhni, E. (2016). K-nearest-neighbor-based service broker policy for data center selection in cloud computing environment. *International Research Journal of Electronics and Computer Engineering, 2(3),* 5–9.
101. Sankla, A. (2015). Analysis of service broker policies in cloud analyst framework. *Journal of the International Association of Advanced Technology and Science, 11(53),* 20–206.
102. Lin, W., Xiong, C., Wu, W., Shi, F., Li, K. et al. (2023). Performance interference of virtual machines: A survey. *ACM Computing Surveys, 55(12),* 1–37.
103. Amipara, H. (2015). A survey on cloudsim toolkit for implementing cloud infrastructure. *International Journal of Science Technology & Engineering, 1(12),* 239–243.
104. Chinnu, A., Madheswari, N. (2013). Performance optimized routing for SLA enforcement in cloud computing. *2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, pp. 689–693. Chennai, India, IEEE.
105. Semwal, A., Rawat, P. (2014). Performance evaluation of cloud application with constant data center configuration and variable service broker policy using CloudSim. *International Journal of Enhanced Research in Science Technology & Engineering, 3(1),* 1–5.
106. Mishra, K., Bhukya, N. (2014). Service broker algorithm for cloud-analyst. *International Journal of Computer Science and Information Technologies, 5(3),* 3957–3962.
107. Lee, J., Kim, J., Kang, J., Kim, N., Jung, N. (2014). Cloud service broker portal: Main entry point for multi-cloud service providers and consumers. *16th International Conference on Advanced Communication Technology*, pp. 1108–1112. Pyeongchang, Korea (South), IEEE.
108. Shah, D., Kariyani, A., Agrawal, L. (2013). Allocation of virtual machines in cloud computing using load balancing algorithm. *International Journal of Computer Science and Information Technology & Security (IJCSITS), 3(1),* 2249–9555.
109. Sharma, V., Rathi, R., Bola, K. (2013). Round-robin data center selection in single region for service proximity service broker in cloud analyst. *International Journal of Computers & Technology, 4(21),* 254–260.
110. Kapgate, D. (2014). Improved round robin algorithm for data center selection in cloud computing. *International Journal of Engineering Sciences & Research Technology (IJESRT), 3(2),* 686–691.
111. Ahmed, S. (2012). Enhanced proximity-based routing policy for service brokering in cloud computing. *International Journal of Engineering Research and Applications, 2(2),* 1453–1455.
112. Limbani, D., Oza, B. (2012). A proposed service broker policy for data center selection in cloud environment with implementation. *International Journal of Computer Technology and Applications, 3(3),* 1082–1087.

113. Rekha, M., Dakshayini, M. (2014). Service broker routing polices in cloud environment: A survey. *International Journal of Advances in Engineering & Technology, 6(6),* 2717–2767.
114. Jaikar, A., Kim, G., Noh, Y. (2013). Effective data center selection algorithm for a federated cloud. *Advanced Science and Technology Letters, 35(15),* 66–69.
115. Kapgate, D. (2014). Efficient service broker algorithm for data center selection in cloud computing. *International Journal of Computer Science and Mobile Computing, 3(1),* 355–365.
116. Cardellini, V., Di Valerio, V., Grassi, V., Iannucci, S., Presti, L. (2013). QoS driven per-request load-aware service selection in service-oriented architectures. *International Journal of Software and Informatics, 7(2),* 195–220.
117. Pawluk, P., Simmons, B., Smit, M., Mankovski, S. (2012). Introducing STRATOS: A cloud broker service. *2012 IEEE Fifth International Conference on Cloud Computing*, pp. 891–898. Berlin, Germany, IEEE.
118. Mishra, K., Kumar, S., Naik, S. (2014). Priority based Round-Robin service broker algorithm for Cloud-Analyst. *2014 IEEE International Advance Computing Conference (IACC)*, pp. 878–881. Gurgaon, India.
119. Kishor, K., Thapar, V. (2014). An efficient service broker policy for cloud computing environment. *International Journal of Computer Science Trends and Technology (IJCST), 2(4),* 11–32.
120. Naha K., Othman M. (2016). Cost-aware service brokering and performance sentient load balancing algorithms in the cloud. *Journal of Network and Computer Applications, 75(11),* 47–57.
121. Sharma, V. (2014). Efficient data center selection policy for service proximity service broker in CloudAnalyst. *International Journal of Innovative Computer Sciences and Engineering (IJICSE), 1(1),* 21–28.
122. Ramzanpoor, Y., Shirvani, M., Golsorkhtabaramiri, M. (2022). Multi-objective fault-tolerant optimization algorithm for deployment of IoT applications on fog computing infrastructure. *Complex & Intelligent Systems, 8(1),* 361–392.
123. Farzai, S., Shirvani, M., Rabbani, M. (2020). Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters. *Sustainable Computing: Informatics and Systems, 28(3),* 1003–1027.
124. Asghari, Y., Shirvani, M., Rahmani, M. (2022). A hybrid bi-objective scheduling algorithm for execution of scientific workflows on cloud platforms with execution time and reliability approach. *The Journal of Supercomputing, 1(1),* 1–53.
125. Netflix, Inc. (2023). Welcome to open connect. https://openconnect.netflix.com/en/
126. Amazon (2023). Elastic load balancing. https://aws.amazon.com/elasticloadbalancing/
127. DigitalOcean (2023). Comprehensive, cost-effective cloud computing. https://www.digitalocean.com/products/droplets
128. Azure (2023). Hybrid and multicloud solutions. https://azure.microsoft.com/en-us/solutions/hybrid-cloud-app
129. Google Cloud (2023). Cloud load balancing. https://cloud.google.com/load-balancing
130. Piraghaj, S., Dastjerdi, A., Calheiros, R., Buyya, R. (2017). ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers. *Software: Practice and Experience, 47(4),* 505–521.
131. Akhter, N., Othma, M. (2016). Energy aware resource allocation of cloud data center: Review and open issues. *Cluster Computing, 19(3),* 1163–1182.
132. Babazadeh, D., Muthukrishnan, A., Mitra, P., Larsson, T., Nordström, L. (2017). Selection of DC voltage controlling station in an HVDC grid. *Electric Power Systems Research, 144(9),* 224–232.
133. Ma, W. J., Wang, J., Lu, X., Gupta, V. (2016). Optimal operation mode selection for a DC microgrid. *IEEE Transactions on Smart Grid, 7(6),* 2624–2632.
134. Toutov, A., Toutova, N., Vorozhtsov, A., Andreev, I. (2022). Optimizing the migration of virtual machines in cloud data centers. *International Journal of Embedded and Real-Time Communication Systems (IJERTCS), 13(1),* 1–19.

135. Yan, J., Huang, Y., Gupta, A., Gupta, A., Liu, C. et al. (2022). Energy-aware systems for real-time job scheduling in cloud data centers: A deep reinforcement learning approach. *Computers and Electrical Engineering, 99(3),* 107688.

136. Zhang, J., Zhan, Z., Lin, Y., Chen, N., Gong, N. et al. (2011). Evolutionary computation meets machine learning: A survey. *IEEE Computational Intelligence Magazine, 6(4),* 68–75.

137. Vemula, D. R., Morampudi, M. K., Maurya, S., Abdul, A., Hussain, M. M. et al. (2022). Enhanced resource provisioning and migrating virtual machines in heterogeneous cloud data center. *Journal of Ambient Intelligence and Humanized Computing, 14(9),* 12825–12836.

138. Loncar, P., Loncar, P. (2022). Scalable management of heterogeneous cloud resources based on evolution strategies algorithm. *IEEE Access, 10,* 68778–68791.

139. Liang, B., Dong, X., Wang, Y., Zhang, X. (2022). A high-applicability heterogeneous cloud data centers resource management algorithm based on trusted virtual machine migration. *Expert Systems with Applications, 197,* 116762.

140. Gao, Y., Wang, L., Xie, Z., Qi, Z., Zhou, J. (2022). Energy-and quality of experience-aware dynamic resource allocation for massively multiplayer online games in heterogeneous cloud computing systems. *IEEE Transactions on Services Computing, 16(3),* 1793– 1806.

141. Gao, Y., Yu, L. (2017). Energy-aware load balancing in heterogeneous cloud data centers. *Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences*, pp. 80–88. Wuhan, China, ACM.

142. Zeng, J., Ding, D., Kang, K., Xie, H., Yin, Q. (2022). Adaptive DRL-based virtual machine consolidation in energy-efficient cloud data center. *IEEE Transactions on Parallel and Distributed Systems, 33(11),* 2991–3002.

143. Magotra, B., Malhotra, D., Dogra, A. K. (2023). Adaptive computational solutions to energy efficiency in cloud computing environment using VM consolidation. *Archives of Computational Methods in Engineering, 30(3),* 1789–1818.

144. Mukherjee, D., Ghosh, S., Pal, S., Aly, A. A., Le, D. N. (2022). Adaptive scheduling algorithm based task loading in cloud data centers. *IEEE Access, 10,* 49412–49421.

145. Wang, X., Cao, J., Buyya, R. (2022). Adaptive cloud bundle provisioning and multi-workflow scheduling via coalition reinforcement learning. *IEEE Transactions on Computers, 72(4),* 1041–1054.

146. Galante, G., Bona, L. (2012). A survey on cloud computing elasticity. *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing IEEE Computer Society*, pp. 263–270. Chicago, IL, USA.

147. Kantarci, B., Foschini, L., Corradi, A., Mouftah, H. T. (2012). Inter-and-intra data center VM-placement for energy-efficient large-scale cloud systems. *2012 IEEE Globecom Workshops*, pp. 708–713. California, USA, IEEE.

148. Yusoh, Z., Tang, M. (2010). A cooperative coevolutionary algorithm for the composite SaaS placement problem in the cloud. *International Conference on Neural Information Processing*, pp. 618–625. Berlin, Heidelberg, Springer.

149. Chen, Y., Shi, T., Ma, H., Chen, G. (2022). Automatically design heuristics for multi-objective location-aware service brokering in multi-cloud. *2022 IEEE International Conference on Services Computing (SCC)*, pp. 206–214. Barcelona, Spania, IEEE.

150. Yi, S., Li, X., Wang, H., Qin, Y., Yan, J. (2022). Energy-aware disaster backup among cloud datacenters using multiobjective reinforcement learning in software defined network. *Concurrency and Computation: Practice and Experience, 34(3),* e6588.

151. Khodayarseresht, E., Shameli-Sendi, A. (2023). A multi-objective cloud energy optimizer algorithm for federated environments. *Journal of Parallel and Distributed Computing, 174(1),* 81–99.

152. Shirvani, M. H. (2023). An energy-efficient topology-aware virtual machine placement in cloud datacenters: A multi-objective discrete JAYA optimization. *Sustainable Computing: Informatics and Systems, 38,* 100856.

153. Xie, T., Li, C., Hao, N., Luo, Y. (2022). Multi-objective optimization of data deployment and scheduling based on the minimum cost in geo-distributed cloud. *Computer Communications, 185(3),* 142–158.

154. Li, K. (2016). Power and performance management for parallel computations in clouds and data centers. *Journal of Computer and System Sciences, 82(2),* 174–190.

155. Hussain, M., Wei, L. F., Rehman, A., Abbas, F., Hussain, A. et al. (2022). Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers. *Future Generation Computer Systems, 132(2),* 211–222.

156. Sangaiah, A. K., Javadpour, A., Pinto, P., Rezaei, S., Zhang, W. (2023). Enhanced resource allocation in distributed cloud using fuzzy meta-heuristics optimization. *Computer Communications, 209(2),* 14–25.