Real-Time Telecom Revenue Assurance

Seamless real-time & batch processing of telecom transaction records

Debnarayan Kar, Prateep Misra, Prasun Bhattacharjee, Aniruddha Mukherjee
TCS Innovation Labs - Kolkata
Tata Consultancy Services Ltd.
Kolkata, India
Emails: {debnarayan.kar, prateep.misra, prasun.bhattacharjee, aniruddha.mukherjee}@tcs.com

Abstract— **Telecommunication industry has seen phenomenal growth and intense competition in recent times. Basic connectivity has been commoditized and service providers need to provide personalized and differentiated services to stay competitive and form lasting customer relationship. One of the ways they can do this is by analyzing the data that are generated or pass through their network in real-time, and use the insight to offer personalized services promptly and dynamically. In this scenario it will be immensely beneficial to them to have solutions which allow them a smooth transition path from offline batch processing of stored data to real-time analysis of data as it is available to them. The current paper demonstrates an approach to address the challenge of solution development using Stream Processing, where the same solution can be used to process stored data in offline mode and analyze data stream in real-time for actionable intelligence. IBM stream processing platform Infosphere Streams has been used for implementation of the solution proposed in this work. As a further benefit, if some problem is discovered during real-time analysis of data streams, the same program can be used to analyze the stored data after necessary update is made to the program. This work focuses on seamless dual mode processing of both offline and real-time analysis. The approach has been carried out with reconciliation component of telecommunication revenue assurance and has used CDR (Call Detail Record) for analysis. However, the approach is very generic and it can be applicable in other areas of telecom like churn management, campaign management and in other sectors, like Utilities, Banking etc., which can benefit from real-time and seamless dual mode processing of account transaction records.**

Keywords-revenue assurance; real-time; CDR processing, stream processing

I. INTRODUCTION

There has been phenomenal growth in mobile telecommunication in different parts of the world and the growth momentum is continuing still, more specifically in the emerging markets. As per International Telecommunications Union (ITU) Telecom World 2011 report, globally, there are 5.9 billion mobile subscriptions [1]. This number will rise even further in the future, as mobile penetration in different parts of the world including India, China, and Africa has significant room for increase [2].

Apart from the large number of subscribers, the operators are also facing huge competitive pressure as the high growth prospect in mobile and telecommunication has attracted multiple players to the field.

Communication Service Providers (CSP) are increasingly adding digital content, applications, and other value added service in their offerings to attract and retain customers. If they do not do this, they loose on two counts:

- They do not have differentiators, and face service-churn and subscriber-churn.
- They incur the cost of expensive network equipments in their network, but cannot get the optimum value out of it. They simply become fat-pipe provider and the 'Over The Top' (OTT) suppliers profit at their cost by selling content, application to consumers using their network.

The service providers have been offering digital content like movies, music, games, and applications. The present day offerings are limited in supply from most service providers and are expected to grow phenomenally. Even at the current level of service offerings, some service providers have to process multiple millions and for some large operators billions of Call Detail Records (CDR) or Usage Detail Records (UDR) per day.

To remain competitive in the increasingly competitive business environment, the service providers need the capability to source content from multiple partners and dynamically create differentiating offerings using the sourced content and applications. They need the ability to charge for the consumed services in real-time and prevent misuse or unauthorized use. This will increase the number of accounting records required to be processed by multiple order of magnitude. In this scenario, there will be multiple billions of usage accounting records which needs to be processed in real-time to get maximum value from installed network and offered services.

These activities require processing and analyzing high volume of data originating or passing through their network in real-time, near-real-time or in very low latency mode.

Phenomenal subscriber growth and emergence of machine-to-machine (M2M) communication have been leading to very high volume of data stream. To derive the maximum benefit, it is necessary to have computing solution to process high volume live data stream in real-time or near-real-time.

Section II provides an overview of the telecommunication reconciliation process and also the problem, which is addressed by the current work. Section III describes the challenges encountered while addressing the problem. Section IV outlines some of the other methods and mechanisms to analyze high volume of data promptly. Section V provides an overview on stream processing techniques in general and also certain key features of IBM Infosphere Streams. Section VI outlines the approach adopted for the solution and Section VII describes

 ISBN: 978-1-61208-193-9

and analyses the results from the current implementation. Section VIII lists some of the ways to enhance the current work and also enumerates some other fields, where the techniques of this work can be utilized. Finally, the concluding section reiterates the usefulness of the current work.

II. OVERVIEW AND PROBLEM STATEMENT

The reconciliation component of Telecommunication Revenue Assurance system was taken as an example for exploring different approaches to speed up high volume CDR processing. The component reconciles CDR data from Mobile Switching Center (MSC) and Telecommunication Billing System. In a mobile communication network, the MSC coordinates communication channels and necessary processesfor end-to-end connection handling, mobility management, routing charging and real-time account monitoring. The initial work has been limited to the case of reconciling pre-paid outgoing voice call because this use case will benefit most through real-time reconciliation. Service providers may have hybrid customers who use pre-paid and post-paid services on the same account. Hybrid customers have been excluded from this work, because their combined service usage can be reconciled in suitable batch modes.

After the calls are setup between the calling party and called party and as the calls progress, the CDRs are generated in the MSC. The CDR contains several fields including identity of calling party and called party, call start date, call start time, call duration etc. The CDRs contain meta-data about the call or communication service usage which are useful for charging and other management purpose, however they do not contain the actual content of the call or communication.In this current work, since the CDRs are used for their intended purpose of rightful charging, the usage does not violatesubscribers' privacy concern.The current work does not, in any way, prioritize or block subscribers' data content in the delivery process. Thus, it does not raise any "Network Neutrality" (Net Neutrality) issues. The format of the CDRs usually differs among MSC equipment vendors. The formats of theCDRs are also different between MSC and other downstream applications like Billing System and Customer Care System which process them. A Mediation Device receives CDRs from MSC, performs aggregation and correlation among relatedCDRs and reformats them to the format of the downstream application like Billing System. The MSCs, Mediation Device, and Billing Systems are from different vendors, and at times due to error in interface between them, discrepancies crop up in some CDRs as it passes through them. To address the problem of discrepancy, the CDRs need to be reconciled between MSC and the Billing System. Reconciliation is the process of comparing records from multiple sources to verify their accuracy.

When the charge duration between MSC and Billing side for corresponding CDRs differ by more than a threshold, an alert needs to be generated and saved. Corresponding CDRs are matched by their respective values of calling party, called party, call date, call direction, and allowable tolerance threshold in call start time.

As per current deployments in various service provider organizations, the reconciliation is an end-of-day processing where reconciliation needs to be completed in a small window of time during the off-peak hours, usually in the night. This work attempts to suggest an approach which will be able to support both the current deployment scenarios of batch processing as well as the real-time reconciliation scenarios for telecom service providers.

The scenario of the problem is described with the help of Fig. 1. The CDRs from the MSC need pass to a few rounds of filter to:

- Filter in only outgoing call CDRs
- Filter out CDRs which comes from hybrid customer types based on information available from a look-up file.
- Filter out CDRs from post paid customer types

The Billing CDRs need to pass through one simple filter to allow only outgoing call CDRs. The MSC and Billed CDRs needs to be joined where calling party, called party, call date, call direction are same in MSC and Billed CDR and Call Start Time in MSC and Billed CDR are within a configurable tolerance threshold of a few seconds. Finally, alerts needs to be generated from the joined CDRs where the Charge Duration in matched MSC and Billing CDR differ by more than a user configurable threshold.

The purpose of current reconciliation process is to capture and report in real-time, the CDRs between MSC and Billing System, which are under-charged or over-charged beyond a user-configurable tolerance threshold value.

III. CHALLENGES

In case of current work, we are required to support an end-of-day processing with huge volume of data accumulated over the day and also we need to process these CDRs very fast with high throughput as well as real-time reconciliation.

In case of real-time processing, the data are processed as they arrive with the use of suitably designed windows to optimally manage the flow of data streams. In case of end-of-day batch processing, we have two accumulated data sets whose sizes are different.

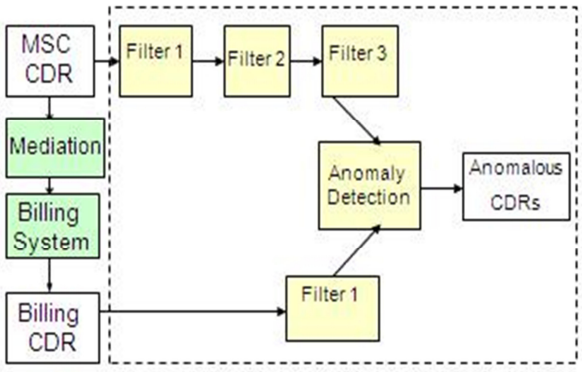

Figure 1. Reconciliation of MSC and Billed CDRs

The data from the MSC have all billable CDRs whereas the CDRs coming from the Billing System have already gone through a few rounds of filtering. Thus, the number of MSC CDRs is more than those of Billed CDRs in any given period of time. MSC CDRs need to pass through three levels of filtering whereas the Billing CDRs need to go through a single simple filter thereby taking a longer time to filter the MSC CDRs. If the application starts the ingestion of MSC and Billing CDR at the same time, and the windows for MSC and Billing CDRs are not properly selected, then it may happen that by the time a particular MSC CDR arrives in its window after

 ISBN: 978-1-61208-193-9

passing through multiple cycles of filters, its corresponding Billing CDR has already been removed from the Billing window. When this happen, then the related MSC and Billing CDRs cannot be joined. If there is a deviation in Chargeable Duration between those MSC and Billing CDRs, then the corresponding alert will not be generated.

This necessitates synchronization of the ingestion of the two data sets with careful selection of the windows and achieve high throughput processing without any data loss due to windowing problems.

IV. STATE OF THE ART

Very large amount of data can be reduced in space using wave-based approximation [3]. Using this approach, queries can be run on stored data promptly and with reasonable accuracy. Signature based methods have been applied for very fast processing of telecommunication data streams in fraud detection applications [4]. This paper utilizes the fact that fraudsters will have usage patterns which are different from regular users, captures the fraudsters' pattern in the form of statistical signature, and use those for prompt comparison with the CDR stream. This approach is very specifically geared towards fraud management applications in telecom and other industries.

Arroyo solution from Telcordia can efficiently extract, transform and load relevant fields from CDRs stored in data ware house [5]. XML-based transformation rules are used to extract the required CDR fields for faster processing.

There are applications outside the telecom industry which require real-time processing of huge volume of transaction records. In capital market surveillance space, stream processing mechanism has been used for very low latency processing of high throughput transactional data [6].

Graph-theory-based innovations are in exploration to promptly answer queries on huge datasets. Graph-based, disk resident indexing has been devised and utilized for fast queries on high volume RDF data [7]. Complex queries have been answered using sub-graph query matching techniques on a cluster of computers in social network domain which can be applicable generically in other areas also [8].

The works explored in the start of art outlines various methods and mechanisms to analyze high volume of data promptly. Majority of them can provide very high throughput processing on stored data in offline mode. There are also mechanisms for real-time processing of data streams on the fly [9]. Current work addresses the space where the same solution can be used to process high volume of data in real-time as well as in offline batch processing mode.

V. OVERVIEW OF STREAM PROCESSING

Stream processing is a relatively new computing paradigm which is useful for processing and analyzing high volume of data very fast [10]. It can be appreciated by contrasting with database management system (DBMS). In DBMS, the data is stored and is static in nature. Queries are periodically executed over the data to gain insight. In contrast, stream processing has queries residing in the system which are known as continuous queries, and which execute continuously over data streams that are passed through the system. Input data stream can be potentially infinite and stream processing employs the concept of windowing to limit the amount of data records which need to be processed at any particular time. Size of a window can be decided by a number of criteria as specified below [11]:

- **By row count (Count-based window)**: For a count-based window with count value of N, it can accumulate a maximum of N tuples or records. Newly arriving tuples cause older tuples to be evicted if the window is already full.
- **By elapsed time (Time based window)**: For a time-based window with aduration of N seconds, the tuples/records arriving in the last N seconds are accumulated. Tuples which arrived earlier than N second are evicted.
- **By delta of an attribute (Attribute Delta based window)**: In Streams, like in many other system, the data records are called Tuples and the columns or fields in the data records are called Attributes. When the difference in an attribute's value between the earliest tuple in the window and latest tuple is more than the delta threshold, the earliest tuple is removed and the latest tuple is added to the window. If an attribute-delta based window is defined on an attribute named CallStartTime and delta value of 300, then earlier tuples are removed from the window, if their CallStartTime value differs by more than 300 seconds. The attribute value needs to be continuously increasing.

Window can be Tumbling Window or Sliding window. In tumbling window, after the content of the window are processed, its entire content is evicted. In sliding window oldest N records are evicted as newer N records arrive for processing. One tuple can be present in multiple processing steps in sliding window, but in tumbling window one tuple will be present in one processing step only.

The IBM product, Infosphere Streams, has been used for its capability of big data processing and high performance computing [12].It can handle petabytes of data per day and can support traditional and non-traditional data (audio, video etc.). It delivers insights with microsecond latencies and supports the user-defined functions (custom analytics) written in languages like C++ or Java, which makes implementation of complex analytics very easy for developers. Moreover, a single instance of it can support multiple applications.

InfoSphere Streams has a set of built-in stream relational operators which can take care of wide range of requirements.

InfoSphere Streams provides

- A programming model and a language (SPL) for defining data flow graphs consisting of datasources (inputs), operators, and sinks (outputs)
- Controls for fusing operators into processing elements (PEs)
- Infrastructure to support the composition of scalable stream processing applications from these components
- Deployment and operation of these applications across distributed x86 processing nodes, when scaled-up processing is required
- A visualization tool can monitor the running Streams applications distributed across hundreds of servers.

The set of built-in stream relational operators of the SPL can take care of wide range of requirements; some of these are as stated below [11]:

- **Source**: a Source operator is used for creating a stream from data flowing from an external source. This

 ISBN: 978-1-61208-193-9

operator is an edge adapter, capable of performing parsing and tuple creation as well as of interacting with *external* devices.

- **Sink**: a Sink operator is used for converting a stream into a flow of tuples that can be used by components that are not part of an InfoSphere Streams instance. This operator is also an edge adapter and its main task consists of converting tuples into corresponding objects, accessible externally through the file system, network, or some other external device.
- **Functor**: a Functor operator is used for performing tuple-level manipulations such as filtering, projection, mapping, attribute creation and transformation.
- **Aggregate**: an Aggregate operator is used for grouping and summarization of incoming tuples. This operator supports a large number of grouping mechanism and summarization functions.
- **Join**: a Join operation is used for correlating two streams. Streams can be paired up in several ways and the join predicate, for example, the expression determining when tuples from the two streams are *joined* can be arbitrarily complex.
- **Sort**: a Sort operator is used for imposing an order on incoming tuples in a stream. The ordering algorithm can be tweaked in several ways.
- **Punctor**: a Punctor operator is used for performing tuple-level manipulations, with the exception of filtering. Unlike a Functor, a Punctor can insert punctuations into the output stream based on a user-supplied punctuation condition.
- **Split**: a Split operator is used for splitting a stream into multiple output streams, based on a split condition that is used to determine which of the output streams a tuple is to be forwarded to.

This platform can be augmented by its simple integrating ability with other visualization products, analytics tools or report-generation tools; through its broad range of stream adapters to consume and publish data from external sources such as network sockets and relational and XML database.

This product is gaining popularity because of its high scalability and robustness in the run-time environment. Additionally it has the ability to add or remove the resources to or from the cluster without impacting the running applications.

VI. APPROACH TO SOLUTION

The application keeps N hours' filtered records in the MSC window, where N can be configured by user and the filtered billing records (or tuples) are compared with the records in the MSC window as they arrive. To ensure that the MSC records are already filtered and are inside the window, a carefully selected small delay is introduced before the ingestion of the billing CDRs.

The window size should be sufficiently large, so that it can hold MSC records of N hours. Count-based window cannot be used as it would require a-priori knowledge of the number of records in N hours' CDR data, which is not available when records are processed in real-time as they arrive. While a time based window can accommodate the requirement real-time processing, it cannot be used in the offline batch processing requirement, because in batch processing mode, the record will arrive at the speed of disk read. So, a one hour window will have all CDRs, which are read in one hour's time rather than all calls those were made in a particular one hour interval.

An attribute-delta based window on the call start time can satisfy both real-time and batch processing requirement. Say for example, for a three hour processing window, records are accumulated in the window as long as the call start time of the arriving records are within N hours (3600xN seconds) of the oldest record's call start time. As this method does not require a-priori knowledge of record number, it can be used in both real-time and batch processing mode.

This means that the filtered billing CDR, as it arrives, is compared with all the available records in the MSC window and once the joining is completed, the billing record is evicted.

As can be seen in Fig. 2, the MSC window is an attribute-delta based sliding window which contains N hours' CDR data. Each Billed CDR arrives in the Billing window and is compared against all the CDRs which are present in the MSC window at that instance of time for the purpose of joining. If the join predicate (joining criteria) is satisfied, the MSC and Billed CDRs are joined. The billing window is a count based window of size 1, and after the comparison is performed, the CDR in the billed window is evicted.

One advantage of this approach is that all the MSC records, which match against a single billing record, are located sequentially in one place in the joined records. This situation is particularly useful to remove anomaly in the joined records which happens if more than one call is made between the same parties within the tolerance threshold of K seconds during joining. The scenario is elaborated through an example bellow.

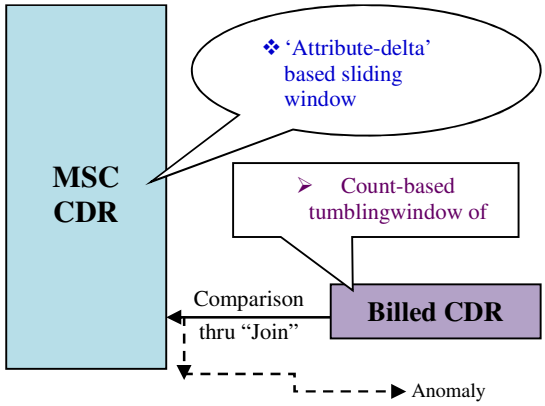

Figure 2. CDR joining with different sized windows

During the joining process, in this example scenario, a tolerance threshold of 60 seconds is allowed on the field Call Start Time. This allowance is made because in many instance the call start time in the billing CDRs is shifted by certain amount from that in the corresponding MSC CDRs.

If same calling and called parties are involved in more than one calls of different duration within the tolerance threshold of 60 seconds, there are multiple matches for same CDRs leading to some false alarms. Table I shows 2 separate calls between the same calling party (A) and called party (B) within the tolerance threshold interval and call start time in the MSC and the corresponding Billed CDRs are shifted by certain amount. The MSC CDRs have among other things, the fields in column

 ISBN: 978-1-61208-193-9

1, 2, 3 and 4 in Table I and the Billed CDRs have the fields in column 1,2, 5 and 6 in Table I

TABLE I. TEST RESULT

Calling Party	Called Party	Call Start Time in MSC	Charge Duration in MSC	Call Start Time in Billing	Charge Duration in Billing
A	B	37030	2	37015	2
A	B	37040	18	37025	18

Table II shows the result after the MSC and Billed CDRs are joined based on calling party, called party and a tolerance threshold of 60 seconds on the "call start time" field.

TABLE II. TEST RESULT (AFTER JOIN)

Calling Party	Called Party	Call Start Time in MSC	Charge Duration in MSC	Call Start Time in Billing	Charge Duration in Billing	Deviation
A	B	37030	2	37015	2	0
A	B	37040	18	37015	2	16
A	B	37030	2	37025	18	16
A	B	37040	18	37025	18	0

In this example, 1st and 4th matches are valid match. The 2nd and 3rd matches are not the intended matches even though they satisfy the joining criteria and subsequently generate alarms which are false alarms. In the subsequent post processing step, the joined matches which cause false alarms are removed from the live data stream.

VII. TEST RESULT & ANALYSIS

The solution was tested on an Intel based X86 HP Server with 12 GB RAM and 2 Quad core CPUs. The CPU in the test environment was equipped with Intel(R) Xeon(R) CPU with 2.00GHz speed and 4MB Cache size. Red Hat Enterprise Linux version 5.4 has been used for this work. IBM Infosphere Streams has been used as stream processing platform. This approach has been tried on 1 hour data set and 3 hour data set from one MSC respectively and tests were repeated 5 times for each case. The time taken for each case is show in Table III.

TABLE III. TEST RESULT (AFTER 5 TIMES REPETITION)

Date Set	Times Taken in Seconds
10,61,421 CDRs in MSC side	Median : 33 , Max : 34, Min : 33
3,53,336 CDRs in MSC side	Median : 13, Max : 13, Min : 13

The tests were repeated 5 times on two different datasets of different durations. As can be seen, the solution can process more than 30,000 CDRs per second. Anomaly or false alerts due to multiple calls by same parties in the tolerance threshold interval are taken care of through some post processing as detailed bellow

The post processing after join cannot be performed by mere use of the Infosphere Streams built-in operator (operator called Functor) for following reasons:

- The Functor operator allows history access or past record access only through numeric literals. Variables cannot be used for history access.
- State variables cannot be used in all part of Functor operator.

To overcome the problem of performing the required post processing using built-in operators, a user-defined operator (UDOP) was written using C++ interface of Stream Processing Language.

For the scenario described above, all the MSC CDRs that are joined against the same billing CDR are present sequentially. For all the joined records in the same sequence the calling party number, called party number, and the Billing CDR call start time are the same. One joined record in this sequence is a valid join and the other invalid joined record(s) in the same sequence needs to be discarded.

For multiple joined records with same calling party number, called party number, call start time, the post-processing component selects the one where the difference in call start timebetween MSC and Billing CDR is the minimum. However if the MSC CDRs has already been joined with another Billing CDR, then that that MSC CDR is not considered by post-processing module. As can be seen in Table III, it takes 13 seconds to process 1 hour data and 33 seconds to process 3 hours' data. This happens because sufficient number of MSC CDRs are filtered and made available before ingestion of Billing CDRs in the batch processing mode. As the CDR volume grows, the impact of this delayed injection of billed CDRs is significantly minimized in the case of offline processing mode.

In case of real-time processing, the MSC window does not need to hold N hours' CDR data. Assuming there is a processing delay of maximum of t seconds between MSC CDRs and Billed CDRs, then MSC window of $\mathbf{t + t_2 + \delta}$ seconds will be sufficient for real-time processing of CDRs where δ is the maximum time required for processing t seconds' CDRs in the stream processing platform and t_2 is if any optional tolerance threshold is required between MSC CDR and corresponding Billed CDR. The Billed CDR window will stay as it is i.e. a count-based window of size 1.

VIII. FURTHER WORK

The anomaly detection logic can be broken up and program flow can be updated to fix different parts of the logic to run on different cores of the host server. The approach is called **core-pinning**. Subsequently the performance needs to be compared with and without core-pinning.

For even more prompt processing, multiple host nodes can be placed in a cluster and thus the processing can be distributed across multiple hosts. Infosphere Streams allows seamless addition and removal of host in the computing cluster. The processing can be distributed across the host in different ways [13]:

- Split the stream of CDR data into different sub-streams and process different sub-streams on different host in parallel.
- Divide the application logic into components, which can be processed sequentially in a pipe-line fashion, and perform different component on different host.

The scenario can be extended to include other type of services in addition to pre-paid voice call. Other type of pre-paid service, services used by hybrid subscribers and even the post-paid services can be brought under scope of near-real-time

 ISBN: 978-1-61208-193-9

reconciliation. With multiple data streams for different services running on InfosphereStreams computing cluster, it will be useful to segment and position the data streams judiciously on the hosts of the cluster to achieve optimum load balancing.

The current work has been tried on a limited set of CDR data. This work and its proposed extension can be run on various CDR data streams of different durations and also on archived and real-time CDR streams. The process of running the application on different data sets and scenarios will provide more insight into the solution domain.

In the deployment scenario where a huge volumes of CDRs needs to be reconciled in a relatively small time window, Hadoop Map-Reduce based approach can be considered [14]. Apache Hadoop is open-source and it can be implemented on large clusters of multiple commodity servers. It scales linearly to handle huge data by adding more nodes to the cluster transparently. This approach is useful for scenario which require very high throughput, but does not need very low latency and is suitable for batch processing operation, whereas the Infosphere Streams base clustering is useful for real-time on the fly stream processing.

The discussion so far in this section has focused on how the current work can be enhanced or extended. However the techniques of this work, i.e., stream-based real-time analytic processing, can be applied in other domains also. Low latency processing of CDRs or other kind of records can be re-used in various other kinds of applications like capacity management, traffic analysis, user trending, Quality of Experience (QoE) metric collection etc some of which are outline below:

- Churn Management - Monitor and correlate key attributes from different sources like Customer Relationship Management (CRM), Tariff Plan, Provisioning, Mediation, Billing, Network Switch to predict probability of churn before it occurs
- Campaign Management - Real-time analytics to deliver right message to right customer segment/prospects at right time and in right place
- Network Traffic Monitoring - Real-time correlation of traffic data with other sources like CRM, historical Usage Summary to set differentiating priority for traffic and optimize network usage for greater profitability.

IX. CONCLUSION

Stream processing platforms are ideally suited for real-time or near real-time processing. They can also be used for complex event processing where one or more real-time data streams are correlated with historical archived data and look-up information to arrive at actionable intelligence on the fly. However with careful consideration, applications can be developed using stream processing which can be used in both real-time and offline batch processing mode, just by providing different input parameter and without requiring any program logic modification. Applications like revenue assurance are currently done by many operators in offline batch processing mode, however they can greatly benefit by being done in real-time. This dual mode application can provide a smooth transition path from batch processing to real-time processing mode. It can validate the processing logic with off line data before deploying it for online real-time analysis.

During real-time analysis, if any problem is found in the processing logic which is found out due to some kind of new or unforeseen data in the field, then the application can be modified to correct the processing logic, and the same application can run in offline mode on the archived data.

ACKNOWLEDGMENT

The authors thank Mr Rajat Garg and his team from Tata Consultancy Services Ltd (TCS) telecommunication business unit for their active support and valuable input during this work.

REFERENCES

[1] The World in 2011 ICT Facts and Figures, ITU Telecom World 2011, http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf, <retrieved: Feb. 2012>

[2] Vodafoneannual report 2010 : http://www.vodafone.com/content/dam/vodafone/investors/annual_reports/annual_report_accounts_2010.pdf (Mobile penetration : pp. 6-6), <retrieved: Feb. 2012>

[3] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin J. Strauss, "Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries", AT&T Labs-Research

[4] Corinna Cortes and Daryl Pregibon,"Signature-Based Methods for Data Streams", AT&T Shannon Labs, Florham Park, New Jersey, USA

[5] Munir Cochinwala and Euthimios Panagos, "Near Real-time Call Detail Record ETL Flows", Telcordia Applied Research.

[6] Aniruddha Mukherjee, Punit Diwan, Prasun Bhattacharjee, Debnath Mukherjee, and Prateep Misra, "Capital Market Surveillance using Stream Processing", Tata Consultancy Services Ltd.

[7] Matthias Bröcheler, V.S. Subrahmanian, and Andrea Pugliese,"DOGMA: A Disk-Oriented Graph Matching Algorithm for RDF Databases", University of Maryland, USA and Universit`a della Calabria, Italy

[8] Matthias Bröcheler, V.S. Subrahmanian, and Andrea Pugliese,"COSI: Cloud Oriented Subgraph Identification in Massive Social Networks", University of Maryland, USA and Universit`a della Calabria, Italy

[9] Miran Dylan, "An Analysis of Stream Processing Languages", Department of Computing, Macquarie University, Sydney, Australia

[10] http://www.smartercomputingblog.com/2011/06/10/ibm-infosphere-streams/ , <retrieved: Feb. 2012>

[11] "IBM InfoSphere Streams: Programming Model and Language Reference", Version 1.2.1.

[12] Roger Rea and Krishna Mamidipaka,"IBM InfoSphere Streams : Enabling complex analytics with ultra-low latencies on data in motion", IBM Software Group

[13] Chuck Ballard, Daniel M Farrell, Mark Lee, Paul D Stone, Scott Thibault, and Sandra Tucker, "IBM InfoSphere Streams Harnessing Data in Motion", First Edition, http://www.redbooks.ibm.com/abstracts/sg247865.html, <retrieved: Feb. 2012>

[14] Tom White, "Hadoop: The Definitive Guide", O'Reilly Media, Inc, June 2009: First Edition

 ISBN: 978-1-61208-193-9