**Week 9 Assignment**

**Real-Time Data**

Monroe College

CS 675: Big Data Management and Analytics

**Group 1**

Gita Gurung

Madeline (Zhiyang) Lin

Priyanka Chaudhari

Sukriya Gurung

Professor Syed Farid

November 6th, 2022

*What you need to do for this assignment is to implement real time data integration. Since you are working with your apache server and Hadoop framework you must try to implement this section into your services.*

*If you are successful integrating this section create a document detaining your implementation on this quest.*

*I know it is a lot of work but this is the only way to practice. This assignment is about 3 weeks long while you finish the other ones and start this one as well.*

*Remember that you can start documenting every aspect for the assignments and creating your final project.*

**ANS:**

**Real-Time** data is a boon. As we know that decision-making is an integral part of any business, data has become a crucial part of this process. From planning and forecasting to strategizing and testing, it’s the data that the data analyst needs. For modern data-driven businesses, meaningful data helps create intuitive decisions. When we talk about data analytics, a data analyst BI (Business Intelligence) supports data-driven business decision-making. This helps in keeping data operational.

**Real Time Data Management:** The term "**real-time data management**" refers to the processing of data in a short amount of time to produce nearly instantaneous results. Because the processing takes place as the data is entered, a continuous stream of input data is necessary to produce a continuous output. A batch data processing system, on the other hand, takes in data and processes it all at once at a later time, indicating that output is also produced later.

Although most data is warehoused before analysis, there is an increasing need for real-time processing of these enormous volumes, such as the 200 million emails, 300,000 tweets and 100 hours of Youtube videos that are passing by every minute of the day. Real-time processing reduces storage requirements while providing more responsive, accurate and profitable responses.

Theplatforms that can handle real time streaming big data:

1. Apache Storm
2. Apache Kafka
3. IBM Infosphere Streams

But the easiest way to get started is without an investment in additional hardware, via a cloud service such as Amazon Web Services, Google Compute Cloud, Microsoft Azure. Cloud services are set up to make using a stack such as Hadoop easier. At the base of the Hadoop stack is HDFS, the Hadoop Distributed File System. HDFS provides a useful starting point for a data lake, a single repository for large quantities and varieties of data.

**Documentation of Reading Materials**

**Lecture Notes: Designing Real-Time Streaming Data Pipelines**

This chapter covers the topics including 1) real-time streaming concepts, 2) real-time streaming components, 3) Apache Flink versus Spark, 4) Apache Spark versus Storm.

In terms of real-time streaming concepts, this chapter firstly talks about the differences between batch processing and real-time data processing. In batch processing, data is collected in batches and each batch is sent for processing. In the Spark framework, data is processed in micro batches. In terms of the real-time processing, the data gets processed immediately in analytics which helps the company gets full insights about its business in real time.

Complex event processing (CEP) is event processing that combines data from multiple sources to discover complex relationships or patterns. The goal of CEP is to identify meaningful events (such as opportunities or threats) and respond to them as quickly as possible. CEP is used in use cases, such as stock trading, fraud detection, medical claim processing and so forth.

Then the article talks about the continuous availability, low latency, scalable processing frameworks, horizontal scalability and storage.

After that, the article talks about real-time streaming components (i.e. message queue, Kafka). Kafka is a fast, scalable, durable, and fault-tolerant publish-subscribe messaging system. It provides a distributed storage layer, which supports massively scalable pub/sub message queues.

When it comes to the features of Kafka, it is scalable, durable, reliable, and it supports high performance throughput. Users can run Kafka Connect in two ways: standalone mode or distributed mode.

Apache Storm is a free and open source distributed real-time stream processing framework. Apache Storm is low-latency, supports multiple programming languages, and with no complexity.

Flink is an open-source framework for distributed stream processing. Flink provides accurate results and supports out-of-order or late-arriving datasets. It is stateful and fault-tolerant and can seamlessly recover from failures while maintaining an exactly-once application state.

The comparison of Apache Flink versus Spark is: The main focus of Spark streaming is stream-batching operation, called micro-batching. This programming model suits many use cases, but not all use cases require real-time stream processing with sub-second latency. Hence, the micro-batching programming model is not suited there. Apache Flink supports millisecond latency and is suted for use cases such as fraud detection and like.

The comparison of Apache Spark versus Storm is: Spark uses micro-batches to process events while Storm processes events one by one. It means that Spark has a latency of seconds while Storm provides a millisecond of latency. Almost the same code (API) can be used for Spark streaming and Spark batch jobs. That helps to reuse most of the code base for both programming models. Also, Spark supports Machine learning and the Graph API. So, again, the same codebase can be used for those use cases as well.

**Chapter 5 Data Acquisition**

This chapter covers data acquisition considerations, publish-subscribe messaging frameworks, big data collection systems, messaging queues and custom connectors.

In terms of data acquisition considerations, we need to consider about the source types. Some examples of batch data sources are files, logs, and relational databases. Some examples of real-time data sources are machines generating sensor data, Internet of Things (loT) systems sending real-time data, social media feeds and stock market feeds.

Publish-Subscribe is a communication model that comprises publishers, brokers and consumers. Publishers are the sources of data. Publishers send data to topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker. There are two publish-subscribe messaging frameworks – Apache Kafka and Amazon Kinesis.

Bid data collection systems refer to data collection system which allows collecting, aggregating and moving data from various sources into a centralized data store. Apache Flume, Apache Sqoop are two systems.

Messaging queues are useful for push-pull messaging where the producers push data to the queues, and the consumers pull the data from the queues. Messaging queues allow decoupling of producers of data from the consumers. Some message queuing systems based on protocols are Advanced Message Queuing Protocol (AMQP) and ZeroMQ Message Transfer Protocol (ZMTP).

Custom connectors and web services for acquiring data from data producers can be developed to meet the application requirements. Custom connectors are REST-based Connectors, WebSocket-based Connectors, MQTT-based Connectors, Amazon loT, etc.

**Chapter 8 Real-time Analysis**

This chapter covers real-time analysis frameworks, Apache Storm and Apache Spark. It describes the Apache Storm and Spark Streaming frameworks for implementing real-time data analytics applications.

Apache Storm is a framework for distributed and fault-tolerant real-time computation. Storm can be used for real-time processing of streams of data. Storm can ingest data from a variety of sources such as publish-subscribe messaging frameworks, messaging queues and other custom connectors. It is a scalable and distributed framework, and offers reliable processing of messages. Storm has been designed to run indefinitely and process streams of data in real-time.

Apache Spark enables scalable, high throughput and fault-tolerant stream processing. Spark Streaming provides a high-level abstraction called DStream. Spark can ingest data from various types of data sources and the data ingested is converted into DStreams. Spark supports DStream transformations which are stateless, and also stateful operations such as windowed operations.

Finally, it uses some case studies of real-time social media, weather and sensor data analysis using Storm and Spark.

**Chapter 9 Security**

It is essential that data is stored, processed, and accessed securely. The aspects of security are usually broken into four domains: authentication, authorization, auditing, and confidentiality.

The TLS protocol establishes the backbone of secure web communications by providing a framework for encrypting data transfers. Despite its official deprecation, this in-flight encryption is still often referred to as Secure Sockets Layer (SSL) communication. Before a secure data communication is initiated, the TLS protocol allows clients to guarantee the identity of a remote server and to agree on a mutual symmetric encryption key through the use of public key cryptography. With TLS, the client can ensure it is talking to the right server by ensuring that the server’s public certificate has been signed by a mutually trusted certificate authority.

Certificates are most often structured according to a common format called X.509. The format defines many standard fields, but for our purposes, the most important are: subject name, issuer name, validity and subject alternative name.

SASL is an abstraction API to allow networked protocols to use a generic security API to transparently handle multiple authentication and data protection mechanisms.

The various storage, processing, metadata, and management services running on a cluster use a variety of authentication mechanisms. The most widely used is Kerberos. Kerberos is a network authentication protocol designed to address the challenge of strong and secure mutual authentication in distributed systems, and to do so in an efficient and scalable way.

Overall, the article gives a summary of the various mechanisms employed by projects that may be running on the cluster. It is essential that operators, architects, and developers have a solid grounding in what security mechanisms are available in each component of the cluster.

**Chapter 10 Integration with Identity Management Providers**

This chapter examines the interactions and outline some common integration architectures. We need identity management providers in the following areas including Kerberos, User accounts and groups and Certificate management. We need to set up integration at the application and at the OS level.

In terms of the integration scenarios, it includes Writing a file to HDFS, submitting a Hive Query and Running a Spark Job. Each integration area has a number of possible providers, and the job of the cluster architect is to choose the providers that fit best within the given enterprise context.

This chapter covers the various ways of integrating enterprise systems into the security controls of the cluster. We need to integrate identity management at several layers of the stack at the OS and application levels, because it provides authentication, authorization through groups, and certificate management for inflight encryption.

**Chapter 11 Accessing and Interacting with Clusters**

This chapter explores typical architectures for providing users access to cluster services and data while applying the authentication and authorization controls we encountered. First, we look at the different ways in which a user might interact with the cluster, and then we explore how we can enable these through our cluster architecture and supporting technologies. After we have established the architecture, we take a look at user workbenches, such as Hue and Cloudera Data Science Workbench (CDSW). Finally, we look at the options for transferring files into and out of the cluster.

Understanding the options for granting users and applications access to the cluster is a critical piece in putting together a solid cluster architecture that will satisfy those users’ demands and requirements. In this chapter, we covered the many ways in which a user or application may interact with the cluster, and we outlined the choices you have as an architect for technologies that support each type of interaction.