International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 6, Issue 4, July- August 2017 ISSN 2278-6856

Spark Streaming-Real time stream processing in credit card fraud detection

M. Likhitha[1] , Y. Rama Mohan[2]

[1]PG student, G. Pulla Reddy Engineering College, Dept. of computer science and Engineering, Kurnool, Andhra Pradesh - 518002, INDIA

[2]Assistant Professor, G. Pulla Reddy Engineering College, Dept. of computer science and Engineering, Kurnool, Andhra Pradesh - 518002, INDIA

Abstract
In the era of globalization, electronic payments and online transactions have increased, apparently resulting in the increasing use of credit cards. At this juncture the number of fraud events happening in the name of financial services caused a great loss to financial industries. Detecting such fraud events has become imperative for the industries. With the existing conventional techniques, it's not possible to provide a better performance in the mode of ever increasing data called Big Data. Big data offers live stream processing frameworks and APIs that are widely being used in many real time applications in the existing environment of distributed datacenters. The proposed system works on processing the real-time data using Spark Streaming. The objective is to build features to process the real time data with Spark Streaming to reduce the workload on the node(s), achieve low latency to provide a better execution plan for a scalable and fault-tolerant processing of data.

Keywords: Real time data, Spark Streaming, Stream processing, credit card fraud, big data.

1. INTRODUCTION
Big Data [2] has become the buzzword in the world of technology, "Big data" is a name given to collection of different types of data sets that are collected from and stored on clusters. Big Data is same as the normal data but huge in size which is increasing tremendously every second. But it is not just about the size, storing and processing such huge volume of data is the main objective of the many organizations including Google, Amazon and FaceBook. Analyzing such large amount of data and processing it poses a great challenge for database and data analytics research. Every day people are creating 54,000 Exabytes of data on the Internet, which needs to be processed instantaneously. Conventional or existing database systems are often unable to deal with such large and complex data. Creating a void in the speed and transparency level at which the data is processed.

In batch processing [12] data is collected beforehand and processed together in batches. When a short response time is not strictly required, batch processing is widely used to process considerable volumes of data without any user intervention. Another mode of processing is Stream processing - where streams of data can be processed instantaneously as they arrive.

Apache Hadoop is a batch processing framework, with a distributed file system called HDFS that also functions as a Meta - store. MapReduce is the heart of Hadoop, a programming model that allows parallel processing of the data. Stream processing likely solves many of the challenging encounters of data processing. Apache Spark is a processing engine and largest open source in big data that has been adopted by many industries to process the petabytes of data across thousands of nodes. Unlike Hadoop, Spark also supports interactive queries and streaming over large volume of data. Being an extension of Spark, Spark Streaming [9] has all the features of Spark such as supporting integration of storage systems and third party libraries, supporting programming languages. Micro batch processing is a special case of batch processing where small size of batches gets processed on stream. The spark streaming analyzes the real time data [11] using this micro batch technique. Spark Streaming has become the best live stream processing framework to deal with the problems in business industries, especially for financial industries.

2. RELATED WORK
With the globalized world moving towards borderless economy, digital payments and online transactions, their security becomes all the more important with increasing cyber crimes. Fraud events happening in the name of financial services cause a great loss to financial industries. Finding a solution for all such fraud events with the help of efficient data processing systems is the need of the hour.

There are several approaches proposed for fraud detection, where data needs to be processed on different stacks; one for batch processing and other for stream processing [7], [10]. Traditional systems can perform only one function at one instance; either stream processing of hundreds of MBs with low latency or batch processing of Terabytes with high latency. It's extremely expensive and risky to maintain two separate stacks for two processing modes - such as separate programming model, thus increasing the implementation effort for both stacks. Existing systems uses the pipeline of nodes for data distribution, which causes a problem of latency.

As a part of my research few lacunae were identified in the wide area analytic models- migrating data and integrating it, which cause latency issues. To overcome these issues a real time analysis of data is necessary. Stream processing is one such model which efficiently does this. So, I have explored areas of Stream processing [1] for one of the use cases called Fraud Detection. In the entire study it is noticed that, the existing systems are not efficient to deal with huge amount of real time data. Due to the pressing need of processing large volumes of streaming data, the proposed system in this paper is aimed at processing massive amount of data in a stipulated time. I am presenting an approach of processing live data streams with Spark Streaming.

3. REAL TIME STREAM PROCESSING

Mapr Converged Data Platform (MCDP) [16] is an enterprise grade distribution for Hadoop, developed by MapR Technologies. It supports a dozens of open source engines and tools that provides various APIs to meet the requirements of real time processing. MapR distribution allows you to install various Hadoop components for processing and analyzing the data on your cluster. The proposed system is associated with MCDP to integrate Hadoop and streaming with real time database capabilities. It also helps the integration of various enterprise storage and event streaming.

Organizations felt a dire need of an efficient processing method to handle the ever increasing data. Though Hadoop 1.0 (MRv1) could do the job, Hadoop 2.0 (MRv2) came with a more efficient processing system of advanced features and APIs that allows working with specialized and interactive programs to provide more scalable and reliable data processing. However, a typical streaming architecture is composed of three key components namely called- a producer to collect the input data from different sources, a streaming system for creating streams from the collected input and a consumer to process the stream generated by the streaming system.

3.1. Data Ingestion

Kafka is a distributed streaming platform that provides several APIs for communication between the producer and consumers. It is a messaging based log aggregator system that allows consumers to consume messages sequentially from particular partitions as shown in Kafka architecture - Fig.1.

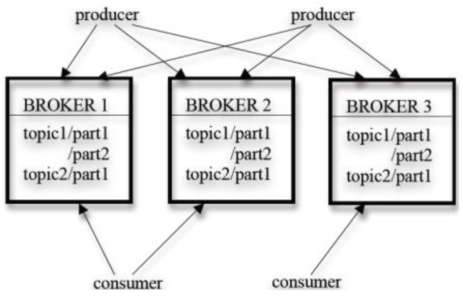


Fig.1. Architecture of Kafka

Kafka has shown its best side in building real-time streaming data pipelines that reliably get data between systems or applications that transform or react to the streams of data. It can be run as a cluster on one or more servers and it stores the streams of records in labelled categories called topics. Kafka has four core APIs [16]: Producer API, Consumer API, Streaming API, and Connector APIs, which are responsible for allowing application to publish, consume, transform streams and connecting to the database respectively. **Kafka API** [4] offers real-time processing and integration with popular stream processing frameworks like Spark Streaming. It replicates the data globally for high availability and it can be used for both online and offline distributed applications. Kafka performs multiple read and writes of topics using partitions and can easily write up to 700 Mbps.

3.2. Data Processing

Apache Hadoop has a core MapReduce engine for distributed computing of large scale data and it has recently came out with a new version called MapReduce 2.0(MRv2) with major changes and improved features to support more distributed computing models. MRv2 allows multiple processing engines to work together, and it has two components – YARN [13] for cluster resource management capabilities and MapReduce [3]. This MRv2 separates the resource management and processing components by allowing a new ResourceManager to manage resource usage across applications, where as execution of jobs is managed by the ApplicationMasters that helps in removing a bottleneck and let clusters scale up to larger configurations. Integrating Hadoop and Spark allows the Spark streaming to use the Spark APIs for both streaming and batch processing of real-time data.

Spark streaming processes datasets that are divided in several discretized streams (DStreams) [8] which are then processed continuously in streaming mini-batches. Input data can be ingested from sources like Kafka, Flume, HDFS or S3 and outcomes pushed out to accessible sources like dashboards, databases and file systems. Architecture of Spark streaming is shown in the Fig.2.

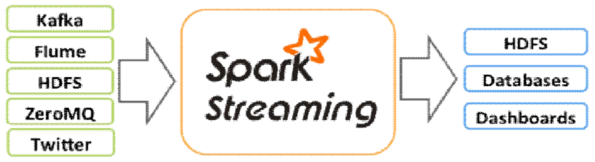


Fig.2. Architecture of Spark streaming

3.3. Data Storage and retrieval

HDFS is a specially designed distributed file system [5], [6] for storing huge data set with cluster of streaming access pattern. It supports multiple reads and writes and can avoid single storage device I\O bottlenecks. HDFS is managed through two nodes – one for holding metadata and other for storing files. Node that holds metadata is namely known as Name Node that acts as a Master node and the node that sore files is known as Data Node which acts as a slave node.

Hive [15] is a SQL inspired query oriented language and a data warehouse infrastructure tool to process structured data in Hadoop. Hive can be configured with Spark and YARN as per the requirement of the application. Hive on Spark [14] allows utilizing Apache Spark as its execution engine. Following are the steps performed by Hive to interact with Execution engine and Executes the query plan by sending it to the driver program.

4. PROPOSED SYSTEM

The proposed work focuses on distinguishing the fraudulent events in credit card usage by processing data streams that are being generated continuously. The strategy is to compare the current transaction requests with historical transactions to distinguish the fraud transactions from genuine transactions.

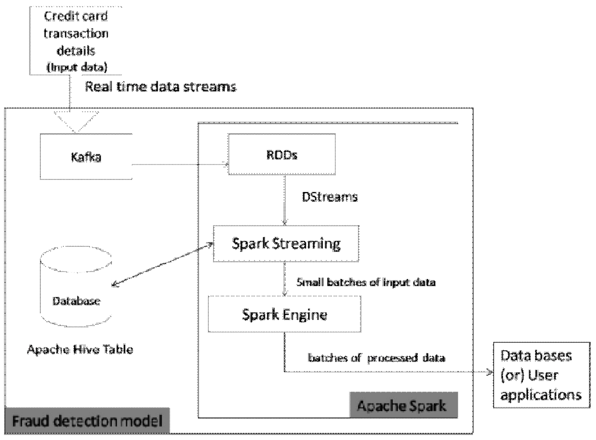

Credit card transaction details (Input data)
Real time data streams
Kafka
RDDs
DStreams
Spark Streaming
Database
Small batches of input data
Spark Engine
Apache Hive Table
batches of processed data
Data bases (or) User applications
Apache Spark
Fraud detection model

Fig.3. Block diagram of proposed system

The databases of the financial industries, serve as producers to deliver the required credit card transaction information, which acts as an input for the further stream processing that takes place through Kafka, a distributed streaming platform which segregates the data, and delivers billions of streams per second. Repositories of Kafka APIs that enabled the data ingestion of streams will now send the processed streams to the consumers. The consumers are the subscribers or processing engines that deploy the major utilization of received data. The entire working of the system is interpreted in the Fig.3.

Spark streaming uses in-memory computation and supports multiple APIs to perform operations on data. Live data streams can be grouped into small size of batches that are defined as Descretized Streams (DStreams) are treated as RDDs (Resilient Distributed Dataset) [8], [11] to compute them. Spark streaming supports real time data processing with micro batching of jobs. Resulting data can be pushed out as batches of processed RDDs. Fig.4 shows the data flow of spark streaming.

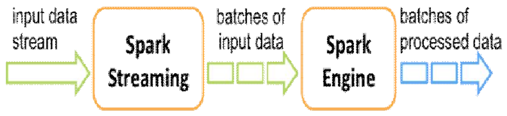


Fig.4. Data flow of Spark streaming

In general, a credit card transaction can have the features associated with the card holder, transaction and transaction history. Some common features are: Card holder name, card number, location, withdrawal amount, transaction id, average withdrawal, credit limit, total spends and some other details of card holder. A sample data set of 3,00,000 records (approximately) that are composed of the specified features, such as transactionId (*id*), cardholder name (*full_name*), gender (*gender*), card number (*card_number*), credit limit (*credit_limit*), total spends (*total spends*), email(*email*), where each row of the data set acts as an RDD as shown in fig.3. This action is followed by the creation of a database in Hive using the following schema (written in " "):

" *CREATE Transaction_data(id int, full_name string, gender string, card_number string, credit_limit double, total_spends double, email string) row format delimited fields terminated by ',' ;* "

Data can be loaded from *.csv* file using the following command:

"LOAD *DATA* *LOCAL* *INPATH '/home/mapr/card_data.csv' INTO TABLE Transaction_data;*"

The proposed system intended to analyze the sequence of transactions from the user inputs and validates the current transaction details based on the *credit limit* of corresponding *card number.* Initially stream creator (kafka) generates sequence of transactions (streams) from user input and initiates the transaction by providing details entered by the user, then it sends the transaction details for stream processing using the following query:

{"type":"transaction","timestamp":3516821.910,"Name":" Sathish","CardNo":"5602241533192786868","transactid" :"2"}

The consumer receives the transaction message from stream creator by entering into an infinite loop. It fetches the details of current transaction's *card number* from the database by submitting the following query:

SELECT full_name,card_number,credit_limit,total_ spends FROM transaction_data WHERE card_number='5602241533192786868'

Then it compares and validates the *credit limit* with current *transaction amount* based on the rules set. It rejects the transaction if the validation failed for the given *card number* and if a transaction is rejected by the analyzer, then it is labelled as the fraud transaction and the stream is deemed as a fraud event and this is not sent for the further processing phases. The following is the pseudo code used for this stream processing.

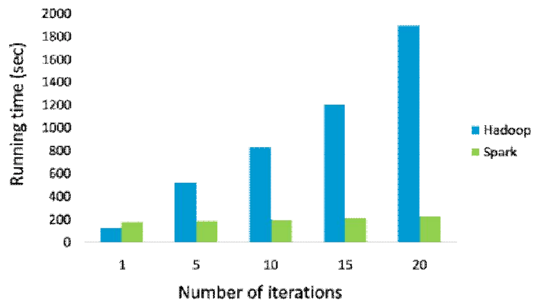
Pseudo code for Transaction analyzer:

```
1. Repeat (Infinite loop to read messages)
2. Read messages from Streams
3. Read Card Number and amount entered
4. Get credit limit based on the card number
     a. Validate credit limit
     b. Reject if not valid
     c. Push the failed transactions to other stream
     d. Send success message if legitimate
5. Read next message
```

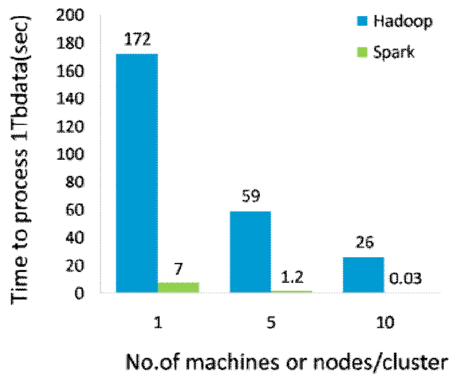
5. OBSERVATIONS

Many key big data applications must run in real time such as click analysis, spam filtering, live stream reporting, etc. Unlike Hadoop's batch processing, Spark streaming's functional APIs provides horizontal scaling of data across the nodes of cluster in a distributed manner, by which the data can get processed hundred times faster.

The statistical measures shows the benchmarks of Hadoop and Spark; i.e. the time (sec) taken to perform iteration(s). The following Fig.5 shows: (a) the comparison of regression performance of Hadoop and spark, (b) the comparison of data scaling of Hadoop and Spark.



(a) comparison of regression performance



(b) comparison of data scaling

Fig.5 comparison of Hadoop and Spark Performance

The graph plotted by using number of nodes and the iteration time as parameters, can be depicted in the following Fig. 6 that illustrates the analysis taking into account both regression and data scaling performances of Hadoop and spark, in comparison with spark streaming.

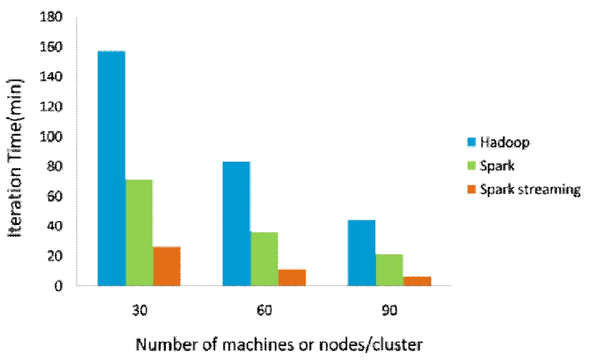


Fig.6 Hadoop and Spark in comparison with Spark streaming

In this test case it is observed that real time analysis using the MapR Converged Data Platform that integrates Hadoop and Spark with real-time database capabilities, which facilitates high performance and low latency, efficient processing methods in real time analysis.

6. CONCLUSION

The entire study and proposed system aim at processing the massive amount of data in a stipulated time, while not compromising on the efficiency and the quality of the data retrieved. The ever growing digitization needs the real time analysis of all the trade transactions and the business of the financial organizations. The entire economy of the world runs on the uninterrupted data flow and the financial goals are set based on this data. Streams processing, lies at the heart of data management and analysis, which is important for providing the accurate and reliable information.

REFERENCES

[1]. Albert Bifet, Silviu Maniu, Jianfeng Qian. StreamDM, "Advanced Data Mining in Spark Streaming". 2015 IEEE International Conference on Data Mining Workshop, Atlantic City, NJ, USA.
[2]. C. Lakshmi, V. V. Nagendra Kumar, "Survey Paper on Big Data". International Journal of Advanced Research in Computer Science and Software Engineering, MCA Department, RGMCET, Nandyal, Andhra Pradesh, India
[3]. J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters". Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.
[4]. J. Kreps, N. Narkhede, and J. Rao. "Kafka: A Distributed Messaging System for Log Processing". In NetDB Workshop, June 2011.
[5]. Jeffrey Shafer, Scott Rixner, and Alan L. Cox; "The Hadoop Distributed Filesystem: Balancing Portability and Performance". Rice University Houston, TX.
[6]. K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system". Proc. of IEEE

on Mass Storage Systems and Technologies (MSST), 2010.
[7]. Krishna Kumar Tripathi, Mahesh A. Pavaskar, “Survey on Credit Card Fraud Detection Methods”. VJTI, Mumbai University, Mumbai, Maharashtra, India. 11, November 2012.
[8]. Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, Ion Stoica; “Discretized Streams: Fault-Tolerant Streaming Computation at Scale”. University of California, Berkeley.
[9]. Patricio Córdova, “Analysis of Real Time Stream Processing Systems Considering Latency”. University of Toronto.
[10]. S. Benson Edwin Raj, A, “Analysis on Credit Card Fraud Detection Methods”. Annie Portia, Department of CSE, Karunya University, Coimbatore. International Conference on Computer, Communication and Electrical Technology – ICCCET2011, 18th & 19th March, 2011.
[11]. Saeed Shahrivari, “Beyond Batch Processing: Towards Real-Time and Streaming Big Data”. Department of Computer Engineering, Tarbiat Modares Univeristy, Tehran; 17 october 2014.
[12]. Siqi Ji and Baochun Li, “Wide Area Analytics for Geographically Distributed Datacenters”. University of Toronto, Toronto; April 2016.
[13]. Vinod Kumar, Vavilapallih Arun, C Murthy, Chris Douglasm, Sharad Agarwali, “Apache Hadoop YARN: Yet Another Resource Negotiator”. 3 Oct. 2013, Santa Clara, California, USA.
[14]. Hive on SPARK; integrating Hive with Apache SPARK: https://cwiki.apache.org/confluence/display/Hive/Hive+on+Spark%3A+Getting+Started
[15]. Hive: https://cwiki.apache.org/confluence/display/Hive/Tutorial
[16]. Kafka: http://cloudurable.com/blog/kafka-architecture/ index.html
[17]. MapR Converged data platform: https://mapr. com/

Authore

M. Likhitha received the bachelor’s. Degree in Computer Science and Engineering from Jawaharlal Nehru Technological University-Kakinada and currently pursuing Master’s in Computer Science from G.Pulla Reddy Engineering college, Kurnool, A.P. Her research interests include big data analytics.

Sri Y. Rama Mohan received bachelor’s Degree from Madras University and Master’s degree from Dr.MGR University Chennai in Computer Science & Engineering. He is currently working as an Assistant Professor in Dept.of Computer Science& Engineering in G.Pulla Reddy Engineering College, Kurnool, A.P and pursuing his Ph.D from Rayalaseema University, Kurnool. His research interests include Fiber Optic Networks