UDC 004.42

V. Mulish, Y. Krylov, V. Nikitin, V. Anikin

USAGE OF A TRANSACTIONAL CLOCK IN A DISTRIBUTED FINANCIAL OPERATIONS SUPPORT SYSTEM TO SPEED UP THE DATA RECONCILIATION PROCESS

Abstract: for many systems, especially financial systems, data consistency and the time it takes for data to become consistent play a critical role in the functioning of the entire system. Therefore, in this article, as one of the ways to speed up the process of data reconciliation, the use of a transactional clock will be considered and verified.

Keywords: information system, distributed system, database, NoSQL, data consistency, transactional clock.

Introduction

Reliability and speed of operation are among the most critical non-functional requirements in the development of any system. Many systems are built or structured around a dataset, whether static or, more commonly, dynamic. Therefore, it is crucial for the data at the core of the system to always be consistent and available. For example, consider a financial operations support system, which we will delve into further. Inconsistencies in its data could lead to significant financial losses for clients, which almost certainly would mark the end of such a system's operation on the financial market.

Moreover, with the development of the Internet, the number of users across various software systems has significantly increased. Consequently, there is a rise in the volume of requests to systems and the amount of data that needs processing and storage. This is why developers of such systems, especially those built around data, must keep up with this pace and strive to make their software products better in various aspects: enhancing user interface convenience and user experience, improving the convenience of programmatic interfaces, enhancing system stability and availability, and most importantly, and perhaps the most challenging, increasing the system's speed.

Therefore, the need to develop new methods of speeding up the process of data reconciliation in distributed information systems becomes even more relevant and pressing. This includes enriching the database of experimental studies that demonstrate the effectiveness of such methods. Hence, this article presents a distributed financial operations support system using a NoSQL database and the transactional clock. Essentially similar system but built using the Replica Set mechanism was taken as a basis for comparison for future experiments. Additionally, the article provides and analyzes the results of experiments investigating the data reconciliation speed in both systems.

 ISSN 1560-8956

Theoretical foundations of transactional clocks usage

The idea of using a transactional clock to improve the efficiency of data reconciliation in various information systems was first proposed in the article by V. Nikitin and Y. Krylov "Usage of transaction clock to speed up the data consistency process in distributed systems" [1]. According to the article, when using a transactional clock, data reconciliation between different nodes of the database is performed according to the following algorithm: firstly, operations that modify data on system nodes are encapsulated in transactions and instantly forwarded to the transactional clock queue. Then, all transactions that modify data are stored in the respective stack and according to the transactional logic are formed into a resulting transaction, which is then transmitted to all database instances. Additionally, to speed up the reconciliation process of critical data, each record receives its priority. In this case, critical data always have the highest priority. An example of how the transactional clock works is illustrated in Fig. 1.

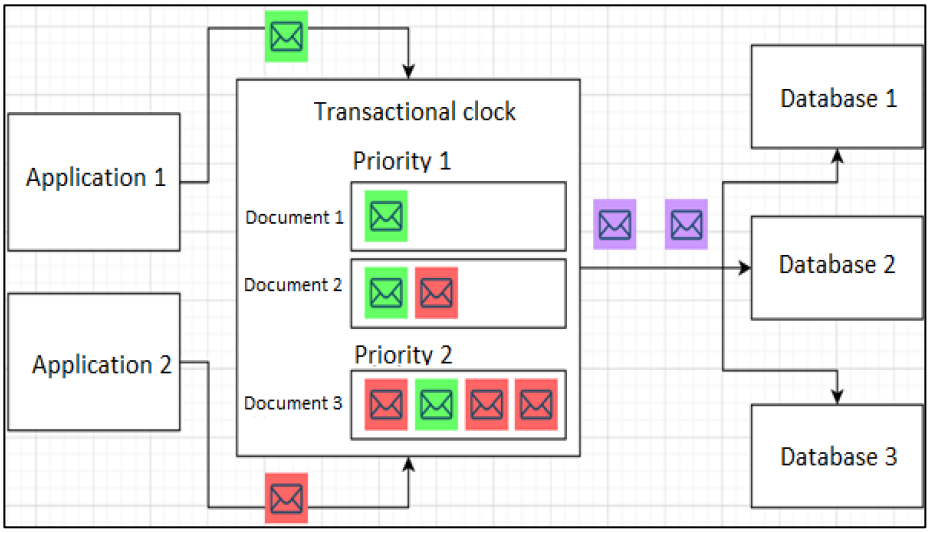

Figure 1. Example of the transactional clock operation

In the transactional clock, the acceleration of data reconciliation process for critically important data occurs through the prioritization of operations and their combination into resulting transactions.

Description of the experimental environment

After analyzing existing non-relational databases that support data replication, MongoDB was chosen. It is one of the most popular open-source non-relational document-oriented databases that offers an excellent balance between speed and scalability. MongoDB provides built-in features for data replication and organizing horizontally scalable clusters with multiple database instances [2].

The next step involved designing and implementing two versions of a distributed financial operations support system: one utilizing the transactional clock as a node responsible for data reconciliation, and the other uses the standard MongoDB Replica Set mechanism. The architecture of both applications is shown in Fig. 2.

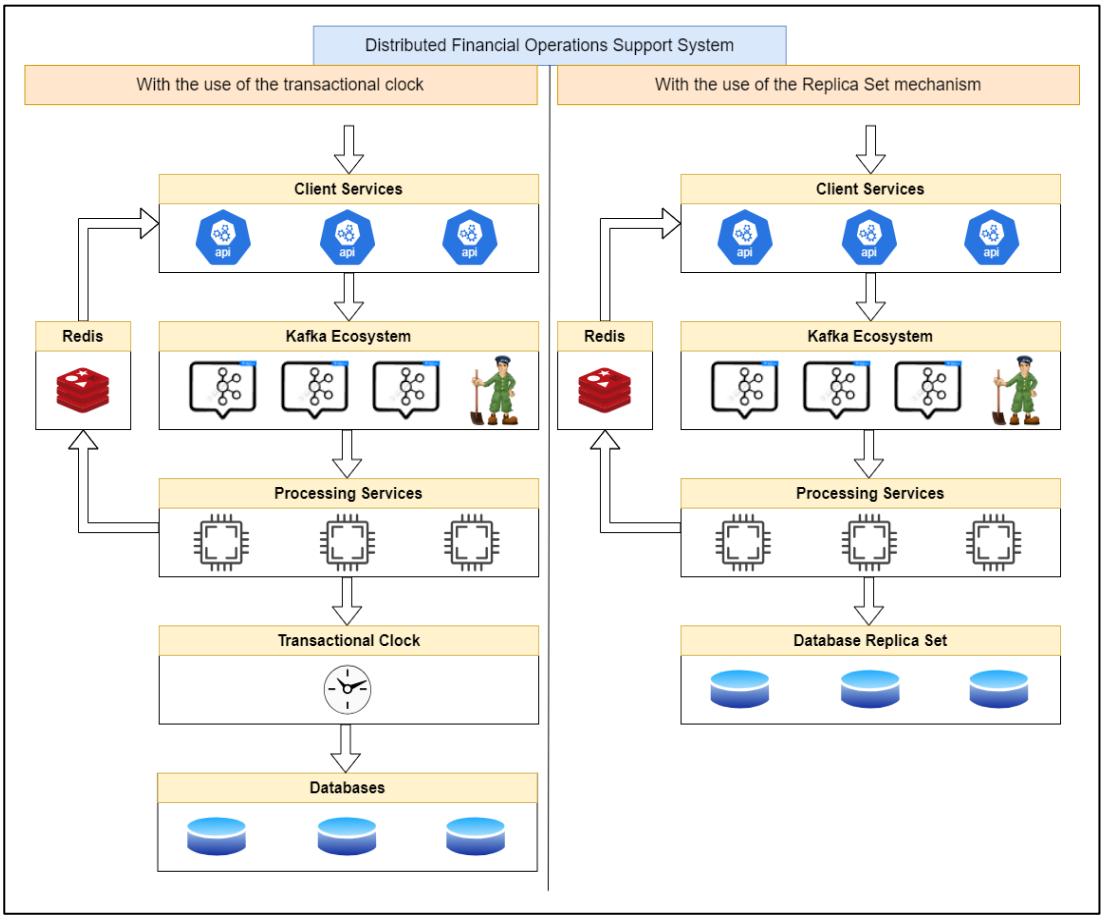

Figure 2. Architectural scheme of both systems, also showing the order of processing a user's request

Both applications have identical functionality: users can register, log into their accounts, update their passport information, create a new bank account, deposit funds, withdraw funds, and make transfers between accounts. Additionally, registration for bank employees and system administrators is also possible.

An experimental study of a transactional clock usage to speed up data reconciliation process

Immediately after the completion of the design and implementation of both systems, a series of experiments was conducted. The results of which not only demonstrate the feasibility of creating a distributed system using a transactional clock, but also prove the effectiveness of the decision to use the transactional clock as a data replication mechanism.

The essence of the first experiment is to check to what extent the use of the

 ISSN 1560-8956

transactional clock gives an advantage in the system's performance due to the acceleration of the data reconciliation process. One of the most challenging scenarios for the developed system was used for the experiment: a single user conducts a large number of operations on the same financial account within an exceptionally short period. These operations include deposits, withdrawals, and transfers, creating a race condition. The experiment was repeated with varying numbers of requests: 1, 10, 20, 50, 70, 100, 500, 1000. Requests are generated one after another, typically within 1-2 seconds. The request processing time is measured from the moment all requests are sent until the final version of the user's financial account state, considering all created operations, is propagated to each database instance.

The experiment was repeated using 3, 5, and 7 instances of the database in order to make its results more representative. Additionally, experiments were conducted with different numbers of launched client and processing services. Various configurations of the test environment validate the results presented below, so they will not be considered separately. Each experiment was repeated 100 times, and the median value of the results is considered the experiment's result. The results of the first experiment using 3 database instances with the use of transactional clock and the Replica Set mechanism are shown in Fig. 3.

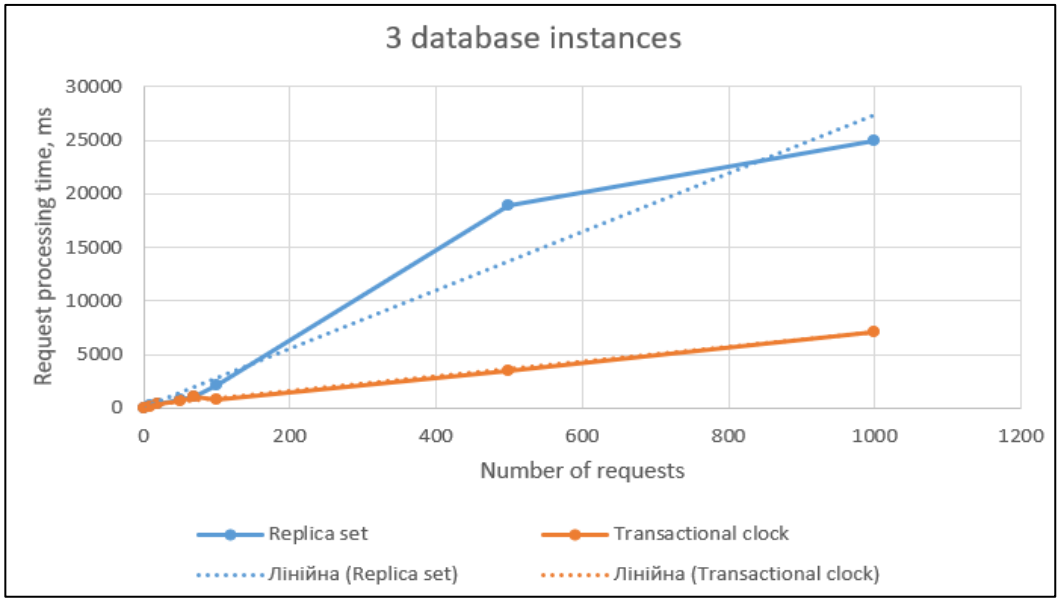

Figure 3. Results of an experiment on processing a large number of simultaneous requests using three database instances

The results of the first experiment, an experiment on processing a large number of concurrent requests, using 7 instances of the database, are shown in Fig. 4.

From Fig. 3 and 4, it is evident that the use of a transactional clock in the distributed system leads to an increase in the speed of the data reconciliation process. When using three database instances, we can observe that the difference in the data reconciliation speed can

reach 3-4 times in favor of the transactional clock. The experiments conducted with 5 and 7 database instances showed similar results, which is why the article reflects only the results of the experiment with seven database instances, showing a difference of approximately two times. This improvement is achieved by creating resulting transactions at the transactional clock node. For both variants of the system (using the transactional clock and the Replica Set mechanism), the trend line of the change in request processing time from their amount follows a linear pattern, indicating the predictability of the performance of both solutions. However, it is noticeable that as the number of requests increases, the performance of the system that uses the transactional clock remains superior to the similar system that uses the Replica Set mechanism.

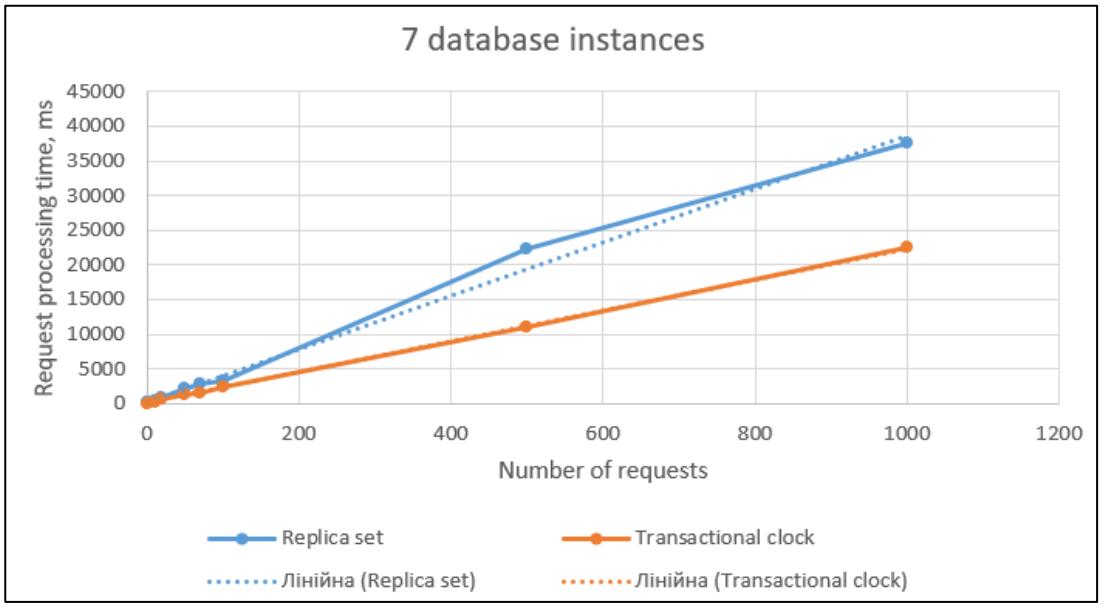

Figure 4. Results of an experiment on processing a large number of concurrent requests using seven database instances

The next experiment aimed to demonstrate the performance improvement in the data reconciliation process using the data prioritization functionality provided by the transactional clock. In the financial operations support system, critical data are data related to user financial account operations: deposits, withdrawals, fund transfers. On the other hand, operations such as registration, creation, deletion of personal accounts and updating user passport data are not as critical. Updating such data in database instances may yield to more important operations in terms of space and time intervals. To demonstrate this, the following experiment scenario was created: different users perform a large number of operations to update their documents, thereby heavily loading the transactional clock and Replica Set with their operations. At the same time, another user attempts to replenish the balance of their account. In this scenario, replenishing the account will lead to the generation of an operation

 ISSN 1560-8956

involving critical data that must be processed and distributed to all database instances with the highest priority. The experiment was repeated with different numbers of document update requests: 100, 500, 1000, 2000, 3000, 4000, 5000. In this experiment, the request to replenish the account is always one and it is generated after 70% of the requests to update documents have been sent to the system. The requests are generated one after another, in most cases, within 3-4 seconds. The account operation request processing time is the time from sending the corresponding request to the moment when the updated version of the user's financial account state is distributed to each database instance, considering the financial operation performed.

This experiment was also repeated using 3, 5, and 7 database instances. Each experiment was repeated 100 times, and the median value of the results is considered as the result of the experiment. The results of the experiment using 3 instances of the database under the guidance of the transactional clock and the Replica Set mechanism are shown in Fig. 5.

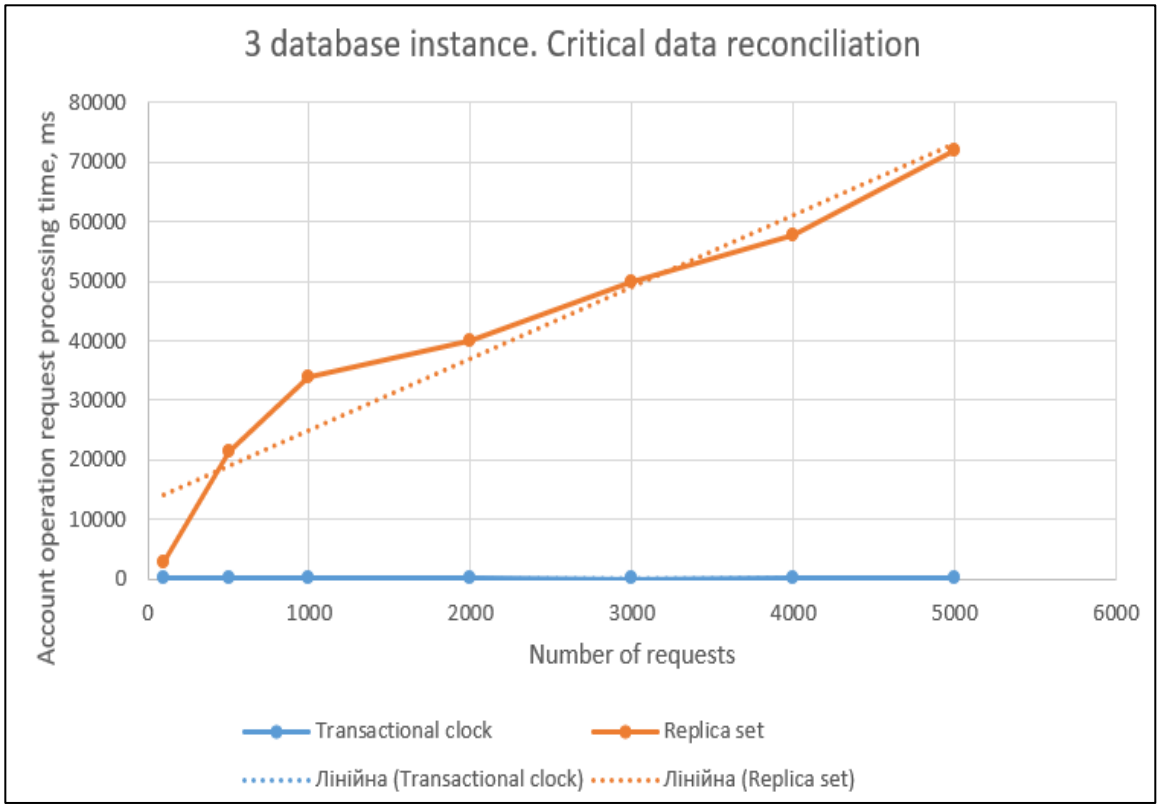

Figure 5. Results of the experiment on critical data reconciliation using three instances of the database

The result of the experiment on critical data reconciliation using 5 instances of the database is shown in Fig. 6. The experiment with seven instances of the database showed results consistent with the following results, so it will not be displayed in the article.

As seen from Fig. 5 and 6, the difference is substantial. The use of the transactional

clock and the concept of critical data embedded in it, in the distributed system shows a striking increase in the speed of data reconciliation process and overall system performance. When using different numbers of database instances, the Replica Set mechanism showed an unsatisfactory data reconciliation times for user financial account-related data. This can be explained by the Replica Set mechanism being overloaded with a large number of concurrent requests, processing them in the order they arrive. Therefore, operations whose results the user wants to see as quickly as possible do not have priority over others. Moreover, the processing time of the account operation increases continuously as the number of requests increases, whereas the transactional clock spreads data across all database instances in a time close to constant, depending on the number of database instances used. The processing time for critical data using the transactional clock is presented in Table 1, with the unit of measurement being milliseconds.

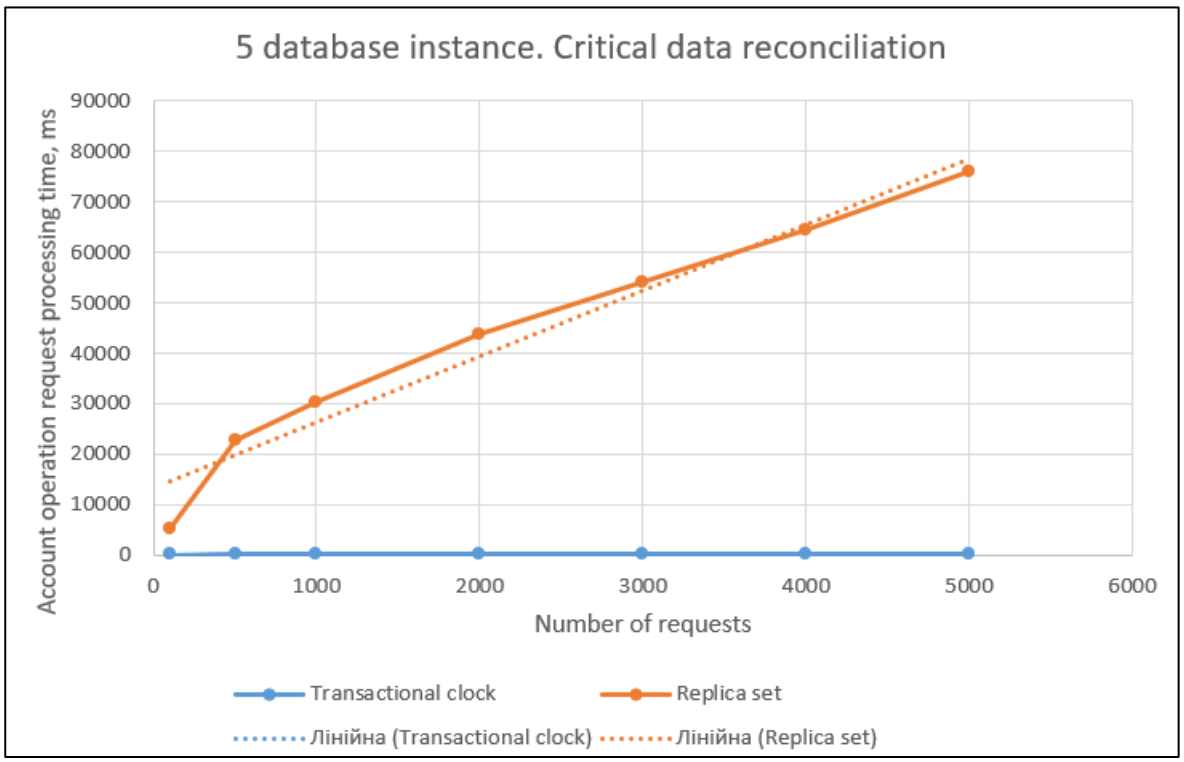

Figure 6. Results of the experiment on critical data reconciliation using five instances of the database

From the table, it is evident that for every two additional instances of the database under the control of the transactional clock, the time of reconciliation of critical data increases by approximately 50 milliseconds. However, with an increase in the number of requests related to data of lower priority than the critical data, the reconciliation time remains close to a constant value. Having those results, it is possible to guarantee that client operations on critical data such as financial account in the financial operations support system will be executed within a fixed time.

 ISSN 1560-8956

This is an extremely valuable assurance for the user, as they can be confident in the speed and accuracy of processing of their financial operations.

Table 1.

Critical data reconciliation time in a system that uses the transactional clock

Database instances	**Number of non-critical requests**							**Average time, ms**
	100	500	1000	2000	3000	4000	5000	
3	253	317.7	250	317.5	240.3	340.3	300.1	288.5
5	295.7	345.6	312.2	330.6	367.1	345.5	327.4	332
7	387.2	401.3	402.7	390.4	391.3	387.9	398.8	394.2

Conclusion

After analyzing the performance of both implementations of the distributed financial operations support system, it can be concluded that the use of the transactional clock in such systems increases its performance by speeding up the data reconciliation process by approximately two times. This statement is correct even in cases of significant load on the system.

Additionally, it is worth noting the excellent efficiency of using the critical data concept of the transactional clock, use of which leads to a sharp and multiple increase in the speed of the data reconciliation process in the developed distributed financial operations support system. Assigning critical priority to the data results in a constant and an order of magnitude faster data reconciliation time between database instances compared to the same system using the Replica Set mechanism.

REFERENCES

1. *Nikitin V.* Usage of transaction clock to speed up the data consistency process in distributed systems / Krylov Y., Nikitin V. / Scientific Bulletin of Uzhhorod University. Series "Mathematics and Informatics" volume 42, №1, 2023. – pp. 188-192.

2. *K. Chodorow* MongoDB: The Definitive Guide (1st edition) / K. Chodorow, M. Dirolf / O'Reilly Media, 2001. – pp. 216-217.