Fortifying the Cloud with Big Data Computing

A Technical Report submitted to the Department of Computer Scienc

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Dagim Tekle
Spring 2024

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Advisor
Briana Morrison, Department of Computer Science

Fortifying the Cloud using Big Data Computing

CS4991 Capstone Report, 2024

Dagim Tekle
Computer Science
The University of Virginia
School of the Engineering and Applied Science
Charlottesville, Virginia USA
ddt8ee@virginia.edu

ABSTRACT
As companies, services, and governments migrate computing resources to the cloud, cloud providers must ensure the security of the zettabytes of data entrusted to them. One of the basic security measures major cloud providers implement is logging user and process activities, enabling security software and teams to swiftly identify malicious activity and mitigate the impacts. In my internship, I assisted with investigating a scalable indexing and querying strategy for security logs. By utilizing distributed computing on the cloud and designing indexing and querying workflows with minimal bottlenecks, the Proof of Concept (POC) was able to index logs at pre-production scale and return a query result promptly. This performance, though significantly impressive, was not up to par with the production scale security logs to be indexed, and it came with the cost of storing ever-growing indexes for the already ever-growing security logs. There is still work to be done on two fronts: 1) improving indexing performance to meet and exceed the production scale and 2) reducing the size of security logs and indexes that persist forever.

1. INTRODUCTION

Cloud computing was first conceived in 1961 when MIT Professor, computer and cognitive scientist John McCarthy imagined a future where computing resources (e.g., networks, servers, storage, applications, and services) would be operated as public utilities (Surbiryala & Rong, 2019). Starting with Salesforce in 1999, Amazon, Microsoft, Google, IBM, Adobe, SAP, Oracle, CISCO, and Alibaba have ventured into providing cloud, a rapidly growing market (Surbiryala & Rong, 2019; Jones, 2024; DeLisi & Howley, 2023).Though less than a century old, cloud computing has become very prevalent. In a study done by Wakefield Research (2012), 95% of Americans reported using social media (e.g., Facebook, Instagram, Tiktok), online banking (e.g., Capital One, Chase, Bank of America), online shopping (e.g., Amazon Prime, eBay, Etsy), online gaming (e.g., Poker, Chess, Fortnite), online photo storage (e.g., iCloud photos, Google photos, Snapchat), online music streaming (e.g., Spotify, Apple Music, YouTube Music), online video streaming (e.g., Netflix, Hulu, Max), or online file sharing (e.g., Google Drive, Dropbox, OneDrive). All rely on some form of cloud computing. Despite the widespread use of cloud computing services, 63% of surveyed participants voiced their concerns regarding privacy and security (Wakefield Research, 2012). With a statistical majority of the respondents holding privacy and security paramount, cloud providers—in order to earn and retain customers and their trust—must commit themselves to satisfy and surpass the security and privacy needs.

2. RELATED WORKS

One of the pillars of cloud security is robust access monitoring of information like who accessed what resources and when (Smith, 2022). Providers like Amazon Web Services (AWS) utilize sophisticated logging mechanisms, such as AWS CloudTrail, which constantly monitors and captures API calls and related events (*AWS CloudTrail*, n.d.). Similarly, Microsoft Azure employs Azure Monitor to track access and activity across every layer and components of its platform (*Azure Monitor*, n.d.). These logging systems not only assist in troubleshooting and auditing but also enhance security by enabling the analysis and detection of suspicious activities or unauthorized access attempts (*AWS CloudTrail*, n.d.; *Azure Monitor Overview*, n.d.). By implementing comprehensive logging solutions, cloud providers offer their customers vigilant and responsive security protocols, augmenting the overall security posture of their platforms.

With the vast amount of data generated by millions of users in modern computing environments, navigating through logs can become quite cumbersome, time-consuming, and expensive, particularly when dealing with time-sensitive queries. Inverted indexing emerges as a vital solution to this challenge. Similar to indexes at the end of books, inverted indexing is a data structure that maps terms to documents (GeeksForGeeks, n.d.). Instead of scanning every page of a book to query a term, indexing allows querying through a smaller set of pages with pre-mapped results, facilitating expedited querying though large datasets. This technique significantly enhances the efficiency of querying security logs, allowing for rapid retrieval of pertinent information without the need for exhaustive scans (Yang & Duo, 2013). By leveraging inverted indexing, security teams can swiftly identify relevant log entries based on attributes such as user IDs, IP addresses, timestamps, or event types, enabling optimized resource utilization for querying.

3. PROPOSAL DESIGN

During my internship, I was tasked with a comprehensive exploration aimed at developing a scalable and cost-effective indexing and querying strategy utilizing Lucene in conjunction with the robust tools available on Amazon Web Services (AWS). Apache Lucene stands out as an open-source Java library renowned for its promise of scalable, high-performance indexing capabilities and its deployment of accurate, efficient search algorithms, with noteworthy Lucene-based enterprise search servers include Apache Solr, Elasticsearch, MongoDB, Atlas Search, OpenSearch, and Swiftype. Lucene boasts impressive indexing performance, reportedly exceeding 800GB/hour on modern hardware, with index sizes typically ranging from 20-30% of the text indexed. This suggests the potential for approximately 1000 modern hardware setups to effectively index the projected petabytes per hour within my project's scope.

3.1 Computing Environments

In order to leverage Lucene's library effectively, it is essential to utilize a Java-based computing environment. AWS offers Java-based computing environments like Amazon Elastic Compute Cloud (Amazon EC2), AWS Lambda, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic MapReduce (EMR)
AWS Elastic Beanstalk, and AWS Batch. AWS Lambda present themselves as particularly compelling choices for implementing Lucene due to their simplicity and ease of use. By abstracting away infrastructure management complexities, Lambda offers a streamlined, event-driven approach to deploying a Lucene-based

solution, enabling more focus on application development and less on infrastructure configuration.

Additionally, AWS offers several storage options for storing and quickly retrieving Lucene indexes efficiently. These include Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS), and Amazon Elastic File System (Amazon EFS). Among these, Amazon S3 stands out as a highly scalable and durable object storage service, capable of storing massive amounts of data reliably at low costs.S3 provides high availability and durability, making it an excellent choice for long-term storage of Lucene indexes. Additionally, its pay-as-you-go pricing model ensures cost-effectiveness, allowing organizations to scale their storage infrastructure according to their needs without incurring unnecessary expenses.

3.2 Integrated Workflow

With these design decisions, we are left with creating the indexing and querying workflows, two almost independent processes with the exception of the way the indexes are stored and accessed. Therefore, the nature of the querying use cases will establish the data organization required in the indexing strategy. One pertinent use case is filtering queries with a time range, where efficient storage and retrieval of time-series data become paramount, such as partitioning by time intervals. This approach can significantly enhance query performance and reduce latency by narrowing down the relevant indexes to search. Now, we can proceed on designing each workflow.

3.3 Indexing Workflow

The indexing workflow begins with an event with details of access logs to index. Using Lambda's event-trigger, these events will start Lucene indexing jobs. Once done with creating indexes, the lambda will upload them into time partitioned S3 buckets.

3.4 Querying Workflow

As for the querying workflow, it begins by users submitting a job. This creates a unique query ID, prompting the spawning of Lambdas to conduct searches across specific time ranges within the indexed files. These Lambdas diligently append results into an Amazon DynamoDB (DDB) table asynchronously, with the query ID serving as the primary key. This setup allows users to access results by querying the DDB with their unique query ID, offering a smooth and responsive experience.

4. RESULTS

One of the first complications discovered with the proposed design is the timeout and memory limits of the Lambda. Lambda is designed to be nimble, and thus intended to work a maximum of 15 minutes and some memory limit per invocation. This was probably caused by a very high variability of data size in the documents being indexed. To deal with the time and memory limits, I created a hard limit on the number of documents indexed by a Lambda invocation based on the statistical average size of the documents indexed.

Another challenge came with the indexing workflow. Since each Lambda invocation would create one directory in the time partition directory, the high pre-production scale often led to thousands of directories per hour partitioning. As a result, for each hour, there needed to be thousands of lambda invocations, adding overhead costs. We found that searching multiple indexes per one Lambda query invocation reduced overall querying time. Nonetheless, because of the Lambda concurrency limits, the subsequent queries would get backlogged often.

5. CONCLUSION

The continued success of the cloud depends on the level of security provided by cloud providers. The first step towards this goal is creating robust monitoring systems that are stored as logs. With large-scale ever-growing internet-scale security logs, however, the naive approach of searching through all logs for a query is ineffective. Querying through Inverted indexing, bridges the gap between storing large data sets while minimizing querying overhead.

This paper walked through one approach: using Apache Lucene on AWS, specifically AWS Lambda. The simplicity of Lambda offered quicker onboarding of this project. This choice, however, added limitations on running time and memory space, resulting in more index directories being generated. As a result, the querying workflow would often backlog after running queries in a short period of time.

6. FUTURE WORK

The very next step on investigating a scalable indexing and querying strategy with Lucene on AWS is implementing other AWS running environments. By using a service like AWS EMR Serverless, the indexing workflow can be augmented into having larger ephemeral memory and almost unlimited time limit. As a result, this new approach could bolster the querying workflow, reducing the number of directories per time portion hour and querying bottlenecks.

REFERENCES

Amazon. (2021, March 17). The deceptively simple origins of AWS. About Amazon.https://www.aboutamazon.com/news/aws/the-deceptively-simple-origins-of-aws

Amazon OpenSearch Service . AWS. (n.d.-a). https://aws.amazon.com/opensearch-service/

Amazon S3. AWS. (n.d.-b). https://aws.amazon.com/s3/ AWS

CloudTrail. AWS. (n.d.-c). https://aws.amazon.com/cloudtrail/

Azure Blob. Microsoft Learn. (n.d.-a). https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction

Azure Monitor Overview. Microsoft Learn. (n.d.-b). https://learn.microsoft.com/en-us/azure/azure-monitor/overview

GeeksforGeeks. (n.d.). Inverted index. GeeksforGeeks. https://www.geeksforgeeks.org/inverted-index/

J. Surbiryala and C. Rong, "Cloud Computing: History and Overview," 2019 IEEE Cloud Summit, Washington, DC, USA, 2019, pp. 1-7, doi: 10.1109/CloudSummit47114.2019.00007.

John McCarthy. Encyclopædia Britannica. (n.d.). https://www.britannica.com/biography/John-McCarthy

Jones, E. (2024, February 13). Cloud market share: A look at the Cloud Ecosystem. Kinsta. https://kinsta.com/blog/cloud-market-share/

Smith, A. (2022, June 21). Best practices for effective cloud data security. CSA. https://cloudsecurityalliance.org/blog/2022/06/21/best-practices-for-effective-cloud-data-security

W. Yang & Y. Dou, "High-Performance Distributed Indexing and Retrieval for Large Volume Traffic Log Datasets on the Cloud," 2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, China, 2013, pp. 185-189, doi: 10.1109/IHMSC.2013.51