УДК 004.65

A.V. Myrhorodskyi, PhD student
O.V. Romaniuk, Ph.D., Associate Professor
[1,2]Vinnytsia National Technical University, Vinnytsia, Ukraine
[1]mirgorodskijav@gmail.com
[2]romaniukoksanav@gmail.com

Metric-Based Evaluation of Adaptive Consistency Benefits in Distributed Database Management Systems

The article presents a simulation-based methodology for testing and comparing data-consistency models in replicated distributed database systems. Such evaluation is important for modern cloud and edge deployments, where the selected consistency policy directly affects response time, coordination overhead, and the probability of stale reads under variable network conditions and skewed workloads. It also addresses the lack of reproducible test procedures for identifying when adaptive strategies are preferable to static consistency settings. In contrast to studies that focus primarily on the conceptual properties of strong, eventual, or adaptive consistency, the proposed paper emphasises the experimental side of the comparison: workload construction, metric selection, scenario design, and interpretation of the obtained results. While the evaluation includes strong and eventual consistency as reference regimes, the central emphasis is the practical benefit profile of adaptive consistency: under what conditions it preserves part of the correctness of strong coordination without inheriting its full performance cost. A modular discrete-event simulator implemented in Python/SimPy is used to model replicated nodes, stochastic communication delays, asynchronous propagation, and Zipfian access patterns. The evaluation protocol combines client-observed read and write latency, consistency violation rate, completed operations per run, and the average number of synchronised replicas per write, while Monte Carlo repetitions are used to reduce the influence of a single stochastic trajectory. The reported results show that adaptive consistency delivers its clearest advantage in hotspot, read-dominant, and moderate-latency conditions, where it remains much closer to the strong regime in correctness while retaining part of the throughput and write-latency advantages of eventual replication. At the same time, the high-latency scenario exposes a boundary case in which the adaptive policy degrades toward near-eventual behaviour. The obtained findings are useful both for evaluating adaptive-consistency mechanisms and for constructing future control policies.

Keywords: distributed databases, software engineering, data consistency, simulation, performance metrics, adaptive consistency

DOI: 10.31474/1996-1588-2026-1-42-137-144

Introduction

Modern information systems increasingly rely on distributed database infrastructures to achieve availability, scalability, geographic distribution, and low-latency access to data [1, 2]. Such infrastructures are used in cloud services, edge platforms, industrial monitoring, IoT data processing, and other environments in which both performance and resilience are critical [3, 4]. However, the practical deployment of distributed databases inevitably raises the question of consistency: how quickly and how reliably updates performed on one replica become visible at others, and what level of temporary divergence can be tolerated by the target application.

This issue is connected with important scientific and practical tasks. From the scientific perspective, consistency remains one of the central dimensions of distributed system design because it directly interacts with latency, availability, fault tolerance, and the observable semantics of client operations [5]. From the practical perspective, the choice of a consistency strategy affects the suitability of a distributed database for financial transactions, collaborative systems, telemetry pipelines, edge–cloud deployments, and other real-world workloads with different tolerance to stale reads and coordination overhead. For this reason, the problem is not limited to the theoretical classification of consistency models; it also includes the development of reproducible testing methods that make it possible to compare such models under controlled but representative conditions.

The study focuses on the testing methodology itself, on the system of evaluation metrics, and on the interpretation of the obtained experimental results. Such an emphasis is useful because the practical value of a comparison depends not only on the selected models, but also on how the experiment is constructed and how the resulting measurements are interpreted. In particular, this perspective makes it possible to show

more clearly where adaptive consistency delivers a practically useful balance between correctness and performance, and where its simple control rules remain insufficient.

The aim of the article is to develop and substantiate a methodology for experimental evaluation of adaptive consistency in distributed database simulation, with emphasis on testing conditions, metrics, and interpretation of the obtained results relative to strong and eventual baselines.

To achieve this aim, the following objectives are addressed.

To define a simulation environment that models replicated nodes, stochastic latency, client requests, and background propagation in a reproducible form.

To specify a system of evaluation metrics that jointly characterises performance and correctness of the compared models.

To construct a suite of scenarios that isolates the influence of load intensity, workload mix, access skew, and network regime.

To perform a comparative analysis of strong, eventual, and adaptive consistency under the selected scenarios, with special attention to the operating conditions in which the adaptive regime provides the most advantageous balance.

To identify the practical insights and limitations that follow from the measured results and can be used in further studies.

Related Work

While the theoretical foundations of consistency models are well established, practical tools for comparing these models under realistic conditions remain limited [6]. Query execution speed, latency, and other measurable characteristics depend not only on the selected consistency model, but also on the implementation details of the underlying DDBMS and the surrounding network conditions. Implementing several competing protocols directly in production systems requires substantial engineering effort, and the instrumentation needed for collecting fine-grained operational metrics can itself affect the behaviour being measured [7, 8].

Recent studies on simulation environments for distributed systems usually emphasise a particular layer of the system stack rather than cross-model consistency evaluation. Existing frameworks frequently concentrate on network protocols, storage behaviour, or a single coordination mechanism [9], [10]. At the same time, adaptive consistency is receiving growing attention because it promises to switch between stronger and weaker coordination modes depending on network delay, workload intensity, or application targets [11, 12, 13]. However, the published literature still leaves several aspects insufficiently resolved.

First, there is no universally adopted experimental methodology that would allow researchers to compare strong, eventual, and adaptive consistency within a common scenario suite and with a shared set of metrics. Second, recent publications often discuss the idea of adaptation itself, but they provide less guidance on how to evaluate whether a particular control rule behaves robustly across read-intensive, write-intensive, hotspot, or high-latency conditions [14, 15]. Third, edge-oriented deployments remain underrepresented in many cloud-centred studies, despite the fact that intermittent connectivity and resource constraints make metric-based evaluation especially important in that domain [16].

Accordingly, the general problem addressed in this article is the development of a compact but informative experimental methodology for evaluating the practical benefits and limitations of adaptive consistency in a simulated distributed database environment. The article is devoted to the unresolved question of how to organise the test scenarios, what metrics to collect, and how to interpret the measured values so that the resulting comparison is reproducible and useful for further design of adaptive consistency mechanisms.

Experimental Methodology

The proposed framework is implemented as a modular event-driven simulator in Python using SimPy, a process-based discrete-event simulation library [17]. The simulation environment represents the distributed database as a set of replicated nodes, each of which contains a local data store, version metadata, latency parameters, and a queue for asynchronous propagation tasks. The use of discrete-event simulation makes it possible to model concurrent requests, blocking operations, and background replication without implementing a production-grade database. Request arrivals follow a Poisson process with exponentially distributed inter-arrival times, while the probability of generating a read or write request is configurable to allow simulation of different scenarios. The access distribution across keys follows a Zipf's law, this choice allows the methodology to represent both moderately skewed workloads and hotspot-like request patterns that are typical of real distributed systems [18].

The comparison covers three representative operating regimes. The first is a strong-consistency implementation based on synchronous replication, where the coordinator waits for acknowledgements from reachable replicas before confirming a write to the client. In methodological terms, this mode serves as a correctness-oriented reference point: it is expected to minimise or eliminate stale reads, but also to amplify the effect of network delay on write latency.

The second regime implements eventual consistency. Writes are acknowledged after completion at the coordinator replica, while

propagation to other replicas is performed asynchronously by a background process. This reduces immediate coordination cost and therefore should improve write latency and the number of completed operations per run, but it also introduces a measurable inconsistency window in which clients may read outdated versions [19].

The third regime is an adaptive consistency strategy. In the implemented simulator, the policy uses stronger synchronisation for hot keys under low observed load, and otherwise falls back to coordinator-side completion with background propagation. The decision rule is intentionally simple so that the influence of metrics and scenario design can be studied clearly. As a result, the adaptive model is treated here not as a finished production mechanism, but as a test object. Existing adaptive strategies employ various concepts for data distribution and synchronisation in distributed systems [20]; therefore, the developed framework may also help to evaluate the effectiveness of new experimental approaches.

The methodology uses four principal metrics that jointly describe the operational efficiency and correctness of the compared models. The first metric is client-observed latency. Read and write operations are measured separately because the compared models impose fundamentally different coordination costs on them. Latency is interpreted as the elapsed time between request initiation and completion from the client perspective. This makes it sensitive both to internal processing and to network conditions, which is especially important in distributed settings [21].

The second metric is the consistency violation rate. A violation is recorded whenever a read returns a value whose version is lower than the current version at the primary reference replica. The normalised metric is computed as

$$R_{viol} = \frac{N_{viol}}{N_{read}}, \quad (1)$$

where N_{viol} is the number of stale-read events and N_{read} is the total number of completed reads. This definition makes the metric comparable across scenarios with different request counts.

The third metric is completed operations per run, which acts as an effective throughput proxy under a fixed simulation duration:

$$T_{eff} = N_{read} + N_{write}. \quad (2)$$

Because stronger consistency increases service time, it can reduce the number of operations that finish within the same simulated time horizon. The fourth metric is the average number of synchronised replicas per write:

$$S_{avg} = \frac{1}{N_{write}} \sum_{j=1}^{N_{write}} s_j, \quad (3)$$

where s_j denotes the number of replicas synchronised during write j. This indicator is useful for interpreting the internal behaviour of the adaptive strategy, because it shows whether the policy tends to operate near strong or near eventual replication.

To reduce the probability that the conclusions are artefacts of one random event ordering, the evaluation is conducted as a Monte Carlo study [22]. For every model–scenario pair, repeated runs are executed under the same parameterisation, and the results are summarised by sample means and standard deviations. This procedure improves the robustness of the comparison and makes the observed differences easier to interpret.

The experimental design includes eight scenarios. They are selected to isolate the influence of workload composition, system scale, access skew, and network regime. The workload parameters are presented in table 1. Latency is specified in abstract time units used in SimPy. The values for this parameter are selected to be close to the real ones in milliseconds. The read ratio is the previously mentioned parameter for read/write skew configuration.

Table 1 – Test scenario parameters

Scenario	Read ratio	Network latency
Baseline	0.7	5
High Load	0.5	5
Read-Intensive	0.9	5
Write-Intensive	0.3	5
Low Latency	0.7	2
High Latency	0.7	15
Hotspot	0.7	5
Uniform Access	0.7	5

The baseline scenario represents moderate operating conditions. The high-load case increases scale and total activity. The read-intensive and write-intensive scenarios isolate the influence of workload mix. The low-latency and high-latency cases preserve the main workload structure while changing the communication regime. Finally, the hotspot and uniform-access scenarios contrast strongly skewed and relatively even distributions of requests across the key space. Such a suite is compact enough for a conference article and at the same time sufficiently diverse to reveal how the same consistency policy behaves under qualitatively different conditions.

Results and Discussion

Figure 1 shows the latency results for both read and write requests. Across most scenarios, the standard deviation remains small relative to the mean, which indicates that the simulation outcomes are stable under randomised event schedules and stochastic network delay. Read latency changes much less across models than write latency, because the main coordination penalties are concentrated on write processing.

The results confirm a clear ordering and also expose the first important benefit of adaptive consistency. Eventual consistency has the lowest write latency in every scenario because it does not wait for

synchronous replication. Strong consistency always has the highest write latency, and the gap becomes especially large when network delay grows. In the high-latency scenario, write latency reaches 46.78 for strong consistency, compared with 24.98 for the adaptive model and 22.53 for eventual consistency. Thus, adaptive consistency reduces the write cost of strong coordination in every scenario, although its exact position depends on workload structure and network regime. Relative to strong consistency, its write-latency improvement is modest in hotspot-like settings and much larger in high-latency or uniform-access conditions.

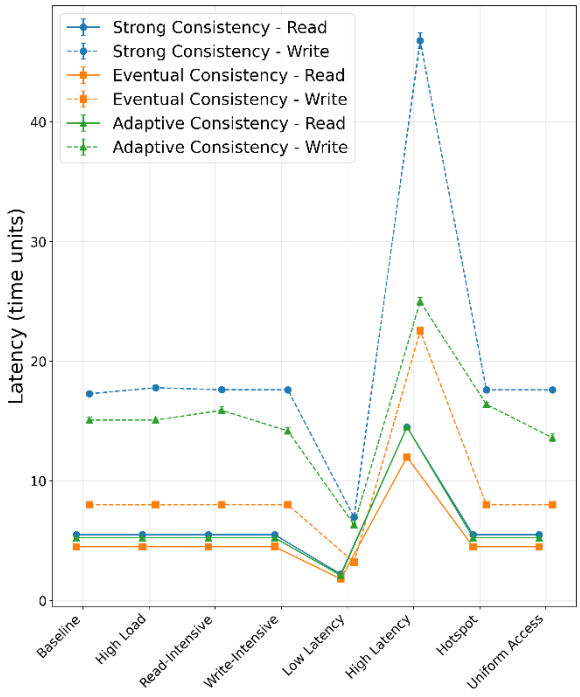

Figure 1 – Read & write latency across scenarios

Table 2 describes achieved consistency violation rates in range from 0 to 1. The strong model serves as the internal reference point and, as expected, exhibits zero recorded violations across all scenarios. Therefore, it is omitted from the table.

Table 2 – Consistency violation rates across scenarios

Scenario	Adaptive	Eventual
Baseline	0.0035	0.3133
High Load	0.0213	0.4373
Read-Intensive	0.0002	0.0425
Write-Intensive	0.0617	0.4182
Low Latency	0.0013	0.2707
High Latency	0.1979	0.2492
Hotspot	0.0019	0.4857
Uniform Access	0.0188	0.1943

From the viewpoint of adaptive consistency, these results are especially informative. Eventual consistency shows the largest staleness values, with violation rates of 0.3133 in the baseline case, 0.4373 under high load, and 0.4857 in the hotspot scenario. The adaptive model usually remains much closer to the strong regime than to the eventual one, which demonstrates its main correctness-related benefit; however, the high-latency scenario reveals a boundary case in which its violation rate rises to 0.1979.

Table 3 presents the synchronised-replica metric, which explains a substantial part of this behaviour. Strong consistency stays close to the configured replica count – minor deviations from the maximum values are due to the fact that the simulation also includes network partition events. Eventual consistency remains fixed at one synchronised replica per write. The adaptive model changes its behaviour across scenarios: in the hotspot case it synchronises about 3.8 replicas on average, which coincides with a very low violation rate of 0.0019; in the high-latency case it drops to approximately single-replica behaviour, which sharply increases staleness.

Table 3 – Synced replicas across scenarios

Scenario	Strong	Adaptive	Eventual
Baseline	2.92	2.59	1.00
High Load	4.88	4.03	1.00
Read-Intensive	3.90	3.61	1.00
Write-Intensive	3.90	2.98	1.00
Low Latency	2.92	2.76	1.00
High Latency	2.92	1.00	1.00
Hotspot	3.90	3.80	1.00
Uniform Access	3.90	2.77	1.00

The throughput results are presented in table 4. Since all models are executed under the same simulated duration, the number of completed operations directly reflects the cumulative impact of their latency and coordination overhead. Across all scenarios, eventual consistency achieves the largest number of completed operations. The adaptive model is consistently second, while strong consistency remains last because synchronous coordination increases per-operation service time.

This means that adaptive consistency preserves a measurable part of the throughput advantage of relaxed replication without collapsing to the eventual regime in every case. The gap is especially visible in write-intensive and high-load conditions, where the cost of coordination accumulates rapidly. At the same time, throughput should not be interpreted separately from the violation rate: a system that completes more operations within the same period also creates more opportunities for reads to encounter partially propagated state.

The obtained results justify several important conclusions regarding the benefits and limits of adaptive consistency. First, the selected metric system is sufficiently expressive to differentiate the models not only by performance, but by the mechanism underlying that performance.

Table 4 – Throughput (completed operations)

Scenario	Strong	Adaptive	Eventual
Baseline	498.6	543.0	763.7
High Load	823.4	939.1	1481.7
Read-Intensive	1037.0	1093.1	1367.6
Write-Intensive	535.1	641.4	1006.2
Low Latency	1081.1	1141.2	1553.6
High Latency	198.9	268.6	310.1
Hotspot	594.4	626.0	916.1
Uniform Access	594.4	684.7	916.6

Read and write latency reveal the external cost of coordination, the violation rate captures correctness degradation, throughput reflects effective service capacity, and the synchronised-replica indicator exposes the internal operating mode of the policy. Taken together, these metrics provide a fuller picture than any single indicator alone.

Second, the scenario design succeeds in isolating the major factors that influence consistency behaviour. Read-intensive workloads reduce the inconsistency window because fewer writes compete with propagation; this is why eventual consistency performs much better in the read-dominant scenario than in the write-intensive one. Hotspots are especially problematic for eventual consistency because repeated writes to the same keys outpace asynchronous propagation. In contrast, the adaptive strategy benefits from hotspot detection because the same access concentration gives it a clear signal to switch toward stronger synchronisation. Uniform access reduces this signal and therefore pushes the adaptive model toward a more relaxed operating regime.

Third, the high-latency scenario exposes a practically important boundary case. Strong consistency becomes very expensive under wide-area delay, while eventual consistency retains comparatively low write latency but accepts substantial staleness. The adaptive model responds by collapsing toward near-eventual behaviour, which preserves part of the throughput advantage but sharply weakens correctness. This observation is useful for future work on adaptive control: it indicates that a simple threshold rule is not sufficient if an application requires strict upper bounds on stale-read probability under adverse network conditions.

The proposed methodology provides a reproducible basis for comparative testing, but several limitations should be acknowledged. The simulator uses simplified failure assumptions and does not model a full protocol stack with timeouts, retries, and complex partition recovery. The staleness detector is based on comparison with an authoritative primary version, which is suitable for the implemented architecture but less directly applicable to multi-leader systems. In addition, the scale of the experiments is intentionally moderate, which helps preserve interpretability but limits direct extrapolation of numeric values to larger deployments.

At the same time, these limitations also indicate promising directions for further research. The metric set can be expanded with tail-latency indicators, staleness-magnitude distributions, and multi-key anomaly measures. The scenario suite can be enriched with trace-driven workloads, more explicit partition patterns, and heterogeneous replica capabilities. Finally, the adaptive policy itself can be improved by replacing the simple threshold rule with richer control mechanisms, including predictive or learning-based strategies. The present methodology is suitable as a foundation for such future experiments because it already establishes a common language for comparing correctness and performance outcomes.

Conclusions

The article developed and substantiated a simulation-based methodology for evaluating the benefits and limitations of adaptive consistency in distributed database systems, with emphasis on testing procedure, evaluation metrics, and interpretation of results. The proposed approach addresses an important practical and scientific task: obtaining reproducible evidence about the behaviour of adaptive consistency relative to strong and eventual baselines under changing workload and network conditions.

The conducted analysis of recent studies showed that, despite substantial theoretical work on consistency models and a growing body of adaptive-consistency research, the question of a compact and reproducible cross-model evaluation methodology remains insufficiently resolved. In response, the article proposed a discrete-event simulation environment, a set of four complementary metrics, and a structured suite of eight scenarios covering load intensity, workload composition, access skew, and network regime, specifically to reveal the operating conditions in which adaptive consistency is beneficial.

The obtained results confirm that the main practical value of adaptive consistency lies in its ability to preserve part of the correctness of strong coordination while avoiding its full performance cost. This benefit is clearest in hotspot, read-dominant, and moderate-latency scenarios, where the adaptive regime remains much closer to strong consistency in violation rate while still improving write latency and completed operations. At the same time, the experiments show that this advantage is not universal: under high network latency, the simple adaptive rule moves toward near-eventual behaviour and loses much of its correctness benefit. The experimental results also demonstrate that the selected metric system is informative enough to explain not only what outcome was obtained, but also why it was obtained.

Prospects for further research include extension of the simulator toward richer failure dynamics, expansion of the metrics suite, and comparative testing of more advanced adaptive policies. Such work can

support both the theoretical study of distributed consistency and the practical design of adaptive consistency strategies for concrete database applications.

Bibliography

1. Sekar A. The future of large-scale distributed systems: trends, challenges, and opportunities. *International journal of scientific research in computer science, engineering and information technology*. 2025. Vol. 11, no. 2. P. 3374–3390. DOI: 10.32628/CSEIT25112792.
2. CORD: parallelizing query processing across multiple computational storage devices / W. U. Zaman et al. *2025 IEEE international parallel and distributed processing symposium (IPDPS)*. 2025. P. 1141–1153. DOI: 10.1109/IPDPS64566.2025.00104.
3. Web-based efficiency of distributed systems and iot on functionality of smart city applications / R. E. A. Armya et al. *Journal of smart internet of things*. 2023. Vol. 2023, no. 2. P. 142–161. DOI: 10.2478/jsiot-2023-0017.
4. Distributed heterogeneous parallel computing framework based on component flow / J. Li et al. *Proceeding of 2021 international conference on wireless communications, networking and applications* / ed. by Z. Qian, M. Jabbar, X. Li. Singapore, 2022. P. 437–445. DOI: 10.1007/978-981-19-2456-9_45.
5. Golab W. Proving PACELC. *SIGACT news*. 2018. Vol. 49, no. 1. P. 73–81. DOI: 10.1145/3197406.3197420.
6. Kozina O., Kozin M. Simulation model of data consistency protocol for multicloud systems. *2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek)*. 2022. P. 1–4. DOI: 10.1109/khpiweek57572.2022.9916343.
7. Lowering entry barriers to developing custom simulators of distributed applications and platforms with SimGrid / H. Casanova et al. *Parallel computing*. 2025. Vol. 123. P. 103125. DOI: 10.1016/j.parco.2025.103125.
8. Investigating performance overhead of distributed tracing in microservices and serverless systems / A. Nou et al. *Companion of the 16th ACM/SPEC international conference on performance engineering*. New York, NY, USA, 2025. P. 162–166. DOI: 10.1145/3680256.3721316.
9. MimicNet: fast performance estimates for data center networks with machine learning / Q. Zhang et al. *Proceedings of the 2021 ACM SIGCOMM 2021 conference*. New York, NY, USA, 2021. P. 287–304. DOI: 10.1145/3452296.3472926.
10. DITIS: an end-to-end system-level simulator and optimizer for distributed tiered storage / E. R. Lucas Filho et al. *SN computer science*. 2025. Vol. 6, no. 6. DOI: 10.1007/s42979-025-04275-9.
11. Mahfoud Z., Nouali-Taboudjemat N. Enhancing data consistency via a context-aware dynamic adaptive model. *International journal of computing and digital systems*. 2024. Vol. 15, no. 1. P. 543–554. DOI: 10.12785/ijcds/160141.
12. Naseri Seyedi Noudoust N., Adabi S., Rezaee A. A quorum-based data consistency approach for non-relational database. *Cluster computing*. 2022. Vol. 25, no. 2. P. 1515–1540. DOI: 10.1007/s10586-021-03531-w.
13. GeoGauss: Strongly Consistent and Light-Coordinated OLTP for Geo-Replicated SQL Database / W. Zhou et al. *Proc. ACM manag. data*. 2023. Vol. 1, no. 1. DOI: 10.1145/3588916.
14. Braun S., Bieniusa A., Elberzhager F. Advanced domain-driven design for consistency in distributed data-intensive systems. *Proceedings of the 8th workshop on principles and practice of consistency for distributed data*. New York, NY, USA, 2021. DOI: 10.1145/3447865.3457969.
15. Taipalus T. Database management system performance comparisons: a systematic literature review. *Journal of systems and software*. 2024. Vol. 208. P. 111872. DOI: 10.1016/j.jss.2023.111872.
16. Mansouri Y., Babar M. A. A review of edge computing: features and resource virtualization. *Journal of parallel and distributed computing*. 2021. Vol. 150. P. 155–183. DOI: 10.1016/j.jpdc.2020.12.015.
17. Cardoso J., Lee E. A., Siron P. Simulating distributed discret event systems. DOI: 10.34849/VV2K-PS49.
18. Smith W., Byrne D., Ding C. Writeback modeling: theory and application to zipfian workloads. *Proceedings of the international symposium on memory systems*. New York, NY, USA, 2022. DOI: 10.1145/3488423.3519331.
19. Kim B.-H. VConMC: enabling consistency verification for distributed systems using implementation-level model checkers and consistency oracles. *Electronics*. 2024. Vol. 13, no. 6. DOI: 10.3390/electronics13061153.
20. Миргородський А., Романюк О., Романюк О. Аналіз динамічних моделей забезпечення узгодженості даних у розподілених системах керування базами даних. *Measuring and computing devices in technological processes*. 2025. № 3. С. 285–292. DOI: 10.31891/2219-9365-2025-83-35.
21. Raptis T. P., Cicconetti C., Passarella A. Efficient topic partitioning of Apache Kafka for high-reliability real-time data streaming applications. *Future generation computer systems*. 2024. Vol. 154. P. 173–188. DOI: 10.1016/j.future.2023.12.028.

22. Lin Z., Briggs A. Beyond the states: developing a discrete event simulation model using R. *PharmacoEconomics*. 2025. DOI: 10.1007/s40273-025-01560-6.

References

1. Sekar, A. (2025), "The future of large-scale distributed systems: trends, challenges, and opportunities", *International journal of scientific research in computer science, engineering and information technology*, Vol. 11, No. 2, pp. 3374–3390. DOI: 10.32628/CSEIT25112792.
2. Zaman, W. U. et al. (2025), "CORD: parallelizing query processing across multiple computational storage devices", *2025 IEEE international parallel and distributed processing symposium (IPDPS)*, pp. 1141–1153. DOI: 10.1109/IPDPS64566.2025.00104.
3. Armya, R. E. A. et al. (2023), "Web-based efficiency of distributed systems and iot on functionality of smart city applications", *Journal of smart internet of things*, Vol. 2023, No. 2, pp. 142–161. DOI: 10.2478/jsiot-2023-0017.
4. Li, J. et al. (2022), "Distributed heterogeneous parallel computing framework based on component flow", *Proceeding of 2021 international conference on wireless communications, networking and applications*, ed. by Z. Qian, M. Jabbar, X. Li, Singapore, pp. 437–445. DOI: 10.1007/978-981-19-2456-9_45.
5. Golab, W. (2018), "Proving PACELC", *SIGACT news*, Vol. 49, No. 1, pp. 73–81. DOI: 10.1145/3197406.3197420.
6. Kozina, O., Kozin, M. (2022), "Simulation model of data consistency protocol for multicloud systems", *2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek)*, pp. 1–4. DOI: 10.1109/khpiweek57572.2022.9916343.
7. Casanova, H. et al. (2025), "Lowering entry barriers to developing custom simulators of distributed applications and platforms with SimGrid", *Parallel computing*, Vol. 123, 103125. DOI: 10.1016/j.parco.2025.103125.
8. Nou, A. et al. (2025), "Investigating performance overhead of distributed tracing in microservices and serverless systems", *Companion of the 16th ACM/SPEC international conference on performance engineering*, New York, NY, USA, pp. 162–166. DOI: 10.1145/3680256.3721316.
9. Zhang, Q. et al. (2021), "MimicNet: fast performance estimates for data center networks with machine learning", *Proceedings of the 2021 ACM SIGCOMM 2021 conference*, New York, NY, USA, pp. 287–304. DOI: 10.1145/3452296.3472926.
10. Lucas Filho, E. R. et al. (2025), "DITIS: an end-to-end system-level simulator and optimizer for distributed tiered storage", *SN computer science*, Vol. 6, No. 6. DOI: 10.1007/s42979-025-04275-9.
11. Mahfoud, Z., Nouali-Taboudjemat, N. (2024), "Enhancing data consistency via a context-aware dynamic adaptive model", *International journal of computing and digital systems*, Vol. 15, No. 1, pp. 543–554. DOI: 10.12785/ijcds/160141.
12. Naseri Seyedi Noudoust, N., Adabi, S., Rezaee, A. (2022), "A quorum-based data consistency approach for non-relational database", *Cluster computing*, Vol. 25, No. 2, pp. 1515–1540. DOI: 10.1007/s10586-021-03531-w.
13. Zhou, W. et al. (2023), "GeoGauss: Strongly Consistent and Light-Coordinated OLTP for Geo-Replicated SQL Database", *Proc. ACM manag. data*, Vol. 1, No. 1. DOI: 10.1145/3588916.
14. Braun, S., Bieniusa, A., Elberzhager, F. (2021), "Advanced domain-driven design for consistency in distributed data-intensive systems", *Proceedings of the 8th workshop on principles and practice of consistency for distributed data*, New York, NY, USA. DOI: 10.1145/3447865.3457969.
15. Taipalus, T. (2024), "Database management system performance comparisons: a systematic literature review", *Journal of systems and software*, Vol. 208, 111872. DOI: 10.1016/j.jss.2023.111872.
16. Mansouri, Y., Babar, M. A. (2021), "A review of edge computing: features and resource virtualization", *Journal of parallel and distributed computing*, Vol. 150, pp. 155–183. DOI: 10.1016/j.jpdc.2020.12.015.
17. Cardoso, J., Lee, E. A., Siron, P., "Simulating distributed discret event systems". DOI: 10.34849/VV2K-PS49.
18. Smith, W., Byrne, D., Ding, C. (2022), "Writeback modeling: theory and application to zipfian workloads", *Proceedings of the international symposium on memory systems*, New York, NY, USA. DOI: 10.1145/3488423.3519331.
19. Kim, B.-H. (2024), "VConMC: enabling consistency verification for distributed systems using implementation-level model checkers and consistency oracles", *Electronics*, Vol. 13, No. 6. DOI: 10.3390/electronics13061153.
20. Myrgorodskyi, A., Romanyuk, O., Romanyuk, O. (2025), "Analysis of dynamic models for ensuring data consistency in distributed database management systems" ["Analiz dynamichnykh modelei zabezpechennia uzhodzhenosti danykh u rozpodilenykh systemakh keruvannia bazamy danykh"], *Measuring and computing devices in technological processes*, No. 3, pp. 285–292. DOI: 10.31891/2219-9365-2025-83-35.

21. Raptis, T. P., Cicconetti, C., Passarella, A. (2024), "Efficient topic partitioning of Apache Kafka for high-reliability real-time data streaming applications", *Future generation computer systems*, Vol. 154, pp. 173–188. DOI: 10.1016/j.future.2023.12.028.
22. Lin, Z., Briggs, A. (2025), "Beyond the states: developing a discrete event simulation model using R", *PharmacoEconomics*. DOI: 10.1007/s40273-025-01560-6.

Надійшла до редакції 01.04.2026

А.В. МИРГОРОДСЬКИЙ, О.В. РОМАНЮК
[1,2]Вінницький національний технічний університет, м. Вінниця, Україна
[1]mirgorodskijav@gmail.com, [2]romaniukoksanav@gmail.com

ОЦІНКА ПЕРЕВАГ АДАПТИВНОЇ УЗГОДЖЕНОСТІ НА ОСНОВІ МЕТРИЧНИХ ПОКАЗНИКІВ У РОЗПОДІЛЕНИХ СИСТЕМАХ КЕРУВАННЯ БАЗАМИ ДАНИХ

У статті запропоновано методологію моделювання для тестування й порівняння моделей узгодженості даних у розподілених СКБД з реплікацією. Основну увагу зосереджено на експериментальній оцінці переваг адаптивної узгодженості порівняно зі сильною та кінцевою за різних навантажень і мережевих умов. Для дослідження використано модульний дискретно-подійне симулятор на основі Python та SimPy. Оцінювання охоплює затримки читання і запису, частоту порушень узгодженості, пропускну здатність і кількість синхронізованих реплік. Результати показують, що адаптивна узгодженість є найефективнішою за помірних затримок, домінування читань і hotspot-сценаріїв.

Ключові слова: розподілені бази даних, програмна інженерія, узгодженість даних, моделювання, показники продуктивності, адаптивна узгодженість