

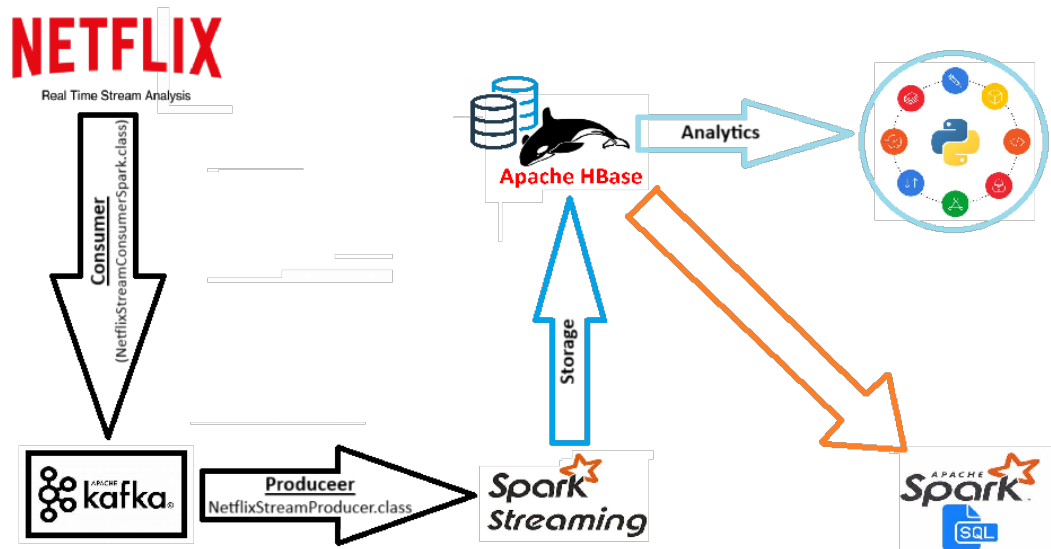
Final Project CS523 (Big Data Technology)

Netflix Movie Data Analysis

[Project GitHub Link for BDT-FINAL-PROJECT \(click to browse\)](#)

Project Overview

- ❖ Objective: The project aims to identify top-rated Netflix movies based on user votes and ratings, providing real-time insights into movie popularity.
- ❖ Data Ingestion: Apache Kafka is used for streaming movie data (in CSV format) into Kafka topics. This enables real-time data ingestion for continuous updates.
- ❖ Data Storage and Processing: Data is processed using Apache Spark, which applies transformations to clean, filter, and aggregate the data. Processed data is stored in HBase for scalable, high-performance access. The pipeline ensures efficient handling of large datasets.
- ❖ Analysis & Visualization: Python handles the analysis and visualization. Libraries like Pandas and Matplotlib are used to identify trends, such as the most frequently top-rated movies. These insights are visualized through charts and graphs to demonstrate movie popularity.
- ❖ System Development: The core system is developed in Java, ensuring seamless integration with Kafka and HBase. Maven is used for managing dependencies and automating builds, supporting smooth development workflows.
- ❖ Architecture and process flow:
 1. Kafka Producer: Streams movie data.
 2. Spark Streaming: Processes real-time data from Kafka.
 3. HBase: Stores processed data for scalable access.
 4. Python Analytics: Runs queries and generates visualizations.



❖ Tools and Technologies:

- ✓ Hadoop: Version 2.6.0-cdh5.13.0
- ✓ HBase: Version 1.2.0-cdh5.13.0
- ✓ Spark: Version 1.6.0
- ✓ Kafka: Version 2.13-2.8.2
- ✓ Java JDK & JRE: Version 1.8.0-181
- ✓ Python
- ✓ Platform: CentOS Version 6.7
- ✓ IDE: Eclipse LUNA 4.4.2, VS Code

Project Implementation Details:

1. A custom Java producer application was developed to read and stream Netflix movie data from a CSV file. This application streams the data to a Kafka topic

for further processing in real-time.

```
24/09/29 21:29:45 INFO utils.AppInfoParser: Kafka version: 2.8.2
24/09/29 21:29:45 INFO utils.AppInfoParser: Kafka commitId: 3146c6ff4a24cc24
24/09/29 21:29:45 INFO utils.AppInfoParser: Kafka startTimes: 1727670585282
24/09/29 21:29:46 INFO clients.Metadata: [Producer clientId=producer-1] Resetting the last seen epoch of partition movie-0 to 0 since the associated topicid changed from null to TDw73-KwQNIcPrf1gZ9EKA
24/09/29 21:29:46 INFO clients.Metadata: [Producer clientId=producer-1] Cluster ID: 0T9TchvtyackGjNjV3305g
Sent Packet to Kafka ....David Attenborough: A Life on Our Planet,2020,9,31180,83,documentary,GB
Sent Packet to Kafka ....Inception,2010,8.8,2268288,148,scifi,GB
Sent Packet to Kafka ....Forrest Gump,1994,8.8,1994599,142,drama,US
Sent Packet to Kafka ....Amélie,2001,8.7,26595,168,comedy,IN
Sent Packet to Kafka ....Bo Burnham: Inside,2021,8.7,44074,87,comedy,US
Sent Packet to Kafka ....Saving Private Ryan,1998,8.6,1346020,169,drama,US
Sent Packet to Kafka ....Django Unchained,2012,8.4,1472668,165,western,US
Sent Packet to Kafka ....Dangal,2016,8.4,188247,161,action,IN
Sent Packet to Kafka ....Bo Burnham: Make Happy,2016,8.4,14356,68,comedy,US
Sent Packet to Kafka ....Louis C.K.: Hilarious,2010,8.4,11973,84,comedy,US
Sent Packet to Kafka ....Dave Chappelle: Sticks & Stones,2019,8.4,25687,65,comedy,US
Sent Packet to Kafka ....3 Idiots,2009,8.4,385782,170,comedy,IN
Sent Packet to Kafka ....Black Friday,2004,8.4,20611,143,crime,IN
Sent Packet to Kafka ....Super Deluxe,2019,8.4,13680,176,thriller,IN
Sent Packet to Kafka ....Winter on Fire: Ukraine's Fight for Freedom,2015,8.3,17710,98,documentary,UA
Sent Packet to Kafka ....Once Upon a Time in America,1984,8.3,342335,229,drama,US
Sent Packet to Kafka ....Taxi Driver,1976,8.3,785222,113,crime,US
Sent Packet to Kafka ....Like Stars on Earth,2007,8.3,188234,165,drama,IN
Sent Packet to Kafka ....Bo Burnham: What.,2013,8.3,11488,60,comedy,US
Sent Packet to Kafka ....Full Metal Jacket,1987,8.3,723386,116,drama,GB
Sent Packet to Kafka ....Warrior,2011,8.2,483276,140,drama,US
Sent Packet to Kafka ....Drishyam,2015,8.2,79075,163,thriller,IN
Sent Packet to Kafka ....Queen,2014,8.2,64885,146,drama,IN
Sent Packet to Kafka ....Pam Singh Tomar,2012,8.2,35888,135,drama,IN
Sent Packet to Kafka ....Conspiracy: The Sustainability Secret,2014,8.2,24845,90,documentary,US
Sent Packet to Kafka ....Virunga,2014,8.2,11403,90,war,CD
Sent Packet to Kafka ....PK,2014,8.2,178012,153,comedy,IN
Sent Packet to Kafka ....Bāhubali 2: The Conclusion,2017,8.2,91560,168,fantasy,IN
Sent Packet to Kafka ....Monty Python and the Holy Grail,1975,8.2,530877,91,comedy,GB
Sent Packet to Kafka ....Article 15-2019,8.2,32336,138,crime,IN
```

2. A custom Java application was developed to leverage Apache Spark's streaming functionality, which pulls data from the Kafka topic. The processed data is then stored in an HBase table for efficient access and scalability.

```
24/09/29 21:32:20 INFO executor.Executor: Running task 0.0 in stage 2.0 (1 of 2)
24/09/29 21:32:20 INFO kafka.KafkaRDD: Computing topic movie, partition 0 offsets 3387 -> 3389
24/09/29 21:32:20 INFO utils.VerifiableProperties: Verifying properties
24/09/29 21:32:20 INFO utils.VerifiableProperties: Property group.id is overridden to movie-consumer-group
24/09/29 21:32:20 WARN utils.VerifiableProperties: Property key.deserializer is not valid
24/09/29 21:32:20 WARN utils.VerifiableProperties: Property value.deserializer is not valid
24/09/29 21:32:20 INFO utils.VerifiableProperties: Property zookeeper.connect is overridden to
Parsing movie record...
Inserting Movie into HBase...
Movie{title='Blade Runner 2049', releaseYear=2017, score=8.0, numberOfVotes=539864, duration=164, mainGenre='scifi', mainProduction='CA'}
Inserting movie data... Data Insertion complete.
Parsing movie record...
Inserting Movie into HBase...
Movie{title='The Imitation Game', releaseYear=2014, score=8.0, numberOfVotes=748654, duration=113, mainGenre='thriller', mainProduction='US'}
Inserting movie data... Data Insertion complete.
```

3. An HBase table was created to facilitate efficient data storage, access, and management using HBase's powerful tools and capabilities.

```
Home x My Computer x cloudera-quickstart-via-S... x
Applications Places System cloudera quickstart-via-S... x
cloudera@quickstart:~$ hbase shell
[cloudera@quickstart ~]$ hbase shell
2024-09-29 21:34:36.122 INFO (main) Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase shell; enter "help<RETURN>" for list of supported commands.
Type "exit<RETURN>" to leave the HBase shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct 4 11:16:18 PDT 2017

hbase(main):001:0> desc 'movies'
Table movies is ENABLED
movies
COLUMN FAMILIES DESCRIPTION
(NAME => 'info', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0')
(NAME => 'stats', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0')
2 row(s) in 0.3780 seconds

hbase(main):002:0> scan 'movies', (LIMIT => 10)
ROW COLUMN+CELL
14 Peaks: Nothing Is Impossible column:info:duration, timestamp=1727678795088, value=101
14 Peaks: Nothing Is Impossible column:info:main_genre, timestamp=1727678795088, valuedocumentary
14 Peaks: Nothing Is Impossible column:info:main_production, timestamp=1727678795088, valueUS
14 Peaks: Nothing Is Impossible column:info:release_year, timestamp=1727678795088, value=2021
14 Peaks: Nothing Is Impossible column:stats:number_of_votes, timestamp=1727678795088, value=22858
14 Peaks: Nothing Is Impossible column:stats:score, timestamp=1727678795088, value=7.8
42 column:info:duration, timestamp=1727678965053, value=128
42 column:info:main_genre, timestamp=1727678965053, valuedrama
42 column:info:main_production, timestamp=1727678965053, valueUS
42 column:info:release_year, timestamp=1727678965053, value=2013
42 column:stats:number_of_votes, timestamp=1727678965053, value=93314
42 column:stats:score, timestamp=1727678965053, value=7.5
42 column:info:duration, timestamp=1727678945018, value=108
42 column:info:main_genre, timestamp=1727678945018, valuefantasy
42 column:info:main_production, timestamp=1727678945018, valueES
42 column:info:release_year, timestamp=1727678945018, value=2016
42 column:stats:number_of_votes, timestamp=1727678945018, value=86614
42 column:stats:score, timestamp=1727678945018, value=7.5
A Monster Calls
A Monster Calls
A Monster Calls
A Monster Calls
A Monster Calls
A Monster Calls
```

Home - Cluster / movies

row_key, row_prefix*+scan_len [col1, family:col2, fam3:, col_prefix* +3, fam: col2 to col3] [Filter1()] AND Filter2()] info: stats:

14 Peaks: Nothing Is Impossible					
info: main_genre	stats: score	info: main_production	info: duration	info: release_year	stats: number_of_votes
documentary	7.8	US	101	2021	22858
42					
info: main_genre	stats: score	info: main_production	info: duration	info: release_year	stats: number_of_votes
drama	7.5	US	128	2013	93314
83					
info: main_genre	stats: score	info: main_production	info: duration	info: release_year	stats: number_of_votes
drama	7.4	IN	163	2021	23781
A Monster Calls					
info: main_genre	stats: score	info: main_production	info: duration	info: release_year	stats: number_of_votes
fantasy	7.5	ES	108	2016	86614

Visualization:

We developed a custom Python application as our visualization tool, hosted on the local machine. To interact with the HBase tables, we utilized the HappyBase library. This application allows users to select different parameters, generating analytical insights and graphical visualizations based on the chosen dataset.

Creating Dataframe based on Data consumed From Hbase table "MOVIES"

```
import happybase
import pandas as pd

# HBase server details
hbase_host = '192.168.117.152'
hbase_table = 'movies'

# Connect to HBase
connection = happybase.Connection(hbase_host)

# Open the table
table = connection.table(hbase_table)

# Scan HBase table and fetch data
data = []
for key, value in table.scan(limit=100): # Limit to 100 rows
    # Decode row key (could be an ID or date)
    row_key = key.decode('utf-8')

    # Decode column names and values
    row_data = {col.decode('utf-8'): value[col].decode('utf-8') for col in value}

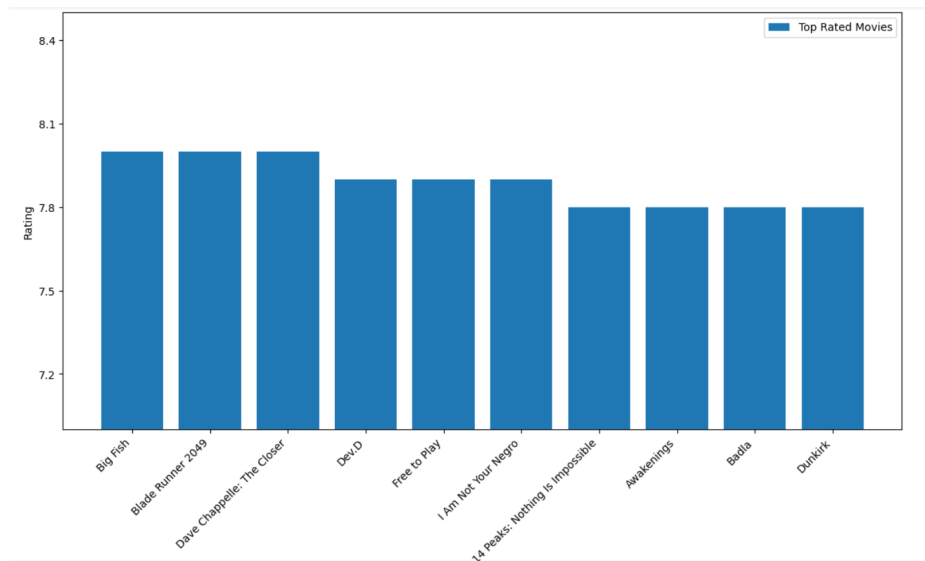
    # Include the row key (e.g., ID) in the row data
    row_data['row_key'] = row_key

    data.append(row_data)

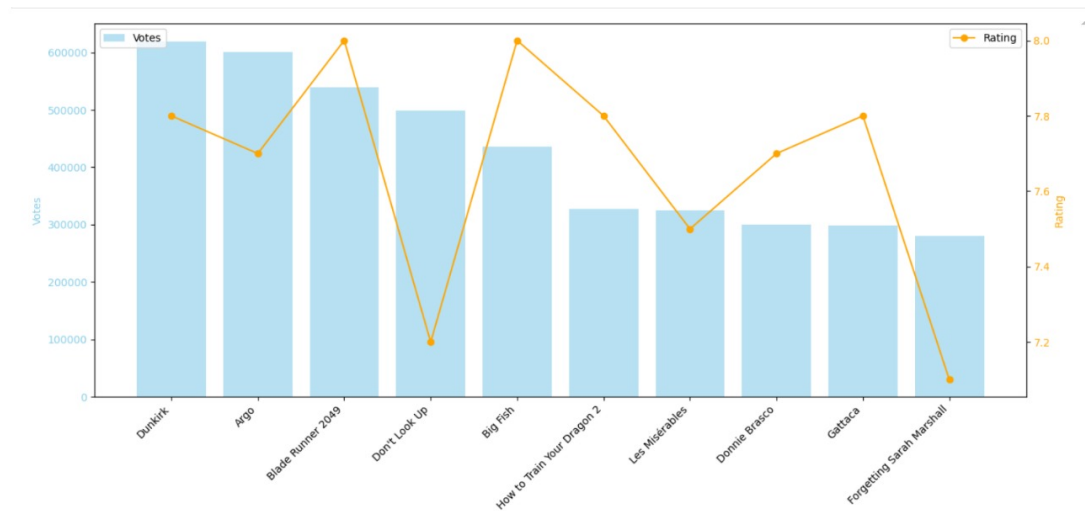
# Create DataFrame from HBase data
df = pd.DataFrame(data)
df['stats:score'] = pd.to_numeric(df['stats:score'], errors='coerce')
df['stats:number_of_votes'] = pd.to_numeric(df['stats:number_of_votes'], errors='coerce')
# Close the HBase connection
connection.close()
```

User Analytics:

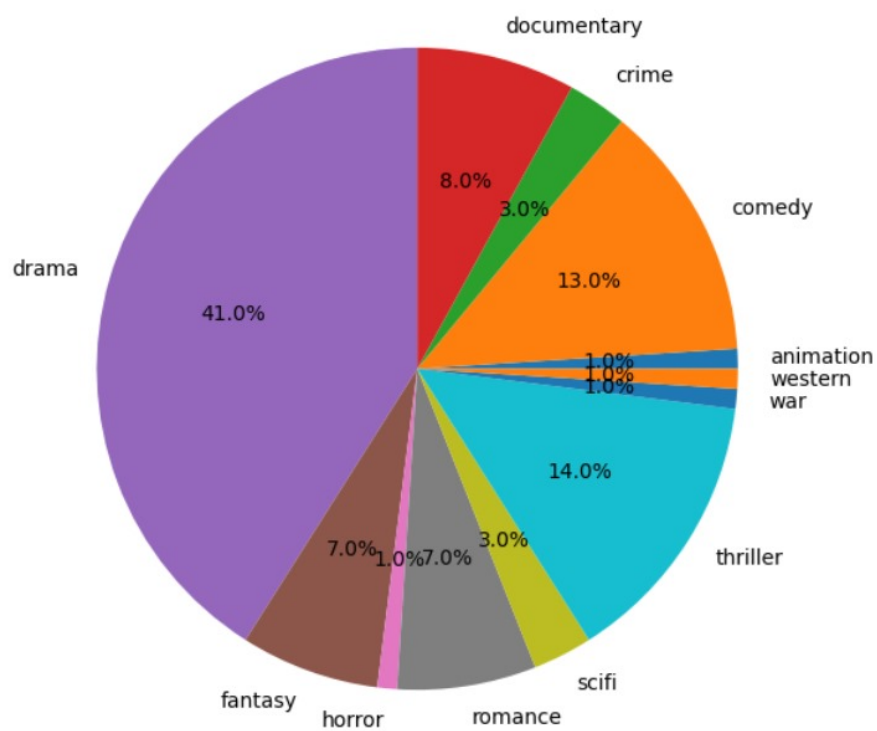
1. List of Top-Rated Movies



2. List of Movies rated by voters



3. Distribution of Movies by Genre



[Project GitHub Link for BDT-FINAL-PROJECT \(click to browse\)](#)