Hazelcast Vs. Ignite: Opportunities for Java Programmers

Bartkov Maxim [1], Katkova Tetiana [2], Kruglyk Vladyslav S.[3], Murtaziev Ernest G.[4], Kotova Olha V.[5]
[1]RooX Solutions Java Team Lead, Khakov, Ukraine, 1233566789b@gmail.com
[2]University of Customs and Finance, Dnipro, UkraineDepartment of Innovative Technologies,
Department of Cyber security, V.Vernadsky street 2/4, Dnipro, Ukraine, 49000, takit777@gmail.com
[3]Bogdan Khmelnitsky Melitopol State Pedagogical University, Faculty of Informatics, Mathematics and Economics, Department of Informatics and Cybernetics, Hetmanska Street, 20, Melitopol, Zaporizhia Region, Ukraine, 72300, kryglikvlad@gmail.com
[4] Bogdan Khmelnitsky Melitopol State Pedagogical University, Faculty of Informatics, Mathematics and Economics, Department of Mathematics and Physics, Hetmanska Street, 20, Melitopol, Zaporizhia Region, Ukraine, 72300, ernest_gaf@mail.ru
[5]Kherson State University, Faculty of Computer Science, Physics and Mathematics, Department of Algebra, Geometry and Mathematical Analysis, University Street, 27, Kherson, Kherson region, Ukraine, 73000, olga-kotova@ukr.net

Summary
Storing large amounts of data has always been a big problem from the beginning of computing history. Big Data has made huge advancements in improving business processes by finding the customers' needs using prediction models based on web and social media search. The main purpose of big data stream processing frameworks is to allow programmers to directly query the continuous stream without dealing with the lower-level mechanisms. In other words, programmers write the code to process streams using these runtime libraries (also called Stream Processing Engines). This is achieved by taking large volumes of data and analyzing them using Big Data frameworks. Streaming platforms are an emerging technology that deals with continuous streams of data. There are several streaming platforms of Big Data freely available on the Internet. However, selecting the most appropriate one is not easy for programmers. In this paper, we present a detailed description of two of the state-of-the-art and most popular streaming frameworks: Apache Ignite and Hazelcast. In addition, the performance of these frameworks is compared using selected attributes. Different types of databases are used in common to store the data. To process the data in real-time continuously, data streaming technologies are developed. With the development of today's large-scale distributed applications handling tons of data, these databases are not viable. Consequently, Big Data is introduced to store, process, and analyze data at a fast speed and also to deal with big users and data growth day by day.

Keywords: *Big Data, Stream processing framework, IMDG, Hazelcast, Ignite.*

1. Introduction

Storing large amounts of data has always been a big problem from the beginning of computing history. Consequently, several databases and architectures were invented to store data, in which database management systems, relational databases (RDBMS), and similar architecture models are commonly used [1]. Relational databases (such as SQL as a standard database) which got popular in the 1980s are still in use today. Although relational databases deal with large amounts of structured data, they cannot fulfill the requirements of the present day's software applications such as scalability, cloud storage, big users, and big data [1, 2]. To eliminate the drawbacks of traditional models and to cater to the needs of new applications, the concept of Big Data was introduced in the 90s that became popular in the 2000s. Big Data is related to storing, collecting, analyzing, and processing large volumes of stream data (structure, unstructured) at high speed [2, 3]. The data is collected from different sources: web pages, social media, and the Internet of Things [4]. Other features of Big Data are dealing with thousands of users at a time (hence termed as Big user), storing data on the cloud, the large volume of data, high speed, variety of data, maintaining the quality of data [1, 2, 5].

Big Data is utilized in other fields such as Artificial Intelligence (AI), Machine Learning (ML) and Internet of Things (IoT) to analyze, predict and make intelligent decisions on large volumes of data. AI is impractical without data and analyses on data are not possible without AI. The combination of AI and Big Data opened new application areas such as, early prediction of fire risk in smart cities [6], air pollution monitoring [7], keeping track of health using mobile apps [8], etc. Likewise, Machine Learning allows computers to automatically learn from the data which is not programmed; stores a bulk of data in Big Data and makes decisions based on the data [4, 9]. Google and Netflix are simple examples of this category that predict what the people want to search, based on the existing search datasets. Similarly, the prediction of cardiovascular risks in a health care system based on the existing datasets is another example [9]. Another source of data collection for Big Data is IoT, which connects physical devices with the Internet [4]. According to a study in the year 2020, billions of IoT devices are connected with the Internet, generating about a trillion of data in Big Data [10]. Handling this tremendous amount of data, using the traditional methods and techniques, is not possible for the programmers.

To make the life of a programmer easy, several technological frameworks are developed each with a distinct purpose. These frameworks provide tools, classes,

Manuscript received February 5, 2022
Manuscript revised February 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.2.52

libraries, and functions to programmers for easy development of their application(s). In addition, these frameworks provide an abstraction to the application programs by interacting directly with the input/output devices, system software, network connections, and so on. Some examples of famous frameworks include .NET, Spring, Hadoop, etc. The .NET framework from Microsoft provides classes and libraries for developing standard window-based applications. Similarly, the Spring framework provides standards for developing Java applications. Some frameworks provide guidelines, for example, the resource description framework describes how the internet resources can be retrieved by the rules of the World Wide Web. Other frameworks are used to perform specific tasks, e.g., Hadoop, developed by Apache software foundation, is open access distributed framework used for storing and processing Big Data. As selecting the right technological framework impacts the application development in several ways, some guidelines for the programmers are needed.

With the rapid generation of a huge amount of data, it is now necessary to process the data in stream mode instead of batches [11]. To this end, specific frameworks are also available which facilitate the processing of data streams in Big Data. In fact, frameworks are available for each of the phases (which are data ingestion, data processing, and analysis and evaluation) involved in the analysis of data streams. Some well-known data streaming frameworks are Flink, Storm, Kafka, and Samza (all developed by Apache). Selecting the most suitable framework from this set, which can help a Java programmer to write code to process streaming data is very challenging. Towards this direction, some comparisons of the existing frameworks can be found in the literature, such as between Spark and Flink [12], and between Spark, Storm, and Samza [13], etc. However, these comparisons are either considering only one property during comparison [12] or considering a particular use case [13] or benchmark for comparison. More importantly, a comparison of the two most famous frameworks i.e., Hazelcast and Ignite, is not available to the best of the authors' knowledge.

To fill this gap, this paper provides insights into these two frameworks for the guidance of Java programmers. Apache Ignite and Hazelcast are selected in this work because they are free from scalability, fault tolerance, and latency issues, which are present in other streaming platforms. The comparison between these two frameworks would not only help individual Java programmers to differentiate them and select the most appropriate but would also allow companies/businesses to know about them and use them to boost up their applications. This paper is organized as follows: the famous streaming frameworks used in big data are discussed briefly in the next section to set the background, along with the descriptions of Hazelcast and Ignite. Section III provides the related work regarding the comparison of big data streaming frameworks. The comparison between Apache Ignite and Hazelcast based on the selected attributes is provided in Section IV. The paper is concluded in the last Section V.

2. Big Data streaming frameworks

The main purpose of big data stream processing frameworks is to allow programmers to directly query the continuous stream without dealing with the lower-level mechanisms. In other words, programmers write the code to process streams using these runtime libraries (also called Stream Processing Engines).

2.1. Popular Frameworks from Apache

Apache Hadoop is an environment that can use a simple programming model to process large data sets distributed among groups of computers. It is designed to scale from a single server to thousands of machines, each machine providing local computing and storage. It is based on the popular Map Reduce model and is the key to developing robust, scalable, and distributed software applications. Apache Spark is another very popular big data framework, and its demand is growing every day. Spark is a fast data memory handler. The programmers need to use a smooth and expressive development API to enable data workers to efficiently run streaming, machine learning, or SQL workloads that require fast iterative access to data sets.

Apache Hive is a big data analysis framework developed by Facebook that combines the scalability of one of the most popular big data frameworks. It can be considered as a data processing tool for Hadoop. It is basically a query tool for HDFS, and its query syntax is almost the same as traditional SQL. Apache Hive is open-source software that programmers can use to analyze large data sets in Hadoop. It is an engine that converts SQL queries into MapReduce task chains. Apache Flink is another powerful open-source distributed big data stream processing framework in which processing is performed in batches. It is the successor of Hadoop and Spark. It is the next-generation big data engine for stream processing. Unlike Spark, (which is not a real threading framework but just an improvisation), Apache Flink is a real threading engine with additional capabilities for processing batches, processing graphs, and tables, and running machine learning algorithms. One of the limitations of this framework is that it doesn't provide scalability [14].

Apache Kafka is another open-source framework that provides scalability and fault tolerance. Compressing and decompressing data in Apache Kafka decrease its performance and output [14]. Apache Samza processes the messages in one stream and produces the output on another stream. It combines the advantages of Kafka and

YORN, as it uses Kafka's architecture and messaging system and YORN's resource negotiation. It offers reliable persistency due to replicated storage and it is highly available, easy, and inexpensive (as it follows publish/subscribe model). However, it does not support low latency, fault tolerance, languages other than JVM, and exactly-once semantics [14].

Apache Storm is designed for real-time stream processing, just like Hadoop is used for batch processing. The framework focuses on processing large data streams in real-time. This implies that programmers can create applications that are very sensitive to the latest data and require to react within seconds. For example, finding the latest hot topics on Twitter or following payment gateway failures. Apache Storm is suitable for real-time processing as it provides results with very low latency. Other main features of Storm are scalability and rapid recovery from downtime. In addition, it is easy to set up and operate [14].

2.2. Apache Ignite

Apache Ignite is distributed in-memory database that processes big data. An In-Memory Database (IMDB) is different from traditional databases as it stores data in the main memory rather than storing it in the disk. As main memory is volatile so the data is also stored on the disk to minimize the risk of data loss [15]. Apache Ignite provides several advantages to an application over previous frameworks such as increased memory capacity and introduction of GPU for fast computing and performance. Apache Ignite is also suitable for transactional operations [16].

In-Memory Data Grid (IMDG) copies the disk data, such as RDBMS, etc. in the main memory.

IMDG works on top of the databases that are used to read and write through cache. It is installed on a virtual machine and has cloud storage. Additionally, it can also deal with the IoT and machine learning workload. Furthermore, the grid supports various API and key values. Applications do read and write operations from the grid and then the grid changes the data value accordingly [17-19]. To increase the performance of IMDG, Apache Ignite is placed between the database layer and application layer as shown in Fig. 1. Through it, multiple APIs and distributed databases can be used [17].

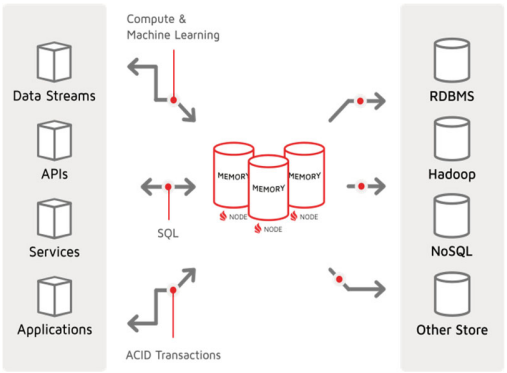

Fig. 1. The architecture of Apache Ignite

Apache Ignite is a memory platform that offers the following advantages:

- It supports both IMDB and IMDG.
- It provides the data loading and data streaming capabilities through which a finite amount of data can be transferred in a scalable and fault-tolerant manner.
- It can combine with other stream technologies and frameworks including Kafka, Camel, etc. to offer more advanced features [20].

2.3. Hazelcast

Hazelcast is an open-source distributed and cloud-native[1] IMDG used for data management and application execution. This memory computing platform provides high speed and scalability. This is achieved by connecting several devices through their main memories and processors so that data structures such as Map, Queue, etc. can be shared, and parallelized workloads can run [21]. Although Hazelcast is written in Java language, it is lightweight, stored as a JAR file, and doesn't depend on any external file. Due to its significance, it is used by many well-known companies, including Platform, Hepsiburada, Infrastructure, Groww, and others. The architecture of IMDG Hazelcast, shown in Fig. 2, describes how devices or computers are connected in a cluster and can view and share the data including maps,

[1] https://github.com/hazelcast/hazelcast

queue, etc. on its center.

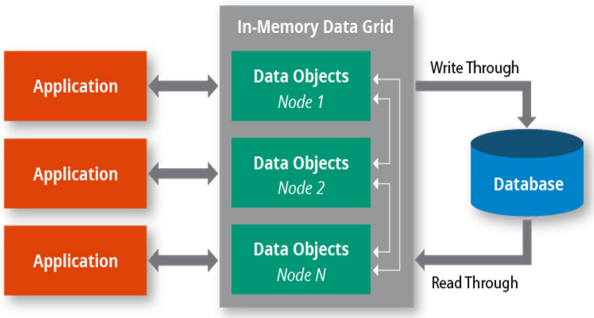

Fig. 2. Detailed Architecture of IMDG Hazelcast

The data and their data types (int, float, double, etc.) are stored in IMDG in the form of an object and these are invoked as variables in application [22]. There are also some drawbacks of IMDG Hazelcast, for example, it has limited memory space, does not deal with large volumes of data, damages the data when the system fails, and hence it is not reliable in all cases [23]. To overcome these weaknesses Hazel Jet [2] is introduced that treat a large amount of data with low latency. Overall, Hazalcast has overcome the limitations of Apache Ignite by having multiple threads through which it deals with any type of CPU core, it gives an infinite number of maps and cache to each cluster, provides distributed environment [24]. In addition, Hazalcast provides the following benefits to Java programmers:

- Store data up to a terabyte in Java without exchanging the interface.
- Established connection of Java client with the Hazelcast cluster.
- Each cluster supports unlimited maps and caches, and all available CPU cores can be used.
- The cluster can be accessed from any JVM language.
- Easy to import the Hazelcast in Java programs just like other libraries, because it is written in Java [25, 26].

3. Related Work

In this section, certain research works providing the comparison of Bid Data streaming platforms are discussed. For instance, a comparison on different big data streaming platforms such as Apache Kafka, Apache Samza, Apache Spark, and Apache Storm based on their features is presented in [27]. The study includes the detailed description, limitations, and advantages of these frameworks, to help companies and businesses in selecting the best platform for data flow. Similarly, the technological frameworks are categorized according to the data processing and data ingestion phases of big data in [28]. The study categorized Kafka, Flume, Nifi as data ingestion frameworks and Spark structure, spark streaming, Strom, Flink, Samza, Apex, Beam as data processing frameworks. The study also compared their features based on performance and latency. Another comparison of big data frameworks such as Hadoop, Spark, Flink, Storm, and Samza is based on different features like programming languages used, data processing mode, etc. [29]. The comparison shows that Flink is more reliable among all these frameworks because it is fault-tolerant and supports data processing as well as batch processing.

Likewise, Hadoop, Spark, Flink, and Storm are compared based on some performance measures i.e., CPU consumption, latency, throughput, Execution time, and fault tolerance [30]. On the other side, a survey paper compares the big data hardware and software platforms based on scalability, real-time processing, fault tolerance, and data size [31]. However, most of these mentioned works are survey papers [28-31], each comparing Big Data frameworks using certain attributes. More importantly, we cannot find and hence conclude that no work has been done on the comparison of Apache Ignite and Hazelcast before. To fill this gap, we have performed a comparison between Apache Ignite and Hazelcast, in this paper.

4. The Comparison

Apache Ignite is widely used around the world and provides the capability of advanced SQL queries and indexing. The well-known alternative of Apache Ignite is Hazelcast. Hazelcast supports multithreading and deals with large amounts of data with low latency. According to Eliana Fernandes [27], the streaming platforms discussed above have common problems such as scalability, and fault tolerance, etc. Whereas Hazelcast and Apache Ignite have excellent scalability and fault tolerance characteristics. This is why these two are selected and compared in this paper so that organizations, businesses, and individuals can easily select the best between them, which suits their requirements. In this work, the following attributes are taken into account for the comparison of Apache Ignite and Hazelcast.

Distributed Network: It refers to the ability of the framework to spread the data, software applications on more than one computer in the network.

Scalability: It is the ability to increase the number of nodes or computers and utilizing the resources to the number of nodes while maintaining the quality of the system.

1. Fastest speed: This allows the application to do complex transactions in milliseconds or sub-millisecond.
2. Backup Recovery: When any disaster happens then the framework can take automatic backup of all data.

[2] https://github.com/hazelcast/hazelcast-jet

3. Non-proprietary software: The software is easily accessible by the user and has the facility for the user to use, and study.
4. Fault tolerance: When a node fails due to any reason then it automatically handles the fault by creating another node and dividing the workload.
5. Cloud storage: The framework allows the applications to store data on the Internet using cloud storage. Cloud storage gives a service to operate and manage data that is free of cost.
6. SQL queries support: It means the framework allows the applications to retrieve, insert data and easily do transactions in the data using SQL supported queries.
7. In-memory storage Transparency: The framework automatically saves the data in the memory without giving any knowledge to the user.
8. Multithreading architecture: This means that more than one thread can execute at a time with any type of CPU core.

Based on the above-selected attributes, the two streaming frameworks are compared and the results are shown in Table 1.

Table 1: Comparison of Hazelcast Vs apache ignite

No.	Elements	Hazelcast	Apache Ignite
1.	Distributed network	✓	✓
2.	Scalability	✓	✓
3.	Fastest speed		✓
4.	Backup recovery	✓	✓
5.	Non-proprietary software	✓	✓
6.	Fault tolerance	✓	✓
7.	Cloud storage	✓	
8.	SQL query support	SQL queries language	ANSI 99 queries
9.	In-memory storage Transparency	✓	✓
10.	Multithreading architecture	✓	

Do not use equations and figures here. The comparison showed that both Hazelcast and Apache Ignite have some common features such as distributed network, scalability, in-memory storage, backup recovery, fault tolerance, and non-proprietary software. But Apache ignite is faster than the Hazelcast in performing transactions of data flow. However, Hazelcast has some additional features, such as cloud storage and multi-threading architecture, which make it more valuable. To conclude, for Java programmers who care about performance, read and write speed, and want to ensure that data is saved correctly, using Hazelcast as a data grid would be very useful and permanent.

5. Conclusion

Different types of databases are used in common to store the data. With the development of today's large-scale distributed applications handling tons of data, these databases are not viable. Consequently, Big Data is introduced to store, process, and analyze data at a fast speed and also to deal with big users and data growth day by day. To process the data in real-time continuously, data streaming technologies are developed. Big Data offers different streaming platforms such as Apache Kafka, Apache Flink, and Apache Storm, etc. Mostly these frameworks have limited scalability, and fault tolerance. Apache Ignite and Hazelcast are also streaming frameworks. Apache Ignite includes the features of both image data grid and in-memory database. Data loading and data streaming capability solve the issue of scalability and fault tolerance. It also combines different streaming platforms such as Kafka, Camel, etc., which makes it too cutting-edge. Hazelcast is a cloud-native, image data grid and has data structures including Map, Queue which can execute parallelly. Hazelcast (IMDG) is improved by the Hazelcast (Jet), which processes the huge data with low latency. In this paper, we have compared Apache Ignite and Hazelcast, to give the knowledge to the Java programmers, companies, and business holders to speed up their application development. Using the comparison, Java programmers are in a good position to decide which framework performs better and suits their application. The comparison is performed using specific attributes selected based on the weakness of the previous frameworks. From the comparison results, it is evident that Hazelcast outperforms Ignite due to the additional features it has such as multi-threading, and cloud storage.

References

[1] Zaki, A. K.: *NoSQL databases: new millennium database for big data, big users, cloud computing and its security challenges*. International Journal of Research in Engineering and Technology (IJRET), vol. 3, no. 15, pp. 403–409 (2014)
[2] Madden, S.: *From databases to big data.* IEEE Internet Computing, vol. 16, no. 3, pp. 4–6 (2012).
[3] Sagiroglu, S., & Sinanc, D.: *Big data: A review*. In: 2013 International Conference on Collaboration Technologies and Systems (CTS). IEEE. (2013, May).
[4] Machine learning and big data. In: Edureka, Big Data and Hadoop. Retrieved October 22, 2021, from Edureka.co website: https://www.edureka.co/blog/machine-learning-and-big-data/(2019, September 24).
[5] Ishwarappa, & Anuradha, J.: *A brief introduction on big data 5Vs characteristics and Hadoop technology*. Procedia Computer Science, vol. 48, pp. 319–324 (2015)

[6] Zhang, Y., Geng, P., Sivaparthipan, C. B., & Muthu, B. A.: *Big data and artificial intelligence based early risk warning system of fire hazard for smart cities*. Sustainable Energy Technologies and Assessments, vol. 45, no. 100986, p. 100986 (2021)

[7] Li, V. O., Lam, J. C., Han, Y., & Chow, K. *A big data and artificial intelligence framework for smart and personalized air pollution monitoring and health management in Hong Kong*. Environmental Science & Policy, vol. 124, pp. 441–450 (2021).

[8] Khan, Z. F., & Alotaibi, S. R.: Applications of artificial intelligence and big data analytics in m-health: A healthcare system perspective. *Journal of Healthcare Engineering*, vol. 2020, p. 8894694 (2020)

[9] Beam, A. L., & Kohane, I. S.: *Big data and machine learning in health care*. JAMA: The Journal of the American Medical Association, vol. 319, no. 13, pp. 1317–1318, (2018)

[10] Asharf, J., Moustafa, N., Khurshid, H., Debie, E., Haider, W., & Wahab, A.: *A review of intrusion detection systems using machine and deep learning in Internet of Things: Challenges, solutions and future directions*. Electronics, vol. 9, no. 7, p. 1177 (2020)

[11] Puentes, F., Pérez-Godoy, M. D., González, P., & Del Jesus, M. J.: *An analysis of technological frameworks for data streams*. Progress in Artificial Intelligence, vol. 9, no. 3, pp. 239–261 (2020)

[12] García-Gil, D., Ramírez-Gallego, S., García, S., & Herrera, F.: *A comparison on scalability for batch big data processing on apache spark and apache flink*. Big Data Analaytics, vol. 2, no. 1 (2017)

[13] Samosir, J., Indrawan-Santiago, M., & Haghighi, P. D.: *An evaluation of data stream processing systems for data driven applications*. Procedia Computer Science, no. 80, pp. 439–449 (2016)

[14] Fernandes, E., Salgado, A., & Bernardino, J. : *Big data streaming platforms to support real-time analytics.* In: Proceedings of the 15th International Conference on Software Technologies. SCITEPRESS - Science and Technology Publications (2020)

[15] In-memory database: Retrieved October 22, 2021, from Hazelcast.com website: https://hazelcast.com/glossary/in-memory-database/ (2019, July 17).

[16] Sojoodi, A. H., Salimi Beni, M., & Khunjush, F.: *Ignite-GPU: a GPU-enabled in-memory computing architecture on clusters*. The Journal of Supercomputing, vol. 77(3), pp. 3165–3192 (2021)

[17] *In-memory data grid with apache ignite.* The Apache Software Foundation. Retrieved October 22, 2021, from Apache.org website: https://ignite.apache.org/use-cases/in-memory-data-grid.html (2021)

[18] Ivanov, N.: *In-memory data grids vs. in-memory databases*. Retrieved October 22, 2021, from Infoworld.com website: https://www.infoworld.com/article/3300747/in-memory-data-grids-vs-in-memory-databases.html (2018, August 29).

[19] Bryant, D.: *John DesJardins on in-memory data grids, stream processing, and app modernization*. Retrieved October 22, 2021, from InfoQ website: https://www.infoq.com/podcasts/imdg-stream-processing/(2020, September 14).

[20] *The Apache Software Foundation*. Data loading and streaming. Retrieved October 22, 2021, from Apache.org website: https://ignite.apache.org/features/streaming.html (2021)

[21] Gencer, C., Topolnik, M., Ďurina, V., Demirci, E., Kahveci, E. B., Lukáš, A. G. O., … Katsifodimos, A.: *Hazelcast Jet: Low-latency stream processing at the 99.99th percentile*. arXiv [cs.DC]. Retrieved from http://arxiv.org/abs/2103.10169 (2021)

[22] *In-memory data grid: A complete overview - hazelcast*. Retrieved October 22, 2021, from Hazelcast.com website: https://hazelcast.com/glossary/in-memory-data-grid/ (2019, November 9)

[23] Guroob, A. H., & Manjaiah, D. H.: *Big Data-based In-Memory Data Grid (IMDG) Technologies: challenges of implementation by analytics tools*. International Journal of Emerging Research in Management &Technology, vol. *6*(5), pp. 829-834 (2017)

[24] Kosandiak, I.: *Spring Boot with Hazelcast*. Retrieved October 22, 2021, from Medium website: https://ihorkosandiak.medium.com/spring-boot-with-hazelcast-b04d13927745 (2018, September 5)

[25] *Why IMDG*. Retrieved October 22, 2021, from Hazelcast.org website: https://hazelcast.org/imdg/why/ (2019, August 5)

[26] Frankel, N. (2021, April 16). Hazelcast, from embedded to client-server. Retrieved October 22, 2021, from Dev.to website: https://dev.to/hazelcast/hazelcast-from-embedded-to-client-server-3j7j

[27] Fernandes, E., Salgado, A., & Bernardino, J.: *Big data streaming platforms to support real-time analytics*. Proceedings of the 15th International Conference on Software Technologies. SCITEPRESS - Science and Technology Publications (2020)

[28] Puentes, F., Pérez-Godoy, M. D., González, P., & Del Jesus, M. J.: *An analysis of technological frameworks for data streams*. Progress in Artificial Intelligence, vol. 9(3), pp. 239–261 (2020)

[29] Gupta, H. K., & Rafat Parveen, D.: *Comparative study of big data frameworks*. 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT). IEEE (2019)

[30] Alkatheri, S., Abbas, S., & Siddiqui, M.: *A comparative study of big data frameworks*. International Journal of Computer Science and Information Security (IJCSIS), vol.17(1), pp. 66-73 (2019)

[31] Singh, D., & Reddy, C. K.: A survey on platforms for big data analytics. Journal of Big Data, vol. 2(1), pp. 8 (2015)

Bartkov Maxim
RooX Solutions Java Team Lead, Khakov, Ukraine, 1233566789b@gmail.com

Katkova Tetiana
University of Customs and Finance, Dnipro, UkraineDepartment of Innovative Technologies,
Department of Cyber security, V.Vernadsky street 2/4, Dnipro, Ukraine, 49000, takit777@gmail.com

Kruglyk Vladyslav S.

Professor of Informatics and Cybernetics Doctor of Pedagogical Sciences, Professor Bogdan Khmelnitsky Melitopol State Pedagogical University, Faculty of Informatics, Mathematics and Economics, Department of Informatics and Cybernetics Hetmanska Street, 20, Melitopol, Zaporizhia Region, 72300 kryglikvlad@gmail.com 0000-0002-5196-7241

Murtaziev Ernest G.
Head of the Department of Mathematics and Physics PHD, Senior Lecturer Bogdan Khmelnitsky Melitopol State Pedagogical University, Faculty of Informatics, Mathematics and Economics, Department of Mathematics and Physics Hetmanska Street, 20, Melitopol, Zaporizhia Region, 72300 ernest_gaf@mail.ru, 0000-0002-2154-5523

Kotova Olha V.
Associate Professor Kherson Candidate of Physical and Mathematical Sciences, Associate Professor State University, Faculty of Computer Science, Physics and Mathematics, Department of Algebra, Geometry and Mathematical Analysis University Street, 27, Kherson, Kherson region, 73000 olga-kotova@ukr.net, 0000-0002-5533-3844