National College of Ireland

Efficient Virtual Machine Migration using Apache Kafka Messaging Services and Spring Boot Microservices

MSc Research Project
Cloud Computing

Arun Kumar Dasari
Student ID: X20155361

School of Computing
National College of Ireland

Supervisor: Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Arun Kumar Dasari
Student ID:	X20155361
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Aqeel Kazmi
Submission Due Date:	31/01/2022
Project Title:	Efficient Virtual Machine Migration using Apache Kafka Messaging Services and Spring Boot Microservices
Word Count:	8703
Page Count:	21

Signature:	Arun Kumar Dasari
Date:	31st January 2022

Efficient Virtual Machine Migration using Apache Kafka Messaging Services and Spring Boot Microservices

Arun Kumar Dasari
X20155361

Abstract

Previously, there has been much research based on the low power usage in data centers. All this research has a lack of data on how the power is reduced in the data centers. So, here the proposed system explains how the migration of virtual machines takes place. While considering the large cloud data centers, using the overall energy consumption, a popular technique is being used and it is called virtual machine consolidation. The maintenance costs in the data centers provided by the cloud providers, Power consumption is one of the most important aspects considered in the cloud-based data center. So, a solution is given in which the Apache Kafka messaging system with spring boot micro-services architecture is implemented in the cloud-based data centers, which reduces the usage of power. By using the spring boot micro services and other efficient dynamic allocation algorithms, it helps handle massive amounts of data. From the experimental results, it is clear that the systems that are using Kafka give higher performance than the systems that are not using Kafka. Experiments have been conducted in order to prove this. One is used in small-scale applications and the other one is used in large-scale applications, and both of these give the same result in that the performance is high for the Kafka-used systems.

1 Introduction

The Internet of Things (IoT) is the consequence of the third phase of the Internet of Everything, which has started to arise. As indicated by Gartner (2014) [1, the Internet of Things (IoT) will become inescapable by 2020, with around 25 billion savvy things giving information. Because of the deluge of bounteous and muddled information created by the Internet of Things, customary information handling frameworks will be transitioned away from being refined into enormous information handling frameworks and stages. Versatile and flexible distributed computing arrangements, which offer considerable computational assets for fast, responsive, and dependable information handling, are needed to appropriately manage the outstanding advancement of IoT created information and client handling related requests. With regards to offering flexible figures, cloud-based server farm offices that store real-arranged PCs and hardware assume a basic part. Such superior execution and quick PC frameworks, for certain, require a lot of energy. According to measurements, the overall energy utilization of server farms has increased by 56 percent in the last five years, between 2005 and 2010, a significant increase over the previous year.According to Gartner (2007) [6, server farms emit 2 percent of total CO2 emissions, which is comparable to the commitment of the flying sector].Energy effectiveness is a top objective in the European Union's energy plans and environmental targets for the years 2020 and 2030.

Reduced datacenter energy use reduces expenses and carbon footprint. A regular disaster evaluation and more efficient electricity generation and transmission are already benefits. Scope quantification and general functional effectiveness tasks require foresight. Developing high performance (IoT and big data) handling server farms is critical for smart computations and IoT-based developments. Moore's law states that the IT industry attempts to boost framework execution by adding elements (Huang, 2014). Though results per watt are improving, the overall power usage of the PC system is improving significantly. Thus, the cost of server energy has surpassed the cost of the physical equipment. These companies are still seeking for ways to reduce their energy costs and carbon footprint. Green computing reduces environmental impacts while increasing performance (Juarez, 2018). The PC foundation business this decade has stressed energy efficiency, customer pleasure, and maintainability (QiS). VMs are predicted to save money and energy. It selects a virtual machine's actual host by minimizing hosts. Enabling ”energy saving” mode on idle servers reduces dynamic figure gadgets (Jumnal and Kumar, 2020). IT expert organizations use a variety of measures to demonstrate registering execution (see list in). The new IBM call recording feature (SLAs). Cloud-based IoT device data management should be scalable. An energy-aware foundation may aid in its development. In this study, distributed computing phases are investigated. In a real private cloud, a study looked at the impact of virtual machine solidification on process power utilization. Then VM methods are evaluated using SLA and energy metrics.

Apache Kafka is the messaging system that is distributed and also highly scalable, which gives an output that ensures high throughput with low latency. Any kind of cloud service that is required to have a messaging system is provided with the Kafka service by the cloud vendors. This defines the system requirement.the configuration of the system which is used in order to provide the Kafka service. By using the queuing model based on the packet flow, the Kafka cloud service performance metrics can be obtained. The configuration of the input parameters used in the Kafka system consists of clusters of Kafka, messages of batch size, and also depends on the number of partitions made in the topic. This helps the user to get the parameters of the configuration, which include the relative payload, throughput producer, and timely change observed in the disk storage.

Log processing is also one of the important processes that use Kafka messaging. The data pipelines of many Internet companies have log processing as an important component. Here, Kafka messaging is used for the development of collection and for giving delivery of the large volume of data logs with very low frequency. The overall idea is implemented by considering the existing log systems and also messaging systems, and this system, which is best suitable for online and offline, is being considered for messaging. By using the Kafka system, the system has been very efficient and even scalable. Kafka has the capacity to process large amounts of data, like 100 gigabytes of fresh data every day. In this way, Kafka messaging is used in log processing.

Apache Kafka is well defined as a scalable messaging system that is publicly subscribed to and well known for its core architecture as it provides the commit logs that are distributed. Apache Kafka was first built and used in the centralised system of LinkedIn for the pipelining platform and for integrating the data online. To use Kafka in different operating systems and to develop it better, the replication of its architecture is being done, which provides a number of applications in the systems that require the distributed messaging system. For many systems which are data-driven, Kafka logs are in use for the purpose of streaming the data and also including the storage of data. Kafka is able to successfully organise the messages in the commit logs and facilitate the pull-based abstraction of messaging, which helps the subscribers in real time to access the offline and online services provided by the organization, like Hadoop or even the data warehouse services. In Apache, Kafka is now the high-level software that is open source. Wang et al. (2015).

Here in this project, it is demonstrated how the migration of virtual machines takes place through communication between the systems, which helps to know the resource availability of the system. If the resource is not available, the migration takes place by using the Apache Kafka messaging system. By using a suitable hardware configuration and the Apache Kafka messaging system, the migration process is successfully observed in the system. This project briefly explains the algorithms used previously and also explains the proposed Kafka messaging system.

1.1 Motivation

The micro service architecture and the service-oriented architecture in the migration of virtual machines contain some conventional methods for software application combinations. This migration of virtual machines is easy when the resources available are defined previously. The architecture of the Apache system is well defined with the structure of the interface between the virtual machines and the micro services that are used in the communication. If CPU utilization is higher and the computing resources are auto-scaled in CPU.The quality of the messaging service in the system is defined by the native applications that could be affected. Auto scaling in large applications is done regularly so as to provide a flexible migration and it is over provisioned. If the resources in the system to which the virtual machines are being migrated are not available,If at any time a resource shortage is observed, then there will be an increase in the cost of the application and architecture. The resources are allocated based on the constant auto scaling of the virtual machines and the algorithms proposed to help in the migration of virtual machines in this research, which has a quality of service based on the Kafka messaging system.

1.2 Research Question

Can an efficient algorithm build the solution for the assignment of virtual machines to the host and is it possible for the virtual machine to be in the sleeping mood by considering the power consumption and data centre's CO2 emissions which can be reduced?

Some of the techniques used to reduce the electricity use, operational cost, and CO2 emissions in the data centers in a cloud environment include resource allocation, virtual machine migration, and virtualization. The various virtual machine migration technologies are used in order to reduce the energy consumption which is utilised by the virtual machines on small server numbers.

While the existing models of energy deal with the processing of energy within, a new model has been developed to address the calculation of energy and server switching from the active modes that stand by. Power consumption can be achieved by implementing the technique of virtual machine migration.

This research topic is a great opportunity to enhance the knowledge and get more information about virtual machines and how they are managed inside the data centers, as well as the non-functioning in the operations, like the cost of maintaining them in data centers.

1.3 Structure of the Paper

The paper's first half is an introduction, while the second describes past research publications and literature reviews. This is the core of the existing and future systems. Section 3 discusses the research and specification design procedures, as well as the methods and tools used for the proposed system's architecture. Later, the evolution is illustrated by using the subsections of hardware configuration performance, performance analysis of Apache Kafka messaging services, and the working of the Kafka system in small-scale applications and large-scale applications. The overall proposed system is discussed in the next section. The study paper cites all the sources used.

2 Related Work

A technique for estimating migration of any virtual machine by construction of cloud servers was developed in 2018 by citing Paulraj et al. (2018). When performing live migration in the context of current and predicted asset utilization, the Combined Forecast Load-Aware method was employed. The observations were finalized in order to test the approach that is being offered for the operation in the current virtual machine migration. Comparing the findings to the present state of the craftsmanship method, the results showed a smaller migration of virtual machines, a lower usage of energy, and a lower number of overhead messages.

According to a Noshy et al. (2018), ”Migration of live virtual machines (VMs) is one of the fundamental components of virtualization since it enables virtual machines to transfer from one region to another without being suspended.” It provides a thorough understanding of live virtual machine migration as well as the principles of the approaches used in this process. It concentrated solely on the investigation of the optimization of the state procedures devoted to increasing live virtual machine migration as well as the migration of memory during the migration of virtual machines. It analyzes and serves, discusses, deconstructs, and evaluates all of the procedures in order to appreciate their issues and optimize them. The project also featured open research topics that were needed for encouragement and were investigated as part of the project. in order to improve the live migration procedure for virtual machines.

In 2017, the citation Bhagyalakshmi and Malhotra (2018) categorizes several methods of Live VM migration. They planned VM migration solutions based on redundancy instruments, paid close attention to the details of the setup, and assessed several live VM migration methods. The authors of this study evaluated numerous execution parameters, such as application service downtime, overall migration time, and the amount of data transmitted. The primary goal of this project was to demonstrate the fundamentals of live VM migration operations as well as an inside-out survey that cloud experts and specialists could utilize to further explore the difficulties and provide appropriate solutions.

Virtualization, according to the citation Ansari et al. (2017), has grown to a recognized scale at the most astonishing levels in each business unit. They proposed a technique for detecting hypervisor attacks in virtual machines, which they believe is feasible. The researchers employed a Bayesian classifier to analyze a publicly available dataset. They used constant numbers to represent the shortcomings of two hypervisors, XEN and VM-ware. A total of three components were considered to be data, and they were as follows: verification, integrity effect, and privacy effect. When they tried, the authors computed the back potential of the weakness, which reveals its level, and used it as a hypervisor attack.

According to Karthikeyan et al. (2020), it has been demonstrated that using Naive Bayes with hybrid optimization can reduce energy usage during VM migrations.Because of this work, it has been proved that it is possible to anticipate VM failures as well as that the results obtained using cloudsim are accurate. The results demonstrate that the proposed strategy outperformed the competition when it came to forecasting VM rejection through cloud server farm environments. Following an examination of the number of failures in VM migration events, it was discovered that the ABC-BA technique has fewer failures in VM migration events than both random migration and ideal migration. The load-mindful migration procedure was used to relocate the data from servers that failed to meet the heap-adjustment criteria to the target servers that did fulfill the criterion. The suggested framework's implementation is evaluated in contrast to previous research and in comparison to the current approach, with success and failure rates, as well as energy consumption, as indications of progress.

It was discovered that by utilizing a bandwidth allocation system that was based on a best-fit policy, the VM placement problem could be solved. Abdel-Basset et al. (2019) looked into this subject. The researchers explained that an energy-efficient solution based on an upgraded version of the whale optimization algorithm was proposed for use in a cloud computing system, the researchers explained. They also used the Cloudsim frame-work to test their approach on 25 sets of data with variable bandwidth and discovered that it reduced the number of active servers when compared to the other meta-heuristics strategies they had previously tested out.

Abdessamia et al. (2020) In an attempt to tackle the VM placement problem in a diversified cloud data center, we offered an alternative solution based on a binary version of the gravitational search algorithm. Furthermore, they employ MATLAB to enforce their own solution, which surpasses best-fit, worst-fit, first-fit, and particle swarm optimization processes in terms of power usage.

Virtual machines can be placed in several virtual machines to reduce power consump-tion, the number of active servers and overall energy waste in cloud computing envir-onments. After defining their problem as a non-linear convex optimization form, they employed a non-dominated sorting evolutionary algorithm to discover the best placement solution for the PMs on the VMs. Finally, they compare their technique to precise com-putation and first-fit reducing policies in terms of power usage, resource loss, and the number of current servers using the Cloudsim tool, and find that it is superior.

Rasouli et al. (2020) presented a learning automata-based technique to deploying virtual machines (VMs) in a data center alongside physical servers. No prior understand-ing of cloud-based apps was necessary for their proposed strategy, which used dynamic migration and pushed idle servers to shut down. Their methodology was also tested on PlanetLab's data set using Cloudsim, which showed significant energy savings while maintaining QoS compared to the current methods.

In order to tackle the Vm allocation problem, Azizi et al. (2020) created a new heuristic-based approach that considers both resource wastage and energy consumption metrics. Their proposed algorithm decreases the amount of wasted resources by requiring physical cloud servers to be used in a balanced manner. Using reward and punishment mechanisms, they also proposed a policy to solve the Vm allocation issue. The results of their Amazon EC2 virtual machine workload simulations show that their proposed method minimizes total energy usage and waste loss in a cloud data center, making it a promising solution for green cloud computing.

Studying multi-objective VM placement methods calls for the application of Donyagard Vahed et al. (2019) recommended environment metaheuristic algorithms. A single-based, population-based, and hybrid Vm allocation technique were categorized. In addition, the researchers looked at VM placement solutions in the form of optimization techniques, minimal resources, assignment types, and environments, in addition to future VM placement research projects.

Using load balancing criteria, Ghasemi and Haghighat (2020) developed a reinforcement study mechanism to address the VM placement challenge.An action can be selected from a list of available actions and executed in the data center using a reinforcement signal based on how well the Vm allocation solution deployed in a virtualized environment performs. The results of their Cloudsim tool simulations reveal that their proposed mechanism outperforms existing strategies in terms of balancing workloads in a shorter amount of time.

Pareto-based reinforcement learning was used by Qin et al. (2020) to reduce energy consumption in the cloud. According to their solution, they employed the Chebyshev function and the weight selection problem as a Pareto approximation to solve the Vm placement challenge. When they ran the code through MATLAB, they were able to demonstrate that it could handle large VM queries.

Wei et al. (2020) created Virtual machines (VMs) are allocated in cloud-based systems based on the number of calculations they require. The researchers used First-Fit, Greedy Heuristics, and Best-Fit Strategies were used by the researchers to solve a three-dimensional bin-packing mixed integer linear program in order to minimize energy usage. Furthermore, they use the Gurobi solver to evaluate their technique in real-world datacenters and discover that by reducing the number of virtual machines, they can achieve linear execution time.

Hybrid model suggested by Abohamama and Hamouda (2020) combines an enhanced genetic algorithm with the best fit allocation method in order to improve energy efficiency in cloud data centers. In order to reduce the amount of resources wasted, they propose a method that takes into account the trade-off between exploration and exploitation, as well as the RAM, bandwidth, and CPU utilization on cloud servers.

Reddy and Ravindranath (2020) created an effective VM allocation approach to decrease data center power consumption. They employed the JAYA optimization technique to find the most effective solution for the task. They also found that their proposed approach minimizes SLA violations, power usage, and VM migration when compared to the modified best fit decrease.

Even though all this literature works to explain the virtual machines and their behavior, It is not clear how the virtual machines actually communicate and undergo migration. Migration is necessary for the virtual machines to find the services that have all the resources. All these innovations describe how the virtual machine is being optimized and how work serialization takes place and the procedure followed during the migration, as well as the drawbacks and other systems that use the migration technique that use it and some of the authors have discussed the virtual machine placements in different cloud centers of data and they have implemented and explained the power consumption, resource wastage, scalability, and energy efficiency systems. But in all these literature works, it is not discussed the implementation part of how it works, and here it is explained how the actual migration takes place. So, to overcome these drawbacks, a system is proposed that is the Apache Kafka messaging system, which explains the complete process of evaluation and migration of the virtual machines and gives a detailed explanation of how the virtual machines communicate with each other in order to find the available resources. So this paper explains the complete details of the algorithms which are used in the process and hardware configurations which are not mentioned by many of the authors, and also the complete workflow of the Apache Kafka messaging service.

3 Methodology

The resource allocation cost and energy usage while providing the cloud services always remains critical continuous services are being provided by the cloud services to increase and attract a lot of customers this section gives a brief about how the proposed methodology have been implemented. Energy efficiency is consolidated and dynamic load balance between servers through the migration of virtual machines between host systems are ensured order automatically in the proposed system allocation algorithms which are already existing are improved. Before the migration of virtual machine from source host to the destination host it ensures double checking.

3.1 Tools and Methods for the proposed system

Spring boot is used for writing tasks that are computationally intensive or performed by supporting the microservice-based application and rest API.

For the completion of the research and for employing different tools and technologies used in the system, including the following:

Cloudsim Simulator:

Data centers have several entities that contain storage area networks and virtual machines, and both of these use Cloudsim plus. Entities are exposed to a wide range of workloads by considering the behaviour of the postulates in Cloudsim. There are some specified users who describe the characteristics of the entities, which include the processing elements, storage, RAM, and other entities. Fundamental simulation capabilities are available in a cloud-based system environment.Cloudsim Plus also figures out the unique capabilities and offers virtual machine scaling, horizontal scaling, vertical scaling, and event listeners.

During the development of a proposed research algorithm, the characteristics are considered for the foundation and development. With respect to this Cloudsim Plus framework, which is written in the Java programming language and the algorithm chosen In the below section, the Cloudsim Simulator framework implementation and evaluation of the algorithm are described. Once the creation of the algorithm is done, the cloud information service registration is present in all of them. In addition to this, there is also the creation and registration of other entities, like in the construction of a data center. When this procedure is completed, The registry containing the information for cloud information services is being created. This information is being registered under CIS. is constructed in a data center. With the help of different hardware characteristics, which include the storage number of processing units, RAM, and bandwidth referring to the hosts and their capacity, the data center is set up. The complete list of all the hosts present here is then moved to the data center as part of the parameters. This helps in the creation of a data center. Several virtual machines that have specific configurations are made available on these hosts, and they are virtualized in the infrastructure of the cloud.

With algorithms, you can control anything from virtual machine configurations to network throughput, storage space, and RAM. In different settings, the number of hosts and the number of virtual machines are observed. It is possible to create an unusual feature that checks the CPU consumption every few seconds for overloading and then provides the particular number of virtual machines that meet the condition. While virtual computers, cloudlets are assigned based on their association with a space-sharing scheduler. The cloudlet has a capacity of around 10 million instructions. Allotment to a cloudlet, or virtual machine, is coupled with a space-sharing scheduler. Using the method setVmDestructionDelayFunction(), you can delay the destruction of virtual machines by a predetermined amount of time (seconds). It's not necessary to wait for the cloudlet to be assigned a time period, including 5 seconds after the virtual version has completed the execution of all cloudlets.

SpringBoot:

The web service design approach, which is a Rest and uses simple HTTP calls

allowing the clients to consume the service. The development of the Rest API service is easily done with the spring boot. A part of the API structure is the spring boot application. Allowing for the easier management dependencies and for the auto configuration, it has built-in CRUD handling and the project flexibility of the startup options.

Apache kafka messaging:

A high-throughput, publish-subscribed distributed messaging system, Apache Kafka, is used and it uses the publish-subscribe model. At Kafka, hundreds of MB are handled by one broker per second in the reading and writing operations, which are taken from multiple clients, making Kafka a maximum-efficiency service. The disk contains the messages that are written and persisted across the cluster. Window stream processing with Kafka can also be used for website activity tracking and also for matrix collection and monitoring with log aggregation and log arriving.

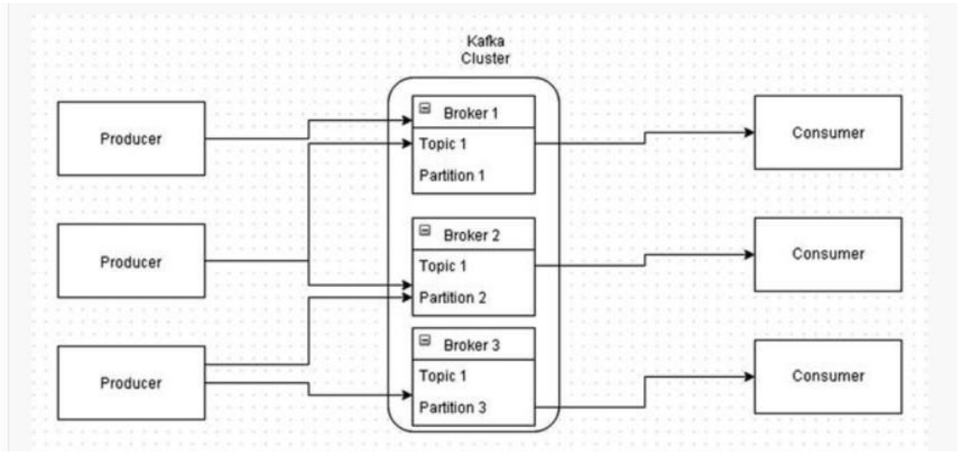

Figure 1: Kafka architecture

3.2 Proposed Architecture

The microservice architecture is being used in the implementation of this project rather than a monolithic architecture. Because when compared to monolithic architecture, microservices offer a number of distinct advantages. When an application is broken down into tiny, self-contained chunks, it is very simple to design and maintain. If necessary, each of the services can be developed, deployed, and managed independently. They can be developed using any of the computer programming, technological aspects, and software environments that are available. Any type of problem can be resolved with relative ease. Each of the four critical steps in the suggested system model is detailed below.

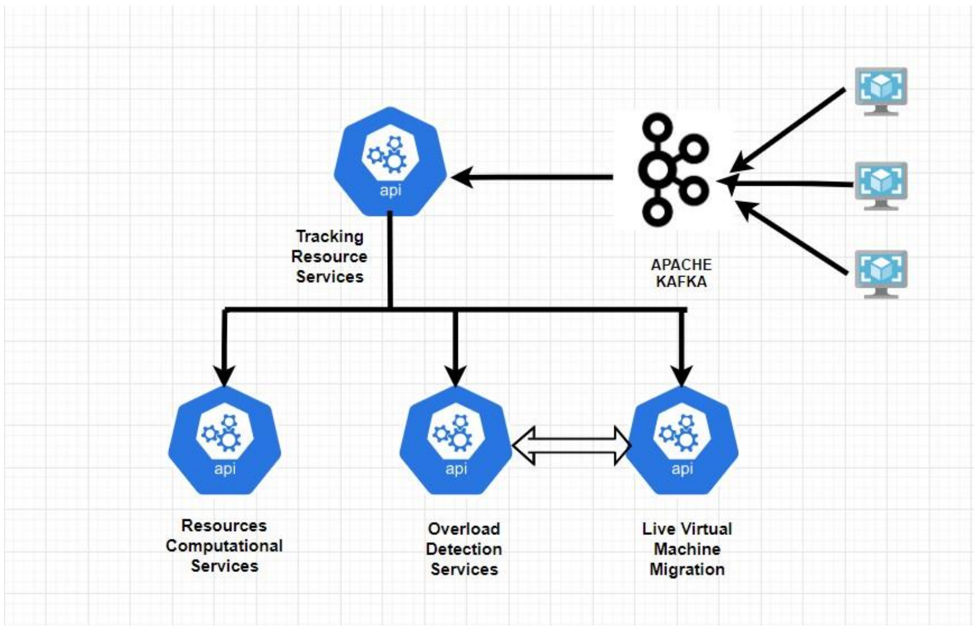
api
Tracking Resource Services
APACHE KAFKA
api
Resources Computational Services
api
Overload Detection Services
api
Live Virtual Machine Migration

Figure 2: Project Architecture

There are four important steps in the suggested system model.

Tracking Resources Services:

To begin with, keeping track of all of the resources in this system is a time-consuming and complex undertaking. Every two seconds, the amount of time that has been spent on each resource is recorded. For optimal resource consumption, all compute nodes are fetched at regular intervals of time in order to maximize resource utilization. It is the system's responsibility to retrieve information about the resource use of all of the compute nodes on a regular basis. The data that is fetched comprises a range of metrics, such as the amount of RAM that is being used, the amount of CPU time being used, and the amount of disk space being used.

Underloaded Detection Services:

In addition to being an incredibly vital service, it also helps to save money on expenses. The major goal of this service is to identify compute nodes that have exceeded a certain threshold value; once they have exceeded the threshold value, the resource is shut off to prevent further damage. As a result, there is a conservation of energy.

Overload Detection Services:

This service is being used in order to ensure that work is spread evenly among all of the servers on the network. Hosting overload detection is used to do this, and it determines whether a particular compute node is overloaded or whether a few virtual machine variables that are available on the compute node in question are not being used by the compute node in question.

Live Virtual Machine Migration Services:

LVM destinations are typically determined by using the Ant Colony Optimization approach, which identifies the compute node that is closest to the current host as the LVM destination by default. However, it can be noted that the virtual machine may cause the destination host to become overwhelmed if the migration is followed closely enough. In this case, it should be possible to determine whether the host is overloaded or not after migrating the previous migrating virtual machine to the destination host, as described above.

It is in charge of the migration of these services, which is accomplished through the implementation of a bespoke virtual migration algorithm. Knowing the CPU utilization and the condition of virtual machines in data centers are two of the most important responsibilities in the industry. It is possible that the data will be lost in the existing system. Consequently, to resolve the issue, we are utilizing Apache Kafka messaging services to do so. By providing a publish-subscribe methodology, this messaging service assists in the transmission of messages from the producer to a specific topic in the Kafka cluster, which is referred to as the messaging brokers, and from that topic, which is being subscribed to, it can be determined whether or not a consumer is reading the messages transmitted.

The producer computes CPU utilization and then sends the results to Kafka brokers for further processing. It is defined what topic is being tracked, and tracking resource services are subscribing to this specific topic on the other hand. After that, the messages sent by the producers are ingested and processed. Because of the replica that is maintained by the brokers, there is no data loss at all. The zookeeper is in charge of keeping track of the state of the messages and brokers.

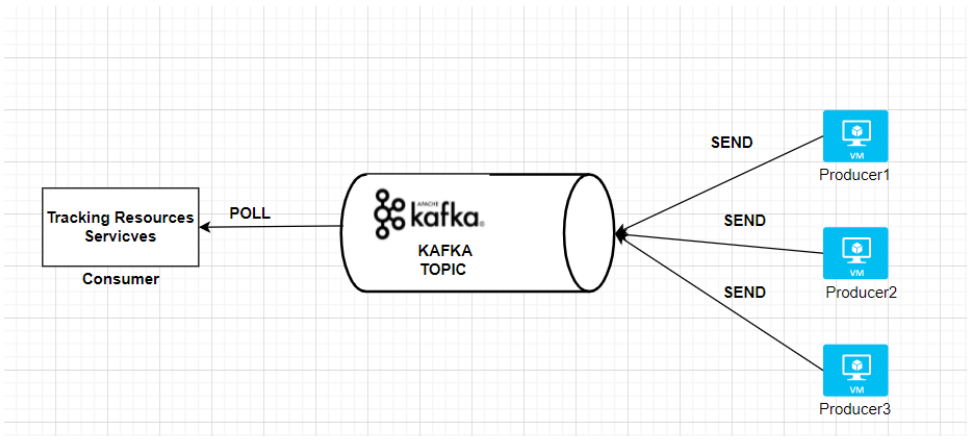

Figure 3: Kafka Messaging work flow

4 Design Specification

The techniques and/or architecture and/or framework that underlie the implementation and the associated requirements are identified and presented in this section. If a new algorithm or model is proposed, a word based description of the algorithm/model functionality should be included.

5 Implementation

Using the Apache Kafka messaging system, algorithm 1 describes and monitors the resources monitoring which gets the host data from different hosts. For the total CPU, RAM, and virtual machine usage, each host CPU Utilization, RAM utilization, and VM usage are all combinedly calculated for each instance, and the total is being considered. Many applications that are based on machine learning and have the need to convert a real number to a probability can benefit from the sigmoid function. A sigmoid function is required for the total computational node, which has the probability value. Initially, the threshold value is defined as 0.5. If the value crosses this threshold value, then there is a need to change the status of the host, which might be overloaded. If the value is less than the threshold value, then it is not required to change as it doesn't get overloaded.

Algorithm 1 Resource Tracking Algorithm

```
1: Input: datacenterList
2: Output: categoryBasedHost
3:
4: slaLimit=0.5
5: for $\mathrm{hostobj} \in \mathrm{datacenterList}$ do
6:     cpuUtilization=getingCPUUtilization(host.getId());
7:     totalCpuUtilization=cpuUtilization;
8:     pbty=1/(1+pow(e,totalCpuUtilization))        ▷ finding the overloaded host and not
   overloaded host is done b with the help of sigmoid function
9:     if $\mathrm{probability} \geq \mathrm{slaLimit}$ then
10:        StatusOfHost.Status=OverloadedHost
11:    else
12:        StatusOfHost.status=UnderloadedHost
13:    end if
14: end for
```

Algorithm 2 describes the migration of a version machine by considering the amount of electricity that can be saved. The SLA threshold, which is specified by the SLA agreement, is considered. The utilization of a host is defined by the HostUtilizationList. If the host usage drops below the SLA threshold, all the virtual machines are moved to the resource-available host and the server is turned off. If not, the load balancing is not going to perform.

Algorithm 3 explains how the overload is detected by considering factors like CPU, RAM, and the number of virtual machines present on the hosts. The main function of this algorithm is to check whether the given host is overloaded or not. Migration is done if the host is overloaded, or else not overloaded is printed. Then it will move on to the next iteration.

Algorithm 4 explains how to track the location of the source virtual machine of the overloaded host and the target host to undergo migration for balancing the load between the servers. One or more virtual machines (VMs) from the set of virtual machines are hosted by each physical machine (PM). More importantly, in the context of VM migration,

Algorithm 2 Algorithm for Underloaded Hosts

```
1: Input: categoryBasedHost
2:  Output: listOfUnderLoadedHost
3:
4: slaLimit=0.5
5: for hostobj $\in$ categoryBasedHost do
6:     if hostCPUUtilization $\leq$ slaLimit then
7:         After that, move the VM to any available host and shut down the server.r
8:     else
9:         Keep the server running.
10:    end if
11: end for
```

Algorithm 3 Overloaded Detection Services

```
1: Input: categoryBasedHost
2:  Output: listOfOverLoadedHost
3:
4: slaLimit=0.5
5: for hostobj $\in$ categoryBasedHost do
6:     Find out the most recent information on the host's configuration..
7:     A data frame with the current values of the Host can be created.
8:     You can predict the outcome with the help of an SVM model.
9:     if hostCPUUtilization $\geq$ slaLimit then
10:        Afterward, Migaration must be completed.
11:    else
12:        There's no need to move.
13:    end if
14: end for
```

each PM is a potential source PM for the VMs that are already residing on that particular PM's hardware. This means that the proposed ACS algorithm generates a set of tris T, where each tris t is composed of three elements: the source PM, the virtual machine (VM) to be transferred (v), and the final destination PM. The consolidation algorithm's computation time is based on the number of tuples. The algorithm reduces the set of tuples to T by removing some of the least important and obsolete tuples.
The first constraint ensures that only overloaded or underloaded PMs are used as source PMs.It also ensures that none of the overloaded P and predicted overloaded PMs become destination PMs.The third constraint prevents VM migrations to sleeping hosts unless the source hostis overloaded.

We are looking for a migration plan that would result in a minimal number of active PMs being required to host all VMs without compromising their performance. The VM consolidation problem does not have a path like the TSP. Because of this, the pheromone is deposited on the tuples in our method. A stochastic state transition rule is used by each nA ant to select the next tuple to traverse. The pheromone value of each tuple is set to 0 using the algorithm above. nI iterations are used in the algorithm. The nA ants build their migration strategies in parallel during each iteration. All the tuples are traversed by each ant. According to formula (6), it selects the next tuple and adds it to the local migration plan at a given time. Machine capacity vectors are updated using the local pheromone trail update rule. Migration plans and global scores are altered when the ant's local migration plan yields a better score than its previous best score. In the end, the best global score and the corresponding migration plan are chosen after all theants have traversed their paths.

Figure 4 represents the completed work of the project and its flow. By default, the first fit technique is used to identify the upcoming compute node after the current host as the virtual machine's destination. At any point, if there is a chance of a host being overloaded, the overloaded host is being migrated to the available resource, and since it mainly considers the status of the host after migration, it must be ensured that it won't be overloaded after the migration. This methodology is being proposed. Initially, the most CPU-intensive virtual machine is selected to migrate, and the overall parameters are verified across all the existing compute nodes. The host, which has high resources, is first verified to check whether the virtual machine can be migrated again or not, and then the parameters of both the host to be migrated and the virtual machine are added. The compute node is often considered for detecting if it is overloaded, and if it is not overloaded, then the virtual machine is moved to the target host.

Algorithm 4 Live Virtual Machine Migration Services

```
1: Input: OverloadHostList[]
2: Output: getting allocated map
3:
4:  OptimizedgolbalMigartionPlan=0,golbalMigartionPlan=0
5: for i $\in$ numberOfItteration do
6:     for k $\in$ numberOfAnts do
7:         temporayMigrationMap=0,migratingMap=0,currentBestScore=0
8:         for tuple $\in$ SetOfTuple do
9:             generate a random variable randomQ with a uniform distribution between
           0 and 1
10:            if randomQ $\geq$ q0 then
11:                compute sourcesPM
12:            end if
13:            choose a tuple t to traverse by using 8
14:            migratingMap= migratingMap U tuple updating local Pheromone
15:            if f(temporaryMigratingPlan) $\geq$ currentBestScore then
16:                currentBestScore=scoringTempaoryMap
17:                migratingMap= migratingMap U t
18:            end if
19:        end for
20:        golbalMigartionPlan=golbalMigartionPlan U migratingMap
21:    end for
        OptimizedgolbalMigartionPlan+ =arg max of migratingMap $\in$
       golbalMigartionPlanf(migratingMap)
22: end for
```

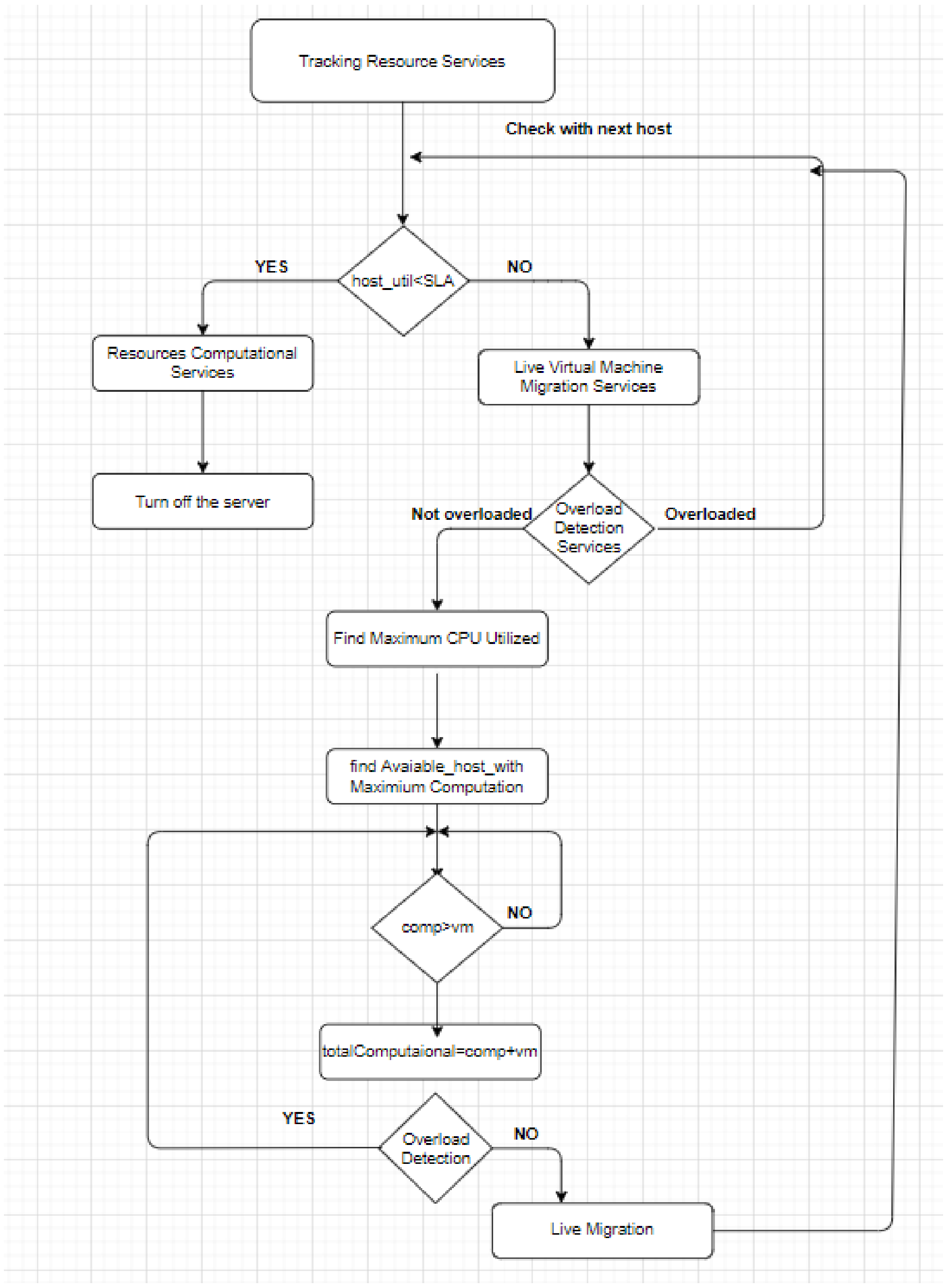
Tracking Resource Services
Check with next host
YES
host_util<SLA
NO
Resources Computational Services
Live Virtual Machine Migration Services
Turn off the server
Not overloaded
Overload Detection Services
Overloaded
Find Maximum CPU Utilized
find Avaiable_host_with Maximium Computation
comp>vm
NO
totalComputaional=comp+vm
YES
Overload Detection
NO
Live Migration

Figure 4: Entire project flow diagram

6 Evaluation

6.1 Hardware Configuration of the proposed system

Protocols can be tested using a variety of methods, including experimentation, analysis, and simulation. The experimental approaches are the most powerful, but they are also the most expensive and difficult to set up. In some cases, an expert must be used to set up the testbeds for experiments, which can cost a lot. Furthermore, running the simulation multiple times on the real cloud platform may result in a high monitoring cost, whereas the analytical approaches are frequently limited in their ability to evaluate migration approaches in a timely fashion. Consequently, simulation approaches are widely used for performance evaluation of underlying migration techniques, which can be carried out in a variety of configurations depending on the situation. The Cloudsim simulation platform is used to evaluate the performance of various migration approaches, including the proposed Live VM Migration with Apache Kafka Messaging services. The simulation experiments were carried out on a workstation using the hardware configuration shown in the following table.

Table 1: The following table shows the hardware configuration for our proposed environment

Components	Specific Parameter
CPU	Intel i5 core
Cores	5
RAM	8GB
Storage	512GB

6.2 Performance review

This section gives an overall analysis of the prepared system by considering the parameters like accuracy, performance, reliability, and efficiency. In this scenario, small and large-scale Kafka implementations are examined, as well as data centers that do not use Kafka at all on a large or small scale. Managed by Apache Kafka, this system’s performance is evaluated. A cloudlet’s evaluation time is used to determine the system’s effectiveness. It is a term used to describe a computer with a large number of resources or a cluster of computers connected to the Internet and accessible to nearby mobile devices. One way to think of this new cloud infrastructure is that it’s an extension of the traditional cloud computing data centers. The work can be delegated to any high-bandwidth, low-latency data center located anywhere in the world using either cloudlets or micro data centers. By reducing operational costs, it makes cloud computing more local. As a result, Cloudlet can provide and deploy virtual machines and containers derived from computational resources via containerization or virtualization. End users in the cloud can take advantage of cloud computing management techniques such as on-demand business models, pass-as-you-go, scalability, availability, and even energy efficiency, once virtual machines and containers have been created and deployed on physical servers.

6.3 Performance analysis of our Apache Kafka messaging producer service

By conducting an experiment, the basic functionality of the Apache Kafka messaging service is obtained. The Kafka messaging service uses resources by scaling when the workload is raised and must be stabilized. As the workload rises, CPU utilization also seems to be high. So, for the experiment conducted, the utilization of CPU and the virtual machine allocated are compared to the cloudlet number, which is running at a regular time interval. The simulation is set in a configuration of an application that is small-scale as well as large-scale.

6.4 Comparison of Kafka stream API with other messaging services

This section gives a review of how the frameworks have been evaluated over time, from traditional to the proposed Apache Kafka Messaging service for the migration of virtual machines. Based on the algorithm of stream processing, the operations are mainly classified into 2 types, namely stateless and stateful. In the operations of stateless, it is observed that the inputs are taken one at a time, which is called input data, and the last taken input is given the output. While in the case of stateful operation, it maintains the data point values so far processed and for the incoming data points, the values are updated with every new input. So, the output is clearly based on the previous and new inputs as well. The Kafka platform accommodates the state stores to manage the state of that application. Taking an example of Kafka streaming, the application developed with the Kafka API will automatically work to create and manage the stateful operations. Kafka is very helpful in providing the recovery of local state stores automatically by the synchronization of the local Kafka messaging state, which comes from middleware and which is the main source of the incoming data and the result that is the output destination. Considering the deployment process Kafka APIs can be deployed in a way that the APIs are embedded within the application, but it doesn't have a particular deployment method. Whereas in traditional frameworks, applications are deployed in standalone clusters or through the use of YARN, containers, or Mesos.When stateful and fault tolerance are considered, Kafka provides the local state stores and helps in synchronization with all the Kafka message queues, which are in native, but in the case of other frameworks, the state store is in local only, whereas the fault tolerance can be done by using configuration of external devices for storage like HDFS. The main source of the data streaming is received only from the queue of Kafka messaging, and it supports the API's, namely the producer API, connect API, and the consumer API in Kafka, to address the data problem from the other systems. In other frameworks, it includes many types of streaming data sources, including Kafka, message queues, and file systems. Kafka is the only unbounded data stream, whereas others can be both unbounded and bounded data streams.

6.5 Experimental Results and Comparison of Proposed system with Kafka and systems without Kafka in Small Scale Applications

Basically, in small-scale applications, comparisons are made between systems that have adopted Kafka and those that have not. It is analyzed by considering some of the factors like the number of cloudlets and, based on the execution time, it can be measured. for a small-scale application consisting of 3 cloudlets. If the system is using Kafka, it takes about 26 milliseconds to run the system and it is called the execution time in which output is obtained. When the same system is executed without the Kafka, it clearly shows that the execution time is delayed, and it is 39 milliseconds. In the case of the application consisting of 5 cloudlets, the execution time when using Kafka is 30 milliseconds and when not in use is about 80 milliseconds. If the application has 8 cloudlets, then the time is 40 milliseconds with Kafka and 55 milliseconds without Kafka, whereas with 12 cloudlets, it is obtained in 47 milliseconds using Kafka and 65 milliseconds by not using Kafka. With Kafka, 16 cloudlets took 58 milliseconds to execute, but without Kafka, it took 120 milliseconds. Finally, when 20 cloudlets are considered, it gives a major difference, like 70 milliseconds by using Kafka and 150 milliseconds by not using Kafka. These are clearly shown in the table below and in the given graph.

Table 2: The following table shows the hardware configuration for small scale application

Entity	No of Entities	RAM	Bandwidth	No of PEs	PE capacity
Host	5	7000	12000	40	1000
VM	10	512	1000	2	1000

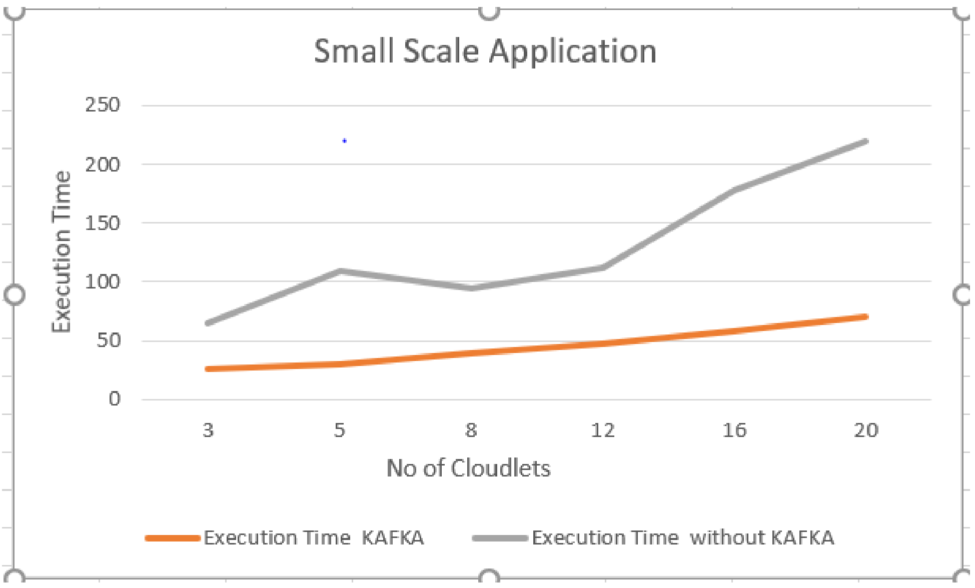

Figure 5: Total Execution time comparison with kafka and without kafka

6.6 Experimental Results and Comparison of Proposed system with Kafka and systems without Kafka in large Scale Applications

In the case of large-scale applications, by considering the same parameters as cloudlet numbers and the time of execution, the efficiency of the system can be seen. The hardware configuration of the considered system has 8 GB of RAM and 80,000 megabytes of bandwidth. Taking this hardware into consideration, the experimental results are shown in the table.

Table 3: The following table shows the hardware configuration for large scale application

Entity	No of Entities	RAM	Bandwidth	No of PEs	PE capacity
Host	25	8000	80000	120	1000
VM	10	2000	10000	12	1000

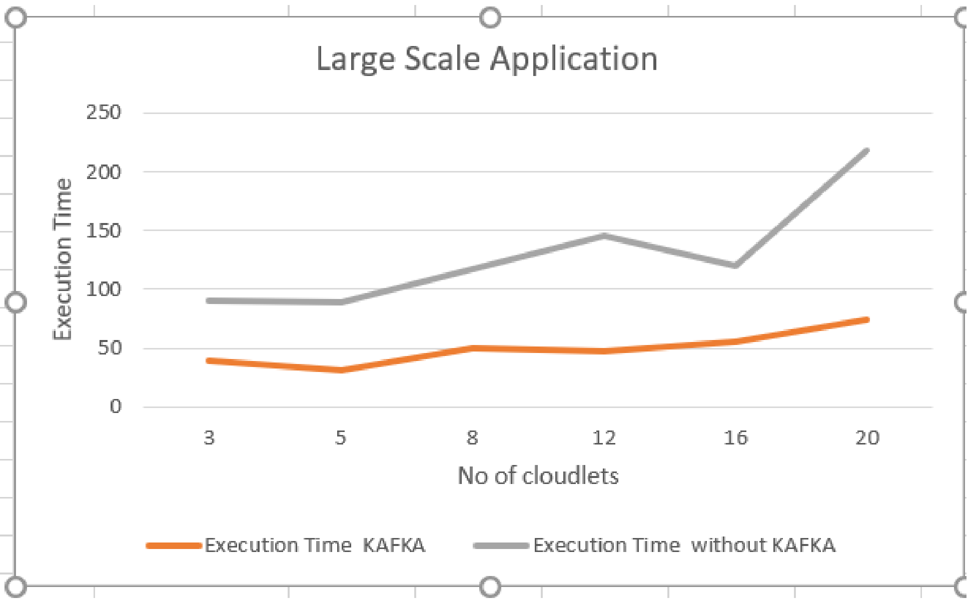

Figure 6: Total Execution time comparison with kafka and without kafka

From the above table, it is noticed that for large scale applications also, the cloudlets take a lot of time to execute without the Kafka, and usually it takes a little more time for large scale applications, like for the system having three cloudlets, it will take 40 milliseconds without the Kafka, and so on up to the experimental results for 20 cloudlets, which consist of a higher difference between 74 milliseconds to 144 milliseconds with using and not using Kafka.

The graph represents the execution time of the large-scale application where the X axis shows the number of cloudlets and execution time is represented on the Y axis. The graph represents the same entities for the system which is not using Kafka.

6.7 Discussion

This section briefly explains the analysis of the small-scale system execution and the large-scale system, which takes different execution times for running the process with the Kafka messaging and Apache Kafka services, which helps in the migration of virtual machines in an efficient way by considering the power usage in the data centers, so the maximum utilization of the given data centers is used by using the first fit algorithm. When comparing the experimental results of the system using Apache Kafka messaging services and without, there is a major difference in execution time. It is important to consider the execution time of the data center because of the efficiency factor that should be reached to reach the target. As discussed in sections 5.4 and 5.5, by considering the experimental values in both the applications, that is, on a large scale and a small scale, The results clearly specify that the execution time of the system in small-scale applications and the time of execution in large-scale applications is always high before the implementation of Apache Kafka messaging services, in which it is difficult to reach the target within the specified time, leading to the effective migration of virtual machines. So, to overcome this disadvantage in the data centers, it is required to implement a service that helps to improve the efficiency and decrease the execution time of large-scale and small-scale applications. Therefore, the proposed system efficiently overcomes this challenge and reduces the execution time in such a way that the system is executed very fast using the Apache Kafka messaging services. This is proved with all the experimental results which are obtained and show the efficiency of the Apache Kafka messaging services.

7 Conclusion and Future Work

We described a live VM component for dynamic VM consolidation that takes SLAs into account. We used Ant Colony Optimisation and Apache Kafka Server to get a near-optimal migration plan because of the problem of VM consolidation. This proposal also minimized SLA violations and the number of migrations when compared with other VM migration algorithms. It is therefore possible to use this method to promote green cloud computing in real data centers.

In our future work, we will continue to reduce the complexity of our algorithm in order to make it more suitable for handling larger workloads. We aim to utilise the power of machine learning to forecast the approximate load on the datacenter in the near future and, as a result, turn off some sleeping hosts in order to save even more electricity.

References

Abdel-Basset, M., Abdle-Fatah, L. and Sangaiah, A. K. (2019). An improved lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment, *Cluster Computing* **22**(4): 8319–8334.

Abdessamia, F., Zhang, W.-Z. and Tian, Y.-C. (2020). Energy-efficiency virtual machine placement based on binary gravitational search algorithm, *Cluster Computing* **23**(3): 1577–1588.

Abohamama, A. S. and Hamouda, E. (2020). A hybrid energy–aware virtual machine placement algorithm for cloud environments, *Expert Systems with Applications* **150**: 113306.

Ansari, S., Hans, K. and Khatri, S. K. (2017). A naive bayes classifier approach for detecting hypervisor attacks in virtual machines, *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*, pp. 1–6.

Azizi, S., Zandsalimi, M. and Li, D. (2020). An energy-efficient algorithm for virtual machine placement optimization in cloud data centers, *Cluster Computing* **23**(4): 3421–3434.

Bhagyalakshmi and Malhotra, D. (2018). A critical survey of virtual machine migration techniques in cloud computing, *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pp. 328–332.

Donyagard Vahed, N., Ghobaei-Arani, M. and Souri, A. (2019). Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: A comprehensive review, *International Journal of Communication Systems* **32**(14): e4068.

Ghasemi, A. and Haghighat, A. T. (2020). A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning, *Computing* **102**(9): 2049–2072.

Karthikeyan, K., Sunder, R., Shankar, K., Lakshmanaprabu, S., Vijayakumar, V., Elhoseny, M. and Manogaran, G. (2020). Energy consumption analysis of virtual machine migration in cloud using hybrid swarm optimization (abc–ba), *The Journal of Supercomputing* **76**(5): 3374–3390.

Noshy, M., Ibrahim, A. and Ali, H. A. (2018). Optimization of live virtual machine migration in cloud computing: A survey and future directions, *Journal of Network and Computer Applications* **110**: 1–10.

Paulraj, G. J. L., Francis, S. A. J., Peter, J. D. and Jebadurai, I. J. (2018). A combined forecast-based virtual machine migration in cloud data centers, *Computers & Electrical Engineering* **69**: 287–300.

Qin, Y., Wang, H., Yi, S., Li, X. and Zhai, L. (2020). Virtual machine placement based on multi-objective reinforcement learning, *Applied Intelligence* **50**(8): 2370–2383.

Rasouli, N., Razavi, R. and Faragardi, H. R. (2020). Epbla: energy-efficient consolidation of virtual machines using learning automata in cloud data centers, *Cluster Computing* **23**(4): 3013–3027.

Reddy, M. A. and Ravindranath, K. (2020). Virtual machine placement using jaya optimization algorithm, *Applied Artificial Intelligence* **34**(1): 31–46.

Wang, G., Koshy, J., Subramanian, S., Paramasivam, K., Zadeh, M., Narkhede, N., Rao, J., Kreps, J. and Stein, J. (2015). Building a replicated logging system with apache kafka, *Proceedings of the VLDB Endowment* **8**(12): 1654–1655.

Wei, C., Hu, Z.-H. and Wang, Y.-G. (2020). Exact algorithms for energy-efficient virtual machine placement in data centers, *Future Generation Computer Systems* **106**: 77–91.