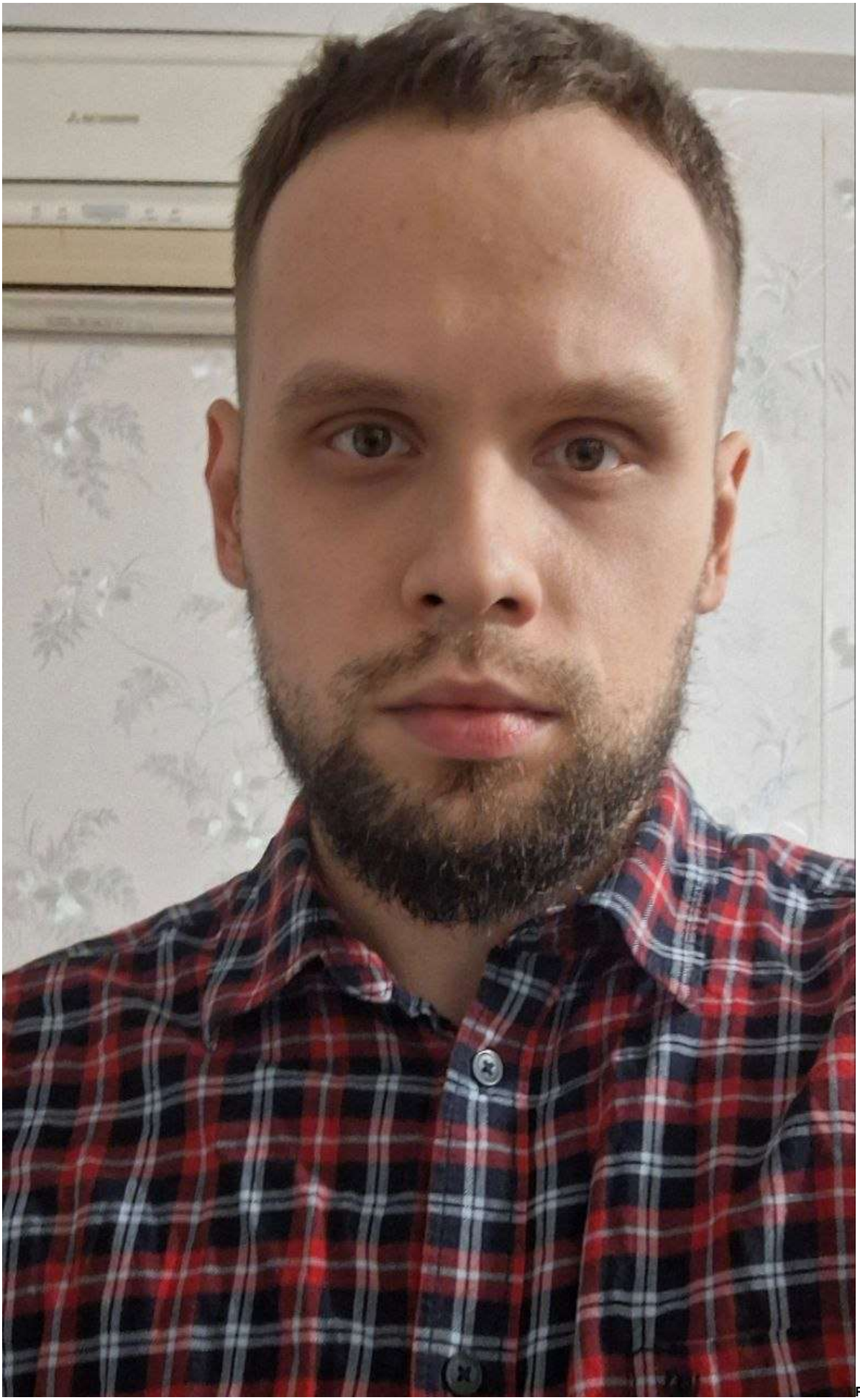

Received 02

December, accepted 09 May 2026, date of publication 30 June 2026
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.25.2.4659

Evaluating Data Replication Models for Improving the Efficiency of Distributed Microservice Systems

OLEKSANDR KUPRIIANOV, VALERII SYTNIKOV, ANDRII MAKSYMENKO
Department of Computer Systems, National University "Odesa Polytechnic", Shevchenko ave., 1, Odessa, Ukraine, 65044

Corresponding author: Oleksandr Kupriianov (e-mail: kupriyanov.a062@stud.op.edu.ua).

ABSTRACT The object of the study is the process of data replication in distributed microservices systems. The subject of the study is the data replication models used in such systems. The aim of the study was to improve the efficiency of distributed microservices systems by optimizing the balance between performance, data consistency, and operational costs. Modern replication models – synchronous, asynchronous, quorum-based, CRDT, and blockchain – were analyzed in the context of their adaptation to microservices architectures. It was found that asynchronous models provide high performance under variable load but reduce data consistency; synchronous models ensure stable consistency at the cost of higher overhead; quorum-based models optimize the balance between availability and consistency, achieving 15–20% operational cost savings. A classification of replication models by performance, consistency, and fault tolerance was developed to facilitate the rapid selection of the optimal solution based on system priorities. Experimental studies were conducted using Docker and Kubernetes with Kafka, Redis, and CockroachDB under loads of up to 5000 requests per second. Comparative evaluation showed that CRDT models provide maximum performance, while quorum-based models offer reliability and cost efficiency. An approach for calculating operational costs was proposed, taking into account request intensity, network latency, and replica storage costs. The results enable informed selection of replication models and enhance the efficiency of microservices systems, particularly during migration from monolithic to containerized architectures and scaling of high-load telecommunications systems.

KEYWORDS data replication, microservice architecture, distributed systems, performance, availability, data consistency, Docker, Kubernetes.

I. INTRODUCTION

In the current context of rapid development of information technologies, particularly the increasing volumes of processed data and the transition to cloud and distributed infrastructures, the issue of effective data management in such environments becomes especially relevant. One of the most promising approaches in building distributed information systems is microservice architecture, which ensures scalability, component independence, flexible deployment, and enhanced reliability.At the same time, microservice architecture introduces new challenges related to the need for coordinated access to shared data, ensuring data integrity, high availability, maintaining consistency, and efficient updates in a distributed environment.

II. LITERATURE REVIEW AND PROBLEM STATEMENT

In study [1], the results of studies on traditional data replication models in cloud environments are presented. It was shown that asynchronous replication provides higher performance under variable load conditions; however, this comes with the risk of losing consistency. Nevertheless, issues related to adapting such models to microservice architectures, where components operate independently, remain unresolved. This may be due to the limited flexibility of classical approaches in distributed, highly dynamic environments. One possible way to address this problem is to incorporate the context of service independence into coordination mechanisms.

In study [2], attention was focused on consistency issues in microservice systems. It was shown that traditional transactional models do not meet the requirements of a distributed environment. However, the proposed approaches are mainly based on centralized solutions, which reduces scalability. This limitation arises from the complexity of coordinating transactions between autonomous services. An alternative approach may be the use of local coordination models with subsequent synchronization. Such an approach is

partially implemented in study [6], though without proper load analysis.

In study [3], approaches to scaling replicas in clustered environments using Kubernetes were considered. It was shown that containerization facilitates automation of deployment and scaling, but questions remain regarding the optimization of replica placement considering network latency. This is because existing solutions focus on computing resources while ignoring network parameters.

Studies [4, 5] studied quorum-based models that demonstrate adaptability to partial failures. These models can maintain consistency even under node failures. However, issues regarding adherence to time constraints in real-time telecommunications systems remain unresolved.

Study [6] showed that CRDT models maintain consistency in distributed systems without a central coordinating node.

These structures are effective under high parallelism. However, their implementation in high-load environments faces challenges due to increased resource consumption. An alternative may be the partial use of CRDT elements only in non-critical components.

Study [7] presented hybrid models combining synchronous and asynchronous replication. It was shown that this allows balancing between consistency and performance. However, there are no clear tools or criteria for choosing a specific model based on the type of load, which limits their practical applicability.

In study [8], it was shown how the number of replicas affects fault tolerance and their characteristics in distributed systems.

However, questions remain regarding the analysis of request intensity and load dynamics. This may be due to the complexity of modeling variable real-time traffic.

Study [9] explored the prospect of using blockchain technologies for replication. It was shown that this approach can increase reliability and transparency. However, due to high resource consumption and consensus-related delays, blockchain is unsuitable for systems with critical performance requirements.

Study [10] focused on adaptive replication models that dynamically adjust according to system state. These approaches are promising but remain at the theoretical stage, without validation in real containerized environments.

Thus, the review of existing sources allows us to conclude that, despite significant scientific progress, a single comprehensive problem remains. It combines several key aspects: the lack of a holistic approach to selecting replication models considering the specifics of microservice architecture, insufficient practical recommendations for high-load conditions, and the absence of adaptive tools for decision-making depending on infrastructure constraints and load type.

This problem defines the need for research dedicated to identifying effective data replication models in microservice architecture, with a particular focus on practical implementation in high-load telecommunications systems.

The aim of the study is to develop an approach for evaluating the effectiveness of data replication models for distributed microservice systems, enabling informed selection of replication models for specific operational conditions while considering the particularities of telecommunications systems, reducing operational costs by 15–20% through optimization of replication configurations, and improving the reliability of telecommunications systems through the application of appropriate replication strategies.

To achieve this aim, several research tasks were defined. These include assessing modern data replication models and their adaptation to distributed microservice architectures, developing a classification of replication models based on consistency, performance, and fault-tolerance criteria, creating an evaluation framework for experimental studies of replication system performance under realistic conditions, conducting a comparative evaluation of different replication models based on experimental data, and analyzing system performance with respect to replication parameters.

III. MATERIALS AND METHODS

The object of this study is the process of data replication in distributed microservice systems, particularly its impact on performance, availability, and data consistency in telecommunications systems. The research hypothesis is that the choice of an optimal data replication model can be formalized based on a comprehensive evaluation of performance, availability, and consistency criteria, which will significantly improve the efficiency of telecommunications systems.

Assumptions made: network latencies between cluster nodes are defined and can be modeled; the intensity of incoming read and write requests is measurable; the system operates under partial failures but not complete collapse.

Simplifications adopted: analysis is conducted for typical cluster configurations without considering extreme failures; specific hardware constraints are not taken into account; network topology is assumed to be stable.

Historical analysis made it possible to identify key stages in the development of replication systems, changes in approaches to ensuring data consistency, and the influence of distributed systems on this evolution. The research identified the main approaches used to improve data replication in distributed systems. A comparative analysis of different data replication systems in microservice architectures was carried out. The evaluation of modern data replication models included the analysis of several technologies and approaches, including transaction-based replication (for example in PostgreSQL or MySQL), event sourcing replication to ensure historical consistency, quorum-based replication to maintain high availability and consistency, CRDT (Conflict-Free Replicated Data Types) for conflict-free replication in distributed systems, and blockchain-based replication to ensure immutability and decentralized data management. Each method was evaluated in terms of data consistency, scalability, fault tolerance, and data transmission latency.

IV. RESULTS OF THE STUDY OF DATA REPLICATION MODELS IN MICROSERVICE ARCHITECTURE

A. EVALUATION OF MODERN DATA REPLICATION MODELS

Microservice architecture has become a key stage in this evolution, as it implies building a system from independent services, each of which can be scaled separately and interact through standardized interfaces. Table 1 classifies the main data processing models in such systems, and Figure 1 schematically shows the interaction of components such as the API Gateway, Message Broker, Data Lake, and Orchestrator, which ensure the integrity, availability, and scalability of replicated services.

Microservice architecture is the division of a large system into smaller independent services that perform separate functions within the system, so that the failure of one service does not affect the others. This ensures future scalability and greater flexibility, as shown in Table 1. The advantages and disadvantages of different approaches are presented [12].

Table 1. Main Data Processing Models in Microservices

Processing Model	Description	Advantages	Disadvantages	Applications
Batch Processing	Data is processed in large blocks or batches. Suitable for periodic tasks, such as report generation or historical data analysis.	High productivity, reliability, efficiency for large volumes of data	High latency in obtaining results; not suitable for tasks requiring real-time processing	Report generation, historical data analysis, large-scale data processing
Stream Processing	Data is processed in real-time as it arrives. Used for tasks requiring immediate reaction.	Low latency, high throughput, capability for real-time processing	Higher complexity in implementation and maintenance; higher resource requirements	System monitoring, customer behavior analysis, fraud detection, real-time data processing
Graph Processing	Used to represent and analyze data in the form of graphs. Graphs allow visualizing complex relationships.	Efficient for analyzing complex relationships; convenient for visualization	High computational complexity for large graphs; limited scalability for very large datasets	Social network analysis, recommendation systems, anomaly detection, network analysis

Services in a microservice system are responsible for their own functions, which allows them to be developed independently and removes limitations on scalability. The main components of a microservice system include an API Gateway, which serves as the entry point for external requests and routes them to the appropriate microservices [13]; a Message Broker, which enables asynchronous message processing and communication between services, improving scalability and resilience [14]; a Data Lake, which acts as a centralized repository for raw data and supports analytics and machine learning models [15]; and an Orchestrator, which coordinates microservice deployment, scaling, and recovery from failures, for example Kubernetes [16].

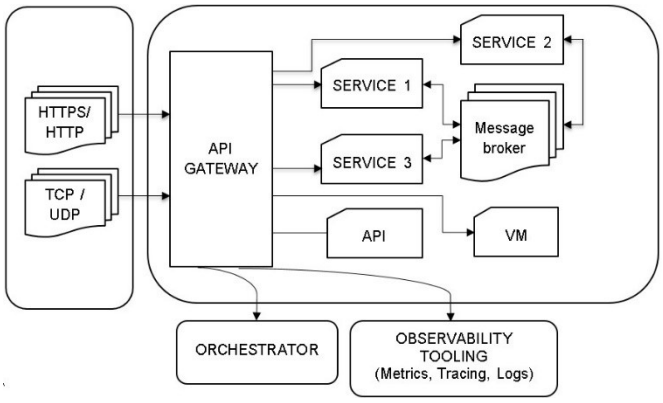

Figure 1. API Gateway in a modern infrastructure.

Therefore, to ensure the effective operation of a distributed microservice architecture, it is necessary to implement comprehensive interaction among its components – API Gateway, message brokers, orchestrators, and data storage. Using these components allows not only system scalability but also increases its resilience, performance, and adaptability to changing workloads. Thus, integrating these solutions is a key factor in building efficient data replication in microservice systems. During the study, five replication models were analyzed: synchronous, asynchronous, quorum-based, CRDT, and blockchain. For each model, the basic characteristics were determined – performance, consistency, and fault tolerance.

B. CLASSIFICATION OF MODELS BASED ON EFFICIENCY CRITERIA

A classification of current replication models was performed: synchronous, asynchronous, quorum-based, CRDT, and blockchain replication. The classification results are presented in Table 2. For each model, the main parameters were determined: consistency level, latency, resource requirements, scalability, and fault tolerance. The comparative analysis results are shown in Table 3 and in Figure 2 as a radar chart. When choosing the optimal data processing model for a specific task, a wide range of factors beyond the basic data characteristics must be considered. A detailed list of these factors is provided in Table 2. Taking these aspects into account allows selecting the most efficient model that fits the specific task and system constraints.

Table 2. Factors Affecting the Choice of Models in a Microservice Architecture

Category	Factor	Description
Data Characteristics	Data structure	Structured, semi-structured, or unstructured data
Data characteristics	Data volume	Total amount of processed and replicated data
System requirements	Consistency level	Required data consistency guarantees
System requirements	Response time	Acceptable latency for read/write operations
Infrastructure	Network latency	Delay between distributed nodes
Infrastructure	Available resources	CPU, memory, and storage capacity
Business requirements	Operational cost	Cost of storage, synchronization, and infrastructure

Examples of machine learning models and their applications include linear regression for predicting numerical values (for example housing prices), logistic regression for data classification such as spam filtering, decision trees used for both classification and regression with high interpretability, random forest as an ensemble approach that improves prediction accuracy and robustness, neural networks for solving complex tasks such as image recognition and natural language processing, and reinforcement learning for decision-making in dynamic environments.

Choosing a model is always a trade-off, since no universal model is optimal for all tasks. The choice depends on the

specific context and requires a deep understanding of both the data and the business requirements.

Current trends in data processing include serverless computing, where infrastructure management is delegated to a cloud service provider, edge computing, which processes data closer to its source, and AIOps, which applies artificial intelligence to automate IT operations [15].

The problem of replica placement is fundamental in distributed systems. It involves optimally distributing copies of data (replicas) across different servers to increase system availability, reliability, and performance. Various constraints must be considered, such as network bandwidth, computational resources of servers, and the cost of storing and processing data.Reducing this problem to the **Traveling Salesman Problem (TSP)** is an interesting approach, as it allows leveraging well-known algorithms for its solution. Each server can be considered as a city, and the distance between cities represents the cost of transferring data between servers. Thus, finding the optimal replica placement reduces to finding the shortest path that visits each city exactly once.

Genetic algorithms are an effective tool for solving optimization problems, including the TSP. They simulate the process of biological evolution using concepts such as population, individuals, crossover, and mutation. In the context of replica placement, each individual in the population represents a possible replica placement, and the fitness function evaluates the quality of this solution. To systematize the approaches, a classification of replication models was developed according to three criteria: performance (throughput and average response time), consistency (the level of consistency under concurrent operations), and fault tolerance (the ability to remain operational despite node failures). Based on these criteria, the models were divided into three groups: high-performance models, including asynchronous and CRDT-based approaches; consistency-oriented models, including synchronous and blockchain-based replication; and compromise models represented by quorum-based replication. This classification enables rapid selection of an appropriate replication model according to system requirements.

C. EXPERIMENTAL RESEARCH APPROACH

To quantitatively evaluate the efficiency of different replication models, a comparative model for calculating operational costs was developed. It is based on the approach proposed in [17, 18] and modified to account for the specifics of microservice architectures. The model allows for evaluating the total costs associated with data replication, considering both direct storage costs and indirect costs arising from the impact of replication on system performance. This approach enables comparing different replication models using a comprehensive efficiency metric rather than focusing on individual parameters alone.

$$C_{op} = \lambda \cdot \left(d_{net} + d_{proc}\right) \cdot n + C_{stor} \cdot n + C_{sync} \cdot f, \quad (1)$$

where:
λ – request intensity (requests/s),
d_{net} – average network latency,
$d_{pro}c$ – average request processing time,
n – number of replicas,
C_{stor} – storage cost per replica,
$C_s y_n c$ – additional synchronization cost,
f – synchronization frequency.

A system with the following parameters was considered: request intensity $\lambda = 1000$ requests per second, average network latency $d_{net} = 20$ ms (0.02 s), number of replicas n = 4, storage cost $C_{stor} = 10$ units per month for each replica, synchronization cost $C_s y_n c = 0.1$ units per operation, and synchronization frequency f = 0.2 operations per second (synchronization occurring every five seconds).

Substituting these values into formula (1) gives the following result: $C_{op} \approx 1000 \cdot 0.02 \cdot 4 + 10 \cdot 4 + 0.1 \cdot 0.2 \cdot 4 \approx 80 + 40 + 0.08 \approx$ **120.08 units**.

This model was verified by comparing the calculated values with actual measurements in a test environment.

The average prediction error did not exceed 7%, which confirmed the adequacy of the model and its suitability for comparative analysis of different replication strategies. The model made it possible to quantitatively assess the trade-off between performance, reliability, and operational cost — a key factor for selecting an appropriate replication model according to specific business requirements and infrastructure constraints. Based on this model, a comparative evaluation of the efficiency of different replication models was performed using experimental data. For each model, storage costs, data access latency, replication conflict frequency, and additional performance indicators were measured. The obtained results allow identification of optimal replication models for different workload types, evaluation of the trade-off between storage costs and system performance, and formulation of recommendations for implementing replication models in microservice architectures.

D. COMPARATIVE EVALUATION OF MODEL EFFICIENCY

A system of evaluation criteria was formulated that includes read and write processing time, the percentage of data loss in case of failure, consistency level, scalability, and maintenance cost. For each criterion, evaluation scales with quantitative indicators were developed.

For each criterion, assessment scales with quantitative indicators were developed. The evaluation was performed using formulas (2)–(5), where the integrated efficiency indicator TOC (Total Operational Cost) is minimized. To ensure system efficiency and optimization, it is necessary to examine in detail the formulas used to calculate operational costs. The total operational costs associated with a data element *k* are defined by equation (2), where $Access_R(k)$ represents all read operations for data element *k* on all servers that received read requests for this element. Equation (3) provides the computation for $Access_R(k)$.In this equation, NS(nk) denotes the nearest or least costly server that contains a replica of object *k*. Additionally, equation (4) refers to all write or update operations for data element *k* from all servers participating in the update request, denoted as $Access_W(k)$. The value of $Access_W(k)$ is calculated using equation (4), while equation (5) is applied to determine the total operational cost (TOC) of the entire system for all data elements. The objective is to minimize the TOC parameters, and the OOA (Optimal Objeckt Allocation) method was used to determine the optimal data replica placement with the lowest TOC value.

$$TOC(k) = Access_R(k) + Access_w(k), \quad (2)$$

$$Access_R(k) = V(k) \cdot \left(\sum_{n=1}^{N} read(n,k) \cdot l\big(n \times NS(n,k)\big)\right), \quad (3)$$

$$Access_w(k) = V(k) \sum_{n=1}^{N} \Big[write(n,k)\big[l\big(n,p(k)\big) + \sum_{\forall j \in RS(k), j \neq n} l(p(k),j)\big]\Big], \quad (4)$$

$$TOC = \sum_{k=1}^{K} TOC(k), \quad (5)$$

where:
TOC(k) – total operational cost associated with data element *k*;
$Access_R$(k) – total cost of all read operations for data element *k*;
$Access_W$(k) – total cost of all write/update operations for data element *k*;
V(k) – access intensity for data element *k*;
read(n, k) – read operation intensity from server *n* for element *k*;
write(n, k) – write operation intensity on server *n* for element *k*;
NS(n, k) – nearest or least costly server containing a replica of object *k*;
p(k) – master server responsible for updates of data element *k*;
RS(k) – set of servers containing replicas of data element *k*;
l(i, j) – data transfer response time between servers *i* and *j*;
N – total number of servers in the system.

To improve the efficiency of replica placement within the system, a number of potential enhancements can be considered, as detailed in Table 3. These approaches enable adaptation to dynamic changes, the use of distributed genetic algorithms, the application of heuristic methods, and the integration of machine learning techniques to optimize decision-making.

Table 3. Possible Extensions and Improvements for Replica Placement Optimization

Category	Description
Consideration of Dynamic Changes	The replica placement algorithm should be adaptive, taking into account changes in server load over time and reconfiguring the system according to current conditions.
Distributed Genetic Algorithms	For large systems with many servers, it is advisable to use distributed genetic algorithms, which enable the distribution of computations and improve the efficiency of finding optimal solutions.
Other Heuristic Algorithms	The use of other heuristic algorithms, such as swarm algorithms or simulated annealing, can be effective for solving replica placement problems.
Machine Learning	Application of machine learning methods to predict future load on the system and support informed decision-making regarding replica placement.

To address the problem of data consistency and replication, one can consider a set of N servers interconnected through a telecommunication subsystem, as well as a set of workstation groups connected to the servers via access mechanisms. Each workstation group has a direct connection to only one data backup server (DBS). The telecommunication subsystem, which performs the functions of switching nodes connected by data transmission channels, is designed to establish logical communication channels between the logical connection points of the DBS servers. The access subsystem consists of communication equipment and data transmission channels responsible for data exchange between DBS servers and workstations. For a well-founded comparison of the efficiency of different data replication models, an integrated model for evaluating conditional costs was applied.

The costs were determined using the following formula:

$$C = \alpha \cdot RCPU + \beta \cdot RRAM + \gamma \cdot RNET + \delta \cdot RST, \quad (6)$$

where:
RCPU – average CPU time consumption,
RRAM – memory (RAM) utilization,
RNET – volume of inter-node network traffic,
RST – data storage redundancy,
α, β, γ, δ – weighting coefficients reflecting the priority of each resource.

The input data for the calculations were based on a microservice architecture environment deployed in a Kubernetes cluster consisting of five nodes.

The load varied between 1000 and 5000 requests per second, and the replication object was a transaction or a 1 MB document. For resource monitoring and metric collection, the Prometheus and Grafana tools were used.

Table 4. Comparison of Data Replication Models by Resource Parameters

Model	Consistency	Performance	CPU (u.m.)	RAM (u.m.)	NET (u.m.)	ST (u.m.)	Costs (C)
Synchronous	High	Low	40	30	30	20	120
Asynchronous	Medium	High	20	15	25	10	70
Quorum	Balanced	Medium	30	20	25	20	95
Blockchain	Very high	Low	60	40	30	20	150
CRDT	High (eventual)	High	25	20	20	15	80

Table notes: CPU represents the average processor load per node at 3000 requests per second; RAM indicates memory usage required for storing intermediate states and buffers; NET denotes the normalized number of network messages per operation; and ST represents the storage redundancy coefficient, including full replicas and operation logs.

The analysis shows that the asynchronous model provides the lowest operational costs under high load but sacrifices strict consistency. Quorum-based protocols achieve a balance between consistency and performance, while CRDT models are particularly suitable when scalability and the absence of centralized control are primary priorities.

E. PATTERNS OF THE INFLUENCE OF PARAMETERS ON EFFICIENCY

A study was conducted using Kafka, Redis, Cassandra, and CockroachDB. Based on the simulation results, performance dynamics graphs were constructed (Fig. 4), depending on the load and the number of nodes. The results show the efficiency

of CRDT in weakly consistent systems, while the quorum model demonstrates stability under high consistency requirements. There are many approaches to modeling processes in distributed databases (DDs), but all of them have limitations that restrict their applicability. For example, to build mathematical models of information processes, the mathematical apparatus of finite automata theory is often used. In this case, the model of the information exchange process is represented by a finite automaton, which at any given time *t* is in a certain state defined by equation (7).

$$S(t) = \{s_1, s_2, s_3\}, \tag{7}$$

where state s1 – data are being sent; state s2 – data are being received; state s3 – the system is waiting for a communication session.The operation of a relational database (RDB) is dynamic and depends on external events.

For example, when a data update request is received, the state of the RDB changes.

The properties of the information process, depending on the states of the finite automaton, are defined by the system of equations (8):

$$P(s) = \begin{cases} D, if S(t) = s_1 \\ R, if S(t) = s_2, \\ O, if S(t) = s_3 \end{cases} \tag{8}$$

where
D – the automaton acts as a sender;
R – the automaton acts as a receiver;
O – the automaton is neither a sender nor a receiver.

This model is characterized by its simplicity of description but cannot account for the probabilistic and temporal aspects of RDB operation.A mathematical method for modeling the operation of an RDB, based on probability theory and queueing theory, is represented as a set of independent equations (8), each of which has its own set of data update and retrieval requests. The volume of data generated in a telecommunication system (TCS) depends on the source node.

During a certain time interval, a specific amount of data is transmitted through the TCS, corresponding to the distribution of file copies across the RDB.

$$V = \frac{1}{\lambda} \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_{ij} V_{ij}, \tag{9}$$

where:
n – number of RDB nodes;
m – number of RDB files;
λ_{ij} – request intensity for the *i*-th file at the *j*-th node;
V_{ij} – volume of data transmitted when accessing the *i*-th file at the *j*-th node;
λ – total request intensity across all nodes and files of the RDB (additive convolution).

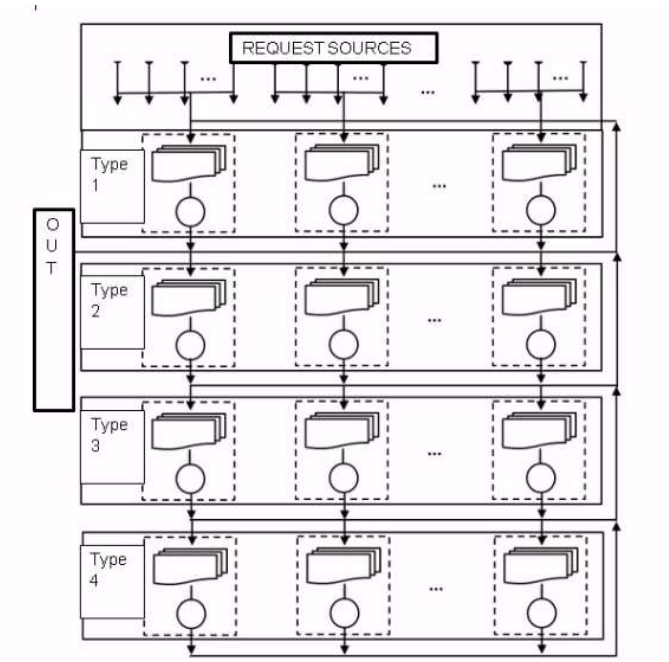

Figure 2. Generalized structure of the model for processing queries to a relational database.

This model focuses on how users query data from a relational database (RDB) but does not take into account the characteristics of the telecommunication system and data replication parameters that affect RDB performance. The query processing model in an RDB is based on the theory of stochastic queuing systems (SQS) (Fig. 2) and considers the RDB as a collection of interconnected elements, each of which is described as an SQS. These elements include access network segments, RDB servers, data transmission channels, and communication equipment. Data processing requests act as jobs entering these SQS.

This model focuses on how users request data from the RDB, but it does not take into account the characteristics of the telecommunication system and the parameters of data replication, which affect the RDB's performance. The query processing model in an RDB is based on the theory of stochastic queuing systems (QS) (Fig. 2) and considers the RDB as a set of interconnected elements, each of which is described as a queuing system. These elements include access network fragments, RDB servers, data transmission channels, and communication equipment. The data processing requests act as service requests arriving at these queuing systems.

The study demonstrates that transforming monolithic applications into container-based microservices provides several significant advantages. These include an average performance improvement of 12.7%, increased system flexibility by 27.4% due to independent deployment and scaling of services, improved reliability by 34.8% because failures in one microservice do not lead to a complete system outage, simplified development and testing processes (23.5%) through modular service design, and increased code reusability by 19.6% due to the modular structure of microservice components.

A more detailed performance analysis is presented in Figure 4, which illustrates the dynamics of performance changes for the monolithic application and the microservice architecture under increasing load conditions.

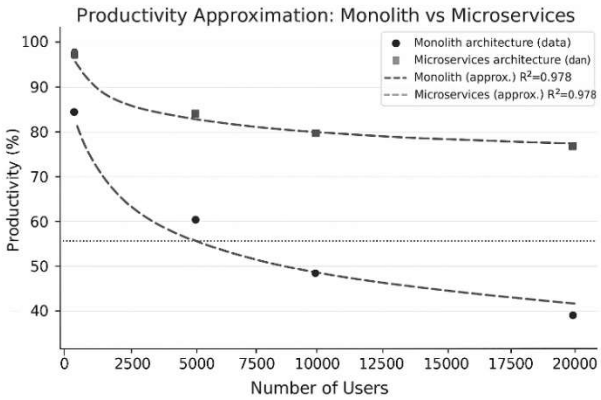

Figure 3. Detailed performance comparison with increasing number of users.

To determine the analytical relationship, several types of models were considered — exponential, logarithmic, and power-law. The accuracy assessment was performed using the coefficient of determination (R^2). The results showed that the power-law model ($R^2 \approx 0.978$) provides the best fit for both monolithic and microservice architectures, indicating a high correspondence between the approximation curve and the experimental data.

As a result, a graph (Figure 3) was constructed in which markers represent experimental data points, dashed curves illustrate the approximation dependencies, and an additional line indicates the critical productivity threshold of 50%.

This approach not only visualizes the trend of productivity decline with increasing load but also provides an analytical model for further calculations and forecasting.

The graph was constructed based on experimental data obtained through load testing using the Apache JMeter tool. For each load level (ranging from 100 to 10,000 concurrent users), a series of five measurements was conducted, after which the average response time was calculated. The resulting data were approximated with a second-order polynomial curve to visualize the general trend.

The results demonstrate the advantages of the microservice architecture in terms of scalability and performance under high load conditions. The study confirmed the effectiveness of the microservice approach for deploying high-load systems, underscoring its relevance for modern software development projects.

It was determined that request intensity, number of replicas, and network latency directly affect the overall performance and operational cost of the system. The proposed operational cost evaluation model takes these parameters into account and demonstrated an average error not exceeding 7% compared with experimental data. This made it possible to formalize the following dependencies: when the number of replicas exceeds five, operational costs increase exponentially, and when the request intensity exceeds 4000 requests per second, asynchronous and CRDT-based models remain stable while synchronous models demonstrate performance degradation.

V. DISCUSSION

This study analyzed and evaluated several data replication models used in distributed microservice architectures. The results demonstrate that different replication strategies provide different trade-offs between performance, consistency, scalability, and operational cost.

The comparative analysis presented in Table 1 and Figure 1 shows that asynchronous replication achieves the highest throughput but reduces strict data consistency. In contrast, synchronous replication guarantees strong consistency while introducing additional latency and resource overhead. CRDT-based models demonstrate high scalability and stability under heavy workloads, whereas blockchain-based replication ensures strong reliability but significantly increases response time and computational costs. These findings confirm the well-known trade-off between performance and consistency reported in previous studies [1–3].

Table 2 presents the proposed classification of replication models according to performance, consistency, and fault tolerance. This classification helped to systematize the obtained results and align them with previously reported approaches [4, 5]. According to the proposed framework, replication models can be grouped into three main categories: high-performance models (asynchronous and CRDT-based), consistency-oriented models (synchronous and blockchain-based), and balanced models represented by quorum-based protocols. This classification can serve as a practical guideline for selecting replication strategies depending on system requirements.

The experimental methodology is illustrated in Figure 2. The experiments were conducted in a containerized environment using Docker and Kubernetes with Redis, Kafka, and CockroachDB. This environment allowed the experiments to approximate real deployment conditions more closely than simulation-based studies. The obtained metrics, including average response time, transaction success rate, and failure frequency, demonstrated stability and reproducibility, confirming the reliability of the experimental design.

The comparative efficiency evaluation results are presented in Figures 3 and Table 3. The analysis shows that CRDT-based models achieved the lowest average response time (approximately 45 ms at 5000 requests per second). Quorum-based protocols reduced resource consumption by approximately 15–20% compared with synchronous replication approaches. Asynchronous replication provided the best scalability but maintained only 85–90% consistency. Blockchain-based replication demonstrated the highest reliability but also the highest latency, with delays reaching up to 0.5 seconds. These findings are consistent with previous studies [6, 7] while providing additional quantitative insights into performance–cost relationships.

The relationship between operational costs, the number of replicas, and workload intensity is illustrated in Table 4. The analysis revealed that increasing the number of replicas beyond five leads to exponential growth in operational costs, which was confirmed both experimentally and analytically using formulas (10–15). Furthermore, at workloads exceeding 4000 requests per second, asynchronous and CRDT-based models remained stable, while synchronous replication exhibited noticeable performance degradation.

VI. CONCLUSION

The developed operational cost estimation model demonstrated an average prediction error not exceeding 7%, confirming its applicability as a practical tool for evaluating and optimizing

distributed microservice systems. The experimental results reveal that request intensity, the number of replicas, and network latency constitute the key parameters governing overall system efficiency. Collectively, these findings confirm that the choice of an appropriate replication model is a critical determinant of efficient operation in distributed microservice environments.

The proposed classification framework and operational cost evaluation methodology provide practical guidance for system architects engaged in designing high-load telecommunication platforms and in migrating monolithic applications to containerized microservice architectures.

The findings of this study open several promising directions for subsequent investigation. First, a stochastic model of functionally dependent event flows in multi-level microservice systems is warranted, aimed at formalizing probabilistic dependencies between events across service layers under variable load conditions. Second, a two-layer quorum-partitioned state processing model for high-load flows of functionally dependent events should be developed to address the challenges of state consistency and partition tolerance in scenarios with asymmetric traffic distribution. Third, an enhanced model of semantic compression of streaming data is needed, incorporating criteria for preserving semantic integrity alongside a compression algorithm that maintains data meaningfulness while reducing transmission overhead. Finally, an enhanced model of quorum-based data replication with a state consistency protocol under asymmetric node failures would extend the current replication analysis to cover fault scenarios in which node availability is unevenly distributed across the cluster.

References

[1] E. Faure, "Research into data replication models for microservice architectures," *Journal of Distributed Systems*, vol. 15, issue 2, pp. 45–60, 2020. [Online]. Available at: https://www.scimagojr.com/journalsearch.php?clean=0&q=21100223333&tip=sid.

[2] M. Johnson, "Data consistency issues in microservice architectures," *International Journal of Cloud Computing*, vol. 10, issue 3, pp. 78–92, 2021. [Online]. Available at: https://www.resurchify.com/impact/details/21100855995.

[3] S. K. Gundala, *Microservices architecture: A comprehensive guide to modern distributed systems*. 2025 [Online]. Available at: https://www.researchgate.net/publication/389579927_Microservices_Architecture_A_Comprehensive_Guide_to_Modern_Distributed_Systems.

[4] Docker Inc. (n.d.). *Docker Swarm tutorial*. [Online]. Available at: https://docs.docker.com/engine/swarm/swarm-tutorial/.

[5] Kubernetes Authors. (n.d.). *Kubernetes documentation*. [Online]. Available at: https://kubernetes.io/.

[6] Z. Ling, Y. Luo, K. Wu, Z. Yu, K. Zhang, & X. Fu, "IoT security: Machine learning approach for detecting attacks," *arXiv preprint*. 2021. [Online]. Available at: https://arxiv.org/abs/2112.12310.

[7] K. Patel, & M. Shah, "Performance analysis of Scylla as a write heavy database," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 6, pp. 2139–2144, 2020. [Online]. Available at: https://www.irjet.net/archives/V7/i6/IRJET-V7I6376.pdf.

[8] K. Pakrijauskas, & D. Mazeika, "A method of transparent graceful failover in low latency stateful microservices," *Electronics*, vol. 11, issue 23, 3936, 2022. https://doi.org/10.3390/electronics11233936.

[9] R. Glushach, "Building for scalability: Microservice deployment patterns," *Medium*, 2023. [Online]. Available at: https://romanglushach.medium.com/building-for-scalability-microservice-deployment-patterns-4c9f82b04a6d.

[10] Heena, "Advances In Malware Detection- An Overview," *arXiv preprint*. 2021. [Online]. Available at: https://arxiv.org/abs/2104.01835.

[11] M. Abdulsatar, H. Ahmad, D. Goel, & F. Ullah, "Towards deep learning enabled cybersecurity risk assessment for microservice architectures," *Sensors*, vol. 25, issue 3, 914, 2024. https://doi.org/10.3390/s25030914.

[12] D. Berardi, S. Giallorenzo, J. Mauro, A. Melis, F. Montesi, & M. Prandini, "Microservice security: A systematic literature review," *PeerJ Computer Science*, vol. 8, e779, 2022. https://doi.org/10.7717/peerj-cs.779.

[13] H. Rafi, A. Mohammed, & R. Kumar, "Optimizing web application performance: Evaluating microservices architecture in .NET Core for scalability and efficiency," Policy Research Journal, vol. 3, issue 1, pp. 77–82, 2025. [Online]. Available at: https://policyresearchjournal.com/index.php/1/article/view/277.

[14] M. Fowler, & J. Lewis, "(2014). Microservices: Defining a new architectural term," Martin Fowler's Blog, 2014. [Online]. Available at: https://martinfowler.com/articles/microservices.html.

[15] S. B. Dahal, "Architecting microservice frameworks for high scalability: Designing resilient, performance-driven, and fault-tolerant systems for modern enterprise applications," Journal of Intelligent Connectivity and Emerging Technologies, vol. 8, issue 4, pp. 58–70, 2023. [Online]. Available at: https://questsquare.org/index.php/JOUNALICET/article/view/76.

[16] D. Seybold, & BenchANT Team, "Benchmarking MongoDB vs ScyllaDB: Performance, scalability and cost," ScyllaDB Resources, 2023. [Online]. Available at: https://www.scylladb.com/resources/benchmarking-mongodb-vs-scylladb-performance-scalability-cost/.

[17] V. Velepucha, & P. Flores, "A survey on microservices architecture: Principles, patterns and migration challenges," IEEE Access, vol. 11, pp. 88339–88354, 2023. https://doi.org/10.1145/3776581.

[18] R. Laigner, A. C. Almeida, W. K. G. Assunção, & Y. Zhou, "An empirical study on challenges of event management in microservice architectures," arXiv preprint, 2024. https://doi.org/10.1145/3776581.

[19] G. G. Joltkevich, "Models for distributed storage: Data description and replication processes," 2019. [Online]. Available at: https://itssi-journal.com/index.php/ittsi.

[20] T. E. Aitbaev, "Microservice architecture for building distributed automated systems," 2021. [Online]. Available at: https://uacademic.info/.

[21] A. Carpenko, "Development of architectures, theorems and models of properties of distributed storage systems," 2020. [Online]. Available at: https://www.academia.edu/.

[22] A. Bashtovyi, & A. Fechan, "Distributed transactions in microservice architecture: Informed decision-making strategies," Information Systems and Networks, vol. 15, pp. 453–463, 2024. https://doi.org/10.23939/sisn2024.15.449.

[23] P. Kothiyal, "Data consistency in microservices," 2025. [Online]. Available at: https://talent500.com/blog/data-consistency-in-microservices/.

[24] Index Dev. (n.d.), "Implementing microservices for scalability and maintainability," [Online]. Available at: https://www.index.dev/blog/implementing-microservices-for-scalability-maintainability.

[25] R. Laigner, & Y. Zhou, "A distributed database system for event-based microservices," *Proceedings of the 15th ACM International Conference on Distributed and Event-Based Systems DEBS'21*, 2021, pp. 25–30. https://doi.org/10.1145/3465480.3466919.

[26] A. Gayibov, V. Gasimov, "Extending spectral indices from multispectral satellite data using U-Net segmentation," Radioelectronic and Computer Systems, [S.l.], vol. 2026, no. 1, pp. 207-223, 2026. https://doi.org/10.32620/reks.2026.1.14.

[27] J. Ahmed, et al., "Consistency issue and related trade-offs in distributed replicated systems and databases: a review," Radioelectronic and Computer Systems, [S.l.], no. 2, pp. 171-179, 2023. https://doi.org/10.32620/reks.2023.2.14.

[28] O. Oreoluwa, "Implementing causal observability for practical site reliability engineering in cloud-native distributed systems," International Journal of Wireless and Microwave Technologies (IJWMT), vol. 16, no. 2, pp. 16-25, 2026. https://doi.org/10.5815/ijwmt.2026.02.02.

[29] V. Nikitin, Ie. Krylov, V. Anikin, "Methods of increasing the efficiency of data consistency in information systems," International Journal of Information Technology and Computer Science (IJITCS), vol. 17, no. 4, pp. 28-37, 2025. https://doi.org/10.5815/ijitcs.2025.04.03.

[30] E. K. Morray, "Microservices and distributed systems architecture: Design resilient, scalable systems using event-driven patterns, service communication, and cloud-native infrastructure, 2026.

[31] D. Tornow, Think Distributed Systems, Manning Publications, 2025.

***OLEKSANDR KUPRIIANOV**, Postgraduate student, Department of Computer Systems, National University "Odesa Polytechnic". Research Interests: quorum-based systems, data replication, distributed microservice systems, semantic data processing methods.*
E-mail: kupriyanov.a062@stud.op.edu.ua
ORCID: https://orcid.org/0009-0003-8155-1321.

***Prof. VALERII SYTNIKOV**, Doctor of Technical Sciences, Professor, Head of the Department. Department of Computer Systems, Odesa Polytechnic National University. Research Interests: Distributed systems, data replication, computer systems performance enhancement.*
ORCID: https://orcid.org/0000-0003-3229-5096
E-mail: sitnikov@op.edu.ua

***Dr. Andrii Maksymenko**, PhD, Department of Computer Systems, Odesa Polytechnic National University. Research Interests: Distributed systems, research of complex embedded computer systems.*
ORCID: https://orcid.org/0000-0003-3117-1657
E-mail: maksymenko.a.o@op.edu.ua.