Retrieving and Processing YouTube Comment Data

IST 769 ADVANCED BIG DATA MANAGEMENT – FINAL PROJECT, TEAM 1 – DR. BLOCK

Kirk Copley, William Blumrosen, Robert Ransom
SYRACUSE UNIVERSITY ISCHOOL | DECEMBER 2023

Introduction and Problem Statement

Since the organization was founded in 2005, YouTube has provided digital media to billions of consumers for anything from infamous cat videos, motivational groups, or instruction on how to work with a new coding language. Particularly, music videos after the fall of MTV have become a major component of the content delivery service provided by YouTube, allowing well known and up-and-coming artists to produce and distribute their chosen form of media.

The purpose of this experiment is to gather comments and information from music videos released by the artist Taylor Swift and show methods for using various non-relational databases to store, manipulate, and demonstrate relationships between media. Using Spark, MongoDB, Neo4j, and Cassandra, the experiment will use comment data to show the different use cases of these databases and the transfer of data between platforms. Rather than chase a specific analytical question, the intent is to show the implementation of multiple technologies for different use cases in Big Data.

Using a generalized process, the solution for the particular question of "how can a system of systems be used to interpret information about generated media" is scalable from examination of a single video to an entire library of artistic works. The goal of the program itself is to run fully with minimal manual interaction from the analyst/engineer to gather data and distribute to multiple storage solutions for analysis later. Once the data is stored for the desired number of videos, relationships can be identified as well as trends for commenting and popularity. The system itself can be applied to any artist and desired number of videos, demonstrating flexibility and scalability of the solution.

Methodology

Generally, the program functions out of Spark with a minimal number of commands required in the Command Prompt to initialize functionality with docker-compose. Initially, the program builds the dataset with a preformatted Google Application-Program Interface (API) that iterates multiple times to gather Video Title and Comment Data; number of videos required for the analyst determines the total number of iterations to construct the dataset. Once the data is ingested by Spark in JSON format, it is stored in MongoDB. From Mongo, the data is read into Spark and stored separately into Cassandra's wide-table format. The data is then read into Neo4j for relationship analysis with the organic Graph Database function. Spark is also used to conduct multiple Pandas representation of popular comments and to direct function for each database system employed.

Command Line Operations

To initialize the required systems using the Azure Lab Virtual Machine, the following commands are required:

```
cd .\advanced-databases\

docker-compose start jupyter mongo cassandra neo4j

# Verify the programs are running with 'ps'

docker-compose ps jupyter mongo cassandra neo4j
```

All other commands are initiated from either Spark or a system's web interface (Neo4j for relationships).

Data Flow

The data flow described above is visually depicted in Figure 1, showing ingestion of data from YouTube comments into Spark and the summary distribution into multiple database technologies for future analysis. As described, the movement of data is curated through Spark allowing for error tolerance in the process and manipulation prior to storage in a new system. Furthermore, it allows for Python-based analysis at multiple steps along the data flow where particular business questions can be developed and answered. Analysis can be injected into the Jupyter Notebook at various locations without changing the overall function of the program. The resulting program can also be run in its entirety prior to analysis, allowing multiple groups to work on different formats of the same data simultaneously without interrupting another's functionality.

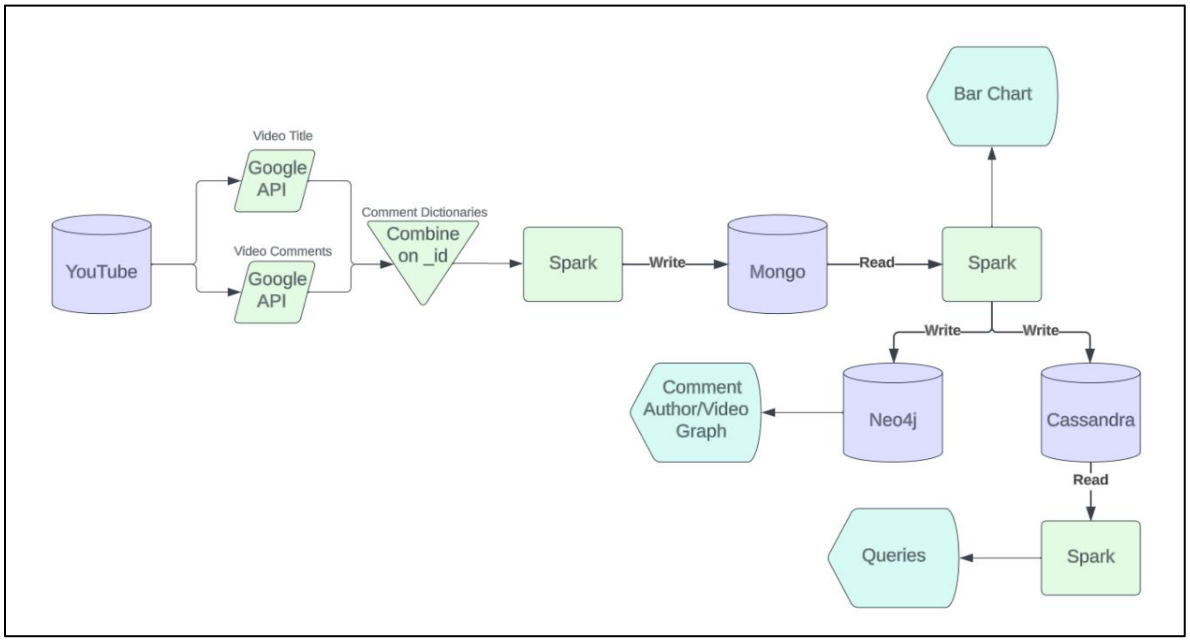


Figure 1 Conceptual data flow from ingest to visualization.

Data Description

As comments are read using the Google API, "comment_id" is used to ensure no duplication takes place in the ingested data. Relevant information is described below along with stored attribute names.

Attribute	Type	Description
_id	text	Comment global unique ID.
author_display_name	text	Original Poster display name.
datetime_posted	timestamp	Date/Time of post.
like_count	bigint	Like count at time of pull.
repliescount	bigint	Replies count at time of pull.
text_original	text	Comment content.
video_id	text	Video global unique ID.
video_title	text	Video title.

Figure 2 Data attributes, format/type, and description.

The API call retrieves the 100 most recent comments and related data from selected videos. As the data is retrieved, it is stored in MongoDB from the designated "video_list." Further iterations would require additional video IDs to be added to the list; a full production program could integrate an Elasticsearch database to allow users to select filters and automatically pass video IDs to the API. This capability is beyond the scope of this experiment.

Implementation of Big Data Systems

Spark

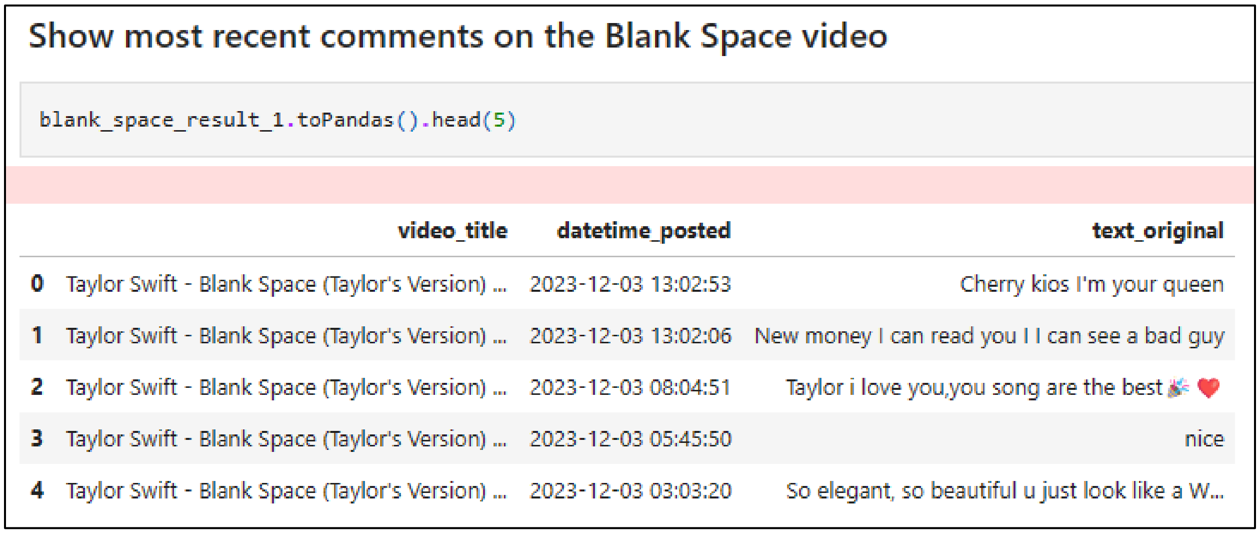
This experiment uses a combination of big data systems, multiple storage solutions and visualization capabilities. Spark serves as the central hub for ingesting and transforming the data as it traverses from one solution to another.

MongoDB

MongoDB is the initial entry and storage for all comment data, validating that no duplication occurs between the elements with total counts of comments and videos retrieved from the API call.

Cassandra

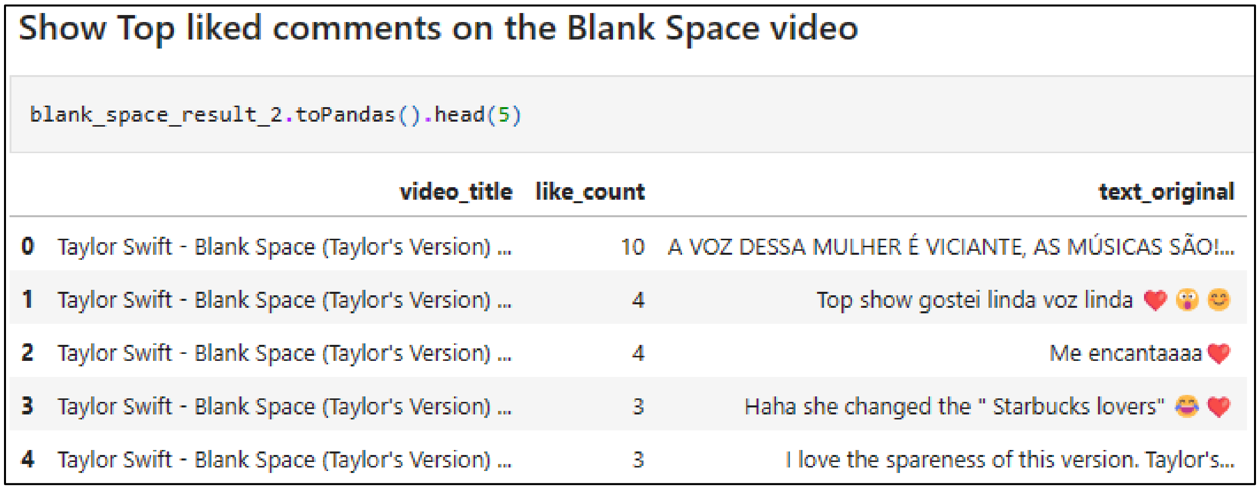
Secondly, the comment data is read into Spark, column labels are manipulated to comply with Cassandra limitations, and stored in the wide-column database. The intent is to pass Cassandra Query Language (CQL) commands from Spark to Cassandra to depict the most recent and most liked comments for a particular video without the additional implementation of the Drill Storage plugin to apply SQL commands to MongoDB data; Figures 3 and 4 depict these results from Cassandra with User Defined Functions.


Show most recent comments on the Blank Space video

```
blank_space_result_1.toPandas().head(5)
```

	video_title	datetime_posted	text_original
0	Taylor Swift - Blank Space (Taylor's Version) ...	2023-12-03 13:02:53	Cherry kios I'm your queen
1	Taylor Swift - Blank Space (Taylor's Version) ...	2023-12-03 13:02:06	New money I can read you I I can see a bad guy
2	Taylor Swift - Blank Space (Taylor's Version) ...	2023-12-03 08:04:51	Taylor i love you,you song are the best 🎉 ❤️
3	Taylor Swift - Blank Space (Taylor's Version) ...	2023-12-03 05:45:50	nice
4	Taylor Swift - Blank Space (Taylor's Version) ...	2023-12-03 03:03:20	So elegant, so beautiful u just look like a W...

Figure 3 Most recent comments on Blank Space Lyrics Video.



Show Top liked comments on the Blank Space video

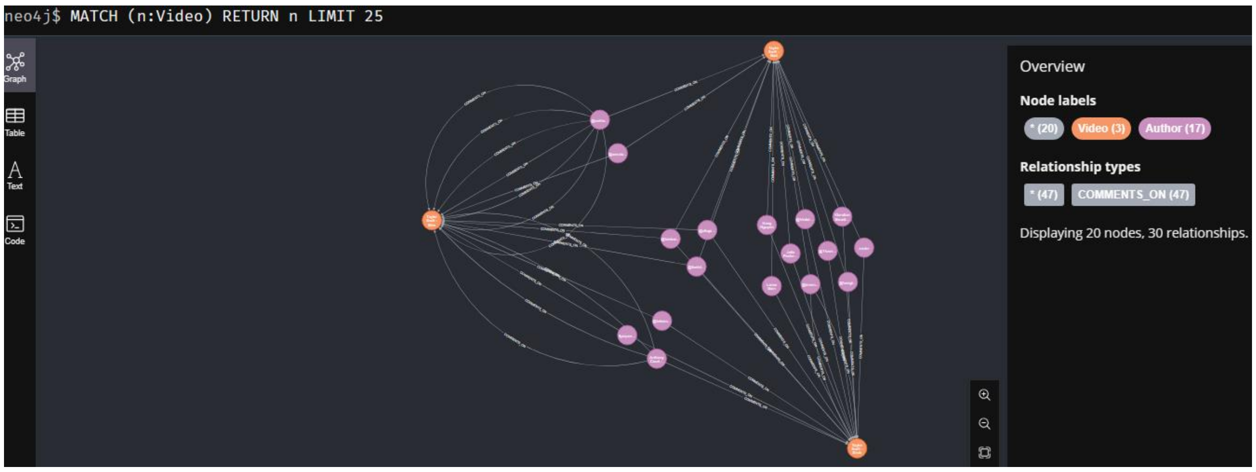
```
blank_space_result_2.toPandas().head(5)
```

	video_title	like_count	text_original
0	Taylor Swift - Blank Space (Taylor's Version) ...	10	A VOZ DESSA MULHER É VICIANTE, AS MÚSICAS SÃO!...
1	Taylor Swift - Blank Space (Taylor's Version) ...	4	Top show gostei linda voz linda ❤️😲😊
2	Taylor Swift - Blank Space (Taylor's Version) ...	4	Me encantaaaa ❤️
3	Taylor Swift - Blank Space (Taylor's Version) ...	3	Haha she changed the " Starbucks lovers" 😂❤️
4	Taylor Swift - Blank Space (Taylor's Version) ...	3	I love the spareness of this version. Taylor's...

Figure 4 Most liked comments on Blank Space Lyrics Video.

Neo4j

Lastly, Neo4j is employed to visualize the relationships between videos and commenters. By linking Author who Commented on a Video, one can visualize commenters that follow the artist and initiate/respond to comments for single or multiple videos. Figure 5 provides the visual representation of the Author to Video, Comment relationship, identifying several users which commented multiple times on a single video (either a response or multiple entries) and only three (with the criteria limited to 25) who commented on all three selected videos.



```
neo4j$ MATCH (n:Video) RETURN n LIMIT 25
```

Overview

Node labels

* (20) Video (3) Author (17)

Relationship types

* (47) COMMENTS_ON (47)

Displaying 20 nodes, 30 relationships.

Figure 5 Neo4j relationship visualization -- MATCH p = (a:Author)-[:COMMENTS_ON]->(v:Video) RETURN p;.

As additional videos are added to the search, and visualizations are expanded to incorporate the hundreds of comments and replies for each, the readability of the Neo4j graph product quickly reduces for human analysts.

Results And Analysis

MongoDB

The initial storage of comment data in MongoDB served as a foundation for the project, validating the uniqueness of each comment and ensuring no duplications. The total counts of comments and videos were 478 and 3, respectively.

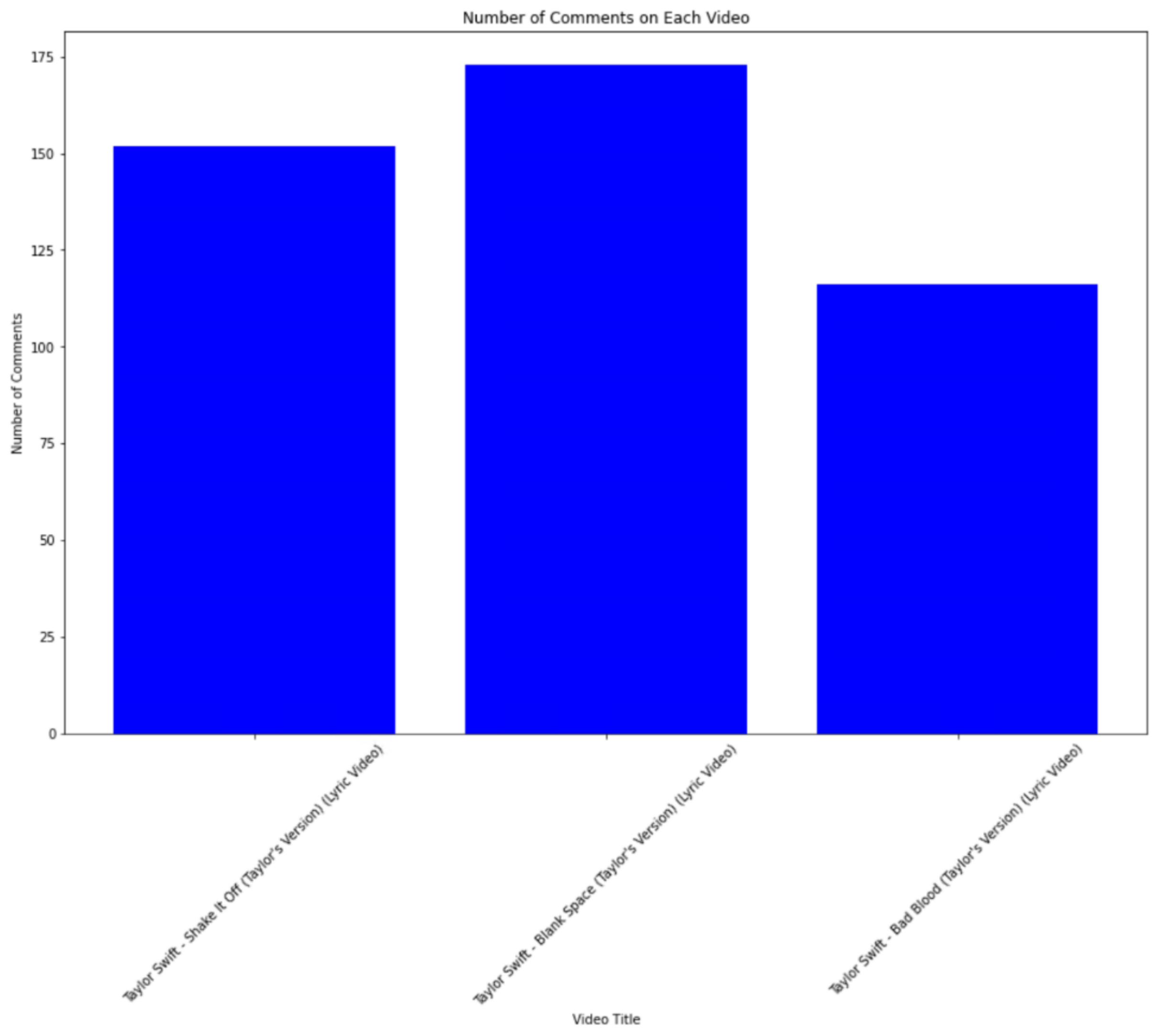


Figure 6 Visualization of comments per video; matplotlib used to visualize both bar and line graphs.

The figure above shows the number of comments per video. Giving us a clear understanding of which video received more engagement and attention.

Cassandra

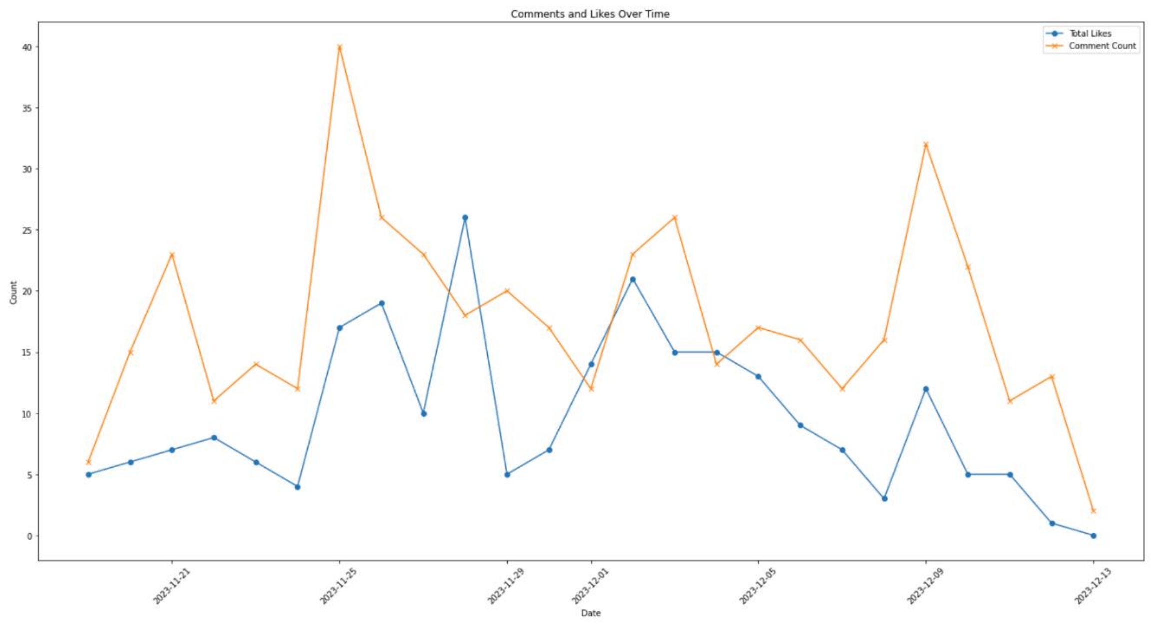
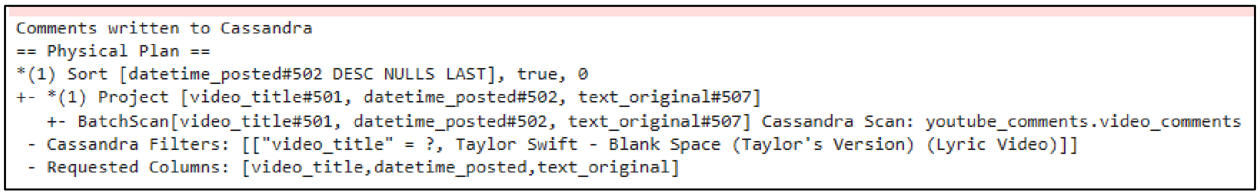


Figure 7 Comments and Likes over time; description below.

Above is the number of likes over time for the top comments. This shows the popularity of each comment and how many times people interacted with it, as well as the date of which comments were posted.



```
Comments written to Cassandra
== Physical Plan ==
*(1) Sort [datetime_posted#502 DESC NULLS LAST], true, 0
+- *(1) Project [video_title#501, datetime_posted#502, text_original#507]
   +- BatchScan[video_title#501, datetime_posted#502, text_original#507] Cassandra Scan: youtube_comments.video_comments
 - Cassandra Filters: [["video_title" = ?, Taylor Swift - Blank Space (Taylor's Version) (Lyric Video)]]
 - Requested Columns: [video_title,datetime_posted,text_original]
```

Figure 8 spark.sql(blank_space_query_1).explain() results showing Physical Plan of Cassandra Query.

The Physical Plan above is provided during execution of Cassandra queries to illustrate the Directed Acyclic Graph (DAG) of the Spark function. ".explain()" shows the requests are properly passed through to the Cassandra node to conduct filtering and query as close as possible to the source without transmitting large volumes of data to Spark to process. In larger data sets, this is important for optimizing process resources and reducing the time to response for business-related queries; an improperly constructed query or wide-column database indexes would slow responses and reduce system efficacy.

Scalability and Flexibility

The project highlighted the scalability and flexibility of the implemented solution. As additional videos and comments were added, the systems proved capable of handling increased data volumes. The use of Spark as a central hub allowed for efficient data processing and transfer between different database systems. This approach not only demonstrated the technical feasibility of the solution but also its applicability to a broader range of scenarios beyond the scope of this project.

Challenges

One of the challenges analyzing YouTube comments is managing the sheer volume of data. YouTube generates an enormous amount of daily comments. This can generate problems in terms of data storage, processing, and analysis. The project depended on YouTube's API to fetch comments and retrieve data, which has strict rate limits imposed by Google. Once we exceeded these limits the API denied any further request given. This limitation can slow down data collection, especially for videos on Taylor Swift, who has an extremely high popularity and large fanbase. Although the purpose of this project was to learn the ins and outs of different ways to handle data. Integrations of various Big Data technologies like Spark, MongoDB, Cassandra, and Neo4j, require persistent and careful planning and optimization so that they all flow well together.

Conclusion

The purpose of this experiment was to develop a repeatable, fault-tolerant process for ingesting, comparing and visualizing comment data for a particular set of videos on YouTube. This process was used for Taylor Swift videos but could easily be applied to any artist that posts content to the streaming service. We set out to employ only a handful of storage and visualization techniques and quickly found methods that can be combined with nearly every method for transforming and storing data in a horizontally scalable and distributed architecture. For the sake of time available, the three most responsive systems were selected for integration with Spark serving as the central hub for commands and data manipulation.

Ultimately, the experiment in Big Data systems and technologies opened the door for more user experience and interaction possibilities, integration with Elasticsearch, and endless areas for data analysis. The groundwork for ingesting data and responsive storage solutions is well founded and integration through Spark and Jupyter Notebook creates a repetitive, fault-tolerant capability to bring these technologies together to share this data story.