A Novel Distributed Software Architecture for Managing Customer Behavior Data: A Case Study in Banking Sector

Özer Batu Kargılı
R&D Department
GTech
Istanbul, Turkey
ozer.kargili@gtech.com.tr

Ahmet Okan Arık
Management Information Systems
Yeditepe University
Istanbul, Turkey
okan.arik@yeditepe.edu.tr

Merve Bekler
R&D Department
GTech
Istanbul, Turkey
merve.bekler@gtech.edu.tr

Osman Uygar Köse
R&D Department
GTech
Istanbul, Turkey
uygar.kose@gtech.com.tr

Mehmet S. Aktaş
Computer Engineering
Yildiz Technical University
Istanbul, Turkey
aktas@yildiz.edu.tr

***Abstract*—We propose a Lambda-based, distributed, big data analysis architecture explicitly designed for the banking sector within the scope of this study. The proposed architecture is designed to extract useful information from user behavior data. The results derived from user behavior data obtained from many sources aim to increase the performance and efficiency of digital banking applications. The prototype implementation of the proposed architecture is designed, developed, and implemented. Performance and scalability tests were carried out on the prototype application. The results obtained reveal the usability of the proposed architecture in the banking sector.**

Keywords—Distributed Big Data Systems, Software Architectures, Lambda Architecture, Big Data Processing, Banking Sector.

I. INTRODUCTION

Today, many companies need to provide 360-degree insight to make critical business decisions and provide customer-oriented solutions. Companies obtain both company data and their customers' data from various data sources and use big data processing techniques to extract useful information from large-scale data.

One of these sources is behavioral data sources, called clickstream data, representing visitor interactions on applications or websites. Clickstream data, which contains much information such as clicks, views, visitor's browser, device information, page loading time, is becoming more critical in creating actionable insights. Clickstream analytics is a set of processes that collect, analyze, and provide visualized results of batch clickstream data. Practical analysis of clickstream data is critical in terms of the breadth of application areas. There are many application areas such as customer path analysis and optimization, website/application optimization, customer analysis, customer traffic analysis, basket analysis, marketing campaign management. In addition, it is used effectively to create personalized customer experiences by segmenting customers with different perspectives, especially for analyzes aimed at improving customers' product experiences.

One of the most significant benefits of clickstream data analysis for companies is that it has developed traditional marketing techniques. Many companies benefit from CRM in conventional marketing activities. There are difficulties in clickstream data analysis, such as the need for real-time analysis and the anonymity of the interacting users. The main goal to be achieved with clickstream data is to combine with data sources in CRM systems. In this way, it is possible to improve traditional marketing. Companies can offer personalized marketing activities by specializing in products and services according to the target audience.

When we look at the place of clickstream data in the market, according to the study of Allied Market Research (2020), we observe that the global clickstream analytics market size is $ 868.8 million in 2018 [1]. With growth, it is estimated to reach 2561.6 million dollars by 2026. Businesses marketing their products online are the most significant contributors to the development of the market. The growth of these companies and the rapid rise of digitalization have increased the demands for clickstream analytics tools. Companies have realized the importance of the data collected from these online platforms. Therefore, the need to capture revenue growth opportunities by collecting, analyzing, and evaluating customer interactions has driven the growth of the clickstream analytics market.

The real-time big data processing architecture that we propose aims to provide a total solution to increase the digital banking experience in the banking sector. It introduces a big data processing analysis approach with the help of customer transaction data. With real-time streaming data analysis in the

proposed architecture, it is possible to take immediate actions. In addition, the proposed architecture makes it possible to detect anomalies in customer behaviors in near real-time.

The lambda-based distributed system architecture that we employ in our proposed software architecture is applied in many different sectors. However, when we examine these studies, we observe that these studies are conducted with cold data from data warehouses. The architecture we propose provides 360-degree insight into the banking sector by considering hot data (real-time user interaction data) and cold data (user's previous interaction batch-data).

This study aims to explain the architecture that enables rapid detection of errors during transactions on digital bank platforms, prevent customer dissatisfaction or loss, manage the problems that occur in different channels through a single track, and analyze the data instantly. This architecture includes collecting continuous stream data, performing machine learning algorithms on them, providing the best solution to the customer, and interacting with the customer as soon as possible.

The organizational structure of this paper is as follows: In the second section, the research questions studied within the scope of this research are stated. In the third section, the literature review and basic concepts are explained, and in the fourth section, the methodology proposed within the scope of the research is introduced. In the fourth section, the prototype application of the proposed method is presented. The evaluation made on the prototype and the results obtained are shared in the fifth section. Section 6 summarizes the experiences gained as a result of the research and presents future studies.

II. Research Questions

In the scope of this study, we investigate the following research questions:

RQ1: What is state of the art regarding the big data processing architecture utilized for the banking sector?

RQ2: What is a design of a distributed system architecture for managing clickstream data collected from digital banking platforms? What are the benefits and challenges of this proposed architecture, designed specifically for the banking sector?

RQ3: Is there a specific benefit of choosing Lambda architecture over Kappa architecture based on the proposed architecture? If so, what is it?

III. Basic Concepts and Literature Review

A. Basic Concepts

Distributed systems are complex structures that use many different technologies on a number of computers. We review some of the distributed software architectures studied in the distributed systems literature below.

Service Oriented Architectures: This architecture enables software systems scale by utilizing independent services with different programming languages and enables the application to be scaled on the horizontal axis. The services have a centralized structure that communicates via transport protocols such as HTTP. Since it allows the services to be created in different languages, it is developed using the most appropriate programming language for the technology used for the service. It transports messages between client and host using XML based message structure, known as the SOAP message. **Resource-Oriented Architecture:** This architecture is based on RESTful services, called as micro services. In this distributed software architecture, HTTP is used as the communication protocol and the micro services use JSON object to exchange messages. **Lambda Architecture:** The lambda architecture is big data processing architecture that allows collective-driven data and streaming data to be processed in a single platform. This architecture, which tries to balance latency, throughput, and fault-tolerance, is frequently adopted in big data processing applications. It consists of three layers: batch-processing layer, speed layer and serving layer. The batch-processing layer is responsible for storing the data on distributed data storages and analyzing it using batch-processing techniques such as map-reduce programming model. Speed layer is responsible for the real-time processing. All stream-processing technologies are used in the speed layer. Both the batch-processing data and stream-processing data are kept in NoSQL databases in the serving layer. **Event-Based Architecture**: In this architecture, the communication of data-carrying processes is usually provided by the progress of events. The primary point in this architecture, which is associated with the publish/subscribe concept in distributed systems, is to ensure that the processes are delivered to the right subscriber by middleware after they are posted. **Shared Data-Based Architecture:** This architecture is similar to the event-based architecture, but in this architecture, the data is separated in a decoupled way. The data transmitted by the publisher may wait for a long time in the shared space [2].

B. Literature Review

In the study conducted by C.Lee and C.Lin [8], the authors created a restaurant recommendation system through open-source technologies. Apache Spark Streaming [3] and Apache Kafka [4] are used in the speed layer where real-time data will be processed, and Apache Spark is used in the Batch layer. Apache Mesos [5], an open-source cluster manager, was used to manage these resources automatically. The architecture consists of a mobile application, RESTful API server, batch layer, and speed layer. The mobile application is used to collect restaurant votes and comments. The API server is developed using Nginx, Node.js, and MongoDB technologies [6]. This system provides scalability by distributing the connections to various servers and provides high availability to check the correctness of the connections from the application. User ratings are taken from the API server on the batch layer, and analysis is performed using Apache Spark collaborative filtering algorithms. Then the result was sent back to the API server to be forwarded to the user. In the speed layer, real-time data from Kafka topics are analyzed with Spark streaming libraries. Apache Mesos has been used to configure clusters dynamically. With Docker [7] containers, it is ensured that different open-source technologies used in the architecture are carried out in an isolated way, efficient resource sharing, and fault tolerance.

The study conducted by Zahid, Mahmood, Morshed, and Timos Sellis [11], which includes the analyses of big data architectures to facilitate big data analytics in telecommunications sector, introduced a state-of-the-art lambda architecture-based system, LambdaTel. The purpose of this architecture is to process both batch and streaming data

in parallel. They used Apache Oozie [9] for workflow management and Apache Zookeeper [10] for resource management. As in the previous studies, it is stated that more effective processing and easy coordination are achieved by ensuring that each activity works in the container by using the container structure. Different from this study, in this manuscript, we discuss the weaknesses and strengths of the kappa and lambda architecture. We also propose a lambda architecture based distributed system with a usage scenario in the banking sector.

In Intelligence Transport Systems, the high rate of utilization of the data produced in high density and low latency value is essential for efficient traffic management and protection of the green environment. In the study published by Darwish and Abu Bakar [12], a lambda architecture based system specific to the Internet of Vehicles applications is introduced. This system uses fog computing and cloud computing in some layers to keep the latency value low in the computing process. This system utilizes a real-time computing system in the speed layer for features such as accident avoidance, danger warning, advanced driver assistance system, and autonomous driving. Different from this study, in this manuscript, we introduce a distributed system architecture for the banking sector that require lower latency values. We leave out fog computing techniques as out of scope in our distributed system design.

In [13], it is suggested that the scalability of traditional intelligent grid systems is insufficient for the storage and processing capabilities. They propose a five-layer distributed system. The first stage is the data stack layer, which is responsible for storing the data generated by smart sensors and devices. It is used to collect data and send it to Big Data Lake in the Hadoop cluster in the cloud. Then, analysis operations are carried out on the speed layer and batch layer. Next, the results are sent to the serving layer with the general name called data querying layer. Spark SQL, Hive, Impala, Bigquery, and AWS Athena technologies are used for query operations in this layer. The last layer was realized as the stage layer, which ensures the stability of the network, and increases its efficiency. In this study, clusters without container technology are potential problems that can reduce standardization and productivity. Different from this study, in our proposed distributed system architecture, we employ container virtualization technology. With the container technology, synchronization can easily be achieved between data clusters, and efficiency can be improved.

Demertzis and Iliadis [14] stated that one of the severe impacts of climate change is the interaction of species that facilitate the spread of invasive species in the ecosystem and noted that these species are ominous threats to biodiversity, affecting a wide range of human populations. In this direction, their study uses the innovative Lambda architecture to define and classify these invasive species, combining batch and real-time audio data for successful classification tasks with various data mining methods. Their lambda-based system architecture provides the usability of complex and hybrid algorithms in different layers.

The study published by Herman et al. [15], stated that the data should be analyzed within the scope of big data analytics to provide sustainable energy in the industry, proposed a hybrid architecture using Lambda architecture and a data lake for the use of big data in the industry. This study focuses on obtaining insights about energy use. They declared that due to the architecture applied, there was a significant reduction in water used and a significant reduction in CO2 emission.

To date, many architectures have been proposed for the creation of complex software architectures. Santos V.M., Misra S., and Soares M.S. [16], who also handled the Health Information System application at the Architecture Conceptualization stage, proposes a compatible, usable, reliable, and secure framework in this context.

Júnior A.A.C., Misra S., and Soares M.S [17] examined studies using the ISO/IEC/IEEE 42020 standard introduced in 2011. According to the results, they observed has been an increase in designing architecture using this standard since 2016. The sectors in which this standard is used the most is transportation, health, and energy, respectively.

There exist studies on distributed systems focusing on designing and implementing e-science workflows [21-25]. Different from these studies, in this manuscript, we focus on introducing a lambda-architecture based distributed system to manage clickstream data in the banking sector.

Also, there exists some studies focusing on analyzing provenance data, representing interactions between data, user agents, and processes [26-30]. Unlike these studies, in this study, we design and implement a system that analyzes the interactions between the banking system and its users.

Based on our findings mentioned above, after conducting a literature review, we can conclude the following. There is no distributed system architecture designed for managing clickstream data collected from digital banking platforms to the best of our knowledge. To this end, we answer the first research question, which is mentioned in Section 2.

IV. PROPOSED METHODOLOGY

Distributed System Architecture: The conceptual design indicating the overall workflow of the study is shown in Figure 1 below. Customer activity data collected from digital banking platforms (web, mobile, ATM) and physical banking platforms (bank branches) will be the primary data sources representing customers' behavior while using banking platforms. The performed transactions are received from the user and integrated into the system using the publish-subscribe messaging bus. Along with analyzing hot and cold data (real-time and past interaction data), real-time analyses such as

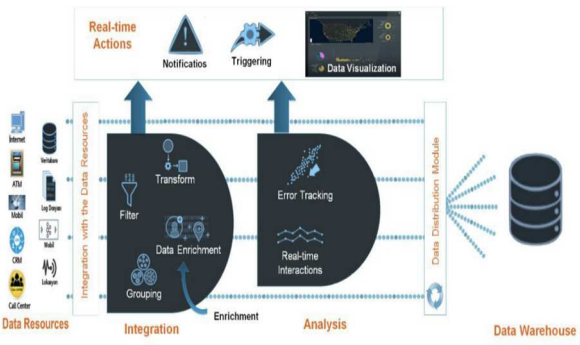

Fig. 1. Conceptual view

error tracking, sending notifications to the user, triggering some operations, and visualizing the result data are obtained. The analysis results are also aimed to be stored in the data warehouse.

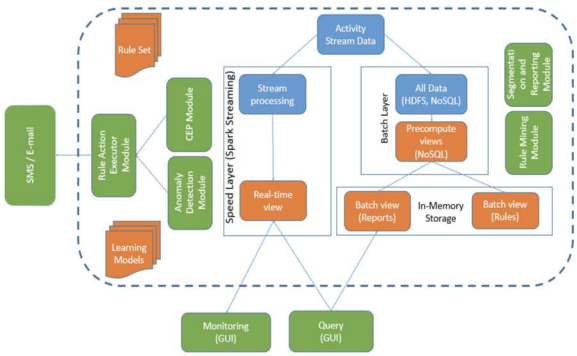

Figure 2 Architectural view

Architectural View: Each component of the architecture is shown in Figure 2. We use lambda-based big data processing architecture, which plays a fundamental role in meeting the big data processing needs. There are three main layers in the proposed architecture for this banking sector: batch layer, speed layer, and serving layer. The batch layer has two main functions: managing all data and pre-computing batch views. The speed layer processes data streams in real-time and deals only with the latest data. The real-time view is available immediately after data is acquired.

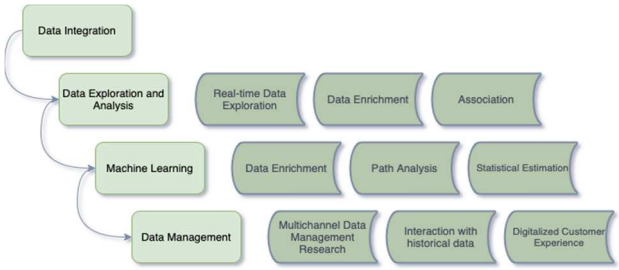

Figure 3 Progress flow

Figure 3 shows the progress flow structure of the proposed system. As illustrated in Figure3, the next step after the integration of various data sources is data discovery and analysis. Since it depends on the scenario, data enrichment can be made if necessary. Besides, real-time data is processed in this step. The third step is the stage where machine learning is carried out. Here, pre-processing steps and path analysis are done. The results obtained in the machine learning step are used to improve the customer experience on digital platforms.

Fig. 4. Digital channel data flow and anomaly detection sample scenario flow

In this study, we also implemented a digital banking platform to facilitate testing of the proposed software architecture. For the advanced analytical operations, data was generated using test automation tools. The events, collected as log files, were transferred to the big data processing platform flowing over a publish-subscribe-based message channel. Activities to be carried out within the scope of a complex event processing module have been determined based on different scenarios. In the prototype application, the following functionalities were provided: Event modeling and collecting events from log files, extracting association rules algorithms, and batch-data analysis using distributed data storage platforms, and real-time data analysis using streaming data processing platforms. The services on this platform are handled in a modular structure based on the microservice architecture. It is built as a restful service approach, where each service runs its jobs independently from other jobs, aiming at minimal communication with other services. The modules discussed in the proposed architecture can be examined as batch processing and stream processing under these two main categories. In this section, by introducing the architectural design of the proposed distributed system, we answer research question #2 mentioned in Section 2.

Modules Utilizing Batch Data Processing:

Preprocessing Module: Data pre-processing methods within the project's scope have been implemented on the Spark platform with the map-reduce programming model.

Association Rule Mining Module: A rule suggestion system has been developed within the scope of this module. While developing this module, FP-Growth, Apriori, Eclat algorithms were implemented/used on the Spark platform.

Micro-Segmentation Module: Within the scope of this module, segmentation functionalities are implemented to identify the customer segments and apply the marketing activities to the appropriate customer group. Segmentation methods are implemented based on map-reduce programming model. Here, we implement K-means and Bisecting K-means unsupervised machine learning algorithms.

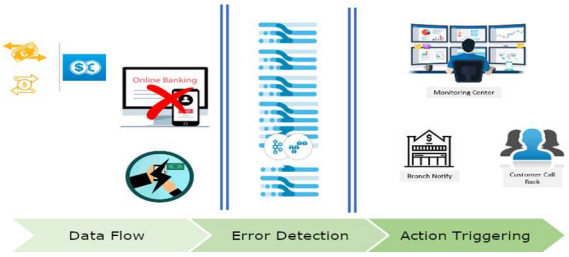

Fig. 5. EFT, money order error detection scenario flow

Modules Utilizing Stream Data Processing:

Instant Action Situation Detection Module: Within the scope of this module, studies were carried out to give instant response actions to the customers as a result of real-time analysis of the errors received by customers while using the bank's digital channels. This module aims to strengthen the communication between the customer and the bank and ensure rapid action against errors in the systems.

Anomaly Detection Module: Within the scope of this module, we worked on classifying the usage cases of the platform as "normal use" and "expected use." To understand the activities performed by the customer, anomaly situations are identified by utilizing DBSCAN, PCA, and K-Means algorithms.

Rule Engine (Business Process Engine) Module: All customer transactions performed on digital channels can be processed in the instant action module. The scenarios entered from the system control screens are kept in a cache mechanism used by the instant action module. Thanks to the Rule Engine, users can create instant actions according to business requirements. Rules are created with rule creation forms on the system control screens.

V. PROTOTYPE AND EVALUATION

A. Prototype

This section includes the key points and evaluation of the prototype implementation of the proposed software architecture. The UI is implemented with Django Framework in python. The latest python version (Python 3.7) is used in the prototype. We use Apache Kafka as a fault-tolerant messaging queue. Kafka has subordinate and master nodes with a coordinated load balancing system controlled by Apache Zookeeper. We use ZooKeeper as a centralized service for providing configuration information, naming, and distributed synchronization. We use Ubuntu 18.04 [18] and JDK 8 to run the Kafka servers. Apache Spark scenarios are written in Python 3.7 and Apache Flink [19] scenarios are written in Java and compiled with Java 1.8. Besides real-time actions, transaction are stored in both MongoDB and HDFS [20]. MongoDB is used as a NoSQL database, and Hadoop is used for batch operations.

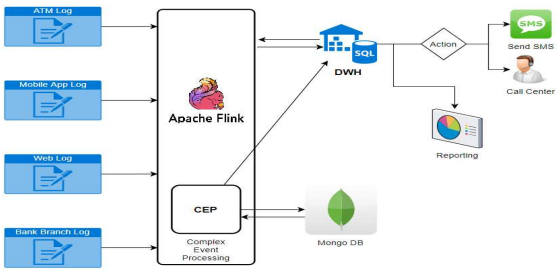

Fig. 6. Data flow schema

B. Evaluation

We conducted various experiments to investigate the performance of the proposed architecture from the perspective of scalability and latency. The design of the experiments is explained below, followed by the results of this evaluation study.

Design of the Experiments: The operations performed in the prototype were carried out on a virtual machine with Red Hat Enterprise Linux Server 7.9 installed with 4 CPUs and 16 GB memory. For the tests, Apache Kafka 2.6.0, Apache Flink 1.12.2, Spark 3.0.2 versions were used.

In the experimental study, we investigate the change the latency in executing the system functions. We also evaluate the scalability of the architectural structure against increasing message loads. We used different message loads such as 100, 1000, 10000, and 100000 events per second. We run the tests 100 times and record average and standard deviation for the observations.

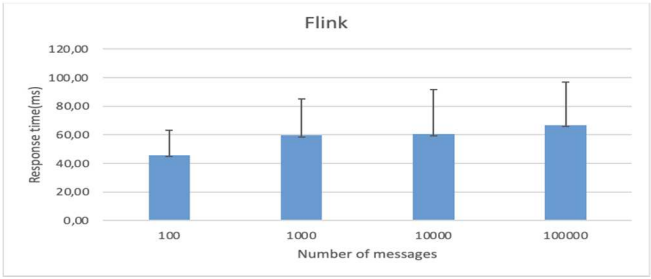

Fig. 7. Apache Flink performance test results

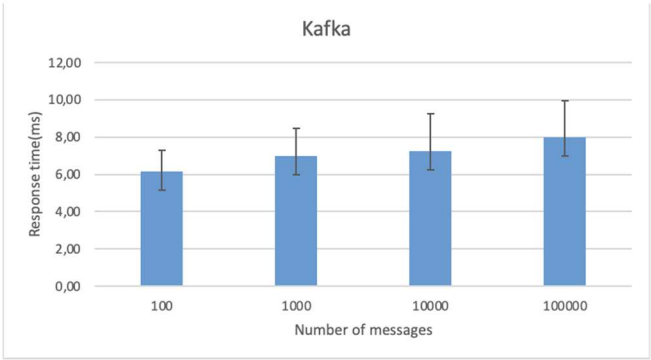

Fig. 8. Kafka performance test results

Results of the Experimental Study and Discussion: Figure 7 and Figure 8 summarizes the results. The results show that the response time for the system is in the order of milliseconds in detecting the patterns within streaming data (i.e., clickstream data). The results also show that the system

performance increases proportionally in a linear manner under increasing message load. While the number of messages per second sent to the system increases, if the delay time requested from the system remains above the requested values, it is recommended to increase the number of nodes in the system. In this experimental study, we mainly focused on how the system performs for analyzing the streaming data. To this end, we investigated the performance of the system components that are primarily designed for processing and analyzing the clickstream data collected from digital banking platforms. We leave out the performance analysis of the batch processing-based modules of the system for future study.

VI. CONCLUSIONS AND FUTURE WORK

This study conducted a literature review covering existing distributed system architectures for managing both streaming and batch data in different domains. We proposed an abstract design of a distributed system that addresses the requirements of digital banking platforms. The proposed architecture provides instant and appropriate actions to the right customer at the right time. In turn, this increases customer satisfaction in digital banking platforms. To facilitate the testing of the system, we implemented a prototype implementation. Also, we provided a comprehensive evaluation study that investigated the performance and scalability of the prototype application. We used distributed streaming data processing technique to facilitate large-scale real-time data analysis. At the same time, we developed a rule suggestion system based on the establishment of association rules. The rule suggestion system enables digital banking platforms to provide assistance to their customers during or after the service or inform them in abuse and fraud cases. We ran machine learning algorithms on customer clickstream data collected near real-time and detected anomalies on the platform's use cases. Using incremental cluster analysis algorithms, we modeled online "normal usage behavior" data. The complex event processing method has been used to detect predetermined situations on the flowing data.

In evaluating the system, the increased workload has been applied to the system, and the maximum number of activities that the system can support was measured. The delay time required for the system to process each event data is also calculated. As a result of the tests, it is understood that the processing overheads of the proposed system are in the order of milliseconds and are negligible.

This architecture has been designed specifically for a specific domain; in future studies, we plan to develop the system for different fields such as retail and automotive.

ACKNOWLEDGMENT

We want to thank G-Tech R&D Center for providing the necessary computational facilities and the dataset for this study. TUBITAK supported this study under project ID# 3190349. We also want to thank Murat Başbuğ and Tayfun Yalçınkaya and Muammer Gülerce for their support in this study.

REFERENCES

[1] Website for clickstream analytics market, available at the following link: https://www.alliedmarketresearch.com/clickstream-analytics-market-A05942, Accessed Date: 6/20/2021.

[2] Distributed Systems: Principles and Paradigms by Tanenbaum, Andrew S., Van Steen, Maarten, 2006, Pearson.

[3] Website for Apache Spark Framework, available at the following link: https://spark.apache.org/, Accessed Date: 6/20/2021.

[4] Website for Apache Kafka Framework, available at the following link: https://kafka.apache.org/, Accessed Date: 6/20/2021.

[5] Website for Apache Mesos Framework, available at the following link:http://mesos.apache.org/, Accessed Date: 6/20/2021.

[6] Website for MongoDB Framework, available at the following link: https://www.mongodb.com/, Accessed Date: 6/20/2021.

[7] Website for Docker Framework, available at the following link: https://www.docker.com/, Accessed Date: 6/20/2021.

[8] C. Lee and C. Lin, "Implementation of Lambda Architecture: A Restaurant Recommender System over Apache Mesos," 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), Taipei, 2017, pp. 979-985, DOI: 10.1109/AINA.2017.63.

[9] Website for Apache Oozie Framework, available at the following link: https://oozie.apache.org/, Accessed Date: 6/20/2021.

[10] Website for Apache ZooKeeper Framework, available at the following link: https://zookeeper.apache.org/, Accessed Date: 6/20/2021.

[11] H. Zahid, T. Mahmood, A. Morshed and T. Sellis, "Big data analytics in telecommunications: literature review and architecture recommendations," in IEEE/CAA Journal of Automatica Sinica, vol. 7, no. 1, pp. 18-38, January 2020, DOI: 10.1109/JAS.2019.1911795.

[12] T. S. J. Darwish and K. Abu Bakar, "Fog Based Intelligent Transportation Big Data Analytics on The Internet of Vehicles Environment: Motivations, Architecture, Challenges, and Critical Issues," in IEEE Access, vol. 6, pp. 15679-15701, 2018, DOI: 10.1109/ACCESS.2018.2815989.

[13] A. A. Munshi and Y. A. I. Mohamed, "Data Lake Lambda Architecture for Smart Grids Big Data Analytics," in IEEE Access, vol. 6, pp. 40463-40471, 2018, DOI: 10.1109/ACCESS.2018.2858256.

[14] Demertzis K., Iliadis L., Anezakis VD. (2019) A Machine Hearing Framework for Real-Time Streaming Analytics Using Lambda Architecture. In: Macintyre J., Iliadis L., Maglogiannis I., Jayne C. (eds) Engineering Applications of Neural Networks. EANN 2019. Communications in Computer and Information Science, vol 1000. Springer, Cham. https://doi.org/10.1007/978-3-030-20257-6_21

[15] Jacobus Herman, Hendrik Herman, Marc John Mathews, Jan Corné Vosloo, "Using big data for insights into sustainable energy consumption in industrial and mining sectors," Journal of Cleaner Production, Volume 197, Part 1, 2018, Pages 1352-1364, ISSN 0959-6526, https://doi.org/10.1016/j.jclepro.2018.06.290.

[16] Santos V.M., Misra S., Soares M.S. (2020) Architecture Conceptualization for Health Information Systems Using ISO/IEC/IEEE 42020. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2020. ICCSA 2020. Lecture Notes in Computer Science, vol 12254. Springer, Cham. https://doi.org/10.1007/978-3-030-58817-5_30

[17] Júnior A.A.C., Misra S., Soares M.S. (2019) A Systematic Mapping Study on Software Architectures Description Based on ISO/IEC/IEEE 42010:2011. In: Misra S. et al. (eds) Computational Science and Its Applications – ICCSA 2019. ICCSA 2019. Lecture Notes in Computer Science, vol 11623. Springer, Cham. https://doi.org/10.1007/978-3-030-24308-1_2

[18] Website for Ubuntu Framework, available at the following link: https://ubuntu.com/, Accessed Date: 6/20/2021.

[19] Website for Apache Flink Framework, available at the following link: https://flink.apache.org/, Accessed Date: 6/20/2021.

[20] Website for Apache HDFS Framework, available at the following link: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, Accessed Date: 6/20/2021.

[21] Aydin, G. et al., Building and applying geographical information system Grids, CONCURRENCY AND COMPUTATION-PRACTICE & EXPERIENCE, cilt.20, sa.14, ss.1653-1695, 2008

[22] Aktaş M. S. , Pierce M., Fox G., Leake D., A Web based conversational case-based recommender system for ontology aided metadata discovery, The 5th IEEE/ACM International Workshop on Grid Computing (GRID-04), 2004.

[23] Aktas, M.S. et al, ISERVO: Implementing the International Solid Earth Research Virtual Observatory by integrating computational grid and geographical information Web Services, PURE AND APPLIED GEOPHYSICS, cilt.163, ss.2281-2296, 2006.

[24] Fox G, Aktas G, Bulut S, et al. Real Time Streaming Data Grid Applications: 253-267; Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements. Springer International Publishing. 2006.

[25] Aktas, M. S., Fox, G. C., Pierce, M., Fault tolerant high performance Information Services for dynamic collections of Grid and Web services, Future Generation Computer Systems, Volume: 23, Issue: 3, 2007.

[26] Tufek, Alper; et al., Provenance Collection Platform for the Weather Research and Forecasting Model, 14th International Conference on Semantics, Knowledge and Grids (SKG), 2018.

[27] Riveni, Mirela; et al., Application of provenance in social computing: A case study, CONCURRENCY AND COMPUTATION-PRACTICE & EXPERIENCE, Volume: 31, Issue: 3, Published: FEB 10 2019.

[28] Baeth, Mohamed Jehad; Aktas, Mehmet S., Detecting misinformation in social networks using provenance data, Concurrency and Computation: Practice and Experience, Volume: 31, Issue: 3, FEB 10 2019.

[29] Baeth, M. J.; Aktas, M. S., An approach to custom privacy policy violation detection problems using big social provenance data, CONCURRENCY AND COMPUTATION-PRACTICE & EXPERIENCE, Volume: 30, Issue: 21, Published: NOV 10 2018.

[30] Tas, Yucel; et. al., An Approach to Standalone Provenance Systems for Big Social Provenance Data, 12th International Conference on Semantics, Knowledge and Grids (SKG), 2016.