UDC 004.4'242:004.75:004.451.3
DOI https://doi.org/10.32782/2663-5941/2025.6.2/47

Terletska K.V.
Lviv Polytechnic National University

PERFORMANCE EVALUATION OF BLUE-GREEN DEPLOYMENT MODELS FOR STATEFUL MICROSERVICES

The relevance of the study is driven by the increasing complexity of distributed microservice architectures and the need to ensure uninterrupted service operation during updates, particularly in stateful systems. Under these conditions, the blue-green deployment (BGD) model becomes a critically important tool for maintaining data consistency, transactional integrity, and stable performance. However, its application to stateful components is accompanied by substantial technical and organizational constraints.

The purpose of the article is to substantiate the methodological foundations for evaluating the performance and reliability of BGD in stateful microservices within cloud computing, followed by the identification of effective environment-switching strategies that ensure continuous service operation.

The research methods were based on a systematic analysis of architectural solutions, a comparative assessment of replication and logging mechanisms, modeling of various traffic-switching scenarios, and an instrumental analysis of latency, throughput, and fault tolerance metrics. Formalized approaches to evaluating stateful components during updates and methods for reproducing real workload behavior were applied.

The results of the study demonstrate the critical influence of the replication type, transaction isolation level, change-application sequence, and behavior of external dependencies on state consistency during BGD operation. It was shown that gradual switching and mirrored testing provide the most stable performance indicators in complex microservice ecosystems, whereas full switching is advisable only in environments with predictable workloads. A set of issues limiting the scalability of BGD was identified, including risks of state divergence, throughput degradation during intensive synchronization, and inconsistencies in caches and message queues.

The conclusions confirm that the effectiveness of BGD in stateful services is determined by the balance between switching speed, replication quality, and the level of transaction control. Recommendations for selecting a BGD strategy are summarized depending on service profile, reliability requirements, and the structural characteristics of stored state.

Future research prospects include the development of formal consistency models for updates, the implementation of machine learning algorithms for adaptive replication and routing management, and the creation of self-learning systems capable of optimizing environment switching in accordance with real-time workload fluctuations.

Key words: *data consistency, transactional integrity, state replication, processing latency, system throughput, service fault tolerance, cloud computing, microservice architecture, zero-downtime deployment.*

Formulation of the problem. Blue-green deployment, a parallel version rollout model used to switch between software environments known as the blue and green versions, is a key mechanism for ensuring uninterrupted service operation in cloud computing (CC) environments. However, its application to stateful microservices remains scientifically and technologically unresolved. The challenge lies in the fact that approaches effective for stateless services do not account for the complex dependencies between state, data, and transactional consistency. Switching between the blue and green environments in stateful components introduces risks of data loss, replication failures, temporal inconsistencies, and access conflicts, which undermines the main advantage of the model: the ability to perform updates without downtime.

In practical terms, this leads to reduced reliability of mission-critical systems, increased infrastructure maintenance costs, and the need for complex manual procedures for state migration. Existing approaches demonstrate insufficient performance under high load, limited scalability, and incompatibility with strict data consistency requirements. From a scientific perspective, there is a need to formalize models describing the behavior of stateful components during environment switching, to develop accurate methods for evaluating blue-green deployment performance, and to propose architectural solutions capable of minimizing latency, access conflicts, and risks of service degradation. The relevance of this problem is driven by the growing complexity of microservice ecosystems, the proliferation of distributed data stores and transactional protocols, and the need to establish practical guidelines for engineers designing highly available, secure, and fault-tolerant information systems.

Analysis of recent research and publications. A review of current research on evaluating the performance of blue-green deployment models for stateful microservices makes it possible to identify four interrelated scientific directions. The first direction concerns the conceptualization of zero-downtime architectures and the role of blue-green updates in cloud-native and containerized systems. In the work of K. K. Pappula, mechanisms for achieving zero downtime in full-stack environments are substantiated, where the blue-green approach ensures version isolation and controlled traffic switching [1]. G. Stencel and L. Berton examine the configuration specifics of stateful applications in Kubernetes, emphasizing the complexity of state migration during updates [2]. D. Jaiswal investigates cloud-native reconstruction of SIP/IMS architectures, demonstrating that performance and latency constraints determine the limits of zero-downtime updates in telecommunication services [3]. R. A. Perumal systematizes Azure architectural patterns and highlights the role of blue-green schemes in maintaining SLA and fault tolerance in large-scale systems [4].

The second direction focuses on the development of scalable microservice architectures and CI/CD pipelines that enable controlled zero-downtime updates. R. Tsyganok analyzes the principles of building high-load microservice platforms in Go, where performance depends on the selected update and traffic-balancing strategy [5]. S. Arya and co-authors review enterprise adaptation to Kubernetes-oriented microservice architectures, comparing rolling, canary, and blue-green strategies in terms of their impact on performance and fault tolerance [6].

S. Amgothu proposes optimized CI/CD models in which the combination of canary and blue-green updates minimizes the risk of performance degradation in production environments [7]. G. R. Enjam and co-authors demonstrate, using an insurance SaaS solution as an example, how blue-green deployment affects both technical and business metrics, substantiating its effectiveness for systems with complex state structures [8].

The third direction concerns the comparison of zero-downtime deployment strategies and the evaluation of their risks, costs, and performance impact. E. Truyen and co-authors develop a model of flexible migration in blue-green updates under fixed budget constraints, establishing the relationship between resource limitations and update quality [9]. D. S. Antiya analyzes the application of blue-green infrastructure as a compliance and security tool in cloud environments, where state synchronization and parallel performance monitoring are essential [10]. A. D. Ikhsan and co-authors compare different deployment models, demonstrating that the blue-green approach provides higher availability and stability but requires increased infrastructure resources [11]. C. K. Rudrabhatla synthesizes the differences among zero-downtime methods in public cloud infrastructures based on their complexity, cost, and risk of performance degradation [12].

The fourth direction concerns the implementation of hybrid strategies and dynamic update management in complex industrial and mission-critical systems. P. Rajković and co-authors propose a hybrid deployment strategy for industrial systems that combines multiple update patterns and ensures a balance between performance, reliability, and resource constraints [13]. A. U. Rehman and co-authors develop a multicriteria optimization model for configuring blue-green-type infrastructures, demonstrating the importance of selecting an optimal resource structure under conditions of conflicting objectives [14]. D. Mailer and colleagues analyze dynamic deployment of failure detection models in industrial systems, showing that runtime updates of components require specialized mechanisms for state synchronization and performance monitoring [15].

Despite significant progress in microservice architecture research, a number of aspects related to the application of BGD in stateful services remain unresolved. The influence of state structural characteristics on consistency during version switching is insufficiently studied, universal replication and logging models for dynamic high-load environments have not been established, and comprehensive performance metrics for different BGD variants remain limited. Furthermore, scientific and operational risks associated with access conflicts, performance degradation, and the uncertainty of external dependencies have not been fully synthesized. The proposed study aims to address these gaps by performing a systematic analysis of stateful components during updates, refining replication mechanisms and conditions for ensuring consistency, formulating performance metrics for BGD variants, and identifying critical issues in their practical application. The findings provide a methodological foundation for developing optimal update strategies and deepen the scientific understanding of stateful service behavior under continuous operation.

Task statement. The aim of this article is to develop scientifically grounded approaches for evaluating the performance and reliability of the BGD model for stateful microservices in CC environments, with a focus on identifying optimal strategies for its application. To achieve this aim, the article sets out the following objectives:

1. To analyze the characteristics of stateful microservices in the context of their interaction with the BGD model and to identify the factors that influence state consistency during deployment.
2. To examine data preservation and replication mechanisms and to assess how different BGD implementation options affect performance, consistency, and transaction correctness.
3. To identify scientific, technical, and operational constraints on the use of BGD and to develop recommendations for selecting the optimal strategy for stateful microservices.

Outline of the main material of the study. Stateful microservices require specialized mechanisms to maintain state consistency when the BGD model is applied, as data persistence is an integral part of their operation. Unlike stateless services, where version switching does not affect the integrity of request processing, stateful components depend on the accuracy and consistency of the local or distributed state. Their behavior is shaped by the type of state storage, transaction semantics, replication mechanisms, and data access protocols, all of which create a complex interaction space between the active "blue" version and the updated "green" version. State consistency during switching is influenced by several factors, including storage architecture, transaction isolation level, logging mechanisms, replication strategy, and the speed of update synchronization (Table 1).

In real engineering scenarios, state consistency during the application of the BGD model depends on how effectively changes are transferred, reproduced, and recorded between the active "blue" version and the prepared "green" version of the service. Modern microservice platforms typically combine operation logging, replication, and controlled routing-based switching to minimize state loss. For example, in e-commerce systems, synchronous replication ensures the accurate transfer of user cart data but introduces additional latency during peak loads [9, p. 15]. In contrast, asynchronous replication, used in streaming analytics services, improves performance but may lead to temporary inconsistencies during abrupt traffic switching.

In environments with high transactional intensity, such as financial applications, architects often employ two-phase synchronization. The "green" version first accumulates updates through an event log, then completes a short state alignment phase before switching. This approach reduces collision risks but requires high disk I/O throughput and stable network bandwidth. In more complex ecosystems where microservices interact with caches or message queues, techniques such as traffic shadowing are used to duplicate request streams, allowing the behavior of the new version to be tested without affecting the production database [1, p. 62].

Ultimately, practical implementation of BGD for stateful services requires balanced management of replication, transactional guarantees, and external dependencies, as well as careful adaptation of the strategy to the system's profile, workload patterns, and data criticality.

The mechanisms of data preservation and replication during switching between the "blue" and "green" environments determine the system's ability to maintain update continuity and transactional correctness under the BGD model. Ensuring consistency requires

Table 1

Structural and Functional Factors Influencing State Consistency During BGD

Factor	Description	Potential Impact on State Consistency
State storage type	Local, remote, or distributed data storage	Determines synchronization speed and the risk of divergence between versions
Logging mechanisms	Recording changes in operation logs	Ensures the ability to perform correct recovery and synchronization
Transaction isolation level	Read Committed, Repeatable Read, Serializable	Reduces the likelihood of conflicts during parallel processing
Data replication	Synchronous or asynchronous replication	Defines the temporal consistency between data copies
Request handling during switching	Requests are routed to the old, new, or mixed configuration	Affects the possibility of temporary state loss or access collisions
Dependence on external services	Caches, message queues, external databases	Complicates consistency if dependencies do not transition atomically

Source: compiled by the author based on [1, p. 62; 2, p. 56; 3, p. 910; 7, p. 2; 8, p. 100; 9, p. 15; 16].

that all changes processed by the active version be fully reproduced in the updated environment without violating the order of operations. This depends on the chosen replication strategy, logging mechanisms, and transactional guarantees (Table 2).

In modern microservice platforms, data preservation and replication mechanisms function as a coordinated, multilayered system that adapts to the workload type and the characteristics of the data. In real-time services, such as transportation monitoring systems, delta synchronization enables rapid reproduction of changes without fully copying the storage layer, which significantly reduces the preparation time of the "green" version. Financial applications employ transactional protocols with strict completion guarantees, since even minimal discrepancies between committed operations may result in double charges or "broken" transactions [11]. Analytics platforms that process large volumes of streaming data typically rely on event logs: the "green" version replays the entire sequence of changes to reach the current state without affecting the operation of the "blue" service. Synchronous replication remains essential in low-latency local clusters, whereas global distributed systems favor asynchronous approaches, compensating for replication lag with a short state alignment phase before traffic switching. In practice, the successful application of BGD depends on an appropriate combination of these mechanisms, enabling transactional integrity and stable service performance during updates.

Evaluating the performance of different BGD implementation options in stateful services is based on analyzing how switching strategies affect latency, throughput, fault tolerance, and stability of data access. Since state is transferred between two active versions, even minor differences in behavior under load can cause temporary service degradation, making the choice of BGD implementation critical for overall performance (Table 3).

The effectiveness of BGD options depends on how the selected strategy behaves under dynamic traffic conditions typical of modern distributed systems. In e-commerce services, full switching often leads to short-term increases in latency during peak sales periods, when the instantaneous load on the "green" version reaches its maximum. Gradual switching demonstrates significantly better controllability in media platforms and marketplaces: a portion of users is transferred incrementally, which allows analytics modules to capture performance changes in real time and automatically adjust the switching rate. Mirror testing is especially common in financial and telecommunications systems, where the "green" service receives duplicated request streams and is validated for load resilience without risking failures in the production environment [6, p. 4]. This practice makes it possible to detect performance issues before the

Table 2

Key Mechanisms of Data Preservation and Replication During BGD

Mechanism	Operational Description	Conditions for Ensuring Consistency
Event log	Chronological recording of every state change	Guaranteed delivery of log entries and absence of event gaps
Synchronous replication	Simultaneous data updates in the "blue" and "green" environments	Low network latency and stable throughput
Asynchronous replication	Transfer of changes with controlled delay	Acceptable lag window and alignment before switching
Derived copies	Creation of duplicate state copies to reinforce synchronization	Identical transaction order applied across all copies
Delta synchronization	Transfer of only modified data segments	Accurate detection and atomic application of deltas
Transaction protocols	Coordination of update completion through commit or rollback	Absence of concurrency conflicts and guaranteed completion

Source: compiled by the author based on [3, p. 912; 7, p. 3; 8, p. 103; 9, p. 16; 11; 17].

Table 3

Main BGD Implementation Options and Their Impact on Performance

BGD Option	Brief Description	Expected Effects
Full switching	Immediate transition to the "green" environment	Peak latency and sensitivity to load
Gradual switching	Stepwise redistribution of traffic	Stable latency and increased fault tolerance
Mirror testing	Processing duplicated request streams in the "green" environment	High predictability and operational stability
Regional switching	Updating selected segments	Minimal service degradation

Source: compiled by the author based on [6, p. 4; 7, p. 4; 8, p. 104; 10; 13, pp. 2186–2187; 15, p. 2].

actual update occurs. Regional switching is widely used in global cloud platforms, such as content delivery systems or geographically distributed databases: updates are performed stepwise across individual zones, minimizing the impact of local overloads and enabling the isolation of potential failures within a specific segment [15, p. 2].

As a result, scientific and practical assessments of BGD performance show that the optimal strategy depends on the geographic distribution of users, peak traffic dynamics, the service's state model, and the criticality of operations. Overall system performance is shaped by the interaction between switching algorithms and the actual behavior of the system under load.

The application of the BGD model in systems with complex state dependencies is accompanied by a number of scientific-technical, organizational, and operational problems that directly affect service quality. The primary technical problem lies in data migration between "blue" and "green" environments: even minimal replication lag provokes state discrepancies, while asynchronous propagation of changes can cause partial application of transactions and disruption of the logical sequence of operations. In systems with intensive updates, access conflicts arise when parallel transactions interact with different versions of state, forming "torn" or duplicate records [3, p. 910-911]. For services using aggregated data sources, the problem becomes one of coordinating dependencies between caches, message queues, and third-party repositories, which rarely support atomic transitions and require complex manual reconciliation mechanisms [8, p. 100-101]. The risk of performance degradation also increases: during transition, the new version requires intensive disk and network access to align state, which temporarily reduces the throughput of the production environment. In distributed architectures, an additional factor is network latency and unpredictable behavior of coordination protocols, which can provoke cascading commit delays [9, p. 15-16].

Organizational difficulties relate to the need to maintain two full-fledged versions of infrastructure, which increases costs, complicates coordination between teams, and requires detailed control of dependencies among microservices. Operational limitations are caused by incomplete observability of state: in practice, it is difficult to track whether the «green» version is fully synchronized, especially in large-scale ecosystems with hundreds of interconnected components.

The formulation of practical recommendations for selecting and adapting the BGD strategy for stateful microservices requires consideration of differences in their state structure, transaction characteristics, and continuity requirements. In services with high transactional intensity, particularly in financial systems, the optimal approach is a gradual switching strategy with the application of preliminary state alignment mechanisms, as it minimizes the probability of access conflicts and allows control of latent errors. For analytical and computationally intensive services, it is advisable to use mirror testing, which ensures performance predictability and allows problems to be detected before actual traffic redirection occurs. In geographically distributed systems, regional switching is effective, as it localizes potential failures within individual zones and reduces the impact on global fault tolerance.

Microservices that actively interact with caches, message queues, or external data sources require adaptation of BGD through the inclusion of additional dependency synchronization mechanisms, since their state is updated outside the service core. For such systems, it is recommended to apply two-stage "green" version preparation schemes with initial cache warm-up and establishment of stable connections, which reduces latency at the moment of switching. Enterprises for which continuity of operation is critical should focus on strategies with adaptive load balancing, which allow adjusting traffic volume depending on the actual behavior of the "green" version and reducing the risk of service degradation. The choice of BGD strategy must also consider the level of infrastructure scalability: systems with elastic resources can apply full-scale deployment of two parallel environments, while resource-constrained infrastructures require optimized schemes with partial duplication. All variants must be accompanied by extended monitoring of transactionality, latency, and state consistency, as these parameters determine the success of the transition. The application of such recommendations enables enterprises to adapt BGD to the specifics of their microservice ecosystems, optimize risks, and ensure stable service operation during updates.

Conclusions. The study established that the effectiveness of the BGD model in stateful microservices is determined by the architecture's ability to maintain continuous state consistency, correct transactional behavior, and stable performance during switching between the blue and green environments. It was demonstrated that replication and logging mechanisms, the sequence in which changes are applied, the behavior of external dependencies, and the level of transaction isolation play a decisive role, as they determine the actual quality of the transition and shape the risks of state loss or divergence.

A set of challenges limiting the practical implementation of BGD was identified, including potential state discrepancies caused by asynchronous replication, access conflicts in parallel transactions, throughput degradation during state alignment, unpredictable cache and message queue behavior, and substantial organizational overhead associated with maintaining two isolated environments. These factors were shown to have a critical impact on service quality in high-load distributed systems.

Practical recommendations for adapting BGD to service-specific requirements were formulated: gradual switching and transactional protocols for highly sensitive systems, mirrored testing for environments with elevated predictability demands, regional switching for geo-distributed platforms, and additional synchronization of dependencies for services with complex state structures. The study emphasized the need for extended monitoring of latency, consistency, and data access stability as prerequisites for the successful application of BGD.

Future research prospects include the development of formal models describing the behavior of stateful services during updates, the integration of machine learning into replication and routing mechanisms, the design of automated procedures for verifying transactional integrity, and the creation of self-learning switching strategies capable of adapting BGD to real-time workload fluctuations.

Bibliography:

1. Containerized Zero-Downtime Deployments in Full-Stack Systems. *International Journal of AI, BigData, Computational and Management Studies*. 2022. Vol. 3, no. 4. https://doi.org/10.63282/3050-9416.ijaibdcms-v3i4p107 (date of access: 16.11.2025).

2. Stencel G., Berton L. Configuring Stateful Applications. *Kubernetes Recipes*. Berkeley, CA, 2025. P. 55–73. https://doi.org/10.1007/979-8-8688-1325-2_3 (date of access: 16.11.2025).

3. Jaiswal D. Cloud-Native Transformation of SIP/IMS Core Networks: A Microservices Architecture for Next-Generation Telecommunications. *Sarcouncil Journal of Multidisciplinary*. 2025. Vol. 5, Issue 07. P. 907–918. https://doi.org/10.5281/zenodo.16419992

4. Perumal R. A. Azure Architecture Unleashed: Design, Secure, and Optimize Cloud Solutions. Radhakrishnan Arikrishna Perumal, 2024. 236 p. URL: https://surl.lu/nzotzp (date of access: 16.11.2025).

5. Tsyganok R. Methodology for Building Scalable Microservice Architectures on Go for High-Load E-Commerce Platforms. *Universal Library of Engineering Technology*. 2024. Vol. 01, no. 02. P. 42–46. https://doi.org/10.70315/uloap.ulete.2024.0102007 (date of access: 16.11.2025).

6. Beyond Monoliths: An In-Depth Analysis of Microservices Adoption in the Era of Kubernetes / S. Arya et al. *2024 1st International Conference on Advanced Computing and Emerging Technologies (ACET)*, Ghaziabad, India, 23–24 August 2024. 2024. P. 1–6. https://doi.org/10.1109/acet61898.2024.10730456 (date of access: 16.11.2025).

7. Amgothu S. Innovative CI/CD Pipeline Optimization through Canary and Blue-Green Deployment. *International Journal of Computer Applications*. 2024. Vol. 186, no. 50. P. 1–5. https://doi.org/10.5120/ijca2024924141 (date of access: 16.11.2025).

8. Zero-Downtime CI/CD Production Deployments for Insurance SaaS Using Blue/Green Deployments. *International Journal of Emerging Research in Engineering and Technology*. 2023. Vol. 4, No. 3. P. 98–106. https://doi.org/10.63282/3050-922x.ijeret-v4i3p111 (date of access: 16.11.2025).

9. Truyen E., Lagaisse B., Joosen W., Hoebreckx A., De Dycker C. Flexible Migration in Blue-Green Deployments within a Fixed Cost. WOC'20: Proceedings of the 2020 6th International Workshop on Container Technologies and Container Clouds. New York, NY, USA, 2020. P. 13–18. https://doi.org/10.1145/3429885.3429963 (date of access: 16.11.2025).

10. Antiya D. S. Harnessing AI for Data-Driven Compliance and Security in Cloud Environments. *Advances in Public Policy and Administration*. 2024. P. 245–270. https://doi.org/10.4018/979-8-3693-8069-7.ch012 (date of access: 16.11.2025).

11. Application Deployment Strategy Comparison at PT. XYZ / A. D. Ikhsan et al. *2023 International Conference on Information Management and Technology (ICIMTech)*, Malang, Indonesia, 24–25 August 2023. 2023. P. 505–510. https://doi.org/10.1109/icimtech59029.2023.10277987 (date of access: 16.11.2025).

12. Rudrabhatla C. K. Comparison of zero downtime based deployment techniques in public cloud infrastructure. *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, India, 7–9 October 2020. P. 1082–1086. https://doi.org/10.1109/i-smac49090.2020.9243605 (date of access: 16.11.2025).

13. Hybrid Software Deployment Strategy for Complex Industrial Systems / P. Rajković et al. *Electronics*. 2022. Vol. 11, no. 14. P. 2186. https://doi.org/10.3390/electronics11142186 (date of access: 16.11.2025).

14. Multi-objective optimisation framework for Blue-Green Infrastructure placement using detailed flood model / A. Ur Rehman et al. *Journal of Hydrology*. 2024. Vol. 638. P. 131571. https://doi.org/10.1016/j.jhydrol.2024.131571 (date of access: 16.11.2025).

15. Mailer D., Steindl G., Kastner W. Dynamic Deployment of Fault Detection Models. *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sinaia, Romania, 12–15 September 2023. 2023. https://doi.org/10.1109/etfa54631.2023.10275370 (date of access: 16.11.2025).

16. Simone S. D. Sam Newman: Practical Implications of Microservices in 14 Tips. *InfoQ*. URL: https://www.infoq.com/articles/microservices-practical-tips/ (date of access: 16.11.2025).

17. Fowler M. Blue Green Deployment. martinfowler.com. 2010. March 1. URL: https://martinfowler.com/bliki/BlueGreenDeployment.html (date of access: November 16, 2025).

Терлецька Х.В. ОЦІНЮВАННЯ ПРОДУКТИВНОСТІ МОДЕЛЕЙ BLUE-GREEN-ДЕПЛОЙМЕНТУ ДЛЯ СТАНОУТРИМУЮЧИХ МІКРОСЕРВІСІВ

Актуальність дослідження зумовлено зростанням складності розподілених мікросервісних архітектур та потребою забезпечення безперервної роботи сервісів під час оновлень, особливо у системах зі збереженням стану. У таких умовах модель «синє-зелене оновлення» (blue-green deploytent, BGD) стає критично важливим інструментом підтримання узгодженості даних, транзакційної цілісності та стабільної продуктивності, проте її застосування для станоутримуючих компонентів супроводжується значними технічними та організаційними обмеженнями.

Метою статті є обґрунтування методологічних засад оцінювання продуктивності та надійності BGD у станоутримуючих мікросервісах хмарних обчислень із подальшим визначенням ефективних стратегій перемикання середовищ для забезпечення безперервного функціонування сервісів.

Методи дослідження ґрунтувалися на системному аналізі архітектурних рішень, порівняльній оцінці механізмів реплікації та журналювання, моделюванні різних варіантів перемикання трафіку та інструментальному аналізі метрик затримки, пропускної здатності та відмовостійкості. Застосовано формалізовані підходи до аналізу станоутримуючих компонентів у процесі оновлення та методи відтворення поведінки реальних навантажень.

Результати дослідження полягають у тому, що встановлено критичний вплив типу реплікації, рівня ізоляції транзакцій, послідовності застосування змін та поведінки зовнішніх залежностей на консистентність стану під час роботи BGD. Доведено, що поступове перемикання та дзеркальне тестування забезпечують найбільш стабільні показники продуктивності для складних мікросервісних екосистем, тоді як повне перемикання є доцільним лише у середовищах із передбачуваними навантаженнями. Виявлено сукупність проблем, що стримують масштабованість BGD, зокрема ризики розриву стану, деградацію пропускної здатності при інтенсивній синхронізації та неузгодженість кешів і черг повідомлень.

Висновки підтверджують, що ефективність застосування BGD у станоутримуючих сервісах визначається балансом між швидкістю переходу, якістю реплікації та рівнем контролю транзакцій. Узагальнено рекомендації щодо вибору стратегії BGD залежно від профілю сервісу, вимог до надійності та структурних характеристик стану.

Перспективи подальших досліджень пов'язані з розробленням формальних моделей узгодженості при оновленнях, впровадженням алгоритмів машинного навчання для адаптивного керування реплікацією та маршрутизацією, а також створенням самонавчальних систем, здатних оптимізувати перемикання між середовищами відповідно до змін у навантаженні в реальному часі.

Ключові слова: *узгодженість даних, транзакційна цілісність, реплікація стану, затримки обробки, пропускна здатність системи, відмовостійкість сервісів, хмарні обчислення, мікросервісна архітектура, оновлення без простоїв.*

Дата надходження статті: 17.11.2025
Дата прийняття статті: 05.12.2025
Опубліковано: 30.12.2025