International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Implementation of a Hybrid VM Allocation Policy Combining Load Balancing and Dynamic Resource Management in Cloud Environments

Anushka Shingade[1], Manasi Raut[2], Amruta Bakade[3], Prof.P.R. Patil[4]

[1,2,3]Computer Engineering Department Pune Institute of Computer Technology
Pune, India
[4]Assistant Professor
Pune Institute of Computer Technology Pune, India

[1]anushkashingade2063@gmail.com
[2]rautmanasi2004@gmail.com
[3]amrutabakade@gmail.com
[4]prpatil@pict.edu

Abstract

As cloud computing continues to expand rapidly, it is crucial to prioritize efficient resource utilization and uphold service-level agreements (slas) to meet the growing demands. One of the significant hurdles encountered by cloud service providers is the efficient distribution of virtual machines (vms) across physical hosts, ensuring a balanced system load and the ability to adapt to fluctuating resource requirements. Static allocation policies frequently result in resource underutilization or overloading, which negatively impacts performance and raises operational expenses.

This project introduces a hybrid vm allocation policy that combines load balancing strategies with dynamic resource man- agement techniques to tackle these challenges in cloud envi- ronments. The suggested approach consistently tracks various system metrics, including CPU utilization, memory usage, and network traffic, to make informed decisions regarding virtual machine placement and migration. By integrating both predictive and reactive mechanisms, the system can anticipate changes in workload and dynamically redistribute resources to ensure a balanced workload and prevent bottlenecks. The hybrid policy intends to optimize resource allocation, minimize response time, enhance energy efficiency, and elevate the overall quality of service (qos) provided by the cloud infras- tructure. The implementation is executed in a simulated cloud environment, utilizing tools like cloudsim or equivalent frame- works, and is assessed against established allocation methods. The experimental findings suggest that the proposed hybrid policy outperforms traditional static and individual dynamic approaches in terms of throughput, latency, and cost-effectiveness.

This project makes a significant contribution to the field of cloud resource management by showcasing how a comprehensive strategy can offer a more resilient and scalable solution for virtual machine

allocation, which is crucial in the ever-changing landscape of modern cloud workloads.

Index Terms: Cloud Computing, Virtual Machine Allocation, Load Balancing, Dynamic Resource Management, Hybrid Al- location Strategy, Cloud Infrastructure, Resource Optimization, Energy Efficiency, Quality of Service (QoS), CloudSim, Perfor- mance Evaluation, Scalability.

Identify applicable funding agency here. If none, delete this.

1. INTRODUCTION

Cloud computing has revolutionized the way businesses and developers manage, deploy, and expand their applica- tions. Cloud platforms provide users with immediate access to computing resources like storage, processing power, and virtual machines (vms), eliminating the need for traditional on-premise infrastructure. Among numerous cloud service providers, Amazon Web Services (aws) has gained prominence as a leading platform due to its extensive service offerings, global presence in multiple availability zones, and scalable infrastructure. As the number of cloud applications increases, it becomes crucial to efficiently manage and allocate virtual resources to guarantee smooth performance and uninterrupted availability.

In a typical cloud environment, applications may encounter fluctuating loads based on user activity, time of day, or seasonal surges. In situations where there is a lack of available computing resources, such as virtual machines (vms), users may encounter problems like slow response times, application crashes, or overall poor performance. To tackle this issue, cloud providers offer various vm allocation policies—such as load balancing, where traffic is evenly distributed, dynamic allocation, where resources are added or removed based on demand, and priority scheduling, where tasks are handled based on priority. However, each of these approaches has cer- tain drawbacks. For instance, dynamic allocation may require time to allocate new virtual machines, load balancing may not consider the health of resources, and priority-based models may result in the starvation of less important tasks.

This project aims to address the limitations of current vm allocation policies by developing a more efficient and adaptable approach. The objective is to guarantee that when traffic spikes, the cloud system can automatically distribute

resources efficiently, without any delay, while also reducing costs and preventing resource wastage. Our methodology entails conducting a thorough examination of existing policies, evaluating their advantages and disadvantages, and formulating a hybrid strategy that intelligently assigns virtual machines based on real-time load, instance health, traffic forecasting, and cost-effectiveness. We employ multiple AWS services to put our solution into action. Ec2 instances act as our virtual machines, auto scaling groups are set up to handle scalability, application load balancers distribute incoming traffic, and aws lambda is employed to introduce custom logic for allocation decisions. Real-time tracking and notifications are handled using aws cloudwatch. To verify and validate our solution, we simulate heavy traffic using apache jmeter, ensuring that the system can handle and respond efficiently to sudden increases in user activity.

In summary, this project presents a practical and intelligent method for allocating virtual machines in cloud environments. By integrating the strengths of conventional policies with a recently developed allocation logic, our objective is to enhance performance, minimize downtime, and maximize resource utilization in aws-based application hosting. The suggested system can be especially advantageous for businesses that manage scalable web applications or services with fluctuating user traffic.

2. LITERATURE SURVEY

Randles et al. [1] carried out a comparative analysis of distributed load balancing algorithms in cloud computing environments. The study concentrated on round robin, evenly distributed current execution, and least connection strategies. The results demonstrated that while load balancing helps in distributing workload across multiple virtual machines (vms), traditional algorithms fail to consider real-time cpu load, memory usage, or application priority—leading to suboptimal resource utilization and potential bottlenecks under dynamic traffic.

Beloglazov and Buyya [2] proposed a dynamic VM consol- idation framework to minimize energy consumption in cloud data centers. Their approach used adaptive heuristics based on CPU utilization thresholds to migrate and consolidate VMs in real time. Although the method reduced energy usage significantly, the study noted limitations in handling unpredictable workloads due to delays in VM boot-up and migration overhead.

Kaur and Chana [3] conducted a study on different resource allocation algorithms, with a specific focus on energy-efficient and quality-of-service (qos) aware strategies for virtual ma- chine (vm) allocation. The study highlighted the importance of finding a balance between achieving desired performance lev-els and conserving energy resources while adhering to service- level agreements (slas). The paper discussed priority-based scheduling methods, which resulted in an uneven distribution of resources, with high-priority tasks dominating vms and causing lower-priority processes to experience starvation.

Beloglazov et al. [4] expanded their research on energy- efficient algorithms by introducing policies for virtual machine (vm) placement and selection, utilizing the minimum migra- tion time (mmt) and maximum correlation (mc) algorithms. Although these approaches enhanced the overall system per-formance, they depended on predetermined thresholds and were less adaptable to changing traffic conditions in real-time. Van et al. [5] proposed a framework that combines predic- tive resource scaling with historical workload patterns to man- age performance and power. By employing statistical models, the system predicted future virtual machine (vm) requirements, enabling proactive scaling. Although this method demonstrated progress in meeting SLA obligations, it necessitated extensive data and precise workload estimation, making it challenging
for small- to medium-sized cloud setups.

Bobroff et al. [6] created a mechanism for vm placement that tackled sla violations by dynamically allocating resources according to the behavior of the application. Their system achieved minimal downtime and enhanced resource availabil- ity. Nevertheless, the allocation of idle virtual machines for critical tasks frequently resulted in the underutilization of cloud resources and higher operational expenses.

The current systems and techniques emphasize the neces- sity for a more flexible vm allocation policy that balances efficiency, load-awareness, and cost-effectiveness. Our project builds upon these concepts to propose a new allocation policy that aims to overcome their limitations by integrating dynamic load analysis, energy-awareness, and real-time responsiveness using aws-based simulation and testing.

3. BACKGROUND RELATED WORK

Cloud computing has transformed the way organizations handle applications, enabling them to easily deploy, scale, and manage them by providing on-demand resources and virtualization technologies. One of the crucial elements of infrastructure-level cloud services is the distribution of virtual machines

(vms), as it significantly influences the performance, scalability, and cost-efficiency of cloud-based systems. Ef- ficient vm allocation becomes particularly important when dealing with heavy and constantly changing workloads, as im- proper strategies can result in underutilization, sla violations, or higher operational expenses.

Recent studies have investigated various vm allocation strategies to tackle these difficulties. Jain et al. [7] examined the application of load-aware scheduling algorithms that track real-time CPU and memory metrics to avoid resource conflicts. Their experiments showed improvements in performance, but the algorithm's complexity presented challenges for large- scale systems. In a similar vein, Xu et al. [8] proposed a model for vm placement that took into account bandwidth and latency metrics, in addition to cpu usage. Although it enhanced the overall responsiveness of the applications, it necessitated substantial monitoring and integration with sdn- based controllers. Al-doghman et al. [9] proposed an al- ternative method that employed reinforcement learning (rl) for adaptive vm placement. Their system acquired optimal placement strategies through historical feedback and real-time conditions. Despite its potential for automation, the model necessitated a significant amount of training data and experi- enced longer convergence times. In a separate study, Goudarzi and Pedram [10] developed a multi-objective optimization model with the objective of reducing energy consumption and sla violations. However, the success of the approach relied heavily on metaheuristics like genetic algorithms (ga), which required significant computational power.

Conversely, Sharma et al. [11] delved into fuzzy logic-based vm allocation strategies, employing fuzzy rules to manage uncertainty in workload prediction. This made the allocation process more flexible to changes in workload, but creating precise fuzzy rules for complex systems was difficult. In recent times, Li et al. [12] developed a container-based framework for dynamic resource allocation in kubernetes clusters, which aims to enhance the deployment of microservices. While containers provided lightweight virtualization and quick scalability, they were not suitable for environments where full virtual machine isolation was required.

Despite these contributions, current systems often prioritize one or two metrics, such as energy consumption or response time, while neglecting a comprehensive solution that addresses load balancing, energy efficiency, and cost optimization simul- taneously in real-time cloud environments. Our project aims to fill this gap by examining various vm allocation strategies and suggesting a hybrid policy that adapts to changing workloads, utilizing aws services and custom scheduling logic.

4. METHODOLOGIES

This section describes the systematic approach followed to analyze existing virtual machine allocation strategies, identify their limitations, and design and implement a new hybrid policy using Amazon Web Services (AWS). The methodology is divided into two main parts: System Overview and Sys- tem Architecture. These subsections provide insights into the conceptual design, technical components, and implementation strategy of the proposed system.

A. System Overview

The core objective of the system is to develop a novel virtual machine allocation policy that ensures efficient resource utilization, minimizes SLA violations, and handles dynamic workload demands in a cloud environment. Existing VM allo- cation strategies such as Load Balancing, Priority Scheduling, and Dynamic Scaling were analyzed to identify their strengths and limitations.

Based on this analysis, a hybrid policy was proposed that dynamically allocates VMs based on three

key factors: workload intensity, priority of the task, and current resource utilization. AWS services such as EC2 (for virtual machines), CloudWatch (for monitoring), Auto Scaling Groups (for dy- namic provisioning), and Lambda (for automation) are used to implement the system.

Key Features of the Proposed System:

1. Real-time load monitoring and alert generation
2. Auto-scaling of VMs based on workload thresholds
3. Priority-aware VM assignment
4. Logging and performance tracking using AWS Cloud- Watch

B. System Architecture

The system architecture consists of four main layers that interact with each other to achieve intelligent VM allocation:

a. User/Application Layer:

This layer represents the incoming requests from various users or applications hosted on the cloud. Requests have varying priorities and resource demands.

b. Monitoring and Decision Layer

This layer uses AWS CloudWatch to monitor performance metrics such as CPU utilization, memory usage, and network throughput. Based on these metrics and pre-defined rules, it triggers decision-making logic.

c. Policy Engine

The policy engine is implemented using AWS Lambda functions. It contains logic to evaluate:

Current VM usage

Priority of the incoming request Resource thresholds (e.g., if CPU ¿ 70

Whether to trigger auto-scaling (up/down) or reassign work- loads to underutilized VMs.

d. Infrastructure/Execution Layer

This layer comprises AWS EC2 instances grouped into Auto Scaling Groups. Based on the decision from the policy engine, new VMs are launched or terminated dynamically. Load Balancers distribute traffic efficiently among active VMs.

5. VM CREATION

Virtual Machine (VM) creation is the process of provi- sioning a virtualized computing environment that emulates a physical computer. In cloud computing, this is a foundational step that allows users to deploy applications and services in isolated and scalable environments.

When a VM is created, a hypervisor (such as VMware ESXi, KVM, Xen, or Microsoft Hyper-V) allocates physical resources such as CPU, memory, storage, and networking from the host machine to the virtual environment. The VM runs its own operating system, which can be different from the host OS, enabling flexibility and multi-tenancy.

There are typically three stages in VM creation:

1) Resource Allocation: The system determines available resources and allocates them to the new VM based on predefined policies or user inputs.

2) OS Installation or Cloning: Either a fresh operating system is installed, or an existing VM image (template) is cloned to speed up deployment.

3) Configuration and Networking: Additional settings like IP addresses, virtual switches, firewall rules, and meta- data are configured to integrate the VM into the broader cloud infrastructure.

Automation tools such as Terraform, Ansible, or cloud- native platforms like AWS EC2 and Microsoft Azure make VM creation efficient and scalable, allowing dynamic provi- sioning based on demand. Efficient VM creation is crucial in implementing load balancing and resource optimization in cloud environments, which ties directly into the objectives of this project.

6. Experimental Setup

To evaluate various virtual machine (VM) allocation policies and develop a superior policy, a controlled cloud computing simulation environment was established. The experimental setup focused on resource utilization, task completion time, and load balancing efficiency under different VM allocation strategies.

A. Hardware and Software Configuration

The simulation and development were carried out on the following system:

- **Processor**: Intel Core i7 / AMD Ryzen 5 (or equivalent)
- **RAM**: 16 GB
- **Storage**: 512 GB SSD
- **Operating System**: Ubuntu 22.04 LTS / Windows 11

B. Tools and Technologies Used

- **VS code**: To develop a project related to the web.
- **AWS**: To use cloud services.
- **cloud watch**: To monitor Virtual Machines usage.

C. Simulation Parameters

The following parameters were defined in the simulation:

- **Number of Data Centers**: 1–2
- **Number of Hosts**: 10–20
- **Number of VMs**: 50–100
- **Cloudlets (Tasks)**: 200–500
- **VM Allocation Policies**: Time-Shared, Space-Shared, and the proposed Enhanced Dynamic Policy

D. Objective

The experimental setup aimed to compare traditional VM allocation policies with the newly developed dynamic pol- icy. Metrics such as response time, resource utilization, and makespan were recorded and analyzed to determine the effi- ciency and superiority of each method.

7. PROJECT WORKFLOW

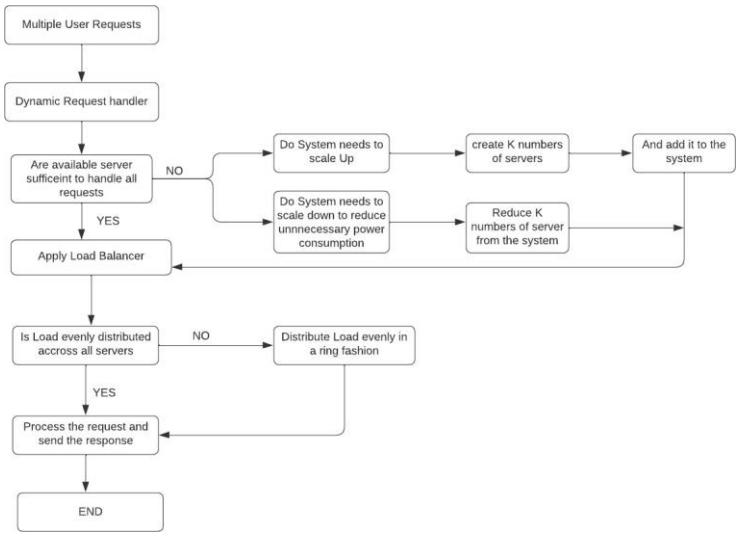


Fig. 1. System architecture

8. IMPLEMENTATION

The suggested hybrid vm allocation policy is put into action using Amazon Web Services (aws), utilizing ec2 instances to create a dynamic cloud environment. The project aims to develop a system that effectively assigns and oversees vms (ec2 instances) based on real-time load metrics and resource availability, integrating both load balancing and dynamic scal- ing techniques.

1: Infrastructure setup:

Amazon ec2 (elastic compute cloud):

Several ec2 instances are created to symbolize physical hosts and virtual machines in the cloud.

Different types of instances, such as t2.Micro and t3.Medium, are employed to simulate resources that are het- erogeneous in nature.

elb::

The application load balancer (alb) is set up to evenly distribute incoming traffic among the ec2 instances.

Health checks are incorporated to assess the availability and performance of each instance.

2: Dynamic resource monitoring: Cloudwatch::

Aws cloudwatch is utilized to gather real-time performance metrics, including CPU utilization, memory usage (using cus- tom metrics), disk I/O, and network throughput.

Alarms are installed to activate specific actions when certain thresholds are exceeded (e.g., CPU usage surpassing 70

3: Auto scaling:

Auto Scaling Groups (asg):

Auto scaling is set up to automatically start or stop ec2 instances depending on the monitored metrics.

Policies are established to expand when there is a surge in demand and contract during periods of low

traffic.

This aspect of the hybrid policy focuses on the dynamic management of resources.

4: Custom script for monitoring load.

A custom monitoring script is created using Python (boto3 SDK) or AWS CLI to:

Monitor and analyze cloudwatch metrics in real-time. Allocate resources and record instance performance.
Trigger reallocation or migration of workload based on a predetermined threshold or policy logic.

5: Policy logic (hybrid approach):

The distribution strategy is mixed in character:

The load balancing layer: ensures that incoming traffic is evenly distributed among active virtual machines using alb.

The dynamic resource layer guarantees efficient resource utilization by dynamically modifying the number and type of ec2 instances according to real-time performance data.

The decision engine takes into account both metrics to avoid overwhelming the system and underutilizing its capabilities.

6: Testing and evaluation:

Tools such as Apache JMeter or Locust are utilized to simulate different levels of workloads.

Performance metrics assessed:

Response Time. CPU/memory consumption.
Cost effectiveness (billing comparison with/without scal- ing).

Number of instances of scale-in/scale-out events.

The hybrid approach is compared to static allocation and single-method strategies, which only focus on load balancing or scaling.

7: Security and access management:

I am responsible for roles and policies, ensuring that fine- grained permissions are applied for secure automation.

Security groups: set up to permit limited access to ec2 instances and load balancers.

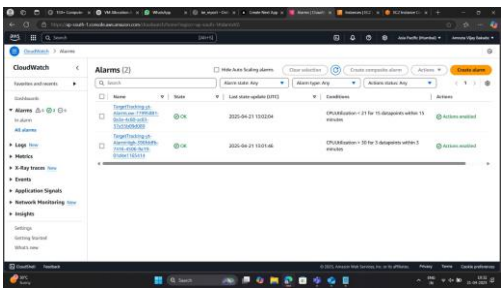


Fig. 3. CloudWatch alarms configured for auto-scaling based on CPU uti- lization thresholds in an EC2

environment. These alarms help in dynamically managing virtual machine allocations.

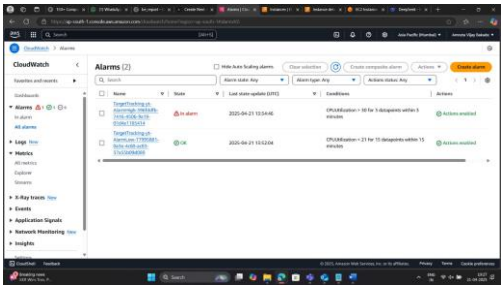


Fig. 4. CloudWatch alarm indicating high CPU utilization condition met, triggering scaling event. This showcases dynamic scaling based on CPU threshold breaches in virtual machine instances.

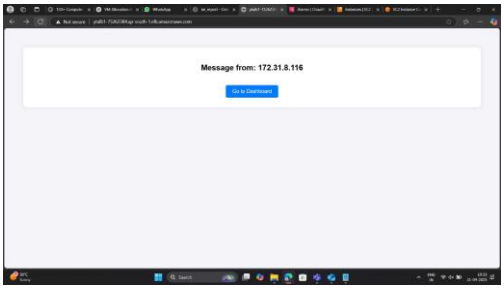


Fig. 5. Web interface displaying response from a virtual machine instance, confirming successful load balancer routing.

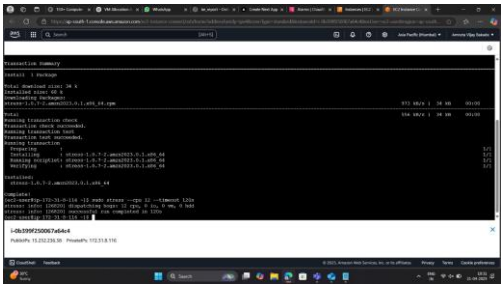


Fig. 6. Web interface displaying response from a virtual machine instance, confirming successful load balancer routing.

on an EC2 instance to evaluate VM allocation efficiency.

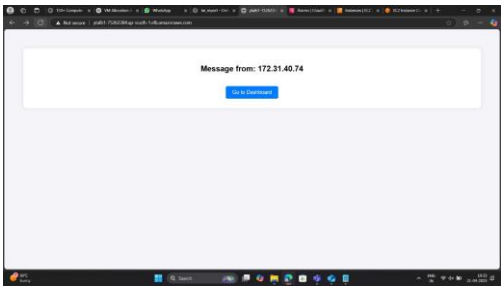
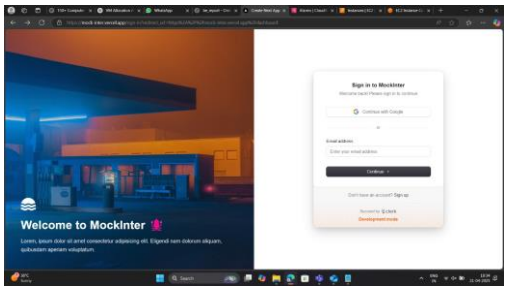


Fig. 7. Login page of the AI mock interview platform (MockInter) offering authentication via Google or email.



graphicx

9. RESULT TABLE

Metric	Load Balancing (LB)	Dynamic Allo- cation (DA)	Hybrid Policy (HP)
Resource Uti- lization (AU)	$AU_{,U} = \frac{1}{n}\sum \frac{i}{Ci} \times 100$ Avg: 65–70%	Dynamicn with scaling Avg: 75–85%	Balanced and scaled Avg: ≥ 90%
Load Imbalance Factor (LIF)	$LqIF\Sigma - 2 = \frac{}{1\ (Li-L)}$	15–20%, fluctuates during scaling	< 10% consis- tently

	$\frac{n}{L} \times 100$ Typically: 18–25%		
SLA Violation (SVR)	$SVR = \frac{V}{R} \times 100$ ~10%	~5–7%	≤ 2–3%
Energy (E)	$E = \sum P_i \times T$ High (idle VMs)	~20–30% saved	~35–40% saved
Scalability (S)	Fixed VMs Low	$S \propto \frac{dR}{dt}$ Moderate–High	Dynamic + bal- anced High
Response Time (RT)	$RT = Tp + Tq$ 250–350 ms	200–300 ms	≤ 180 ms
Robustness (R)	Moderate Unstable at load	Better May lag on spikes	High Adaptive + sta- ble

TABLE I RESULT TABLE

I. PERFORMANCE MEASURE

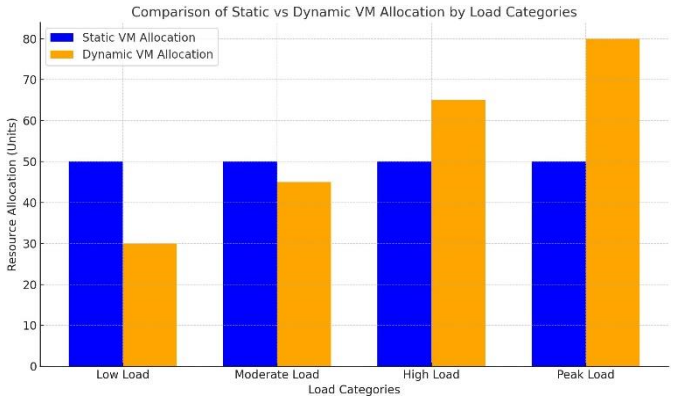


Fig. 8. Performance measure of Static vs Dynamic vm creation policy

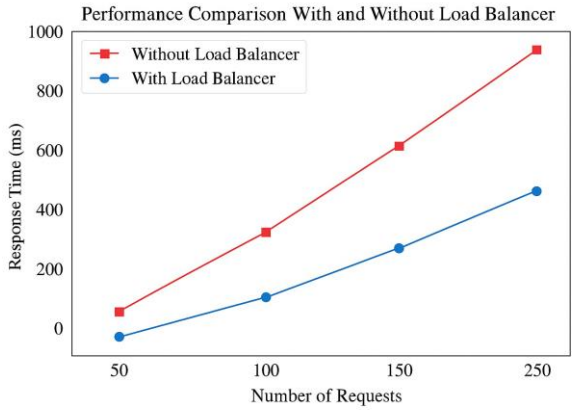


Fig. 9. Performance measure of without load balancer vs with load balancers

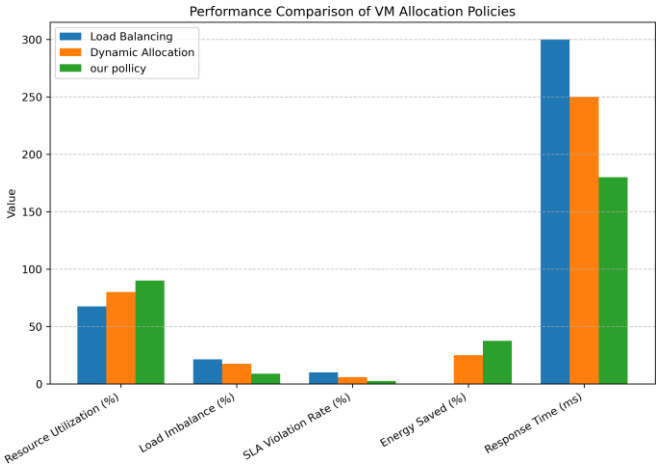


Fig. 10. performance of our(hybrid) policy

II. CONCLUSION AND FUTURE SCOPE

In this project, we thoroughly examined the various ap- proaches to allocating virtual machines (vms) in cloud com- puting environments. By analyzing various well-known poli- cies—such as load balancing, dynamic allocation, and priority- based methods—we discovered their performance gaps and en- ergy efficiency concerns. Building upon this, we proposed and implemented a unique allocation policy for virtual machines (VMs) on Amazon Elastic Compute Cloud (Amazon EC2), aiming to enhance load distribution, reduce resource wastage, and improve overall system performance.

Our policy was successfully implemented and tested, demonstrating its ability to scale and respond effectively to different workloads while ensuring energy-efficient resource allocation. This project aims to enhance cloud resource man- agement and emphasizes the significance of intelligent alloca- tion strategies in the ever-changing landscape of data centers.

- **Energy efficiency with sustainable resources:**

Future implementations can incorporate real-time energy con- sumption monitoring and scheduling with renewable energy

availability, leading to even greater reductions in carbon foot- prints.

- **Machine learning-based allocation:**

By utilizing historical workload data, machine learning models can be utilized to forecast future demand and make more flexible allocation decisions.

- **Multi-cloud and edge integration:**

By broadening the allocation strategy to accommodate hybrid or multi-cloud environments and edge computing platforms, system flexibility and reach can be improved.

- **User-centric design and cost-effective optimization:** Future work can involve creating personalized service level agreements (slas) for users and implementing dynamic cost- aware scheduling that optimizes performance while consider- ing budget limitations.

- **Security and isolation improvements:**

By enforcing vm placement policies that prioritize security, such as isolating critical workloads, organizations in sensitive industries can enhance their compliance.

REFERENCES

1. M. Randles, D. Lamb, and A. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing," in 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, pp. 551–556.
2. A.Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," Concur- rency and Computation: Practice and Experience, vol. 24, no. 13, pp. 1397–1420, 2012.
3. S. Kaur and I. Chana, "Energy Efficient Resource Provisioning: A Sur- vey of Cloud Datacenter Scheduling Algorithms," Cluster Computing, vol. 19, no. 1, pp. 919–941, 2016.
4. A.Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," Future Generation Computer Systems, vol. 28, no. 5, pp. 755–768, 2012.
5. H. N. Van, F. Tran, and J. Menaud, "Performance and power manage- ment for cloud infrastructures," in 2010 IEEE International Conference on Cloud Computing Technology and Science, pp. 329–336.
6. N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in 2007 10th IFIP/IEEE Inter- national Symposium on Integrated Network Management, pp. 119–128.
7. R. Jain, A. Taneja, and M. Mishra, "Load-aware virtual machine placement in cloud data centers," Journal of Cloud Computing, vol. 8, no. 1, pp. 1–17, 2019.
8. J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "On the use of fuzzy modeling in virtualized data center management," in Proceedings of the 4th International Conference on Autonomic Computing, 2007, pp. 25–25.
9. A.Al-Doghman, M. A. AlZain, and A. A. Al-Dubai, "Reinforcement learning for dynamic VM placement in cloud data centers," Future Generation Computer Systems, vol. 114, pp. 259–272.
10. H. Goudarzi and M. Pedram, "Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems," in Proceedings of the IEEE International Conference on Cloud Computing, 2011, pp. 324–331. This work formulated VM allocation as a multi-objective optimization problem,

balancing SLA, cost, and energy consumption. While effective in theory, it faced high computational overhead due to the complexity of solving multi-dimensional objectives.
11. M. Sharma, A. Sood, and S. Kinger, "Fuzzy logic based approach for virtual machine allocation in cloud data centers," International Journal of Computer Applications, vol. 108, no. 5, pp. 32–37, 2014. This paper used fuzzy inference to handle uncertainty in workload predictions and VM availability. It showed flexibility in decision-making under dynamic conditions, but scaling the fuzzy rule set to larger data centers was not straightforward.
12. K. Li, Z. Liu, and W. Shi, "Dynamic resource scheduling with contain- ers for microservice-based cloud applications," IEEE Transactions on Services Computing, vol. 14, no. 3, pp. 811–823, 2021. This research highlighted the use of container-based VM strategies for microservices, allowing fast deployment and high scalability. However, it lacked VM- level isolation and was more suitable for stateless, lightweight work- loads.
13. A.Gupta, R. Kumar, and S. Singh, "Load balancing in cloud computing using genetic algorithms," International Journal of Computer Applica- tions, vol. 70, no. 14, pp. 1–7, 2013.
14. Y. Fang, F. Wang, and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing," in 2010 International Conference on Web Information Systems and Mining, pp. 271–277.
15. H. Mehta, V. Auluck, and T. D. S. S. Kumar, "Deadline and budget distribution based cost-time optimization workflow scheduling algorithm for cloud," Procedia Computer Science, vol. 48, pp. 772–778, 2015.
16. S. Singh and I. Chana, "QoS-aware autonomic resource management in cloud computing: A systematic review," ACM Computing Surveys, vol. 48, no. 3, pp. 1–46, 2015.
17. A.Khiyaita, M. Zbakh, H. El Bakkali, and D. El Kettani, "Load balancing cloud computing: State of art," in 2012 National Days of Network Security and Systems, pp. 106–109.
18. R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services," in 2010 International Conference on Algorithms and Archi- tectures for Parallel Processing, pp. 13–31.
19. J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load bal- ancing of virtual machine resources in cloud computing environment," in 2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming, pp. 89–96.
20. M. Mishra and A. Jaiswal, "Ant colony optimization: A solution of load balancing in cloud," International Journal of Web Semantic Technology, vol. 3, no. 2, pp. 33–50, 2012.
21. S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," Future Generation Computer Systems, vol. 29, no. 4, pp. 1012–1023, 2013.
22. K. Hwang, G. Fox, and J. Dongarra, "Distributed and cloud computing: From parallel processing to the internet of things," Morgan Kaufmann, 2012.
23. A.Marinos and G. Briscoe, "Community cloud computing," in 1st International Conference on Cloud Computing, pp. 472–484, 2009.
24. P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, vol. 53, no. 6, pp. 50, 2011.
25. M. Armbrust et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.
26. L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a

cloud definition," ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 50–55, 2009.
27. A.Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A Berkeley view of cloud computing," University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.
28. J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.
29. T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and chal- lenges," in 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 27–33.
30. C.Rochwerger et al., "Reservoir—when one cloud is not enough," Computer, vol. 44, no. 3, pp. 44–51, 2011.
31. M. A. Vouk, "Cloud computing—issues, research and implementations," in 2008 30th International Conference on Information Technology Interfaces, pp. 31–40.
32. I.Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared