Distributed Database Management System Final Project Report

Department of Computer Science and Technology
Tsinghua University

Group Members

Namie Turi Abu - 2023280116
Lim Ze Dong - 2023280123

January 2024

Contents

1 Abstract

In this project, our primary objective was to implement a distributed database management system capable of handling both structured and unstructured data. The motivation behind choosing a distributed approach stemmed from the need for improved scalability and performance, addressing limitations observed in centralized systems. The technological stack involved the utilization of MySQL Workbench, Flask Python framework, HTML, CSS, and MinIO object storage. Integration with other systems was a key consideration, and we detailed the seamless integration process through APIs or middle ware. Data sources varied, incorporating internal data generation, external API data collection, and other acquisition methods. The collaborative nature of the project was highlighted, showcasing team roles responsible for database design, programming, testing, and other aspects. The end-to-end data processing workflow, from ingestion to presentation, was elucidated, emphasizing the steps involved in managing structured and unstructured data. Overall, the project aimed to make a significant impact by contributing to enhanced efficiency and effectiveness in data management processes.

2 Problem Background & Motivation

Traditional database systems, while widely used and effective in many scenarios, do have some drawbacks. Some common drawbacks associated with traditional database systems are scalability, complexity, limited support for unstructured data, not ideal for big data, performance bottleneck and etc.

It's worth noting that advancements in database technology, including the rise of NoSQL databases, distributed databases, and cloud-based solutions, have addressed some of these drawbacks for specific use cases. Choosing the right type of database depends on the specific requirements and characteristics of the application or system.

The purpose of this project is to help us gain deep and insight understanding on the knowledge of the advanced big data management techniques in a distributed environment through a hands-on software design and implementation experiment and also to grasp the very latest big data management technologies and apply them to solve real-world problems.

The requirements that the system aims to fulfill are outlined below. At the very least, the system should be capable of:

1. Loading bulk data into table
2. Execute insert, update or delete queries
3. Evaluate the running status of servers

The paper is structured in the following manner. An analysis of the existing solutions concerning big data management systems is provided in **section 3**. It is followed by an explanation of the architecture under which the platform is supported, as well as a definition of its primary characteristics/features in **section 5**. **Section 6** explains the monitoring tools employed to evaluate the status of the system. Lastly, concluding remarks are highlighted in **section 10**.

3 Existing Solutions

Historically, developers have typically had two choices when it comes to storing data: relational databases and non-relational databases. Relational databases utilize a powerful query language like SQL and secondary indexes to establish relationships within structured data models or schemas. On the other hand, non-relational databases offer a more flexible and convenient approach to storing data, prioritizing scalability and performance[1].

Corporate entities have played a significant role in integrating features from both relational and non-relational models. A notable example is Google's Cloud Spanner, which seamlessly incorporates both relational and non-relational characteristics in its architecture. This allows for highly scalable data storage while ensuring robust performance and consistency driven by MySQL. However, the associated setup and maintenance costs are prohibitive for our budget. Another option considered is Amazon Neptune, which offers convenient features to expedite development but comes with similarly high associated costs. Utilizing private database management systems is thus not a viable option due to cost considerations.

Achieving scalability and distributed properties is a crucial requirement to reach a critical mass of consumers. Containerized applications in the cloud have proven to be effective in meeting this goal. They are characterized by decomposing into independently monitored components, each serving a specific function within the system. This approach enhances maintenance and fault tolerance, but it demands sophisticated technical knowledge for timely implementation. On the flip side, Hadoop also offers a robust solution with distributed features. It utilizes clusters of nodes to store data documents, replicating and allocating data conveniently across the nodes. In terms of performance, it processes queries in parallel across the nodes storing the information, making it an efficient alternative. Additionally, it integrates seamlessly with modern back-end development languages such as Node.js and Django.

4 Problem Definition

Traditional database systems exhibit several drawbacks when compared to distributed database systems. One significant limitation lies in scalability, where traditional databases struggle to efficiently handle growing volumes of data and increased workloads. Distributed databases, on the other hand, excel in scalability by distributing data and processing across multiple nodes.

Another drawback of traditional databases is their lack of fault tolerance, as a single point of failure can disrupt operations. Distributed databases inherently provide better fault tolerance, ensuring continuous availability even if one node fails. This resilience is crucial for maintaining uninterrupted services in dynamic and demanding environments.

Flexibility in handling evolving data schemas is another area where traditional databases fall short. They typically have fixed schemas, making any changes a complex and often disruptive process. In contrast, distributed databases offer greater flexibility, enabling easier adaptation to evolving data structures without significant downtime or migration challenges.

Additionally, distributed databases outperform traditional ones in terms of performance, adaptability to cloud environments, and cost efficiency. Horizontal scaling with commodity hardware becomes more feasible in distributed systems, allowing for cost-effective expansion as opposed to the often expensive vertical scaling required by traditional databases. This adaptability is particularly relevant in the context of cloud computing, where dynamic infrastructure demands a more scalable and flexible approach.

Overall, these drawbacks underscore the advantages offered by distributed databases in addressing the dynamic and complex requirements of modern applications and data management.

5 Proposed Solutions

This section describes the underlying architecture of the proposed solution. In particular, it outlines the resources employed as well as the fragmentation and replication strategies derived to store the data in efficient manner. We strive to design a system architecture that assimilates to that in a production environment, despite working on a local environment. It consists of the following components:

1. Database: MySQL
2. Cluster: MinIO Cluster
3. App
    - Back-end Application Server: Flask python framework
    - Front-end Server: HTML & CSS

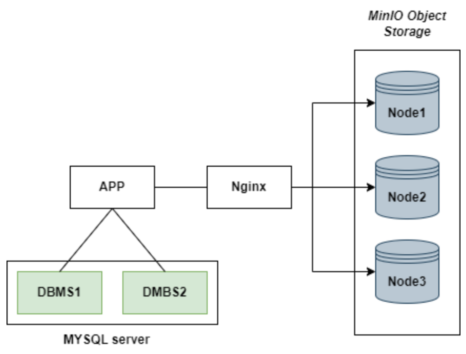


Figure 1: Proposed system architecture

Figure 1 illustrates the architecture and components of our approach. The database component utilizes MySQL, wherein we have implemented two distinct databases, DBMS1 and DBMS2. These databases store fragmented portions of the overall database in accordance with the specific fragmentation requirements outlined in the project. To accommodate the storage of unstructured data, we have deployed MinIO object storage, which consists of three nodes. The details of the deployment and functionalities of MinIO are discussed in section 5.1.

Additionally, we have developed a web application that serves as an interface between users and the MySQL database and MinIO object storage using Flask, which is a python framework for web. This web application facilitates seamless interaction with the database and storage system, allowing us to effortlessly retrieve data and display it within a user-friendly web-based user interface.

Nginx service is used for reverse proxy and load balance. Nginx provides:

1. Load Balancing: Nginx can distribute incoming traffic across multiple MinIO nodes, enhancing the system's performance and reliability.

2. Proxying: Nginx acts as an intermediary for requests from clients, forwarding them to the MinIO servers. This can improve security, manageability, and scalability.

3. Access Point: Clients can connect to the MinIO cluster through a single endpoint, simplifying the network configuration.

4. Handling of Static and Dynamic Content: Nginx excels at serving static content and can be used to efficiently handle dynamic content requests to the MinIO cluster.

Through the utilization of MySQL, MinIO, and our custom web application, our architecture ensures efficient data management, streamlined data retrieval, and a user-friendly web app.

5.1 MinIO Cluster

In this project we used MinIO object storage for storing unstructured articles files like images, videos and article body. MinIO is an open-source object storage server that is compatible with Amazon S3 cloud storage service. It's designed for high performance, scalability, and ease of use[2]. Unlike file storage (which organizes data into a hierarchy of files within folders) or block storage (which segments data into blocks), object storage manages data as objects. Each object includes the data itself, metadata, and a globally unique identifier. MinIO stores these objects in a flat organizational structure, making it highly scalable and efficient for storing large amounts of unstructured data, such as photos, videos, and backups.

5.2 Erasure Coding

Erasure coding is a data protection and redundancy technique used in distributed storage systems like Minio to ensure data durability and availability. It is an alternative to traditional replication methods like RAID (Redundant Array of Independent Disks) or simple replication, and it can be more storage-efficient while providing the same or even higher levels of data resilience.

In Minio, erasure coding is used to divide data into smaller chunks, encode these chunks with additional parity information, and then distribute them across multiple storage nodes. This approach offers fault tolerance, data recovery, and efficient use of storage resources. How erasure coding works:

1. Data Chunking: When a file or object is uploaded to Minio, it is divided into smaller data chunks. The size of these chunks is configurable but is typically a few megabytes in size.

2. Parity Calculation: For each data chunk, Minio calculates one or more parity chunks using mathematical algorithms, such as Reed-Solomon coding. These parity chunks contain additional information that can be used to recover the original data in case of node failures or data corruption.

3. Data and Parity Distribution: The data chunks and parity chunks are distributed across multiple storage nodes or drives in a Minio cluster. Each chunk, whether it's data or parity, is stored on a separate node, ensuring that the loss of any single node does not result in data loss.

4. Fault Tolerance: Erasure coding provides fault tolerance by allowing Minio to reconstruct lost or corrupted data chunks using the remaining available data and parity chunks. The number of parity chunks generated determines the level of fault tolerance. For example, a configuration with 4 data chunks and 2 parity chunks can recover from the loss of up to 2 nodes without data loss.

5. Efficient Storage: Erasure coding is more storage-efficient compared to traditional replication methods. With replication, you might need to store multiple copies of the same data on different nodes. In contrast, erasure coding stores a minimal number of parity chunks, reducing storage overhead.

6. Reconstruction: In the event of a node failure or data corruption, Minio can use the remaining data and parity chunks to reconstruct the lost or corrupted data chunks. This reconstruction process is done on-the-fly when a client requests the data.

Erasure coding is a key feature in Minio that provides high data durability, fault tolerance, and efficient use of storage resources. It allows Minio to continue serving data even in the presence of hardware failures, making it suitable for building highly available and reliable object storage systems. Minio also provides various erasure coding configuration options, allowing users to tailor the redundancy and storage efficiency to their specific needs.

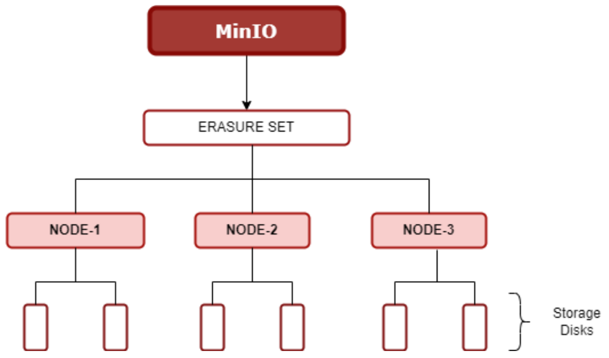


Figure 2: MinIO Deployment with multiple nodes

As shown in figure 2, our MinIO deployment consists of three nodes with two drives each. MinIO initializes with a single erasure set consisting of all 6 drives across all three nodes.

6 Solution Evaluation

In this section we asses the proposed solution relative to the project requirements.

1. Bulk data loading with data partitioning and replica consideration.

2. Efficient execution of data read, insert and update operations.

3. Monitoring the running status of the cluster. MySQL Workbench and MinIO provide real-time relevant metrics concerning the status of the cluster and server.

7 Database tables

The users, articles, user read table generation are provided. Based on the provided tables, we generate be-read and popular-rank tables.

7.1 Be-Read Table Generation

To generate Be-Read table, firstly we fetch the data from users, articles and user read tables. Then we create 4 dictionaries to keep track of read, comment, agree, and share counts, as well as another 5 dictionaries to store lists of uids for read, comment, agree, share, and timestamp of each aid. Then, we group by aid, and reduce by summing the number fields (commentNum, readNum, etc.), appending ids to the corresponding fields (commentUidList, readUidList, etc.). Once the process is done, we proceed to generate a SQL file to load the data into database. Once the SQL file is generated, we continue to fragment it into two different fragments based on Article table with duplication, where category = "science" allocated to DBMS1 and DBMS2, and category = "technology" allocated to DBMS2. We will end up having two new SQL files, and finally we run the SQL file to load the data into database.

7.2 Popular-Rank table Generation

The Popular-Rank table is generated based on the Be-Read table. First we fetch all the data from be-read table, then we create 3 empty arrays for daily articles, weekly articles and monthly articles respectively. We also declare 3 different threshold values to determine the temporal granularity, whether it is daily, weekly or monthly. Then, We will loop through all the data and for each record, we find out the smallest timestamp and add one day, seven days, and thirty days to it in order to create 3 new timestamps as next_day, next_week and next_month. Next, we initialize 3 counters for each temporal granularity as daily_count, weekly_count and monthly_count. We will then find out the timestamp that lies within the range between smallest timestamp to each of the next_day, next_week and next_month timestamp. If it matches, then we will add a count to the counter that it matched. For instance, if the timestamp lies between smallest timestamp and next_day timestamp, then we add a count to daily_count, and so on. After that, we will proceed to check if the counts exceed their respective threshold value to determine the temporal granularity. Once the granularity is decided, we will append the aid into the respective array, i.e. if temporal granularity is daily, we append the aid into daily articles array. Once all the data are done, we will proceed to generate a SQL file, and the final step is to fragment it based on temporal granularity.

8 Result

In this section, we show the result of distributed database system by the use of the app that interacts with database and MinIO server to fetch the data. As discussed earlier we use Flask for this web app.

8.1 Connecting the app to servers

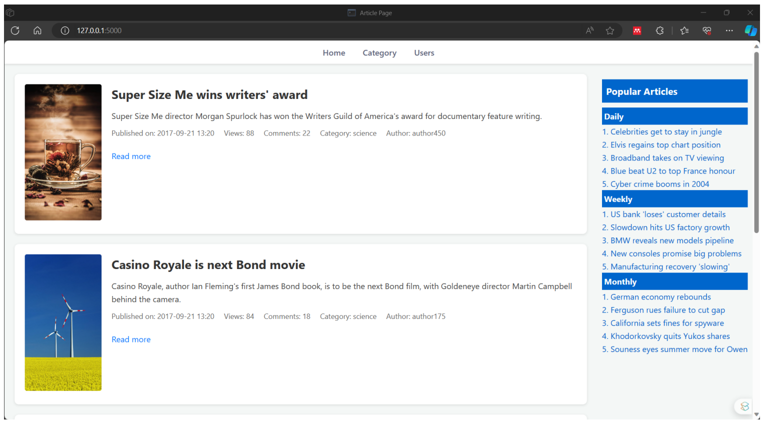
For connecting the Flask app to server we used python libraries pymsql and minio library to connect our app to the servers for making queries to the database and for retrieving data from MinIO server. The following code segment shows how to connect the app to the servers. MinIO client, cursor1, cursor2 are used to fetch database and files.

Listing 1: Connecting the app to servers

```
    minio_client = Minio('127.0.0.1:9000',
                         access_key='minioadmin',
                         secret_key='minioadmin',
                         secure=False)
    def get_db(dbms):
        connection = pymysql.connect(
         host='127.0.0.1',
         user='root',
         password='mypassword',
         database=dbms,
         port=3306
         )
        return connection
    dbms1_conn = get_db("dbms1")
    dmbs2_conn = get_db("dbms2")

    cursor1 = dbms1_conn.cursor()
    cursor2 = dmbs2_conn.cursor()
```

When we start our flask app, it queries the database and fetch articles information and also fetch article's data like images, videos and text from Minio server. The app contains sever pages. The home pages being the first one, it displays article information as shown in figure 3. It also includes daily, weekly and monthly popular articles as shown in the right side of home page. This home page includes preview of each article which include article title, abstract, first image,comments, views, category and author. There is also read more link which takes the user to view articles detail. Popular articles are listed with their title, the title are clickable. Upon clicking one can view the article details.



Super Size Me wins writers' award

Super Size Me director Morgan Spurlock has won the Writers Guild of America's award for documentary feature writing.

Published on: 2017-09-21 13:20 Views: 88 Comments: 22 Category: science Author: author450

Read more

Casino Royale is next Bond movie

Casino Royale, author Ian Fleming's first James Bond book, is to be the next Bond film, with Goldeneye director Martin Campbell behind the camera.

Published on: 2017-09-21 13:20 Views: 84 Comments: 18 Category: science Author: author175

Read more

Popular Articles

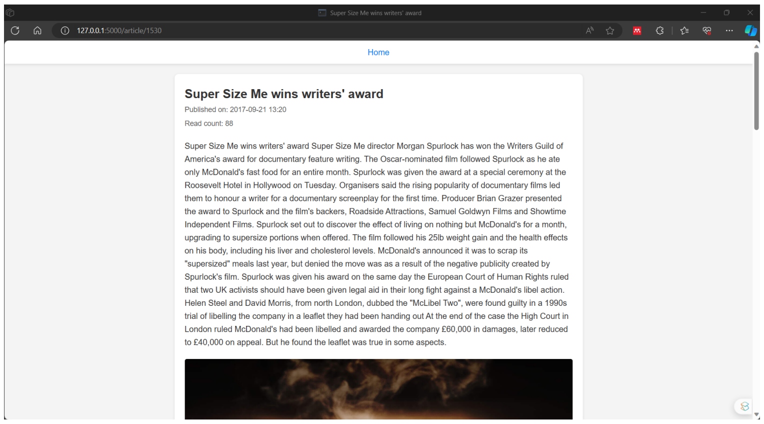
Daily
1. Celebrities get to stay in jungle
2. Elvis regains top chart position
3. Broadband takes on TV viewing
4. Blue beat U2 to top France honour
5. Cyber crime booms in 2004

Weekly
1. US bank 'loses' customer details
2. Slowdown hits US factory growth
3. BMW reveals new models pipeline
4. New consoles promise big problems
5. Manufacturing recovery 'slowing'

Monthly
1. German economy rebounds
2. Ferguson rues failure to cut gap
3. California sets fines for spyware
4. Khodorkovsky quits Yukos shares
5. Souness eyes summer move for Owen

Figure 3: Web app home page

The article detail page, which is shown in Figure 4 displays detail for a particular article.



Super Size Me wins writers' award
127.0.0.1:5000/article/1530
Home

Super Size Me wins writers' award

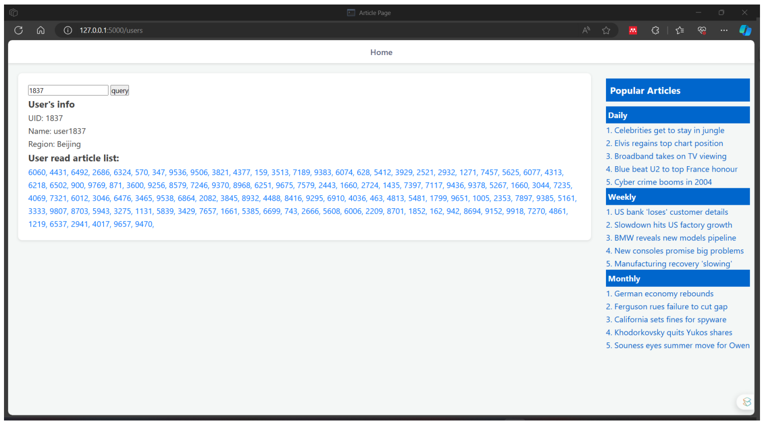
Published on: 2017-09-21 13:20

Read count: 88

Super Size Me wins writers' award Super Size Me director Morgan Spurlock has won the Writers Guild of America's award for documentary feature writing. The Oscar-nominated film followed Spurlock as he ate only McDonald's fast food for an entire month. Spurlock was given the award at a special ceremony at the Roosevelt Hotel in Hollywood on Tuesday. Organisers said the rising popularity of documentary films led them to honour a writer for a documentary screenplay for the first time. Producer Brian Grazer presented the award to Spurlock and the film's backers, Roadside Attractions, Samuel Goldwyn Films and Showtime Independent Films. Spurlock set out to discover the effect of living on nothing but McDonald's for a month, upgrading to supersize portions when offered. The film followed his 25lb weight gain and the health effects on his body, including his liver and cholesterol levels. McDonald's announced it was to scrap its "supersized" meals last year, but denied the move was as a result of the negative publicity created by Spurlock's film. Spurlock was given his award on the same day the European Court of Human Rights ruled that two UK activists should have been given legal aid in their long fight against a McDonald's libel action. Helen Steel and David Morris, from north London, dubbed the "McLibel Two", were found guilty in a 1990s trial of libelling the company in a leaflet they had been handing out At the end of the case the High Court in London ruled McDonald's had been libelled and awarded the company £60,000 in damages, later reduced to £40,000 on appeal. But he found the leaflet was true in some aspects.

Figure 4: Article detail

We can also query with user ID to view the user read list. Figure 5 shows the demo. The user information include users name, region and list of article IDs that user read. This article IDs are clickable and takes the user to the article detail page when clicked.



1837

User's info

UID: 1837

Name: user1837

Region: Beijing

User read article list:

6060, 4431, 6492, 2686, 6324, 570, 347, 9536, 9506, 3821, 4377, 159, 3513, 7189, 9383, 6074, 628, 5412, 3929, 2521, 2932, 1271, 7457, 5625, 6077, 4313, 6218, 6502, 900, 9769, 871, 3600, 9256, 8579, 7246, 9370, 8968, 6251, 9675, 7579, 2443, 1660, 2724, 1435, 7397, 7117, 9436, 9378, 5267, 1660, 3044, 7235, 4069, 7321, 6012, 3046, 6476, 3465, 9538, 6864, 2082, 3845, 8932, 4488, 8416, 9295, 6910, 4036, 463, 4813, 5481, 1799, 9651, 1005, 2353, 7897, 9385, 5161, 3333, 9807, 8703, 5943, 3275, 1131, 5839, 3429, 7657, 1661, 5385, 6699, 743, 2666, 5608, 6006, 2209, 8701, 1852, 162, 942, 8694, 9152, 9918, 7270, 4861, 1219, 6537, 2941, 4017, 9657, 9470,

Popular Articles

Daily

1. Celebrities get to stay in jungle
2. Elvis regains top chart position
3. Broadband takes on TV viewing
4. Blue beat U2 to top France honour
5. Cyber crime booms in 2004

Weekly

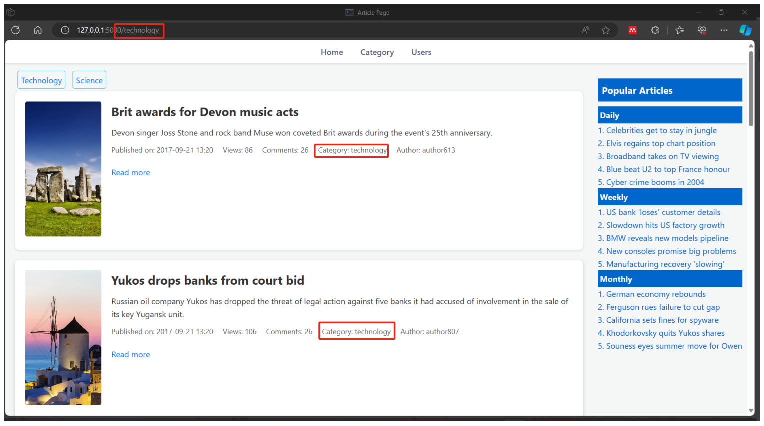
1. US bank 'loses' customer details
2. Slowdown hits US factory growth
3. BMW reveals new models pipeline
4. New consoles promise big problems
5. Manufacturing recovery 'slowing'

Monthly

1. German economy rebounds
2. Ferguson rues failure to cut gap
3. California sets fines for spyware
4. Khodorkovsky quits Yukos shares
5. Souness eyes summer move for Owen

Figure 5: User information

We also implemented filter the articles based on the category. This helps users to get the category of the articles they want.



Brit awards for Devon music acts

Devon singer Joss Stone and rock band Muse won coveted Brit awards during the event's 25th anniversary.

Published on: 2017-09-21 13:20 Views: 86 Comments: 26 Category: technology Author: author613

Yukos drops banks from court bid

Russian oil company Yukos has dropped the threat of legal action against five banks it had accused of involvement in the sale of its key Yugansk unit.

Published on: 2017-09-21 13:20 Views: 106 Comments: 26 Category: technology Author: author807

Popular Articles

Daily

1. Celebrities get to stay in jungle
2. Elvis regains top chart position
3. Broadband takes on TV viewing
4. Blue beat U2 to top France honour
5. Cyber crime booms in 2004

Weekly

1. US bank 'loses' customer details
2. Slowdown hits US factory growth
3. BMW reveals new models pipeline
4. New consoles promise big problems
5. Manufacturing recovery 'slowing'

Monthly

1. German economy rebounds
2. Ferguson rues failure to cut gap
3. California sets fines for spyware
4. Khodorkovsky quits Yukos shares
5. Souness eyes summer move for Owen

Figure 6: Filter Articles by category

9 Monitoring

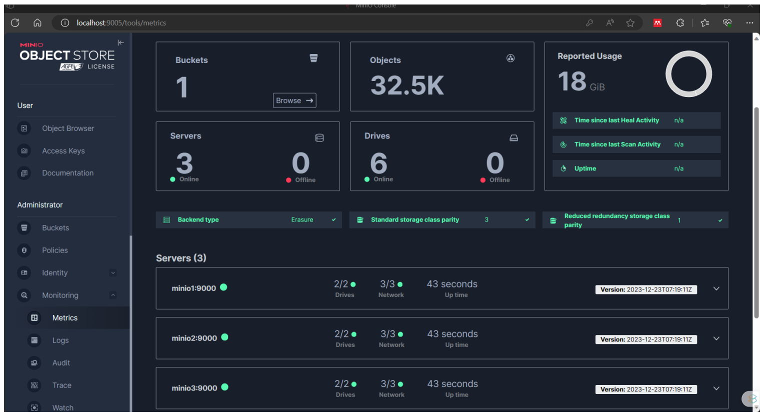
9.1 Cluster Monitoring

For the cluster monitoring we used builtin MinIO monitoring. MinIO provides point-in-time metrics on cluster status and operations. MinIO Monitoring is a feature within the MinIO ecosystem, focused on tracking and analyzing the performance and health of a MinIO storage cluster.

To visually represent these metrics, the MinIO Console is equipped with a graphical interface that displays this data, allowing for an intuitive and comprehensive understanding of the cluster's performance and operations. MinIO Monitoring involves the following functionalities.

- Cluster Health and Performance: Monitoring involves keeping an eye on the overall health and performance of the MinIO cluster. This includes metrics like server uptime, system health, and resource utilization (CPU, memory, network bandwidth).
- Data Usage and Efficiency: It tracks data usage statistics, including the amount of data stored, data transfer rates, and access patterns. This is crucial for optimizing storage efficiency and planning for capacity expansion.
- Tracking: MinIO Monitoring tracks various operations performed on the storage cluster, such as the number of read/write requests, any errors or failures in operations, and the latency of these operations.
- Alerts and Notifications: The monitoring system can be configured to send alerts or notifications in case of any anomalies, failures, or significant changes in performance metrics. This helps in proactive troubleshooting and maintenance.
- Security Monitoring: It may also include monitoring access patterns and security logs to ensure that the data stored in MinIO is secure and accessed only by authorized entities.

Figure 7 shows cluster metrics including number of servers, buckets, total objects stored and total drives used across the clusters. In addition to reporting the statics it also tells whether the servers are online or not.



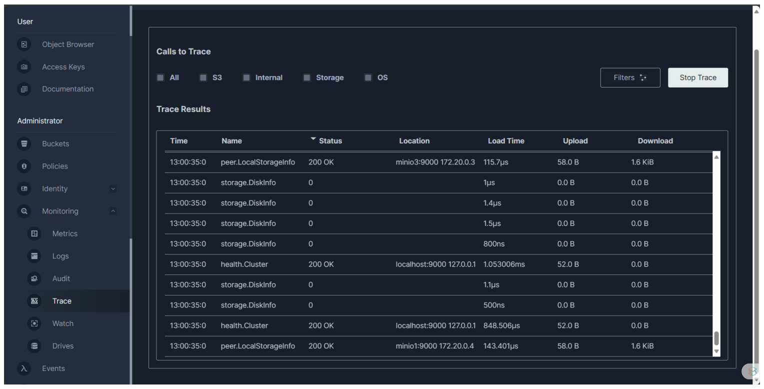
Buckets 1
Objects 32.5K
Reported Usage 18 GiB
Time since last Heal Activity n/a
Time since last Scan Activity n/a
Uptime n/a
Servers 3 Online 0 Offline
Drives 6 Online 0 Offline
Backend type Erasure
Standard storage class parity 3
Reduced redundancy storage class parity 1

Servers (3)

```
minio1:9000   2/2 Drives   3/3 Network   43 seconds Up time   Version: 2023-12-23T07:19:11Z
minio2:9000   2/2 Drives   3/3 Network   43 seconds Up time   Version: 2023-12-23T07:19:11Z
minio3:9000   2/2 Drives   3/3 Network   43 seconds Up time   Version: 2023-12-23T07:19:11Z
```

Figure 7: MinIO Monitoring Metrics

In the monitoring, we also have trace which provides the real-time operation monitoring. This monitoring includes the status, location, Load Time, Upload time and Download. To trace this metrics, first we need to start tracing.



Calls to Trace

All S3 Internal Storage OS

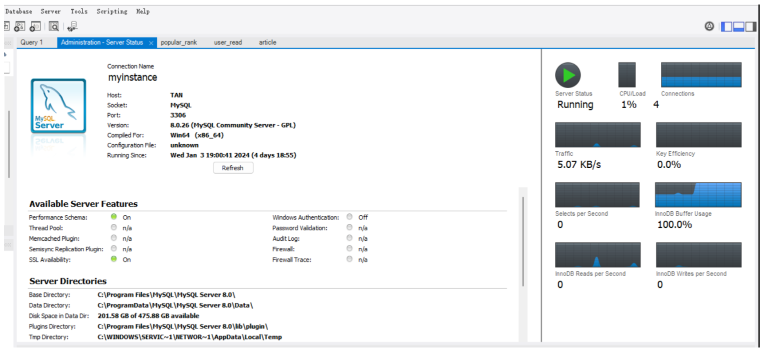
Trace Results

Time	Name	Status	Location	Load Time	Upload	Download
13:00:35:0	peer.LocalStorageInfo	200 OK	minio3:9000 172.20.0.3	115.7µs	58.0 B	1.6 KiB
13:00:35:0	storage.DiskInfo	0		1µs	0.0 B	0.0 B
13:00:35:0	storage.DiskInfo	0		1.4µs	0.0 B	0.0 B
13:00:35:0	storage.DiskInfo	0		1.5µs	0.0 B	0.0 B
13:00:35:0	storage.DiskInfo	0		800ns	0.0 B	0.0 B
13:00:35:0	health.Cluster	200 OK	localhost:9000 127.0.0.1	1.053006ms	52.0 B	0.0 B
13:00:35:0	storage.DiskInfo	0		1.1µs	0.0 B	0.0 B
13:00:35:0	storage.DiskInfo	0		500ns	0.0 B	0.0 B
13:00:35:0	health.Cluster	200 OK	localhost:9000 127.0.0.1	848.506µs	52.0 B	0.0 B
13:00:35:0	peer.LocalStorageInfo	200 OK	minio1:9000 172.20.0.4	143.401µs	58.0 B	1.6 KiB

Figure 8: MinIO Tracing

9.2 Database monitoring

MySQL provides it's server status monitoring as well. In MySQL workbench, there server tab from which we can view server tab[3]. This feature provides a comprehensive overview of the health and performance of a MySQL server, including it's status (running/stopped), CPU utilization, connections, Traffic, InnoDB Buffer usage, selects per Second and Read/write per second.


Database Server Tools Scripting Help
Query 1
Administration - Server Status
popular_rank
user_read
article
MySQL Server
Refresh

Connection Name
myinstance

Host:	TAN
Socket:	MySQL
Port:	3306
Version:	8.0.26 (MySQL Community Server - GPL)
Compiled For:	Win64 (x86_64)
Configuration File:	unknown
Running Since:	Wed Jan 3 19:00:41 2024 (4 days 18:55)

Available Server Features

Performance Schema:	On	Windows Authentication:	Off
Thread Pool:	n/a	Password Validation:	n/a
Memcached Plugin:	n/a	Audit Log:	n/a
Semisync Replication Plugin:	n/a	Firewall:	n/a
SSL Availability:	On	Firewall Trace:	n/a

Server Directories

Base Directory:	C:\Program Files\MySQL\MySQL Server 8.0\
Data Directory:	C:\ProgramData\MySQL\MySQL Server 8.0\Data\
Disk Space in Data Dir:	201.58 GB of 475.88 GB available
Plugins Directory:	C:\Program Files\MySQL\MySQL Server 8.0\lib\plugin\
Tmp Directory:	C:\WINDOWS\SERVIC~1\NETWOR~1\AppData\Local\Temp

Server Status Running
CPU/Load 1%
Connections 4
Traffic 5.07 KB/s
Key Efficiency 0.0%
Selects per Second 0
InnoDB Buffer Usage 100.0%
InnoDB Reads per Second 0
InnoDB Writes per Second 0

Figure 9: MySQL server status Monitoring

10 Conclusion

This project presented a fantastic learning experience. It challenged our capabilities and encouraged the practical application of the concepts explained in the lectures. We are able to gain deep and insight understanding on the knowledge of the advanced big data management techniques in a distributed environment through a hands-on software design and implementation experiment. Although our solution presented in this paper predominantly relies on

MySQL Workbench and MinIO as the storage system and clustering, we did try to implement Hadoop, yet it had a steep learning curve and we ran out of time. Thus, we resorted to MySQL Workbench and MinIO and managed to fulfill all of the project requirements.

10.1 Future Work

a. Hot / Cold Standby DBMS for fault tolerance

b. Expansion at the DBMS-level allowing a new DBMS server to join

c. Dropping a DBMS server at will

d. Data migration from one data center to others

References

[1] Cornelia Győrödi, Robert Győrödi, George Pecherle, and Andrada Olah. A comparative study: Mongodb vs. mysql. In *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, pages 1–6, 2015.

[2] MinIO. Minio object storage for linux, 2024. Accessed: 2024-01-04.

[3] MySQL. Mysql workbench manual server status, 2024. Accessed: 2024-01-04.

A Manual

In the section we provide manual how to setup the database and MinIO cluster on Docker. In addition, we also provide steps on how to run the app to test the system.

A.1 Mysql server

Mysql Workbench is used to setup the two databases, namely DBMS1 and DBMS2 Run the following sql command in Mysql Workbench.

Listing 2: Database creation

```sql
CREATE DATABASE DBMS1;
CREATE DATABASE DBMS2;
```

The above mysql statement creates our databases. Next we need to creates tables and populate the tables with our data. Since table creation and data insertion code are too long, we have created github link were can find all the project setup. Clone the the github repository which includes the all table creation and insertion of fragmented tables into respective database.

Listing 3: Clone github repository

```
$ git clone https://github.com/zedonglim/DDBMS.git
```

The fragmented user, article, user read, be read and popular sql statement in this github repo can be run directory in mysql workbench to populate the tables. Their names indicates where to be inserted. For example `article_dbms1.sql` is article data to be inserted into database DMBS1.

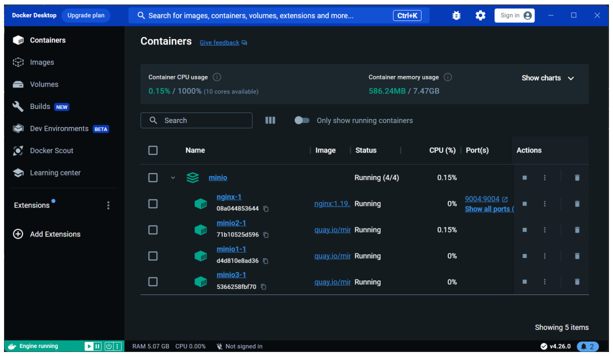
A.2 MinIO Setup

MinIO cluster is configure on docker container using docker compose. The MinIO docker compose file is also in the github repository. The requires the intallation docker. Assuming the docker is installed on our machine, we can run the following commands to set up our minio cluster with 3 nodes.

Listing 4: MinIO set up on docker compose

```
// naviagate into directory where the docker compose
   file is located
$ cd minio
$ docker-compose up -d
```

This sets up the docker container with three minio nodes, namely minio1, minio2 and minio3 and nginx images on docker as shown in Figure 10 below.



Container CPU usage 0.15% / 1000% (10 cores available)
Container memory usage 586.24MB / 7.47GB

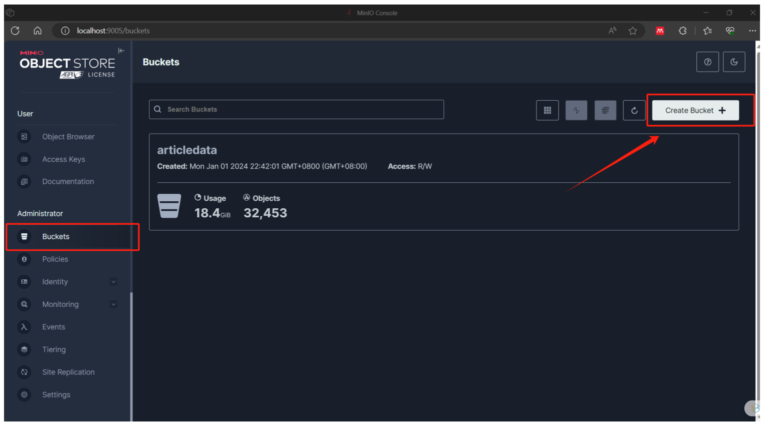
Name	Image	Status	CPU (%)	Port(s)	Actions
minio		Running (4/4)	0.15%		
nginx-1 08a044853644	nginx:1.19.	Running	0%	9004:9004 Show all ports (	
minio2-1 71b10525d596	quay.io/mir	Running	0.15%		
minio1-1 d4d810e8ad36	quay.io/mir	Running	0%		
minio3-1 5366258fbf70	quay.io/mir	Running	0%		

Showing 5 items

RAM 5.07 GB CPU 0.00%

Figure 10: Docker container

After setting up, we can visit the MinIO console interface on our local machine. We can also see from the logs where the console API is running. As indicated in docker compose file. It runs at localhost:9005. Visiting think link will open console web interface we can login using the root user and password set in docker compose file. After login, the first think we need to do is to create our bucket, which is the object storage bucket, as shown in Figure 11.



articledata
Created: Mon Jan 01 2024 22:42:01 GMT+0800 (GMT+08:00) **Access:** R/W
Usage 18.4GiB
Objects 32,453

Figure 11: Creating bucket on MinIO cluster

Then we upload our data the bucket we just created. MinIO will distribute the data among the drives and across the cluster with replication on the three nodes with just setup using Erasure coding technique discussed in section 5.2. Figure 12 indicates bucket and how to upload data. We can select a folder and upload the whole data at once.

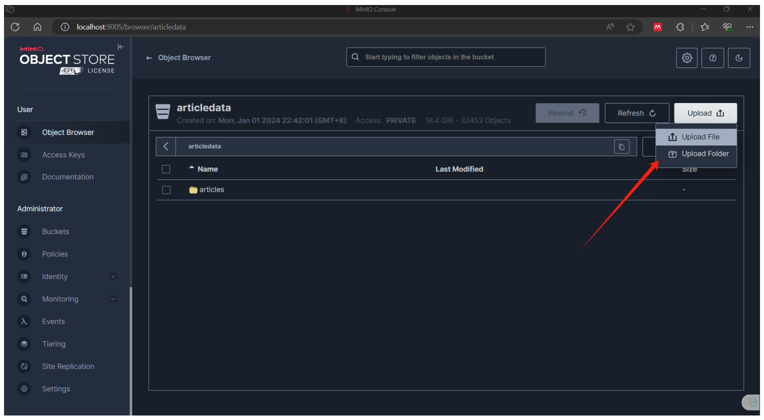


Figure 12: Uploading data to MinIO Cluster

A.3 Running The APP

As mentioned earlier Flask framework is used to implement the system. The app is located in web director in inside the github repo.

Listing 5: Running the APP

```
$ cd web
$ flask --app newsapp --debug run
```

Then visit the http://127.0.0.1:5000 the localhost link on which the app is running. This will start the web interface.

B Group Members Role

Name	Role
Lim Ze Dong	Database creation and fragmentation
Namie Turi Abu	MinIO cluster and web app

Table 1: Group members role