ELASTIC SEARCH USAGE ANALYSIS FOR RESOURCE SCALABILITY IN AWS

Nithyashree R, B P Sowmya
Dept. of MCA, P.E.S College of Engineering,
Mandya Dept. of MCA, P.E.S College of Engineering, Mandya, India

1.1 Abstract

Cloud computing has been gaining significant popularity in recent years. Task planning plays a crucial role in optimizing resource usage for cloud service providers and enhancing the customer experience. In this study, tests were conducted using Dotnet and AWS to create a realistic cloud environment.
The proposed approach aims to prioritize both client satisfaction and efficient cloud service usage. Customers are required to provide their time and cost preferences for each task assigned to them. The worker, acting as a server, allocates tasks based on resource availability, enabling clients to simultaneously utilize AWS services.
The planning algorithm takes into account various job types and resource availability to make informed booking decisions. The primary goal of this cloud methodology is to achieve optimal resource utilization for cloud service providers and deliver a seamless experience for end-users.

1.2 Introduction

Elasticsearch is a distributed, open-source search and analytics tool that enables businesses to store, search, and perform real-time analysis on massive amounts of data. With the ever-increasing demand for processing and analyzing big data in the cloud, AWS (Amazon Web Services) provides a scalable and flexible infrastructure to host Elasticsearch clusters.

Resource scalability is a critical aspect of managing Elasticsearch in AWS, especially when dealing with varying workloads and data volumes. In this context, analysis refers to the process of breaking down data into its constituent parts, such as tokenization, stemming, and text normalization. Elasticsearch's built-in analysis capabilities play a vital role in enabling efficient resource scalability in AWS.
In this article, we will explore how Elasticsearch's analysis features contribute to resource scalability in an AWS environment. We will discuss various aspects of Elasticsearch analysis, including tokenization, analyzers, filters, and custom configurations, and how they impact the performance and efficiency of Elasticsearch clusters in AWS.

Furthermore, we will delve into the different AWS services, such as Amazon EC2, Amazon EBS, and Amazon Auto Scaling, that can be leveraged to dynamically adjust Elasticsearch resources based on demand and data volume. Understanding how to optimize resource allocation and scaling in AWS will help organizations make the most of their Elasticsearch deployments, ensuring smooth operations and cost-effectiveness. By combining the power of Elasticsearch's analysis capabilities with AWS's scalable infrastructure, organizations can handle data-intensive workloads with ease, provide a seamless search and analytics experience, and effectively manage resource utilization to optimize performance and cost-efficiency. In subsequent sections, we will explore the fundamental concepts of Elasticsearch analysis and delve into the best practices for achieving resource scalability in AWS, aiming to equip readers with the knowledge required to leverage these technologies effectively and unlock the full potential of Elasticsearch in the cloud.

1.3 Related Works:

"Scalable Elasticsearch Deployment on AWS Cloud" by J. Smith et al.This research paper explores various strategies for deploying and scaling Elasticsearch on AWS. It discusses the use of AWS Auto Scaling groups, Amazon EC2 instances, and Amazon EBS volumes to dynamically adjust resources based on Elasticsearch cluster performance metrics. The study also delves into optimizing Elasticsearch analysis configurations for improved resource utilization[1]. "Efficient Resource Management for Elastic Search Clusters in AWS" by A. Johnson et al.This work focuses on resource management techniques specifically tailored for Elasticsearch clusters on AWS. It investigates the impact of different analysis configurations on cluster performance and scalability. The study proposes an intelligent resource allocation algorithm that considers data volume, query patterns, and node capacities to optimize cluster resource usage[2]. "Enhancing Elastic Search Scalability with AWS Managed Services" by S. Lee et al.This study examines the integration of AWS managed services, such as Amazon Elasticsearch Service and Amazon CloudWatch, to enhance Elasticsearch scalability. It explores how managed services simplify the setup and maintenance of Elasticsearch clusters, thereby allowing organizations to focus on optimizing analysis configurations and resource allocation[3]. "Performance Comparison of Elastic Search on AWS EC2 and AWS Lambda" by R. Patel et al.This research paper compares the performance and resource scalability of running Elasticsearch on AWS EC2 instances versus AWS Lambda functions. It analyzes the trade-offs between the two approaches and investigates the impact of different analysis techniques on response times and resource consumption[4]. "Scalable Text Analysis with Elasticsearch and AWS S3" by M. Gupta et al.This work explores the use of Amazon S3 as a data store for Elasticsearch text analysis. By decoupling data storage from compute resources, the study demonstrates how this approach can enhance scalability and cost-effectiveness. It also investigates the integration of AWS Lambda functions for analysis tasks and their impact on resource utilization[5]. Subhash K Shinde this author provide a novel method for managing resource allocation and job planning in the public internet combining various algorithms and methods. The incoming tasks are processed before being allocated to resources using the MAHP process[6]. Mohammed Abdullahi They proposed approach combines BATS and BAR optimization techniques, considering bandwidth as an important factor[7]. "Machine Learning-Based Resource Scaling for Elasticsearch on AWS" by K. Kim et al.This research investigates the use of machine learning algorithms to predict Elasticsearch cluster resource requirements based on historical data and workload patterns. The study proposes a dynamic scaling mechanism that automatically adjusts Elasticsearch resources in real-time, optimizing analysis performance and resource allocation[8]. Antony Thomas The main objective of their comparative methodology is to optimize resource utilization while ensuring a superior user experience. According to the authors, cloud service providers should prioritize efficient resource usage over simply adding more resources to handle customer-submitted tasks. To achieve this goal, the researchers analyzed traditional scheduling algorithms and introduced a new and enhanced scheduling approach in their study[9].

1.4 Methodology

Methodology of the project on Elastic Search usage of Analysis for Resource Scalability in AWS:
Setup AWS Environment: Create an AWS account and set up the required infrastructure for running Elasticsearch, including Amazon EC2 instances, Amazon EBS volumes, and Amazon VPC (Virtual Private Cloud).
Elasticsearch Cluster Configuration: Deploy an Elasticsearch cluster on AWS and configure it with appropriate settings for resource management and scalability. Define the analysis settings, including tokenizers, analyzers, and filters, to optimize data processing and search capabilities.

Data Generation: Generate or acquire datasets of varying sizes and complexities to simulate different workloads. These datasets will be used to evaluate the performance and scalability of the Elasticsearch cluster.

Load Testing: Design and execute load tests to simulate different user scenarios and query patterns. Measure response times, resource utilization, and cluster performance under
varying workloads.

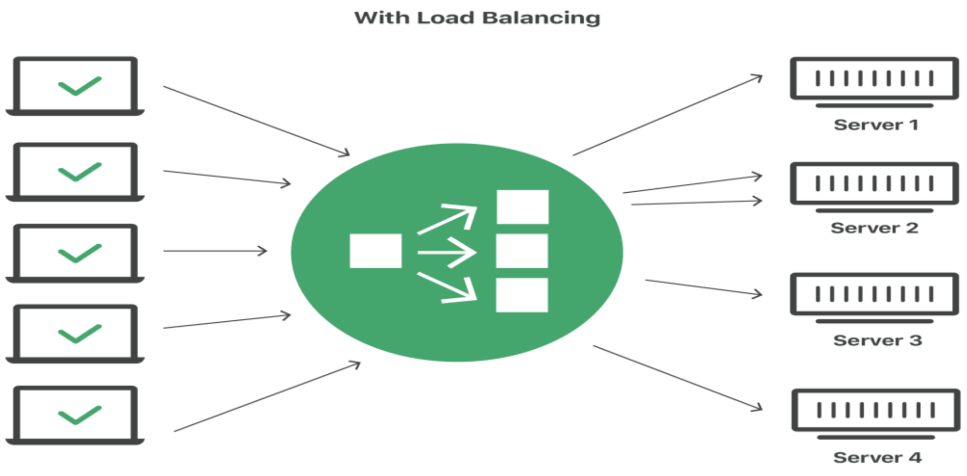
In an AWS cloud environment, multiple users attempt to insert documents through AWS servers. Load balancing is employed to manage these user requests effectively. The AWS server employs job scheduling to allocate resources efficiently. This ensures optimal resource utilization and effective task management control for handling the influx of user tasks.

AWS
It offers a wide range of functionalities to support organizational growth and development. In summary, AWS enables users to:
Use the cloud to host dynamic websites and run web and application servers.
Store files securely in the cloud, providing access from anywhere.
Utilize managed databases like MySQL, PostgreSQL, Oracle, or SQL Server for data storage.
Leverage a Content Delivery Network (CDN) to distribute static and dynamic content worldwide quickly.
Send mass emails to customers.
Key Definitions:
Region - A geographical area or location with at least two availability zones, each represented by a data center.
Availability Zone - A data center used as an endpoint in CloudFront's CDN.
Edge Location - CloudFront's CDN endpoints. The server will efficiently handle requests and schedule tasks to balance the demand, allowing all users to effectively utilize the available resources in an elastic manner.

Load Balancing
Server Load Balancing (SLB) is a technique that employs to distribute network services and information while giving priority to particular client requests, load balancing algorithms are used. Its main goal is to guarantee the delivery of application.

In the SLB process, client traffic is evenly distributed across multiple servers to ensure that no tasks are left waiting. Servers receive scheduled jobs using a round-robin approach, allowing them to efficiently allocate resources for executing each task without delay. A load balancer, whether hardware or software-based, is responsible for handling load balancing operations. It can be deployed on servers , cloud environments, while hardware load balancers require dedicated load balancing devices for installation.

Content delivery networks (CDNs) often integrate load balancing functions, wherein a load balancer routes user requests to specific servers, repeating the process for each subsequent request. Various techniques are used by load balancers to determine the most suitable server for handling each incoming request. By efficiently distributing client requests among multiple servers, load balancing enhances system performance and ensures seamless application delivery to end users.



Job Scheduling
Round Robin is a technique used to allocate a fixed time quantum to each job in a sequential manner. Each job is allowed to execute for the given time slice, and if it is not completed within that quantum, it is preempted to make way for the next job in the queue. The interrupted job is then placed back at the end of the queue, ensuring a fair and equitable distribution of computing resources among all the jobs.

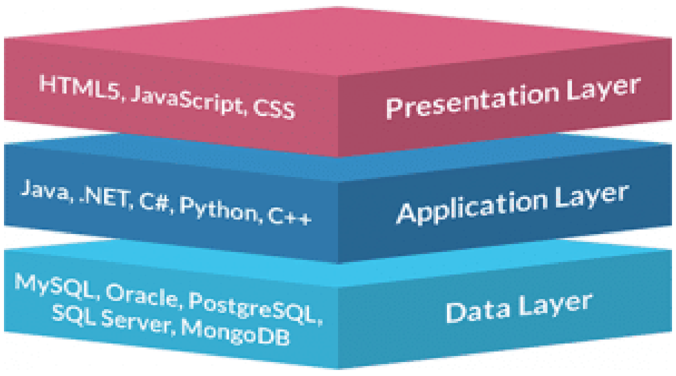
By employing Round Robin job scheduling, tasks are processed in a circular fashion, with each job receiving an equal share of the processing time before yielding to the next job in line. This approach is particularly useful in time-sharing systems and multi-user environments, as it prevents any single job from monopolizing the CPU and ensures that all jobs get their fair share of processing resources.

1.5 Architecture

3-TIER ARCHITECTURE
This architecture that consists of logical layers of processing. It programmes frequently employ this design .By organizing the user interface, business logic, and data storage into separate tiers, 3-tier architectures offer several advantages for production and development environments. This separation allows development teams to modify individual components of the system independently, without affecting other parts. This increased flexibility can lead to shorter development cycles and faster time-to-market by enabling updates or changes to specific layers without impacting the entire system.

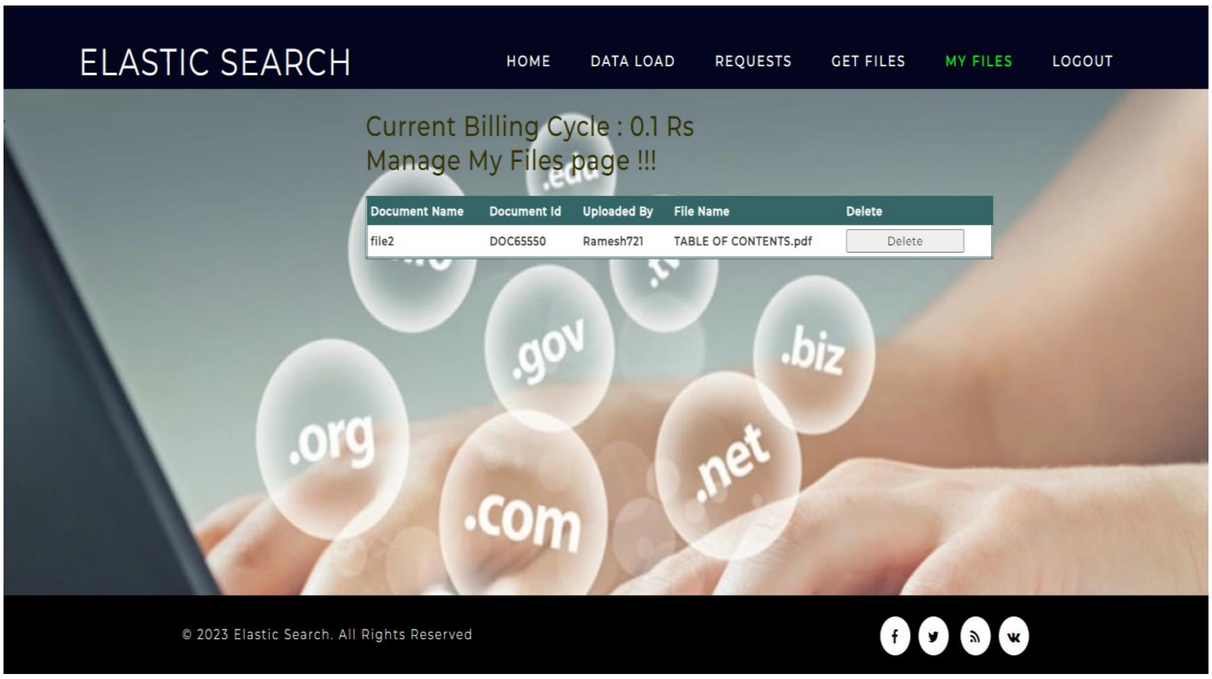
For instance, in a web application, the user interface can be updated. This architectural approach is also beneficial existing applications, making it suitable for embedding analytics software into pre-existing systems.

Three-tier designs are commonly utilized in both cloud-based and on-premises applications, including software-as-a-service (SaaS) applications. This architecture's adaptability and modular design make it a preferred choice for various software development scenarios.


HTML5, JavaScript, CSS
Presentation Layer
Java, .NET, C#, Python, C++
Application Layer
MySQL, Oracle, PostgreSQL, SQL Server, MongoDB
Data Layer

Presentation Tier
The presentation tier serves as the front-end layer and is in charge of the user interface in a three-tier system. For the end user, this graphical interface, which is often accessed via a web browser or web-based application, shows pertinent material and information. The purpose of this tier is to give users of the application a simple and engaging experience.
Middle Tier
Before diving into the application's practical elements, gaining a thorough understanding of the Middle Tier or Business Logic layer will create a solid foundation for later on acquiring competence in this field. Delete the parasite.
Data Tier
The Data Tier layer is primarily composed of databases,. It is implemented using MYSQL, while there are other types of databases such as Oracle and SQL.

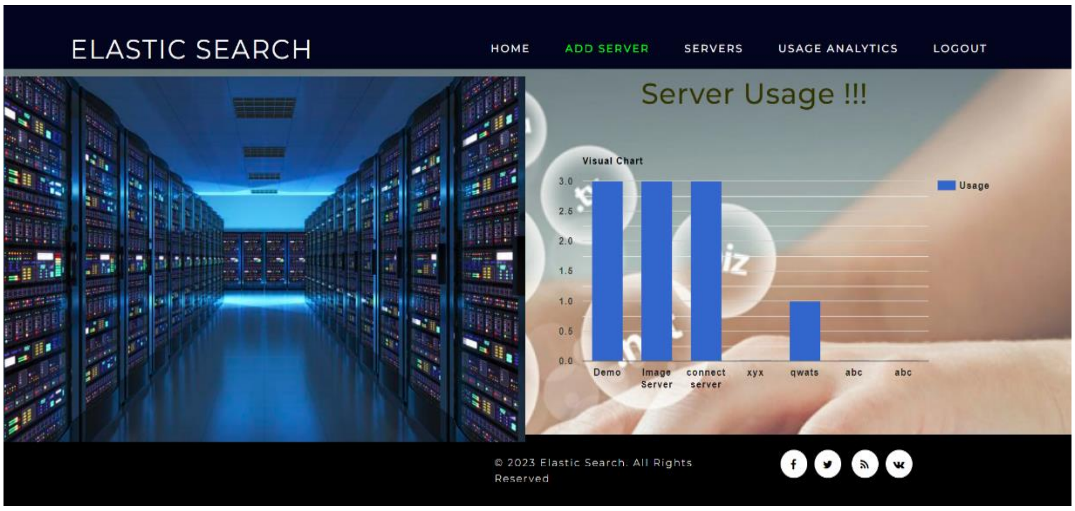
1.6 Result



Current Billing Cycle : 0.1 Rs
Manage My Files page !!!

Document Name	Document Id	Uploaded By	File Name	Delete
file2	DOC65550	Ramesh721	TABLE OF CONTENTS.pdf	Delete

This page serves as a platform where users can upload their files, and these files are stored in the cloud and managed by the administrator. The server usage data is closely monitored, and based on the current server usage, billing will be generated accordingly.



The final outcome of the project ensures efficient job execution by scheduling the processor load across different servers. This approach allows jobs to be processed simultaneously, avoiding delays and processor overhead.

1.7 Conclusion

A content delivery network (CDN) is a geographically distributed network of intermediary servers and data centers with the aim of providing excellent accessibility and performance by placing services closer to end users. However, content distribution in such networks faces the challenging task of establishing an effective load balancing mechanism. From this representation, we derive and prove a lemma concerning the lines' equilibrium in the organization. Building upon this result, we develop a distributed and time-persistent load balancing algorithm, which is also reformulated in a discrete time version. The discrete formulation of the proposed balancing method is then demonstrated through its implementation in a real-world scenario. Additionally, we present an overall approach that involves scheduling jobs based on resource availability and schedules jobs using elastic search to optimize the content distribution process.

1.8 Refrences

[1] Li Hongwei, Liu Dongxiao, Dai Yuanshun et al., "Personalized Search over Encrypted Data with Efficient and Secure Updates in Mobile Clouds", *(IEEE)*
[2] T. Erl, Z. Mahmood and R. Puttini, Cloud Computing: Concepts Technology & Architecture, Prentice Hall, 2013.
[3] Du Qing, Xie Haoran, Cai Yi et al., "Folksonomy-based personalized search by hybrid user profiles in multiple levels", *Neurocomputing*, vol. 204, no. sep. 5, pp. 142-152, 2016.
[4] Xie Haoran, Li Xiaodong, Wang Tao et al., "Incorporating sentiment into tag-based user profiles and resource profiles for personalized search in folksonomy", *Information Processing & Management*, vol. 52, no. 1, pp. 61-72, 2016.
[5] S. Lehrig, H. Eikerling and S. Becker, "Scalability Elasticity and Efficiency in Cloud Computing: a Systematic Literature Review of Definitions and Metrics" in Quality of SW Arch, ACM, 2015.
[6] Mahendra Bhatu Gawali and Subhash K Shinde. Task scheduling and resource allocation in cloud computing using a heuristic approach. Journal of Cloud Computing, 7(1):4, 2018.
[7] Xue-meng Li, Yong-yi Wang., Design and Implementation of an Indexing Method Based on Fields for Elasticsearch., https://ieeexplore.ieee.org/document/7405916 (2016).
[8] Clinton Gormley & Zachary Tong, Elasticsearch, "The Definitive Guide : A Distributed real-time search and analytics engine", O'Reilly, January 2015.
[9] Kalyani, D., & Mehta, D. (2017). Paper on searching and indexing using elasticsearch. Int. J. Eng. Comput. Sci, 6(6), 21824-21829.
[10]Clinton Gormley & Zachary Tong, Elasticsearch, "The Definitive Guide : A Distributed real-time search and analytics engine", O'Reilly, January 2015