Name: Deep Pawar(A20545137)
Professor: Joseph Rosen
Institute: Illinois Institute of Technology

CSP 554: Big Data Technologies

Fall 2024 - Assignment 8

- **Questions and Answers:**

Exercise 1: Read the article "The Lambda and the Kappa" found on our Canvas site in the "Articles" section and answer the following questions using between 1-3 sentences each. Note this, article provides a real-world and critical view of the lambda pattern and some related big data processing patterns:

1. (1 point) Extract-transform-load (ETL) is the process of taking transactional business data (think of data collected about the purchases you make at a grocery store) and converting that data into a format more appropriate for reporting or analytic exploration. What problems was encountering with the ETL process at Twitter (and more generally) that impacted data analytics?

Ans:

The ETL process at Twitter ran into several significant problems between 2010 and 2012 that affected data analytics. It was challenging to handle the volume and complexity of Twitter's data since the ETL pipelines were hard to design and maintain. ETL pipelines also added latency because business intelligence frequently depended on day-old data and nightly tasks were common. Organizations like Twitter needed more recent data to make fast choices as business accelerated, but increasing the frequency of ETL to lower latency simply put more strain on these already delicate systems, frequently driving them over the edge.

To tackle these problems, Twitter started looking into real-time analytics options. Adopted around 2012 or thereabouts, the lambda architecture sought to deliver historical and real-time data by combining batch and real-time processing levels. But even though the lambda architecture added additional features, it also increased complexity, necessitating distinct implementations for the batch and real-time layers. The wider difficulty of striking a balance between real-time insights and the requirements for large-scale data aggregation was mirrored in the added complexity.

2. (1 point) What example is mentioned about Twitter of a case where the lambda architecture would be appropriate?

Ans:

An example from Twitter where the lambda architecture was highly suitable involved counting tweet impressions in real-time. Twitter had to keep precise historical estimates of those impressions while simultaneously providing real-time updates on tweet interactions when users interacted by touching, swiping, and clicking. This was solved by the lambda design, which used a real-time processing layer with Storm for immediate aggregations and a batch processing layer with MapReduce for handling historical data. After the logs were processed in a Hadoop data warehouse, the batch layer produced the finalized truth, while the real-time layer provided temporary data for real-time insight. With this strategy, Twitter was able to handle massive amounts of historical data while satisfying the need for low-latency analytics.

3. (2 points) What did Twitter find were the two of the limitations of using the lambda architecture?

Ans:

Twitter found two key limitations of the lambda architecture while working with it:

- **Increased Complexity:**

The lambda architecture required both the batch processing layer and the real-time processing layer to write the same logic twice. Due to its complexity, two distinct implementations must be maintained continuously, which can be resource-intensive and prone to error. For example, updates to machine-learned models need to be applied in both layers, which can lead to inconsistencies if not done correctly

- **Unclear computation semantics:**

The unpredictability of aggregate results resulting from the distinct processing of batch and real-time data was another problem Twitter had with unclear computation semantics. For example, aggregate values may fluctuate unexpectedly due to temporary problems such as log data drops in the real-time layer. The batch layer must handle this irregularity carefully, which makes the architecture even more complex.

4. (1 point) What is the Kappa architecture?

Ans:

As an alternative to the Lambda design, the Kappa architecture was proposed for data processing. The Kappa model eliminates the requirement for a distinct batch processing layer by treating all data as a stream. Both real-time and historical data are handled by this architecture's stream processing engine, which streams through old data to reprocess it. By avoiding the hassle of maintaining two distinct codebases (batch and real-time), as required by the Lambda design, the fundamental idea is to simplify the system.

Tools like Kafka Streams and Samza are frequently used in the Kappa architecture to process data streams. In this architecture, a stream is viewed as a log of updates, and a table is regarded as a cache of the most recent values for keys in the stream. By employing a single processing framework, this log/table duality enables efficient processing and makes system development, testing, and debugging easier.

5. (1 point) Apache Beam is one framework that implements kappa architecture. What is one of the distinguishing features of Apache Beam?

Ans:

One distinguishing feature of Apache Beam, which implements the Kappa architecture, is its ability to use the same processing model to handle both bounded (batch) and unbounded (streaming) data. This is especially accomplished via its Dataflow paradigm, which combines the processing of historical (bounded) and real-time (unbounded) data. Batch processing is treated as simple streaming over bounded data, which removes the requirement for distinct batch and real-time processing pipelines. The framework also clearly separates event time from processing time and offers advanced features like watermarks and triggers to handle late data which makes it an excellent option for real-time analytics. For example, an event occurring at 5:17 (event time) isn’t observed until 5:20 (processing time) due to delays in the logging pipeline.