Middle-East Journal of Scientific Research 20 (12): 2184-2193, 2014
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2014.20.12.21106

Realtime Development and Testing of Distributed Data Processing System for Industrial Company

[1]Andrey Ostroukh and [2]Artur Pomazanov

[1]Department of Technology Sciences, "Automated Control Systems",
Moscow Automobile and Road Construction State Technical University,
125319, Moscow, Leningradsky Prospect, 64
[2]Master of Engineering and Technology Specialist, Department "Automated Control Systems",
Moscow Automobile and Road Construction State Technical University,
125319, Moscow, Leningradsky Prospect, 64

Abstract: This paper offers a general structure of the distributed system for the remote sites as well as the procedure and principle of their operation; in addition, a comparison is made, identifying the advantages and disadvantages. The overall test results of the developed system at different loads and different configurations are shown; the system efficiency dependency on the configuration and load is revealed. It concludes with a brief description of the advantages of a distributed system as compared to the existing system.

Key words: Databases · Transaction systems · Data streams · Distributed systems · Remote sites · Testing systems · Parallel processing · OLTP systems

INTRODUCTION

OLTP (Online Transaction Processing) system [1, 2] is a real-time transaction processing system. It is a way of organizing the database in which the system operates with small-sized transactions in a large flow, requiring the client the minimum system response time. However, modern OLTP systems do not only work with small amounts of data. As part of the databases may well be millions and millions of processes. At the same time, the system requires the same minimum response time, as it does when working with small volumes of data [5-8].

Furthermore, many OLTP systems operate with multiple datastreams often located at a considerable distance from each other [9-23]. When possible, these sources are grouped into nodes called "remote sites".

The current system had sequentially worked with remote sites. But over time, the amount of data has increased and the system could not meet the imposed requirements. At peak loads, the current system shows low speed and slow context switching between the remote sites. Due to the fact the remote sites areconsistently searched, the sampling time of the first and the last site vary greatly. This leads to a mismatch of data at the remote sites and the hub server [5].

In view of the above, a distributed system of parallel processing with multiple remote sites was developed. Instead of the sequential polling ofremote sites used by the current system, a distributed system performs a parallel poll of remote sites and combinesthe results [4] by means of a coordinating server. Besides solving the problem of low speed operation at high loads, the new distributed system solves the problem of context switching if one or more remote sites are unavailable.

System Architecture and the Operation Features: The new system was designed to replace the existing system; all the shortcomings of the existing system were taken into account. In addition, the established system implements the principle of parallel processing of sites, which allows for the independence from the remote sites availability. The new system, in contrast to the existing one, has a system of asynchronous request processing, thus expanding the capacity of the system.

Corresponding Author: Andrey Ostroukh, Department of Technology Sciences, "Automated Control Systems", Moscow Automobile and Road Construction State Technical University, 125319, Moscow, Leningradsky Prospect, 64.

The main difference of the new system in terms of architecture is transfer of the operating logic from the remote sites into the coordinator's database [1]. This allows refusing from the support of any third-party programs for query processing that use the database as a repository for information necessary for work. Coordinator's database is an analogue of the query-processing core of the existing system; however, it has a more complex and efficient structure and operating principles.

General principle of a single query processing:

- Sending a request to the coordinating server [1] and launching the main procedure;
- Obtaining a unique process index;
- Searching the necessary remote sites;
- Accounting the number of running threads; temporary tables creation;
- Sending allthe necessaryparameters;
- Sampling messages by means of the queue procedures and performing the necessary remote site procedures; accounting the data obtained into the temporary tables;
- Entry on the completion of each procedure queue in the table of flows;
- Checking the basic procedure thread table [6] for compliance with the number of registered and completed threads;
- Combining and withdrawing all the data obtained from the sites; temporary tables deletion;
- Completion of the basic procedure.

General scheme of the distributed system is shown below (Fig. 1):

The central component of the coordinator's database is an asynchronous message queue. It stores the messages containing the necessary information in the order received: remote site used for the sampling, selection criteria, a name for the temporary table, the process index. The queue is processed by a certain number of queue procedures, which sample and analyze the stored messages and perform all the necessary work with the specified remote server. The increase in concurrent queue procedures leads to the increase in system availability, while the response time of the system is reduced; however, the amount of resources consumed increases. Independent execution of a sufficient number of queue procedures allows for a complete independence of processes from each other [4].

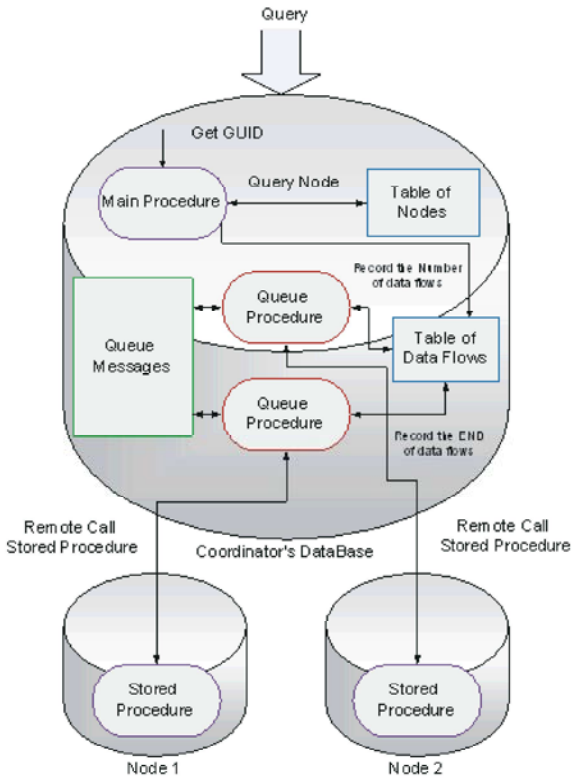

Fig. 1: General structure of the distributed system

When you access the coordinating server, the main procedure is launched, creating a group of messages that are placed in the message queue for processing by means of a table of nodes. In addition, the basic procedure indicates the process index, calculates the required number of threads and monitors the table of data flows. Once all the necessary queue procedures have reported their completion, the basic procedure combines the results and reports them to the user or an external system. External monitoring of the queue procedures is necessary for the correct integration of their performance results.

Advantages:

- Asynchronous handling of requests;
- Independence from the remote sites availability;
- Parallel operation with remote sites within a single query;
- System scalability[6].

Disadvantages:

- The complexity of implementing:
- Increased consumption of the system resources.

The new system takes into account the errors of the current system, so that the new system has excellent scalability. In fact, the scalability of a distributed system is only limited by the available hardware used for the system operation. In the future, this will be confirmed by tests.

System Load Testing: This test is conducted in order to check whether the system meets the demands placed upon it and how well it does [6]. In addition, we would like to reveal the dependence of the load degree to the system and its capacity. We test the system at different bandwidth settings to assess its scaling ability. Such terms as "easy" and "hard" queries are used in testing.

"Easy" queryis a querythat requires fetching data from each node, with the single sampling from one node not exceeding 1000 data rows. Percentage of suchqueries in the existing system makes >80%.

"Hard" query is a query, which requires selecting a sample from each node, whereas a single sampling node may exceed the value of 500,000 data rows.

Requirements for the tested system:

- Runtime not exceeding the expected;
- Parallel asynchronous operation of multiple processes of the incoming queries.

Applicable methods for testing the established system:

- Implementation and collection of statistics on the performance of a single query with a different size of data. Request to sample from 100 to 800,000 rows of data from each required remote site;
- Implementation of parallel execution of the multiple processes with a subsequent analysis of statistics. Simultaneous execution of 3 to 20 queries that return 400,000 rows of data from each required remote site.Subsequent change in the number of concurrent queue procedures from 10 to 20 and a retest for assessing the scaling ability of the system;
- Simultaneous launch of "hard" and "easy" queries and collection of the run-time statistics. The simultaneous launch of "hard" queries in the quantity equal to the number of running queue procedures and 10 "easy queries", to identify the dependence of the "easy" queries performance on the number of "hard" queries in the system. Thesimultaneouslaunchof "hard" queries in an amount less than the number of running queue procedures per 5 and 10 "easy" queries;
- System resource consumption monitoring. Identifying the CPU resources, memory, I / O disk and network resource consumption at the returned data volume of 100, 800,000 from each required remote server at the simultaneous run of queue procedures in the amount of 10 and 20, with the number of concurrent queries equaling 5, 10 and 20;

- Testing a large number of "easy queries":
- Queries that return an empty set – 17%;
- Queries that return 1 row– 15%;
- Queries that return 5 rows – 15%;
- Queries that return 10 rows – 10%;
- Queries that return 10 rows – 10%;
- Queries that return 1000 rows – 33%.

Results of the test are shown in Fig. 2.

As seen in Fig. 2, due to parallel processing of several remote sites, the sampling from sites with up to 200,000 rows of data does not affect the response of the system. However, ifthisvalueisexceeded, the reisasharpspikein the responsetime.This is due to the fact that such data volume cannot be transmitted over the network in a smaller amount of time. In addition, the amount of data storage on the coordinating server increases, just as does the processing and integration of such volumes of data. All this gives rise to the system response time.

It should be noted that queries whose return data volume exceeds 400,000, are automated and serve for the internal purposes of other systems using the established system as a data stream. Long-term performance of these automated processes does not affect the response of the user queries through their independent asynchronous processing, provided there is a due reserve of bandwidth.

As seen in Fig. 3, as the number of queue procedures increases and so does the system resources consumption, we get a graph smoother by the number of requests. This is due to the fact that at the lack of queue procedures, queries are queued for processing. This prevents you from losing data, but has a negative impact on the system capacity.

Fig. 4 below shows that at the shortage of queue procedures, i.e. at the lack of system resources, the execution time of "easy" queries increases dramatically when the number of "hard" queries exceeds or is equal to the queue procedures. This is due to the above-described process of forming a queue [1].

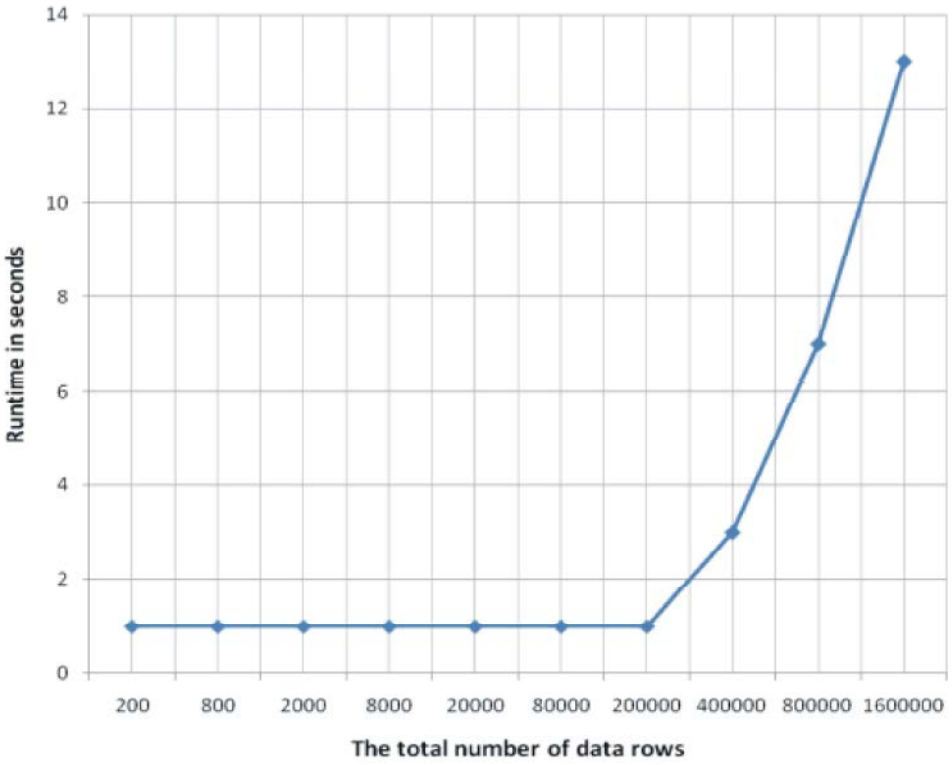

Fig. 2: Runtime dependence on the amount of data returned

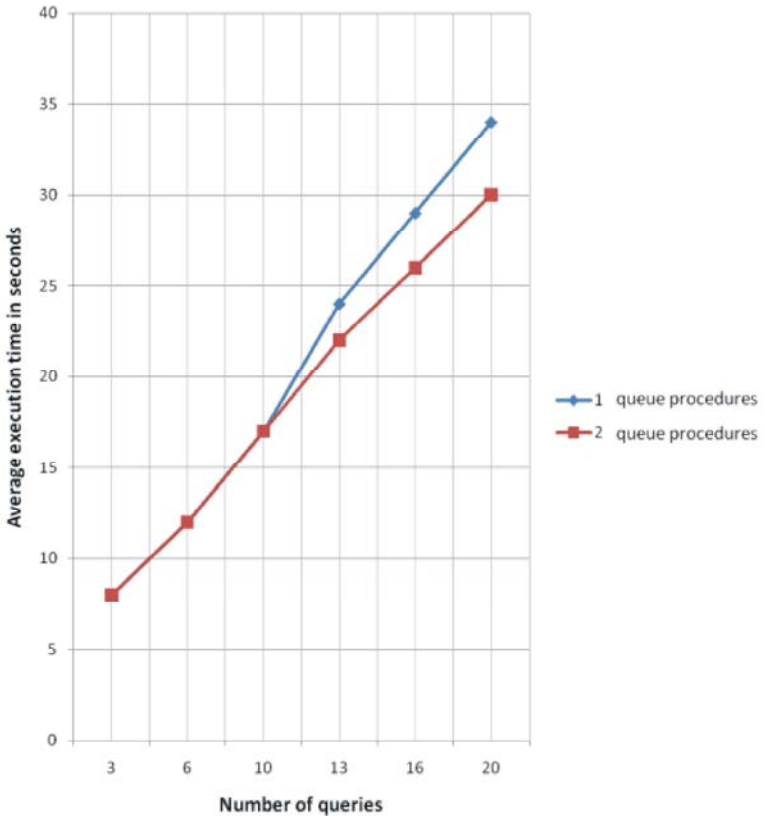

Fig. 3: Runtime dependence on the number of simultaneous queries with a different number of queue procedures

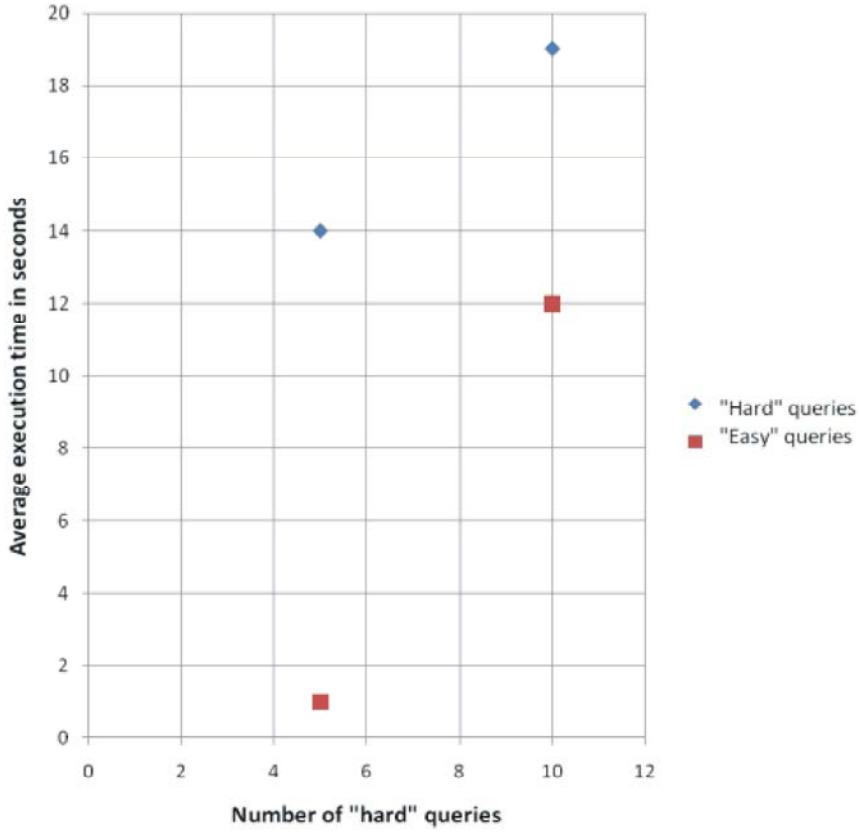

Fig. 4: Dependence of the "easy" queries runtime on the number of "hard" queries (the number of queue procedures is 10)

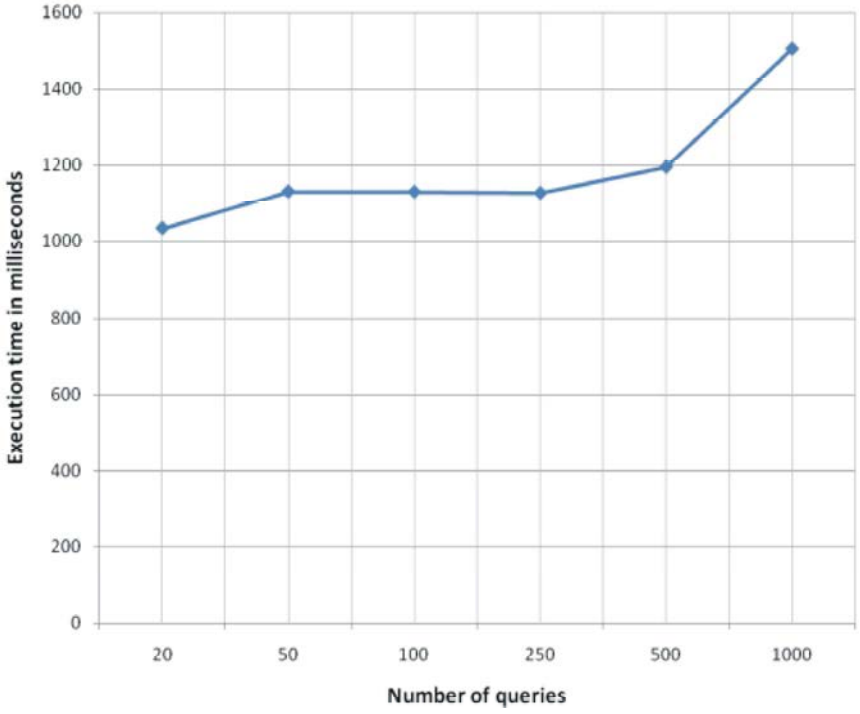

Fig. 5: "Easy" queriestest results

Table 1: "Easy" queriestest matrix

Number	Empty	1	5	10	100	1000
20	3.4	3	3	2	2	6.6
50	8.5	7.5	7.5	5	5	16.5
100	17	15	15	10	10	33
250	42.5	37.5	37.5	25	25	82.5
500	85	75	75	50	50	165
1000	170	150	150	100	100	330

Table 1 shows the distribution of particular "easy" queries in respect to the amount of all easy queries in the system. For example, if the total number of simultaneous "easy" queries is 500, there will be 75 queries returning 5 rows of data from a single node.

As can be seen from Fig. 5, the established system copes well with the processing of a large number of simultaneous "easy" queries.

Stability Testing: This test was conducted to ascertain the system stability under the different external conditions, such as when one or more required nodes [6] are unavailable. In addition, testing of the remote sites unavailability audit system was conducted, revealing the reasons for their unavailability.

Requirements for the tested system:

- Operation at the unavailability of one or more required nodes;
- Operation in case of errors or blocking on one or more nodes;
- Error notification.

Applicable methods for testing the established system:

- Running a query when one or more nodes are out of reach. Running a query on the two nodes and analyzing its work when one or both required remote sites are unavailable;
- Query execution and its intentional blocking of one or more nodes. Running a query on the two nodes and the analysis of work in the temporal or constant blocking of one or two critical remote sites;
- Process errors auditing. Verification of errors reporting in the system errors audit table and their compliance with the real errors observed in the previous test.

The test results:

- At one of the nodes being unavailable, the system has produced a parallel sampling from the remaining available nodes; upon expiration of connection to the unavailable node, the unhindered delivery of the results has been performed.
- At the unavailability of both nodes, upon expiration of connection to the both servers,the system has issued a blank sampling, as it had failed to get any data.
- Atoneblocked node, the system has been sampling other nodes; it then waited until the blocking expired; when the blocking ceased, sampling from the node was conducted and complete data was issuedat the coordinator's database.
- Whenbothnodeswereblocked, the system waited until the blocking of one of the nodes ceased and performed sampling from that node. When both nodes were unblocked, complete data was issuedin the coordinator's database.

If any of the above problems arises, the system will correctly record the information into the audit table specifying the date and cause of the contingency.

As seen from the above information, the system does not lose efficiency if one or more nodes are unavailable. This is because the queue procedures are not connected with each other and the node inaccessibility affects only the queue procedures that work directly with the inaccessible remote node.

Furthermore, if at least one remote node is available for sampling, i.e. for example, there are 12 nodes and only one is available, then the sampling can be freely performed.

Stress Testing: This test was carried out in order to verify that the partial or full operation of a distributed system in excess of the maximum allowable load [7, 8] is preserved. In addition, the ccorrelation between the excess load on the system and the response time of the created system is determined.

Requirements for the tested system:

- Preservation of complete or partial efficiency at the excess of the expected load.

Applicable methods for testing the established system:

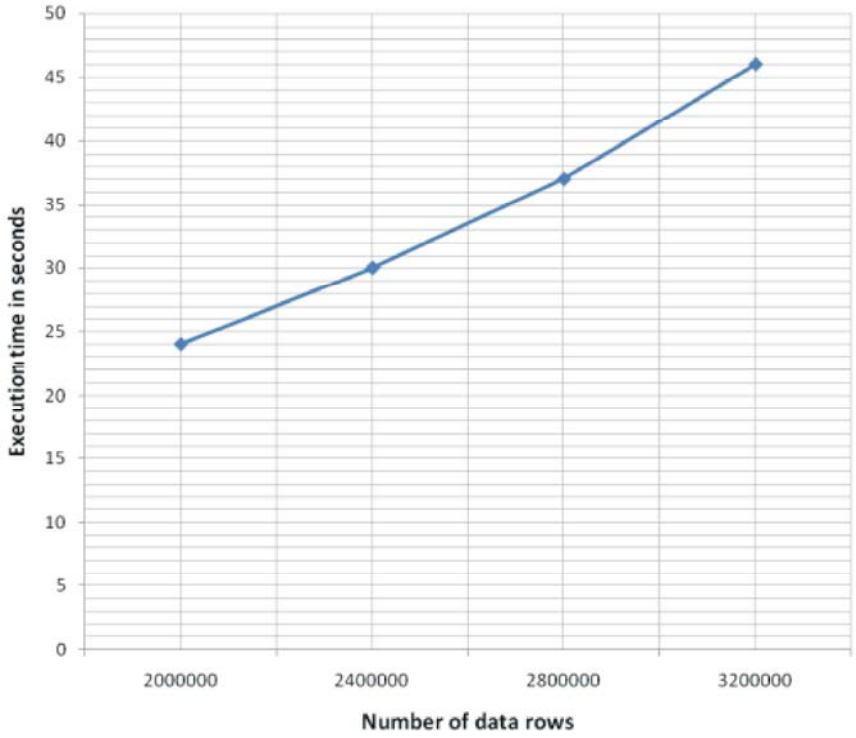

Fig. 6: Dependence of the runtime on the amount of data returned

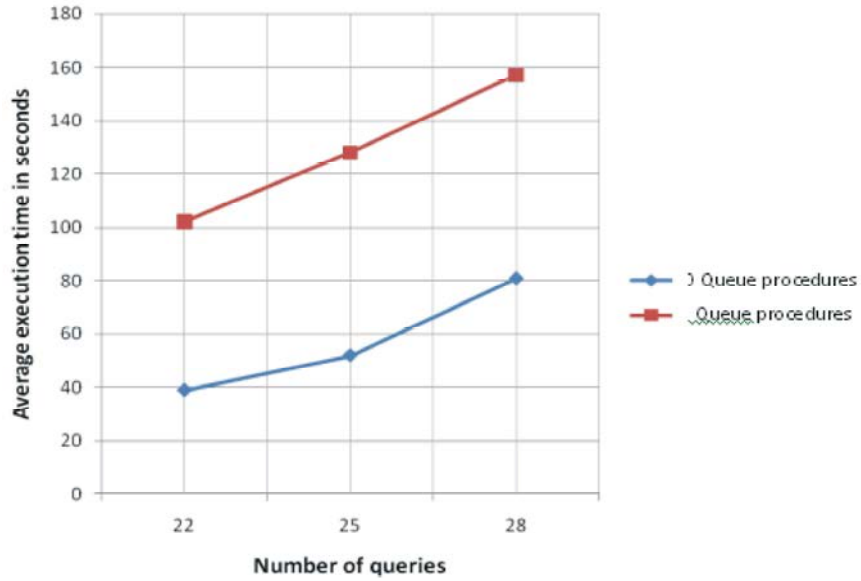

Fig. 7: Dependence of the runtime and the number of simultaneous queries with different amount of queue procedures

- Implementation and collection of statistics on execution of a single query with a data size greater than expected. Query for sampling from 800,000 to 1,600,000 rows of data from each required remote site;
- Implementation of the parallel running of processes, the amount of which exceeds the expected. Simultaneous execution of 20 to 30 queries that return 400,000 rows of data from each required remote site. The subsequent change in the number of simultaneous queue procedures from 10 to 5 and a retest.

The test results are shown in Fig. 6 and 7 below.

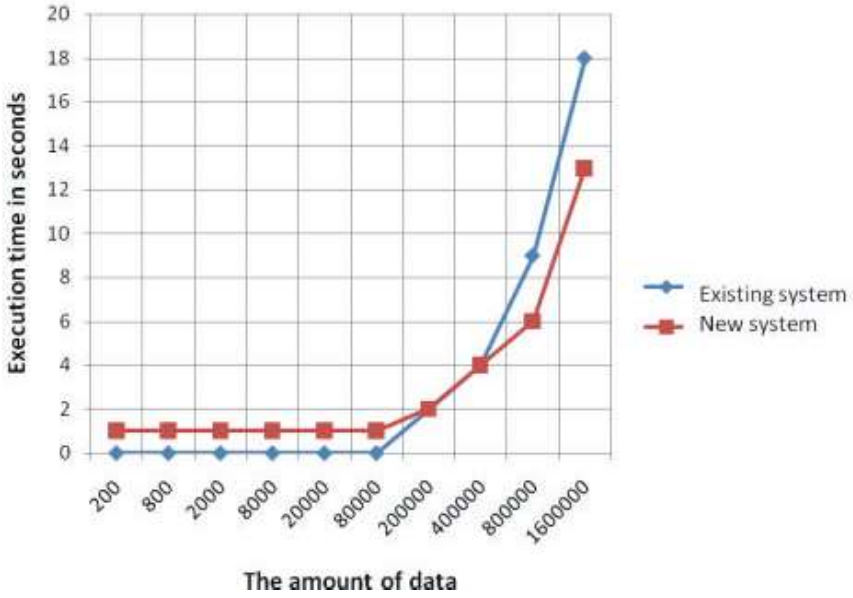

Fig. 8: Runtime dependence on the amount of data returned on different systems

As seen in Fig. 6, at the exceeded load on the system, during the attempts to fetch up to 3.2 million rows of data in a single query to a node, the system remains fully operational, but the response time increases dramatically. This is due to the problems of transmitting a large amount of information from the remote sites via the network. Furthermore, a sufficiently large amount of time is spent on the arrangement of the resulting data [3].

As can be seen from Fig. 7, the system does not lose efficiency at the exceeded maximum load, at the critical shortage of the system capacity. In that case, queries form a queue; it leads to the increase in response time, but does not affect the overall performance of the system. It should be noted that in this case the system response time increases mainly for "hard" queries.

Comparative Testing of the Two Systems: This test was conducted to determine the amount of increased productivity when implementing a distributed system, compared to the existing system. The test results will allow determining whether the created system is far superior to the existing one and whether its implementation is worththe cost and time invested in its development.

Requirements for testing:

- Testing to be conducted under the same conditions and on the same amount of data.

Applicable methods of system testing:

- Comparison of the existing and the new system using various data by sequentially executing queries on the two systems.

Test results are shown in Fig. 8.

Byanalyzing Fig. 8, we can say that then ewsystemisfarsuperiorthan the existing system in all respects. The slight superiority of the existing system's response time on the interval to 200,000 rows of data is due to the small number of remote sites; this test used two nodes, including the cost of processing the parallel processing on the remote sites. With an increase in the required remote sites, this superiority is completely leveled.

CONCLUSION

Regarding the test results, it is safe to say that the new system results exceed the results of the existing system by 30-35% under all types of load on the system.

Implementation of this system into the production will allow to eliminate the consequences of the existing system shortcomings, as well as to increase the overall efficiency of not only the system that provides access to data, but also of all systems that depend on it. Sufficient system scalability is the key to its long and smooth operation, even with an increase in the average volume of data increase per month.

Thus, it has been objectively proven that the solution to replace the existing system proposed in this paper is both functional and useful for further work on the introduction of the advanced information technology in the OLTP system.

REFERENCES

1. Jamie Macclenny, Zhaohui Tang and Bogdan Krivat, 2009. Data Mining with Microsoft SQL Server 2008. M: BHV-Petersburg, pp: 700.
2. Dušan Petković, 2008. Microsoft SQL Server. A Beginner's Guide. M: BHV-Petersburg, pp: 752.
3. Ken Henderson, 2006. Professional Guide to SQL Server. Architecture and Internals. M: Williams, pp: 1056.
4. Pomazanov, A.V., 2012. Database optimization method/ A.V. Pomazanov, A.I. Belousova, A.M. Vasilieva, A.V. Ostroukh // In the world of scientific discoveries. M: Research and Innovation Center, 12: 49-53.
5. Ostroukh, A.V., 2011. Information technology in the research and production activities / [Ed. A.V. Ostroukh] M: Techpolygraphcentre LLC, pp: 240. ISBN 978-5-94385-056-1.
6. C.J. Date. AnIntroductiontoDatabaseSystems, Eighth Edition- «Addison–Wesley»,2006, pp: 1328.
7. Paul Bertucci, 2004. Microsoft SQL Server High Availability. «Sams», pp: 456.
8. Sanjeena Subedi, Antonio Punzo, Salvatore Ingrassia, Paul D. McNicholas, 2013. Clustering and classification via cluster-weighted factor analyzers. Advances in Data Analysis and Classification March 2013,. DOI: 10.1007/s11634-013-0124-8, 7(1): 5-40.
9. Sijia Liu, Anastasios Matzavinos and Sunder Sethuraman, 2013. Random walk distances in data clustering and applications. Advances in Data Analysis and Classification March 2013, DOI: 10.1007/s11634-013-0125-7, 7(1): 83-108.
10. Ricardo Fraiman, Badih Ghattas and Marcela Svarc, 2013. Interpretable clustering using unsupervised binary trees. Advances in Data Analysis and Classification June 2013, DOI: 10.1007/s11634-013-0129-3, 7(2): 125-145.
11. Hamid Parvin, Behrouz Minaei-Bidgoli, 2013. A clustering ensemble framework based on elite selection of weighted clusters. Advances in Data Analysis and Classification June 2013, DOI: 10.1007/s11634-013-0130-x, 7(2): 181-208.
12. Anastasios Bellas, Charles Bouveyron, Marie Cottrell and Jérôme Lacaille, 2013. Model-based clustering of high-dimensional data streams with online mixture of probabilistic PCA. Advances in Data Analysis and Classification September 2013, DOI: 10.1007/s11634-013-0133-7, 7(3): 281-300.
13. Angela Montanari, Daniela G. Calò, 2013. Model-based clustering of probability density functions. Advances in Data Analysis and Classification September 2013, DOI: 10.1007/s11634-013-0140-8, 7(3): 301-319.
14. Nema, Dean and Rebecca Nugent, 2013. Clustering student skill set profiles in a unit hypercube using mixtures of multivariate betas. Advances in Data Analysis and Classification September 2013, DOI: 10.1007/s11634-013-0149-z, 7(3): 339-357.
15. Michio Yamamoto, 2012. Clustering of functional data in a low-dimensional subspace. Advances in Data Analysis and Classification October 2012, DOI: 10.1007/s11634-012-0113-3, 6(3): 219-247.
16. Zoltán, Prekopcsák and Daniel Lemire, 2012. Time series classification by class-specific Mahalanobis distance measures. Advances in Data Analysis and Classification. October 2012, DOI: 10.1007/s11634-012-0110-6, 6(3): 185-200.
17. Ruwet, C., L.A. García-Escudero, A. Gordaliza and A. Mayo-Iscar, 2012. The influence function of the TCLUST robust clustering procedure. Advances in Data Analysis and Classification. July 2012, DOI: 10.1007/s11634-012-0107-1, 6(2): 107-130.
18. Carlos Ordonez, Naveen Mohanam and Carlos Garcia-Alvarado, 2013. PCA for large data sets with parallel data summarization. Distributed and Parallel Databases September 2013, Volume 31, Issue 3. DOI: 10.1007/s10619-013-7134-6. pð.37-41.
19. Javier García-García, Carlos Ordonez and Predrag T. Tosic, 2013. Efficiently repairing and measuring replica consistency in distributed databases. Distributed and Parallel Databases. September 2013, DOI: 10.1007/s10619-012-7116-0, 31(3): 377-411.
20. Aditya Telang, Sharma Chakravarthy, Chengkai Li. Personalized ranking in web databases: establishing and utilizing an appropriate workload. Distributed and Parallel Databases. March 2013, DOI: 10.1007/s10619-012-7106-2, 31(1): 47-70.
21. Shiyuan Wang, Divyakant Agrawal and Amr El Abbadi, 2013. Towards practical private processing of database queries over public data. Distributed and Parallel Databases January 2013. DOI: 10.1007/s10619-012-7118-y, 31(1): 87-110.

22. Malcolm Atkinson, Chee Sun Liew, Michelle Galea, Paul Martin, Amrey Krause, Adrian Mouat, Oscar Corcho and David Snelling, 2012. Data-intensive architecture for scientific knowledge discovery. Distributed and Parallel Databases October 2012, DOI: 10.1007/s10619-012-7105-3. 30(5-6): 307-324.
23. Naser Ayat, Reza Akbarinia, Hamideh Afsarmanesh, Patrick Valduriez, 2013. Entity resolution for distributed probabilistic data. Distributed and Parallel Databases December 2013, DOI: 10.1007/s10619-013-7129-3, 31(4): 509-542.