International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 1, February 2023, pp. 872~883
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i1.pp872-883

Dynamic congestion management system for cloud service broker

Tariq Alwada'n[1], Abdel-Karim Al-Tamimi[2,5], Adel Hamdan Mohammad[3], Maher Salem[4], Yar Muhammad[1]
[1]School of Computing, Engineering and Digital Technologies, Teesside University, Middleborough, United Kingdom
[2]Computer Engineering Department, Yarmouk University, Irbid, Jordan
[3]Computer Science Department, The World Islamic Sciences and Education University, Amman, Jordan
[4]Department of Informatics, Faculty of Natural, Mathematical and Engineering Science, King's College London, London, United Kingdom
[5]Department of Computing, Sheffield Hallam University, Sheffield, United Kingdom

Article Info

Article history:

Received Mar 1, 2022
Revised Jul 18, 2022
Accepted Aug 19, 2022

Keywords:

Cloud broker
CloudSim
Congestion management
Service level agreements

ABSTRACT

The cloud computing model offers a shared pool of resources and services with diverse models presented to the clients through the internet by an on-demand scalable and dynamic pay-per-use model. The developers have identified the need for an automated system (cloud service broker (CSB)) that can contribute to exploiting the cloud capability, enhancing its functionality, and improving its performance. This research presents a dynamic congestion management (DCM) system which can manage the massive amount of cloud requests while considering the required quality for the clients' requirements as regulated by the service-level policy. In addition, this research introduces a forwarding policy that can be utilized to choose high-priority calls coming from the cloud service requesters and passes them by the broker to the suitable cloud resources. The policy has made use of one of the mechanisms that are used by Cisco to assist the administration of the congestion that might take place at the broker side. Furthermore, the DCM system is used to help in provisioning and monitoring the works of the cloud providers through the job operation. The proposed DCM system was implemented and evaluated by using the CloudSim tool.

Corresponding Author:

Tariq Alwada'n
School of Computing, Engineering and Digital Technologies, Teesside University
Middleborough, United Kingdom
Email: t.alwadan@tees.ac.uk

1. INTRODUCTION

Cloud computing, cluster computing, and grid computing models are designed to give access to a vast amount of computing power by sharing information technology (IT) resources or services by running a single or a group of system interfaces. Service computing is defined as IT resources (hardware or software) where service providers establish those resources and provide them on demand [1], [2]. Service requesters can pay the service providers on pay-per-use charging mode. It is essentially like a public service (such as: gas, water, power, and telephone) in which the clients are charged on a daily or monthly basis as a pay-per-use of the given unit [3], [4].

Cloud computing is an example of a model for offering on-demand network access to configure and share computing resources. By now, cloud computing has spread widely in several applications and usages. It becomes a significant part of the future generation of services and computing infrastructure at a reasonable cost. This feature will enable cloud users to develop and operate their resources as a pay-per-use, as described in Figure 1 [5]. Infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), and software as a service (SaaS) are the models of cloud services offered by cloud providers such as online file storage, social networking sites, and webmail [6].

Generally, the cloud service broker (CSB) is a software or a computer that works as a mediator between two or more groups to discover and decide about the suitable resources for specific tasks, to send input files of the jobs to the cloud resources, to monitor jobs, and to send the results of the job back to the clients [7], [8]. Figure 2 illustrates the job of the CSB. In the cloud environment, the cloud requesters can send their jobs to the cloud, specifically to the CSB to find suitable resources for them [9]. In this case, the cloud broker holds a database that contains the available resources (physical devices and virtual machines) on the cloud. The cloud can apply one or more algorithms to match the user jobs to the resources. These algorithms can be forced by the CSB policy. Also, the CSB performs some other tasks such as monitoring, controlling, provisioning, security, and substitution [9].

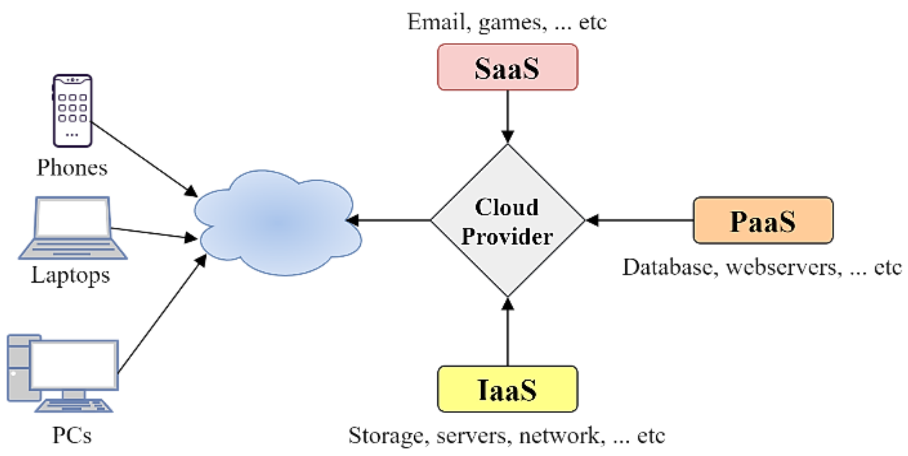
Email, games, ... etc
SaaS
Phones
Cloud Provider
PaaS
Laptops
Database, webservers, ... etc
PCs
IaaS
Storage, servers, network, ... etc

Figure 1. Cloud computing overview

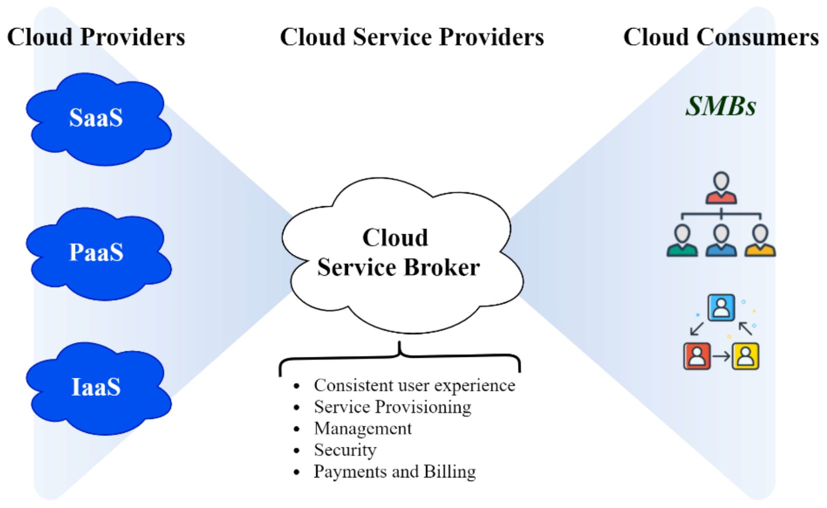

Figure 2. Cloud service broker [9]

In addition, the cloud broker can have the right to discuss arrangements with the cloud providers on the side of the cloud requester. In this situation, the CSB is given the power to split the job between many service providers to reduce the cost as possible. Furthermore, the cloud broker can provide the cloud users with an interface to hide some complexity and lets the cloud requesters deal with different cloud providers as if they were being purchased from one provider. This type of broker is named a cloud aggregator [10].

The architecture of CSB is shown in Figure 3 and its components are defined by Buyya *et al.* [11]. In this architecture, the CSB is divided into four layers:

- User interface: To offer the access connection between the broker and a user application interface. The application interpreter defines what is to be executed from the user, the task 'descriptions', and the desired quality of services. The service interpreter identifies the service prerequisites required for the execution. These requirements include service type, service location, and some other necessary services. The credential interpreter inspects the credentials for retrieving required services.
- Core services: This is where the main function of the broker takes place. The service negotiator receives the service requirements from the user interface. The scheduler decides the most suitable cloud providers for the user requested services based on its service and application requirements. The service monitor continually monitors the condition of cloud services through a periodic checking of the availability of recognized cloud services and searching for available new services.
- Execution interface: This offers execution provision for the user request. The dispatcher generates the required broker-agent and attaches the data files along with the user job to be posted to the cloud resources for implementation. The job monitor detects the execution level of the task to the results of the task is departed to the user upon job completion.
- Persistence: This layer is very important in the case of broker failure. It keeps the state of the core services, execution interface, and user interface in a database.

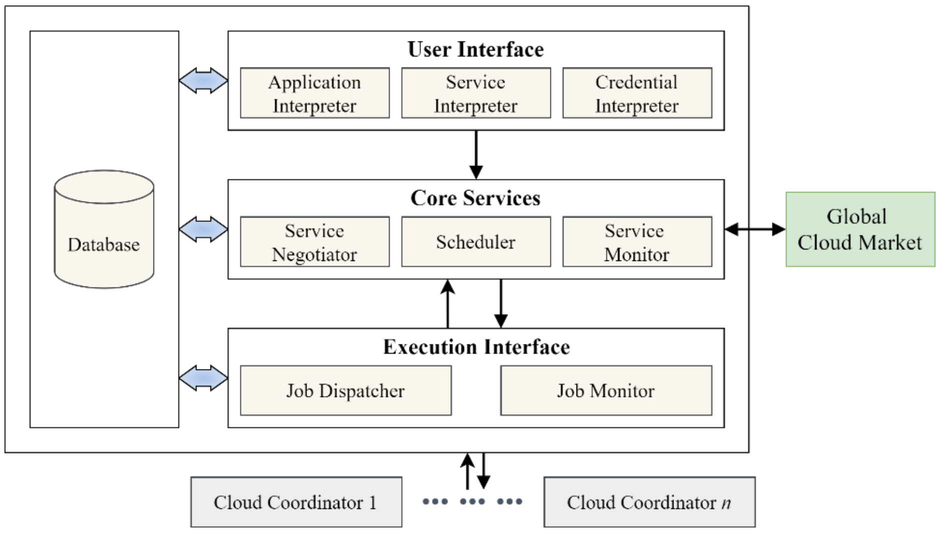

Figure 3. Cloud service broker architecture [11]

However, one situation that can be occurred when the broker rejects some requests if it reaches its full capacity of cloud requesters. In this case, the broker may not be able to deal with such a huge number of requests. The broker may (by the end) reject some requests to prevent the CSB from being overwhelmed. In this paper, a new dynamic congestion management (DCM) system is created, which is derived from Cisco queuing algorithms. This system can handle the huge amount of cloud requesters while considering the customers' quality of service conditions as regulated by the service-level agreement (SLA). Also, it is used to monitor and provision the work of the cloud providers during the jobs running. There are many cloud-computing models that have methods to provision and monitor their resources and define the cloud service broker job. The following are the most standard models in the field of cloud computing and brokering systems.

The dynamic resources provisioning and monitoring (DRPM) system proposed by Al-Ayyoub *et al.* [12] is a multi-agent system designed to handle the resources in the cloud provider whilst taking into consideration the required quality for the clients' requirements. These requirements are controlled by the SLA. DRPM system also comprises a new virtual machine with an algorithm named the host fault detection (HFD) algorithm to select which virtual machines to move to if the hosted physical machine becomes overloaded, varying on the source of the overload. The DRPM system was tested and evaluated using the CloudSim tool. The results showed that the DRPM system has increased resource utilization and, at the same time, has decreased power consumption by avoiding SLA violations.

Vecchiola *et al.* [13] proposed the Aneka system, which is a NET-based application (PaaS) for cloud computing. This system offers a group of runtime environment applications and APIs by several programming models and implements them on public and private cloud computing platforms such as GoGrid and Amazon EC2. The Aneka system also offers resource monitoring methods built on SLA orientation. This method runs as if the system accepts new tasks from cloud customers. It evaluates the time required to finish these tasks, in turn, to complete them with existing resource and match it with the SLA deadline time. If the estimated time to finish the new tasks is equal, then the system remains working; else, the system increases the cloud resources and remains run to avoid any SLA violation.

Siddiqui *et al.* [14] proposed the Elastic-JADE system which has three elements (local machine, Amazon EC2 cloud, and the cloud user). By using those elements, the system automatically balances Amazon's EC2's resources (by scaling up or down the resources) via the Java agent development framework (JADE) platform once heavy loads take place in the local platform. The agent at the client machine is in control of provisioning the whole system on both sides. Also, it communicates with the administrative agent in the Amazon EC2 then forwards directive orders for increasing or decreasing assets according to the system load. Bonvin *et al.* [15] introduced the scattered autonomic resources (Scarce) model, which is a multi-agent platform to dynamically administrate the resources by using an economic-based method. The agents work on the server side and are responsible for directing and scaling the resources and constantly checking the condition of the systems.

Venticinque *et al.* [16] proposed the open cloud computing interface (OCCI) framework, which supports monitoring, provisioning, and auto-configuration for the cloud resources to fulfill the application requirements at the infrastructure level (IaaS). The OCCI also contains a collection of API and protocols with a self-contained supplier and neutral platforms, which resolve several problems in the administration of usual tasks with the fulfillment of portability, interoperability, and integration requirements plus autonomic scaling, monitoring, and deployment.

Morrison [17] considered three distinct types of CSB: i) customer-centric, created for customers' service requirements and quality of service (QoS) providing to the customer; ii) solution centric, created for collecting services presenting from a wide of technical services presented in the cloud. One of the main jobs of this type is to ensure the integrity and the security of services; and iii) resource-centric, which works as a service assembler. The author proposed that a service broker comprises self-service entitlement, application catalogs, role-based access control, billing, SLA monitoring, metering, auditing, and reporting. He identified the CSB as a business model that helps the customers to choose, manage, organize, and coordinate the different services they require.

Praveen *et al.* in [18] introduced a new task scheduling and resource allocation schema for the cloud environment. The proposed load balancing schema uses the social group optimization (SGO) algorithm and consists of two autonomous phases: optimal resources allocation using SGO and effective scheduling of tasks using shortest-job-first (SJF). Their experimental results have shown the SGO-based SJF scheduler has reduced the makespan time and the number of active servers when compared to other first-come-first-serve (FCFS) and genetic algorithm (GA) based schedulers, and thus reduces the associated cost of using cloud services.

Guo *et al.* [19] introduced a load balancing schema for the edge cloud environments. The challenges that edge cloud environments face are emphasized due to the high level of granularity of the required resource billings and allocations processes. They proposed the use of auto regressive integrated moving average (ARIMA) and back propagation (BP) neural networks to estimate the required load. Accurate estimations can allow user data to be migrated to a less congested working node to ensure service continuity. When compared to load estimation using ARIMA models and GA models, the proposed model outperformed the other models in terms of load estimation accuracy. They also showed that using the proposed model could reduce the associated service expenditures effectively.

In this research, the authors focus on identifying the need for an automated system CSB that can help in utilizing the cloud power, enhancing its functionality, and improving its performance. The contributions of this work can be summarized in the following points: i) presents a DCM system which can manage the massive amount of cloud requesters while considering the required quality for the clients' requirements as regulated by the service-level agreement; ii) introduces a forwarding policy that can be utilized to choose high-priority requests coming from the cloud service requesters and passes them by the broker to the suitable cloud resources. The policy has made use of one of the mechanisms that are used by Cisco to ease the administration of the congestion that might take place at the broker side; and iii) furthermore, the DCM system is used to help in provisioning and monitoring the work of the cloud providers during the job running.

The following are some backgrounds about cloud service models, the SLA, and the congestion management algorithms. Figure 4 shows the cloud service models and the relations between them [20]. These models can be explained as:

- Infrastructure as a service: In this model, the provider is the holder of the equipment and tools. This business model presents the virtualization of resources on-demand [21]. The providers give the cloud users the ability to use those tools and equipment virtually instead of buying them. In this case, the customer pays per use on-demand [3]. This model also allows the customer to perform self-provision when using the 'provider's services.
- Software as a service: This is the most common model in cloud service. In this model, the client can access the applications and use them without the need to download or buy them. Also, it is a storage and supply model where the user obtains the storage area from the supplier on rent [22]. The users can use and access SaaS services by using the web browser.
- Platform as a service: This is the service for operating applications over the internet by hiring software and hardware structures from cloud providers [23]. It is mainly for application developers who can develop and test their applications.

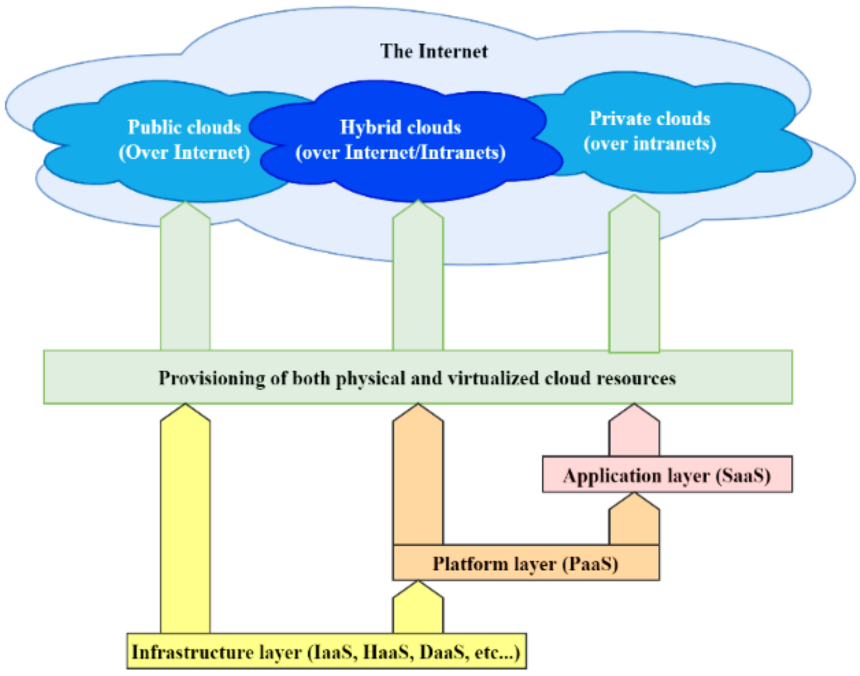
The Internet
Public clouds (Over Internet)
Hybrid clouds (over Internet/Intranets)
Private clouds (over intranets)
Provisioning of both physical and virtualized cloud resources
Application layer (SaaS)
Platform layer (PaaS)
Infrastructure layer (IaaS, HaaS, DaaS, etc...)

Figure 4. Cloud service models [20]

The clients under the cloud computing model do not have to worry about the problem of possessing and running the physical base needed for their job. In other words, they do not have to be concerned about programming teams and developers. Furthermore, the customers need not to concern about how their task will be performed or where it will be applied. The only thing that they need to be concerned about is the charge of using the resources, the quality of services promised, and the type of services they can acquire from the cloud providers. The ones who are worrying about reaching efficient utilization and monitoring resources are the CSB. By using smart ways and methods, the broker should obtain the finest quality of service assures without affecting or creating any breach in the SLA [12]. A cloud SLA is an agreement between the cloud providers and a cloud customer that ensures the lowest level of service is preserved. It guarantees levels of availability, reliability, and responsiveness of applications and systems while determining who will have control when there is a service interruption [24].

Because of the growing demands on cloud services, nowadays, applications and cloud service requesters are asking to downgrade the completion time for their jobs while reducing the associated cost. At the same time, they ask for the maximum QoS during their 'jobs' execution. Most of the requests now are quite delicate for the delays faced when running and transferring over the internet. That is why it is required to support a different type of traffic with different quality of services. The greatest significant side of this; is how to distribute the available resources while facing congestion. To continue with this, it is required that several methods and tools help to distinguish between the types of traffic coming to the cloud. This can be done through prioritizing [25].

This research introduces a forwarding policy that can be employed to choose high-priority calls coming from the cloud service requesters (as an extra service) and deliver them by the cloud service broker to the suitable cloud resources. So, there is a need to line up the low priority calls and hold them in the broker memory till the high priority calls get handled by the cloud. This paper has made use of one of the mechanisms that are used by Cisco to smooth the administration of the congestion that may take place at the cloud service broker side. Cisco has many mechanisms which support the queuing on Cisco router interfaces using hardware and software modules [25]. The chosen mechanism is the weighted fair queuing (WFQ). This mechanism does not permit classification options to be configured. Instead, WFQ categorizes packets automatically, with every flow being placed into a distinct queue according to its priority, and each one of these queues has a specific weight [26]. The process starts as a round robin (RR) mechanism, where time slices are allocated to each process in equal portions and circle order, but the processer allows the flow of traffic to get through based on the weight of that queue in the round. In this case, the WFQ can solve the starvation problem that could cause if the RR mechanism is being used. This paper is organized as: section 2 presents the proposed system model. The experiment and results are discussed in section 3. Finally, the conclusions are presented in section 4.

2. SYSTEM MODEL

The details of the DCM system are explained in this part. The DCM system is a multi-agent model that takes into consideration various elements when forming the decision, for example, the number and the shortcoming of cloud resources, the customers' fulfilment, and the customers' QoS requirements. This system is created by adapting and combing both the DRPM system introduced in [12] and the cloud broker architecture introduced in [11]. Figure 5 shows the architecture of the DCM system. The model is split into three parts: The cloud service users part, the CSB part, and the cloud service provider part.

2.1. The cloud service users part

In this part, a local agent is allocated to every customer. This agent is responsible for remarking the customers' requests based on their priority. This is considered as an extra paid service that the users can pay for the cloud to guarantee the priority of their jobs over the cloud among other users. If this service is not paid (exist), in this case, the local agent will consider the priority as a low priority. The local agent then sends the marked requests to the CSB, precisely to the classifier inside the broker. At the user's request, the job specifications are determined, such as the type of virtual machine (VM), the hardware, and the software required. Once a new call for a VM with particular features is collected from the client, the local agent marks these requests as one of three cases: high, low, and medium. This will help the user to process some urgent jobs where needed. The local agent can utilize the history of the user requests to help in marking the new requests. The output of remarking step is sent to the classifier in the CSB for the next step.

2.2. The cloud service broker part

This part is the core of the cloud system and the DCM system. The element that is accountable for getting the users' jobs in the CSB is the classifier. The main job of the classifier is to recognize which job should be processed first in the case of job congestion. As the CSB continues to receive jobs from different cloud users, it may reach to reject some requests due to the huge number of requests. The DCM system plays a very important role in this case. Instead of rejecting the requests randomly, the classifier can make use of the jobs marking that has been done by the local agent on the cloud service user's part to organize the requests in a way that can take into account the QoS requirements as regulated by the SLA. If the CSB is not congested the classifier applies the first in first out (FIFO) order for the incoming requests, but if the CSB got congested, then the classifier depends on the WFQ mechanism to solve the congestion. WFQ categorizes packets automatically, with every flow being placed into a distinct queue according to its priority, and each one of these queues has a specific weight. The process starts as a RR mechanism, where time slices are allocated to each process in equal portions and circle order, but the processer allows the flow of traffic to get through, based on the weight of that queue in the round. The advantage of using the WFQ mechanism is that it can solve the starvation problem that could cause if the RR mechanism is being used. Figure 6 shows how the classifier deals with the incoming requests from cloud users. The classifier can make use of the Broker Buffer in case of some requests need to be put on hold for further classification.

After the classifier decides which users' requests should be processed first, it handles them to the Scheduler for further process. The Scheduler decides the most suitable cloud services for the client requested services built on its service requirements and application. The service monitor continually monitors the status of cloud services through a periodic checking of the availability of recognized cloud services and searching for available new services.

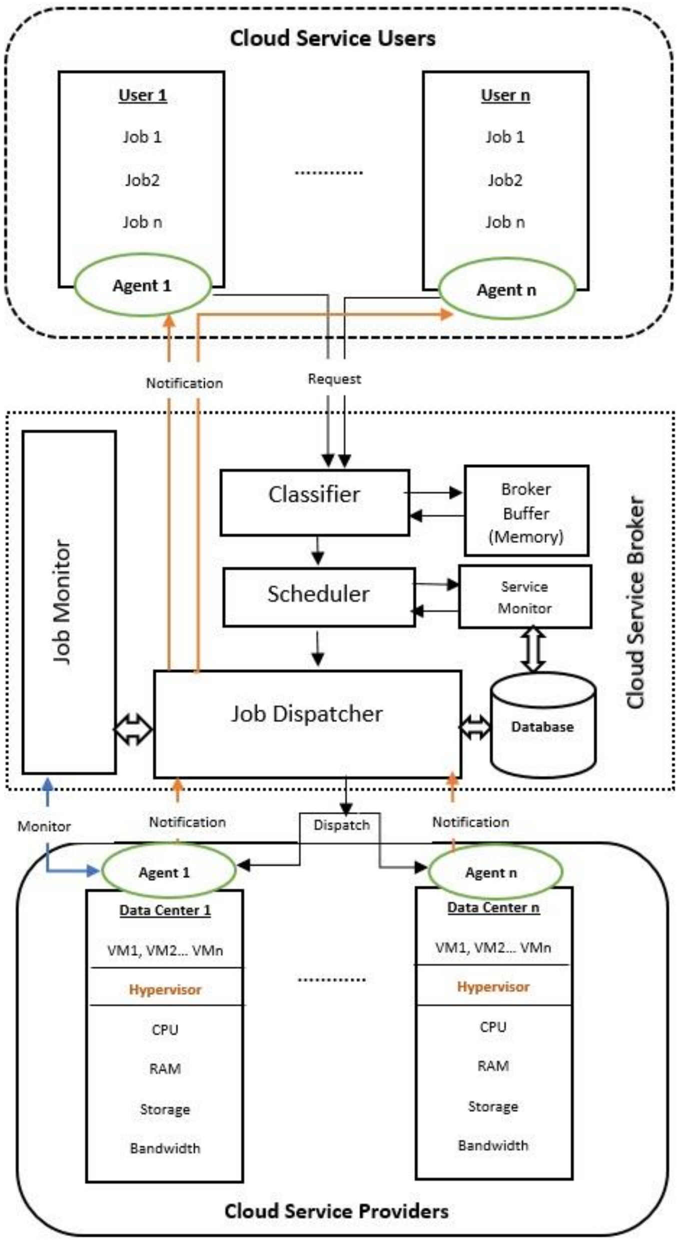

Figure 5. Architecture DCM system

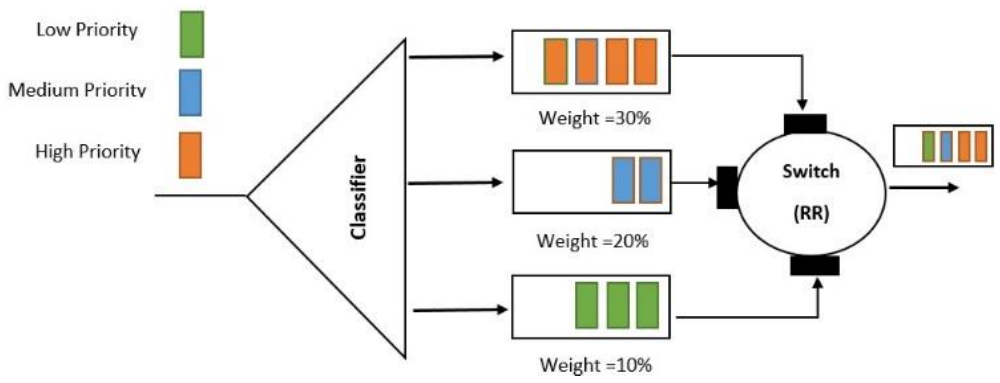

Figure 6. Classifier mechanism

Later, the job dispatcher receives the user application (the user's job) and attaches the data files with the user application to be posted to the chosen cloud resources for implementation. The job monitor detects the execution levels of the job to send back the results of the job to the user as soon as the job is finished. This is done by checking the status of the job with the agent attached to the cloud resources. If the broker is crashed for any reason, each element in CSB maintains its state in a database for recovery later. Finally, when the job is accomplished, the agent attached to the cloud resource sends the results back to the job dispatcher which in its turn notify the job monitor about the completion of the job and sends the results back to the local agent of the cloud users.

2.3. The cloud service provider's part

This part contains the physical machine (central processing unit (CPU), random access memory (RAM), storage, and bandwidth) and the virtual machines. Each physical machine may contain one or more VMs. For each resource, there is an agent that is responsible for receiving the job description from the CSB and notifying the job monitor with parodic messages about the status of the job. It is used to provision and monitors the work of the cloud providers during the job running.

3. EXPERIMENTS AND RESULTS

In this part, two experiments were created and evaluated using a CloudSim [27]. A CloudSim is employed to evaluate the fulfilment of the suggested DCM system. CloudSim is one of the primary simulation environments for the cloud computing model. Both experiments have the same settings. The first experiment was divided into two parts. The settings include 300 cloudlets (jobs) from 3 cloud users (100 each), with one machine (one physical machine and three virtual machines) as cloud providers. The service broker is designed to accept up to 250 cloudlets and reject all the requests above that. The simulation period of 24 hours in the scheduling interval is 300 seconds. Tables 1, 2, and 3 show the features of the physical machine, the VMs, and the cloudlets, respectively.

The difference between the two parts is that in the first one, no paid services for medium or high priority were done. In this situation, the local agents assigned for each of the three requesters will count the priority for each job as equal (low priority for the three users. In the next part, the local agent at one of the requesters marked the cloudlets as a high or medium priority whereas the other two requesters with no marking priority. Both experiments intended to direct all the cloudlets to the cloud broker simultaneously to examine the time needed to process all the jobs together. Also, it is planned to examine the number of rejected jobs due to the huge number of requests.

Table 1. Virtual machines features

Feature	Quantity/Size
Quantity of VMS	3
VMs categories	4
Million instructions per second of CPU	1000
RAM size	1024
Bandwidth	100 Mbit/s
Storage size	5 GB

Table 2. Physical machine features

Feature	Quantity/Size
Number of machines	1
VMs categories	4
Million instructions per second of CPU	1000
Number of PEs	2
RAM size	1024
Bandwidth	1 Gbit/s
Storage size	10 GB

Table 3. Cloud features

Feature	Quantity/Size
Number of Jobs	300
Output size	400
File size	400
Job length	800

The two parts of the simulations were compared and evaluated on several aspects such as the time needed to finish the cloudlets and the number of cloudlets rejected for every user. First Part: 300 cloudlets were sent to the service broker from three users. There is no priority at this stage. The broker received the cloudlets and processed them as first in first out jobs. The first 100 cloudlets coming from the first user were processed without any rejected jobs. The same for the second user. But for the third user, the broker rejected the last 50 jobs as they reached the boundaries of the accepted requests (as in its policy). The third user had to send back the rest of the jobs later for processing. Figures 7 and 8 show the processing time and the rejected jobs, respectively, for the first part of the experiment.

Second part: again 300 cloudlets were directed to the broker. But at this stage, the local agent marked the jobs for user number 3 as paid service-high priority service. The other users remain a low priority (normal priority). Figures 7 and 8 show the processing time and the rejected jobs, respectively, for the second part of the test.

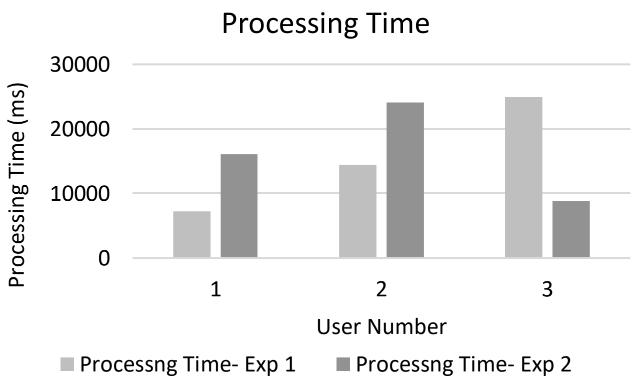

Figure 7. Processing time for both parts-experiment 1

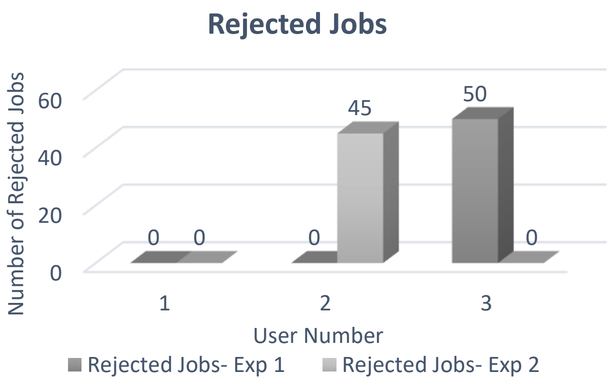

Figure 8. Rejected Jobs for both parts-experiment 1

The results show that user 3 has taken advantage of the paid service, as all of user 3's tasks were processed, and the time needed for the processing was very short compared to the first part. Also, there are no rejected tasks for this user. So, by using this way, it has given the guarantee of completion of all the jobs in a short time. The other results for users 1 and 2 show that user 1 still has no rejected jobs while user 2 has 45 rejected jobs, as the broker processed their cloudlets according to the first in first out mechanism. But the overall processing time for the 300 cloudlets is slightly less than the first part. Also, the number of rejected jobs in the second part is slightly less than in the first one.

To validate the previous results, another scenario was applied. The settings this time include 300 cloudlets (jobs) from 4 cloud users (75 each), with three virtual machines and one physical machine as a cloud provider. The broker is designed to handle up to 200 cloudlets and rejects all the requests above that. The same characteristics of the physical machine, the VMs, and the cloudlets, as in the first experiment as shown in Tables 1, 2, and 3 respectively.

Again, the experiment was divided into two parts. The two parts of the experiments were compared and evaluated on several aspects as the time needed to finish the cloudlets and the number of cloudlets rejected for each user. First part: 300 cloudlets were sent to the broker from 4 users. No priority at this stage. The broker received the cloudlets and processed them like a FIFO job. The first 75 cloudlets coming from the first user were processed without any rejected jobs. The same for the second user. But for the third user, the broker rejected the last 25 jobs as they reached the boundaries of the accepted jobs (as in its policy). The cloudlets for the last user were rejected. The third and the fourth users had to resend the rest of their jobs later for processing. Figures 9 and 10 show the processing time and the rejected jobs, respectively, for the first part of the experiment.

Second part: again, 300 cloudlets were transmitted to the broker. In this round, the local agent marked the jobs for user number 3 as a medium priority (paid service), and the local agent for user 4 marked the jobs as high priority. The rest of the users remain a low priority. Figures 9 and 10 show the processing time and the rejected jobs, respectively, for the second part of the experiment.

The results show that users 3 and 4 have taken advantage of the priority service when all of the users 3's jobs and '4's jobs were processed, and the time needed for the processing was very short compared to the first part. Also, there are no rejected jobs for those users. The other results for users 1 and 2 show that user 1 still has no rejected jobs whilst user 2 has 73 rejected tasks, as the broker processed their cloudlets according to the first in first out mechanism. But the overall processing time for the 300 cloudlets is slightly bigger than the first part. Also, it is noted that the rejected jobs are fewer than the ones in part 1 of the second experiment.

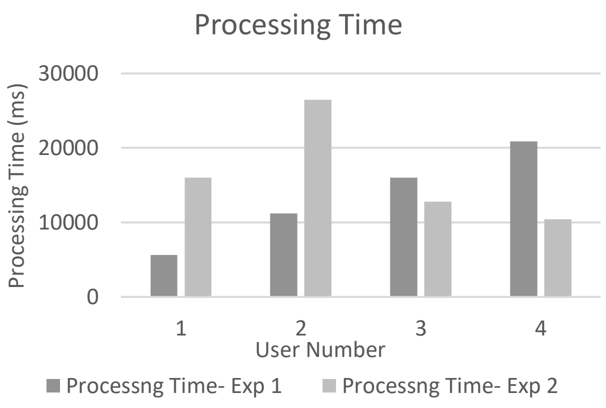

Figure 9. Processing time for both parts-experiment 2

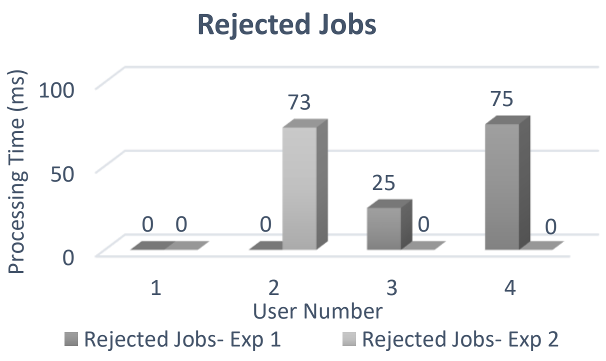

Figure 10. Rejected jobs for both parts-experiment 2

4. CONCLUSION

This paper presents a DCM system which can handle a huge amount of cloud requesters while considering the required quality for the clients' requirements as regulated by the SLA. This system is being applied by the CSB and local agents attached to the cloud service requesters and providers. The proposed DCM system is assessed using the CloudSim tool. The results indicate that using the DCM system improves the degree of customers' QoS conditions during resource employment and, at the same time, avoids cloud SLA violations. Numbers of future works that can be implemented in this field, such as user responsive (Encoding); where clouds are usually very complex and difficult to encode and enforce the user policy within the broker decisions.

REFERENCES

[1] W. Chai and S. J. Bigelow, "cloud computing: What is cloud computing?," *TeachTarget*, Dec 2021. https://www.techtarget.com/searchcloudcomputing/definition/cloud-computing (accessed Jul. 18, 2022).

[2] T. Alwadan, O. Al-Zitawi, S. Khawaldeh, and M. Almasarweh, "Privacy and control in mobile cloud systems," *International Journal of Computer Applications*, vol. 113, no. 1, pp. 12–15, Mar. 2015, doi: 10.5120/19789-1170.

[3] DOCPLAYER, "Managing the real cost of on-demand enterprise cloud services with chargeback models," *Cisco*, 2010. Accessed: Jul. 18, 2022. [Online]. Available: https://docplayer.net/968632-Managing-the-real-cost-of-on-demand-enterprise-cloud-services-with-chargeback-models.html (accessed Jul. 18, 2022).

[4] T. Alwadan, O. Al-Zitawi, and J. O. Atoum, "Cloud computing: privacy, mobility and resources utilization," *International Journal of Computer Trends and Technology*, vol. 41, no. 1, pp. 29–36, Nov. 2016, doi: 10.14445/22312803/IJCTT-V41P106.

[5] A. M. Manasrah, T. Smadi, and A. ALmomani, "A variable service broker routing policy for data center selection in cloud analyst," *Journal of King Saud University-Computer and Information Sciences*, vol. 29, no. 3, pp. 365–377, Jul. 2017, doi: 10.1016/j.jksuci.2015.12.006.

[6] T. Alwadan, "Mobility in cloud systems," *International Journal of Computer Trends and Technology*, vol. 11, no. 5, pp. 202–205, May 2014, doi: 10.14445/22312803/IJCTT-V11P143.

[7] T. Alwadan, "Cloud computing and multi-agent system: monitoring and services," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 9, pp. 2435–2444, 2018.

[8] K. Bubendorfer and D. Abramson, "The nimrod/G grid resource broker for economics-based scheduling," in *Market-Oriented Grid and Utility Computing*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2009, pp. 371–402.

[9] J. E. Brown, "Cloud service broker," *Machine 2 Machine*, 2011. https://machine2twomachine.wordpress.com/2011/08/04/hello-world/ (accessed Jul. 18, 2022).

[10] M. Rouse, "Cloud computing," *Techopedia*, 2022. https://www.techopedia.com/definition/2/cloud-computing (accessed Jul. 18, 2022).

[11] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and Architectures for Parallel Processing*, Springer Berlin Heidelberg, 2010, pp. 13–31.

[12] M. Al-Ayyoub, Y. Jararweh, M. Daraghmeh, and Q. Althebyan, "Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure," *Cluster Computing*, vol. 18, no. 2, pp. 919–932, Jun. 2015, doi: 10.1007/s10586-015-0449-5.

[13] C. Vecchiola, X. Chu, and R. Buyya, "Aneka: a software platform for .NET-based cloud computing," *High speed and large scale scientific computing*, vol. 18, no. 267–295, 2009.

[14] U. Siddiqui, G. A. Tahir, A. U. Rehman, Z. Ali, R. U. Rasool, and P. Bloodsworth, "Elastic JADE: dynamically scalable multi agents using cloud resources," in *2012 Second International Conference on Cloud and Green Computing*, Nov. 2012, pp. 167–172, doi: 10.1109/CGC.2012.60.

[15] N. Bonvin, T. G. Papaioannou, and K. Aberer, "Autonomic SLA-driven provisioning for cloud applications," in *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2011, pp. 434–443, doi: 10.1109/CCGrid.2011.24.

[16] S. Venticinque, L. Tasquier, and B. Di Martino, "Agents based cloud computing interface for resource provisioning and management," in *2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, Jul. 2012, pp. 249–256, doi: 10.1109/CISIS.2012.139.

[17] D. Morrison, "The evolution of cloud service brokerage," *Experts Forum*, 2012. Accessed: Jul. 18, 2022. [Online]. Available: https://easi-clouds.eu/wp-content/uploads/2014/12/Deliverable_1_5_Final_business_models.docx

[18] S. P. Praveen, K. T. Rao, and B. Janakiramaiah, "Effective allocation of resources and task scheduling in cloud environment using social group optimization," *Arabian Journal for Science and Engineering*, vol. 43, no. 8, pp. 4265–4272, Aug. 2018, doi: 10.1007/s13369-017-2926-z.

[19] J. Guo, C. Li, Y. Chen, and Y. Luo, "On-demand resource provision based on load estimation and service expenditure in edge cloud environment," *Journal of Network and Computer Applications*, vol. 151, Feb. 2020, doi: 10.1016/j.jnca.2019.102506.

[20] T. Alwadan, "Cloud computing topology: towards enhancing the performance," *International Journal of Computer Science and Information Security*, vol. 14, no. 11, pp. 654–658, 2016.

[21] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14–22, Sep. 2009, doi: 10.1109/MIC.2009.119.

[22] DOCPLAYER, "Quattra s cloud vision and framework value," *Quattra Inc.* https://docplayer.net/8761493-Quattra-s-cloud-vision-framework-value.html (accessed Jul. 18, 2022).

[23] W. Chai, K. Brush, and S. J. Bigelow, "What is PaaS? Platform as a service definition and guide," *Tech Accelerator*, 2012. https://www.techtarget.com/searchcloudcomputing/definition/Platform-as-a-Service-PaaS (accessed Jul. 18, 2022).

[24] J. Montgomery and S. Lelii, "Cloud SLA (cloud service-level agreement)," *TechTarget*, 2011. https://www.techtarget.com/searchstorage/definition/cloud-storage-SLA (accessed Jul. 18, 2022).

[25] S. Szilágyi and B. Almási, "A review of congestion management algorithms on cisco routers," *Journal of Computer Science and Control Systems*, vol. 5, no. 1, 2012.

[26] Cisco, "Implementing quality of service," *Cisco*, 2008. https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-packet-marking/13747-wantqos.html (accessed Jul. 18, 2022).

[27] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011, doi: 10.1002/spe.995.

BIOGRAPHIES OF AUTHORS

Tariq Alwada'n is a senior lecturer in Computer Networks/Cybersecurity at the School of Computing, Engineering and Digital Technologies at Teesside University, United Kingdom since July 2020. He holds a PhD in Computer Science from De Montfort University, UK since 2012. His master's degree is in Computer and Information Networks from University of Essex, UK in 2007. His BS is in Computer Engineering from Al Balqa Applied University, Jordan in 2005. His main research interests are cloud computing, fog computing, grid computing, iot, big data mobility in distributed systems, and security. He is certified and licensed as a CCNA instructor from Cisco Network Academy before joining Teesside University, He worked as a lecturer at the Higher Colleges of Technology, UAE between 2019 and 2020. He also worked as an Associate Professor of Computer Science at WISE University, Jordan between 2012 and 2019. He can be contacted at email: t.alwadan@tees.ac.uk.

Abdel-Karim Al-Tamimi is a senior lecturer in the Department of Computing at Sheffield-Hallam University and a member of the Applied Software Engineering Research Group. Received his M.Sc. and Ph.D. degrees in Computer Engineering from Washington University in St. Louis (ranked 50th by The World University Ranking) under the advisement of Prof. Raj Jain in 2007 and 2010, respectively. His research interests include machine learning, multimedia systems, and cyber security. He can be contacted at email: a.al-tamimi@shu.ac.uk.

Adel Hamdan Mohammad is an Associate Professor in Computer Science Department, The World Islamic Sciences and Education University, Amman-Jordan. Born in Jordan in 1973. He received His B.Sc. from Philadelphia University in 1998. Then Master's degree in Computer Information Systems from Arab Academy for Banking and Financial Science in 2005, and Ph.D. in Computer Information Systems in 2009 from Arab Academy for Banking and Financial Science. He Works in the Applied Science Private University and The World Islamic Sciences and Education University. His research interest is machine learning, artificial intelligence, and cybersecurity. Certified CCNA, CEH. Also certified and licensed to teach and issue certificates in the following courses from Cisco Network Academy: Cybersecurity Essentials, IoT, NDG Linux, Introduction to Cybersecurity and others. He can be contacted at email: adel.hamdan@wise.edu.jo.

Maher Salem is a lecturer in the department of informatics at King's College London, UK. He received his M.Sc. from Duisburg-Essen University, and the Ph.D. from Kassel University, both from Germany. He is also the department Career Officer at King's College. Dr. Salem has more than 20 years of industrial and academic experience in cybersecurity. His research interests include intrusion detection system, threat intelligence, machine learning, and cybersecurity in education. He can be contacted at email: maher.salem@kcl.ac.uk.

Yar Muhammad is working as a Senior Lecturer (Assistant Professor) with Teesside University, United Kingdom, where he is also a part of the Centre for Digital Innovation. He received a Ph.D. degree in information communication technology (ICT) from the Tallinn University of Technology, Estonia, in 2015 and a master's degree in computer engineering from Mid Sweden University, Sweden, in 2009. He taught at the University of Tartu, Estonia. He received a Young Investigator Award, which was awarded by Springer and IFMBE at 16th Nordic-Baltic Conference on Biomedical Engineering and Medical Physics and Medicinteknikdagarna 2014, Sweden; and he was runner-up for the Best Paper Award in the 26th ISSC, Ireland, in 2015. He can be contacted at email: Yar.Muhammad@tees.ac.uk.