A Comprehensive Analysis of Architectures and Methods of Real-Time Big Data Analytics

Sinan AY
Department of Computer Engineering, Turkish Military Academy, Ankara, Turkey
Email: xsinanay@gmail.com

M. Ali Akçayol
Department of Computer Engineering, Gazi University, Ankara, Turkey
Email: akcayol@gazi.edu.tr

***Abstract*—The studies of big data analytics has emerged due to the lack of data analysis methods, and storage problems with traditional database systems. Some big data applications require real time analysis, and there is time constraint to analyze for the applications. Various methods and have been proposed to overcome this difficulty. In this study, several architectures and applications for real-time big data analysis have been investigated and compared with each other in details. Valuable suggestions have been proposed for researchers working in real-time big data analytics.**

***Index Terms*—big data, real time analysis, real time big data architecture**

I. Introduction

The big data defines data that are collected from different sources and grows quickly. The big data is expressed by variety, volume, velocity, veracity, value (5V). Big data obtained from various sources such as social networks, sensors, financial transactions, health system, and telecommunication. Nowadays, it is quite difficult to storage, manage and analyze fast growing big data with traditional database systems [1]. For this reason, in literature, numerous different methods have been proposed for the analysis of big data.

Also in some areas, big data have to be analyzed in real-time. There is a time constraint for real-time big data analysis and different architectures are applied for real-time big data analysis.

In this paper, several architectures and applications have been investigated to improve performance of real-time big data analysis. The contribution of this paper is that the applications performed in different areas have been compared with each other, and some suggestions have been proposed for researchers working in real-time big data analytics.

The rest of this paper is organized as follows, in section 2, proposed architecture and indexing methods to perform big data analysis in real time have been investigated, and in section 3 the implemented applications to analyze big data in real time have been presented in details.

Manuscript received February 10, 2017; revised June 26, 2017.

II. Architectures Used for Real-Time Analysis of Big Data

Marz and Warren [2] proposed Lambda architecture that is illustrated in Fig. 1. The system has the ability to respond quickly with precomputed and indexed batch views. The Lambda architecture consists of three layers. These are batch, service and speed layers. The batch layer stores the master dataset and computes arbitrary functions on that master dataset. The service layer indexes the batch views. Thus, the system can respond effectively to desired query. The speed layer computes real time data that has not been processed by the batch layer to produce real time views.

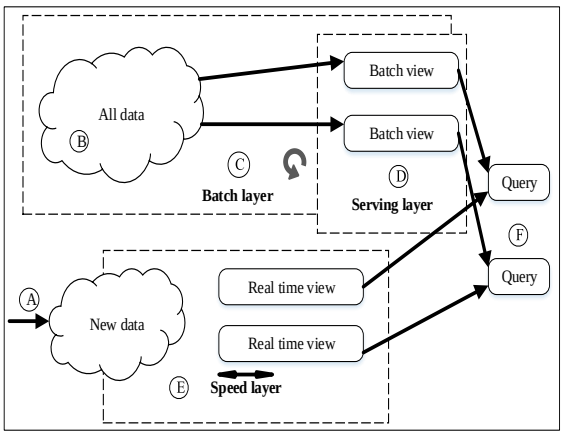

Figure 1. Lambda architecture

Twardowski and Ryzko [3] presented the multi-agent architecture for real-time big data analysis as shown in Fig. 2. In this architecture, each of agents which are autonomous and distributed is responsible for a particular job. The input data is processed as stream data. The stream is collected by stream receiver agents which are responsible for pre-processing. Finally, all the data is sent to the archiver and the stream processing agents. The agents are responsible for processing the new data in the batch layer and the speed layer respectively. In the batch processing the new data is stored to the Hadoop

doi: 10.18178/lnit.5.1.7-12

Distributed File System (HDFS). The batch driver agents coordinate the computations. The batch worker agents create batch view by performing their assigned jobs. The speed layer works with a similar mechanism where stream processing agent assigns the jobs to appropriate worker agent. The worker agent creates the output of real time views. The service layer creates service agent if needed and collects the necessary data for this agent. The service agent is terminated when the request is completed.

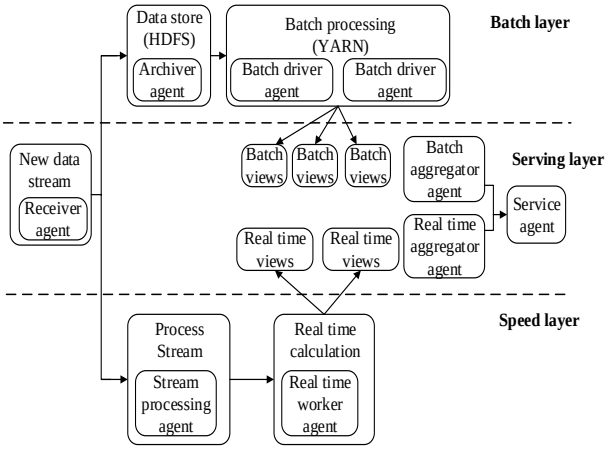

Figure 2. Multi-agent big data analysis architecture

Martínez-Prieto, Cuesta, Arias and Fernández [4] presented the Service-OnLine-Index-Data (SOLID) architecture for real-time analysis of big semantic data. In the architecture shown in Fig. 3. Real time operations and big semantic data operations are separated from each other. This provides efficiency for data management and processing. However, the approach requires the coordination of the two data stores. SOLID architecture consists of three tiers: content, merge and service. The content tier consists of online, index and data layers. The ata layer is similar to batch layer of Lambda architecture. The index layer reduces the complexity of big data to make effective querying at real time. The online layer performs fast write and query operations of the real-time system. The merge tier ensures that the data at runtime is integrated with the big semantic data. The query processor module in the service tier collects and parses queries and creates dynamic pipeline to solve the queries.

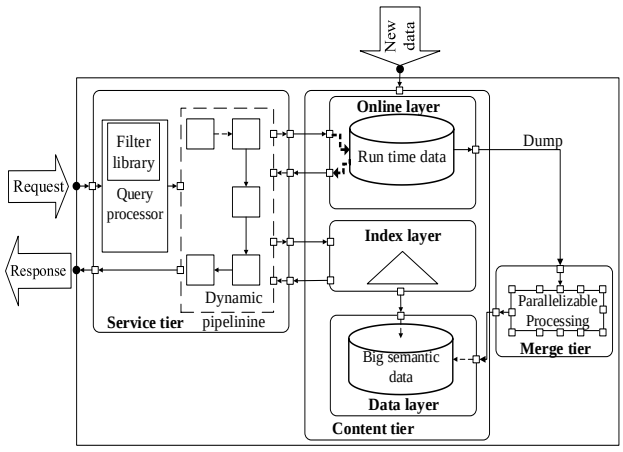

Figure 3. SOLID architecture

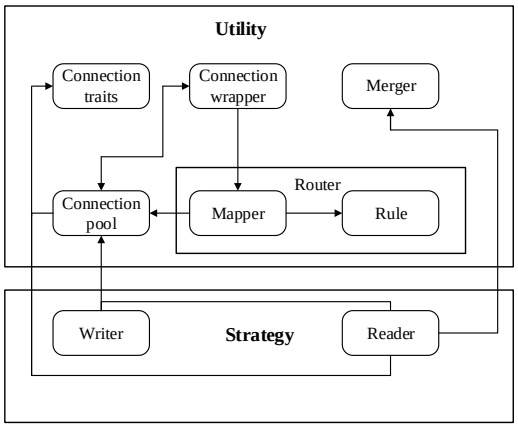

Figure 4. AIS Architecture

Mo and Wang [5] presented the Asynchronous Index Strategy (AIS) structure. This structure has shown high performance for time series real-time big data stream storage. AIS architecture, shown in Fig. 4, includes the utility part and strategy part. The connection properties within the strategy part define common actions such as creating a connection and query command. The rule module manages the mapping rules for database addresses from keywords. The mapper module maps the database addresses to database connections. The merging module is responsible for merging the query results. The strategy section consists of the writer and reader modules.

The writer module is responsible for adding and updating on MongoDB [6] and the reader module is responsible for query requests. Insertion performance of AIS-based MongoDB is better than MongoDB sharding and single MongoDB. AIS-based MongoDB has 17 times better performance than MongoDB sharding cluster with 200 million documents.

Wang, Zhang, Gao and Xing [7] have proposed the Punt Log Structured Merge (pLSM), a variant of LSM-Tree. In this model, LSM-Tree is used to improve writing performance and Cache Oblivious Look-ahead Array (COLA) is used to accelerate query response. In the experimental study, the performances of pLSM, B-Tree and LSM-Tree structures were compared. For random insertion and sequential insertion, pLSM has performed better than the other two methods. In order to evaluate the query performance, point query and range query were performed. According to query performance, performance of pLSM was near the best-performing B-tree method in point query. However, pLSM has shown poor performance in the range query.

II. Real-Time Big Data Analysis Architectures

In this section, real-time big data analysis applications on mobile, cloud and other environments have been examined.

A. Applications in Cloud Environments

Wang, Zhang, Zhang and Lim [8] have presented the Smart Traffic Cloud infrastructure shown in Fig. 5. This architecture allows the collection and management of traffic data. This infrastructure enables distributed and

parallel data management and analysis using the Map-Reduce and ontology database. The proposed architecture has three layers. These are infrastructure, data processing/analysis and application layers. The infrastructure layer provides the cluster of server to high-level modules and services. The data processing/analysis layer achieves meaningful results using machine learning algorithms such as analysis, clustering and classification. The application layer performs data send/receive, storage and administrator operations. An application of real-time traffic map has been implemented using the proposed architecture. In this application, users send the information of timestamp, their position using Global Positioning System (GPS), speed and acceleration. The master node creates job files receiving this data and adds these files to the queue. The master node generates job lists using this information and puts them in a job queue. Then, the global job scheduler sends each job to the appropriate work node. The work nodes extract some information using this data: (1) road segments according to users' coordinates and direction of the user, (2) filtering invalid and erroneous speed and acceleration measurement, (3) speed of the user.

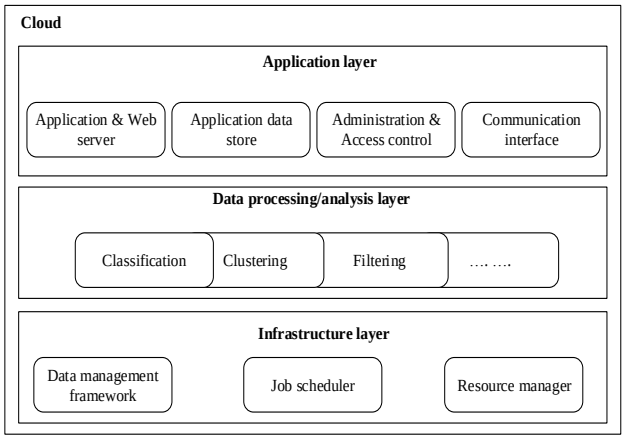

Figure 5. Smart traffic cloud architecture

Yu, Jiang and Zhu [9] designed the RTIC-C for cloud-based data mining for real-time traffic. RTIC-C provides the distributed data management service based on HDFS and HBase [10]. RTIC-C consists of four layers as shown in Fig. 6. These layers are resource layer, cloud infrastructure layer, mining virtualization layer and mining application layer. Resource layer integrates data incoming from different sources. The cloud infrastructure is based on Hadoop [11] and integrates distributed resources. The traffic data is described as key/value pairs and stored on HDFS or HBase. Distributed massive storage component distributes the data to different data nodes to provide reliability. The map-reduce parallel computing framework provides parallel computing for data processing on distributed sources. Open service provides to access the cloud environment. The mining virtualization layer provides the mining tools such as traffic jam detection, traffic signal control model. The mining application layer combines various services for mining and performs mining applications such as weather forecasting, discovery of restricted area and urban transport.

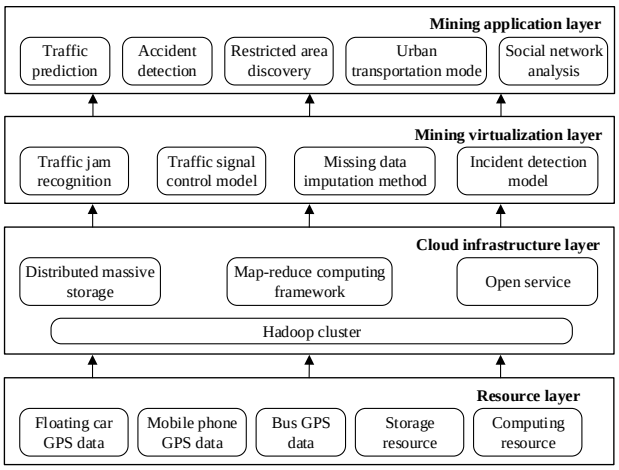

Figure 6. RTIC-C architecture

B. *Applications Using Mobile Technology*

Garzo, Benczur, Sidlo, Tahara and Wyatt [12] have presented distributed streaming algorithms and infrastructures for efficient processing of large-scale mobile data. In the system shown in Fig. 7, Storm [13] and S4 are used in the stream processing layer. Since the stream processing layer does not guarantee to store history information, a persistence module has been created to protect history data even in case of an error. In the system for caching and virtualization layers, Cassandra [14] was used because of its high throughput writing. User defined functions have been used in the mobile data processing layer to perform location estimation by collecting history data. According to experimental results, the proposed system provides low latency and high throughput for real time application based on motion prediction.

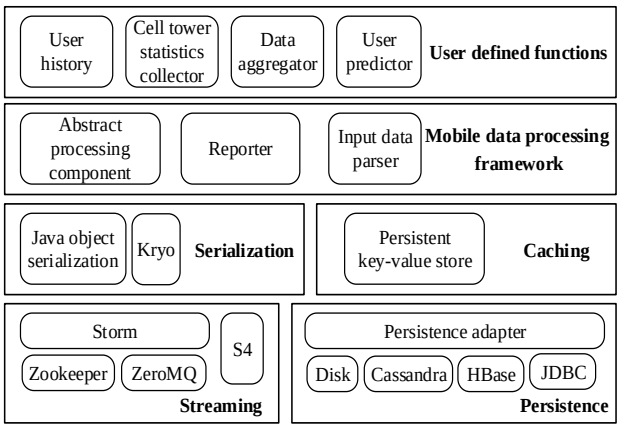

Figure 7. Mobility prediction architecture

Zhao, Sun and Liao [15] have developed a system for the analysis of large-scale GPS data by combining Spring and Storm. They have implemented K-means algorithm on Storm. As shown in Fig. 8, the architecture includes data collecting part, data analysis part and data storage part. The data collector gathers the data and sends them to

the message queue of Kafka. The messages include time, longitude, latitude and other information. The proposed method almost doubles the performance for execution time.

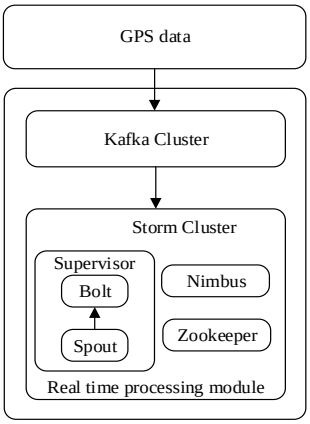

Figure 8. Real time processing system for GPS data

Jayawardhana, Kumara, Perera and Paranawithana [16] have presented the Kanthaka for the requirement of telecom operator. Kanthaka, shown in Fig. 9, can analyze 30 million records per day. Kanthaka consists of front-end layer and back-end layer. The front-end is used to define the promotions. The promotions in the database are converted into Cassandra queries in the compiler module. The back end periodically receives the Call Detail Record (CDR) data from the operator. The preprocessor stores the CDR data in the hashmap in memory module according to queries from the receiving front-end. The selection of the appropriate subscriber for particular promotions is carried out by the Periodic Eligibility Checking module. Experimental results show that the latency increases when the number of promotions in the database increases.

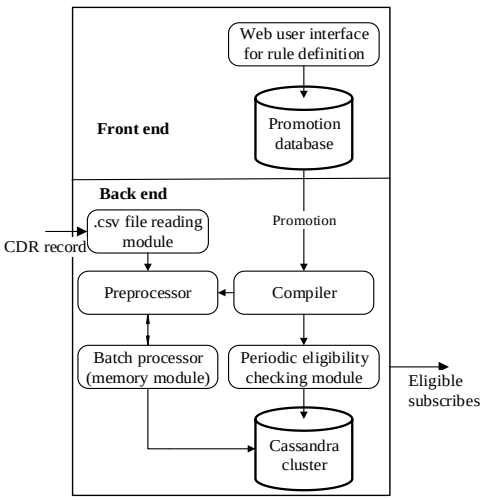

Figure 9. Kanthaka architecture

C. Applications in Other Areas

Mouro and Sarno [17] used Storm and Scalable Advanced Massive Online Analysis (SAMOA) [18] together. Architecture is shown in Fig. 10. SAMOA consists of processing item, processor and stream. The processing item wraps a processor to perform machine learning algorithms using nodes provided by the stream processing engine. Processor performs the machine learning algorithms. Stream is a connection and allows data exchange between processing items. The second component of the system is the stream processing engine called Apache Storm. In the implementation, "Skype" and "Normal" class data for training purposes were created. “Skype” class includes 50 Skype sessions data and “Normal” class contains other traffic data such as http, ftp. According to different experiments, Skype traffic is classified at accuracy rate of 90.05%.

Yang, Liu, Zhang and Yang [19] have developed a Storm-based architecture that includes data creation, data processing, and data storage. In this architecture, RabbitMQ is used as the data generator and Cassandra is used as the distributed database. RabbitMQ receives incoming messages according to particular rules and transmits them to the appropriate recipients. Nginx, a high performance HTTP server, was used as a load balancer in the system. Nginx is responsible for providing a balanced workload for each processor.

Bai [20] has presented a Hbase-based real-time search method for big log data. In the proposed method, flume agents collect log records from end users. ElasticSearch is used for the analysis and indexing of logs. In the experimental study, 7 GB log file containing 148.928.992 log was used. They searched the “Bigdata” keyword, the number of total matched log events are 4375. The first 25 results in the search are returned in 6 seconds.

Bakır, Aydoğan, Aydın, Khodabakhsh, Arı and Ercan have presented a sensor based data validation solution [21]. 60.000 sensors located in the Tüpraş refinery create an average of one hundred thousand records per day. They have used Complex Event Processing (CEP) engine and Cassandra Query Language (CQL) which is a query tool similar to SQL language. In the study, it was shown that at 15,000 events/second, relations and measurement mistakes were detected and classified correctly.

He, Lu and Swanson [22] have developed a real-time MapReduce scheduler. System has three components. These are admission controller, job dispatcher and feedback controller. The admission controller defines the sequence of tasks that should be given to resources. The job scheduler allows assigning new tasks to the worker nodes. The feedback controller provides that the input controller is up to date. In the experimental study, the proposed system was compared with the deadline constraint scheduler. It has been concluded that the recommended scheduler is better according to the feedback rate.

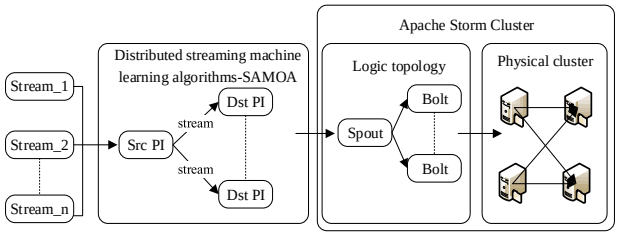

Figure 10. Storm/SAMOA architecture.

TABLE I. COMPARISON OF EXAMINED STUDIES

Study	Explanation	Data	Performance
[2]	Real-time big data architecture is presented.	-	-
[3]	Multi-agent big data analysis architecture is presented.	The system has not been implemented. The proposed system was discussed.	-
[4]	Big semantic data analysis architecture is presented.	Meteorological data.	Storage requirements have reduced.
[5]	Asynchronous Index Strategy is presented.	Randomly generated stream document.	AIS based on MongoDB has 17 times better insert performance with 200 million documents than others.
[7]	pLSM index is designed.	Randomly generated key-value pairs in string format.	For insertion, pLSM has performed better than B-Tree and LSM-Tree methods. pLSM was near the best-performing B-tree method in point query. However, pLSM has shown poor performance for range query.
[8]	An infrastructure is proposed for traffic data management.	Traffic sensor data and data files with GPS, timestamp, speed and acceleration information.	-
[9]	System based on cloud computing is designed.	Traffic dataset.	With enough data node, HDFS write speed is 40 MB/s, and HBase write speed is 35.000 record/s.
[12]	Distributed streaming algorithms and infrastructures are deployed.	Fine Resolution Mobility Trace Data Set (SET2) that includes 50 million events.	Input data processing latency is about 1023 ms.
[15]	Storm topology model is presented by combining with Spring.	GPS data.	Proposed method almost doubles the performance of execution time.
[16]	System is designed for promotion recommendation to eligible users.	CDR data.	Latency increases with the number of promotions increases in database and records increases in the file.
[17]	Storm and SAMOA based on architecture is proposed for real time big data processing and analysis.	Internet traffic data.	Skype traffic is classified as accuracy rate of 90.05%.
[19]	Storm based on real-time big data processing system is proposed.	Messages generated from RabbitMQ.	-
[20]	Real time big data search method is presented.	7GB log event data.	Improved the performance for execution time.
[21]	Sensor based data validation solution is presented.	Sensor data.	15.000 events/second, relations and measurement mistakes were detected and decided correctly.
[22]	Novel real time scheduler is presented.	-	Proposed system was compared with the deadline constraint scheduler and is better according to the feedback rate.
[23]	Intrusion detection system is presented.	84.030 instances of internet traffic data.	Accuracy is %99.7 when random forest algorithm is used as classifier.

Singh, Guntuku, Thakur and Hota [23] developed a semi-real-time application to detect peer-to-peer botnet attacks via machine learning algorithms. System has three components: traffic sniffer module, feature extraction module, and machine learning module. The traffic sniffer module saves the packets and performs the pre-processing phase. The feature extraction module obtains attributes using Apache Hive [24]. Mahout [25] was used in the machine learning module and the classification accuracy rate was 99.7% when random forest algorithm is used as classifier.

The comparison of the architectures have been given in Table I.

III. CONCLUSION

In this study, architectures and applications of real-time big data analytics have been investigated and compared with others. According to studies in different fields, big data applications need to be well organized in time-critical systems to let the system gain the ability to response in a short time.

REFERENCES

[1] J. Manyika, B. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, *Big Data: The Next Frontier for Innovation, Competition, and Productivity*, McKinsey Glob. Inst., 2011, pp. 1-3.

[2] N. Marz and J. Warren, *Big Data : Principles and Best Practices of Scalable Realtime Data Systems*, Manning Publications, 2012, ch. 1, pp. 13-25.

[3] B. Twardowski and D. Ryzko, "Multi-agent architecture for real-time big data processing," in *Proc. IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Warsaw, 2014, pp. 333-337.

[4] M. A. Martínez-Prieto, C. E. Cuesta, M. Arias, and J. D. Fernández, "The solid architecture for real-time management of big semantic data," *Futur. Gener. Comput. Syst.*, vol. 47, pp. 62-79, October 2014.

[5] X. Mo and H. Wang, "Asynchronous index strategy for high performance real-time big data stream storage," in *Proc. 3rd IEEE International Conference on Network Infrastructure and Digital Content*, Beijing, 2012, pp. 232-236.

[6] Introduction to MongoDB - MongoDB Manual. [Online]. Available: https://docs.mongodb.com/manual/introduction/

[7] J. Wang, Y. Zhang, Y. Gao, and C. Xing, "pLSM: A highly efficient LSM-tree index supporting real-time big data analysis," in *Proc. IEEE 37th Annual Computer Software and Applications Conference*, Kyoto, 2013, pp. 240-245.

[8] W. Q. Wang, X. Zhang, J. Zhang, and H. B. Lim, "Smart traffic cloud: An infrastructure for traffic applications," in *Proc. IEEE 18th International Conference on Parallel and Distributed Systems*, Singapore, 2012, pp. 822-827.
[9] J. Yu, F. Jiang, and T. Zhu, "RTIC-C: A big data system for massive traffic information mining," in *Proc. Int. Conf. on Cloud Comput. and Big Data*, Fuzhou, 2013, pp. 395-402.
[10] Apache HBase TM Reference Guide. [Online]. Available: https://hbase.apache.org/book.html
[11] Hadoop Tutorial-YDN. [Online]. Available: https://developer.yahoo.com/hadoop/tutorial/module1.html
[12] A. Garzó, A. A. Benczúr, C. I. Sidló, D. Tahara, and E. F. Wyatt, "Real-time streaming mobility analytics," in *Proc. IEEE International Conference on Big Data*, Silicon Valley, CA, 2013, pp. 697-702.
[13] Tutorial. [Online]. Available: http://storm.apache.org/releases/current/Tutorial.html
[14] The Apache Cassandra Project. [Online]. Available: http://cassandra.apache.org/
[15] J. Zhao, Z. Sun, and Q. Liao, "Implementation of K-means based on improved storm model," in *Proc. 15th IEEE International Conference on Communication Technology*, Guilin, 2013, pp. 728-732.
[16] P. Jayawardhana, D. Perera, A. Kumara, and A. Paranawithana, "Kanthaka: Big Data Caller Detail Record (CDR) analyzer for near real time telecom promotions," in *Proc. 4th International Conference on Intelligent Systems, Modelling and Simulation*, Bangkok, 2013, pp. 534-538.
[17] M. D. Mauro and C. D. Sarno, "A framework for internet data real-time processing: A machine-learning approach," in *Proc. International Carnahan Conference on Security Technology*, Rome, 2014, pp. 1-6.
[18] G. D. F. Morales and A. Bifet, "SAMOA: Scalable advanced massive online analysis," *J. Mach. Learn. Res.*, vol. 16, pp. 149–153, February 2015.
[19] W. Yang, X. Liu, L. Zhang, and L. T. Yang, "Big data real-time processing based on storm," in *Proc. 12th IEEE Int. Con. Trust. Sec. Priv. Comp. Comm.*, Melbourne, 2013, pp. 1784–1787.
[20] J. Bai, "Feasibility analysis of big log data real time search based on Hbase and ElasticSearch," in *Proc. Ninth International Conference on Natural Computation*, Shenyang, 2013, pp. 1166-1170.
[21] M. Bakır, B. Aydoğan, M. Aydın, A. Khodabakhsh, İ. Arı, and A. Ö. Ercan, "Real-Time data reconciliation solutions for big data problems observed in oil refineries," in *Proc. Signal Processing and Communications Applications Conference (SIU)*, Trabzon, 2014, pp. 1612–1615.
[22] C. He, Y. Lu, and D. Swanson, "Real-Time scheduling in MapReduce clusters," in *Proc. 10th Int. Conf. High Perform. Comput. Commun. & 2013 IEEE Int. Conf. Embed. Ubiquitous Comput.*, Zhangjiajie, 2013, pp. 1536–1544.
[23] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Inf. Sci.*, vol. 278, pp. 488–497, September 2014.
[24] Tutorial - Apache Hive - Apache Software Foundation. [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/Tutorial
[25] Apache Mahout: Scalable machine learning and data mining. [Online]. Available: https://mahout.apache.org/users/basics/algorithms.html

Sinan AY received the BS degree from Kocaeli University, Department of Computer Engineering in 2009. He received MSc degree Computer Engineering Department, Yalova University in 2013. He is currently PhD Candidate at Department of Computer Engineering, Gazi University. His research interests include data mining and big data analysis.

M. Ali Akcayol received the BS degree in Electronics and Computer Systems Education from Gazi University in 1993. He received MSc and PhD degrees in Institute of Science and Technology from Gazi University in 1998 and 2001, respectively. His research interests include Mobile Wireless Networking, Web Technologies, Web Mining, Cloud Computing, Artificial Intelligence, Intelligent Optimization Techniques, Hybrid Intelligent Systems.