DOI: 10.37943/AITU.2020.95.28.005

A. Zhumekenov
Master's student of Information Technologies
zhumekenov@gmail.com, orcid.org/0000-0002-2904-0327
Kazakh-British Technical University, Kazakhstan

COMPLEX EVENT PROCESSING APPROACH ON SUBSCRIBERS' DATA OF TELECOM OPERATOR

Abstract: Nowadays the usage of mobile phones has reached extremely large worldwide proportions and is increasing dramatically. There is a stronger need to decrypt the important information that is hidden among them. Even all required information is gained, processes of companies remain static and can not be changed dynamically to adapt to actual business needs, reducing the advantages that can be achieved. Every second millions of raw information are being generated by mobile users, which handled by Telecom operators in data servers. By using Complex Event Processing (CEP) approach in real-time, we can obtain the information that really matters to our business and use it to monetize the vast amount of data that is being collected through mobile phone usage. In this paper, we present an internally developed framework that combines the strengths of CEP and business process implementations which, allows us to react to the needs of today's fast-changing environment and requirements. We demonstrate 3 simple use case scenarios to show the effectiveness of the CEP approach in our situation. The importance of implementing the CEP approach on subscribers' data should not be overlooked as means of trying to capitalize on new services, however, have to be considered as a challenge to give subscribers the opportunity to get more customized offers and services.

Key words: Complex Event Processing, Telecom Data Analysis, Information Processing, Targeted campaigns.

Introduction

In 2019 the mobile phone user database has reached almost 70% of the global population with more than 5 billion devices in use [1]. Given the dynamicity of today's business environments, there is a need to continuously adapt and keep up to date the business processes in order to respond to the changes in those environments and keep a competitive level among Telecom operators and solution providers. One of the main concerns for Telecom operators' applications is to handle and interpret online raw data. By using the CEP approach and developed framework we can facilitate the solution of this problem by gathering needed information in real-time about the subscribers' different events on the Telecom network in order to determine the necessity of making specific offers and suggesting new promo.

The Real-time data processing systems are used widely to provide insights about events as they happen. Many companies have developed their own systems: Twitters Storm and Heron, Googles Millwheel [8], LinkedIns Samza and Facebooks´ Puma, Swift and Stylus [7, 9-11].

One of the main challenges for processing data dynamically is the size and the speed of incoming data at which it is being generated. Every second, phone mobile users generate millions of data and entire traffic allocates in Telecom operator's data centers. Despite having a common goal, solutions based on CEP differ in a wide range of aspects, including architectures, data models, rule languages, and processing mechanisms. In part, this is due to the fact that they were the result of the research efforts of different communities, business

Scientific Journal of Astana IT University
ISSN (P): 2707-9031
ISSN (E): 2707-904X

market demands, each one bringing its own view of the problem and its background for the definition of a solution [3].

In the paper an internally developed framework is presented that combines the advantages of the CEP and dynamic business process adaptation, which allows us to respond to the needs of today's rapidly changing environments. Sub- scribers information for different business-specific use cases and triggers is handled such as balance information, mobile internet usages, the location of the device to enrich data about subscribers, and give deep insights on user's interest, lifestyle patterns in a given time period. The picture below illustrates our main idea of implementing the CEP approach on Telecom operator's data traffic.

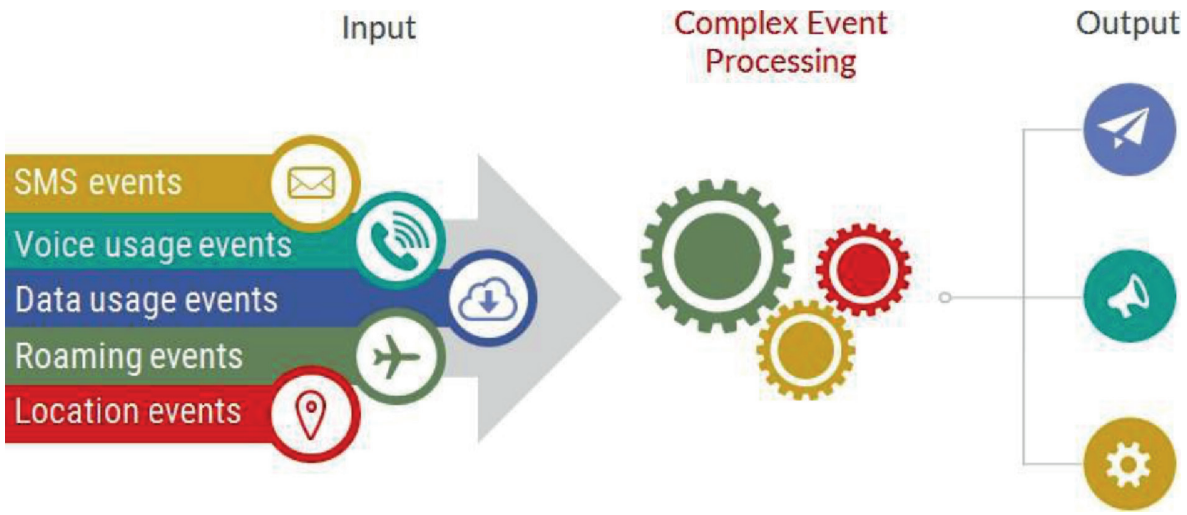

Fig. 1. Main idea

The framework

Before we start our journey on solution architecture it is important to say about motivations, limitations and business/technical requirements that were faced.

Firstly, we need to mention current limitations of legacy system where transactions per second (TPS), in our case Events Per Second is more precise was limited by 2000. Taking into account that a subscriber base reaches 10 million and active subscriber base is almost 6 million that threshold is insufficient to cover and handle all network events in real-time. Besides, the system was consuming all resources all the time which caused performance issues and several critical incidents on the system. Analyzing the data traffic throughout (22.5 TB/month), dynamicity of business requirements, and capabilities of the new solution, challenged us to handle at least 165 000 events per second.

Secondly, from the technical perspective our approach was to create one job that will run all triggers and business rules. By this approach, we should benefit that all triggers and business rules will run effectively and manage rules without restarting the system.

Our proposed solution uses a component-based architecture, where each component plays a critical role in whole system. Fig. 2 on page 2 illustrates high level overview of the system. On the left, the ingestion layer is responsible for fetching and receiving raw data from different data sources (SMS events, Voice usage, Internet usage, Location-based triggers) and saves it into the Events hub. Then the data is consumed by the event execution engine, which in turn, based on current trigger definitions, outputs notifications to the Events hub. Finally, the outgestion layer fetches the notification and dispatches them to appropriate subsystems. All main components in the architecture will be described in the next sections to give you a more detailed overview of the solution.

DOI: 10.37943/AITU.2020.95.28.005

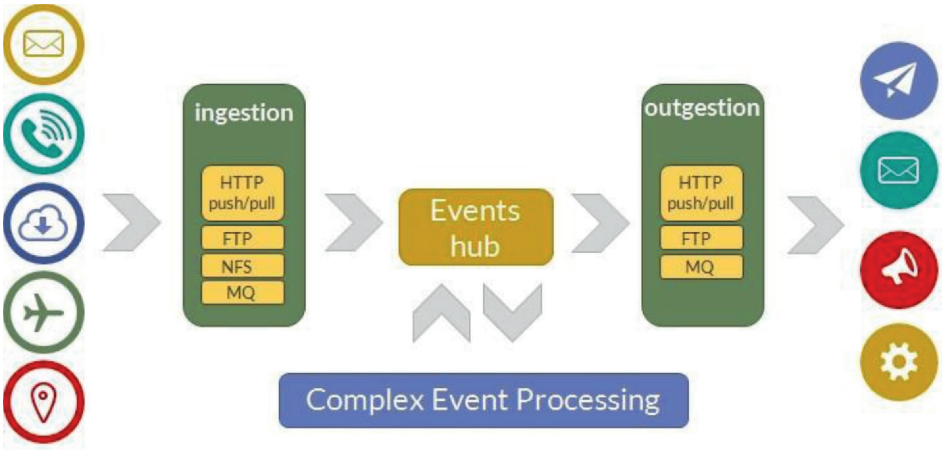

Fig. 2. High level architecture overview

The ingestion

The process of collecting data from various data sources, preparing it, and saving it for complex processing and generating triggers. Currently, there are several different approaches to how data ingestion pipeline can be built:

The first approach consists of two steps. Firstly, the raw data is fetched from a data source and is a subject for some preliminary processing, e.g. splitting a huge file containing multiple events. After that, the data is being transformed, serialized, and finally pushed to Events hub, where it waits for consumption by the event execution engine. This approach is preferred whenever the event transformation is complex or the data volume is too big to be processed in the ingestion layer.

The data is fetched/received, transformed, and serialized within the same pipeline.

The raw data is immediately pushed to the Events hub. Transformation takes place in the execution engine.

It is worth mentioning that both intermediate results and transformed events are stored in Events hub topics. Each event type (more precisely, events originating from the same data source) is kept in a separate topic. Despite which ingestion approach suits better, we need to notice what file types and protocols can be ingested in our system. To illustrate how various file types are fetching and transforming to one input format, see the Fig. 3 on page 2.

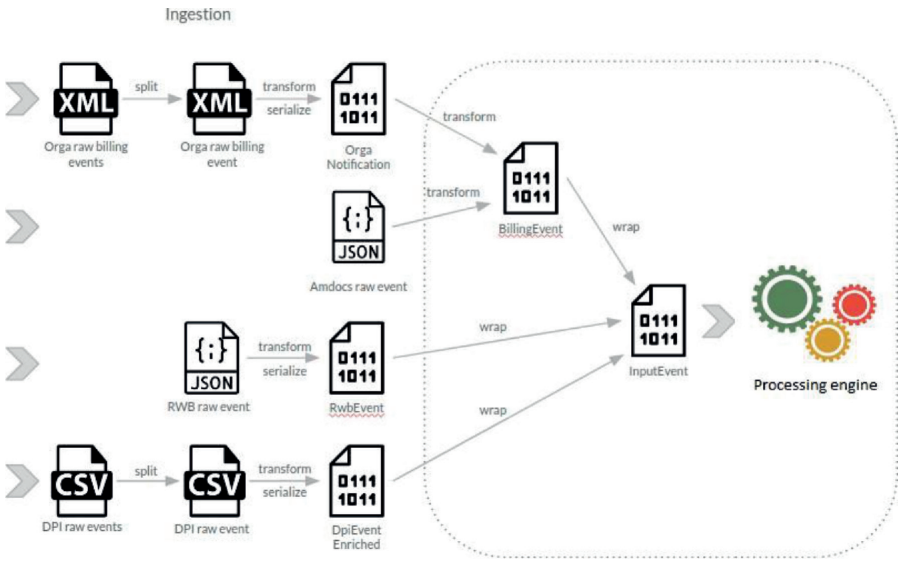

Fig. 3. Ingestion file types

Scientific Journal of Astana IT University
ISSN (P): 2707-9031
ISSN (E): 2707-904X

Events Hub

It is a system that keeps all the in-flight data streams in a reliable and persistent way. The data in an Events hub topic is divided into partitions. The partitions are distributed over the servers in the cluster with each server handling data and requests for a share of the partitions. Each partition is replicated across a configurable number of servers for fault tolerance. For a topic with replication factor N, the Events hub tolerates up to N-1 server failures without losing any records committed to the topic.

The processing engine

It is an execution engine for complex event processing applications. In this component, streaming applications are implemented that consume all the prepared events by the ingestion system and apply trigger procedures defined by business users. The provided management functionalities over the CEP applications, like define, start/stop of the application. Execution jobs are stateful. In order to make state fault- tolerant, the processing engine makes periodic checkpoints of the state. Checkpoints allow to recover state and positions in the streams in case of a program failure (due to machine-, network-, or software failure). Any records that are processed as part of the restarted job are guaranteed to not have been part of the previously checkpointed state. In other words, each event from an input stream affects the data flow state exactly once. However, it does not mean each event is processed only once. More precisely, when the state and input streams' offsets are restored, the processing engine needs to replay the events that have been processed between the last checkpoint and the failure. In consequence, for instance, some notifications might be generated twice. To illustrate how state and checkpoints are working, please see figure 4 below.

Fig. 4. Checkpoint and failure

The outgestion

The process of communicating with external layers based on processed data. Outgestion layer is pretty straightforward, all Processing job's notifications are stored in a single Events hub topic. Then the outgestion pipeline reads them, transforms to the format-specific for the destination system, and pushes them there.

DOI: 10.37943/AITU.2020.95.28.005

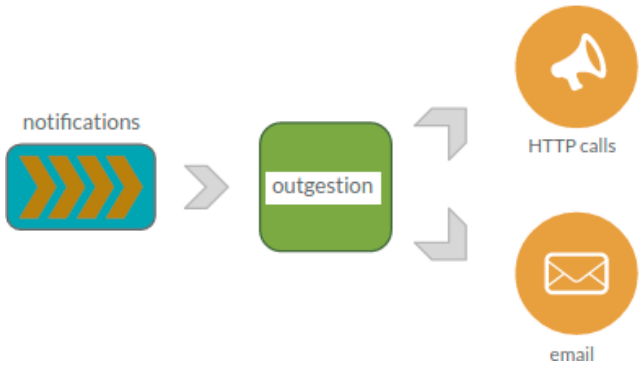

Fig. 5. Outgestion

Technology stack

Regarding the technology stack that was used, take a look on the list below, all components are based on a free and open-sourced Apache product:

- Apache Flink – event processing engine. Provides flexible event time support. Fault-tolerant, high throughput and low latency.
- Apache Hadoop – a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. In the provided solution, Hortonworks distributive is using.
- Apache Kafka – central events hub, where all events come through. Simple and reliable. The ingestion system saves data to it, the outgestion system is reading data from it for sending it further. It is the only component that communicates with Apache Flink directly.
- Apache NiFi – engine for managing ingestion and outgestion pipelines. NiFi collects data from data sources and transfers them to Apache Kafka. With NiFi it is possible to deploy different servers, for example, HTTP, FTP, etc.

Use cases

In this section, there would be described top three business use cases, which could be easily configured in the CEP and give win-win results when the whole system starts working.

Balance top-up – A subscriber top-ups his/her balance too often in short period of time. We can offer him/her a less expensive tariff or auto-payment services.

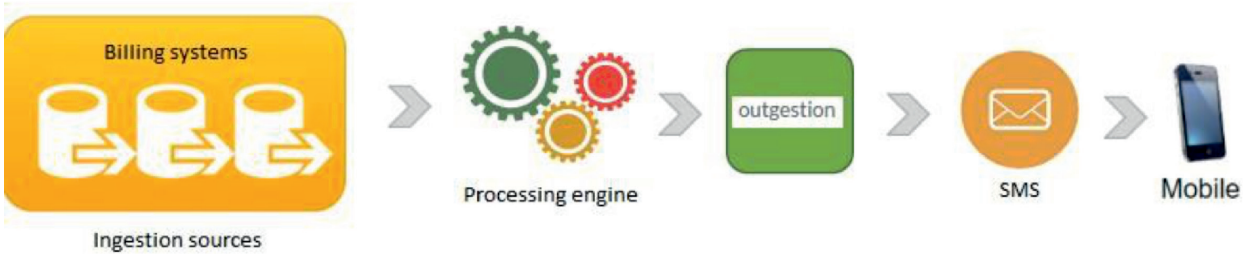

Fig. 6. Balance top-up

Billing systems data sources are ingested and can track subscriber's balance activities. CEP is able to capture certain pre-defined criteria (triggers). When any subscriber makes a top-up to his balance, CEP captures that and sends a request to notify the user of the balance after top-up via SMS Center.

Fraud detection – Sends an email to the anti-fraud unit if subscriber registered in roaming, however, his balance now is equal to 0.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031
ISSN (E): 2707-904X

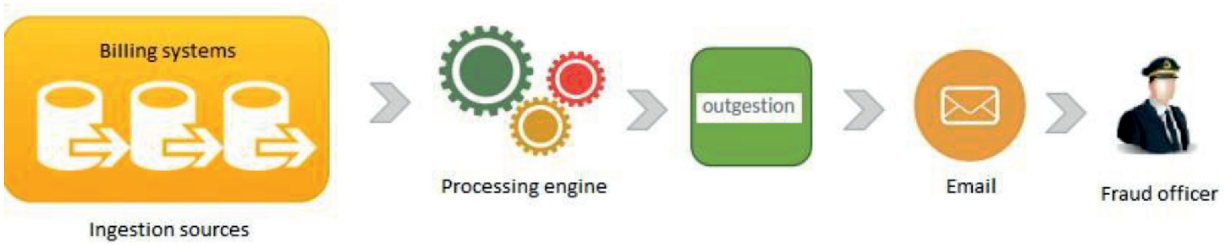

Fig. 7. Fraud detection

Extra bonuses – Top-up subscriber balance with extra SMS/Voice usage/Internet usage bonuses when the subscriber fulfill all conditions at promo campaigns.

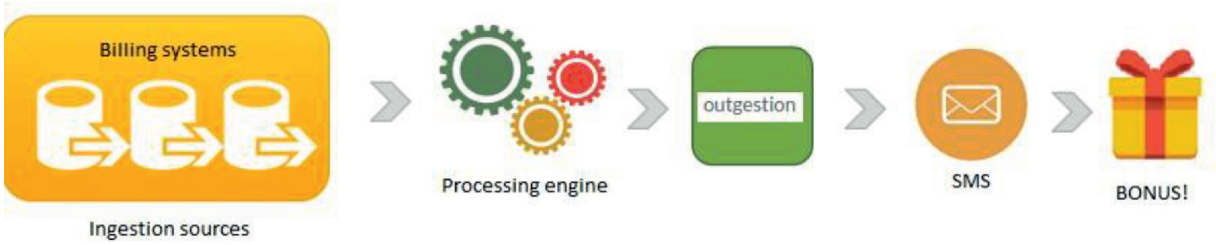

Fig. 8. Extra bonus

Conclusion

In this paper, the solution of how the CEP approach could be implemented in Telecom operator is presented, which helps to solve many technical limitations of our legacy systems and business requirements issues, process, and analyze the vast amounts of subscribers' data in near real-time that was impossible before. Furthermore, there were have been defined triggers and business rules which could give a win-win effect after project launches. Apache components that are typically used in our architecture provide the possibility to build the CEP ecosystem, which will play a vital role in the next decade for the Telecom operator business. Ongoing work will focus on further optimization and improvement possibilities, some of the directions in which this work can be improved are:

One of our approach was to build one job to run all triggers/rules. However, any coin has a reserve side, and in our case, one bad rule could affect to all other triggers/rules impacting on a whole system. That is the main challenge that should be addressed in future works.

Response latency time will be always actual and crucial issue, due to the dynamics of today's business environments and requirements. We need to pay attention to how effectively ingestion and outgestion pipelines are executing the data traffic.

Due to the data traffic is extremely huge and grows dramatically, the challenge is to continuously write and store data in Data Lake/Data Warehouse and implement an Online Analytical Processing engine for analytical purposes and reports. Separating written data from data traffic should help to avoid performance and sustainability issues.

DOI: 10.37943/AITU.2020.95.28.005

References

1. Digital 2019: Global digital overview (2019). https://datareportal.com/reports/digital-2019-global-digital-overview
2. Hermosillo, G., Seinturier, & L., Duchien, L. (2010, July 5-10). Using Complex Event Processing for Dynamic Business Process Adaptation. IEEE International Conference on Services Computing, Miami, FL, USA. https://doi.org/10.1109/SCC.2010.48
3. Cugola, G. & Margara, A. (2012). Processing Flows of Information: From Data Stream to Complex Event Processing. https://doi.org/10.1145/2187671.2187677.
4. Cao, H., Dong, W.S., Liu, L.S., Ma, C.Y., Qian, W.H., Shi, J.W., Tian, C.H., Wang Y., Konopnicki, D., Shmueli-Scheuer, M., Cohen, D., Modani, N., Lamba, H., Dwivedi, A., Nanavati, A.A., & Kumar, M. (2014). SoLoMo analytics for telco Big Data monetization. IBM Journal of Research and Development. https://doi.org/10.1147/JRD.2014.2336177.
5. Ottenwalder, B., Koldehofe, B., Rothermel, K., & Umakishore, R. (2013, June). MigCEP: Operator Migration for Mobility Driven Distributed Complex Event Processing. DEBS '13: The 7th ACM International Conference on Distributed Event-Based Systems Arlington Texas USA.
6. https://doi.org/10.1145/2488222.2488265.
7. Stonebraker, M., Cetintemel, U., & Zdonik, S. (2005). The 8 Requirements of Real-Time Stream Processing. ACM SIGMOD Record, 34(4). https://doi.org/10.1145/1107499.1107504
8. Samza. http://samza.apache.org.
9. Akidau, T., Balikov, A., Bekiroglu, B., Chernyak, S., Haberman, J., Lax, R., McVeety, S., Mills, D., Nordstrom, P., & Whittle, S. (2013). Millwheel: Fault- tolerant stream processing at internet scale. PVLDB, 6(11),1033–1044
10. Kulkarni, S., Bhagat N., Fu, M., Kedigehalli, V., Kellogg, C., Mittal, S.J., Patel, M., Ramasamy, K., & Taneja, S. (2015). Twitter heron: Stream processing at scale. In SIGMOD, 239–250.
11. Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., Bhagat, N., Mittal, S. & Ryaboy, D (2014). Storm@twitter. In SIGMOD, 147–156.
12. Chen, G., Wiener, J., Iyer, S., Jaiswal, A., Lei, R., Simha, N., Wang, W., Wilfong, K., Williamson, T., & Yilmaz, S. Facebook, Inc. Realtime Data Processing at Facebook
13. Luckham, D. C. (2002). Addison-Wesley Longman Publishing Co., Inc., (2001). The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. (1st edition). Addison-Wesley Professional.