IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

A Cloud Secure Storage Mechanism Based On Fault And Load Balancing

[1] Miss. Rajshree Devidas Patil , [2] Mr.Hirendra Hajare
[1] M.Tech. Scholar, Assistant Professor
[1]Computer Science & Engineering,
Ballarpur Institute of Technology,India

Abstract: What if one day u wake up and your application starts getting a lot of unexpected traffic? This is amazing, isn't it? But is your application ready for it? Can it handle so much traffic? In this project we propose a Elastic load balancer. ELB (Elastic Load Balancer) is a service provided by AWS (Amazon Web Services) which is used to balance the incoming load (traffic) on multiple EC2 instances which can be in multiple availability zones. Load balancing distributes the network traffic in such a manner that increases speed and the performance of web applications. Modern ELB uses "Round robin algorithm". The real time applications of ELB are high traffic websites, E- commerce applications, Media streaming, Gaming applications, Mobile applications. ELB improves scalability, availability and fault tolerance of web applications and services.

Keywords–Elastic Load Balancer, Amazon Web Services, Elastic Compute Cloud, Round Robin, Scalability.

I. INTRODUCTION

Amazon Web Services (AWS) is a subsidiary of Amazon that provides on- demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis. Oftentimes, clients will use this in combination with autoscaling (a process that allows a client to use more computing in times of high application usage, and then scale down to reduce costs when there is less traffic). These cloud computing web services provide various services related to networking, compute, storage, middleware and other processing capacity, as well as software tools via AWS server farms. This frees clients from managing, scaling, and patching hardware and operating systems

Servers
Mobile Development
Networking
amazon web services
Security
Storage
Email
Remote Computing
Mobile

Figure 1.1 Amazon web services

AWS Services

Reliability and Availability: AWS offers high availability and redundancy through its data centers, ensuring minimal downtime and improved reliability for your applications.

Security: AWS provides robust security features and tools to help you protect your data and applications. It includes encryption, identity and access management, and compliance AWS (Amazon Web Services) offers a wide range of cloud computing services that provide numerous benefits to individuals, businesses, and organizations. Here are some reasons why people use AWS services:

Scalability: AWS allows you to scale your computing resources up or down based on demand. This flexibility is particularly useful for businesses with varying workloads.

Cost Savings: AWS operates on a pay-as-you-go model, where you only pay for the resources, you use. This eliminates the need for large upfront investments in hardware and infrastructure.

Global Reach: AWS has data centers in multiple geographic regions, enabling you to serve customers and users from around the world with low latency.

Wide Variety of Services: AWS offers a vast array of services, including computing power, storage, databases, machine learning, analytics, content delivery, and more. This allows you to choose the services that best suit your specific needs.

Innovation and Speed: AWS provides tools and services that enable rapid development and deployment of applications, allowing you to innovate and bring products to market faster.

Managed Services: AWS manages many aspects of the underlying infrastructure, such as server maintenance, security patching, and updates.

2. LITERATURE SURVTVEY

A comparative study on distributed load balancing algorithm for cloud computing.

Lua et al. [1] proposed a load balancing scheme called as Join-Idle-Queue. This scheme is based on distributed load balancing which is done by distributed dispatcher. At the 1st step all distributed dispatchers make idle processors queue. Then join these idle queues to assign the jobs coming to the servers to reduce the load of other overloaded nodes. It also claims to reduce the response time.

Chen et al. [2] proposed (1995) a load balancing scheme called as User-Priority guided min-min scheduling algorithm for load balancing in cloud computing. First, they found the minimum execution time of all tasks then the maximum value was selected. This algorithm was proposed for static environment and simulated on cloudsim.

Liu et al [3] proposed (2006) a lock-free multiprocessing load balancing scheme. This scheme reduces the use of shared memory and improves the overall performance in multi-core environment.

Nitish C et al. [4] proposed load balancing scheme HEFT based workflow scheduling for cost Optimization with in Deadline in Hybrid clouds. They have simulated their work on workflows. They have taken deadline completion as main issue in load balancing for independent tasks and tried to reduce the overall cost. Randles et al. [5] done comparative study on distributed load balancing algorithm for cloud computing. They said that there are three methods for large scale load balancing in cloud systems- 1. Nature inspired 2. Random sampling of system domain, 3. Restructured system to optimize job assignments.

Chunling C. et al [6] proposed energy saving task scheduling strategy based on vacation queuing theory in cloud computing for dynamic environment. They have used vacation queuing model with exhaustive service to schedule the tasks. On the basis on busy period and busy time they have analyze the energy consumption of nodes. They have simulated algorithm using Matlab tool.

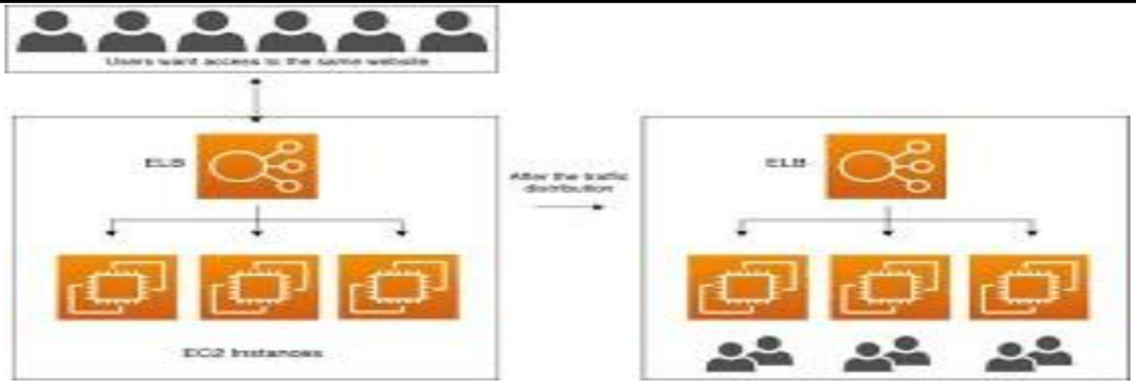
Users want access to the same website
ELB
After the traffic distribution
ELB
EC2 Instances

Fig 2.1 Working of ELB

In AWS, the Elastic Load Balancer (ELB) service, now known as the Application Load Balancer (ALB) and Network Load Balancer (NLB), we can use a flow-based, rather than a traditional round-robin algorithm. This flow-based approach ensures that incoming requests are directed to the appropriate target based on factors like the request's source IP, the destination IP, and the TCP/UDP port.
However, you can achieve a round-robin-like behavior in AWS ELB/ALB by configuring the load balancer's target group to use a "Least Outstanding Requests" or "Least Connections" routing algorithm. While these options don't provide a strict round-robin approach, they distribute traffic fairly evenly among healthy targets by selecting the target with the fewest outstanding requests or connections.
With the configuration of load balancer in Amazon Web Services Management Console, the ALB will distribute incoming requests to targets based on the algorithm you selected that is Round Robin which will result in a more evenly distributed load.

3. PROPOSED SYSTEM

In AWS, the Elastic Load Balancer (ELB) service, now known as the Application Load Balancer (ALB) and Network Load Balancer (NLB), we can use a flow-based, rather than a traditional round-robin algorithm. This flow-based approach ensures that incoming requests are directed to the appropriate target based on factors like the request's source IP, the destination IP, and the TCP/UDP port.
However, you can achieve a round-robin-like behavior in AWS ELB/ALB by configuring the load balancer's target group to use a "Least Outstanding Requests" or "Least Connections" routing algorithm. While these options don't provide a strict round-robin approach, they distribute traffic fairly evenly among healthy targets by selecting the target with the fewest outstanding requests or connections.
With the configuration of load balancer in Amazon Web Services Management Console, the ALB will distribute incoming requests to targets based on the algorithm you selected that is Round Robin which will result in a more evenly distributed load.

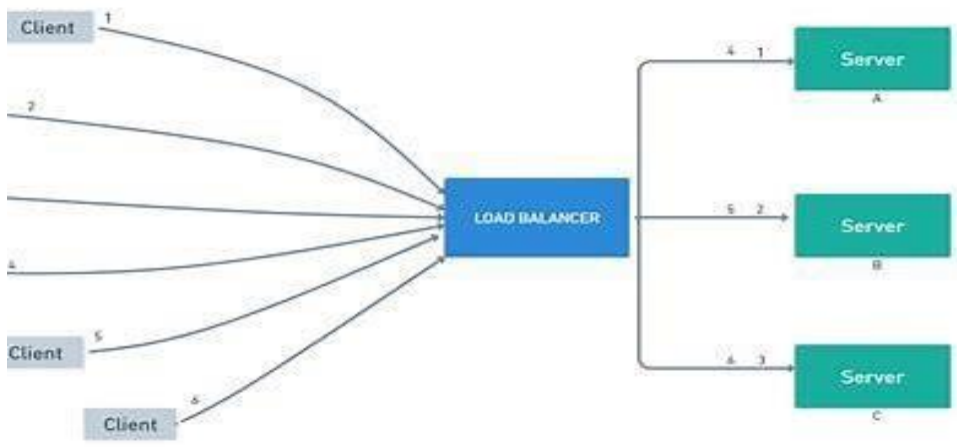

Figure 3.1 Nginx Load Balancer

Nginx:

Nginx (pronounced "engine-x") is a popular open-source web server, reverse proxy server, and load balancer. It's designed to efficiently handle HTTP, HTTPS, and other web protocols. Nginx is known for its high performance, scalability, and versatility, making it a widely used software in web hosting and server management. Nginx can act as a reverse proxy server, sitting in front of application servers (e.g., web applications, API servers) and forwarding client requests to the appropriate backend servers. This helps distribute traffic, improve security, and provide features like load balancing. It can distribute incoming traffic across multiple backend servers, ensuring that each server shares the load evenly. This is crucial for high availability and scalability in web applications.
It offers security features like access control, rate limiting, and the ability to filter and block malicious traffic. It can be used as part of a defense strategy against DDoS attacks and other security threats. Its lightweight and efficient nature make it a popular choice for serving web content and managing network traffic effectively.

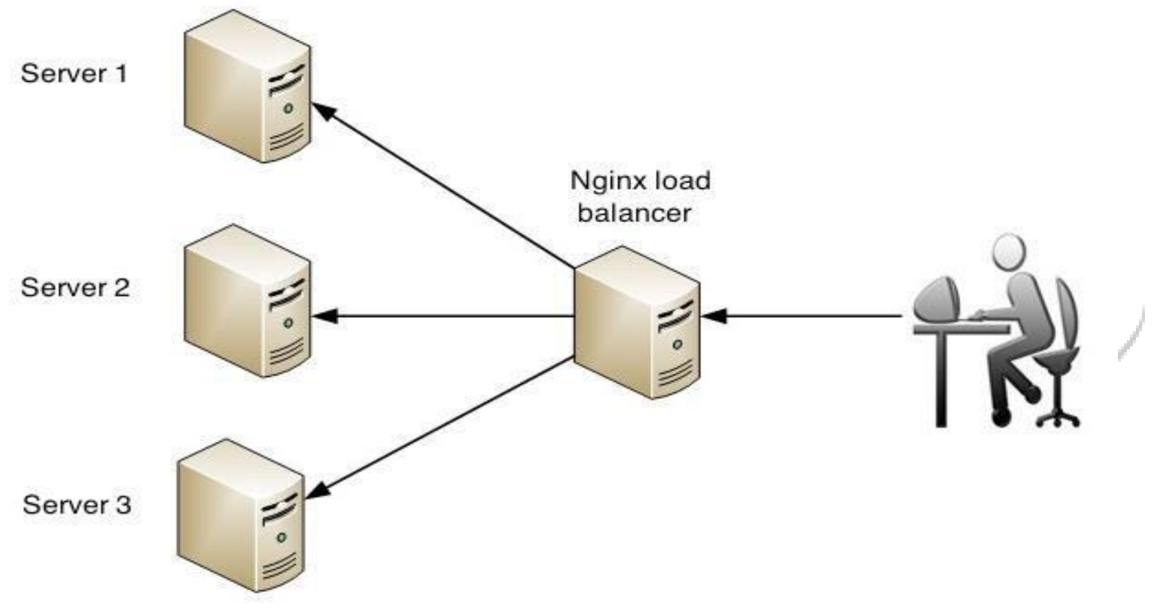

Figure 3.2 Round Robin

In documentation, a proposed approach typically refers to a recommended plan or strategy for achieving a certain goal or solving a problem. It outlines the steps, methods, and considerations that should be taken into account when implementing a particular solution. In the context of AWS documentation, a proposed approach could involve using specific AWS services, best practices, architectural considerations, and implementation guidelines to achieve a desired outcome or address a specific use case. It includes creating AWS designs and defining migration strategies and patterns thatsupport the move to the cloud and further optimization. This type of assessment istypicallytime consuming, and it's approached in a sequence ofsmaller chunks, aligned to migrated waves, throughout theproject lifecycle.

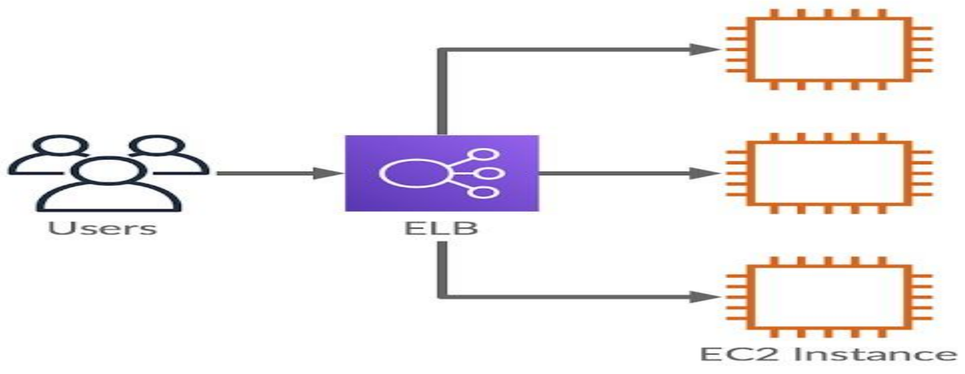

Figure 3.3 Traffic Distribution To Multiple Instance

The proposed system provides a user-friendly system through which farmers can generate detailed information. These profiles contain important information about their farm, their development and the products they offer. Farmers can list their crops, animals and other products, and provide descriptions and photos.

Existing Problem

Elastic Load Balancers (ELBs) are an integral part of distributing incoming network traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses. One exciting problem in this context could involve optimizing the ELB configuration to achieve the best performance, high availability, and cost In documentation, a proposed approach typically refers to a recommended plan or strategy for achieving a certain goal or solving a problem. It outlines the steps, methods, and considerations that should be taken into account when implementing a particular solution. In the context of AWS documentation, a proposed approach could involve using specific AWS services, best practices, architectural considerations, and implementation guidelines to achieve a desired outcome or address a specific use case. It includes creating AWS designs and defining migration strategies and patterns thatsupport the move to the cloud and further optimization. This type of assessment istypicallytime consuming, and it's approached in a sequence ofsmaller chunks, aligned to migrated waves, throughout theproject lifecycle.

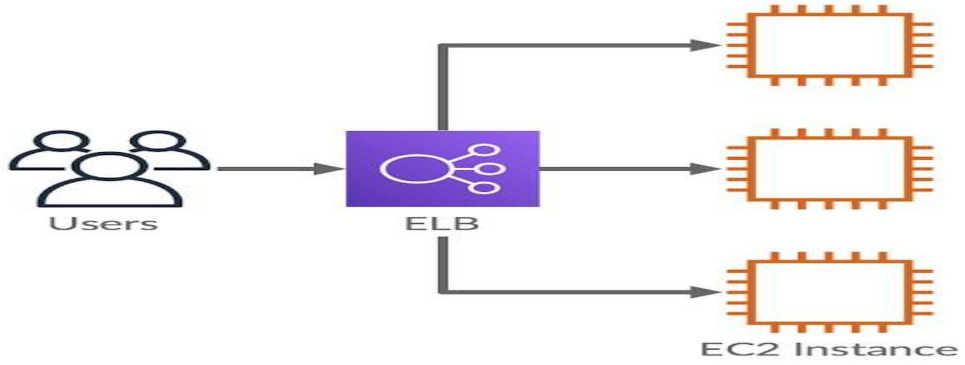

Figure 3.3 Traffic Distribution To Multiple Instance

The proposed system provides a user-friendly system through which farmers can generate detailed information. These profiles contain important information about their farm, their development and the products they offer. Farmers can list their crops, animals and other products, and provide descriptions and photos.

Existing Problem

Elastic Load Balancers (ELBs) are an integral part of distributing incoming network traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses. One exciting problem in this context could involve optimizing the ELB configuration to achieve the best performance, high availability, and cost effectiveness for your application. This might include:

Auto Scaling: Designing the ELB setup to dynamically adjust its capacity based on traffic spikes or drops, ensuring your application can handle varying loads efficiently.

effectiveness for your application. This might include:

Auto Scaling: Designing the ELB setup to dynamically adjust its capacity based on traffic spikes or drops, ensuring your application can handle varying loads efficiently.

Health Checks: Ensuring that the ELB accurately monitors the health of its registered instances and routes traffic only to healthy instances.

Cross-Zone Load Balancing: Configuring the ELB to distribute traffic evenly across multiple Availability Zones, providing better fault tolerance and reducing latency.

SSL/TLS Termination: Implementing SSL/TLS termination at the ELB to offload the decryption process from backend instances, improving performance and security.

Path-Based Routing: Utilizing path-based routing to direct different types of requests to specific backend instances or groups, enabling more efficient resource utilization.

Session Persistence: Handling session persistence effectively, either through sticky sessions or other mechanisms, to maintain user sessions across requests.

Monitoring and Logging: Setting up comprehensive monitoring and logging for the ELB to quickly identify and troubleshoot issues, helping maintain high availability.
Cost Optimization: Fine-tuning the ELB configuration to optimize costs while maintaining performance by using appropriate instance types, scaling policies, and traffic routing rules.
Geo-based Routing: Implementing geolocation-based routing to direct traffic to the nearest data center or region, improving latency for users around the world.
Customizing Load Balancing Algorithms: Exploring different load balancing algorithms offered by the ELB and selecting the one that best suits your application's requirements.

4. RESULTS

Output on Home Screen:

```
League 1 will NOT be changed
League 2 will NOT be changed
League 3 will NOT be changed
League 4 will NOT be changed
League 5 will NOT be changed
League 6 will NOT be changed
League 7 will NOT be changed
League 8 will NOT be changed
League 9 will NOT be changed
League 10 will NOT be changed
League 11 will NOT be changed
League 12 will NOT be changed
League 13 will be changed number of VMs:5
League 14 will NOT be changed
League 15 will NOT be changed
League 16 will NOT be changed
League 17 will NOT be changed
League 18 will NOT be changed
League 19 will NOT be changed
Mean fitness for this season:6032.0977
LCA Threshold for this season:2121.3716
Season 8
League 0 will be changed number of VMs:0
League 1 will be changed number of VMs:2
League 2 will be changed number of VMs:6
League 3 will be changed number of VMs:9
League 4 will be changed number of VMs:2
League 5 will be changed number of VMs:7
League 6 will be changed number of VMs:5
League 7 will be changed number of VMs:2
League 8 will be changed number of VMs:8
League 9 will be changed number of VMs:4
League 10 will be changed number of VMs:0
League 11 will be changed number of VMs:1
League 12 will be changed number of VMs:8
League 13 will be changed number of VMs:2
League 14 will be changed number of VMs:4
League 15 will be changed number of VMs:9
League 16 will be changed number of VMs:0
League 17 will be changed number of VMs:3
League 18 will be changed number of VMs:3
League 19 will be changed number of VMs:0
Mean fitness for this season:13653.094
LCA Threshold for this season:4801.528
Season 9
League 0 will NOT be changed
League 1 will be changed number of VMs:0
League 2 will be changed number of VMs:3
League 3 will NOT be changed
League 4 will NOT be changed
League 5 will be changed number of VMs:8
League 6 will be changed number of VMs:5
League 7 will be changed number of VMs:3
League 8 will NOT be changed
League 9 will be changed number of VMs:0
League 10 will be changed number of VMs:1
League 11 will be changed number of VMs:8
League 12 will be changed number of VMs:8
League 13 will NOT be changed
League 14 will be changed number of VMs:1
League 15 will NOT be changed
League 16 will NOT be changed
League 17 will be changed number of VMs:0
League 18 will be changed number of VMs:0
League 19 will be changed number of VMs:5
Mean fitness for this season:5998.4424
LCA Threshold for this season:2109.5356
Season 10
League 0 will be changed number of VMs:9
League 1 will be changed number of VMs:2
League 2 will be changed number of VMs:4
League 3 will be changed number of VMs:6
League 4 will be changed number of VMs:4
League 5 will be changed number of VMs:8
League 6 will be changed number of VMs:1
League 7 will be changed number of VMs:6
League 8 will be changed number of VMs:3
League 9 will be changed number of VMs:1
League 10 will be changed number of VMs:3
League 11 will be changed number of VMs:6
League 12 will be changed number of VMs:3
League 13 will be changed number of VMs:1
League 14 will be changed number of VMs:7
League 15 will be changed number of VMs:5
League 16 will be changed number of VMs:9
League 17 will be changed number of VMs:3
League 18 will be changed number of VMs:4
League 19 will be changed number of VMs:5
Mean fitness for this season:143650.95
LCA Threshold for this season:50519.25
Best solution found at index:9
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #5 in Datacenter_0
0.0: Broker: Trying to Create VM #4 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.0: Broker: Trying to Create VM #3 in Datacenter_0
Task 58, executing on Vm:2
Task 72, executing on Vm:0
Task 77, executing on Vm:1
Task 79, executing on Vm:3
Task 82, executing on Vm:2
Task 90, executing on Vm:2
Task 94, executing on Vm:2
Task 99, executing on Vm:3
Simulation completed.
All tasks completed
Season 1
League 0 will be changed number of VMs:4
League 1 will be changed number of VMs:5
```

CONCLUSION

In conclusion, our mini-project successfully demonstrated the implementation of an Elastic Load Balancer (ELB) in Amazon Web Services (AWS) using the round- robin algorithm, all achieved through the power of the C programming language. By leveraging AWS services and the round-robin algorithm, we've created an efficient and scalable load balancing system. Throughout this project, we've gained valuable insights into cloud computing, networking, and load balancing principles. We've learned how ELB can evenly distribute incoming requests to a group of backend servers, ensuring high availability and optimizing resource utilization. The round-robin algorithm, implemented in C, proved to be a simple yet effective means of achieving this load balancing. Implementing Elastic Load Balancing (ELB) in Amazon Web Services (AWS) with the round-robin algorithm offers a straightforward and effective approach to distributing incoming network traffic across multiple EC2 instances. This load balancing methodology ensures even traffic distribution, promoting high availability and scalability for applications. It is particularly suitable when the backend servers are homogeneous and capable of handling requests independently. By systematically routing requests to each instance in a circular fashion, round-robin load balancing optimizes resource utilization and enhances fault tolerance, contributing to a robust and efficient workload management solution within the AWS ecosystem. However, it's important to consider the specific requirements of your application and workload when choosing a load balancing algorithm to ensure it aligns with your performance and reliability objectives. While this miniproject provided a solid foundation, there is always room for further enhancements and exploration. Future iterations could involve integrating more advanced load balancing algorithms, adding monitoring and logging features, or extending the project to include other AWS services for a comprehensive cloud-based solution. In a rapidly evolving field like cloud computing, continuous learning and innovation are essential, and this miniproject serves as a stepping stone toward that ongoing journey.

REFERENCES

1. **[1]** Lua et al (2011) "Load Balancing Algorithms in Cloud Computing Environment: A Review", International Journal on Recent and Innovation Trends in Computing and Communication, Vol 2, Aug 2011. **[2]** Chen (2014)"An Overview of Cloud Computing Adoption Challenges in the Norwegian Context." Utility and Cloud Computing (UCC), 2014.
2. Liu (2014) "A comparative study into distributed load balancing algorithms for cloud computing." Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on. IEEE, 2014.
3. Nitish (2015) "Comparative study on load balancing techniques in cloud computing." Open Journal of Mobile Computing and Cloud Computing 1.1 (2014).
4. M. Randle's (2010) "An in-depth analysis and study of Load balancing techniques in the cloud computing environment." Procedia Computer Science 50 (2015).
5. Chunling (2018)" Load balancing and resource monitoring in cloud", Proceedings of the CUBE International Information Technology Conference. ACM, 2018.
6. Brian Chang (2023) "Leading Load Balancing" ICDCN '23: Proceedings of the 24th International Conference on Distributed Computing and Networking January 2023.
7. Steven Hofmeyr (2010) "Load Balancing on Speed" PPoPP '10: Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming January 2010.
8. Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari, "What is an object?",
9. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010.

10. Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. "Measuring the objectness of image windows", IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11), 2189-2202, 2012.
11. Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala, "Material recognition in the wild with the materials in context database." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,3479-3487, 2015".
12. Design of a Convolutional Neural Network Based Smart Waste Disposal System.
13. Thung, Gary and M. Yang. "Classification of Trash for Recyclability Status." (2016).
14. Gary Thung and Mindy Yang, "Classification of Trash for Recyclability Status," CS 229, Stanford University, 2016.Available:
15. C. Liu, L. Sharan, E. H. Adelson, and R. Rosenholtz, "Exploring features in a bayesian framework for material recognition," in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010, pp. 239–246.
16. George E Sakr, Maria Mokbel, Ahmad Darwich, Mia Nasr Khneisser and Ali Hadi, "Comparing Deep Learning And Support Vector Machines for Autonomous Multidisciplinary Conference.
17. S. Kaza, L. Yao, P. Bhada-Tata, and F. Van Woerden, What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050. The World Bank, 2018.

18. "Convolutional networks and applications in vision," in Proceedings of 2010 IEEE International Symposium on Circuits and Systems, Paris, France, pp. 253–256, May 2010. doi: 10.1109/ISCAS.2010.5537907.

19. N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, vol. 1, pp. 886–893, 2005. doi: 10.1109/ CVPR.2005. 177.

20. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image.
21. Recognition," in arXiv:1409.1556 [cs], San Diego, CA, USA, pp. 1–14, May 2015, [Online]. Available: http://arxiv.org/abs/1409.1556 [Accessed: Sep. 27, 2019].
22. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in 2016 IEEE
23. Conference on Computer Vision and Pattern
24. Recognition (CVPR), Las Vegas, NV, USA, pp. 770–778, Jun. 2016. doi: 10.1109/CVPR.2016.90.
25. C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," ACM Trans. Intell. Syst. Technol., vol. 2, no. 3, pp. 1– 27, Apr. 2011. doi: 10.1145/1961189.1961199.

26. M. Yang and G. Thung, “Classification of Trash for Recyclability Status,” Stanford University, CS229, 2016.

27. C. Bircanoglu, M. Atay, F. Beser, O. Genc, and M. A. Kizrak, “RecycleNet: Intelligent Waste Sorting Using Deep Neural Networks,” in 2018 Innovations in Intelligent Systems and Applications (INISTA), Thessaloniki, pp.1–7, Jul. 2018. doi: 10.1109/INISTA.2018.8466276.

28. H. Khan, “Transfer learning using mobilenet,” 2019. [Accessed Sep. 23, 2019].