VERİ BİLİMİ DERGİSİ
www.dergipark.gov.tr/veri

The Effect of Asynchronous Message Queues on the Communication of IoT Devices

Ahmet TOPRAK[1]*, Abdül Halim ZAİM[2], Feyzanur Sağlam TOPRAK[2]

[1]Istanbul Commerce University, Computer Engineering, Istanbul, TURKEY
[2]Istanbul Commerce University, Computer Engineering, Istanbul, TURKEY
[2]Istanbul Commerce University, Computer Engineering, Istanbul, TURKEY

Abstract

Nowadays, IoT (Internet of Things) devices have reached quite high numbers. This situation brought with it the presence of high density, unstructured data. The difficulties in obtaining, processing, storing and visualizing this data made it necessary to use the components of the big data system. High density data should be taken from IoT devices instantly, meaningful data should be obtained from these unstructured data, the meaningful data should be stored and presented at the request of the user when needed. In this article, a model is designed to process the data obtained from IoT devices and transmit them instantly to the end user. In the study, unstructured data collected primarily from IoT devices were subjected to data pre-processing steps. Meaningful words were determined from the data obtained after the data pre-processing steps. For this purpose, the Helmholtz Principle has been applied. After meaningful word detection, it is directed to both RabbitMQ and IBM MQ separately to instantly process data on the subject of each meaningful word. Apache Storm topology was used to instantly receive and process the messages transmitted to the queues. According to the results obtained, messages sent to the IBM MQ are consumed 30% faster than the RabbitMQ.

Makale Bilgisi

Başvuru:
06/08/2021
Kabul:
04/11/2021

Keywords: *Apache Storm, Big Data, Elasticsearch, Hadoop, Helmholtz Principle, Ibm Mq, Internet of Things, RabbitMq*

***** İletişim e-posta: ahmetoprak190363@gmail.com

1 Introduction

Nowadays, the amount of data produced with the developing technology has increased significantly. This makes it difficult to access information and various methods are used to make sense of the data, if possible, to be classified into classes, and to facilitate use and reporting. At this point, the big data concept enters our lives. Big data is the form of all this data that we obtain from different sources such as social media shares and photo archives, which has been transformed into a meaningful and processable form. In other words, social media, search engines, information document archives, log files, machine data is a phenomenon brought forward by software companies that carry out R&D (Research and Development) activities to process the information stacks created by different sources into meaningful data.

It is very important today to make sense of unstructured data collected from different sources. However, this is not as easy as it seems. For this purpose, different text mining methods such as information retrieval, lexical analysis, word requirement distribution, pattern recognition, tagging, information extraction, data mining and visualization are used. By using these techniques, classification and segmentation of texts, extracting topics from texts, sentiment analysis, creation of text summarization systems, etc. works are being done [1]. In Figure 1, the text mining techniques that they use while going through the training, testing and validation stages of the data to be used in text mining studies are given.

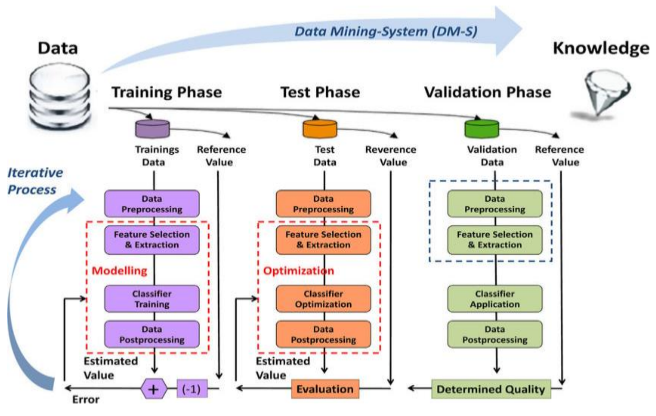


Fig. 1. Data mining process [2]

In this study, a workflow is proposed and experimentally tested in order to process the data collected from IoT devices in real time. First of all, data from IoT devices were taken in an unstructured format. The data was subjected to pre-processing steps in order to bring these unstructured data into uniform order. Helmholtz Principle was used to determine the weight of the words obtained after data pre-processing. Then, using these words, RabbitMQ and IBM MQ were directed to distributed message queues according to their meaning weights.

It is directed to different RabbitMQ and IBM MQ queues according to the topics of the weighted words. Apache Storm was used to instantly process the data transmitted to the message queues. Apache Storm, by analyzing the data that has been passed through data pre-processing and monotonous data, instantly records its results in the Elasticsearch database. After these processes, REST API was used to extract the processed data from Elasticsearch database and present it to the user.

The purpose of this study is to scientifically analyze the data storage and processing performance of RabbitMQ and IBM MQ distributed systems during instantaneous real-time processing of data collected from different sources.

In the second part of the article, the studies covering the steps that are similar or used in this study are mentioned. In the third section, the general system topology is explained in detail. In the fourth section, the topology steps mentioned in the third section are explained, in the last section, the results and future studies are mentioned.

2 Related Works

Studies on big data, Hadoop, HDFS and MapReduce have been carried out from past to present and these studies have included different analysis and techniques until today. One of these works is the 2014 work of Shankar Ganesh Manikandan and his team [3]. In this study, various methods for the problems at hand are proposed through the MapReduce framework over HDFS. A minimization technique using map indexing, sorting, mixing and finally reduction and file indexing has been developed using MapReduce. This technique has been studied on large unstructured data from different sources. According to the study, it is seen as a promising approach to reach the desired analytics with MapReduce in the processing and interpretation of these data obtained from different sources.

One of the works carried out to analyze large-scale data is Mrunal Sogodekar's and his team's 2016 [4] study. In this study, it is aimed to analyze the data collected from sources such as e-mail, social media

(Facebook, Twitter etc.). Especially, it is aimed to examine the user comments of the web pages that are visited extensively and make inferences from these comments. For this purpose, different data analytical tools were used and their properties were compared against each other.

[4] is a study similar to that of Kamalpreet Singh and his team in 2014 [5]. In this study, Hadoop MapReduce and HDFS structure are discussed in detail. It is stated that Hadoop is the most sought-after platform because it uses Cloud-based, open source and applications running mostly on Linux operating system.

One of the works on MapReduce is the 2016 work of Ankush Verma and his team [6]. In this study, Hadoop MapReduce and Spark structures are explained in detail and the differences between them are gradually stated. In the study, the following inferences are made. Hadoop MapReduce operates by analyzing large amounts of data across multiple nodes in parallel. However, MapReduce has two functions: map and reduce, big data is stored on HDFS. Due to the lack of this possibility in MapReduce, Spark is designed to run for real-time streaming data and quick queries. The Spark works on durable distributed datasets and a directed circular graphics execution engine. For the stated reasons, the study was shown one step ahead of Spark MapReduce.

In their 2013 work [7], Seref Sagiroglu and his team talk about very important basic information on big data. This paper presents an overview of big data's content, scope, samples, methods, advantages and challenges and discusses privacy concern on it.

One of the main areas with big data is the health area. Gaspard Harerimana and his team [8] also addressed this issue in their work. This study explores key challenges, data sources, techniques, technologies and future trends in big data analytics in healthcare. A do-it-yourself review is presented, offering a holistic, simplified and easily understandable view of the various technologies used to develop an integrated health analytics application.

One of the studies on big data in the field of health is the study [9]. This did not work, the general structure of processing, storing and obtaining the real-time data flowing to the information systems in the hospitals was mentioned, and then the distressed parts in this structure were explained in detail. At the end of the study, some suggestions were made for the studies that should be done for these problems.

The processing of big data in live systems is usually accomplished using distributed message queues. Because the living system has to be in a continuous operating and scalable structure. A large number of studies have been conducted on processing large data in distributed message queues. One of these studies is [10]. In this study, a general distributed messaging framework is presented for online transaction processing applications in the e-commerce industry. This messaging system is called Metamorphosis (MetaQ). This system was later deployed and used on Taobao.com and Alipay.com. According to the results obtained, the actual uses in both types of online trading applications have proven that the nature of MetaQ can perform well for such large Internet applications.

3 Designed Model

The data collected from IoT devices are passed through the pre-processing step before they are included in the system. For the words that pass the pre-processing step, word weighting is done later. It is then routed to the respective RabbitMQ and IBM MQ distributed systems according to their subject. These messages are then retrieved from the queue by Apache Storm and processed instantly. Finally, the messages are saved in the Elasticsearch database. The methods applied are described below. The current general system schema is given by Figure 2.

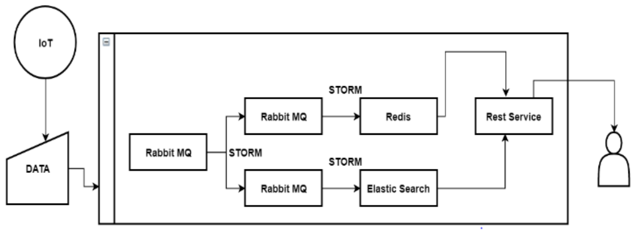


Fig. 2. General structure of current experimental system

3.1 Pre-processing of documents

The most important point of obtaining successful results in text mining and natural language processing studies is that the data to be used must be of high quality. Usually, before the data is put into practice, it consists of dirty parts that do not have a certain standard. In order to ensure the quality of these data, the data is subjected to a data pre-processing step, which includes certain standards.

Data pre-processing processes include different types and methods. They are data cleaning, data consolidation, data conversion and discretization, data reduction techniques. For this reason, the data should be analyzed effectively and the data pre-processing type should be determined correctly [11]. Different steps involved in data pre-processing are given by Figure 3 [12].

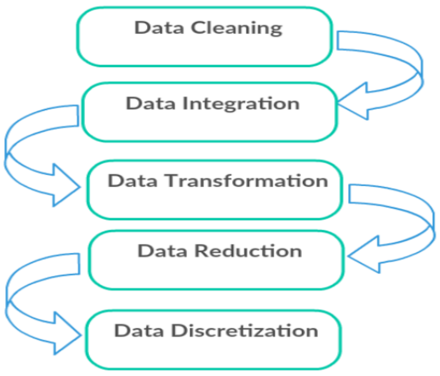


Fig. 3. Data pre-processing techniques

3.2 Weighting words

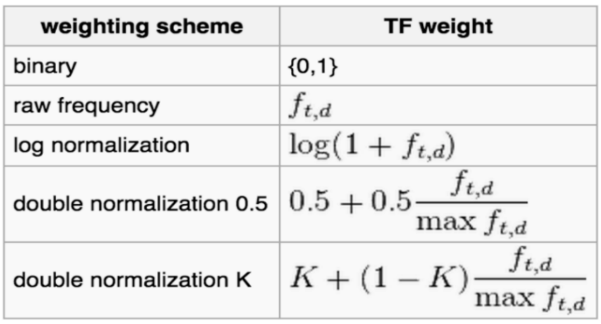
After the given data collected from IoT devices with signals are subjected to the pre-processing step, the data obtained are sent to RabbitMQ and IBM MQ message queues according to their subject. The purpose of this process is that when we send all processes to the same queue, a bottleneck will occur and the whole process will not be interrupted when the load is placed on the system or the system crashes for a different reason, only the related message queue will be disabled.

Keyword selection is applied in order to determine the meaningful words after pre-processing. There are different keyword selection methods. Helmholtz-based Gestalt Human perception theorem [13;14], and TF-IDF [15], metrics are the most used.

In the study, the documents presented as seed or obtained as a result of web search are subjected to preprocessing step. Then, meaningful words are determined from these words. TF-IDF metric is used for the determination of meaningful words. The top n words with the highest TF-IDF meaning value are added to the dictionary and then searched on the web via these words added to the dictionary.

TF-IDF metric is the weighting factor calculated by statistical methodology, which shows the importance of a term in a document. It is used for statistical analysis on the texts which are the common application of natural language processing and text mining. It is also often referred to as a ranking algorithm under topics such as information retrieval.

Term Frequency (TF): A method used to calculate term weights in a document. The methods used to calculate the weight with TF are described in Figure 4.



weighting scheme	TF weight
binary	{0,1}
raw frequency	$f_{t,d}$
log normalization	$\log(1+f_{t,d})$
double normalization 0.5	$0.5+0.5\frac{f_{t,d}}{\max f_{t,d}}$
double normalization K	$K+(1-K)\frac{f_{t,d}}{\max f_{t,d}}$

Fig. 4. Term frequency weight calculation methods

Inverse Document Frequency (IDF): Measures the rank of the specific word for its relevancy within the text. Stop words which contain unnecessary information such as "a", "into" and "and" carry less importance in spite of their occurrence.

In this study, formulas (1) and (2) were used for TF and IDF calculations.

n: Total number of documents
t: Number of documents containing the term
f: Number of the term in the document
d: Total number of terms in the document

$$\mathrm{TF} = \frac{\mathrm{f}}{\mathrm{d}} \tag{1}$$

$$\mathrm{IDF} = \log\frac{\mathrm{n}}{\mathrm{t}} \tag{2}$$

To explain the TF-IDF metric through the example;

Let's have a 10000-word document named "X1". Assuming that the word "ICONDATA" is mentioned 100 times in this document, the TF value of the word "Computer" is calculated as follows.

$\mathrm{TF} = 100/10000 = 0,01$

We have 20 documents. Assuming that 10 of these documents refer to the word " ICONDATA", the IDF value of the word "ICONDATA" is calculated as follows.

$\mathrm{IDF} = \log(20/10) = 0,30$

The TF-IDF calculation is equal to the multiplication of the TF and IDF values.

TF-IDF = 0, 01 * 0, 30= 0, 003

3.3 Assigning Data to the Message Queue

Data weighted with TF-IDF metrics were sent to RabbitMQ and IBM MQ distributed system according to their subjects.

RabbitMQ is an increasingly popular messaging queue structure in today's server-to-server / app-to-app communication needs, developed as an open-source with the Erlang language and built on the Open Telecom Platform library. By implementing Advanced Message Queuing Protocol (AMQP), it enables data exchange between applications and servers [16].

IBM WebSphere MQ (WMQ), previously known as MQSeries, is a MOM (Message Oriented Middleware), a message-based database type product, which IBM company acquired in 1992. This product became widespread towards the end of the 90s, in the data transfer between companies. The data transfer principle is based on "Message Queueing". Applications the WMQ interface writes messages to queues managed by MQI and queue manager, and the queue manager forwards this message to other queue managers. The other application reads these messages and adds them to the database. WMQ has become a widely used communication application in recent years thanks to EAI (Enterprise Application Integration), a kind of general communication base and SOA (Service Oriented Architecture).

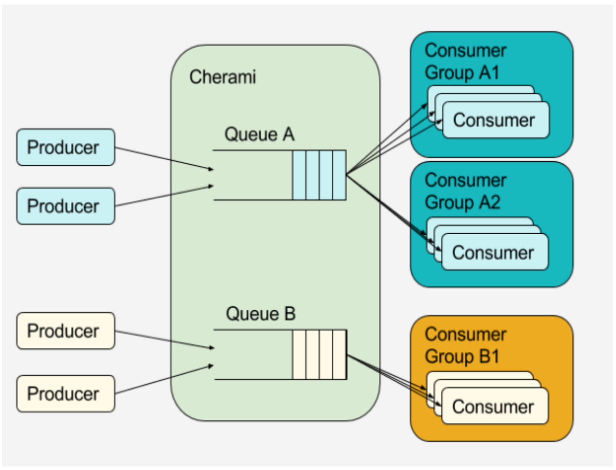


Fig. 5. MQ life cycle

The server sends a message to the RabbitMQ and IBM MQ server, and the server forwards it to the corresponding queue. Another application then listens to this queue and ends the process by consume these messages in the queue that works with FIFO (First in First Out) logic. It allows us to view many queues, requeue, delete, purge, etc., with its web management interface.

3.4 Real-time Processing of Data

Apache Storm infrastructure was used in this study to analyze the data sent to RabbitMQ and IBM MQ message queues instantly after data pre-processing steps. Apache Storm is a free and open source distributed real-time computing system [17]. It provides the convenience to reliably process unlimited data streams for real-time processing of what Hadoop does for batch processing. It is not complicated to use. It can also ensure guaranteed processing of data, with the ability to replay unprocessed data successfully the first time.

Apache Storm is used for distributed machine learning, real-time analytics systems and many other situations, especially with high data rate. It runs on the YARN package manager and is integrated with Hadoop ecosystems as well. It is a stream processing engine, rather than a sequence in small batches. It has low latency and is well suited for data that needs to be received as a single entity. It is also a bridge between batch and stream processing. Apache Storm processes one million hundred bytes msgs / sec / node messages and performs these operations very securely. It guarantees that each tuple will be processed once. The messages are repeated only when there is an error [18]. However, it is not designed to run on the Hadoop ecosystem.

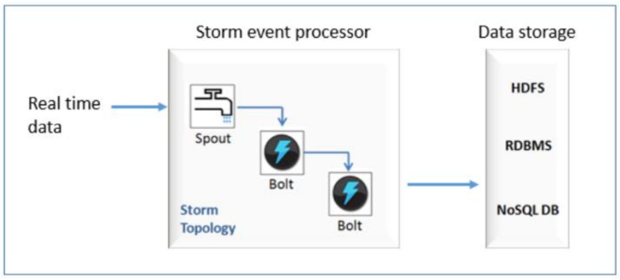


Fig. 6. Apache Storm working life cycle [19]

3.5 Saving the Processed Data

After the data in MQ message queues are processed in real time, this data has to be saved in order not to be lost. In our study, Elasticsearch database was used for this purpose. Elasticsearch is an open source, full text search engine and data analysis tool built on Apache Lucene, developed in Java. The data storage format is documented oriented, not relational [20].

Elasticsearch database; It produces results very quickly by searching through indexes instead of searching directly through text. Indicates in which document a word is passed while the data is being recorded. Then, when we want to search for words, instead of searching all the data, the results are quickly found through the index list created earlier.

In Figure 7, the way data is stored in Elasticsearch database is shown.

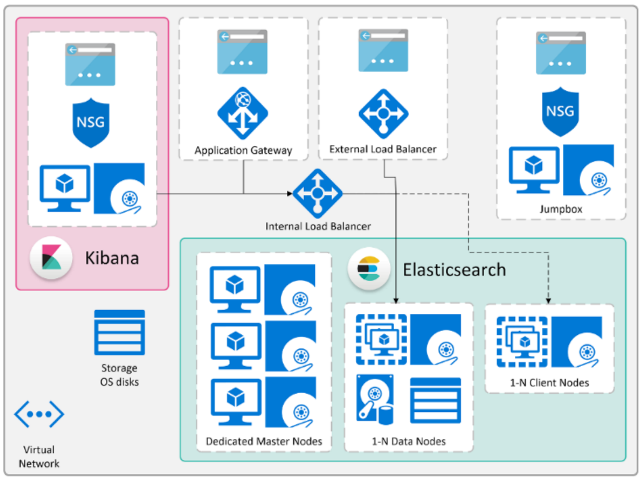


Fig. 7. Elasticsearch database topology [21]

The advantages provided by Elasticsearch are as follows.

- The cluster structure is very simple.
- It does not consume a lot of resources compared to its competitors.
- It offers high availability in itself.
- Searches fast because it has indexing logic.
- The concept of document and indexing is widely used.
- It indexes documents in JSON (JavaScript Object Notation) format.

3.6 Saving the Processed Data

After the processed data is saved in the Elasticsearch database, this data should be transmitted to the end user. It is very important to perform this process extremely quickly and safely. For this purpose, REST API was used in this study. The data in the Elasticsearch database are presented to the end user using the relevant API.

REST stands for Representational State Transfer. Representational State Transfer, or Representative State Transfer, is a simple data communication path between client and server, using HTTP (Hypertext Transfer Protocol) methods, with a flexible structure [22]. It is an architectural style rather than a protocol with strict rules. REST API is stateless. In other words, it is unaware of current user status or history. In the data carried between the client and server in REST standards, no extra header information is stored, the details of the client are not found, this information is not carried between the client and server. Therefore, REST offers us a lightweight solution structure in service oriented applications. REST API features are as follows.

REST and SOAP services are often confused with each other. The differences between REST and SOAP services are shown in Figure 8.

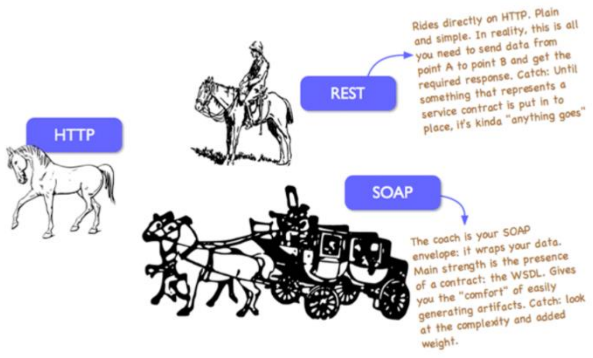

HTTP
REST
Rides directly on HTTP. Plain and simple. In reality, this is all you need to send data from point A to point B and get the required response. Catch: Until something that represents a service contract is put in to place, it's kinda "anything goes"
SOAP
The coach is your SOAP envelope: it wraps your data. Main strength is the presence of a contract: the WSDL. Gives you the "comfort" of easily generating artifacts. Catch: look at the complexity and added weight.

Fig. 8. Differences between REST and SOAP services [23]

- REST API is an architecture related to client-server communication.
- REST API operates over HTTP protocol instead of complex architectures like SOAP, RPC.
- REST API is simpler and faster than SOAP.
- REST API does not have strict standards like SOAP.
- Of course, while providing security on SOAP can be achieved more easily and quickly due to strict rules, the process can be complicated for REST.
- REST API does not need to use proxy such as SOAP and WSDL (Web Services Description Language).
- SOAP incorporates security protocols and stores state information in requests and responses.

REST API working life cycle is given in Figure 9.

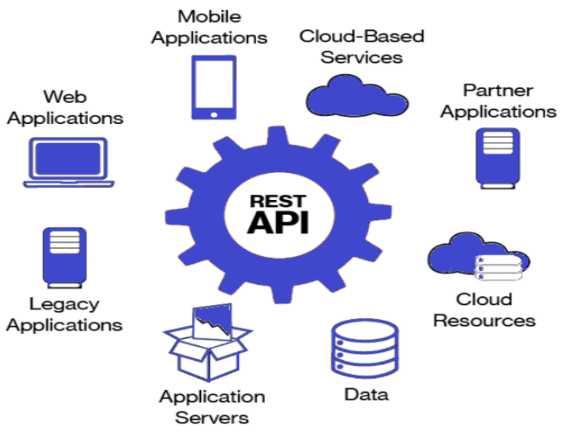


Fig. 9. REST API life cycle topology [24]

4 Experiments

In this section, performances of 3 different size message sets and RabbitMQ and IBM MQ message queues are compared. The messages were added to the queue by prioritizing FIFO in all experiments. In order to consume the messages added to the queue, Windows services hosted on the server have been created. In order to process the messages faster, experiments have been carried out on different numbers of consumers. The results will be discussed in detail.

4.1 Experiment I

In Experiment I, 1000 messages were sent to both the RabbitMQ and the IBM MQ distributed system. Then, these messages were tried to be consumed with 10,100,1000 consumers. The messages were sent to the message queues with FIFO logic. The processing life of the messages is marked as 3 seconds.

The processing times of RabbitMQ and IBM MQ messages obtained using different number of consumers are given in Figures 10, 11, 12, respectively.

Parameter Name	Parameter Value
Message Count	1000
Consumer Count	10
Total Processing Time (sec) - RabbitMQ	3
Total Processing Time (sec) -IBM MQ	1

Fig. 10. Processing of 1000 messaging data with 10 consumers

Parameter Name	Parameter Value
Message Count	1000
Consumer Count	100
Total Processing Time (sec) - RabbitMQ	1.3
Total Processing Time (sec) -IBM MQ	0.8

Fig. 11. Processing of 1000 messaging data with 100 consumers

Parameter Name	Parameter Value
Message Count	1000
Consumer Count	1000
Total Processing Time (sec) - RabbitMQ	1
Total Processing Time (sec) -IBM MQ	0.2

Fig. 12. Processing of 1000 messaging data with 1000 consumers

According to the results obtained in Experiment I, the processing time of 1000 messages was much shorter in IBM MQ than in RabbitMQ. This rate increased up to 5 times in some experiments. This is due to the fact that IBM MQ systems are cluster and the listener structure is stronger than RabbitMQ. At the same time, the speed of both distributed message systems increased as the amount of consumers increased in all experiments.

4.2 Experiment II

In Experiment II, 10000 messages were sent to both the RabbitMQ and the IBM MQ distributed system. Then, these messages were tried to be consumed with 10,100,1000 consumers. The messages were sent to the message queues with FIFO logic. The processing life of the messages is marked as 3 seconds.

The processing times of RabbitMQ and IBM MQ messages obtained using different number of consumers are given in Figures 13, 14, 15, respectively.

Parameter Name	Parameter Value
Message Count	10000
Consumer Count	10
Total Processing Time (sec) - RabbitMQ	72
Total Processing Time (sec) -IBM MQ	21

Fig. 13. Processing of 10000 messaging data with 10 consumers

Parameter Name	Parameter Value
Message Count	10000

Consumer Count	100
Total Processing Time (sec) - RabbitMQ	64
Total Processing Time (sec) -IBM MQ	16

Fig. 14. Processing of 10000 messaging data with 100 consumers

Parameter Name	Parameter Value
Message Count	10000
Consumer Count	1000
Total Processing Time (sec) - RabbitMQ	60
Total Processing Time (sec) -IBM MQ	16

Fig. 15. Processing of 10000 messaging data with 1000 consumers

According to the results obtained in Experiment II, the results are in parallel with the results obtained in Experiment I. The most striking point in this experiment is that when the number of consumers exceeds 1000, the effect level decreased. This shows that excessive branching in multi-tasking processes will not have positive results. In the investigations carried out to determine the cause of this problem, the following situations stand out. When the number of threads exceeds a certain threshold value (100 in this experiment), the thread passes to wait status and wait for each other. This situation has caused deadlocks.

4.3 Experiment III

In Experiment III, 100000 messages were sent to both the RabbitMQ and the IBM MQ distributed system. Then, these messages were tried to be consumed with 10,100,1000 consumers. The messages were sent to the message queues with FIFO logic. The processing life of the messages is marked as 3 seconds.

The processing times of RabbitMQ and IBM MQ messages obtained using different number of consumers are given in Figures 16, 17, 18, respectively.

Parameter Name	Parameter Value
Message Count	100000
Consumer Count	10
Total Processing Time (sec) - RabbitMQ	120
Total Processing Time (sec) -IBM MQ	47

Fig. 16. Processing of 100000 messaging data with 10 consumers

Parameter Name	Parameter Value
Message Count	100000
Consumer Count	100
Total Processing Time (sec) - RabbitMQ	83
Total Processing Time (sec) -IBM MQ	29

Fig. 17. Processing of 100000 messaging data with 100 consumers

Parameter Name	Parameter Value
Message Count	100000
Consumer Count	1000
Total Processing Time (sec) - RabbitMQ	81
Total Processing Time (sec) -IBM MQ	32

Fig. 18. Processing of 100000 messaging data with 1000 consumers

When the results obtained in Experiment III were analyzed, the processing time of these messages took longer due to the 1000 times higher number of messages than the other experiments. In this experiment, the processing time of both distributed systems increased. As in the other two experiments, the results obtained with IBM MQ were better than RabbitMQ in this experiment. It has been observed that the experiments using more than 100 consumers detected in Experiment II did not yield successful results. One of the remarkable points in this experiment was that it was transmitted to error queue without being processed because the size of some messages is more than the default value (5 MB). To avoid losing this situation and the message sent to queue due to an error for any reason, consumer must open a transaction block before receiving each message. When it is certain that the message has been sent to the recipient, session must be commit. If an error occurs while sending the message to the recipient, session rollback should be done.

5 Results and Future Studies

In this study, a model is proposed to transmit the data generated by IoT devices to the end user in real time. All technologies used in this study are today's most popular products. In this respect, this study differs from other studies. There have been cases where we could not use these products directly in the experiments. For this reason, special solutions have been produced on APIs. According to the results obtained from the experiments, IBM MQ performs message transmission up to 5 times faster than RabbitMQ. RabbitMQ, on the other hand, can instantly distribute 1 million data to queues. However, it is necessary to purchase institutional packages for more messages. It has been observed that when more than 1 million messages are distributed to the queues, it negatively affects the system performance. In IBM MQ, the situation had the opposite effect. The system works stably even when it receives more than 1 million messages. However, when the number of consumers processing the messages exceeds 100, there were negative effects on both distributed message queues. This reason is that deadlocks occur between threads. In addition, it has been observed that the warranty message processing feature offered by Apache Storm, together with the wrongly designed topology structure, can reduce the system to a state that cannot produce output. The data stored in the Elasticsearch database has been shown to be quite good in terms of speed since it is not structurally captured. When Elasticsearch database was disabled and replaced with a structural database (MS SQL), the results were seen to be quite bad.

The following studies, which are thought to contribute to the literature in the future, can be discussed.

- The Helmholtz Principle can be used to weight data obtained from IoT devices.
- MQTT (Message Queuing Telemetry Transport) asynchronous communication system can be used instead of RabbitMQ communication system.
- Apache Hadoop can be used instead of Apache Storm, which is used to provide real-time processing of data.
- Redis database can be used instead of Elasticsearch database where the processed data is saved.
- SOAP services can be used instead of the REST API used to transmit the processed data to the end user.

References

[1] Lausch, Angela & Schmidt, Andreas & Tischendorf, Lutz. (2015). Data mining and linked open data – New perspectives for data analysis in environmental research. Ecological Modelling. 295. 5-17. 10.1016/j.ecolmodel.2014.09.018.

[2] Shaqiri, Bledi. (2017). Exploring Techniques of Improving Security and Privacy in Big Data. 10.13140/RG.2.2.23201.10089.

[3] S. G. Manikandan and S. Ravi, "Big Data Analysis Using Apache Hadoop," 2014 International Conference on IT Convergence and Security (ICITCS), Beijing, 2014, pp. 1-4, doi: 10.1109/ICITCS.2014.7021746.

[4] M. Sogodekar, S. Pandey, I. Tupkari and A. Manekar, "Big data analytics: hadoop and tools," 2016 IEEE Bombay Section Symposium (IBSS), Baramati, 2016, pp. 1-6, doi: 10.1109/IBSS.2016.7940204.

[5] K. Singh and R. Kaur, "Hadoop: Addressing challenges of Big Data," 2014 IEEE International Advance Computing Conference (IACC), Gurgaon, 2014, pp. 686-689, doi: 10.1109/IAdCC.2014.6779407.

[6] A. Verma, A. H. Mansuri and N. Jain, "Big data management processing with Hadoop MapReduce and spark technology: A comparison," 2016 Symposium on Colossal Data Analysis and Networking (CDAN), Indore, 2016, pp. 1-4, doi: 10.1109/CDAN.2016.7570891.

[7] S. Sagiroglu and D. Sinanc, "Big data: A review," 2013 International Conference on Collaboration Technologies and Systems (CTS), San Diego, CA, 2013, pp. 42-47, doi: 10.1109/CTS.2013.6567202.

[8] G. Harerimana, B. Jang, J. W. Kim and H. K. Park, "Health Big Data Analytics: A Technology Survey," in IEEE Access, vol. 6, pp. 65661-65678, 2018, doi: 10.1109/ACCESS.2018.2878254.

[9] A. R. Reddy and P. S. Kumar, "Predictive Big Data Analytics in Healthcare," 2016 Second International Conference on Computational Intelligence & Communication Technology (CICT), Ghaziabad, 2016, pp. 623-626, doi: 10.1109/CICT.2016.129.

[10] J. Liao, X. Zhuang, R. Fan and X. Peng, "Toward a General Distributed Messaging Framework for Online Transaction Processing Applications," in IEEE Access, vol. 5, pp. 18166-18178, 2017, doi: 10.1109/ACCESS.2017.2717930.

[11] F. Famili, W. Shen, R. Weber and E. Simoudis, “Data preprocessing and intelligent data analysis,” Intell. Data Anal, vol. 1(4), pp. 3-23, 1997. https://doi.org/10.1016/S1088-467X (98)00007-9

[12] V. Agarwal, “Research on data preprocessing and categorization technique for smartphone review analysis,” International Journal of Computer Applications, vol. 131(4), pp. 30-36, 2015. https://www.doi.org/10.5120/ijca2015907309

[13] B. Dadachev, A. Balinsky, H. Balinsky and S. Simske, “On the Helmholtz Principle for data mining,” Third International Conference on Emerging Security Technologies, pp. 99-102, 2012. https://www.doi.org/10.1109/EST.2012.11

[14] S. Jabri, A. Dahbi, T. Gadi and A. Bassir, "Ranking of text documents using TF-IDF weighting and association rules mining," 2018 4th International Conference on Optimization and Applications (ICOA), pp. 1-6, 2018. https://www.doi.org/10.1109/ICOA.2018.837057

[15] A.G. Jivani, "A comparative study of stemming algorithms," Int. J. Comp. Tech. Appl, vol. 2(6), pp. 1930-1938, 2011.

[16] Caner Tosun. "RabbitMQ Nedir? Windows Üzerinde Kurulumu". http://www.canertosuner.com/post/rabbitmq-nedir-windows-uzerinde-kurulumu (06.08.2021).

[17] Oguzhan İnan. "Hadoop Ekosistemi ve Kullanılan Araçlar". https://oguzhaninan.gitlab.io/Hadoop-Ekosistemi-ve-Kullanilan-Araclar/#what-is-storm (06.08.2021).

[18] Womaneng. "Kafka-Flink-Storm-Platformları". https://womaneng.com/kafka-flink-storm-platformlari/ (06.08.2021).

[19] Büyükveri. "Büyük Veri Ekosistemi". http://www.buyukveri.co/buyuk-veri-ekosistemi/ (06.08.2021).

[20] Devnot. "Bir Bakışta ElasticSearch". http://devnot.com/2017/bir-bakista-elasticsearch/ (06.08.2021).

[21] Elastic. "Elasticsearch and Kibana Deployments on Azure". https://www.elastic.co/blog/elasticsearch-and-kibana-deployments-on-azure

[22] Burcu Altınok. "REST API Nedir?" https://burcualtinok.com.tr/blog/rest-api-nedir/ (06.08.2021).

[23] Devnot. "REST Mimarisi ve RESTful Servisler". http://devnot.com/2016/rest-mimarisi-ve-restful-servisler/(06.08.2021).

[24] Deniz İrgin. "REST ve RESTful Web Servis Kavramı". https://denizirgin.com/rest-ve-restful-web-servis-kavram%C4%B1-30bc4400b9e0/(06.08.2021).