Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.034775
Article

Tech Science Press

A Proposed Architecture for Local-Host and AWS with Multiagent System

Jaspreet Chawla[1,*] and Anil Kr Ahlawat[2]

[1]Department of Computer Science & Engineering, JSS Academy of Technical Education, Noida, Affiliated to AKTU, Lucknow, 201301, India
[2]Department of Computer Science & Engineering, KIET Group of Institutions, Ghaziabad, Affiliated to AKTU, Lucknow, 201001, India
*Corresponding Author: Jaspreet Chawla. Email: jaspreetkaur@jssaten.ac.in
Received: 27 July 2022; Accepted: 04 November 2022

Abstract: Cloud computing has become one of the leading technologies in the world today. The benefits of cloud computing affect end users directly. There are several cloud computing frameworks, and each has ways of monitoring and providing resources. Cloud computing eliminates customer requirements such as expensive system configuration and massive infrastructure while improving dependability and scalability. From the user's perspective, cloud computing makes it easy to upload multiagents and operate on different web services. In this paper, the authors used a restful web service and an agent system to discuss, deployments, and analysis of load performance parameters like memory use, central processing unit (CPU) utilization, network latency, etc., both on localhost and an Amazon Web Service Elastic Cloud Computing (AWS-EC2) server. The Java Agent Development Environment (JADE) tool has been used to propose an architecture and conduct a comparative study on both local and remote servers. JADE is an open-source tool for maintaining applications on AWS infrastructure. The focus of the study should be to reduce the complexity and time of load performance parameters by using an agent system on a cloud server instead of establishing a massive infrastructure on a local system, even for a small application.

Keywords: Amazon web service; jade; local host; rest web service

1 Introduction

Monitoring resources and applications in cloud computing are critical for cloud clients and providers. Several ways are provided for customers to monitor their apps and keep resources to ensure that objectives are completed appropriately. In the cloud computing paradigm, customers don't have to own and run the physical infrastructure for their business needs. Also, customers don't have to worry about how or where the work will be done. They only pay attention to how much it costs to use the resources, what services consumers can get from cloud providers, and how good the services are. The cloud providers are the ones who manage the resources being used and given out in the best way possible. The cloud does this by using efficient methods to monitor and manage resources and give the best quality of service without breaking the service-level agreement (SLA). Over the past five years, cloud computing has been considered and deployed in response to a large-scale computer infrastructure to develop and market service-oriented commercial solutions to the industry's requirements [1]. Agent-based cloud computing needs to be built to better manage cloud resources and services and effectively find and put together agent-based solutions.

Infrastructure as a Service (IaaS) is a cloud layer from which users can select virtual machines (VMs) with various configurations and capacities. Nowadays, several companies, like Amazon EC2, IBM SmartCloud, Aneka, Microsoft Azure, and Google App Engine, provide platforms and infrastructures for cloud computing [2]. AWS is the latest technology in the IT cloud industry, with a scalable and reliable platform for on-demand application deployment. The user builds services with minimum maintenance and organizational expenditures. Amazon's business model offers On-Demand, Reserved, and Spot Instances. Users access anything from any computer system without worrying about infrastructure, cost, security, or storage management. AWS is a technology that solves all the issues that come up with cloud computing, such as data security, choosing the right cloud, monitoring in real-time, unauthorized access, etc [3].

AWS provides two-layer security using Multi-factor Authentication (MFA) on username and password during user sign-in. It also contains many registered services, such as storage and content delivery network (CDN), database, analytics, computing, network, deployment and management, and a lot of app services. The EC2 service is a sub-part of the computational and network services that are best suited for cloud computation capacity [4]. EC2 has many benefits, such as autoscaling, virtual private clouds, elastic block stores, elastic load balancing, Internet Protocol (IP) addresses, route services, and high-performance computing clusters [5].

Amazon EC2 accelerates the process by generating new instances in minutes, allowing you to scale up and down services as needed [6]. Application deployment in Amazon EC2 differs based on various standards, application types, components utilized, etc. The main feature of EC2 is that it allows you to pay as you use the cloud setup. Customers can choose from various memory instances, including free and paid instances [7]. To check the CPU utilization and memory usage of the AWS-EC2 instance (Remote server) and local server, a restful-based web service, also known as an application programming interface (API), has to be created. The multiagent system will call this API when the system load is calculated. The performance of the load utilization parameters will be checked first on the local host and then on the AWS server [8].

A multiagent system is a decentralized method where all the agents work together in an agent container. Agents are heterogeneous, scalable, and distributed to achieve the goals established during the requirement and design phases. Agents have different responsibilities depending on their usage, but they all follow agent properties such as atomicity, integrity, message transport, sharing, adaptive, rational, cognitive, and many more. The Foundation for Intelligent Physical Agents (FIPA) standards are met when JADE is used to build the multiagent system and applications [9].

Autonomous agents build clouds more intelligently and effectively by allocating resources to applications and communicating with users. It is also critical in clouds to develop and implement approaches and strategies that consider the dynamic nature of cloud computing systems. JADE helps in integrated programming, which means that JADE can create code and databases within a single development environment. This structure of JADE makes it easier for the programmer to design, build, and show how the whole model works all in one place.

Agents use an ACL (Agent Communication Language) for sending and receiving messages between them. The main advantage of JADE is in the implementation part. Instead of a program's code being compiled simultaneously, each method is compiled as it is completed. JADE provides an API for languages such as .NET, Java, C/C++, and web services. Consumers can create agents or use the cloud to resolve challenges [10]. An agent is programmed as an object that works well with other objects to accomplish a job. Even the agents help to provide the best match among cloud providers. Adding a multiagent system to the cloud makes it more flexible, automatic, and high-performing. This paper uses JADE as a middleware to send and receive system information results, first between the local host server and the display app and then between the AWS server and the display app.

The rest of the paper is organized as follows. Section 2 illustrates the structure of agent services in cloud computing. Section 3 describes the proposed local architecture with JADE agent; Section 4 explains the proposed AWS architecture with JADE agent; Section 5 discusses the comparative analysis of system load on the local host and AWS server; Section 6 includes the benefits of cloud computing in engineering applications; Section 7 includes discussion, and Section 8 is the conclusion of the paper.

2 Related Works

The cloud platform incorporates diverse functionalities, such as reasoning and learning. However, client management is becoming increasingly difficult as the demand for cloud computing rises. Very few studies have been conducted on agent-based cloud computing. Agents and cloud computing are complementary technologies. Agent systems can benefit from the improved computing qualities of cloud computing.

Kiran et al. [11] discussed how the Amazon EC2 agent monitors the health, availability, and performance of EC2 instances. Amazon EC2 is a combination of multiple services with unique characteristics that are utilized to form a single architecture. With Amazon EC, auto-scaling can automatically change the size of an instance, and there are also many load balancers to choose from. Amazon CloudWatch provides detailed monitoring options for snapshots and AMIs. The free tier is available for one year from the date of registration.

Fernando de la et al. [12] reviewed the applications for integrating multiagent systems and cloud computing. In addition, they showed the classification of agents and cloud computing architecture, the role of each agent application from a cloud computing perspective, and the advantages of agents on the cloud. In this regard, the authors have also emphasized the primary problem of sustaining quality of service (QoS) when user demand is high.

Distributed-MASON in the cloud provides a Simulation-as-a-Service (SIMaaS) architecture to execute simulations that involve multiple computers. This work was proposed by Michele et al. [13]. He also suggested a new infrastructure for figuring out how AWS-EC2 instances can be used to run distributed multiagent simulations from a computational and economic point of view.

Amir et al. [14] discussed the vast infrastructure for high-performance computing provided by cloud computing platforms. This study investigates comparisons and possible interactions between cloud computing and multiagent computing systems. They also set up a multiagent system on the cloud to make high-performance, complex systems easier to use.

Sara et al. [15] discussed cloud computing integrated into the Service-Oriented Multiagent (CISM) framework. They did this by adding various levels of integration for SOA with cloud computing and enabling the distribution and organization of functions. Based on the idea of cloud computing, CISM enables the flexible allocation of resources and introduces new functions in very dynamic scenarios. In addition to the JADE and FIPA requirements, the CISM model also utilizes JADE to deploy agents in cloud environments.

Tariq [16] have devised an automated method that makes it easier to use the cloud to run and work better. This paper uses a multiagent system to describe a new architecture. This architecture is used by cloud computing to find the best resources and give cloud providers and handlers a way to communicate and use the cloud as much as possible.

Araunjo et al. [17] have focused on increasing the dependability of Amazon EC2 spot instances. The author developed a fault-tolerant multiagent architecture. The authors suggest a fault-tolerant multiagent architecture as an interface for cloud users and providers to help them get access to different kinds of resources. The authors also looked at the suggested architecture using historical data from Amazon EC2 prices. Compared to other methods, authors achieved a high accuracy (98%) and gains of up to 74.48 percent in overall execution time, lowering total cost.

According to Muhammad et al. [18], Amazon implements data analysis and mining using AWS cloud computing with a high level of security. The data is stored in the cloud, which eliminates security issues and minimizes confidentiality breaches. It is significantly less expected that data saved in the AWS cloud will be lost or interfered with by third parties. AWS also focuses on aspects like data availability, privacy, monitoring suspicious information, encryption, firewalls, and preventing unauthorized user invasion.

Ali Reza [19] developed a framework to build applications with elastic cloud resources. The agents are divided into two categories: contractors and monitors. Intelligent automatic load balancing is possible in cloud configuration models with the help of agent properties like being proactive, negotiating, learning, etc. The previous work techniques, advantages, and disadvantages of cloud computing using multiagent systems are shown in Table 1 below.

Table 1: Summarization of previous work on cloud computing with multiagent system

Ref no.	Published year	Approach name	Advantages	Disadvantages
11	2015	A FLAME framework is a tool for agent-based modeling architectures.	The tool applies to modeling and simulation techniques.	There are absolutely no unsolved theories about PaaS or IaaS. During their simulations, images couldn't burst to more resources as needed, when memory got low
12	2013	+Cloud is a platform based on the cloud computing paradigm.	+Cloud has a centralized information system, a large computational load per node, and the ability to recover from decision-making process errors.	+Cloud is a technological component of the system, and its dependency on the environment is required.
13	2016	D-MASON works on the cloud and provides a Simulation-as-a-Service (SaaS) architecture.	D-MASON provides a structure that streamlines the process of setting up distributed simulations and bidding on the cost-performance ratio.	Authors need to compare the techniques with other cloud instances on more cloud simulations so that comparisons will be better.
14	2010	Authors work on components of MAS and implement a simulation framework.	The authors explored the problem of data security in cloud data storage. The user interface agent acts as a link between the user and the rest of the agents.	The author used six agents to work on the MAS architecture, which increases the complexity and management of agents in the cloud.

(Continued)

Table 1 (continued)

Ref no.	Published year	Approach name	Advantages	Disadvantages
15	2010	This paper uses the approach of the Cloud CISM@ architecture set on top of the platforms.	CISM@ has several features capable of being executed in dynamic and distributed environments to work on devices with limited storage and processing capabilities.	CISM@ does not include a service discovery mechanism, and applications, use only the services listed on the platform. All communication is handled by the platform only.
16	2018	The author practices the cloud service broker technique. The broker is considered a database center for the users and the providers in the cloud environment.	MAS is used in the cloud environment to help choose the best resources and develop ways for cloud providers and users to negotiate.	Authors have considered the cloud broker as the core of the cloud system. If the broker fails to collect data and services from an agent, the cloud broker must be controlled by a backup.
17	2019	The authors propose a fault-tolerant multiagent architecture for cloud providers and users.	Fault tolerance methods are suitable for dynamic resource allocation and scheduling, compute reliability, and flexible infrastructures to guarantee the quality of service, ensure accessibility, and achieve cost optimization.	The fundamental issue addressed by this study is the trade-off between reliability to ensure the efficacy of user applications and the low cost of spot instance resources with no guarantee of availability.
18	2020	The authors developed a framework that can be used to build an application that flexibly uses MAS cloud resources.	The authors provide agent-based solutions for designing and developing software agents to enhance cloud resources and service discovery.	A security risk associated with elastic cloud computing means it runs for a limited time, and you must rethink how you perform incident response, root cause analysis, and audits.

3 Proposed Local Architecture with JADE Agent

As shown in Fig. 1, JADE acts as a gateway that creates a multiagent system and applications that conform to FIPA guidelines. In this paper, JADE features are utilized on local and AWS servers. With the help of JADE, load balancing can be easily calculated on remote servers with reduced time, as JADE provides a set of valuable tools that help debug and deploy applications [19,20]. Agents are decision-makers and adapt themselves to user needs. Each agent knows how to connect to and work through the different layers of the cloud.

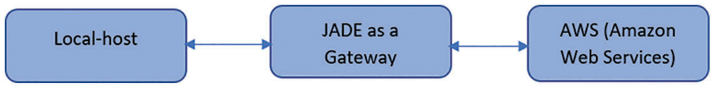


Figure 1: JADE architecture

First, JADE is installed on a local server that requires a specific system configuration setup. After that, system and hard-drive information are calculated using an API service. Secondly, JADE is installed on an AWS server, where users can create free and paid instances to check the performance of JADE (in terms of CPU utilization, memory usage, etc.) in a cloud environment by calling the same API as on the local host.

A run-time environment is provided by JADE, where agents can live, destroy, and communicate with each other. Agents have the behaviors of autonomy, reactivity, proactivity, and social behavior [21] throughout their lives. JADE supports all the features of an object-oriented language. JADE also improves the quality and complexity of applications because of its interoperability. Fig. 2 shows the architecture of a local computer system with JADE and Apache Tomcat Maven. The two agents created by JADE are:

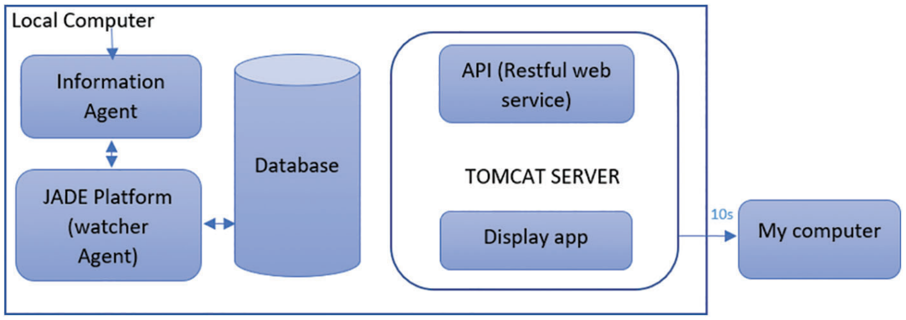


Figure 2: Architecture of local computer system

- **Information Agent**

The information agent receives the system information of the local computer, like total memory, free memory, used CPU, number of processors, latency, and hard drive space, and passes it to the watcher agent every 5 s.

- **Watcher Agent**

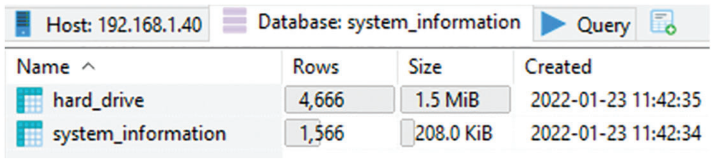
The Watcher agent watches the information agent and retrieves the system information from the information agent, and stores it in the database. The database is maintained in MySql and HeidiSql servers.

- **Restful Web Service (Stress App)**

Apart from the JADE agents, a RESTful web service is created in Java to put a load on the system. The Restful web service has a specific purpose: it helps two servers in different parts of the world communicate with each other. Restful APIs make it simple to communicate with cloud computing. They are lightweight and use the Hypertext Transfer Protocol (HTTP) protocol for implementation. These web services easily retrieve computer resources from servers and provide them to clients. In their current work, the authors checked the system information using local-host port number 8080 when no load was put on the system and again checked the system information after putting the load on port number 8080 through the stress app (Restful web service). The Tomcat Server gets web services, runs them, controls the parameters, and sends a response back to the sender. The Stress app calls the API and hits it on the system load; when putting the stress (load) on the system, the CPU and memory load change. The CPU increases its utilization with the increase of network latency.

- **Display App**

This app is used to display the result on the computer screen. This display app (localhost:9090) works before and after putting a load on the system. The MySql and HeidiSql databases, as shown in Fig. 3, are connected to this graphical user interface (GUI) based display app that will show the system information and hard-drive databases on the computer web page every 10 s.

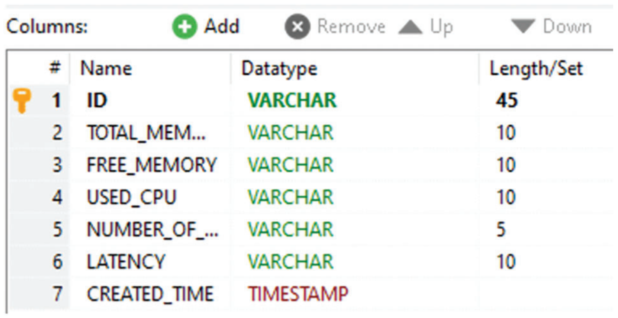


Name	Rows	Size	Created
hard_drive	4,666	1.5 MiB	2022-01-23 11:42:35
system_information	1,566	208.0 KiB	2022-01-23 11:42:34

Figure 3: Hard drive & system information in database

- **Load Checking on the Local Server**

As shown in Fig. 4, seven parameters are created to check the system information, including System Id, total memory of the system, free memory, used CPU, number of processors (CPU cores), network latency, and time of creation. All parameters check the system load at that time before and after putting the load through the stress app.



#	Name	Datatype	Length/Set
1	ID	VARCHAR	45
2	TOTAL_MEM...	VARCHAR	10
3	FREE_MEMORY	VARCHAR	10
4	USED_CPU	VARCHAR	10
5	NUMBER_OF_...	VARCHAR	5
6	LATENCY	VARCHAR	10
7	CREATED_TIME	TIMESTAMP	

Figure 4: Database created for system information for both local and AWS servers

Our primary purpose is to check the system performance on the local host and the AWS server so that a comparative analysis can be easily calculated. The configuration of the local system is as below in Table 2:

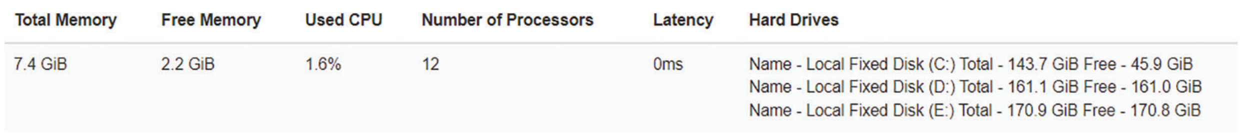
Table 2: The configuration of the local system

Local host components	System configuration
CPU	AMD Ryzen 5 5500U
RAM	8.00 GB (7.36 GB usable, SSD: 143 GB).
SSD	143 GB
Softwares	JADE 4.5, Java 11
Database	MySql, Heidisql
Servers	Tomcat,FileZila

The above system configurations, like JADE 4.5 and Java 11, are used to create agents, MySql for the database, and HeidiSql for better display of the local server and AWS database. JADE has created two agents (watchers and information agents) to check the system information. Objects are created through the setobject () method, and each object is assigned different parameters, like one object checking the total memory and the other object checking the used memory. So on. These agents talk to each other, save the information in the database, and display it on the local computer system.

```
stm.setObject(1, id);
stm.setObject(2, systemInformation.getTotal_memory());
stm.setObject(3, systemInformation.getFree_memory());
stm.setObject(4, systemInformation.getUsed_cpu());
stm.setObject(5, systemInformation.getNumber_of_processors());
stm.setObject(6, diff);
stm.setObject(7, new Date());
```

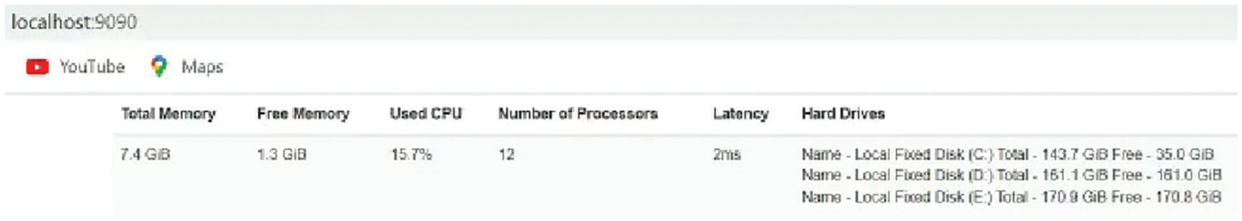
Fig. 5 shows the result of the display app when the stress app (localhost:8080/api/v1/stress) is hit on the local server. The figure below shows the most important system information for load parameters and hard drives. The CPU is used 16% of the time, and 2.2 GB of memory is used.



Total Memory	Free Memory	Used CPU	Number of Processors	Latency	Hard Drives
7.4 GiB	2.2 GiB	1.6%	12	0ms	Name - Local Fixed Disk (C:) Total - 143.7 GiB Free - 45.9 GiB Name - Local Fixed Disk (D:) Total - 161.1 GiB Free - 161.0 GiB Name - Local Fixed Disk (E:) Total - 170.9 GiB Free - 170.8 GiB

Figure 5: Display app of system information through the local host before putting the load

Fig. 6 shows the result on the display app when a load (stress app) is put on the local host server through the JADE agent. The JADE agent starts calling the web service and, through this web service agent, puts a load on the server. As the load increases, the CPU starts using the computer resources, slowly decreasing memory and increasing system latency. After a few seconds, CPU utilization drops to 15.7%, latency is 2 ms, free memory reduces from 2.2 to 1.3 GB, and the hard drive also starts impacting badly. The database snapshot before and after the API hit is also shown in Fig. 7, where CPU usage data is kept in descending order. In the first row, it is also shown that CPU utilization is 100% in a few seconds when the load is gradually increased.



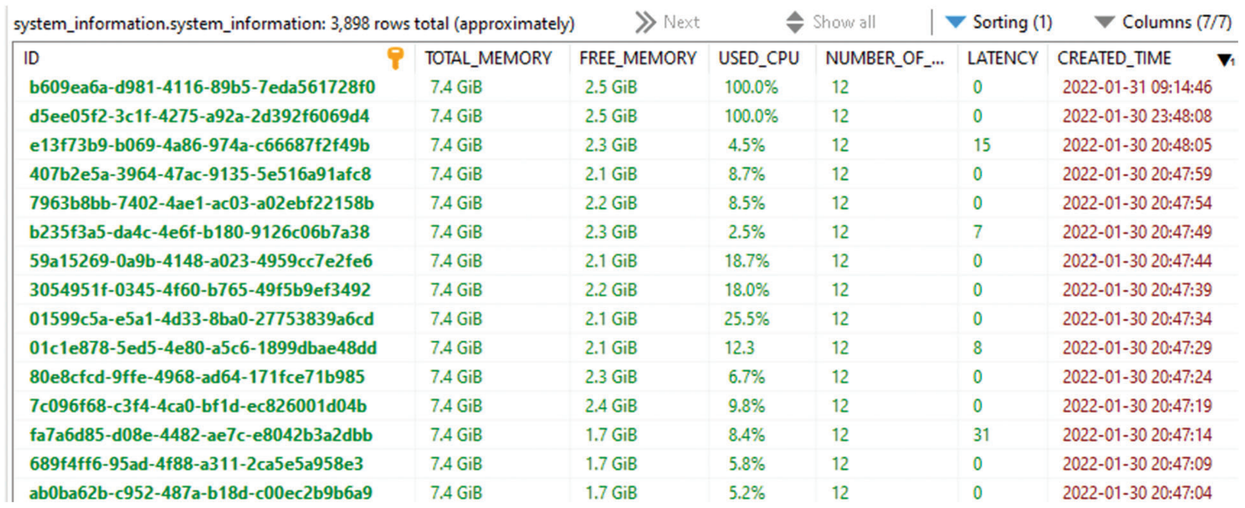
localhost:9090

YouTube Maps

Total Memory	Free Memory	Used CPU	Number of Processors	Latency	Hard Drives
7.4 GiB	1.3 GiB	15.7%	12	2ms	Name - Local Fixed Disk (C:) Total - 143.7 GiB Free - 35.0 GiB Name - Local Fixed Disk (D:) Total - 161.1 GiB Free - 161.0 GiB Name - Local Fixed Disk (E:) Total - 170.9 GiB Free - 170.8 GiB

Figure 6: Display app of system information through the local host after putting the load

From this, the authors conclude that even if a local system has a perfect configuration, a small load on the web service makes the CPU busier and slows down the other running applications. System load is the major problem when sometimes putting software agents and infrastructure on a local host.



system_information.system_information: 3,898 rows total (approximately) Next Show all Sorting (1) Columns (7/7)

ID	TOTAL_MEMORY	FREE_MEMORY	USED_CPU	NUMBER_OF_...	LATENCY	CREATED_TIME
b609ea6a-d981-4116-89b5-7eda561728f0	7.4 GiB	2.5 GiB	100.0%	12	0	2022-01-31 09:14:46
d5ee05f2-3c1f-4275-a92a-2d392f6069d4	7.4 GiB	2.5 GiB	100.0%	12	0	2022-01-30 23:48:08
e13f73b9-b069-4a86-974a-c66687f2f49b	7.4 GiB	2.3 GiB	4.5%	12	15	2022-01-30 20:48:05
407b2e5a-3964-47ac-9135-5e516a91afc8	7.4 GiB	2.1 GiB	8.7%	12	0	2022-01-30 20:47:59
7963b8bb-7402-4ae1-ac03-a02ebf22158b	7.4 GiB	2.2 GiB	8.5%	12	0	2022-01-30 20:47:54
b235f3a5-da4c-4e6f-b180-9126c06b7a38	7.4 GiB	2.3 GiB	2.5%	12	7	2022-01-30 20:47:49
59a15269-0a9b-4148-a023-4959cc7e2fe6	7.4 GiB	2.1 GiB	18.7%	12	0	2022-01-30 20:47:44
3054951f-0345-4f60-b765-49f5b9ef3492	7.4 GiB	2.2 GiB	18.0%	12	0	2022-01-30 20:47:39
01599c5a-e5a1-4d33-8ba0-27753839a6cd	7.4 GiB	2.1 GiB	25.5%	12	0	2022-01-30 20:47:34
01c1e878-5ed5-4e80-a5c6-1899dbae48dd	7.4 GiB	2.1 GiB	12.3	12	8	2022-01-30 20:47:29
80e8cfcd-9ffe-4968-ad64-171fce71b985	7.4 GiB	2.3 GiB	6.7%	12	0	2022-01-30 20:47:24
7c096f68-c3f4-4ca0-bf1d-ec826001d04b	7.4 GiB	2.4 GiB	9.8%	12	0	2022-01-30 20:47:19
fa7a6d85-d08e-4482-ae7c-e8042b3a2dbb	7.4 GiB	1.7 GiB	8.4%	12	31	2022-01-30 20:47:14
689f4ff6-95ad-4f88-a311-2ca5e5a958e3	7.4 GiB	1.7 GiB	5.8%	12	0	2022-01-30 20:47:09
ab0ba62b-c952-487a-b18d-c00ec2b9b6a9	7.4 GiB	1.7 GiB	5.2%	12	0	2022-01-30 20:47:04

Figure 7: Database snapshot of system information on the local host

4 Proposed AWS Architecture with Jade Agent

The objective of working with AWS is that it allows companies, governments, and individuals to easily store their data and APIs [22] efficiently and easily. AWS provides the user with a better way to build their application by using cloud servers and interfaces because configuring their infrastructure is quite complex and costly [23]. To implement web services on the AWS cloud, the system is divided into three parts: local host, Amazon EC2, and users. The system information is monitored when the user starts creating and destroying agents on the local host through JADE. The agents and web services are deployed on the AWS cloud so that the performance of cloud instances can be checked on a local machine.

AWS and the separate cloud server communicate via Restful web services and agents to monitor the load on the main server. When the resources in an AWS server reach a certain peak level, it means that Amazon EC2 has depleted local resources such as CPU processing and memory. The information agent coordinates and checks the whole system on both platforms. The information agent in Amazon's EC2 cloud gives information to separate servers by sending command messages to scale up or down resources depending on how much the system is occupied.

- **Load Checking on AWS Server**

To set up the whole architecture on the cloud, as shown in Fig. 8, first create an Amazon EC2 free instance named t2.micro on the main AWS server. The web service (stress app) and agents have to be uploaded onto the AWS server with the help of the FileZilla server. After that, the project must be complied with and run using Java commands through a command prompt. The same process is repeated with a separate server (client-server) for a free instance of t2.micro created on AWS. After that, the MySql server, agents, and display app have to be uploaded and compiled on the cloud server with the help of the FileZilla server. Both the running instances of t2.micro, public and elastic IPs are also created, as shown in Fig. 9. It shows the AWS server is ready to work on the application. Creating an instance and uploading the whole project and agents to the AWS cloud is convenient. An AWS-EC2 instance works on the virtual server where users can request and get provisioned to work in the cloud and install multiple instances, packages, and software [24]. EC2 allows the customer to choose the size of the boot partition, the operating system, the amount of storage space, and the application based on their requirements [25].

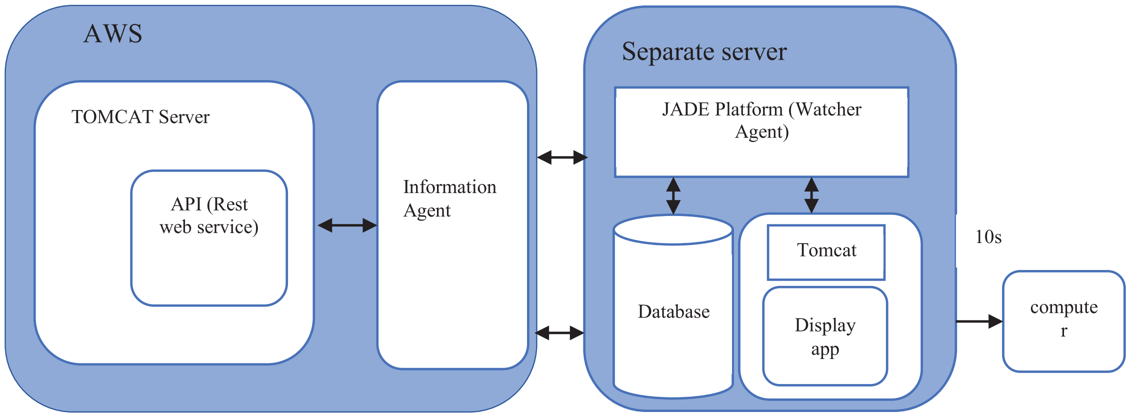
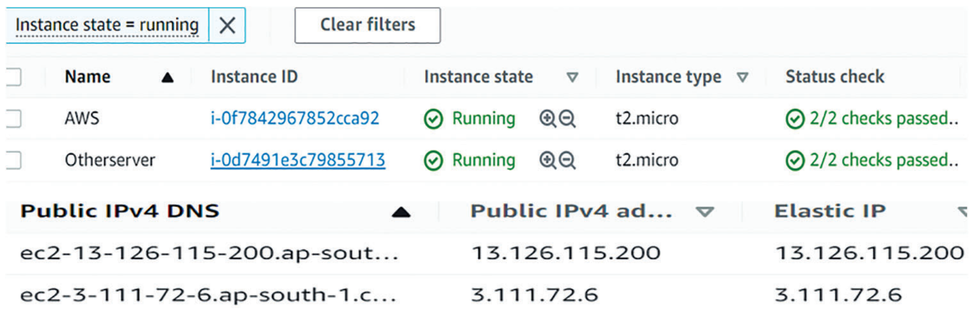


Figure 8: AWS Architecture



Instance state = running | Clear filters

Name	Instance ID	Instance state	Instance type	Status check
AWS	i-0f7842967852cca92	Running	t2.micro	2/2 checks passed..
Otherserver	i-0d7491e3c79855713	Running	t2.micro	2/2 checks passed..

Public IPv4 DNS	Public IPv4 ad...	Elastic IP
ec2-13-126-115-200.ap-sout...	13.126.115.200	13.126.115.200
ec2-3-111-72-6.ap-south-1.c...	3.111.72.6	3.111.72.6

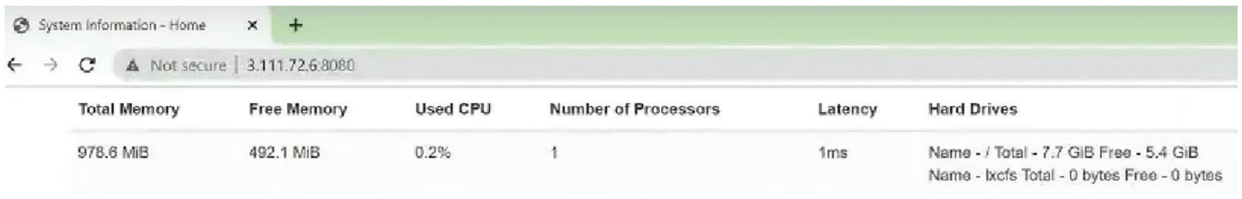
Figure 9: AWS and separate server instances & IP address

The cloud instances can be charged hourly depending on the user's requirements. AWS provides different instances for different business projects. Users can access their CPU and memory as long as a project runs. A t2.micro instance is free, so users have minimal memory and CPU usage access. The free configuration that we get from AWS is below in Table 3.

Table 3: AWS configuration

AWS components	System configuration
CPU	1 vCore
RAM	1 GB
SSD	30 GB
Instance	t2. micro
Servers	AWS EC2/another server

System information like CPU utilization, latency, memory usage, etc., is checked first on the AWS. The information agent takes all the information in the system. The watcher agent, installed on a separate cloud server, observes and retrieves the system information and stores it in the database. Through the display app, the system information is displayed on the computer after almost 10 s, as shown in Fig. 10. The used CPU is 0.2%, and the total memory is 978.6 MB.



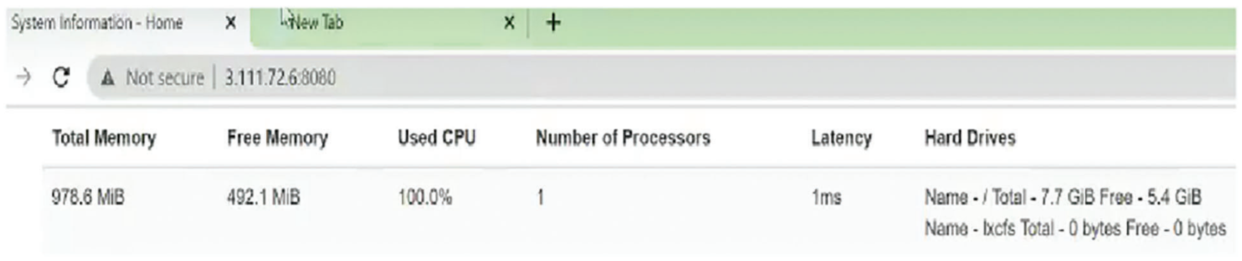
System Information - Home

Not secure | 3.111.72.6:8080

Total Memory	Free Memory	Used CPU	Number of Processors	Latency	Hard Drives
978.6 MiB	492.1 MiB	0.2%	1	1ms	Name - / Total - 7.7 GiB Free - 5.4 GiB Name - lxcfs Total - 0 bytes Free - 0 bytes

Figure 10: Display app shows system information before putting the load on the AWS server

The same process is repeated when the API is hit on the AWS server. The system information is again checked through information and watcher agents. As shown in Fig. 11, the stored information can be retrieved from the database and shown on the system using the display app. The AWS EC2 cloud can easily calculate the load difference before and after hitting the API. After hitting the web service, the load starts increasing on the CPU. With minimal total memory available, the CPU utilization becomes 100% instantly.



System Information - Home | New Tab

Not secure | 3.111.72.6:8080

Total Memory	Free Memory	Used CPU	Number of Processors	Latency	Hard Drives
978.6 MiB	492.1 MiB	100.0%	1	1ms	Name - / Total - 7.7 GiB Free - 5.4 GiB Name - lxcfs Total - 0 bytes Free - 0 bytes

Figure 11: Display app shows system information after putting the load on AWS server

The best part about the AWS server is that it calls the other servers available on the cloud, uploads the agents and web services, runs the application, calculates the load before and after hitting the web service, and updates the database. The cloud user does not have to be set up and pay for infrastructure; calculations also get easier and faster [26].

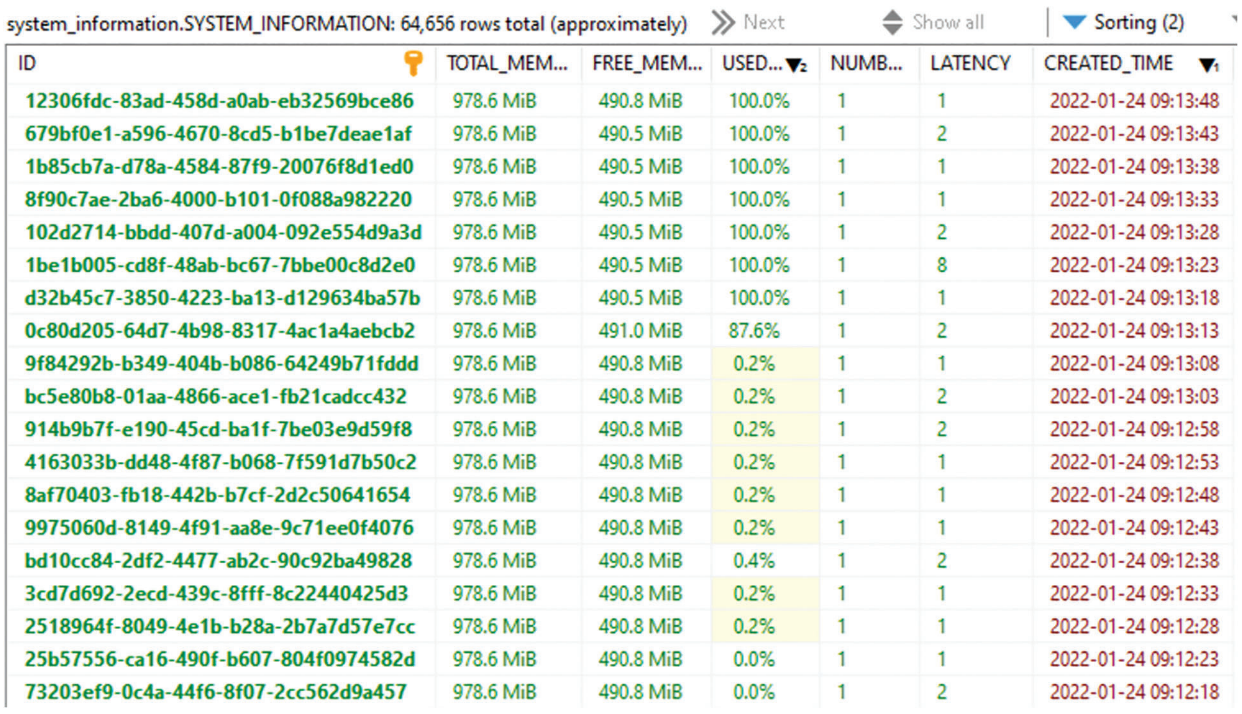
Within the minimum system configuration on the AWS server, the information regarding memory usage, CPU utilization, latency, hard-drive information, etc., can be easily calculated. The snapshot of the system information database is shown in Fig. 12. When the server starts to get busy, the CPU usage is at 0%, but it goes up right away and reaches 100% in a short time.

5 Comparative Analysis of System Load on Local Host and AWS Server

The current research activities have been focused on the local host and AWS infrastructures, with little attention paid to how agents work well on cloud servers. The multiagent system in the cloud can operate, coordinate, and share the results in the cloud with the local host system. Some of the essential things the authors notice about the AWS server compared to local servers are:

- A considerable volume of memory and processing resources can be dynamically organized for the execution of multiagent systems on an AWS server compared to a local server. Working and coordinating with many agents at a time on local servers is a little cumbersome.
- To set up a project on a local server, researchers need a lot of applications, software, etc., but with AWS, they only need a few things to get the best results [25].
- It is easier to scale up the application on AWS by creating more instances (free or paid). AWS charges extra for every service, but it's more or less what you use, so you only have to pay for that.

- Reliability, security, speed of response, and storage space are the main good things about AWS that a local server doesn't have.


system_information.SYSTEM_INFORMATION: 64,656 rows total (approximately)

ID	TOTAL_MEM...	FREE_MEM...	USED...	NUMB...	LATENCY	CREATED_TIME
12306fdc-83ad-458d-a0ab-eb32569bce86	978.6 MiB	490.8 MiB	100.0%	1	1	2022-01-24 09:13:48
679bf0e1-a596-4670-8cd5-b1be7deae1af	978.6 MiB	490.5 MiB	100.0%	1	2	2022-01-24 09:13:43
1b85cb7a-d78a-4584-87f9-20076f8d1ed0	978.6 MiB	490.5 MiB	100.0%	1	1	2022-01-24 09:13:38
8f90c7ae-2ba6-4000-b101-0f088a982220	978.6 MiB	490.5 MiB	100.0%	1	1	2022-01-24 09:13:33
102d2714-bbdd-407d-a004-092e554d9a3d	978.6 MiB	490.5 MiB	100.0%	1	2	2022-01-24 09:13:28
1be1b005-cd8f-48ab-bc67-7bbe00c8d2e0	978.6 MiB	490.5 MiB	100.0%	1	8	2022-01-24 09:13:23
d32b45c7-3850-4223-ba13-d129634ba57b	978.6 MiB	490.5 MiB	100.0%	1	1	2022-01-24 09:13:18
0c80d205-64d7-4b98-8317-4ac1a4aebcb2	978.6 MiB	491.0 MiB	87.6%	1	2	2022-01-24 09:13:13
9f84292b-b349-404b-b086-64249b71fddd	978.6 MiB	490.8 MiB	0.2%	1	1	2022-01-24 09:13:08
bc5e80b8-01aa-4866-ace1-fb21cadcc432	978.6 MiB	490.8 MiB	0.2%	1	2	2022-01-24 09:13:03
914b9b7f-e190-45cd-ba1f-7be03e9d59f8	978.6 MiB	490.8 MiB	0.2%	1	2	2022-01-24 09:12:58
4163033b-dd48-4f87-b068-7f591d7b50c2	978.6 MiB	490.8 MiB	0.2%	1	1	2022-01-24 09:12:53
8af70403-fb18-442b-b7cf-2d2c50641654	978.6 MiB	490.8 MiB	0.2%	1	1	2022-01-24 09:12:48
9975060d-8149-4f91-aa8e-9c71ee0f4076	978.6 MiB	490.8 MiB	0.2%	1	1	2022-01-24 09:12:43
bd10cc84-2df2-4477-ab2c-90c92ba49828	978.6 MiB	490.8 MiB	0.4%	1	2	2022-01-24 09:12:38
3cd7d692-2ecd-439c-8fff-8c22440425d3	978.6 MiB	490.8 MiB	0.2%	1	1	2022-01-24 09:12:33
2518964f-8049-4e1b-b28a-2b7a7d57e7cc	978.6 MiB	490.8 MiB	0.2%	1	1	2022-01-24 09:12:28
25b57556-ca16-490f-b607-804f0974582d	978.6 MiB	490.8 MiB	0.0%	1	1	2022-01-24 09:12:23
73203ef9-0c4a-44f6-8f07-2cc562d9a457	978.6 MiB	490.8 MiB	0.0%	1	2	2022-01-24 09:12:18

Figure 12: Snapshot of system information database on the AWS server

As shown in Figs. 13 and 14, the graphs of CPU utilization on the local server and the AWS server are shown. The values have been taken from the database as shown in Figs. 7 and 12. Since the AWS server has 1GB of total memory and a 12 vcore processor available, the CPU touches 100% utilization very sharply when the stress app loads on the server. That is why the curve is so steep. However, in the case of a local server, when the same stress app is applied with JADE agents on the 8 GB of memory and 12 vcore processors, the CPU makes itself busy running the same API. After a period, CPU utilization again reaches 100%. The total system speed falls off drastically in a few minutes.

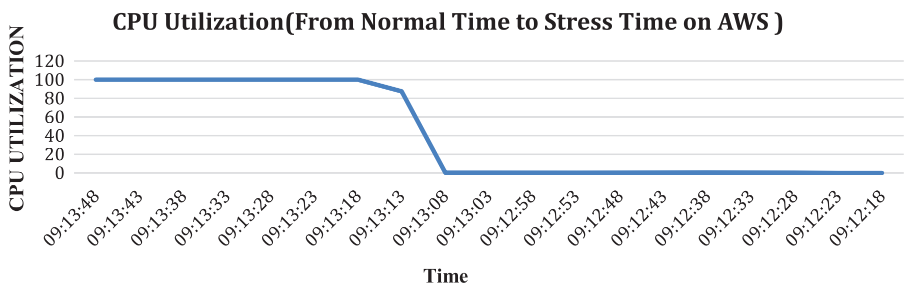


Figure 13: CPU utilization on the AWS server

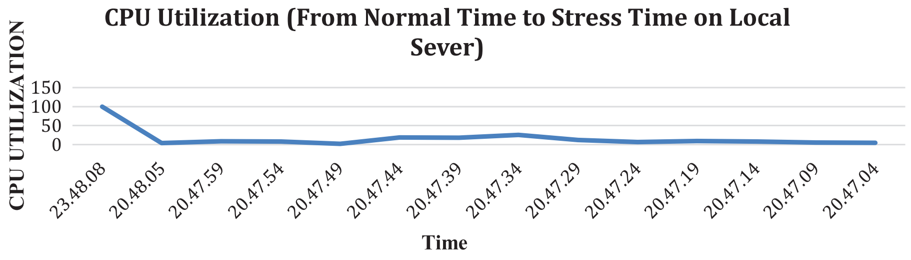


Figure 14: CPU utilization on the local host

From the above comparison of processor speed and CPU utilization, we conclude that either the system has an excellent configuration or it can be halted at any time, even with a small load on the server. Working on AWS-EC2 is relatively easy. We get our separate servers without having to establish a vast infrastructure. The biggest strength of EC2 is that it is conceptually simple; anyone can easily understand and work on it.

Figs. 15 and 16 show the network latency on AWS and the local server. Network latency, also known as lag, refers to the time it takes data packets to travel from source to destination. We observed the AWS graph of network latency and found that latency transmission levels are even lower than local servers. The latency peak is difficult to achieve at only one point of load. The graph of the local-host network latency shows that the latency peak is higher at certain times; this means that the rate at which data is sent is slower at some points. Network latency also decreases the throughput and limits the bandwidth of a system. So, data packets will take more time on a local server as compared to an AWS server.

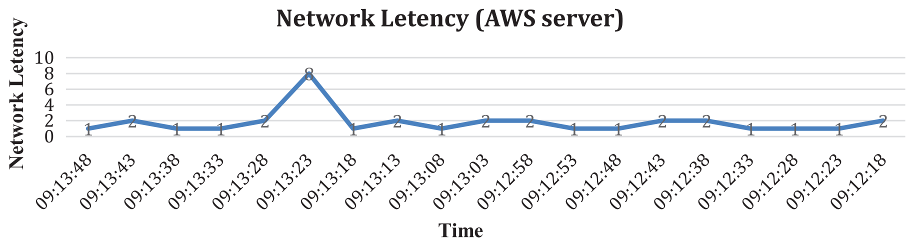


Figure 15: Network Latency on AWS server

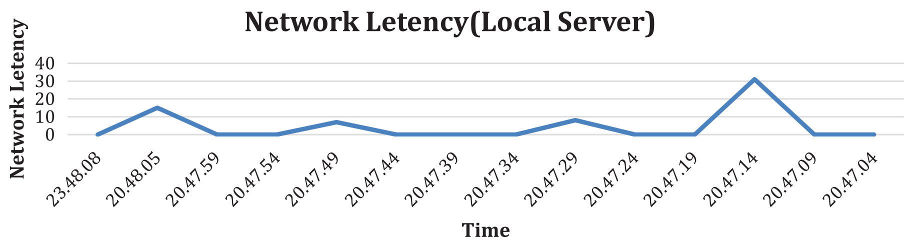


Figure 16: Network Latency on local server

6 Benefits of Cloud Computing in Mathematical Modeling and Engineering Applications

- Mathematical modeling and engineering applications are useful in providing relationships with cloud computing. It helps to determine the best values and methods and predict how things will work accurately in the cloud.
- Islam et al. [26] have made a model for the elasticity of cloud instances like Amazon CloudWatch.

- Pallavi et al. [27] discussed the comparative analysis of Microsoft Azure, Google Cloud, and Amazon EC2 with their specifications, support, pricing, databases, and different architectures.
- Beloglazov et al. [28] use statistical data analysis from virtual machines to figure out how CPUs should be used in a large heterogeneous cloud.
- Mi et al. [29] present a mathematical model of virtual machine distribution as a multi-constraint optimization problem that balances CPU utilization and energy consumption.
- Open-source simulator software applications include CloudSim, Networkcloudsim, greencloud, MDCsim, and icancloud [30].

7 Discussions

Amazon EC2 instances are designed for architectures ranging from simple to sophisticated. It is used for small-scale to enterprise-level applications, including big data and analytics workloads. The developer can easily upload and deploy the services on an AWS-EC2 server and get the desired results. The agents help in utilizing the full use of cloud resources effectively. The main feature of the agent system is that it works well on cloud servers. So, in this paper, the authors work on AWS cloud computing and local servers to check the CPU utilization and memory usage of EC2 instances on both platforms by using multiagent systems. The authors used memory usage, CPU utilization, network latency, and other system configuration parameters to prove the best results on an agent's system on an Amazon server. They proved that for small setups, the local host is best with agents, but as the configuration and data sets expand, it is better to use an agent system on the cloud, and JADE is the best tool for building and configuring agent systems.

8 Conclusion

The benefits of cloud computing are huge today. Some of the main advantages of cloud computing are data protection, cost-effectiveness, multiple storage, and auto-scaling. As part of the cloud provider, the multiagent system and cloud resources respond to client requests and complete their tasks. One of them is an Amazon Web Services EC2 cloud resource. EC2 is a service that provides flexible computing capacity, and instances of EC2 can be installed and run in a few minutes. In this paper, the authors discussed and compared the multiagent system's results on the local and AWS servers. Agent deployment on clouds was previously complex. However, with the help of the AWS Server, memory usage, CPU utilization, network latency, and other system configurations have significantly improved. To prove the result, users have to go through a huge set of innovative configurations on localhost. However, the results are far better on AWS, even with the minimum system configuration. Even when infrastructure, software, operating systems, and applications are considered, AWS resources are still very flexible, easy to use, and affordable.

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

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

References

[1] V. Jinesh and M. Sajee, "Overview of Amazon web services," *Amazon Web Services*, vol. 105, pp. 1–22, 2014.

[2] M. Sourav, "Benefits of AWS in the modern cloud," Available at SSRN 3415956, pp. 1–21, 2019.

[3] A. Al-Sajid and A. Peter, "Scalability analysis comparisons of cloud-based software services," *Journal of Cloud Computing*, vol. 8, no. 1, pp. 1–17, 2019.

[4] P. M. Rajendra, N. R. Lakshman and V. Bapuji, “Cloud computing: Research issues and implications,” *International Journal of Cloud Computing and Services Science*, vol. 2, no. 2, pp. 133–139, 2013.

[5] N. Saakshi and J. Arushi, “Cloud computing security: Amazon web service,” in *Fifth Int. Conf. on Advanced Computing & Communication Technologies*, IEEE, Haryana, India, pp. 501–505, 2015.

[6] N. Jose Pergentino Araujo, M. P. Donald and R. G. Celia, “An agent-based fog computing architecture for resilience on Amazon EC2 spot instances,” in *7th Brazilian Conf. on Intelligent Systems (BRACIS)*, IEEE, Sao Paulo, Brazil, pp. 360–365, 2018.

[7] A. Merizig, K. Okba and L. Maite, “A dynamic and adaptable service composition architecture in the cloud based on a multi-agent system,” *International Journal of Information Technology and Web Engineering (IJITWE)*, vol. 13, no. 1, pp. 50–68, 2018.

[8] A. Azma, K. Nima and C. Hossein, “Research and development on cloud computing,” in *5th Int. Conf. on Advance Research in Science and Technology*, Berlin, pp. 1–13, 2021.

[9] K. Neuman, T. Galib Ahmad and B. Peter, “Persistent and scalable JADE: A cloud-based in memory multi-agent framework,” arXiv preprint arXiv:2009.06425, 2020.

[10] M. Abdelhak, O. Kazar and M. Lopez-Sanchez, “A multi-agent system approach for service deployment in the cloud,” *International Journal of Communication Networks and Distributed Systems*, vol. 23, no. 1, pp. 69–92, 2019.

[11] M. Kiran, M. Kabiru, M. Haroon, M. Bashir and O. Ashraf Al, “Agent-based modeling as a service on Amazon EC2: Opportunities and challenges,” in *IEEE 8th Int. Conf. on Utility and Cloud Computing (UCC)*, Limassol, Cyprus, pp. 251–255, 2015.

[12] P. P. Fernando de la, G. Sara Rodríguez, P. Javir Bajo and C. R. Juan Manuel, “A multi-agent system for resource distribution into a cloud computing environment,” in *Int. Conf. on Practical Applications of Agents and Multi-Agent Systems*, Berlin, Heidelberg, Springer, pp. 37–48, 2013.

[13] C. Michele, C. Gennaro, S. Flavio, S. Carmine, P. Szufel *et al.,* “DMason on the cloud: An experience with Amazon web services,” in *European Conf. on Parallel Processing*, Cham, Springer, pp. 322–333, 2016.

[14] T. M. Amir, R. Atan, R. Abdullah and M. Masrah Azrifah Azmi, “A framework of multiagent system to facilitate security of cloud data storage,” in *Int. Conf. on Cloud Computing & Virtualization*, Singapore, pp. 241–257, 2010.

[15] R. Sara, D. I. Tapia, E. Sanz, C. Zato, P. Fernando de la *et al.,* “Cloud computing integrated into the service-oriented multi-agent architecture,” in *Int. Conf. on Information Technology for Balanced Automation Systems*, Berlin, Heidelberg, Springer, pp. 251–259, 2010.

[16] A. Tariq, “Cloud computing and multi-agent system: Monitoring and services,” *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 5, pp. 2435–2444, 2018.

[17] J. P. N. Araunjo, P. Jose, D. M. Pianto and R. G. Celia, “Towards increasing reliability of Amazon EC2 spot instances with a fault-tolerant multi-agent architecture,” *Multiagent and Grid Systems*, vol. 15, no. 3, pp. 259–287, 2019.

[18] T. Muhammad, M. Sohail and H. Hajar, “Analysis of research on Amazon AWS cloud computing seller data security,” *International Journal of Research in Engineering Innovation*, vol. 4, no. 3, pp. 131–136, 2020.

[19] H. Ali Reza, “Developing an elastic cloud computing application through multi-agent systems,” *International Journal of Cloud Applications and Computing (IJCAC)*, vol. 3, no. 1, pp. 58–64, 2013.

[20] L. Upasana, I. Elamvazuthi, F. Meriaudeau, G. Ramasamy and J. Ajay, “Developing a multi-agent system in JADE for micro grids,” in *2018 IEEE 4th Int. Symp. in Robotics and Manufacturing Automation (ROMA)*, IEEE, Perambalur, India, pp. 1–5, 2018.

[21] M. Artan, D. Minarolli and B. Freisleben, “Distributed resource allocation in cloud computing using multi-agent systems,” *Telfor Journal*, vol. 9, no. 2, pp. 110–115, 2017.

[22] S. Prashansa, “Cloud computing using AWS: An analysis,” *Indian Journal of Computer Science*, vol. 5, no. 6, pp. 27–35, 2020.

[23] K. Balaji, “Load balancing in cloud computing: Issues and challenges,” *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 2, pp. 3077–3084, 2021.

[24] W. Michael and W. Andreas, “Amazon web services in action,” in *The Simon and Schuster*, 2^{nd} ed., New York, US: Manning Publications, pp. 1–528, 2018.

[25] K. Gurudatt, S. Ramesh and G. Jayant, “Cloud computing-infrastructure as service-Amazon EC2,” *International Journal of Engineering Research and Applications*, vol. 2, no. 1, pp. 117–125, 2012.

[26] S. Islam, K. Lee, A. Fekete, and A. Liu, “How a consumer can measure elasticity for cloud platforms,” in *3rd Joint WOSP/SIPEW Int. Conf. on Performance Engineering, ICPE '12*, Boston, Massachusetts, USA, pp. 85–96, 2012.

[27] W. Pallavi, T. Minaiy, and C. Rutuja, “Comparative study of cloud platforms-microsoft azure, google cloud platform and Amazon EC2,” *International Journal of Research in Engineering and Applied Sciences*, vol. 5, no. 2, pp. 60–64, 2020.

[28] A. Beloglazov, R. Buyya, “Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers,” in *8th Int. Workshop on Middleware for Grids, Clouds and e-Science, MGC '10*, Bangalore, India, pp. 41–46, 2010.

[29] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi *et al.,* “Online self-reconfiguration with a performance guarantee for energy-efficient large-scale cloud computing data centers,” in *7th Int. Conf. on Services Computing (SCC)*, Miami, FL, USL, pp. 514–521, 2010.

[30] A. Nunez, J. L. Vazquez-Poletti, A. C. Caminero, G. G. Castane, J. Carretero *et al.,* “Icancloud: A flexible and scalable cloud infrastructure simulator,” *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, 2012.