Real-Time Big Data Warehousing and Analysis Framework

Abbas Raza Ali
Cognitive and Analytics Practice Leader IBM
e-mail: abbas.raza.ali@gmail.com

Abstract —**Big Data technology is gradually becoming a dire need of large enterprises. These enterprises are producing large amount of offline and streaming data in both structured and unstructured forms on daily basis. Moreover, it is a challenging task to effectively extract useful insights from this type of complex data. On the other hand, the traditional systems are unable to efficiently manage transactional data history of more than a few months. This paper presents a framework to efficiently manage and effectively analyze massively large and complex datasets. The framework can be very effective for the communication industry which is bound by the regulators to manage significant history of their subscribers' call records, where every single action of a subscriber generates a packet containing more than 500 attributes. The analyses of transactional data allows the service providers to better understand their customers' behavior, for example, deep packet inspection requires transactional internet usage data to explain the data usage behaviour of the subscribers. On the contrary, relational database systems limit the service providers to only maintain semantic level call history which is aggregated at subscriber-level. The framework addresses the mentioned challenges by leveraging Big Data technology which optimally manages and allows deep analysis of large and complex datasets. The framework has been applied on a communication service provider producing massive amount of streaming data in binary format. It has been used to offload the existing Intelligent Network Mediation and Relational DataWarehouse of the service provider on Big Data technology. The service provider has 50+ million subscriber-base with yearly growth of 7-10%. The endto- end CDR processing takes not more than 10 minutes which involves binary to ASCII decoding of call detail records, stitching of various interrogations against a call (transformations) and aggregations of transformed call records of a subscriber.**

Keywords—Big Data, Communications Service Provider, deep Packet Inspection, Enterprise Data Warehouse, Stream Computing, Telco IN Mediation.

I. INTRODUCTION

The pace of large enterprises is aggressively accelerating which demands prompt action on subscribers' activities. Even few systems in an enterprise require real-time consumption of the processed data to smoothly run their key operations. The advancement in Big Data technologies make it possible to effectively and efficiently manage and analyze massively large and complex transactional data [4]. Mostly the Communications Service Providers (CSPs) are the early adopters of this technology because of the massive volume and complexity of their data [1]. The framework proposed in this paper addresses the above mentioned challenges. This framework enabled a CSP to offload their Intelligent Network (IN) mediation data and its Extract, Transform and Load (ETL) from traditional Telco mediation and Enterprise Data-warehouse (EDW), respectively, to Big Data platform.

The proposed framework allows the ingestion of both streaming and offline data, generated by different systems within the CSP, like Relational Database Management Systems (RDBMS), files and Telco Operations/Business Support Systems (OSS/BSS). It consists of several offline and streaming data-flows which integrate various key components of the framework. The data ingestion component, which is also known as Real-time Persistent Data Hub (RPDH), consists of various connectors and integrators. These connectors and integrators are used to fetch data from different sources, mentioned earlier, to maintain queues of data streams where they eventually land in Big Data Warehouse (BDW). The BDW is a key component of the framework which manages raw decoded data in Hadoop data-lakes. These data-lakes contain IN mediation CDRs that are first converted in JSON format and then stored in compressed Optimized Row Columnar (ORC) format. This compression reduces the size of the data from 90-95% of the actual size.

The data-lakes and ETLed data sources are considered as a central repository for various applications [2] including analytics, visualization, reporting, campaign management and marketing. Thus, Active Data Analysis Platform (ADAP) is considered as the brain of the framework which is further divided into three logical layers. These layers perform different tasks which include pre-processing of raw data, data modeling and visualization. The outcome of this module is stored back in the BDW from where these are consumed by campaign and reporting modules. These analytical insights are merged with other structured attributes to target new and existing subscribers with intelligent real-time campaigns. The highlevel architecture of the framework is shown in Figure 1 where the process flow is divided into infrastructure technology stack, functional and data modules, and custom implementation of the framework.

The framework has been applied successfully on a CSP to primarily offload Telco IN Mediation and ETL systems. Therefore, the IN mediation offloading requires real-time ingestion of raw binary CDRs and their conversion from binary streams to ASCII format. Likewise, the ETL offloading is an offline process which applies aggregations and transformations on converted ASCII CDRs. The outcomes of both real-time and offline processes have been dumped in Hadoop upon completion of all the key operations on the CDRs. It includes the raw decoded CDRs, which are stored in data-lakes, as well as Hadoop dumps of both aggregated and transformed CDRs. Finally, the subscriber-level aggregated data, which is known as Semantic Data Model (SDM), is stored in the existing EDW and its copy is maintained in Hadoop. On the other hand, there are few components of the framework which provides system administration, cluster and Hadoop tables level security, and flexible integration to connect different modules. Figure 2 shows key components of the system.

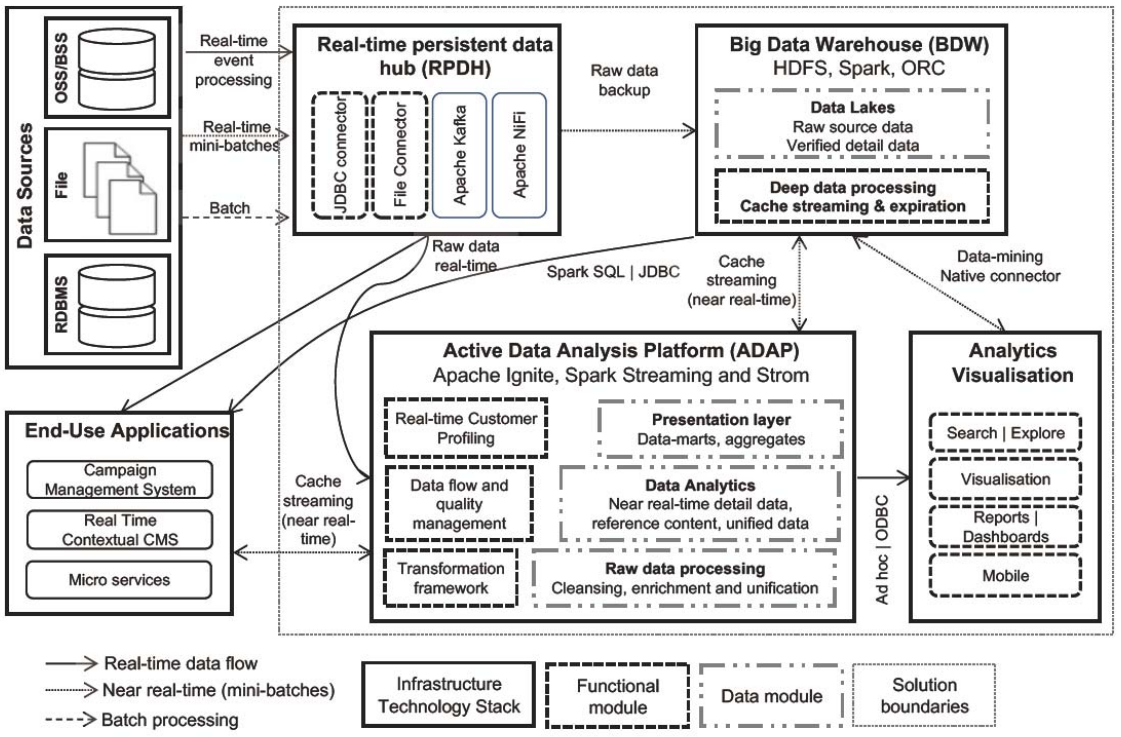

Figure. 1. High-level architecture of Real-time Big Data Framework

In the remainder of this paper, Section II discusses the existing system that is being used by the CSP. Section III is devoted to components and methodology of the overall framework. The real-time ingestion and decoding of the raw streaming data has been described in Section IV followed by a brief on batch ETL processing to build BDW (Section V). The analysis and results of various experiments are summarized in Section VI. The paper is concluded in Section VII.

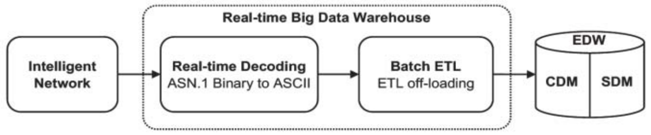

Figure 2. Block diagram representing the key components of the Framework

II. Existing System

The CSP was previously using Nexus, Informatica and Teradata systems for IN mediation, ETL and database management respectively. These systems were already overloaded, making delays in processing huge datasets which was costing alot in-terms of expansion, scalability and management. Despite the fact, CSP continued using these systems for more than a decade but the change of technology became critical at one time to continue their operations.

The CSP's IN Mediation generates four different types of CDR source which include CCN, HxC, SDP and AIR. The Charging Control Node (CCN) deals with subscribers credit control, billing and traffic related information whereas HxC is managing offer bundles subscription. The Service Device Point (SDP) is responsible for a subscriber's life cycle management and the Account Information and Refill (AIR) handles all types of recharges. So when a call event is registered on IN, an initial ticket is created which in-turn generates a trigger on SDP. Eventually, SDP links that event with the remaining three sources of IN. All IN sources are linked with each other to get required information from the other three sources.

The IN is being used as the charging system which keeps inquiring about the call operational details from the network; this process is known as interrogation. Accordingly, the charging system ensures to the network whether an on-going call is eligible for next 5 minutes session (an interrogation) or not. Perhaps to ensure the charging system integrity, Network Management Center (NMC) calls interrogation cycle, as described above, and confirms whether the subscriber has privileges to make or continue a call. The subscribers call is kept alive in case either enough credit, Discount Accounts (DA) or any offer has been found in the subscriber's account. Therefore, all four IN sources generate CDR files in periodic manner based on calling life-cycle. These CDR sources are sent to the mediation which converts it into ASCII JSON format and split them in small batches which becomes input of the ETL system. Figure 3 explains the existing mediation and ETL processing environment of the CSP.

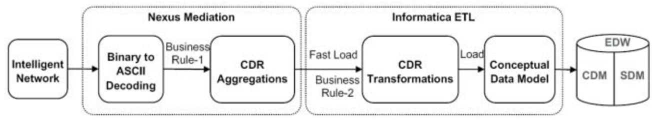

Figure 3. Existing IN Mediation and ETL process flow

Previously, the CSP was using traditional tools and technologies to manage and ETL their data. The traditional system uses Fast-load recursive workflows, which execute after every minute to perform CDR decoding. Furthermore, the data pooling is also a continuous process where the fast-load table is overridden when it gets new batch of CDR. There is another workflow executed to perform the required ETL operations and loads the new batch in Conceptual Data Model (CDM). Finally, this batch is dumped in SDM using a merge utility which synchronizes new batches with the existing ones. Figure 4 gives detailed understanding of CDRs data mapping from IN sources to final target tables.

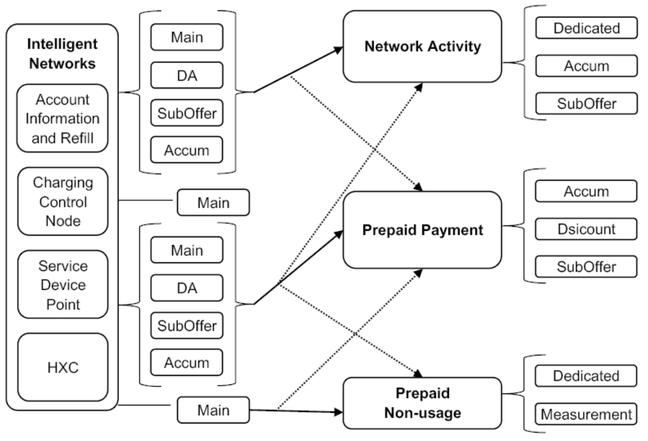

Figure 4. IN sources to EDW-CDM mapping

III. Methodology

This section discusses the applicability of the proposed framework to offload the CSP's IN Mediation and ETL. The IN generates four different types of information against a subscriber's call, the sources are mentioned in above section. The Telco switch cuts the stream of continuous incoming CDRs into batches based on either time or file size. The batches are split into four different sources and pushed to the respective folders in the form of compressed files. These files are consumed by a directory scanning module as they land. The directory scanning module keeps scanning the directories for new incoming files. The newly arrived files are fetched from the source directory using Apache NiFi in a faulttolerant manner. The NiFi uncompresses these files and convert them into binary streams. These streams are further pushed to Apache Kafka data pool which acts as queuing system. The decoding module consumes binary stream from the queue and converts it into ASCII format in near real-time mode. The decoding process is driven by various grammars, provided by the vendors of the Telco switch, which specifies the rules and guidelines against all the attributes of the CDR. The ASCII CDRs are loaded back in the queues which provide real-time feeds to various systems including ETL, analytics and campaign management. Furthermore, the rate of data flow between real-time and offline module is also controlled by the Kafka. These queues have capability to automatically expand and shrink based on the load of data on the system. The ASCII CDRs are dumped in Hadoop to build data-lakes as well as they are consumed by the offline ETL module. This ETL module computes aggregations and transformations to build a replica of EDW-CDM (the existing system) in Hadoop as mentioned in Figure 7. The Hadoop-CDM is further aggregated at a subscriber-level and stored in EDW-SDM. The SDM becomes the input source of various other systems of the CSP.

The CDRs stored in SDM are aggregated on hourly and daily windows and synchronized in EDW-CDM after every 4 hours. The Hadoop-CDM batches have been initially dumped in staging EDW-CMD to perform number of lookups, and after that a merge tool pushes them into EDW-CDM. To move data smoothly from Hadoop platform to EDW, message brokers and queues are implemented to provide a flexible connectivity between RDBMS and Hadoop. Accordingly, the message brokers and queues also contribute key functionalities like check-pointing, communication between two different systems and control rate of data flow and queuing. The overall framework including Hadoop clusters are administered and secured by various tools which are discussed in the upcoming sections.

A. Data Integration and Queuing

The data integration and queuing components are usually used for flexible communication between different modules of a distributed system [7]. In this work there are two key components used for this purpose, NiFi and Kafka. NiFi is a real-time integrated data logistics and a simple event processing platform. It has been used for directory scanning and reading the binary Abstract Syntax Notation One (ASN.1) format. It automates the movement of data between different data-sources and systems to make data ingestion fast, easy and secure [9]. Also, it handles data redundancy issues so that no single file can be fetched more than once from the FTP directory. The decompressed binary stream is pushed from NiFi to a fast, scalable and a fault tolerant publish-subscribe messaging system - Kafka. Kafka is a high-throughput distributed messaging system that has become one of the most common streaming data management component [16]. Spark Streaming consumes data stream from Kafka using direct API integration which is a receiver-less approach. Additionally, a fault-tolerant system has been implemented to handle failover cases and to achieve more parallelism. Therefore, the applicability of proper data provision rules helps Kafka to minimize chances of data loss. Both NiFi and Kafka are also used to replicate ETLed data from Hadoop-CDM to EDWCDM.

B. System Administration

Another important pillar of any Big Data application is its administration which includes provisioning, management and monitoring of Hadoop clusters [12]. The System Administration has been handled by Apache Ambari which exposes an easy to use interface to provision, monitor and manage various services within Hadoop cluster. Ambari simplifies installation, configuration and management of various components required by Big Data ecosystem. Additionally, it provides centralized security, customization, extensibility and full visibility of clusters health [13].

C. System Security

A set of comprehensive security measurements has been incorporated in Hadoop cluster so that the CSP can securely extend Hadoop data access to their users having different roles. The framework security has been achieved using Apache Knox gateway and Apache Kerberos. Along with the perimeter level security, comprehensive approach to secure Hadoop cluster at services level has been achieved using Apache Ranger. Ranger provides a centralized platform to define, administer and manage security policies consistently across Hadoop components [12]. Furthermore, Kerberos is used to link Hadoop ecosystem with the existing Lightweight Directory Access Protocol (LDAP) of the CSP using secure Single Sign On (SSO).

D. Operational Model

The production environment has been deployed on Intel architecture. The operational model representing various components of the framework is explained in Figure 5. This framework requires several software components which are packaged under Hortonworks Data Platform (HDP) [10] and Hortonworks Data Flow (HDF) [11]. The operating system used for development, staging and production environments is RedHat Enterprise Linux (RHEL) v7.x. The Master, Edge and Data nodes are connected on the same Top of the Rack (TOR) switches to maximize data transfer speed among data nodes.

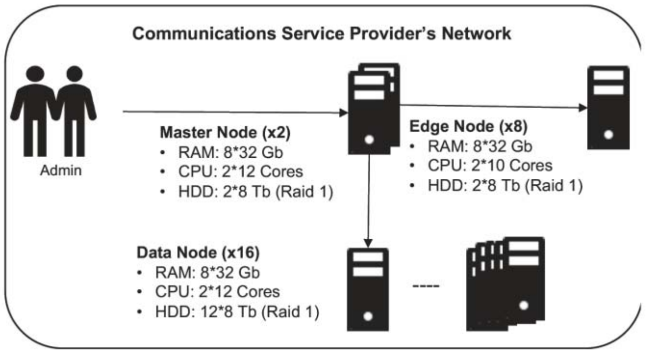

Figure 5. Operational Model

IV. REAL-TIME DATA INGESTION AND DECODING

The primary goal of Real-time IN Mediation in a CSP is the conversion of streaming ASN.1 binary CDRs to readable ASCII format. The binary streams land in FTP directories in compressed archives form. These directories are under continuous monitoring of NiFi. Thus upon arrival of a new file NiFi uncompresses it, read the contents and push it in Kafka. On the other hand, Spark Streaming consumes the batch of stream and decodes it by following the infrastructure specific grammar rules. The decoded stream writes back on Kafka from where several other systems consumes this realtime feed. Similarly, Spark SQL consumes the incoming batch from Kafka and dump it into Hadoop data-lakes in ORC format. Figure 6 is explaining the functional architecture of IN Mediation offloading module. Following are the key steps that this module performs:

1) NiFi scans FTP folder to ingest binary CDR archived files and load them to respective pipeline.

2) Kafka reads data from NiFi queues to create data streams as well as it stores the converted CDRs in Hadoop.

3) Spark Streaming consumes Kafka queues and loads them in Resilient Distributed Dataset (RDD) for inmemory binary to ASCII decoding.

4) The decoded CDRs are published back to Kafka.

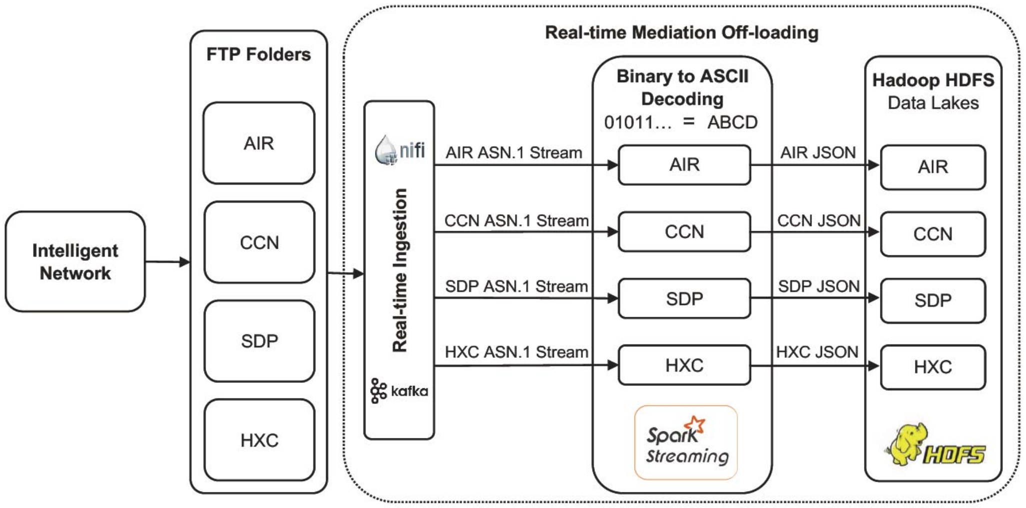

Figure 6. IN Mediation offloading Architecture

Spark Streaming API processes binary stream in the form of in-memory mini-batches known as Dataset [15]. The decoded batches are converted into JSON format where each JSON record contains a CDR of a subscriber. These JSON CDRs are further compressed and then stored into Hadoop data-lakes via Kafka. Moreover, JSON contains data in a standard data interchangeable format which is lightweight, text-based and human-readable. It has a compact message size which makes it faster for I/O operations as compared to other formats like XML.

A. Directory Scanning

This module keeps scanning the input FTP folders and generate an event upon arrival of a file. The event invokes NiFi to process incoming files. The volume of files varies from 70,000 to 100,000 per day where Table I is showing average file sizes and frequencies.

TABLE I. DAILY IN FILE SIZES AND FREQUENCIES

Statistics	AIR	CCN	SDP	HXC
Average Frequencies of the Files	26,036	51,937	30,419	9,440
Average size of a file	12 MB	35 MB	15 MB	45 MB

Apart from the directory scanning and file ingestion, NiFi also ensures deduplication of the incoming CDRs. It is an integrated data logistics platform for automating the movement of data between different systems [9]. It provides realtime control that makes it easy to manage the movement of streaming data between any two sources. Likewise, NiFi takes care of redundant files by analyzing the filename, timestamp and other attributes.

B. Failover

The Failover is managed via Kafka which is built to be faulttolerant and horizontally scalable [16]. It is positioned in the framework to solve two key challenges: a) to assure reliability which makes it capable of scaling against the growing demand of events passing, b) to interact with various applications and act as a central message hub for the framework [3].

V. BIG DATA ETL

Big Data ETL is the second key component of the framework. This module invokes upon arrival of new decoded batch on Kafka. The ETL operations mainly consist of transformation and aggregation of the CDRs. The IN produces several interrogations against a call session where each session lasts for 5 minutes. These interrogations are stitched against a subscriber's call. Therefore, voice call session can have maximum 12 interrogations, where it is automatically terminates after 60 minutes. On the contrary, in case of data (internet) usage, a session can last for 6 hours. Thus, all the interrogations of a call require aggregations at the subscriber-level. The CCN CDRs are split in different target tables for revenue and usage verification purpose. The different tables that a CCN CDR produces are CCN Main, DA (discounts), Refill, Sub-DA (sub discounts) and Accum (accumulator). Accordingly, AIR CDRs are split in DA (discounts), Refill, Sub-DA (sub discounts) and Accum (accumulator) after applying aggregations. The aggregated CDRs are fed back in Kafka queues from where they are consumed by transformation module as well as a copy of aggregated CDRs is dumped in Hadoop.

The aggregated CDRs are further processed to perform transformations and definition table lookups which are used to normalize the ETLed data. The transformation operation combines all the CDRs against a subscriber and perform computations on the call utilization related data. The data normalization basically maps the network operation data terminologies with the business terms, for an instance, Telco switch uses a unit to represent 'free minutes' whereas financial application refer it with a different unit. The transformed data source is known as CDM which is stored in Hadoop (known as Hadoop-CDM). The aggregations and transformations are stored in a row-column compressed format in Hadoop. Further, the Hadoop-CDM is synchronized with the existing EDWCDM 6 times in a day where a batch runs after every 4 hours. However, there is a possibility that latencies occur in the system due to circumstances like:

- the time-frame between a call and closing of the corresponding CDR file.
- the transfer of the CDR files from the network source to the location where Spark Streaming can read the files.
- peak load traffic and corresponding huge file volumes which cause back pressure in processing, either reading, decoding, aggregating or transforming files.

The decoded CDRs are ETLed in-memory to minimize the end-to-end processing time. Thus, Kafka becomes a bridge between different modules to ensure the consumption of the streaming data in different rates for real-time and offline modules. Moreover, a scheduler keep synchronizing the Hadoop-CDM with staging EDW-CDM via NiFi. Eventually, the staging EDW-CDM synchronizes newly processed batch in EDW-CDM from where a 'merge utility' incorporates it with existing data of EDW-SDM. The SDM becomes central data repository for all the business network, finance and other applications. Figure 7 explains the Hadoop ETL architecture in detail.

A. Technology Overview

The Big Data ETL component is mainly using Spark SQL to perform transformation and aggregation operations. It is invoked by Spark Streaming module upon successfully decoding the CDRs. Spark SQL pooling processor consumes the decoded CDRs from Kafka queues and loads them in the form of mini-batches in Spark SQL Dataset. A Spark Dataset is used to manage CDRs in Spark SQL's memory. A Dataset is a distributed collection that provides the benefits of both RDDs and Spark SQLs optimized execution engine [14]. The SQL function in a SparkSession enables the application to run SQL queries programmatically and returns the results back to the Dataset. However, instead of using Java serialization or Kryo, the Dataset uses a specialized encoder to serialize the objects for processing or transmitting over the network. Thus, both encoders and standard serialization are responsible for converting an object into bytes. Moreover, the encoders generate code dynamically and use a format that allows Spark to perform different operations including filtering, sorting and hashing without de-serializing the bytes back into an object [14].

B. Failovers

Kafka and NiFi together are responsible for moving data from one module to another one where they are also responsible to manage failovers. These components handle any situation that either occurs due to system failure like network failure, disk failure or software crash. In case of a failure the CDRs are queuing-up on Kafka. Kafka interacts with various modules and applications to broadcast events, which helps to orchestrate todays complex architectures and acts as a central message hub [6].

NiFi assures guaranteed delivery of pre-mediation ASN.1 files to Spark Streaming for real-time decoding by pushing the data files in Kafka consumer pool. In case any packet gets lost due to any unseen reason, NiFi fetches that packet again from the source. In order to avoid bottlenecks during the CDRs decompression process, NiFi provides support for buffering of all queued data at ingestion and decompression points.

NiFi automatically records, index and make available premediated data as it flow through the system. NiFi starts recording this information from file ingestion point till the end when data batches are pushed to Kafka pool. This information is considered extremely critical and helpful in supporting compliance, troubleshooting, optimization, and other scenarios.

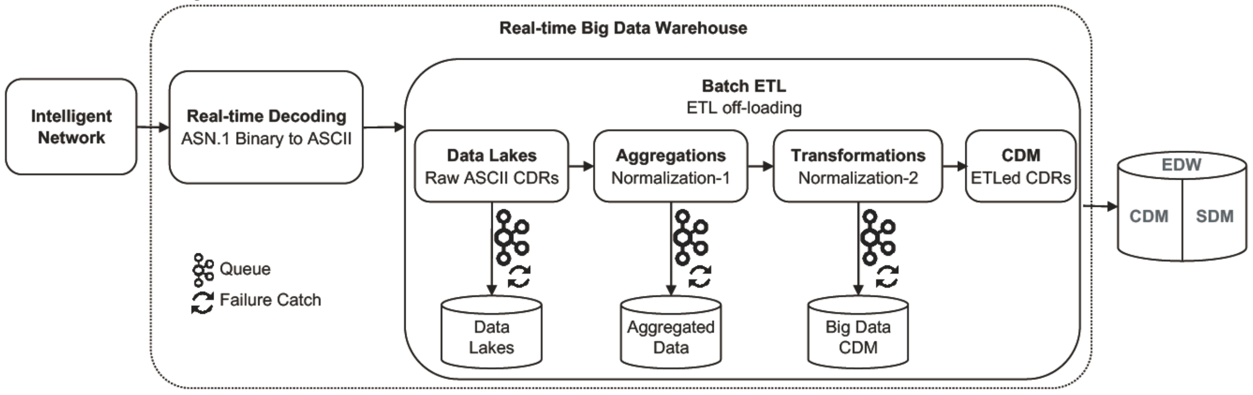

Figure 7. ETL offloading architecture

VI. Analysis and Results

The proposed framework has been tested on a CSP with around 51 million subscriber-base generating millions of events per hour. Initially both real-time and offline modules were performing well independently. However, after integrating both modules the CDRs were getting queued up because batch processing module's data ingestion frequency was not synchronized with the real-time module. This problem has been solved by tuning Spark Streaming and SQL parameters which includes windows size, number of batches, interval, etc. Most of the real-time analysis and campaign management applications of the CSP consumes decoded data from Kafka whereas the final ETLed data is being fed to rest of the systems. Finally, the end-to-end system has been processing a batch of stream in under 10 minutes which was the primary objective of positioning Big Data technology.

VII. Conclusion

Large enterprises use Big Data to overcome the limitations of traditional data-warehouse and analysis platforms. This paper presents a flexible framework to deal with massively large and complex datasets both in real-time and offline modes. The paper also discusses the applicability of this framework on a CSP and the typical challenges they face during the implementation of these types of complex systems. After managing the data in Hadoop, the raw CDRs have been used in two real-time use-cases initially; deep packet inspection (DPI) and location based marketing (Geo-fencing) [8]. The real-time streams of binary IN CDRs convert into ASCII to build data-lakes where all the interrogations against a call are stitched and all the calls of a subscriber are aggregated. The end-to-end processing takes no longer than 10 minutes. The DPI and Geo-fencing analysis of subscribers usually require transactional real-time streams which was not possible using traditional EDW. Also, the key challenges that the CSPs are facing in absence of Big Data are: the late-arrival of input data to other CSP applications, highly heterogeneous data arrival patterns and latencies, and frequent changes in data schemes. This framework makes it possible to efficiently manage raw data, and implement more intelligent and complex use-cases which help in improving other areas of the CSP including core networks.

References

[1] Y. He and F. R. Yu and N. Zhao and H. Yin and H. Yao and R. C. Qiu. Big Data Analytics in Mobile Cellular Networks. Special Section on Theoretical Foundation For Big Data Applications: Challenges and Opportunities, Apr 2016.

[2] D. Z. Yazti, S. Krishnaswamy. Data Stream Warehousing In Tidalrace. CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Jun 2015, Asilomar, CA, USA.

[3] Vibha Bhardwaj, Rahul Joharis. Big data analysis: Issues and challenges. In Proceedings of the 16th Electrical, Electronics, Signals, Communication and Optimization (EESCO), Jan 2015, Visakhapatnam, India.

[4] Fei Su, Yi Peng, Xu Mao. The research of big data architecture on telecom industry. In Proceedings of the 16th Communications and InformationTechnologies (ISCIT), Sep 2016, Qingdao, China.

[5] D.Z. Yazti, S. Krishnaswamy. Mobile Big Data Analytics: Research. Practice and Opportunities 2014 IEEE 15th International Conference on Mobile Data Management, vol. (l), pp. 1-2, 2014.

[6] M. Sindhu, P. N. P. Hegde. A Survey of Tools and Applications in Big Data. IEEE 9th International Conference on Intelligent Systems and Control, pp. 1-7, 2015.

[7] Z. Jie, X. Yao, G. J. Han. A Survey of Recent Technologies and Challenges in Big Data Utilizations. International Conference on Information and Communication Technology Convergence (ICTC), pp. 497-499, 2015.

[8] A. Rehman and A. R. Ali. Customer Churn Prediction, Segmentation and Fraud Detection in Telecommunication Industry. In Proceedings of the 4th ASE International Conference on Big Data, Harvard University, Dec 2014. ASE, Cambridge, MA.

[9] Apache NiFi. Hortonworks. https://docs.hortonworks.com, 2017.

[10] Apache Spark SQL. Hortonworks. https://docs.hortonworks.com, 2017.

[11] Hortonworks Data Flow (HDF). Hortonworks. https://docs.hortonworks.com, 2017.

[12] HDFS Administration. Hortonworks. https://docs.hortonworks.com, 2017.

[13] S. Wadkar and M. Siddalingaiah. Apache Ambari. In: Pro Apache Hadoop, Apress, Berkeley, CA, 2004.

[14] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, M. Zaharia. Spark SQL: Relational Data Processing in Spark. SIGMOD, June 2015.

[15] M. Zaharia, T. Das, H. Li, S. Shenker, I. Stoica. Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters. HotCloud, June 2012.

[16] J. Kreps, N. Narkhede, and J. Rao. Kafka: a distributed messaging system for log processing. SIGMOD Workshop on Networking Meets Databases, 2011.