International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Empirical Analysis of Performance Parameters for Consistency in Distributed Databases

[1]Kewal Krishan, [2]Gaurav Gupta, [3]Gurjit Singh Bhathal

Submitted: 13/09/2023 **Revised**: 21/10/2023 **Accepted**: 08/11/2023

Abstract: The contemporary era presents a significant obstacle in terms of data storage and retrieval. Horizontal scaling of data across various standard hardware configurations can yield benefits such as improved availability, enhanced performance, and increased fault tolerance. This objective can be attained through the process of replication. Nonetheless, a significant obstacle encountered in this endeavour is the lack of uniformity. This indicates that a single datum is associated with varying values across multiple locations simultaneously. Numerous theoretical frameworks have been posited in the field of literature to tackle this particular predicament. Therefore, this paper undertakes an analysis of multiple consistency models and subsequently compares them based on their respective contributing factors. This paper considers various factors such as Linearizability, Serializability, Strict Serializability, Sequential Consistency, Casual consistency, ACID and BASE in order to compare different models. Consequently, by virtue of this comparison, diverse consistency models may be chosen for the system to enhance its performance and availability.

Keyword: *Consistency, Latency, replication, Transaction*

1. Introduction

Client Server architecture is not feasible for today's requirements of computing and performance. Because, nowadays the major challenge faced is storing huge data and accessing large databases. There are two different approaches to handle this issue. One is vertical scaling, in which more memory and computing resource is added to the existing system that is going to sort the issue up to some level. But, for today's requirements, even this is not sufficient. The second is horizontal scaling where the commodity hardware is used with the help of networking. Data is stored on different systems also known as data store. Basically, data is divided into fragments (Horizontal, Vertical, and Hybrid). The Data fragments are stored on different database systems or data stores that are running in parallel. These nodes are accessed by many users concurrently. The major advantages we are getting from this distribution is availability, scalability, fault tolerant and performance. For these benefits' replication is required between data stores. Depending on nature and type of replication one of the issues faced is consistency among the data available on different systems used concurrently by different users. Consistency is one of the characteristics of the distributed system according to that every node or replica (may be data store) has the same view of data for a particular moment irrespective of who so ever has updated the data. This consistency depends on the underlying distributed systems and databases. Both communities treat this distributed word in a different manner. The distributed system treats this term as same value of an item on different data store, while databases this is C form ACID implies implementing integrity. Based on this, we are using terms *Data consistency* mainly for distributed system and *transactional consistency* for databases. A consistency model is a defined set of rules that a transaction or parts of transactions follow for consistency (accuracy or expected output). Therefore, this consistency modelling is divided into two parts one is the data consistency model and other is the transactional consistency model. Data consistency considers the consistency models that are depending on distributed systems *while* transactional consistency models treat this word for transactions from the database community that majorly depends on concurrency and isolation levels. There are Several reasons which necessitate empirical analysis of performance parameters for consistency in distributed databases:

- It is essential to optimise the efficacy of distributed databases that operate in complex and dynamic environments.
- Researchers and practitioners are able to comprehend the effect of various performance parameters on system behaviour through the use of empirical analysis.
- Comparison and Benchmarking: Empirical analysis enables comparison and benchmarking of various distributed database systems or consistency protocols.
- In conclusion, empirical analysis of performance parameters for consistency in distributed databases

[1]*Research Scholar Punjabi University Patiala, Punjab*
kakkarkewal@gmail.com
[2]*Assistant Professor Punjabi University Patiala, Punjab*
gaurav.shakti@gmail.com
[3]*Assistant Professor Punjabi University Patiala, Punjab*
gurjit.bhathal@gmail.com

is necessary for optimising performance, validating theoretical models, making informed design decisions, comparing systems, and comprehending how distributed databases act in real-world situations.

- It offers invaluable insights and quantitative data for enhancing the performance and dependability of distributed database systems.

The manuscript is structured into multiple segments. The first section will discuss the pertinent research endeavours carried out in the domain of consistency models, encompassing their classifications, consistency-related challenges, and diverse factors that influence them. The subsequent section shall furnish the outcomes and deliberations of the scrutinised literature. Additionally, a succinct bibliometric analysis shall be proffered in this section. The final section will serve as the concluding segment of the entire study.

2. Related work

Various types of models are used to address this issue of consistency in distributed databases. Based on distributed systems and transactional various consistency models are proposed. Below sections will discuss their working style and the parameters on which it depends.

2.1 Linearizability

[1]Introduced the concept of linearizability.In this type of consistency**,** one operation is done on one object at a time and is feasible only in a local system with the preferred user at a time. Here, all operations are done in a predefined order. The writes appear to be visible simultaneously. Any write or read after this is going to see the previous update. This is difficult to implement in the real world, where data is stored in a distributed fashion and network latency is the major factor. Examples of these systems are RAFT, PAXOS. It provides strong consistency, but there is an issue with performance and availability of the databases system. This type of consistency falls under data consistency that depends on the underlying distributed system. The major contribution is network latency and replication. It also depends on data allocation, fragments allocation, and data modelling [1]. Maurice Herlihyet. al. [2] introduced the concept of linearizability and provided a formal definition for it. Maged M. Michael and Michael L. Scott [3] presented a concurrent queue algorithm that satisfied the linearizability consistency model. Seth Gilbert and Nancy Lynch [4] linearizability and serializability consistency models and provided a formal definition for serializability.

2.2 Serializability:

This type of consistency is known for multi operations on multi objects by many users. It guarantees a set of transactions or parts of transactions over multiple items is equivalent to the serial order of the transactions. There is some order for the correctness of the value and order of the operations. This depends on the *concurrency* factors. In concurrency management, it depends on the isolation levels between different transactions to maintain correctness. Further, in this type of consistency, three major issues are dirty read, on repeatable and phantom read. To avoid these issues different levels of isolation are provided, which are of four types: i) Read Uncommitted, ii) Read Committed, iii) Repeatable Read, and iv) Serializable. These are implemented with the help of locks. Another type of isolation called snapshot, implemented by the technique of temporary copy of updated data unlike based on locks [5, 6]. Peter Bailiset. al. [7] discussed the trade-offs between serializability and availability in distributed data stores, and presented a new consistency model called "serializability, not serial" that relaxed the strict requirements of serializability.

2.3 Strict Serializability:

In this scenario along with the order real timestamp of the start of the execution is also considered. It is similar to linearizability but, the only difference is linearizable works for one operation on one object preferably on a local system, while serial serializability works for multiple operations on different objects on distributed systems. It assumes the presence of a global clock. It's very difficult to achieve this spectrally in distributed system in the presence of network failure. This depends on distributed systems and the concurrency of the databases [8].

2.4 Sequential Consistency:

In this type of consistency model, there is an order of operations on individual processes, not all the processes like in strict consistency models. Here interlaying is allowed. Each process maintains its order. It's a concept of single-copy data [9].

2.5 Casual consistency:

In this type of consistency, there is an order of only those operations that are related. For this to handle concurrency ordering of related operations is required. If the operations are on not dependent on each other then they can be put in any order for better performance. In this type, casually related writes are to be stored in one place to keep them in an ordered fashion [10, 11].

2.6 Eventual consistency:

Another approach is eventual consistency. Unlike in sequential or serializable the updated values must be visible to all the replicas at the same time. In this

approach, the updated value is not immediately visible to all the replicas but later on. Mostly the data is modified at one copy and other replicas get those modified values after some time. Majorly all these types of consistency come under data-centric i.e. how the stored data is updated and this updating is visible to other replicas. But what about the client? What happens to the client if the same user accesses the same data value from different instances? The data has yet to be committed. If Data is not committed by the user from the same instance the latest update is visible to that or not? This kind of issue is well taken in distributed databases under client-centric models [12,13]. Broadly it can be divided in following types:

- **Monotonic Read:** If the data item is read by the user, and any read done after this will be the same or updated values if applicable.
- **Monotonic Writes:** A write must be propagated to other replicas before a new write by the same process.
- **Read your writes:** Any read done by the process shows the latest write done by the same process.
- **Write Follows Read:** Any write on the item will show the latest read on the same item

2.7 Transaction:

Transactions are a set of operations that must be done in one unit. This is required to make ensure data integrity. In the basic model of a transaction, one user is using one database on a single machine in starting of database transactions. Once the transaction is complete the updated values are visible after this. As concurrency is allowed where many users are accessing the same data at the same place is allowed this modelling requires review. So, the ACID model is used for this type of consistency [13, 14, 15, 16]. This model is good for client-server architecture where data is stored in one place with limited concurrency levels and temporary and flat transactions. Flat transactions are for a short duration and have one single point of start and end. The next phase is nested and distributed transactions. In the nested transaction, a transaction within transaction is allowed. Here, the database is one means resource is one can say that this is the case of local transactions. Distributed transactions [17, 18] are those whose operations are spread across multiple data servers. In distributed systems [19, 20, 21], one of the servers is acting as a coordinator to manage all these issues. It depends on distributed systems in which the protocol is used. In the distributed database, the common protocol is used as a two-phase commit protocol. Even stricter the word is three phases commit is used for this [14, 22, 23].

The objective of conducting an empirical analysis on performance parameters for consistency in distributed databases is to evaluate consistency models, quantify performance metrics, understand trade-offs, optimise configuration parameters, provide insights for design decisions, validate theoretical models, and benchmark and compare various systems or consistency protocols. The primary objective is to evaluate and compare the behaviour of diverse models across various scenarios in order to determine which models provide the desired assurances while meeting performance criteria. By acquiring empirical data, researchers can gain insight into the performance of consistency models in relation to various metrics, thereby facilitating the identification of their respective strengths and weaknesses. When configuring distributed databases for optimal performance, system administrators can benefit from an understanding of trade-offs and the optimisation of configuration parameters. Empirical analysis serves to validate theoretical models and hypotheses concerning the coherence of distributed databases. This, in turn, facilitates enhanced decision-making processes in the design and implementation of distributed database systems.

3. Methodology

The empirical analysis of performance parameters for consistency in distributed databases entails a comprehensive methodology that encompasses various stages. These stages include defining research objectives, selecting appropriate consistency models, designing an experimental setup, defining performance parameters, developing test scenarios, implementing a distributed database system, collecting empirical data, conducting multiple experimental runs, systematically varying performance parameters, analysing performance data, interpreting results, validating and refining models, discussing implications and recommendations, and documenting and sharing findings. This methodology facilitates informed decision-making, system performance optimisation, and the advancement of knowledge pertaining to distributed database systems among researchers. Through the implementation of this methodology, scholars can acquire valuable knowledge regarding the performance parameters that impact the consistency of distributed databases. This knowledge can be utilised to make well-informed decisions, enhance system performance, and further the comprehension of distributed database systems. Apart from the literature study conducted on consistency models, a bibliometric analysis has also been performed. The targeted database is

Scopus wherein the used key to search the required dataset of publications was as follows:

(TITLE-ABS-KEY(("consistency model") AND ("distributed database")))

The dataset collected in filtered specifically for a period of 1994 to 2022. A total of 63 publications have came into limelight out of which 22 are open access, 1 Gold, 2 Hybrid Gold, 6 Bronze and 20 from Green subscription category of Scopus.

4. Results and Discussions

This study has identified the parameters on which consistency either data or transaction depends which are discussed as follows. Also, Table 1 presents the dependency of consistency type on these factors.

a. **Network latency:** It is the delay due to the communication network. If the delay is small and tolerable without affecting consistency, then this type of latency is known as low-level latency. Otherwise it's known as high level latency. High level affects the bandwidth of network communication that delays the communication between data servers or nodes. So higher level of consistency like linearizability [24, 25], strict serializability, etc. is not achievable in high latency network-based systems. Various approaches like SPANNER [16, 26, 27] and PLANET [28-32] are utilized to minimize the latency.

b. **Concurrency:** Concurrency in database can be accessed by many transactions may be from one server or from different servers simultaneously. This is done with the help of locks and isolation levels. For high level of concurrency accordingly isolation has to be chosen. There are levels like serializability in various systems [33-38] and snapshot levels. For stricter type of concurrency, we need more isolation so concurrency level will be Low.

c. **Replication**: The best advantage of distribution is data redundancy for availability and scalability. For this replication [21, 24, 27] plays an important role. But this becomes one of the challenges for data inconsistency. It uses fully replicated databases or partially replicated databases or other types. If a strict or similar type of consistency is required then immediate replication is applied. In the consistency level for stricter kind synchronous kind of replication required that hampers the availability of the databases.

d. **Transaction Nature:** Different types of transactions are a flat or nested type of transaction. Flat means that it may be categorized based on the execution location i.e. local transaction or global transaction [13, 15, 25, 37]. The stricter kind of transaction prefers the flat and local type of transaction for better results, and the casual kind of consistency can handle the nested and global, which is distributed kind of transaction.

e. **Level of redundancy:** Data redundancy means the same data in different locations. This is achieved with the help of replication. In distributed systems [38-40] data redundancy is required for availability and fault tolerance. So, level means the required redundancy is desired instead of fully relation of the data. Here, how redundancy plays an important role is consistency [41, 42] selection?

f. **System Design:** Designing distributed system means which the framework we are going to use. What kind of scalability is required horizontal or vertical? Similarly, in decision-making of the architecture of a system, shading is required or not. Shading [41] means a part of the database is at one particular location. It also decides where the query is going to execute preferably where minimum traffic is required to move from one location to another location.

Table 1: Dependency of consistency type on various factors

Consistency Factors/Types of Consistency	**Network Latency**	**Level of concurrency**	**Replication Type**	**Transaction Type**	**Redundancy level**	**System Design**
Linearizability	Low	Low	High	High	High	High
Serializability	Low	Low	High	Moderate	Low	Moderate
Strict Serializability	Low	Low	High	High	Moderate	High

Sequential Consistency	Moderate	Moderate	Moderate	Moderate	Low	Moderate
Casual consistency	Moderate	Moderate	Moderate	Low	Low	Low
ACID	High	Moderate	Moderate	Moderate	Moderate	High
BASE	Moderate	High	Low	Low	Low	Low

To conclude, the above table revealed that for network latency dependency type, lower level of dependency exist for Linearizability[43], Serializability [44], Strict Serializability and ACID [45,46] properties of transactions in the databases whereas moderate level of dependency exists for Sequential and Casual consistency along with BASE [46]. For concurrency, only BASE has the highest level of dependency whereas Linearizability, Serializability and Strict Serializability [47] will have low depenedency. Sequential and Casual consistency along with ACID are moderate in tendency to depend on concurrency. Replication is a dependency where majority of the consistency types shown high level of dependencies which are namely, Linearizability, Serializability, Strict Serializability and ACID. BASE has the lower level of dependency with moderate level of dependency for Sequential and Casual consistency [48]. In case of Transaction type as a parameter of dependency, only casual consistency and BASE has lower dependency whereas ACID, Sequential Consistency [49] [55] and Serializability have moderate dependency, Strict Serializability and Linearizability have shown as significantly dependent on Transaction type. Redundancy level for Linearizability is high, low in case of BASE, Serializability, sequential consistency and casual consistency, moderate for strict Serializability and ACID. System Design dependency is low for casual consistency and BASE but high for ACID, Linearizability and Strict Serializability, moderate for Serializability and Sequential consistency [50] [56]. The below figure 1 is representing the table findings in detail wherein to represent the ordinal data as low, moderate and high levels of dependency on the factors of consistency models, 1,2 and 3 are the numeric values used for indicated the categorical data of low, moderate and high respectively.

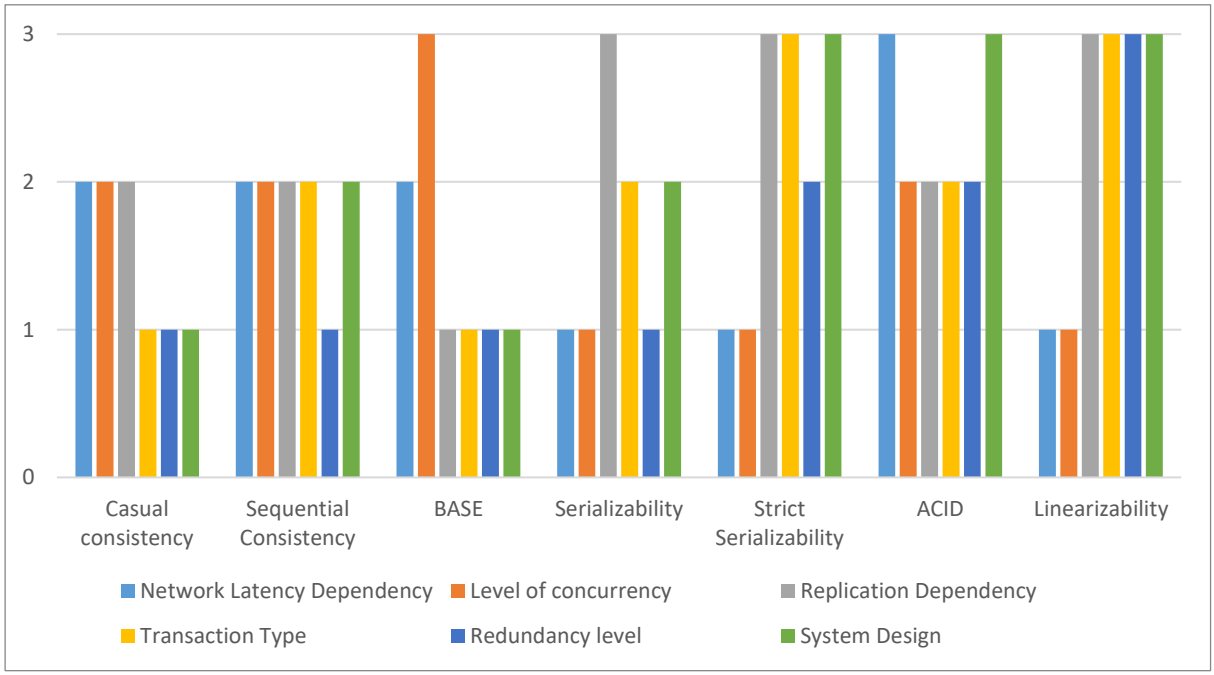

Fig 1: Consistencies with the dependencies on various parameters

The issue of consistency of distributed databases is a topic which is explored well in the United States followed by China, France and Germany. United States contributed a total of 17 whereas 10 from China, 7 each from Frame and Germany are there. Japan, Portugal, Sweden and United Kingdom are having the equal contribution of 3 publications among the

publications extracted from Scopus as a dataset to analyse. The Figure 2 depicts the top 10 countries of affiliations under the selected list of publications for analysis.

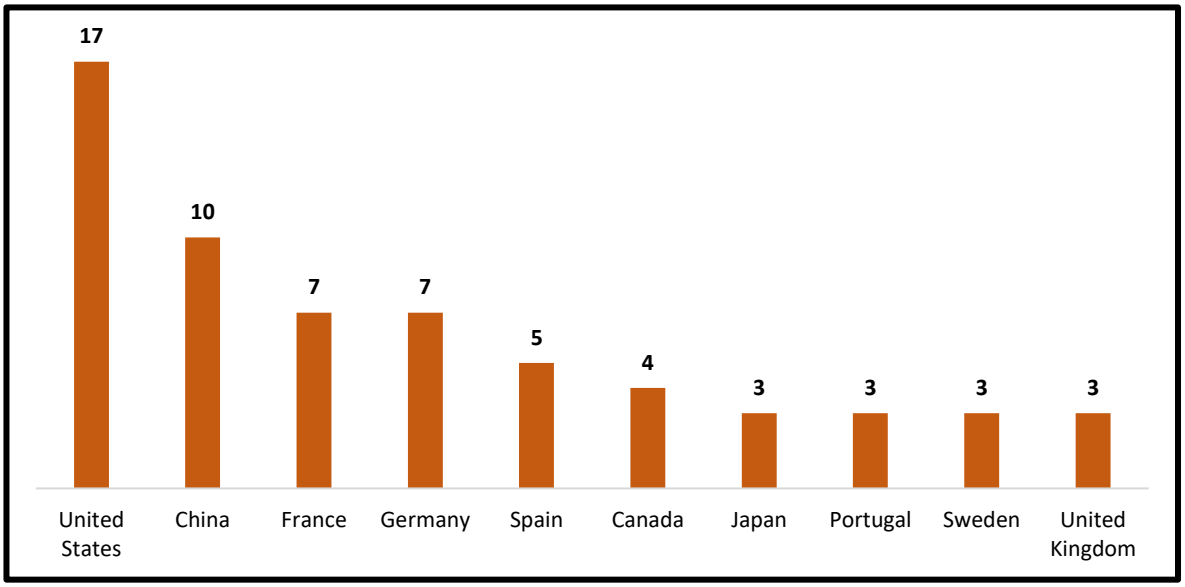

Fig 2: Publication count in Top 10 Countries for consistency issues in Distributed databases

The publications have revealed the need of exploring the field more rigorously. Looking at the sources for relevance of the topic, it has been seen that "Lecture Notes in Computer Science" followed by "Leibniz International Proceedings in Informatics Lipics" are having the highest count of publications on the topic. In Figure 3 below, it is evident that a total of 8 among the selected dataset are from the "Lecture Notes in Computer Science" and 4 are from the "Leibniz International Proceedings in Informatics Lipics".

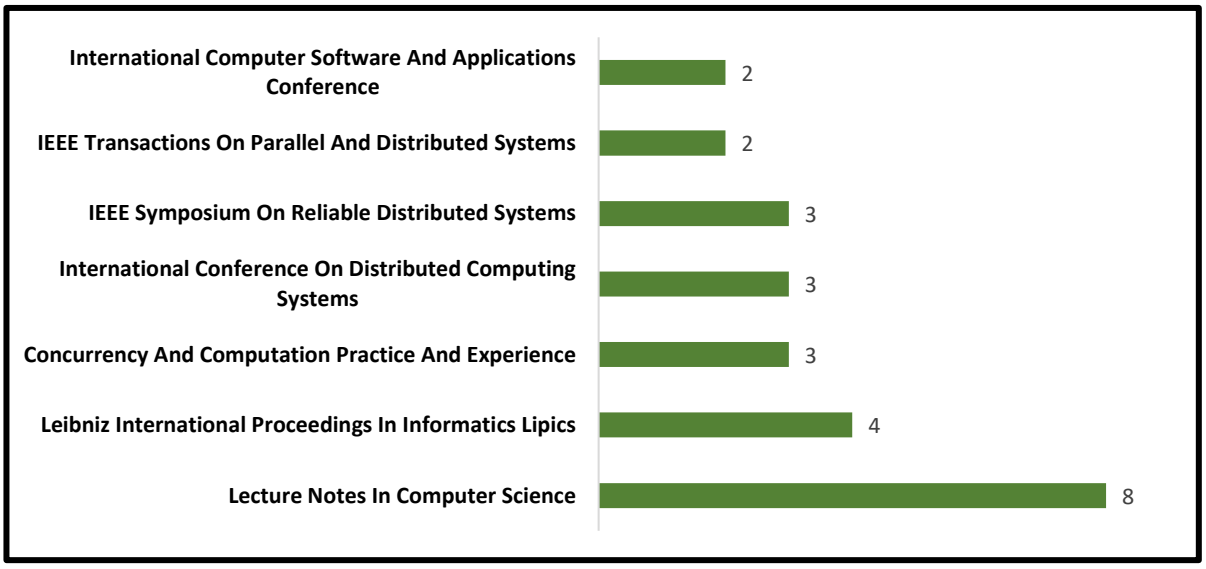

Fig 3: Publications as per source for the taken dataset

The analysis has been drawn for the number of publications based on category of work. The analysis has restricted the type of work to any of the 3 types i.e. Conference publication, Article or a review. It can be seamlessly viable from the figure 4 that the majority of work published is of type conference paper. Around 41 research papers got published in prominent conferences and 21 have been published as articles and remaining 1 paper is a review paper. This analysis clearly indicates the level of research going on for the distributed databases and issues related to consistency models.

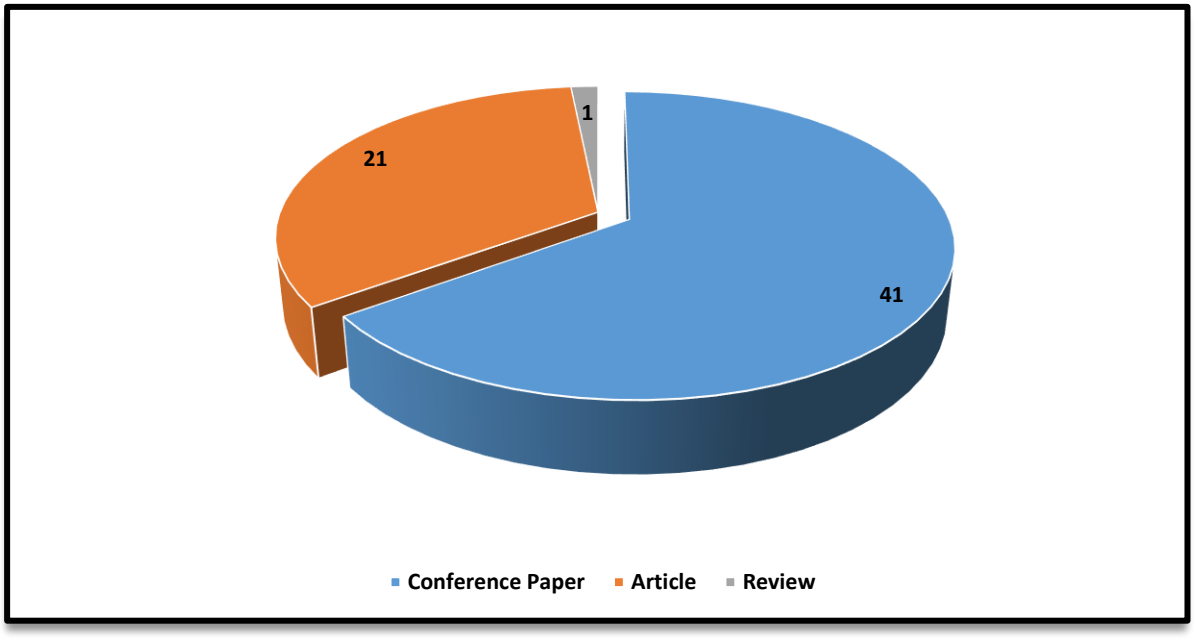

Fig 4: Type of Study conducted on consistency issues in distributed databases

The next research question is to focus on the area of selected topic. The Scopus dataset taken using the keywords consistency model and distributed databases brought the Computer science field to the limelight as around 62 publications are from Computer science field which got published comparing to the 24 publications from decision sciences. The Figure 5 below presents the broader aspect of fields contributing to the topic and its trend in research.

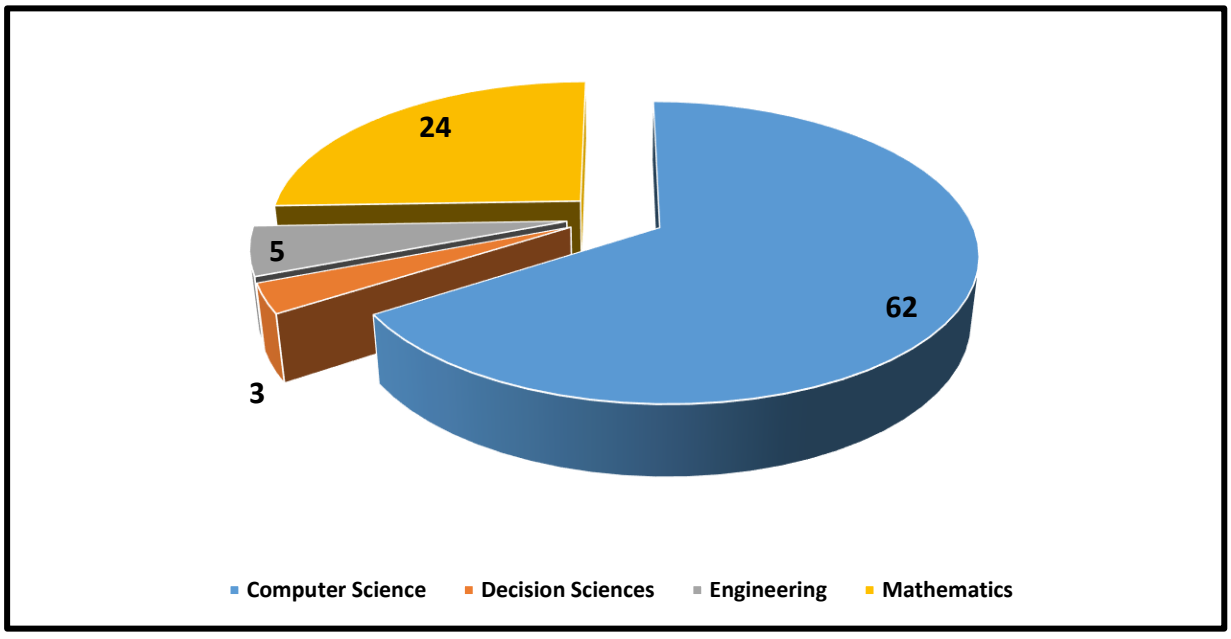

Fig 5: Fields Contributing the Consistency Models and Their Dependencies in Distributed Databases

It has been evident from the Figure 6 below that the concept was being taken up in the year 1994 as 1 research work got published then. Slowly the distributed databases and their issues became prominent to the researchers to focus in the year 2015 wherein around 7 publications hit the research world but slowly the trend started declining. The analysis revealed the need of the work yet to be done in the field of distributed databases under the computer science domain to resolve the issues of consistency models.

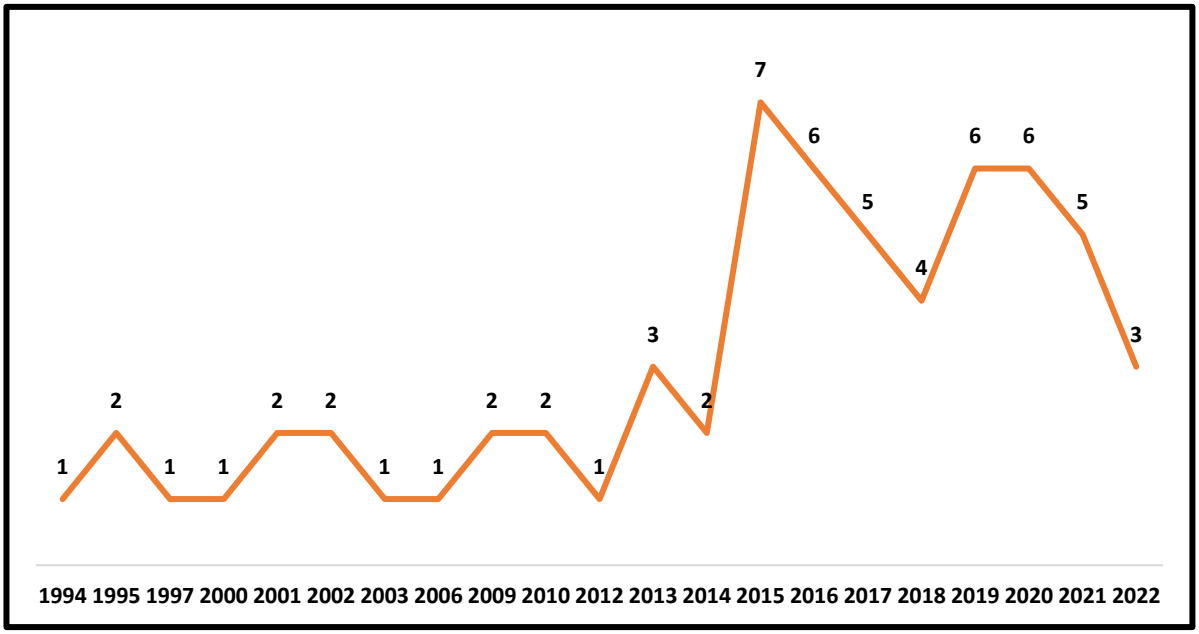

Fig 6: Number of publications Since 1994

5. Conclusion

This study analyses multiple consistency models and then performs a theoretical comparison of said models on the basis of significant parameters pertinent to ensuring consistency in distributed databases. Consequently, the system can choose from a variety of consistency models in order to improve its performance and availability. The ACID consistency model places a greater emphasis on network consistency as its principal parameter, while linearizability, which can only be achieved in smaller systems, is of lesser importance. In a diminutive system, such as linearizability, the aforementioned parameter is of lesser importance, whereas it has a greater impact in the context of the ACID or BASE model. Concurrency refers to the implementation of multiple transactions simultaneously. According to the model description, the frequency of linearizability transactions is comparatively low due to the system's small size. The system significantly relies on both the ACID and BASE models, which allow for the concurrent execution of multiple transactions. Replication is the process by which any temporal lag and the extent of changes, whether comprehensive or partial, are propagated to all nodes. Linearizability is a consistency model that requires immediate visibility of modifications across nodes and is therefore highly dependent on replication. However, the delay of replication may differ based on the type of replication employed by the ACID or BASE model. In terms of transaction classification, the linearizability model is best adapted for simple transactions, whereas the ACID and BASE models can accommodate both simple and nested transaction types. These transactions may be distributed across multiple nodes. The term "redundancy" refers to the condition of duplication. In the context of linearizability, the degree of linearity is significantly increased because identical data must be provided to the adjacent system. Both the ACID and BASE variants are deemed manageable. The focus of this investigation is the design parameter of the system. The level of dependence varies considerably between consistency models. In particular, linearizability demonstrates an exceptionally high level of dependence, whereas casual, ACID, and BASE models typically exhibit moderate to low levels of dependence. Given this conclusion, it is difficult to choose linearizability in the current era of Big Data, in which data is dispersed across multiple geographic locations. A significant amount of duplication and redundancy is required at an elevated level. A meticulous design of the system is required. The Base consistency paradigm is preferable when distributing databases across a geographically dispersed network-based system. The following consideration to be made is concurrency. Due to its low reliance on replication and transactional characteristics, the BASE model may be chosen in this regard. Causal consistency and BASE consistency are the best options for dealing with large amounts of data and scenarios where replication may experience delays. This degree of consistency enables greater degrees of concurrency. In instances where replication latency is present, the use of ACID consistency is preferable because it provides a consistency guarantee. In situations where BASE and Casual consistency are utilised, however, availability is prioritised, albeit without a guarantee of consistency.

References

[1] Wang, Y., Wang, Z., Chai, Y. and Wang, X., 2023. Rethink the Linearizability Constraints of Raft for

Distributed Systems. *IEEE Transactions on Knowledge and Data Engineering*.

[2] Herlihy, M.P. and Wing, J.M., 1990. Linearizability: A correctness condition for concurrent objects. ACM Transactions on Programming Languages and Systems (TOPLAS), 12(3), pp.463-492.

[3] Michael, M.M. and Scott, M.L., 1996, May. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing (pp. 267-275).

[4] Gilbert, S., Lynch, N., Shvartsman, A. and II, R., 2003. Implementing atomic memory in dynamic networks, using an aggressive reconfiguration strategy. Tech. Rep., LCS, MIT.

[5] Zhao, Z., 2021, June. Efficiently supporting adaptive multi-level serializability models in distributed database systems. In Proceedings of the 2021 International Conference on Management of Data (pp. 2908-2910).

[6] Helt, J., Burke, M., Levy, A. and Lloyd, W., 2021, October. Regular Sequential Serializability and Regular Sequential Consistency. In Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (pp. 163-179).

[7] Patterson, S., Elmore, A.J., Nawab, F., Agrawal, D. and Abbadi, A.E., 2012. Serializability, not serial: Concurrency control and availability in multi-datacenterdatastores. arXiv preprint arXiv:1208.0270.

[8] Zhao, Z., 2021, June. Efficiently supporting adaptive multi-level serializability models in distributed database systems. In Proceedings of the 2021 International Conference on Management of Data (pp. 2908-2910).

[9] Diogo, M., Cabral, B. and Bernardino, J., 2019. Consistency models of NoSQL databases. Future Internet, 11(2), p.43.

[10] Wiling, B., 2022. Scientific Study of CAP Theorem and Understanding its Different Implementation Methods. Mathematical Statistician and Engineering Applications, 71(1), pp.133-137.

[11] Tyulenev, M., Schwerin, A., Kamsky, A., Tan, R., Cabral, A. and Mulrow, J., 2019, June. Implementation of cluster-wide logical clock and causal consistency in mongodb. In Proceedings of the 2019 International Conference on Management of Data (pp. 636-650).

[12] El-Hindi, M., Heyden, M., Binnig, C., Ramamurthy, R., Arasu, A. and Kossmann, D., 2019, June. Blockchaindb-towards a shared database on blockchains. In Proceedings of the 2019 International Conference on Management of Data (pp. 1905-1908).

[13] Sadoghi, M. and Blanas, S., 2019. Transaction processing on modern hardware. Morgan & Claypool Publishers.

[14] Abadi, D., 2012. Consistency tradeoffs in modern distributed database system design: CAP is only part of the story. Computer, 45(2), 37-42.

[15] Thomson, A., Diamond, T., Weng, S. C., Ren, K., Shao, P., & Abadi, D. J., 2012. Calvin: fast distributed transactions for partitioned database systems. In Proceedings of the 2012 ACM SIGMOD international conference on management of data (pp. 1-12).

[16] Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., ... & Woodford, D., 2013. Spanner: Google's globally distributed database. ACM Transactions on Computer Systems (TOCS), 31(3), 1-22.

[17] Pang, G., Kraska, T., Franklin, M. J., & Fekete, A., 2014. Planet: making progress with commit processing in unpredictable environments. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data (pp. 3-14).

[18] Chandra, D. G., 2015. BASE analysis of NoSQL database. Future Generation Computer Systems, 52, 13-21.

[19] Pankowski, T., 2015. Consistency and availability of Data in replicated NoSQL databases. In 2015 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE) (pp. 102-109). IEEE.

[20] Briquemont, I., Bravo, M., Li, Z., & Van Roy, P., 2015. Conflict-free partially replicated data types. In 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom) (pp. 282-289). IEEE.

[21] Guo, J., Chu, J., Cai, P., Zhou, M., & Zhou, A., 2017. Low-overhead paxos replication. Data Science and Engineering, 2, 169-177.

[22] Hsu, T. Y., Kshemkalyani, A. D., & Shen, M., 2018. Causal consistency algorithms for partially replicated and fully replicated systems. Future Generation Computer Systems, 86, 1118-1133.

[23] Harding, R., Van Aken, D., Pavlo, A., & Stonebraker, M., 2017. An evaluation of distributed concurrency control. Proceedings of the VLDB Endowment, 10(5), 553-564.

[24] Sun, H., Xiao, B., Wang, X., & Liu, X., 2017. Adaptive trade-off between consistency and performance in data replication. Software: Practice and Experience, 47(6), 891-906.

[25] Zakhary, F. N. V. A. V., & El Abbadi, D. A. A., 2017. A System Infrastructure for Strongly Consistent Transactions on Globally-Replicated Data.

[26] Shapiro, M., & Sutra, P., 2018. Database consistency models. arXiv preprint arXiv:1804.00914.

[27] Katembo K. E., Shri K., Ruchi A., 2018. Analysis of Database Replication Protocols. International Journal of Latest Trends in Engineering and Technology Special Issue ICRMR-2018, pp. 075-083

[28] Aldin, H. N. S., Deldari, H., Moattar, M. H., & Ghods, M. R., 2019. Consistency models in distributed systems: A survey on definitions, disciplines, challenges and applications. arXiv preprint arXiv:1902.03305.

[29] Keleher, P. J., 1996. The relative importance of concurrent writers and weak consistency models. In Proceedings of 16th International Conference on Distributed Computing Systems (pp. 91-98). IEEE.

[30] Adve, S. V., & Gharachorloo, K. (1996). Shared memory consistency models: A tutorial. computer, 29(12), 66-76.

[31] Hoellrigl, T., Dinger, J., & Hartenstein, H. (2010, July). A consistency model for identity information in distributed systems. In 2010 IEEE 34th Annual Computer Software and Applications Conference (pp. 252-261). IEEE.

[32] Birman, K. P. (1994). Integrating runtime consistency models for distributed computing. Journal of Parallel and Distributed Computing, 23(2), 158-176.

[33] Kaur, V., & Singh, A. (2015). An Encryption Scheme Based on AES and SHA-512. International Journal of Applied Engineering Research, 10(10), 25207-25218.

[34] Pokharel, B., Ganesh, K., Timilsina, B., Pokharel, Y., Makam, M., & Kaur, V. (2022, September). An Interactive AI-Powered Web Healthcare System. In 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA) (pp. 1-5). IEEE.

[35] Kaur, V., Gupta, K., Baggan, V., & Kaur, G. (2019). Role of cryptographic algorithms in mobile ad hoc network security: an elucidation.

[36] Mapanga, I., & Kadebu, P. (2013). Database management systems: A nosql analysis. International Journal of Modern Communication Technologies & Research (IJMCTR), 1, 12-18.

[37] Gupta, A., Tyagi, S., Panwar, N., Sachdeva, S., & Saxena, U. (2017, October). NoSQL databases: Critical analysis and comparison. In 2017 International conference on computing and communication technologies for smart nation (IC3TSN) (pp. 293-299). IEEE.

[38] Radoev, M. (2017). A comparison between characteristics of NoSQL databases and traditional databases. Computer Science and Information Technology, 5(5), 149-153.

[39] Davidson, S. B., Garcia-Molina, H., & Skeen, D. (1985). Consistency in a partitioned network: a survey. ACM Computing Surveys (CSUR), 17(3), 341-370.

[40] Biswas, R., & Enea, C. (2019). On the complexity of checking transactional consistency. Proceedings of the ACM on Programming Languages, 3(OOPSLA), 1-28.

[41] Bouajjani, A., Enea, C., Guerraoui, R., & Hamza, J. (2017, January). On verifying causal consistency. In Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (pp. 626-638).

[42] van der Linde, A., Fouto, P., Leitão, J., & Preguiça, N. (2020, April). The intrinsic cost of causal consistency. In Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data (pp. 1-6).

[43] Hsu, T. Y., & Kshemkalyani, A. D. (2016, May). Performance of causal consistency algorithms for partially replicated systems. In 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 525-534). IEEE.

[44] Huang, X., Gao, J., Wang, L., & Yang, R. (2007, August). Examplar-based shape from shading. In Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007) (pp. 349-356). IEEE.

[45] Bichsel, M., & Pentland, A. P. (1992, January). A simple algorithm for shape from shading. In Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (pp. 459-460). IEEE Computer Society.

[46] Herlihy, M. P., & Wing, J. M. (1990). Linearizability: A correctness condition for concurrent objects. ACM Transactions on Programming Languages and Systems (TOPLAS), 12(3), 463-492.

[47] Brutschy, L., Dimitrov, D., Müller, P., & Vechev, M. (2018, June). Static serializability analysis for causal consistency. In Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (pp. 90-104).

[48] Bailis, P., & Ghodsi, A. (2013). Eventual consistency today: Limitations, extensions, and beyond. Communications of the ACM, 56(5), 55-63.

[49] Banothu, N., Bhukya, S., & Sharma, K. V. (2016, March). Big-data: Acid versus base for database transactions. In 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (pp. 3704-3709). IEEE.

[50] Sethi, R. (1981, October). A model of concurrent database transactions. In 22nd Annual Symposium on Foundations of Computer Science (sfcs 1981) (pp. 175-184). IEEE.

[51] Dalessandro, L., Scott, M. L., & Spear, M. F. (2010). Transactions as the foundation of a memory consistency model. In Distributed Computing: 24th International Symposium, DISC 2010, Cambridge,

MA, USA, September 13-15, 2010. Proceedings 24 (pp. 20-34). Springer Berlin Heidelberg.

[52] Helt, J., Burke, M., Levy, A., & Lloyd, W. (2021, October). Regular Sequential Serializability and Regular Sequential Consistency. In Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (pp. 163-179).

[53] Mizuno, M., Raynal, M., & Zhou, J. Z. (1995). Sequential consistency in distributed systems. In Theory and Practice in Distributed Systems: International Workshop Dagstuhl Castle, Germany, September 5–9, 1994 Selected Papers (pp. 224-241). Springer Berlin Heidelberg.

[54] Bernardi, G., & Gotsman, A. (2016). Robustness against consistency models with atomic visibility. In 27th International Conference on Concurrency Theory (CONCUR 2016). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[55] de Oliveira, V.F., Pessoa, M.A.D.O., Junqueira, F. and Miyagi, P.E., 2022. SQL and NoSQL Databases in the Context of Industry 4.0. Machines, 10(1), p.20.

[56] Valavi, R., Guillera-Arroita, G., Lahoz-Monfort, J.J. and Elith, J., 2022. Predictive performance of presence-only species distribution models: a benchmark study with reproducible code. Ecological Monographs, 92(1), p.e01486.

[57] Mr. Vaishali Sarangpure. (2014). CUP and DISC OPTIC Segmentation Using Optimized Superpixel Classification for Glaucoma Screening. International Journal of New Practices in Management and Engineering, 3(03), 07 - 11. Retrieved from http://ijnpme.org/index.php/IJNPME/article/view/30

[58] Chaudhary, D. S. . (2021). ECG Signal Analysis for Myocardial Disease Prediction by Classification with Feature Extraction Machine Learning Architectures. Research Journal of Computer Systems and Engineering, 2(1), 06:10. Retrieved from https://technicaljournals.org/RJCSE/index.php/journal/article/view/12