Abstract - This Paper talks about using Apache NIFI, Kafka and Spark for next generation of ETL based on their specific key features and architectural frame works. By considering benefits of these 3 technologies, we can create ETL data pipeline in near real time. To achieve this, use NIFI for Data Extract and Load. Use Spark for Data Transformation. To integrate these two use Kafka as the integrating tool.

Keywords - NIFI, Kafka, Spark, Spark Structured Streaming, ETL.

1. INTRODUCTION

Traditional ETL process supports structured data and batch processing, hence stream process and semi structured & unstructured data is not much supported in practical. Initially Big data supports batch processing. In recent years, big data has moved from batch processing to stream-based processing. Currently many stream processing frameworks available in the market [1]. Every stream processing framework supports high level relational operations. Hence streaming tools can be used to achieve ETL data pipe line for next generation in IT industry. Lets discuss the key specificications and architectural frameworks of NIFI, Kafka and Spark structured streaming with respect to ETL process.

2. NIFI FOR DATA EXTRACTING AND LOADING

NIFI was built to automate and manage flow of data between two systems. Initially it was known as Niagra files(project) aimed to address the issues of data flow in the context of security, interactivity, fault tolerance, scalability, routing, data transformation, conversion, encryption, governance, provenance, buffering with back-pressure, prioritized queuing, clustering, and many more [2]. Once NSA introduced NIFI, further development, support continued by Onyara company as open source project. later Horton works acquired Onyara and made it Horton works data flow platform.

Key Features of NIFI

Guaranteed Delivery

At High scale, NiFi provides guaranteed delivery, which is a core philosophy.

Data Buffering

NiFi supports back pressure and pressure release.

Prioritized Queuing

Determines what is important for flow data and Provides Queue prioritization as FIFO, Newest flow file first, oldest flow file first etc.

Flow Specific QoS

Latency vs Throughput, Loss Tolerance.

NiFi enables fine-grained flow specific configuration of these concerns. We can choose low latency and higher throughput on each processor in scheduling tab [3].

Data Provenance

NiFi records, indexes the provenance of data as objects flow for entire system states. View attributes, content at given point of time.

Supports Push and Pull Models

Recovery / Recording a Rolling Buffer of Fine-Grained History

Visual Command and Control

Drag and drop processors to build a flow

Start, stop, terminate and configure in real time

View error, error messages, statistics and health of data flow
Flow Templates

Create templates for common processors and connections.

Pluggable/Multi-Role Security

Designed for Extension

Clustering

NIFI Processing Framework

Simple Event Processing Tool

Feed data to processing framework, process data is mainly focus on simple event processing. Operate on single piece of data. i.e., enrich, parse, split and transform on that data [4].

Provide Data Flow Solution

It provides good data flow with centralized management, from edge to edge. Provides great traceability, data provenance at event level very clearly. Provides visual representation of global data flow [5].

Data Flow Management Tool

Supports out of the box solution for data flow management.

Has flexible failure handling at each point of the flow.

It is not Schema Dependent

Supports dataflow management for both structured and unstructured data, powered by separation of metadata and payload. Schema is not required, can use as optional if you want to use schema. It requires minimum meddling effort and all about is to manage the data flow [6].

Considering all above key features and processing framework NiFi is much suitable for data Extracting and Loading phases in ETL process.

Drawback

As NiFi can't support replica of Data, when a node is failed, it routes to another node but queued data is available when the failed node is up. It means fault tolerance is maintained at flow level and data level not possible. Can't support complex data processing and distributed processing [7].

3. USE KAFKA TO INTEGRATE NIFI WITH SPARK

Kafka is a distributed publish-subscribe messaging system, handling high volume of data from one-end point to another. It is suitable for both online and offline message consumption. It maintains the replica of messages in the cluster.

3.1 Some of the Key Features of Kafka

Scalability

Maintains scalability in all four dimensions, event procedures, event processors, consumers and event connectors. Other words it scales easily without downtime.

High-Volume

Works with huge volume of data easily.

Data Transformations

Derives new data streams from producer data streams.

Fault Tolerance

Kafka clusters handles failures with masters and databases.

Reliability

With the features of distributed, partitioned, replicated, and fault tolerant, it is very reliable.

Durability

As it maintains distributed commit logs, i.e, messages persist on disk as fast as possible.

Performance

Kafka provides high throughput on publishing and subscribing messages as it stores TB’s of data easily.

Zero Downtime

It is very fast, which guarantees zero downtime and zero data loss.

Extensibility

Provides plug-in facility as well as offers ways to write new connectors as needed.

Replication

Ingest pipelines causes replicate events.
3.2 Kafka Processing Framework

It is a producer-consumer model. Kafka cluster acts as broker between producer and consumer. Streams of records organized into categories called Topics. Topics can be partitioned/replicated. Records consists of Key, value and timestamp. Partitions are used for published messages and are replicated for fault tolerance [10].

A producer publishes messages to a topic. Producer decides which message to pass which partition. Partition consists ordered, immutable sequence of messages. New messages are added at end of the partition. Partitions are replicated for fault tolerance. A replicated partition has one broker, which acts as leader and remaining replicated partitions have followers. Sender uses offset to fetch messages from a partition.

3.3 Consumer Consumes Messages Using Consumer Groups

Each Consumer has its own specific offset to fetch messages from a partition. Messages are consumed by consumer group. Each consumer is labelled with a corresponding consumer group. Each message in a topic routes to one consumer in a group.

3.4 NIFI's KAFKA Integration

Here NIFI is used as Producer and Kafka is used as consumer. Due to NIFI's isolated class-loading capability, NIFI supports multiple versions of the Kafka client in a single NIFI instance. To avoid Data fault tolerance, we integrate NIFI with Kafka. Which means to achieve zero downtime in NIFI.

4. SPARK FOR DATA TRANSFORMATION

Spark is a general purpose distributed data processing engine that is suitable for use in a wide range of circumstances. On top of Spark core data processing engine, has libraries for SQL, Stream processing, machine learning and graph computations. Previously discrete stream processing was used. Spark structure streaming built on top of spark SQL library. This model of streaming is based on Data frame and Dataset API instead using RDD [8].

Key Features of Spark Structured Stream Processing

Consistency

As it uses unbounded tables instead RDD, output tables are always consisting with all records in a prefix data (sent timestamp).

Fault Tolerance

Maintained historically, including in interactions with output sinks.

Out-of-Order-Data

Out of order is clear as it maintains sent timestamp as prefix to data record.

4.1 Sprak Structured Processing Framework

Event occurs timely to produce data, which streams from messages queue to receiver of the streaming platform. During this process we can face the time skew. To overcome this spark structured streaming is using time window with watermarking. Watermarking is useful method which helps to deal with lateness of data in streaming process. Watermark is a threshold to specify how long the system waits for late events. Spark structured streaming turns into SQL, which is fast, scalable and fault-tolerant. It unifies with high level APIs with spark to deal with complex data and complex workload. This architecture uses unbounded table for each slide of the time window to receive data stream. Each data stream has timestamp to identify to which unbounded window it belongs. Hence even the input stream comes late also route to corresponding time unbounded table. All the slide window tables are aggregated and forms single output table [9].

4.1.1 Operator

Spark RDD performs two operations. Transformation and Action. Transformation are trigger to creates new RDD, Data frame or Dataset. Actions are performed to work with actual dataset. With structured streaming it takes extracts data in memory and does data cleansing, aggregation for analysis purpose.
4.1.2 Challenges

Be very cautious of forcing order in Kafka. Pick
the data store which matches your query pattern.
Deployment of NIFI from Dev, UAT and PROD.
Version upgrade of the NIFI to be careful.

We need to handle logic to avoid duplicate
processing while writing the output. We should focus
on lazy sources and sinks of spark structured
streaming.

5. CONCLUSION

ESS(Event Standardization Services) captures
customer activity across all channels, such as Online
Banking, Mobile Apps, Bank Branch, Advice centre
etc. Goal is to move traditional ESS(batch processing)
to Modern Event Stream Processing.

REFERENCES

[1] Darryl Sutton, T4G, Principal Consultant & Kenneth Poon,
RBC, Director, Data Engineering, DataWorks summit, Jun
22, 2018.

[2] Andrew Psaltis, Principal Solution Engineer, Hortonworks,
DataWorks Summit, Feb 12, 2017.


[4] Spark Structured Streaming, Matei Zaharia, Tathagata Das,
Michael Armbrust and Reynold Xin Posted in Engineering Blog
July 28, 2016

[5] Debasis Bandyopadhyay and Jaydip Sen. Internet of Things -
Applications and Challenges in Technology and Standardization,
2011.

on Internet of Things. In: International Journal of Engineering

the Internet of Things: threats and challenges, In: Security and


the Internet of Things: threats and challenges, In: Security and