Big Data Technologies - CSP 554

Assignment 8 - Kafka and Big Data Patterns

Prabhu Avula | A20522815
Illinois Institute of Technology, Chicago

Exercise 1

1. The ETL process at Twitter introduced latency, with nightly jobs causing data to be a day old. Increasing the frequency of ETL to hourly stressed the already fragile ETL pipelines, often pushing them past their breaking points.
2. The lambda architecture was appropriate for counting tweet impressions in real-time and providing historical counts. This setup used a combination of batch processing and real-time processing layers to aggregate and process tweet impression data captured by frontend logs.
3. Twitter encountered significant challenges in managing the lambda architecture, primarily due to handling separate batch and real-time processing layers. Merging results from both layers added to the overall system complexity, highlighting the intricate nature of real-time data processing.
4. The Kappa architecture, proposed by Jay Kreps, offers an elegant solution to the complexities of data processing. Treating everything as a stream simplifies the process, with batch processing viewed as streaming through historical data, thereby eliminating the need for separate batch and real-time processing layers.
5. One distinguishing feature of Apache Beam is its rich API, which explicitly recognizes the difference between event time (when an event occurred) and processing time (when the event is observed in the system). This distinction is crucial for managing complexities in streaming data processing.

Exercise 2

The study "Real-time Car Tracking System Based on Surveillance Videos" by Seungwon Jung, Yongsung Kim, and Eenjun Hwang presents a novel approach to car tracking utilizing diverse video surveillance devices like CCTV, drones, and dashboard cameras. As the popularity of these devices grows, so does the need for effective car tracking systems that can handle the volume and diversity of video data produced.

The introduction highlights the proliferation of various video surveillance devices and their role in enhancing security and surveillance. It outlines the challenge of car tracking, particularly the limitations of separately analyzing frames from single video sources. The authors propose a system integrating video data from multiple sources to address these challenges and provide a comprehensive tracking solution.

The proposed system is built on a distributed processing framework, which ensures scalability and fault tolerance. It consists of three main components:
1. **Frame Distributor**: This component distributess video frames from various devices to the processing nodes.
2. **Feature Extractor**: This extracts key vehicle features such as plate number, location, and time from each frame.
3. **Information Manager**: This stores the extracted features in a database and handles user requests by retrieving relevant information from the database.

The methods section details the system's implementation. The Frame Distributor ensures the efficient distribution of video frames across processing nodes, while the Feature Extractor employs computer vision techniques to identify and extract critical vehicle features. The Information Manager organizes these features in a database and facilitates real-time queries and tracking.

The authors implemented a prototype system and conducted several experiments to evaluate its performance. The results demonstrated the system's effectiveness in accurately tracking vehicles in real-time across multiple video sources. The experiments also showcased the system's ability to handle high volumes of data and maintain accurate tracking under various conditions.
The study concludes that the proposed real-time car tracking system offers a robust solution for integrating and analyzing video data from diverse surveillance devices. This integration enhances the accuracy and reliability of vehicle tracking, making it a valuable tool for security and surveillance applications.