ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

Real Time Monitoring, Alerting and Anomaly Detection Analytics Platform Using Reactive Programming

[1] Jagreet Kaur, [2] Dr. Kulwinder Singh Mann
[1] Research Scholar, IKGPTU, Jalandhar, India.
[2] Professor and Head of the IT Department, GNDEC, Ludhiana, India

***Abstract*— The IT world is moving towards innovation and technology. The growth of data is exponentially increasing in various domains like healthcare, Iot, Biometrics and many more. With periodic batch processing data, it is not possible to provide the required information to take instant decisions. Streaming data analytics is the bloodstream of modern applications. The traditional cloud-based storage model is giving way to do in-memory analytics processing of big data streams. There are many domains where real-time processing of data is used for taking timely decisions that can minimize the risks of human lives and resources, enhance the quality of human lives, increase efficiency of resources management and proficiency, etc. Therefore, Real time Data is required in every field. This brings the requirement of Real time Analytics Platform . Adaptation of Data First Approach is needed for Data-Driven applications to address the many issues like removal of Data Silos to create Single Integrated Platform, Complex Data Governance, analytics too time consuming and expensive, High cost on current systems and enabling real time analytics. In this Paper, we will discuss Real Time analytics platform based on Reactive Machine learning ,Functional Programming and Kappa Architecture for Monitoring, Alerting, IoT Based applications or event Driven applications.**

***Index Terms*— reactive machine learning, linked Analyses, Kafka, Nifi, Flink, Druid, Reactive Programming.healthcare,IoT.**

1. INTRODUCTION

In this changing world, Performing analytics in real time becomes a highlighted aspect. So, the concept of real time should be clear. Real time data is the streaming of data having a tight deadlines in terms of time [1]. We normally consider that if our platform is able to capture any event within 1 ms, then we call it as real-time data. This can be explained with an example. Let us suppose an application is running in the system. This running process is an event. Rather than querying explicitly from the system, the system will automatically notify to other system about the event and push the relevant data of event to other systems. This whole procedure is known as Real Time Processing.
Real time data is used in every domain. Many organizations are collecting large amount of data regarding their products, services or even about their organizational activities like tracking employees activities through various methods used like log tracking, taking screenshots at regular intervals. When we have to work in log analytics, fraud detection or real-time analytics, healthcare analytics, the data should be act upon at that instant only when it receives. Therefore, processing of huge volumes of data is not enough. There is a need to process them in real-time so that any organization can take decisions immediately whenever any important event occurs. This is required in Intelligence and surveillance systems, fraud detection, Alerting Platforms, etc.

The performance of query also plays an important part in IT world domain. The time interval from the submission of query to the required service to receive the required response should be less than 1ms. This feature become possible by the use of Real Time Processing [2]. With the use of efficient performance of query it becomes possible to trace the applications. This query performance is dependent upon response time and throughput of the application.

The data that is produced continuously like mobile applications, web clickstreams, application logs and IoT sensors need to monitor [3]. Recent data is highly valuable in decision making if act on it in time but the analysis of historical as well as as recent data can helps to take a better decisions. The key requirement to stream real time data are correctness, durability, reactivity and reliability [4]. Sometimes, real time data is also generated in queues like a resource is shared among multiple consumers or data is transferred asynchronously between two processes [5]. So, there is a need to design a system that receives events, archives them, performs offline and real-time computations, and merges the results of those

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

computations into coherent information. All of this should be happened at the scale of millions events per second.

Stream is said to have unbounded data that is carried sequentially from a set of producers to a set of consumers. This means that the data coming from infinite sources is sequentially flows from set of producers to set of consumers in the form of topics. Producers and consumers works independently and do not have awareness about each other rather they share topics with each other. This complete process is known as publish-subscribe.

Queues is one of the traditional system in which computation is performed individually per message. But in case of stream processing multiple inputs and records are processed simultaneously. In traditional messaging system one cannot go back, but this issue is resolved by stream processing. For example kafka is a pull based model for stream processing that keeps the messages for a configurable amount of time. This allows consumers to "go back" and consume messages multiple times or if a new consumer is added, it can read the complete history. This makes stream processing possible, because it allows for more complex applications

Event driven applications like alerts can automatically generated in case memory of CPU is full or a loan application is accepted or rejected. To handle event driven applications effectively, reactive machine learning is very helpful[6]. Generally, a one main loop of events are created, and whenever any event occurs that main loop will listen and react to that event by using callback function. Events are delivered in the order they are received, like a queue. Unlike with a queue, events are persisted even after they're delivered. There are many Event driven applications such as Failure alerts, Ad Optimization, Real-Time Fraud Detection, Real Time Application and network monitoring, Web Personalized Offers and many more.

The unusual behaviour pattern of the data needs to consider to detect the presence of an error in the system. The system produces log that contains the information about the state of the system. Outiers can be detected by analysing the log data so that security of the system can be protected [7].

In this paper a approach is introduced to perform anomaly detection and method for prevention. Detection of anomaly is not sufficient. The evaluation behind the occurrence of anomaly is also an important aspect. So, Linked analysis is used to find the cause of anomaly within the data. It is a technique that finds the relationships between the nodes or objects. Next approach is preventive maintenance of data from anomaly. To implement it reactive machine learning is used that works smartly and provides response in timely manner.
This paper is organised into three sections. Section 1 describes introduction. Section 2 highlights the concept of Reactive programming and Reactive machine learning. Section 3 enlighten about the detail structure and working of architecture.

2. REACTIVE PROGRAMMING

Reactive programming is a programming with asynchronous data streams. Reactive programming manages asynchronous data flows between sources of data and components that need to react to that data. Programming paradigms that help you build ***Reactive Systems*** [8]. In other terms, Reactive programming is about registering callbacks to an event to allow callback executions whenever a concerned event is occurred. This optimized usage of system resources such as threads, memory, CPUs etc.
It decomposes problem in
to multiple discrete steps where each can be executed in an asynchronous (move on to another task before it finishes) and non-blocking fashion, and then can be composed to produce a workflow possibly unbounded in its inputs or outputs. Reactive programming is related to data flow since the emphasis is on the flow of data rather than the flow of control.
Reactive programming is a great paradigm for managing internal logic and dataflow transformation,as a way of optimizing code clarity, performance and resource efficiency and utilization.

2.1 Why Should Adopt Reactive Programming

1. We can divide the system into multiple monoliths/ micro-services/ components that are going to communicate with each other using messages.
2. To build a scalable system, we can handle large amount of data or large numbers of user requests and react to the ever-growing data.

ISSN (Online) 2394-2320

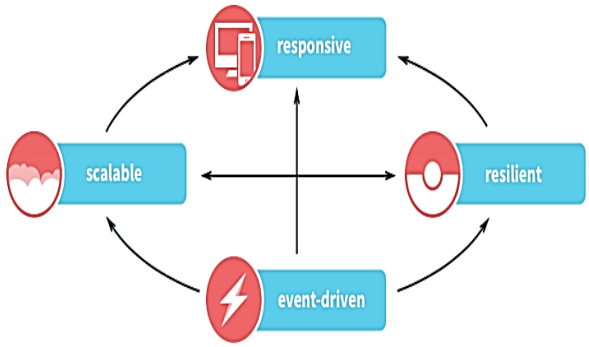
International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

3. Handles failures/ errors will make the system more fault-tolerant.
4. Responsive system is fast and timely available.

Four Reactive Principles :

- Responsive
- Resilient
- Scalable
- Message-driven

Reactive application are built on four guiding principles. Systems built as Reactive are more Scalable, Resilient , loosely-coupled and flexible. This makes them easier to develop. Reactive systems are significantly more tolerant to failure and when failure does occur they meet it with elegance rather than disaster. Reactive Systems are highly responsive. Responsiveness is impossible to achieve without both scalability and resilience.



Responsive : The System responds in a timely manner . Problems may be detected quickly and deal with it effectively . A responsive system is quick to react to all users. quickness under various conditions, such as failure of an external system. Responsive systems provide rapid and consistent response times , establishing reliable upper bounds so they deliver a consistent quality of service . This consistent behavior builds end user confidence , and encourages further interaction.

Resilient : System is resilient if it can recover quickly from failure. The system stays responsive in the face of failure. It means that application stay responsive even in case of any failures. reacting to failure will make the system more fault-tolerant. system that is not resilient will be unresponsive after a failure . Resilience is achieved by replication, containment, isolation and delegation. Failures can be software , hardware or connection failure. Failure are contained within each component of the system , in isolating system component independent from each other. when any part of the system can fail recover without compromising the system as a whole . Replication reduces the risk of data loss, and impact of failure on the availability of retrieval and storage of information .

Scalable : The system stays responsive under varying workload. Scalable is about responsiveness under varying workload meaning that the throughput of a system scales up or down automatically to meet varying demand as resources are added or removed. System is scalable if it is able to expanded according to its usage. scalable system is easily upgraded on demand or varying workload in order to ensure responsiveness under various workload .

Message-driven : Reactive Systems rely on asynchronous message-passing. To construct a boundary between components that ensures loose coupling , isolation and location transparency . Location transparency makes it possible to work within a single machine / node or across a cluster . Non-Blocking communication allows recipients to only consume resource while active, leading to less system overhead.

2.3 Need of Reactive Programming and Functional Programming

Reactive programming is programming with asynchronous streams of data. For example, facebook feed would be a data stream, click events, user inputs, caches, data structures. User listen to that stream and 'react' accordingly. So, reactive programming involves programming with flow of time and computational events. It also focuses on multiple events and tasks can be completed at same time as code is asynchronous in nature. It also focuses on managing the synchronisation of states between multiple events.

The idea behind functional reactive programming is to model things like user input in a more declarative way by making their behavior more specific. Functional reactive modelling models change over time by introducing two data types 'events' and 'behaviors'. Events represent a value at a particular time whereas behavior represent values that vary continually over time. When these concepts are combined in a functional way, the entire program becomes a combination of events and behaviors

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

[10]. The use of functional reactive programming raises the level of abstraction of code, so need to focus on interdependence of events that define logic, rather than focussing on implementation details.

2.4 Need of Reactive Machine Learning in Prevention of Anomalies

AI, machine learning, and mathematics provide a powerful combination that changes the endpoint protection scenario. The previous reactive approaches are being replaced with a new approach, where prevention is the method for protection [9]. The predictive and preventative techniques used by AI can be applied across platforms, operating systems, file types and devices. The machine learning systems can very well recommends detections and generate alerts but the expectation is that the system will stay return predictions despite failure or changes in load. These requirements can be hard to achieve in machine learning. So, Reactive machine learning solves various problems. In reactive machine learning, Events from the outside world are sensed and turned into facts which can be further used to produce features [11] and labels using feature generators. The learning models can be learned from implementation of either external or internal learning algorithms. After the evaluation of the model, the model is published. To make the model reactive, the model should follow the reactive strategy like replication, containment and supervision. The reactive model can replicate ie execute the same component at more than one place at a same time. To prevent the failure of single component without affecting the other components, reactive model can use containment. Containment can be implemented using different systems. Lastly, reactive systems should use the strategy of supervision to organize components.The strategy of supervision gives a point of control, where the reactive traits are being achieved by the actual runtime behavior of the system.

Systems built as Reactive Systems are more scalable, loosely-coupled and flexible. This makes developers easier to develop and amenable to change. They are more tolerant of failure and when failure does occur they can handle with elegance rather than disaster.

2.5 Requirement of Interactive Analytics and Druid

Interactive analytics is an extension of real time analytics. It provides the facility to run complex queries using complex data landscapes where we have the complexity intelligence to visit thousands of nodes in real time [12]. Druid is a distributed database that stores the data column wise i.e. column-oriented database. Today users need that application should posses low query latency and druid takes less than a second to execute the query. It also helps to store the large amount of historical metrics and can ingest high velocity of data [13].

3. PROPOSED KAPPA ARCHITECTURE OF EVENT/DATA DRIVEN APPLICATIONS

Kappa Architecture: it is an architecture that makes possible to process the data in real time. It simply removes the batch processing layer and introduces Streaming Processing Architecture for the processing of Streaming data.

As shown in fig1 the components of Architecture are Data Sources, Apache NiFi, Apache Kafka, Apache Flink and Druid. Each component is explained below:

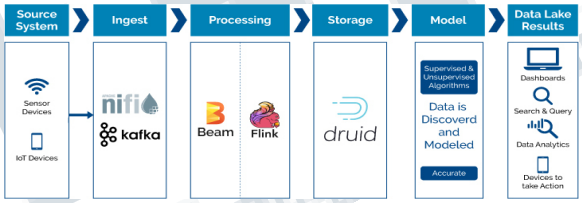


Fig 1. Components of Architecture

●***Data Sources:*** it is the first component of architecture. These sources are responsible for providing the data. Various data sources are available such as API (Application Program Interface), databases, Agents and Files.

API is consist of HTTP protocol which is responsible for sending the data to the data collector. Next is databases, various types of databases are available such as RDBMS i.e. Relational Database Management System that stores the data in rows and columns. Another database is NoSql. It is used to store the unstructured type of data. Next comes Agents, they are used when there is no direct way to collect the data from data source. So, an agent is used that will run on that framework and sends the data to the data collector. Lastly, files can directly provide the data to the data collector in textual format.

●***Apache NiFi:*** it is an powerful and reliable system to receive and process the data from various data sources. It is a UI platform which receives the data from defined source and stores the information to defined destination.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

Here, processors are used to pull the data from a source, performs transformation method and stores to the mentioned destination. But, in case of Agents as a data source data is pushed from an agent to NiFi.

●***Apache Kafka:*** it is a fast and scalable publish-subscribe messaging system. It consist of two components such as producer and consumer [14]. The incoming data is written sequentially to the disk. Firstly, the producer will publish the data or messages to topic (user defined category like Apache NiFi) and the consumer such as Apache Spark, Apache Flink, etc will subscribe the messages to perform further processing of data.

●***Apache Beam:*** Apache Beam is an open source model that is used for defining both batch and streaming processing pipelines. It is basically used for parallel data processing in the which the data is divided into various small chunks and they are processed independently and in parallel form also. Apache Beam consist of runners that translate the pipeline of data into the defined in the beam program. To run the Apache Beam program appropriate runner has to defined which we want to execute in the pipeline.

●***Apache Flink:*** it is an open source stream processing framework for data streaming applications. It can handle both in-memory and disk-based processing. Therefore, apache flink is used for both batch and streaming data processing. It supports iterative processing for machine learning algorithms, string management, graph processing and many more.

●***Druid:*** it is an open source data storage that can quickly ingest the large amount of data. It is used for executing queries in low-latency. The main usage of druid is to store the time series data in an aggregate way [15]. Therefore, while making schema of this database there is a need to set minimum aggregation time interval.

3.1 Need of Unified Processing

Unified Process is a framework used for guidance in tasks and projects of the organisation. Each activity is consist of input and output but they can be used differently in different situations. There is a need of this framework as its main objective is to examine which tools is performing what kind of activity, at what time tools will perform a particular activity, what are the inputs and outputs of each activity and how one can reach to the solution.

3.1.1 Selection of Components

Apache Nifi: Apache NiFi is used for both real time and batch processing. It supports both standalone and cluster mode and performs better error handling.

Apache Kafka: in the digital environment, the services are working on wider stream of data. Therefore, performing analytics in real time becomes essential. But, the traditional batch processing not able to solve the problem in real time. Therefore, apache kafka is used to extract the wider amount of data in real time. The role of Apache Kafka is to develop the real time pipelines and streaming applications.

Apache Beam and Apache Flink: they both can be used for streaming and batch processing. They allow to process the data at once only i.e. there is no need to process streaming and batch data separately. Apache Beam is used to maintain the efficiency of the application by providing the excellent portability facility in such a way that it can execute the pipelines on different runners. As mentioned above, Apache Flink consist of Flink ML library. This library provides in-memory data streaming, iterative processing of algorithms and streaming ML exclusively for the processing of streaming data.

Druid: Druid is an open source Data Warehouse and recently it is used highly in analytics industry for advanced aggregation queries like top 5 users by some dimensions (like cpu usage, disk usage etc). So The Main Use Case of Druid is for running low latency aggregation queries over petabytes of data. The underlying architecture stores the data in aggregate way as we define our minimum level of aggregation we required while designing our schema. So Druid is highly optimized for aggregation queries.

3.1.2 Working:

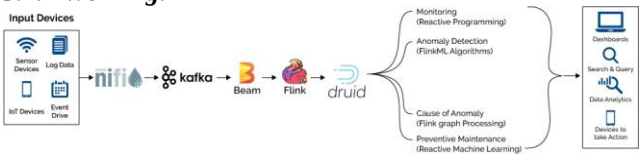


Fig 2. Working of Architecture

As shown in fig 2 we have sensor data or log data. The

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

data is present in the form of time series [16] i.e. the attributes are dependent on time. Firstly, the data from sensor device is collected and stored in the druid database. The flow of data pipeline is made using Apache Nifi. As whole tool cannot be uploaded into the sensor device, therefore it is decided to upload the developed data flow pipeline into MiniNiFi. MiniNiFi is a small part of Apache NiFi itself. After data ingestion, the producer component of Apache Kafka will publish the data in the form of messages to topic. Then, Apache Beam as consumer subscribe to topics and processes the published messages. Apache Beam will execute the Apache Flink runner that allows the Apache Flink to be execute. First, monitoring of data is performed. To do so, rules are defined in the database like mongodb for eg, temperature > 55. Therefore when the data is fetched by Apache Flink, it will compare the values of data with the defined rules and check whether the data will break or cross the rule. If the rules are breaking an event will be triggered. This process is known as reactive programming. All the events are notified through email and are stored in mongodb to maintain the track of notifications.

Next is anomaly detection. After receiving data from Apache Beam FlinkMl library will be used. There are two ways to detect anomaly such as supervised and unsupervised learning. The type of approach used for anomaly detection depends upon the data received by Apache Flink. In supervised learning labelled data is present on the other hand in unsupervised learning unlabelled data is given. Firstly, supervised learning is performed for training data points so that they can be classified into anomalous and non-anomalous data points. But, for supervised learning there should be labeled anomalous data points. The most common supervised learning algorithms are: supervised neural networks, support vector machine learning, k-nearest neighbours, Bayesian networks and Decision trees. In case of k-nearest neighbours, approximate distance between the data points is calculated and then assignment of unlabeled data points is made according to the class of k-nearest neighbour. On the other hand, bayesian networks can encode the probabilistic relationships between the variables. This algorithm is mostly used with the combination of statistical techniques.

Another approach for detecting anomaly is unsupervised learning. One can apply unsupervised learning to train CART so that prediction of next data points in the series could be made. To implement this, confidence interval or prediction error is made. Therefore, to detect anomalous data points Generalised ESD-Test is implemented to check which data points are present within or outside the confidence interval. The most common unsupervised algorithms are self-organizing maps (SOM), K-means, C-means, expectation-maximization meta-algorithm (EM), adaptive resonance theory (ART), and one-class support vector machine.

Next part is finding the cause of occurrence of anomaly [17]. The approach that will be used is graph processing using Apache Flink. Every node in the graph represents the sensor data. The pattern matching score of each sensor data is computed. When the anomalous data is detected, the similar patterns are examined. After that random walk algorithm is implemented over the call graph. Basically, random walk is performed over the call graph depending upon the similarity score. The sensor data is picked up randomly in sequence among the neighbours in the call graph. The pickup probability of sensor data depends upon the relevance of anomaly sensor node. More the visits on a certain sensor by random walk gives that the anomaly on that sensor can best explain the anomalies of all the other sensors. Thus, the probability of visiting each score can be taken as root cause of the sensor node.

Lastly, preventive maintenance has to be performed. The approach used is Reactive Machine Learning. Firstly, as mentioned above monitoring of sensor data is performed using reactive programming. When any suspicious activity is observed, machine learning algorithms using apache flink is used for maintenance. The machine learning algorithms are neural network, Bayesian network and support network machine.

CONCLUSIONS

In this paper we have discussed different ways in which the problem of anomalies has been detected and have attempted to provide an overview of the detailed system architecture on various techniques. The proposed architecture can support for analytics by providing batch and stream computing, extendable storage solution and query management. To achieve more efficient result, there is still a need to handle the data that is growing time

by time at high rate and larger scale with tons of inconsistent data sources. To process data at high rate, concurrency and effectivity is the main issue and the performance of analytics job depends upon memory allocation/deallocation.To build alerting platform, reactive programming can be used. To maintain the analysis logic in both batch and real time layer, it is easy to used kappa architecture as it is very helpful to process data with less overhead and more flexibility as discussed in paper. In the end, we only want to say that to monitor real time data, to generate alerts and to do analytics, we have many open source platforms like Apache Kafka for providing high ingestion rate and Apache Storm, Apache Flink to provide true real-time streaming, processing and analyzing but it all depends on us that how smartly we use these components and use them efficiently to make Data Pipeline Great.

ACKNOWLEDGEMENT

Authors are highly thankful to the department of RIC, IKG Punjab Technical University, Kapurthala, Punjab, India for providing the opportunity to conduct this research work.

REFERENCES

1. LI Zhao, ZHANG Chuang, CHEN Meng-meng, XU Ke-fu.,"SpeedStream: a Real-time Stream Processing Platform in the Cloud" International Conference on Computer Engineering, November. 2014.

2. hao δ, Chuang Z, Ke-Fu X, et al. "A Computing model for Real-Time Stream Processing, Cloud Computing and Big Data", International Conference on. IEEE, 134-137, 2014.

3. Zhengping Qian, Yong He, Chunzhi Su, et al. TimeStream: Reliable Stream Computation in the Cloud. EuroSys, 2013:1-14.

4. Schmidt S., δegler T., Schaller D., et al. Real-time Scheduling for Data Stream management Systems. Real-Time Systems, 2005.

5. Mishne, G., Dalton, J., Li, Z., Sharma, A., & Lin, J.,. Fast data in the era of big data: Twitter's real-time related query suggestion architecture. In Proceedings of the ACM SIGMOD International Conference on Management of Data (pp. 1147-1158), 2013.

6. Abraham, B. and Chuang, A., Outlier detection and time series modeling. Technometrics, pp-241–248, 1989.

7. Abe N., Zadrozny B., A Langford, "Outlier detection by active learning" In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press, New York, 504–5, 2006.

8. E. Czaplicki and S. Chong. "Asynchronous functional reactive programming for GUIs. In Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, pages 411–422, New York, NY, USA, 2013.

9. C. Elliott and P. Hudak. "Functional reactive animation" In Proceedings of the Second ACM SIGPLAN International Conference on Functional Programming, ICFP '97, pages 263–273, New York, NY, USA, 1997.

10. T. Salmon, D. Bhamare, A. Erbad, R. Jain, M. Samaka, "Machine Learning for Anomaly Detection and Categorization in Multi-cloud Environments," The 4th IEEE International Conference on Cyber Security and Cloud Computing (IEEE Cloud 2017), New York, June 26-28, 2017.

11. Li, H., Achim, A., Bull, D.: "Unsupervised video anomaly detection using feature clustering", IET Signal Proc. 6, 521–533, 2012.

12. Fangjin Yang, Eric Tschetter, "Druid: A Real-time Analytical Data Store", SIGMOD '14 Proceedings of the ACM SIGMOD international conference on Management of data, June 2014

13. Samy Chambi, Daniel Lemire, Robert Godin, Kamel Boukhalfa, Charles R. Allen, Fangjin Yang, "Optimizing Druid with Roaring bitmaps", IDEAS '16: Proceedings of the 20th International Database Engineering & Applications Symposium, July 2016.

14. Apache Kafka. http://kafka.apache.org/

15. Fangjin Yang, Eric Tschetter, Xavier Léauté, Nelson Ray, Gian Merlino, Deep Ganguli, "Druid: a real-time analytical data store", SIGMOD '14: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, June 2014.

16. E. Bainomugisha, A. L. Carreton, T. v. Cutsem, S. Mostinckx, and W. d. Meuter. A survey on reactive programming. ACM Comput. Surv., 45(4):52:1–52:34, Aug. 2013. ISSN 0360-0300. doi: 10.1145/2501654.2501666

17. Laura Rettig, Mourad Khayati,"Online Anomaly Detection over Big Data Streams", ΄IEEE International Conference on Big Data, Switzerland, 2015.