The Indonesian Journal of Computer Science
www.ijcs.net
Volume 14, Issue 6, December 2025
https://doi.org/10.33022/ijcs.v14i6.5045

Concurrency Control in Distributed Databases: A Systematic Review

Tshidiso Lloyd Molema[1], Bukohwo Michael Esiefarienrhe[2]
lloydtshidiso@gmail.com [1], Michael.Esiefarienrhe@nwu.ac.za [2]
[1,2] Computer Science and Information Systems Department, North-West University, South Africa

Article Information

Received : 17 Nov 2025
Revised : 9 Dec 2025
Accepted : 21 Dec 2025

Keywords

Concurrency control, Distributed databases, Database scalability, Database transaction processing, Advanced DBMS techniques.

Abstract

This paper provides a comprehensive review of concurrency control techniques used in distributed database systems. It focuses on recent developments by examining articles and other academic documents published between 2016 and 2025. Using PRISMA 2020 guidelines, 197 scientific and academic studies were screened across major databases, and 10 articles met the final criteria for detailed analysis. The review classifies concurrency control approaches into four areas: types of locks, performance, accuracy, and efficiency. Each classification is then evaluated based on throughput, latency, detection accuracy, scalability, and technique applied to enhance these metrics. The findings demonstrate that traditional algorithms maintain consistent performance in general conditions but often struggle under heavy contention. Contrastingly, multi-version concurrency control and optimistic techniques improve scalability but introduce high abortion rates. Emerging adaptive techniques that depend on workload profiling show increasing promises for dynamic environments. The review highlights these trends and outlines future research direction for resilient distributed systems.

A. Introduction

In recent years, advancements in distributed database management systems (DDMS) have become essential in applications that demand high availability, fault tolerance, and scalability[1]. These systems depend on concurrency control to ensure that transitions from geographically distributed users are processed reliably and consistently[2]. Databases are designed to apply multiple techniques to efficiently process data through available computer resources [3]. Foundational theories introduced by past studies on lock, timestamp ordering, serializability, and deterministic scheduling have influenced the principles that guide current protocols. Classical approaches such as timestamp ordering, two-phase lock, and deterministic scheduling are widely adapted but remain insufficient and incapable of supporting the dynamic and heterogenous workloads characteristics of modern distributed platforms [4]. Research examining the performance of these traditional approaches underscores the underlying challenges such as lock contention, increased latency, and reduced throughput as systems scales across multiple servers. As computing workloads expanded into cloud-services, microservices, and geographically distributed architectures, new instances of concurrency control emerged. Research on optimistic concurrency control, multi version concurrency control and hybrid schemes show enhanced scalability and reduced conflict rates under high contention or increased storage overhead because of version maintenance. Studies have shown that distributed caching, memory management, and large-scale data processing further demonstrate that maintaining correctness while sustaining performance introduces complexities as data volumes increase with distributed servers across regions. This goes to show that approaches that consistently satisfy the reliability, latency, and efficiency requirements of the current distributed environment are yet to be developed[5]. The rapid evaluation of data intensive applications, along with the shift towards decentralized systems necessitates the need for improve concurrency control traditional and modern perspective. Although research has evaluated specific techniques which provide limited analysis that compare techniques across accuracy, performance, scalability, and practical relevance challenges. The correctness and efficiency of these systems remains a critical aspect of concurrency control to be addressed [6]; a condition wherein multiple transactions execute to maintain consistent data while maintaining optimum performance. The development of concurrently inclined distributed database applications seeks to immediately revolutionize data transmission across regions. Thus, concurrency control transitioned from simple locking to advanced yet flexible protocols that perform at optimum level. This includes the likes of Multi Version Concurrency Control (MVCC), deterministic scheduling, timestamp ordering, and Strong Strick Two-phase lock (SS2PL). Two-phased locking prioritize both serializability and atomicity at the cost of decreased throughput and scalability due to lock contention, making throughput not suited for larger scale microservices in diverse environment[7, 8]. Hence, the evolution of systems from centralized to web services, microservices, and geo-distributed regions indicates intensive exponential growth in the fundamental support the concept of concurrency control has in transaction management [1]. Therefore, this systematic review examines concurrency control techniques used in distributed database

systems, synthesizes results from recent studies, and evaluates their effectiveness under various conditions. The result of this review aims to assist researchers in understanding current trends, identify their strengths and limitations of existing solutions, and provides future research direction toward efficient and reliable distributed database development.

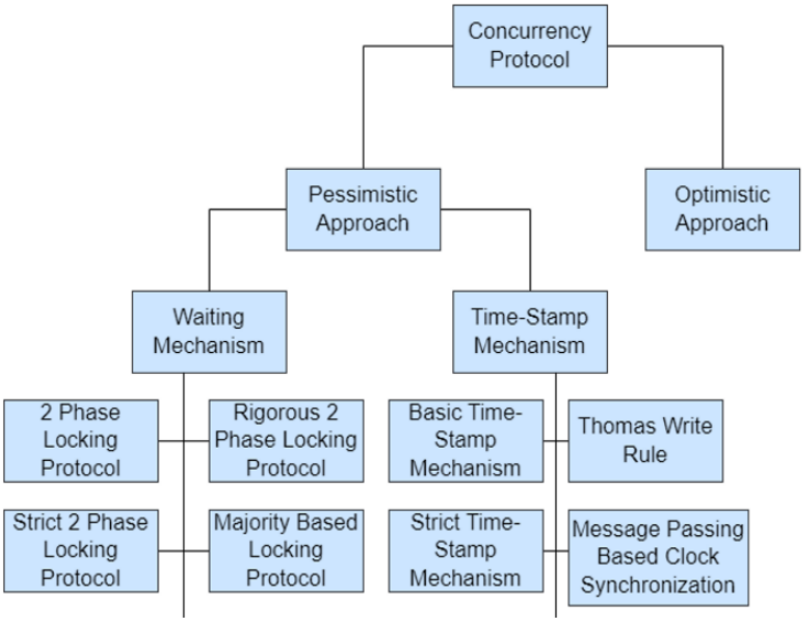

Figure 1. Types of concurrency control

Figure 1 depicts various protocols used by schedulers in response to changes in hardware and workloads. These protocols are classified into pessimistic, the existence of non-serializable behavior is likely to occur and as a results transaction delay is introduced and optimistic, which guarantees that non serializable behavior does not exist and proactively resolves in instances where they occur [9]. Evolution guarantees the balance between maintaining accuracy and increasing concurrency. The enhanced approaches that offer optimized concurrency control are essential to coordinate transactions while ensuring the trade-offs among availability, consistency, and scalability [1, 9]. The limitations on traditional methods have necessitated the new pathway of research into adaptive and multi versioned concurrency control approaches. The rapid development of large scaled applications intensified the demand for effective and efficient distributed database systems. However, this requires careful design approaches to prevent data anomalies such as lost updates and read inconsistencies which may compromise the trustworthiness of the whole system [1]. These modern approaches seek to enhance the dynamic workloads in diverse environments which aims to reduce communication overhead yet optimize resilience and efficiency of the system. Emerging technologies intend to shift towards decentralization, less locking and smart approaches that coordinate processes [7, 10]. The reason behind this shift is to ensure scalable databases and applications at the cost of communication overhead, particularly in distributed, cloud-based, and mobile applications. Thus, it is through the adaptation of these techniques that high throughput-based

applications can efficiently process concurrent transactions to prevent bottlenecks [8, 11]. One of the persistent challenges that proactively undermines the reliability of distributed database management systems is deadlocks. Despite extensive research on single and multiple cycle detection methods which are prone to significant latency, complexity, and overhead which compromise system performance, deadlocks are extremely pronounced. The study is driven by the need to critically uncover the state of concurrency control in distributed databases and the contributions of researchers in recent years to the evolving discipline distributed databases.

B. Methods

The systematical review method used is this study allows for transparent, rigorous, replicable steps in identification, critical evaluation, analyzing, and synthesizing current literature in concurrency control in distributed database management system (DBMS) [12, 13]. [12, 13]. Context specific concurrency control protocols were thoroughly reviewed by scooping of various databases following the PRISMA 2020 guidelines for reporting systematic review. This approach allows for transparent execution and documentation of keyword search strings, selection, and analysis process as illustrated by figure 3. The study seeks to provide a comprehensive understanding of concurrency control in distributed database systems and researchers' contributions to an ever-evolving discipline of concurrency control in distributed systems and environment.

The study sought articles reporting on various aspects of concurrency control such as design, development, and evaluation techniques in distributed environments, which underscore the significance of deadlock detection. The primary objective was to provide comprehensive understanding of the contextual nuances of articles addressing deadlock related research where deadlocks were discussed as a significant component. Initially, the study reviewed 197 articles retrieved from various search engines such as IEE Xplore, Springer, ACMD Digital Library, ScienceDirect, Google Scholar, Others (Scopus, arXiv, Citebase Search, CiteULike, CiteSeer, Compendex), and only 10 articles met a final set of the inclusion criteria. These search engines were utilized to source academic journals and articles that from 2016 to 2025 as shown in figure 2 through the search keywords; "Concurrency control", "Transaction processing", "distributed transactions", "Locking", "Two-phase locking", "multi-version concurrency control OR optimistic concurrency control", "Replicated database", "NOSQL". These keywords were combined with logical operators to build more complex search strings; "distributed database AND concurrency control OR concurrency control techniques in distributed systems", "database replicas OR distributed database AND two-phase locking OR timestamp ordering", "transaction management AND cloud database AND concurrency control protocols".

C. Selection process

a. Inclusion criteria

The study included the article that met inclusion criteria 1-2

1. inclusion criterion 1 – Articles written in English and for which full text is available whether open access or not.

2. Inclusion criteria 2 - Peer review conference proceedings or peer journal articles.

b. Exclusion criterion

The study excluded the articles that met any of the exclusion criteria 1 to 4 below.

1. Exclusion criterion 1– studies which were not published between 2016 – 2025.
2. Exclusion criterion 2– studies which do not present the performance evaluation results.
3. Exclusion criterion 3- studies which only the abstract was accessible, and the full texts were not available.
4. Exclusion criterion 4- Studies not written in English.

c. Quality criteria

Only articles that properly address performance metrics such as throughput, latency, accuracy, overhead etc. and handles concurrency in distributed databases were utilized in this review.

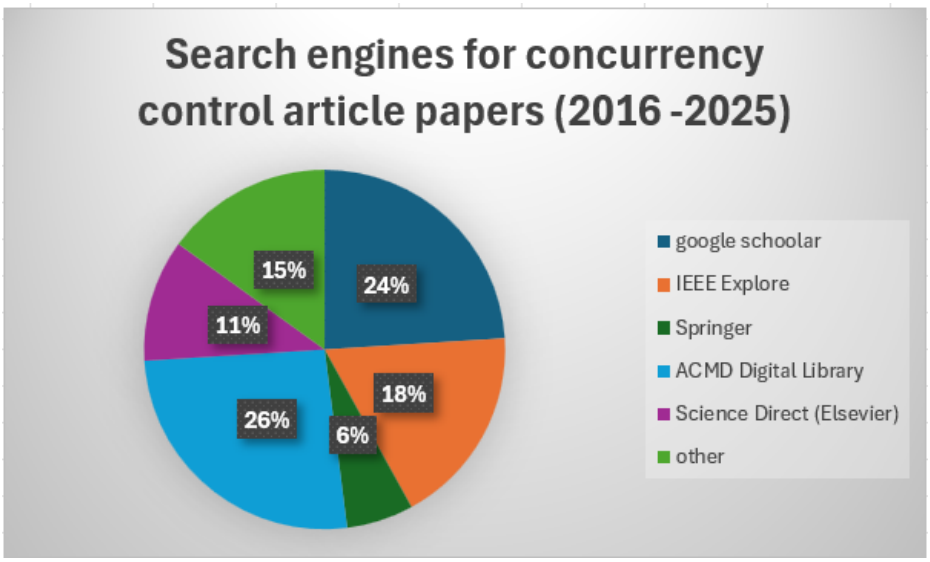

Figure 2. Articles identified per database engine

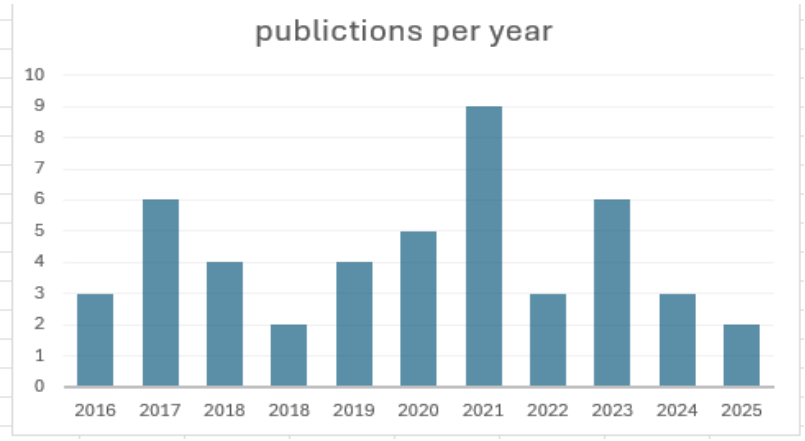

Figure 3. Publications vs year

Figure 4 illustrates the article selection procedure for this study. Initially, a total of 197 articles were identified from various databases. After removing 40 duplicates, a total of 157 articles remained for further screening. Similarly, 61 records were further excluded based on the predefined exclusion criterion discussed in section 2.1.1, criterions were equally assigned unique number of articles breakdown of articles. The study was then left with 96 articles sought for retrieval from which 71 articles were successfully assessed and deemed eligible. Twenty-five records were further excluded following the quality criteria. Finally, 10 articles met the inclusion criteria and formed part of the review, with 4 reports associated with the included articles.

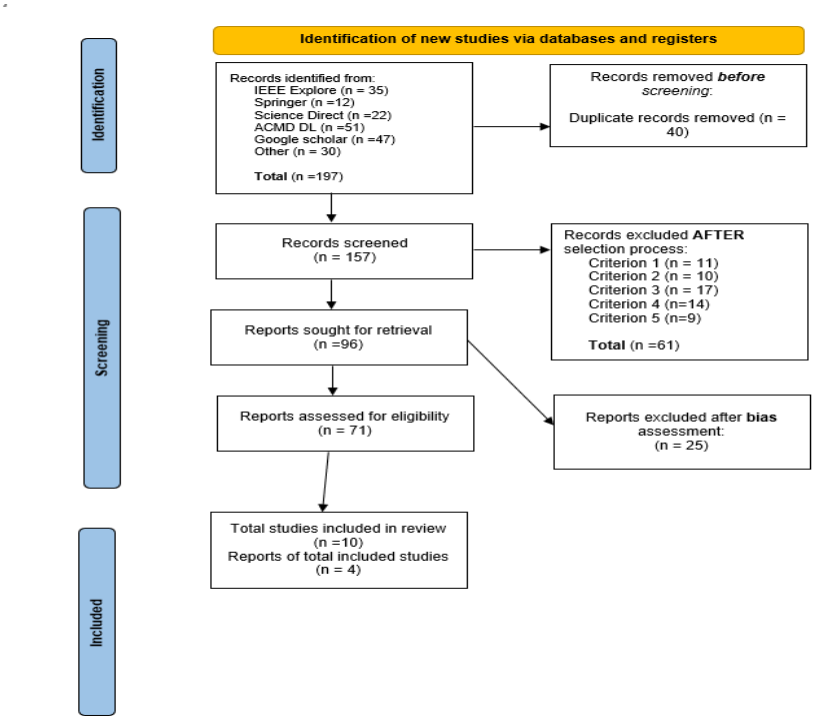

Figure 4. Article selection procedure

Table 1. Tabulation of article analysis with criteria assessed

Ref.	Concurrency control type	Algorithm	Performance analysis	Accuracy	Efficiency	Limitations
[14]	Optimistic	Adaptive and Speculative Optimistic Concurrency Control (ASOCC)	The proposed algorithm yields 35% enhanced throughput effectiveness with medium correlation.	Good	Fast processing	The algorithm is best suited for the centralized environment as opposed to larger volumes of transactions in decentralized systems at the cost increased overhead by 2PC
[15]	Pessimistic	LEAP based Online Transaction Processing (OLTP)	LEAP performance at optimal speed of 2.7 higher throughput and 7.4 lower latency with minimal abortion rates of 4%.	Excellent	Highly efficient	Prone to overhead as It diversify data across nodes. Sensitive to high contention leading to sacrificed accuracy
[16]	Lock based	ES2PL	Efficient and accurate	Improved	Efficient and accurate.	-
[17]	OCC	Read-write validate	Fast processing with accuracy rate of almost 100%	Excellent	Highly efficient.	Gradual decrease with high contention. restart overhead. Lacks real-time performance insights
[18]	optimistic	Time Traveling Optimistic Concurrency Control (TicToc)	Fast processing with enhanced throughput of 92% observed and 3.3 times reduced abortion rate.	Excellent	Highly efficient.	When serializing, there may be additional phantoms inserted introducing inconsistencies and data anomalies. Under heavy high writing contention, throughput

						decreases
[19]	optimistic	Optimistic Concurrency Control Two tracks (OCC2T)	Highly scalable, improved throughput under larger processed data at low latency	improved	Good	-
[20]	optimistic	Optimistic Concurrency Control	Efficient in low contention environments. Larger data processing introduces abortions, thus low throughput.	improved from MVCC	Efficient	Storage issues impact scalability and ultimately result in increased overhead. Transaction validations cause overhead and increased computational complexities.
[21]	Deterministic	Deterministic Concurrency Control	Increased throughput under diverse environments, resulting in gradual yet proportional scalability in those settings. DODO averages around 2.4 ms latency when processing larger data.	excellent	Highly efficient	Throughput decrease at the cost of transactions validation, particularly in high contention workloads.
[22]	-	RDMA-enabled Concurrency Control (RCC) framework	Increased throughput ranges from 32% to 67%	Excellent	Highly efficient	Struggles with heavy workload result in scalability bottlenecks.

C. Results and Discussion

Concurrency control is a popularly known concept particularly in operating systems, programming and databases with the aim of ensuring correctness when executing concurrency operations in systems. The study conducted by Wang and Qian [22, 23] explored concurrency control in RDMA environment, highlighting that lightweight lock-based approaches such as NOWAIT, WAITDIE outperforms OCC and MVCC with increased throughput where communication delays is

dominantly pronounced. Their results aimed to enhance traditional methods that view optimistic and multi version approaches as superior in distributed systems. However, that means systems should be intelligent enough to recognize and account for the distributed nature of the system's structure. Correspondingly, Wang [23] proposed a PC3 with empirical support that OCC underperformance in high contention. This validates the assertion that new approaches have sparked interest growth that seek to address optimistic limitations. Conversely, a scalable optimistic deterministic concurrency control protocol (DODO) by Wang et al. [21] presented alternative approach that having fixed commit order whilst incorporating lazy decision and early write visibility, it resolves abortion rates and enhances scalability without the foreknowledge of transactions pattern. Even though the system vulnerability of conventional deterministic techniques was revealed, assessments focused on distributed cloud transactions which revealed that they were susceptible to blocking in the event of a resource coordination failure, raising doubts about the continued use of the two-phase commit protocol. Although Wang [21] enhances scalability, the assumption of various workloads may not be suitable but rather limiting particularly on diverse and dynamic cloud-based deployments. Arika and Chaudhry [24] further extends the debate to mobile and real -time systems wherein centralized protocols which do not adhere to the temporal constraints require urgent resolution, suggesting that approaches that are response-based are of paramount importance. Thus, both contrasting ideas, emphasizes the significant of ensuring the alignment of protocols with the context specific constraints regardless of deterministic, RDMA, or mobility. Similarly, Lin et al. [15] focused on the elimination of 2PC by converting distributed transaction to local ones by utilizing aggressive data placement with the hope to potentially reduce round-trip while maintaining enhanced commits coordination. Gbaranwi and Asagba [25] reviewed the significance of synchronization in distributed systems, and argued that adequate concurrency control techniques are of paramount important to preserve the ACID properties, highlighting the need for robust techniques as a prerequisite to managing concurrent operations. Finally, Jitendra et al. [26] stated that nested transaction models that demonstrates concurrency control must adapt structurally while flat transactions are incapable with handling long-lived operations. Thus, through it nested transaction-controlled parallelism is achievable. The advancements from traditional approaches such as OOC and 2PC showed a progressive enhancement as opposed to replacements, as each approach builds on the weakness of the other while being subject to challenges. These resolutions aim to rigorously develop protocols that evolve with technological advancements in the context of distributed systems.

a. Concurrency control techniques

concurrency enhances distributed database systems from low contention into predictable, high throughput, and low latency-based execution of processes to ensure correctness. Zhang et al. [27] investigated graph-based concurrency control that focused on serializing multi region transactions which achieved a lower throughput value. To address the limitations, subsequent optimistic approaches were introduced as breakthroughs that prioritized validation than locks particularly at commit stage, underscoring a significant advancement toward more

scalable concurrency control protocols. Cellular databases have employed the use of optimistic with priority scheduling, leading to lower abortion rates, considerable resource efficiency, and timely responses [28]. Methods like DODO, which prioritized specified commit orders, improved upon state-of-the-art techniques that scaled dynamically across dispersed nodes and moved away from the dependence on prior knowledge to operate at 16.5 throughput. Further advancements on hybrid concurrency control by Nguyen and Kawashima [29] investigated decentralized techniques that advocated for fast responsiveness and produced lower latency by to 1.5x and enhanced throughput by up to 1.6x. This goes to show how decentralization reinforces concurrency control's role in guaranteeing correctness and efficiency.

Moreover, Lungu and Nyirenda [7] reviewed microservice-based architecture that incorporates concurrency control to preserve ACID properties across distributed transactions in microservices environments ensuring scalable, consistent, reliable, and efficient performance across distributed systems. Furthermore, Arika and Cheruiyot [30] surveyed protocols such as 2PC and 3PC that are atomically efficient and ensure consistent data even in conditions of node failures, making aborted processes retrievable by reinforcing reliability across distributed nodes, positioning them as fundamental concept for reliable distributed database architectures. According to Haroon [31] object oriented distributed databases experience extreme challenges and system credibility relies on robust techniques such as 2PL and MVCC to ensure serializability across concurrent transactions while maintaining shared resources against conflicting operations. This eliminated manipulation of shared resources. Moreover, Neha and Banita [32] examined potential causes of system degradation and opined that traditional approaches are directly proportional to the increment of throughput and response time in dynamic environments. This shows that these insights are intertwined on the argument that concurrency control is a mechanism that proactively safeguards data correctness under failures, maintain accuracy and reliance in object-oriented systems, enhance scalability and efficiency in diverse multi-user environments.

D. Limitations of concurrency control in distributed database systems

Despite the necessity of concurrency control, its reliance heavily on correctness is based on various techniques as the central component of the system. Concurrency control on the other hand is crucial to enabling consistent, accurate, and resilient systems to efficiently understand the concurrent execution of processes in dynamic environments. Therefore, the findings of concurrent studies demonstrated that each class of protocols has inherent limitations that limit their applicability. Atomicity is typically ensured by 2PC and 3PC. However, according to Arika and Cheruiyot [30] [30], 2PC experiences blockages which result in coordination failures and further increasing latency, causing communication overhead being inefficient compared to optimistic approaches such as ASOCC which enhanced throughput by 35% under less workloads and are suitable in centralized environments [24].

Similarly, advanced optimistic approaches exhibit significant limitations. For instance, OCC2T remains scalable with increased throughput but experienced poor performance in high contention systems leading to phantom anomalies at the cost of throughput [31]. Although MVCC enhances throughput by proactively allowing for concurrent reads and writes, Bondugula [20] uncovered data anomalies, inconsistencies, and communication overhead based on various existing versions. The tradeoff between consistency and performance in geo-distributed environments is emphasized. Notably, Ferreira et al. [33] reports that the enforcement of strong consistency in geo-distributed NoSQL databases suffers enormous performance degradation as opposed to weaker consistency level, leading to increased throughput and latency incurred, highlighting the differences between performance and strict consistency measures. Thus, high latency-based approaches such as GeoTP showed that prolonged lock contention spans can potentially result in decreased throughput [24].

The objective has been to realize a complete concurrent compatible system that performs efficiently and scales with various workload with minimal latency, throughput and communication overhead particularly in dynamic environments. The study aimed to analyze the state-of-the-art limitations that exist to proffer viable conclusion in terms of accuracy, efficient response time, scalability and minimal communication overhead [21, 28, 34]. Even though the tradeoffs between scalability, consistency, and availability have undergone intense study, the need to bridge the gap between theoretical models and practical deployment still exists. Researchers have proposed many algorithms that fail to address the requirements of real-world scenarios [4], often this includes network fluctuations, hardware failures, and operational conditions that hinder the expected performance [35]. Similarly, resource overhead remains an understudied challenge since many approaches are dependent on redundant replication strategies to optimize bandwidth and power consumption [36].

Finally, additional challenges on operational complexity, particularly in distributed concurrency control, were introduced. Systems that frequently experience concurrency bugs and data anomalies may compromise system performance due to conflicting implementation of lock management and conflict detection in dynamic environments. Moreover, debugging and maintaining distributed systems often requires expertise and resources which are underestimated [37, 38] making the human element a major constraint to efficiency of concurrency control. The emerging applicability of concurrency control in blockchain, IoT, autonomous vehicles, and edge computing continuously exposes the weaknesses of traditional approaches which lack fault tolerance abilities. The need for efficient systems and strict latency requirements have influenced many sectors to find alternative approaches as traditional approaches such as two-phase locking cannot address these limitations [5]. Consequently, these problems become operational and contextual, requiring the development of new adaptive approaches.

E. Conclusion and Future work

This review shows that concurrency control remains central to achieving reliable and consistent high performing distributed database systems. The analysis

demonstrates that traditional methods ensure strong correctness, but moder workloads increasingly are compatible with approaches such as multi-version and optimistic concurrency control due to their scalability benefits. Research revealed the persistent challenge related to latency, fault tolerance, and conflict resolution in large and decentralized environments is yet to resolved by advanced adaptive techniques. The results of this study are useful for guiding the selection and design on concurrency control approaches in emerging distributed architectures. They also provide a comparative analysis that supports researchers in identifying which approaches are best suited for specific performance and consistency requirements. Future research should primarily focus on developing adaptive approaches that scale with various workloads to minimize communication overhead while maintaining reliability and consistency under diverse conditions. Further development of advanced detection and resolution techniques, particularly for deadlocks will be essential for establishing more resilient and efficient distributed database systems.

F. Acknowledgment

I wish to acknowledge the invaluable support of my supervisor, Prof Esiefarienrhe, whose guidance, feedback, and encouragement were essential to the successful completion of this article.

G. References

[1] O. Oloruntoba, "Architecting Resilient Multi-Cloud Database Systems: Distributed Ledger Technology, Fault Tolerance, and Cross-Platform Synchronization," *International Journal of Research Publication and Reviews,* vol. 6, no. 2, pp. 2358-2376, 2025.

[2] P. A. Bernstein and N. Goodman, "Concurrency control in distributed database systems," *ACM Computing Surveys (CSUR),* vol. 13, no. 2, pp. 185-221, 1981.

[3] R. Altaher, "Transparency Levels in Distributed Database Management System DDBMS," *Preprints (April 2024). doi,* vol. 10, 2024.

[4] R. Harding, D. Van Aken, A. Pavlo, and M. Stonebraker, "An evaluation of distributed concurrency control," *Proceedings of the VLDB Endowment,* vol. 10, no. 5, pp. 553-564, 2017.

[5] H. Nejati Sharif Aldin, H. Deldari, M. H. Moattar, and M. Razavi Ghods, "Consistency models in distributed systems: A survey on definitions, disciplines, challenges and applications," *arXiv e-prints,* p. arXiv: 1902.03305, 2019.

[6] H. Gadde, "Optimizing Transactional Integrity with AI in Distributed Database Systems," *International Journal of Advanced Engineering Technologies and Innovations,* vol. 1, no. 2, pp. 621-649, 2024.

[7] S. Lungu and M. Nyirenda, "Current trends in the management of distributed transactions in micro-services architectures: A systematic literature review," *Open Journal of Applied Sciences,* vol. 14, no. 9, pp. 2519-2543, 2024.

[8] Y. Lu, X. Yu, and S. Madden, "Star: Scaling transactions through asymmetric replication," *arXiv preprint arXiv:1811.02059,* 2018.
[9] S. Kanungo and R. D. Morena, "Original Research Article Concurrency versus consistency in NoSQL databases," *Journal of Autonomous Intelligence,* vol. 7, no. 3, 2024.
[10] V. Veeramachaneni, "Edge computing: Architecture, applications, and future challenges in a decentralized era," *Recent trends in computer graphics and multimedia technology,* vol. 7, no. 1, pp. 8-23, 2025.
[11] D. Ullmann, S. Rezaeifar, O. Taran, T. Holotyak, B. Panos, and S. Voloshynovskiy, "Information bottleneck classification in extremely distributed systems," *Entropy,* vol. 22, no. 11, p. 1237, 2020.
[12] K. E. Gunnell, V. J. Belcourt, J. R. Tomasone, and L. C. Weeks, "Systematic review methods," *International Review of Sport and Exercise Psychology,* vol. 15, no. 1, pp. 5-29, 2022.
[13] L. Uttley *et al.*, "The problems with systematic reviews: a living systematic review," *Journal of Clinical Epidemiology,* vol. 156, pp. 30-41, 2023.
[14] Q. Lin, G. Chen, and M. Zhang, "On the design of adaptive and speculative concurrency control in distributed databases," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 2018: IEEE, pp. 1376-1379.
[15] Q. Lin, P. Chang, G. Chen, B. C. Ooi, K.-L. Tan, and Z. Wang, "Towards a non-2pc transaction management in distributed database systems," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 1659-1674.
[16] E. Abduljalil, F. Thabit, O. Can, P. R. Patil, and S. Thorat, "A new secure 2PL real-time concurrency control algorithm (ES2PL)," *International Journal of Intelligent Networks,* vol. 3, pp. 48-57, 2022.
[17] O. Hlaing, H. H. Htwe, and W. W. Myint, "Read-Write-Validate Approach for Optimistic Concurrency Control about Michael Kors (MK) Sale Transaction," in *2024 IEEE Conference on Computer Applications (ICCA)*, 2024: IEEE, pp. 1-5.
[18] X. Yu, A. Pavlo, D. Sanchez, and S. Devadas, "Tictoc: Time traveling optimistic concurrency control," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 1629-1642.
[19] A. Alhajri, "Performance and forensic applications of a novel optimistic concurrency control algorithm," University of Warwick, 2023.
[20] V. K. Bondugula, "Storage Optimization in Distributed Environments using Optimistic Concurrency Control," *IJSAT-International Journal on Science and Technology,* vol. 15, no. 2, 2024.
[21] X. Wang, Y. Peng, H. Huang, and X. Li, "Dodo: A scalable optimistic deterministic concurrency control protocol," *Future Generation Computer Systems,* vol. 159, pp. 15-26, 2024.
[22] C. Wang and X. Qian, "RDMA-enabled concurrency control protocols for transactions in the cloud era," *IEEE Transactions on Cloud Computing,* vol. 11, no. 1, pp. 798-810, 2021.

[23] Z. Wang *et al.*, "PC 3: Enhancing Concurrency in High-Conflict Transactions with Prior Cascading Control," in *2024 IEEE International Conference on Data Mining (ICDM)*, 2024: IEEE, pp. 881-886.

[24] Y. Cai, H. Yun, J. Wang, L. Qiao, and J. Palsberg, "Sound and efficient concurrency bug prediction," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 255-267.

[25] P. B. Gbaranwi and P. O. Asagba, "Distributed transactions and distributed concurrency control," *International Journal of Computer Science and Mobile Computing,* vol. 10, no. 1, pp. 61-68, 2021.

[26] S. Jitendra and V. Gupta, "Concurrency issues of distributed advance transaction process," *Research Journal of Recent Sciences* _____ *ISSN,* vol. 2277, p. 2502, 2012.

[27] T. Zhang, A. Tomasic, and A. Pavlo, "Intelligent Transaction Scheduling via Conflict Prediction in OLTP DBMS," *arXiv preprint arXiv:2409.01675,* 2024.

[28] P. K. Singh and U. Shanker, "Priority heuristic in mobile distributed real time database using optimistic concurrency control," in *2017 23RD Annual International Conference in Advanced Computing and Communications (ADCOM)*, 2017: IEEE, pp. 44-49.

[29] T. Nguyen and H. Kawashima, "Fairly Decentralizing a Hybrid Concurrency Control Protocol for Real-Time Database Systems," *Concurrency and Computation: Practice and Experience,* vol. 37, no. 4-5, p. e70018, 2025.

[30] R. N. Arika and W. Cheruiyot, "A survey on efficient concurrency control algorithm in distributed database systems," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology,* pp. 1176-1185, 2019.

[31] M. Haroon, "Challenges of concurrency control in object oriented distributed database systems," *Int. J. Modern Comput., Inf. Commun. Technol.,* vol. 2, no. 7, pp. 48-52, 2019.

[32] Neha and Banita, "High-Performance Concurrency Control in Multi-User Database Systems: Analysis and Optimization Strategies," in *2024 3rd Edition of IEEE Delhi Section Flagship Conference (DELCON)*, 2024: IEEE, pp. 1-5.

[33] S. Ferreira, J. Mendonça, B. Nogueira, W. Tiengo, and E. Andrade, "Benchmarking Consistency Levels of Cloud-Distributed NoSQL Databases Using YCSB," *IEEE Access,* 2025.

[34] Q. Zhuang *et al.*, "Geotp: Latency-aware geo-distributed transaction processing in database middlewares (extended version)," *arXiv preprint arXiv:2412.01213,* 2024.

[35] S. Ferreira, J. Mendonça, B. Nogueira, W. Tiengo, and E. Andrade, "Impacts of data consistency levels in cloud-based NoSQL for data-intensive applications," *Journal of Cloud Computing,* vol. 13, no. 1, p. 158, 2024.

[36] T. Acquah, R. Amankwah, and B. Appiah, "Empirical Insights into Replication Models for Distributed Database Environments," *International Journal of Computer Applications,* vol. 975, p. 8887.

[37] K. R. Thondalapally, "Event-Driven Architectures: The Foundation of Modern Distributed Systems," *IJSAT-International Journal on Science and Technology,* vol. 16, no. 1, 2025.

[38] J. Zappin, T. Stalnaker, O. Chaparro, and D. Poshyvanyk, "Challenges and Practices in Quantum Software Testing and Debugging: Insights from Practitioners," *arXiv preprint arXiv:2506.17306,* 2025.