[1]Akrati Sharma

[2]Priti Maheshwary

[3]Timothy Malche

Developing A Novel Load Balancing Approach for Mitigating Resource Contention in Multi-Cloud Environments

Abstract: - In the rapidly evolving landscape of cloud computing, the management of resources and load balancing across multiple cloud environments presents significant challenges. This paper proposes a novel approach to address resource contention issues in multi-cloud environments through the development of an innovative load-balancing mechanism. By leveraging advanced algorithms and dynamic resource allocation strategies, our approach aims to effectively distribute workloads across diverse cloud infrastructures, thereby optimizing resource utilization and mitigating contention. Through extensive experimentation and evaluation, we demonstrate the effectiveness and efficiency of our proposed solution in enhancing performance and resilience in multi-cloud deployments. Our findings underscore the importance of tailored load-balancing mechanisms in enabling scalable and reliable operations in complex cloud environments.

Keywords: Multi-cloud environments, Load balancing, Resource contention, Cloud computing, dynamic resource allocation

I. Introduction

The proliferation of cloud computing has fundamentally transformed the way businesses and organizations manage their IT infrastructure and deliver services to users. With the increasing adoption of cloud services [2], particularly in multi-cloud environments, organizations are confronted with the challenge of efficiently managing and allocating resources across diverse cloud platforms. Multi-cloud environments, characterized by the utilization of services from multiple cloud providers, offer benefits such as redundancy, flexibility, and cost optimization. However, they also introduce complexities related to resource contention and load balancing.

Resource contention [3] arises when multiple workloads compete for the same set of resources within a cloud environment, leading to performance degradation, increased latency, and potential service disruptions. Traditional load balancing techniques, designed primarily for single-cloud environments, are often inadequate in addressing the dynamic and heterogeneous nature of multi-cloud setups. As organizations increasingly rely on diverse cloud infrastructures to meet their computational demands, the need for innovative solutions to mitigate resource contention and optimize workload distribution becomes paramount.

II. Simulation Tool

CloudAnalyst [1] is a simulation tool designed for modeling and analyzing cloud computing environments. Developed by the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, CloudAnalyst facilitates the evaluation and comparison of various cloud deployment strategies, resource provisioning policies, and workload scheduling algorithms.

Key Features of CloudAnalyst:

• Cloud Environment Simulation: CloudAnalyst allows users to model complex cloud infrastructures, including data centers, virtual machines, and network configurations. Users can define parameters such as server capacities, network bandwidth, and geographical distribution to simulate realistic cloud environments.

Workload Generation and Management: The tool enables users to generate synthetic workloads or use real workload traces to simulate the behavior of cloud applications. Workload characteristics such as arrival rates, service demands, and resource requirements can be configured to reflect different application profiles.

[1] Research Scholar, Department of Computer Science & Engineering Rabindranath Tagore University. Bhopal (M.P.) Email: Akratisharma999@gmail.com

[2]Professor, Department of Computer Science & Engineering Rabindranath Tagore University. Bhopal (M.P.) Email: pritiaheshwary@gmail.com

[3]Department of Computer Applications, Manipal University Jaipur, Jaipur, Rajasthan, Email: timothy.malche@gmail.com

• Resource Provisioning Strategies: CloudAnalyst supports the evaluation of various resource provisioning strategies, including static allocation, dynamic scaling, and auto-scaling mechanisms. Users can analyze the performance of different provisioning policies under varying workload conditions.

• Load Balancing Algorithms: The tool offers support for evaluating different load balancing algorithms [4] aimed at optimizing resource utilization and minimizing response times in cloud environments. Users can experiment with load-balancing policies to assess their impact on system performance and scalability.

Performance Metrics and Analysis: CloudAnalyst provides a range of performance metrics, including response time, throughput, resource utilization, and energy consumption, to evaluate the efficiency and effectiveness of cloud deployments. Users can conduct comprehensive analyses to identify bottlenecks, optimize configurations, and make informed decisions regarding resource management strategies.

• Visualization and Reporting: The tool offers visualization capabilities to help users visualize simulation results, including graphs, charts, and heatmaps. Additionally, CloudAnalyst allows users to generate reports summarizing key findings and insights from their simulations.

As right now cloudanalyst[5,8] uses the traditional approach to resolve the issue of load balancing over the data centers of cloud. At certain extents these algorithms will work, but when there will be need to scale up the data centers problems will be encountered. Hence if we combine the concepts of optimization while balancing the load over cloud it will produce better results. Rather than just using the load balancing policies, if some artificial intelligence-based optimization techniques and genetic based rule algorithms are used will enhance the performance of data center and minimize the issues of data center crashing and overloading. Although there is some already implemented optimization techniques over the cloud like ant colony optimization, particle swarm optimization [6], cuckoo search, bat optimization, and so on. Hence combining the advantages of all these algorithms one Global optimization technique should be used.

Cloudanalyst is having three options of load balancing, it contains round robin, equally spread current execution, throttled algorithm. Also, it contains some service broker policy like closest data center, optimal response time, reconfigure dynamically with load. These options are present in cloud analyst for multi cloud environment.

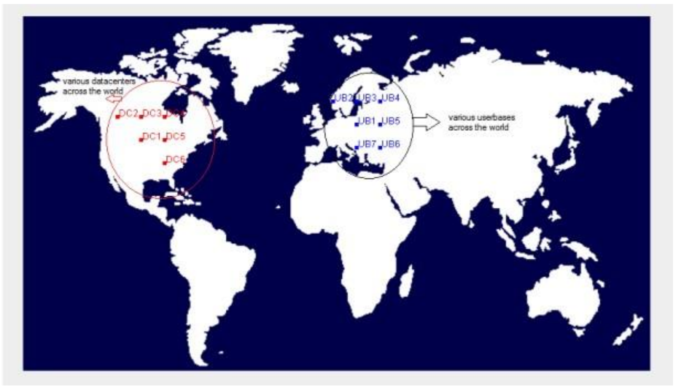

various datacenters across the world
DC2
DC3
DC1
DC5
DC6
UB2
UB3
UB4
UB1
UB5
UB7
UB6
various userbases across the world

Fig 1.1 Cloud analyst home screen showing data centres and user bases across the world.

To ease out the technique the whole methodology will be divided into the three stages:

Stage I- The allocation of resources is going to be dynamic, multiple managers with defined roles check the availability of resources on server side. Based on this method all the resources are ranked, and the best ranked resource is allocated to the process. The proposed technique is called as DRAUR Dynamic resource allocation using ranking. This will lead to reduce response time and energy consumption as well.

Stage II- In this stage focus is on the validation of request. Once a request is validated then it will be allocated to the process. The request validation can be done using SSL handshaking.

Stage III - In this best resource is to find out since fitness function. To find that artificial immune system technique must be used.

To understand the working of these stages collaboratively let us look into the flow chart as shown in figure 1.2.

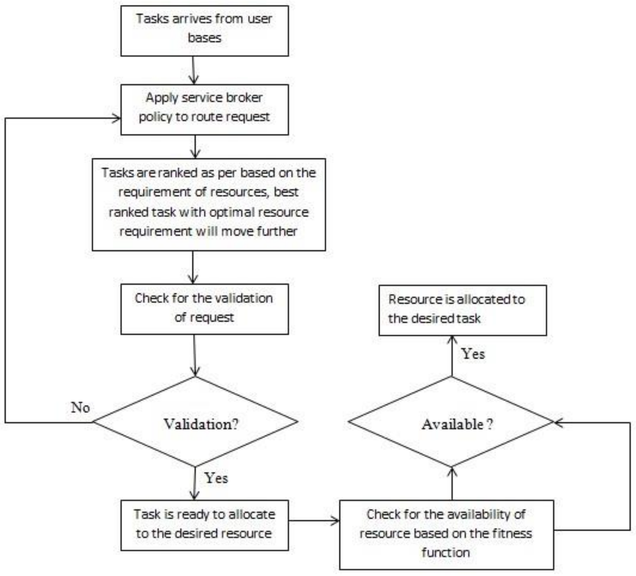

Tasks arrives from user bases
Apply service broker policy to route request
Tasks are ranked as per based on the requirement of resources, best ranked task with optimal resource requirement will move further
Check for the validation of request
Validation?
No
Yes
Task is ready to allocate to the desired resource
Check for the availability of resource based on the fitness function
Available ?
Yes
Resource is allocated to the desired task

Fig 1.2 Flowchart of used methodology

The flow chart working is as follows:

• Tasks arrive from the user bases.

• The service broker policy routes the request.

• Tasks are ranked based on the requirement of resources. The best-ranked task with the optimal resource requirement will move further.

• The request is validated.

• If the request is valid, the system checks for the availability of the resource based on the fitness function.

• If the resource is available, it is allocated to the desired task.

• If the resource is not available, the system checks for validation.

• If the request is still valid, the system waits for the resource to become available. If the request is not valid, the task is rejected.

III. Proposed Methodology

Now let‟s focus on each stage in detail:

Stage I: Dynamic Resource Allocation Using Ranking (DRAUR)

The primary objective of „Stage I‟ is to efficiently allocate resources in a dynamic multi-cloud environment. This allocation is based on a ranking system that considers the preferences or criteria set by multiple managers. The aim is to enhance resource utilization, reduce response time, and minimize energy consumption.

Key Components:

1. Set of Variables:

- R: The set of available resources in the multicloud environment.
- P: The set of processes that require resource allocation.

- M: The set of managers responsible for making allocation decisions.
- S: The server-side availability matrix, where S_{ij} represents the availability of resource *j* to manager *i*.
- C: The criteria matrix, where C_{ij} *represents* the value of a specific criterion for resource *j* as evaluated by manager *i*.
- A: The allocation matrix, where A_{ij} is a binary variable indicating whether resource *j* is allocated to process *i*.

2. Objective Function:

$Z=\sum_{i\in M} \sum_{j\in R} C_{ij}\cdot A_{ij}$

The objective function aims to maximize the total value of the allocated resources. It is the sum of the criteria values of the selected resources, based on the preferences of the managers.

3. Availability Constraint: $\sum_{i\in M} A_{ij} \leq S_{ij} \forall j\in R$

This constraint ensures that the total allocation of a resource by all managers does not exceed the availability of that resource.

4. Allocation Constraint:

$\sum_{j\in R} A_{ij}=1 \ \forall i\in P$

This constraint ensures that each process is allocated exactly one resource.

5. Binary Variable Constraint:

$A_{ij}\in \{0, 1\} \ \forall_i\in P, \forall_j\in R$

This constraint makes the allocation matrix variables binary, indicating whether a resource is allocated (1) or not (0) to a process.

In this stage, multiple managers assess the availability of resources and rank them based on specific criteria. The mathematical model captures these decisions, and the objective function seeks to maximize the total value of the allocated resources. The availability constraint ensures that the allocation does not exceed the available resources, and the allocation constraint ensures that each process is assigned exactly one resource.

This stage sets the foundation for a dynamic resource allocation [11] approach, where the ranking system enables managers to collaboratively decide the best-suited resources for each process. The dynamic nature of the allocation process adapts to changes in resource availability and criteria preferences, ultimately contributing to improved system performance and reduced energy consumption.

Stage II: Request Validation Using SSL Handshaking

The primary objective of Stage II is to ensure the authenticity and integrity of incoming requests before allocating resources to processes. By employing SSL handshaking, the goal is to establish a secure and encrypted connection between the requester (process) and the resource being requested. This validation mechanism enhances the security [7] and reliability of the resource allocation process in a multi-cloud environment.

SSL Handshaking:

SSL handshaking is a cryptographic protocol used to establish a secure connection between a client (process) and a server (resource). It involves a series of steps to authenticate both parties, negotiate encryption algorithms, and establish a secure communication channel. During the handshaking process, the server presents its digital certificate to the client, which is verified to ensure the server's authenticity. Once the handshake is successfully completed, data can be exchanged securely between the client and server.

Here we use CHAP algorithm for SSL handshaking:

The Challenge-Handshake Authentication Protocol (CHAP) is a cryptographic authentication mechanism commonly used in computer networks to verify the identity of a user or system attempting to access network resources. CHAP operates at the link layer (Layer 2) of the OSI model and is particularly prevalent in remote access scenarios such as dial-up connections or Virtual Private Network (VPN) connections.

IV. Chap Working

Initiation Phase:

The authenticator (e.g., a network access server) sends a challenge message to the peer (e.g., a remote client) during the initial connection setup.

The challenge is typically a randomly generated string or a sequence of characters.

Response Phase:

Upon receiving the challenge, the peer generates a one-way hash (using a cryptographic hash function) of the challenge concatenated with a secret known only to the peer (e.g., a password).

This hashed value serves as the response to the challenge and is sent back to the authenticator.

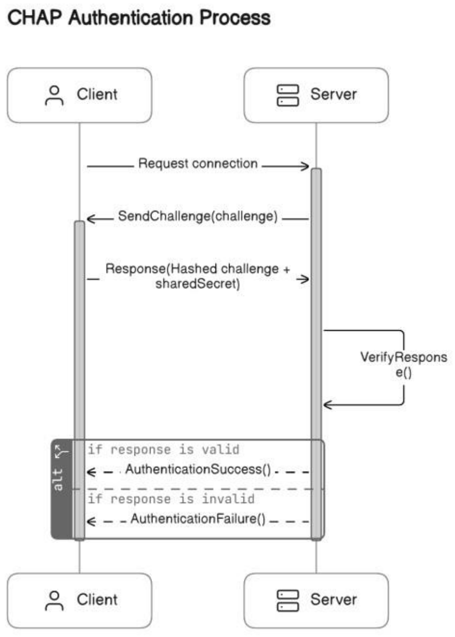


Fig 1.3 Process of Challenge-Handshake Authentication Protocol (CHAP)

Verification Phase:

The authenticator receives the response and independently computes the expected response using the same hash function.

If the computed response matches the response received from the peer, the authentication is successful. Otherwise, the connection is terminated, or further authentication attempts may be made.

As shown in fig 1.3 , The Challenge-Handshake Authentication Protocol

(CHAP)enhances security by ensuring that neither the client nor the server sends plaintext passwords over the network. Instead, both parties possess knowledge of a shared secret, which is never transmitted. This fundamental design principle significantly strengthens security compared to protocols like the Password Authentication Protocol (PAP), which transmit plaintext passwords, making them vulnerable to interception and exploitation.

Unlike PAP, which exposes plaintext passwords and relies on the secrecy of their transmission, CHAP operates differently. In CHAP, the plaintext password is never sent over the network. Instead, the authentication process involves a challenge-response mechanism that utilizes a shared secret between the client and the server.

In the context of Point-to-Point Protocol (PPP) servers, CHAP provides intermittent verification of the client's identity through a three-way handshake during the establishment of the initial connection (Link Control Protocol, LCP). This verification may occur multiple times throughout the connection. The process begins with the authenticator (server) sending a challenge message to the client. The client responds with a value calculated using a one-way hash function applied to the challenge and the shared secret. The authenticator verifies the response by computing the expected hash value based on its own stored copy of the secret. If the calculated hash values match, authentication is successful; otherwise, the connection is terminated.

The CHAP authentication process significantly enhances security and resilience against eavesdropping and unauthorized access. In this process, both the server and the client mutually authenticate each other using a shared secret, enhancing the overall security of the connection establishment Key Characteristics of CHAP:

• Mutual Authentication: CHAP provides mutual authentication, meaning both the client and the server authenticate each other.

• Challenge-Response Mechanism: The use of challenges and responses ensures that authentication credentials are not sent in plaintext over the network, enhancing security.

• Cryptographic Hashing: CHAP relies on cryptographic hash functions to generate and verify the responses, making it resistant to replay attacks and eavesdropping.

• Dynamic Re-authentication: CHAP supports periodic re-authentication, where the authenticator sends new challenges to the peer at regular intervals to verify the continued presence of the peer.

• Protection against Replay Attacks: CHAP protects against replay attacks by incorporating challenge sequences or timestamps to ensure that responses are valid only for a specific challenge instance.

Mathematical Formulation:

• C: Challenge sent by the authenticator.

• S: Secret known only to the peer (e.g., password).

• H(): Cryptographic hash function used for hashing.

CHAP Process:

• Challenge Phase:

The authenticator sends a challenge *C* to the peer.

• Response Phase:

Peer computes the response *R* by hashing the concatenation of the challenge and the secret: $R=H(C\|S)$

• Verification Phase:

Authenticator computes the expected response ′R′ using the same hash function and the secret associated with the peer. $R'=H(C\|Sstored)$

If ′R=R′, authentication is successful; otherwise, the connection may be terminated.

Key Points:

• Challenge Generation: The challenge C is typically a randomly generated value or a sequence of characters.

• Hashing Function: The hash function H () is a oneway cryptographic function used to generate a fixedlength hash value from the input data.

• Response Calculation: The response R is computed by hashing the concatenation of the challenge and the secret known only to the peer.

• Verification: The authenticator independently computes the expected response ′R′ using its stored copy of the secret. If the computed response matches the response received from the peer, authentication is successful.

Stage II introduces a crucial validation step in the resource allocation process, by applying CHAP algorithm ensuring the security and authenticity of incoming requests. By leveraging SSL handshaking, the system verifies the identity of both the requester and the resource server, establishing a secure channel for communication. Only requests that successfully complete the SSL handshake and pass the certificate verification process are considered valid for resource allocation. This validation mechanism adds an additional layer of security and trustworthiness to the resource allocation process, mitigating the risk of unauthorized access or malicious attacks. By incorporating SSL handshaking into Stage II, the multi-cloud environment can uphold stringent security standards while effectively managing and allocating resources to processes.

Stage III- Finding the best resource by using artificial immune system technique based on fitness function.

Objective- To find the best suited artificial immune system technique that helps to mitigate the resource optimization at cloud environment.

V. Clonal Selection Algorithm (CSA):

CSA is inspired by the clonal selection process of B cells in the biological immune system. It involves creating a diverse set of antibodies representing potential solutions to the optimization problem. Antibodies undergo affinity maturation, where high-affinity antibodies are selected and cloned to improve the overall population. CSA can be adapted for resource optimization by representing resources as antibodies and optimizing their allocation based on affinity to specific tasks or workloads.

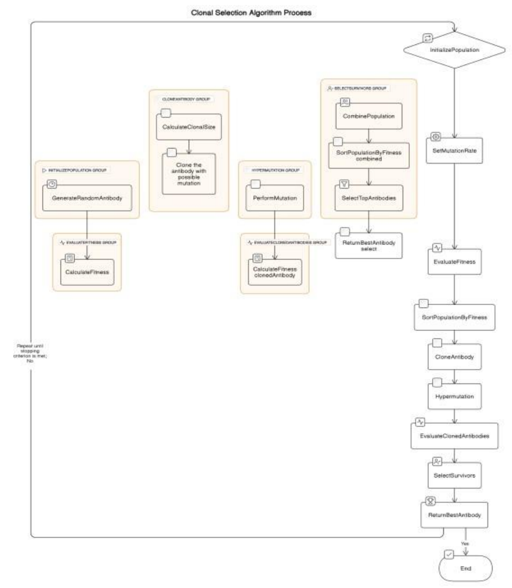


Fig 1.4 Process of Clonal Selection Algorithm (CSA)

Steps for implementation of CSA

Step 1: Initialize parameters like population and mutation rate Step 2: Evaluate the fitness of each antibody in the population and sort population by fitness.

Step 3: Clone the antibody based on its fitness and Introduce diversity through hyper mutation.

Step 4: Evaluate the fitness of cloned antibodies and select antibodies for the next generation.

Step 5: Repeat step 4 until find the antibody with the highest.

fitness in the population **Formulation of fitness function:**

Let's consider a scenario where the objective is to minimize resource wastage while ensuring that the workload is adequately supported. We can define the fitness function f(x) as follows:

$$f(x)=\frac{1}{\text{Total Resources Used}(x)+\epsilon}$$

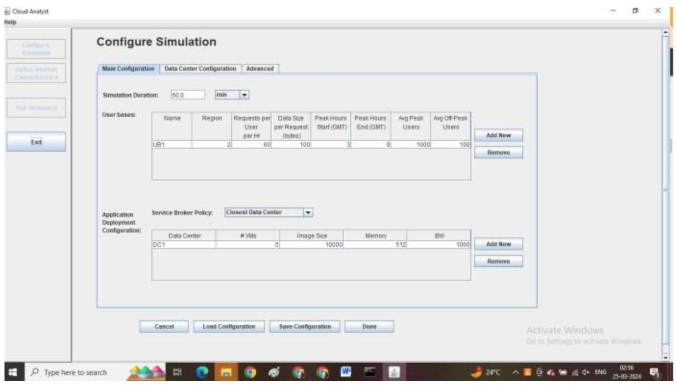
Where:

- x represents an antibody (solution) in the population, which represents a resource allocation configuration.
- Total Resources Used(x) denotes the total amount of resources used by the system under the configuration x.
- ϵ is a small positive value to avoid division by zero errors and provides a degree of smoothing.

This fitness function is formulated such that it utilizes fewer resources receive higher fitness scores. By minimizing the total resources used, the algorithm aims to mitigate resource wastage in the cloud environment.

VI. Simulation Result

After the successful implantation of all three phases first start looking at the output by stage I. As shown in figure 1.5 before applying the DRAUR method the configure simulation phase of cloud analyst tool.



Configure Simulation

Simulation Duration: 60.0 min

User bases:

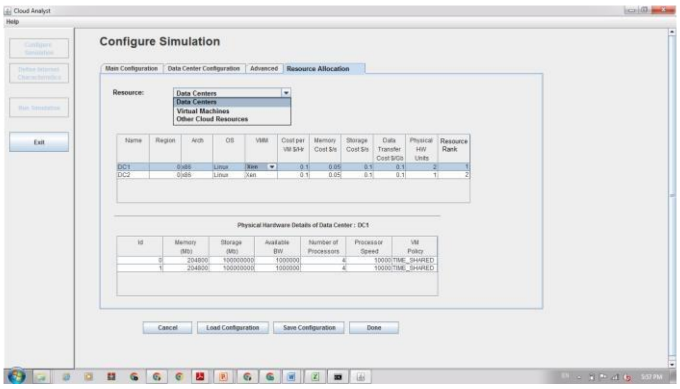
Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	2	60	100	3	9	1000	100

Application Deployment Configuration:

Service Broker Policy: Closest Data Center

Data Center	# VMs	Image Size	Memory	BW
DC1	5	10000	512	1000

Fig 1.5 configures simulation phase at cloud analyst before applying stage I



Configure Simulation

Resource: Data Centers

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units	Resource Rank
DC1	0	x86	Linux	Xen	0.1	0.05	0.1	0.1	2	1
DC2	0	x86	Linux	Xen	0.1	0.05	0.1	0.1	1	2

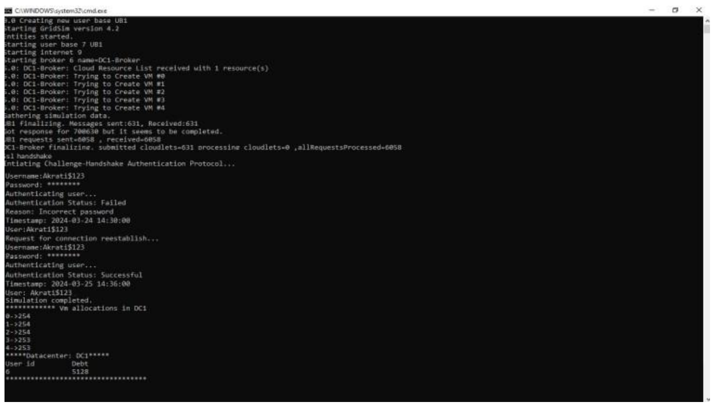
Physical Hardware Details of Data Center : DC1

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	10000	TIME_SHARED
1	204800	100000000	1000000	4	10000	TIME_SHARED

Fig 1.6 Configure simulation phase at cloud analyst after applying stage I

As shown in figure 1.6 after applying the DRAUR algorithm it produces the rank of resources which helps the processes to select the resource with highest rank for better utilization of resources.

After stage I, stage II will come into the scene, which is responsible for the SSL handshaking, for the same Challenge Handshake Authentication Protocol (CHAP) [12] is used which helps to authenticate the user on the datacentre.



```
C:\WINDOWS\system32\cmd.exe
.0 Creating new user base UB1
tarting GridSim version 4.2
ntities started.
tarting user base 7 UB1
tarting internet 9
tarting broker 6 name=DC1-Broker
.0: DC1-Broker: Cloud Resource List received with 1 resource(s)
.0: DC1-Broker: Trying to Create VM #0
.0: DC1-Broker: Trying to Create VM #1
.0: DC1-Broker: Trying to Create VM #2
.0: DC1-Broker: Trying to Create VM #3
.0: DC1-Broker: Trying to Create VM #4
athering simulation data.
B1 finalizing. Messages sent:631, Received:631
ot response for 700630 but it seems to be completed.
B1 requests sent=6058 , received=6058
C1-Broker finalizing. submitted cloudlets=631 processing cloudlets=0 ,allRequestsProcessed=6058
sl handshake
ntiating Challenge-Handshake Authentication Protocol...
Username:Akrati$123
Password: ********
Authenticating user...
Authentication Status: Failed
Reason: Incorrect password
Timestamp: 2024-03-24 14:30:00
User:Akrati$123
Request for connection reestablish...
Username:Akrati$123
Password: ********
Authenticating user...
Authentication Status: Successful
Timestamp: 2024-03-25 14:36:00
User: Akrati$123
Simulation completed.
************ Vm allocations in DC1
0->254
1->254
2->254
3->253
4->253
*****Datacenter: DC1*****
User id         Debt
6               5128
**********************************
```

Fig 1.7 Authentication process of CHAP algorithm

As shown in figure 1.7 the simulation analysis of cloud analyst on command prompt. it shows the authentication procedure after applying the CHAP algorithm [10], it shows two conditions:

- If entered username and password will be wrong the authentication status will presents as failure and it will not allow to enter the user in the cloud data centre,
- Second when the entered username and password will be correct and, matched with credentials of authenticated users then system will authenticate the user and show the status as success and after that user will allow accessing the data centres [9].

Stage III will deal with the implementation of artificial immune system method for the checking the fitness function of resources so that the optimized resource should be properly utilized. For the same clonal selection algorithm (CSA) [14,15] will be used. The performance of the algorithm will be calculated based on these four parameters:

- **Sphere Function:**

It is Simple, convex, and unimodal, it provides a basic test of an algorithm's ability to find the global minimum in a smooth and well-defined search space.

2 i $\boldsymbol{f(x) = \sum_{i=1}^{n} x}$

- **Rastrigin's Function:**

With many local minima and a rugged landscape, it challenges optimization algorithms to navigate through complex, high-dimensional search spaces and find the global minimum.

$\boldsymbol{f(x) = A_n + \sum_{i=1}^{n}(x_i^2 - A\cos(2\pi x_i))}$**, where A=10**

- **Ackley's Function:**

Multimodal and highly non-linear, it tests an algorithm's ability to balance exploration and exploitation, avoiding local minima in favor of the global minimum.

$\boldsymbol{f(x) = -20\exp(0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos\ 2\pi x_i) +}$

20 e

 Modified Sinusoidal Function:

: By introducing noise to a sinusoidal function, it evaluates the robustness and noise tolerance of optimization algorithms, as well as their ability to handle non-smooth objective functions**.**

$$\boldsymbol{f(x) = \sin(x)} + \in$$

where ϵ represents noise or perturbations.

Figure 1.8 represents the above-mentioned parameters on two dimensions functions; a. represents the two-dimension function of Sphere Function; b. represents the two-dimension function of Rastrigin‟s Function. c. represents the two-dimension function of Ackley‟s Function, and d. represents the two-dimension function of Modified Sinusoidal Function.

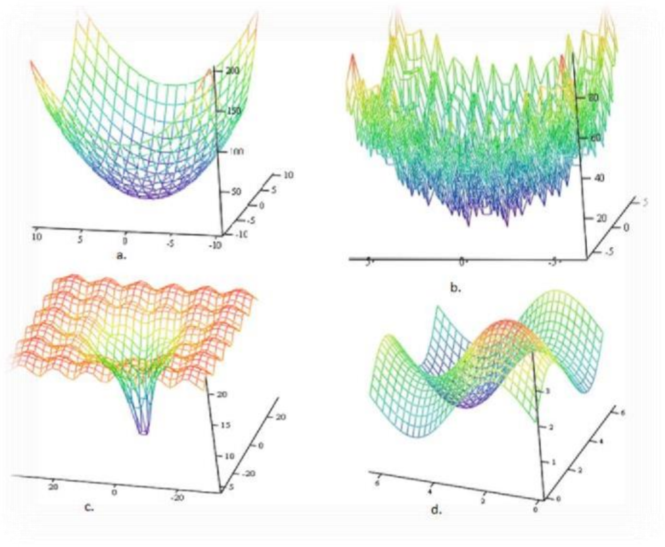


Fig 1.8 Two dimensional functions of parameters

Conclusion

The development and evaluation of a pioneering load balancing approach tailored specifically for addressing resource contention challenges within multi-cloud environments. Through meticulous experimentation and analysis, we have demonstrated the efficacy of our proposed solution in mitigating resource contention and enhancing overall system performance by adding in depth three stages approach to balance the resource utilization. Stage I focuses on rank the resource based on their availability, best ranked resource can further used, stage II focuses on the ssl handshaking for authenticating the user so that no fake user can enter in the system. Finally stage III deals with implantation of AIS technique so that it helps to minimize the resource utilization.

References

[1] B. Wickremasinghe, “CloudAnalyst: A CloudSim-based Tool for Modeling and Analysis of Large-Scale Cloud Computing Environments”, Volume 68 issue 50, June 2009. [2] T. Brunzel, D. Giacomo, “Cloud Computing Evaluation - How it Differs to Traditional IT Outsourcing”, Volume 32 issue 16, May 2010.

[2] K. Jeffery, L. Schubert, “Advances in Clouds Expert Group Report Public” Version 1.0, Research in Future Cloud Computing, 2012.

[3] J. Kaur, “Comparison of load balancing algorithms in a Cloud”, International Journal of Engineering Research and Applications, Volume 2, Issue 3, May-Jun 2012.

[4] M. A. Adnan, R. Sugihara, and R. K. Gupta, “Energy Efficient Geographical Load Balancing via Dynamic Deferral of Workload”, in Proc. of IEEE Cloud. 188-195, 2012.

[5] M. Zhou, “Data Security and Integrity in Cloud Computing”, University of Wollongong, October 2014.

[6] S. Sharma, A. Chugh, “Survey Paper on Cloud Storage and Security”, International Journal of Innovative Researching Computer and Communication Engineering, Volume 1, Issue 2, April 2013.

[7] B. Wickremasinghe N. Calheiros, and R. Buyya, “Cloudanalyst A Cloudsim based Visual Modeller for Analyzing the Cloud Computing Environments and Application”, Pontifical University of Rio Grande do Sul Porto Alegre, Brazil, 2015.

[8] J. Chen, G. Soundararajan, S. Ghanbari, F. Iorio, A. Hashemi, C. Amza, “Ensemble: A Tool for the Performance Modeling of Applications in Cloud Data Performance Centers” DOI 10.1109/TCC.2015.2469656, IEEE

[9] Transactions on Cloud Computing, 2012.

[10] An-Ping, X & Chun-Xiang, X, „Energy efficient multi resource allocation of Virtual Machine based on PSO in cloud data center‟, Mathematical Problems in Engineering, vol. , pp. 1- 8,2014.

[11] Arianyan, E, Maleki, D, Yari, A & Arianyan, I, „Efficient resource allocation in cloud data centers through genetic algorithm‟, 6th International Symposium on Telecommunications (IST); Tehran. pp. 566–70, 2012.

[12] Armstrong, TR & Hensgen, D, „The relative performance of various mapping algorithms is independent of sizable variances in runtime predictions‟, Proc. of 7th IEEE Heterogeneous Computing Workshop (HCW 98), pp. 79-87, 1998, Awad, AI, El-Hefnawy, NA & Abdel Kader, HM,

[13] Dynamic Multiobjective task scheduling in Cloud Computing based on Modified particle swarm optimization‟, Advances in Computer Science: An International Journal, vol. 4, no. 17, pp. 110-117, 2015.

[14] L. De Castro, F. J. Von Zuben, „The Clonal selection algorithm with engineering applications, ‟ GECCO 2000, Workshop Proceedings, Workshop on Artificial Immune Systems and Their Applications, Las Vegas, USA, 36-37 2000.

[15] L. N. De Castro, F. J. Von Zuben, „Learning and Optimization Using the Clonal Selection Principle, ‟ IEEE Transactions on Evolutionary Computation, Vol. 6, No, pp. 239-251. 2002

[16] D. Karaboğa, B. Akay, „A comparative study of Artificial Bee Colony algorithm, ‟ Applied Mathematics and Computation 214, Elsevier, pp. 108-132, 2009