Faculty of Natural and Applied Sciences Journal of Scientific Innovations
Print ISSN: 2814-0877 e-ISSN: 2814-0923
www.fnasjournals.com
Volume 5; Issue 1; December 2023; Page No. 84-90.

DISTRIBUTED STORAGE SYSTEMS AND HOW THEY HANDLE DATA CONSISTENCY AND RELIABILITY

Aikins, M.V.
OLA College of Education, Department of Mathematics and Computer Studies, Cape Coast, Ghana

***Corresponding author email:** mark.aikins@stu.ucc.edu.gh

Abstract
This paper examines the difficulties and solutions associated with data consistency and reliability in distributed storage systems. Distributed storage systems are necessary for managing data in distributed computing environments but ensuring data consistency and reliability can be challenging. The paper examined various consistency models, such as eventual consistency, strong consistency, and conflict-free replicated data types (CRDTs), as well as techniques for maintaining data reliability, such as replication, erasure coding, and versioning, in a comprehensive literature review. The trend analysis reveals significant advancements in the field, with distributed storage systems continuously improving their ability to balance consistency, latency, and availability tradeoffs. The discussion section of the paper examined the implications of these findings for the design and implementation of distributed storage systems and identifies areas for future research. Examining the impact of emerging technologies such as non-volatile memory on the design of distributed storage systems. The findings provide researchers, practitioners, and system designers working with distributed storage systems and related technologies with valuable insights to advance knowledge in this vital field and develop innovative solutions to meet the rising demands of distributed computing environments.

Keywords: Distributed Systems, Distributed Storage, Data Consistency, Data Reliability

Introduction

In the realm of modern applications generating an exponential growth of digital data, the reliance on distributed storage systems has surged. These systems, offering enhanced scalability, fault tolerance, and availability compared to traditional centralized storage systems (Ghemawat et al., 2003), face persistent challenges in ensuring data consistency and reliability across distributed environments, especially in the presence of network failures, node failures, or data corruption (Vogels, 2009). The critical aspect of data consistency is pivotal for maintaining a coherent view of stored data among all nodes within the system, even when faced with concurrent updates or node failures (Terry et al., 1994). Key mechanisms for achieving data consistency and reliability involve the strategic use of replication and data redundancy, enabling data distribution across multiple nodes to ensure fault tolerance and recovery. Extensive research in recent years has centred on the challenge of maintaining data consistency and reliability in distributed storage systems. Various algorithms and techniques are proposed to address these challenges, as evidenced by the works of Lakshman and Malik (2010) and Decandia et al. (2007). However, a deeper understanding of the intricate interplay between consistency, reliability, and other system aspects such as performance, scalability, and availability is imperative.

The study aims to address significant research gaps in the existing literature and contribute to a deeper understanding of critical aspects in this domain. Existing literature has touched on mechanisms employed to maintain data consistency and reliability in distributed storage systems, but there is a need for a more in-depth exploration, as highlighted by Ghemawat et al. (2003) and Vogels (2009). Vogels (2009) introduced the concept of 'eventual consistency which further emphasizes the fact that eventual consistency is not some esoteric property of extremely distributed systems. Eventual consistency is an extensively utilized model, ensuring that in the absence of new updates to the shared state, all nodes will ultimately converge to the same data (Bailis & Ghodsi, 2013; Burckhardt, 2014; Terry et al., 1994; Vogels, 2009). Numerous contemporary relational database management systems (RDBMSs) that ensure primary backup reliability incorporate replication techniques in both synchronous and asynchronous modes. In synchronous mode, the replica update is an integral part of the transaction, while in asynchronous mode, updates reach the backup with a delay, often facilitated through log shipping. In the latter scenario, if the primary fails before the logs are shipped, reading from the promoted backup may yield outdated and inconsistent values. Additionally, to

Cite this article as:
Aikins, M.V. (2023). Distributed storage systems and how they handle data consistency and reliability. *FNAS Journal of Scientific Innovations*, *5*(1), 84-90.

enhance scalable read performance, RDBMSs have begun offering the option to read from the backup. This represents a classic implementation of providing eventual consistency guarantees, wherein the inconsistency windows are contingent on the periodicity of the log shipping. The dynamic nature of technology necessitates continuous monitoring of advancements. Staying updated with the latest developments, building upon insights from Ghemawat et al. (2003) and Vogels (2009). While prior research frequently concentrated on isolated components, this study endeavours to offer a comprehensive perspective, synthesizing insights from the works of Terry et al. (1994) and drawing on the findings of Vogels (2009).

The issue of data consistency in distributed storage systems has been extensively studied, and several approaches to maintaining a consistent view of stored data across multiple nodes have been proposed. Consensus algorithms, such as Paxos (Lamport, 1998) and Raft (Ongaro & Ousterhout, 2014), ensure that system nodes agree on the values of the stored data despite network and node failures. These algorithms are commonly used in distributed databases and other storage systems and are designed to provide strong consistency guarantees. Using conflict resolution techniques, as seen in Amazon's Dynamo (Decandia et al., 2007) and Apache Cassandra, is an additional method for ensuring data consistency (Lakshman & Malik, 2010). These systems employ an "eventually consistent" model in which updates are propagated asynchronously and conflicts are resolved using techniques like vector clocks, version vectors, and timestamps. In exchange for improved performance and availability, this strategy sacrifices some consistency guarantees, making it suitable for large-scale, geographically distributed storage systems. In addition, numerous techniques for detecting and recovering from data corruption in distributed storage systems have been studied by researchers. Reed-Solomon erasure coding (Reed & Solomon, 1960) and Tornado codes (Byers et al., 1998) are examples of error-correcting codes used to recover corrupted data in the presence of node failures or data corruption. In addition, systems like Google's File System (GFS) (Ghemawat et al., 2003) and Hadoop Distributed File System (HDFS) (Shvachko et al., 2010) use checksumming and data replication to detect and recover from data corruption.

In distributed storage systems, replication and data redundancy are fundamental mechanisms for achieving data consistency and reliability. Distributed storage systems can tolerate node failures and guarantee data accessibility even in the presence of network partitions by replicating data across multiple nodes. Various strategies, such as primary-backup replication (Alsberg & Day, 1976), chain replication (van Renesse & Schneider, 2004), and quorum-based replication, can be used to implement data replication (Gifford, 1979). These replication strategies have varying trade-offs regarding consistency, availability, and performance, making them suitable for a variety of application requirements and system constraints. On the other hand, data redundancy can be achieved via techniques such as erasure coding and data stripping (Plank, 1997). Erasure coding divides data into smaller chunks and encodes them with redundant chunks, enabling the recovery of the original data even if some chunks are lost or corrupted. This method has been implemented in several distributed storage systems, including Facebook's f4 system and Microsoft's Azure Storage (Calder et al., 2011).

Challenges of Data Consistency and Reliability in Distributed Storage Systems

The widespread adoption of distributed storage systems has been spurred by the proliferation of data-intensive applications such as cloud computing, social media, and the Internet of Things (IoT) (DSS). These systems store and manage data across multiple nodes or servers, providing greater scalability, availability, and fault tolerance than centralized systems (Bhattacharjee et al., 2017). Due to factors such as network latency, node failures, and concurrent data access, maintaining data consistency and reliability in distributed storage systems has proven to be a formidable challenge. Concerning data consistency and dependability in distributed storage systems, researchers have identified a variety of obstacles and proposed many solutions. Dealing with the trade-offs between consistency, availability, and partition tolerance, as described by the CAP theorem, is one of the primary challenges of maintaining data consistency in distributed storage systems (Brewer, 2000). This theorem asserts that a distributed system can only simultaneously guarantee two of the three properties. Therefore, designers must carefully select which consistency models to implement based on the specific requirements of their applications (Terry, 2013). Some applications may prioritize strong consistency to ensure that all nodes have the most recent data, whereas others may prioritize eventual consistency to maintain high availability and fault tolerance at the expense of temporary data inconsistencies (Vogels, 2009). Distributed storage systems must be able to recover from node failures and data corruption while mitigating the effect on overall system performance. Data replication, erasure coding, and versioning are some of the techniques proposed to improve data reliability in distributed storage systems (Ghemawat et al., 2003). Data replication entails storing multiple copies of the same data across multiple nodes, thereby enhancing fault tolerance and availability but also necessitating additional storage resources and posing potential consistency issues (Hendricks et al., 2007). Erasure

Cite this article as:
Aikins, M.V. (2023). Distributed storage systems and how they handle data consistency and reliability. *FNAS Journal of Scientific Innovations*, *5*(1), 84-90.

coding, on the other hand, involves dividing data into fragments and adding redundant pieces to allow for the reconstruction of the original data in the event of node failures. Erasure coding provides better storage efficiency than replication, but a higher computational overhead (Xin et al., 2013). Finally, versioning techniques enable the detection and recovery of corrupted data in a distributed storage system by tracking changes to data over time (Lakshman & Malik, 2010).

Different Approaches to Addressing Data Consistency and Reliability in Distributed Storage Systems

In recent years, distributed storage systems have attracted considerable interest due to their capacity to provide scalable, reliable, and highly available storage. Nonetheless, ensuring data consistency and dependability in such systems remains difficult. This review of the relevant literature discusses various approaches to addressing these challenges.

Replication and Quorum-based Approaches: One common approach to ensuring data consistency and reliability in distributed storage systems is through replication (Lakshman & Malik, 2010). Replication involves creating and maintaining multiple copies of data across different nodes in the system. This redundancy allows the system to tolerate failures and maintain availability. Quorum-based approaches, such as the Paxos algorithm (Lamport, 1998), extend this idea by requiring most nodes (a quorum) to agree on a value before it is considered committed. This method helps ensure consistency by requiring that any two quorums have at least one node in common, reducing the likelihood of conflicting updates.

Consistency Models and Protocols: Different consistency models have been proposed to guarantee the order and visibility of updates in distributed storage systems in different ways. Strong consistency models, such as linearizability (Herlihy & Wing, 1990) and serializability (Bernstein et al., 1987), offer strict guarantees but frequently incur substantial latency and overhead. Weaker consistency models, such as eventual consistency (Vogels, 2009) and causal consistency (Lloyd et al., 2011), sacrifice some consistency guarantees in exchange for enhanced performance and availability. Consistency protocols are frequently used by distributed storage systems to enforce consistency models. Two-phase commit (Gray, 1978) and three-phase commit (Skeen, 1981) are examples of protocols used in distributed transactions to ensure atomicity and isolation. Protocols developed more recently, such as chain replication (Van Renesse & Schneider, 2004) and conflict-free replicated data types (CRDTs) (Shapiro et al., 2011), prioritize consistency while minimizing coordination overhead.

Erasure Coding and Fault Tolerance: Erasure coding (Huang et al., 2012) is an additional technique for ensuring the integrity of data in distributed storage systems. Erasure coding divides data into fragments and encodes them using a redundant encoding scheme rather than duplicating entire data items. This enables the system to tolerate a certain number of failures with less storage overhead than replication. Fault-tolerant distributed storage systems frequently employ various fault tolerance techniques, such as Byzantine fault tolerance (Castro & Liskov, 2002), to guarantee data consistency and dependability in the presence of failures or malicious nodes. To achieve their objectives, these techniques frequently employ consensus algorithms, cryptographic methods, and redundancy.

Trend Analysis of Distributed Storage Systems

A trend analysis of distributed storage systems and how they handle data consistency and reliability reveals several significant directions and advancements in the field. These trends reflect the increasing significance of scalable and fault-tolerant storage systems in a world that is increasingly data-driven, as well as the need to accommodate diverse application requirements and resource constraints. The paper discusses the following trends.

Stronger consistency models: As distributed storage systems continue to evolve, there is an increasing emphasis on delivering more consistent models without sacrificing performance or availability. For example, research on consensus algorithms such as Paxos (Lamport, 1998) and Raft (Ongaro & Ousterhout, 2014) has led to more efficient and comprehensible implementations that can be applied to a broader range of applications, such as distributed databases, blockchains, and coordination services.

Geo-replication and edge computing: With the proliferation of edge computing and Internet of Things (IoT) devices, there is a growing demand for distributed storage systems that can handle data consistency and reliability across geographically dispersed nodes. Geo-replication techniques such as Multi-Paxos (Lamport, 2005), Chain Reaction,

and CRDTs (Shapiro et al., 2011) can provide high levels of consistency, fault tolerance, and data locality for applications that span multiple data centres or edge locations.

Adaptive consistency models: The development of adaptive consistency models that can dynamically adjust the level of consistency provided based on application requirements and system conditions is an additional emerging trend in distributed storage systems. Techniques such as consistency rationing (Kraska et al., 2009), RedBlue consistency (Li et al., 2012), and hybrid consistency models (Terry et al., 1994) permit the fine-tuning of consistency, availability, and performance trade-offs, thereby enabling systems to better adapt to changing workloads, network conditions, and resource constraints.

Self-healing and autonomous systems: As the size and complexity of distributed storage systems continue to increase, there is a growing demand for self-healing and autonomous mechanisms that can automatically detect and recover from failures, data corruption, and other problems. Recent research has focused on machine learning and artificial intelligence techniques for automating the management of distributed storage systems, including failure prediction (Li et al., 2006), anomaly detection (Mace et al., 2018), and automated tuning (Mace et al., 2018). (Arulraj & Pavlo, 2017).

Integration with modern data processing frameworks: The increasing popularity of big data processing frameworks such as Apache Spark (Zaharia et al., 2016) and Apache Flink (Carbone et al., 2015) has increased interest in developing distributed storage systems that can integrate seamlessly with these platforms and provide consistent and dependable storage for large-scale data processing jobs. Alluxio and Delta Lake are two examples of distributed storage systems designed specifically for big data workloads (Armbrust et al., 2020). These trends underscore the need for further research and development in this area, as well as the integration of new technologies and approaches to address the ever-growing challenges of managing data in large-scale, distributed systems

This paper addresses two critical research questions:

1. How do distributed storage systems uphold data consistency across multiple nodes in the face of network failures, node failures, or data corruption?
2. What roles do replication and data redundancy play in achieving data consistency and reliability in distributed storage systems?

The significance of this study lies in its exploration of the mechanisms employed by distributed storage systems to maintain data consistency and reliability amidst challenges such as network and node failures. Additionally, it delves into the pivotal roles of replication and data redundancy in attaining these objectives. This exploration holds the potential to contribute substantially to the ongoing development of more robust, efficient, and reliable distributed storage systems—crucial components supporting modern data-intensive applications and services. By investigating the challenges and best practices associated with ensuring data consistency and reliability, this research aims to provide valuable insights and recommendations for both academia and industry. Improved understanding of the mechanisms and techniques used in maintaining data consistency and reliability not only enhances the design and implementation of current storage systems but also lays the groundwork for future advancements in this critical field.

Methodology

This research paper's methodology is comprised of a systematic literature review, a comparative analysis of existing techniques and approaches, and an evaluation of their effectiveness in addressing the challenges of data consistency and reliability. The researcher employs PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) guidelines to ensure a rigorous and transparent review process, a comprehensive literature search was conducted in major databases to identify relevant articles, conference papers, and technical reports published over the years. The identified literature was analyzed to extract information about the various consistency models and techniques used in distributed storage systems to ensure data reliability and consistency. The strengths and weaknesses of these approaches, as well as the trade-offs and complexities involved in implementing them in distributed storage systems, were examined through a comparative analysis. Based on the comparative analysis, the efficacy of the various consistency models and data reliability techniques in addressing the challenges of data consistency and reliability in distributed storage systems was assessed. The evaluation considered scalability and performance in real-world scenarios, as well as the ability to handle varying levels of consistency, latency, and availability requirements. The results of the systematic literature review, comparative analysis, and evaluation of effectiveness were synthesized to

Cite this article as:
Aikins, M.V. (2023). Distributed storage systems and how they handle data consistency and reliability. *FNAS Journal of Scientific Innovations, 5*(1), 84-90.

answer the research questions and the implications of the findings for the design and implementation of distributed storage systems.

Discussion

The first research question concerned the mechanisms and strategies utilized by distributed storage systems to maintain data consistency across multiple replicas. Distributed storage systems use a variety of consistency maintenance strategies, including eventual consistency, strong consistency, and conflict-free replicated data types (CRDT) (Shapiro et al., 2011; Terry et al., 1994; Yu & Vahdat, 2000). The disadvantages of each of these strategies include latency, availability, and system complexity. The selection of a specific consistency model is determined by the requirements of the application and the underlying system architecture. Eventual consistency is commonly used in distributed storage systems due to its low latency and high availability, but it can lead to temporary inconsistencies between replicas (Terry et al., 1994). Strong consistency models, such as Paxos and Raft, ensure that all replicas always have the same state, but at the cost of increased latency and decreased availability (Lamport, 1998; Lamport, 2005). CRDTs offer a middle ground by offering a collection of data structures that can be updated concurrently without conflicts and eventually converge to a consistent state (Shapiro et al., 2011).

The second research question centred on how distributed storage systems guarantee data integrity in the face of data loss and failure. The literature demonstrates that distributed storage systems employ techniques such as replication, erasure coding, and versioning to guarantee data integrity (Li et al., 2006; Lamport, 2005). Replication is a common method for ensuring data durability, with systems creating multiple copies of data across multiple nodes (Armbrust et al., 2020; Li et al., 2014). Erasure coding is another data-reliability-providing technique that divides data into fragments and encodes them so that the original data can be reconstructed even if some fragments are lost (Mace et al., 2018). Versioning helps maintain data integrity by tracking changes to data objects over time, facilitating data recovery in the event of data loss or corruption (Zaharia et al., 2016). The literature review reveals that distributed storage systems have made substantial strides in addressing data consistency and reliability issues. Nevertheless, further research is necessary in several areas. Future research could, for instance, investigate novel strategies for balancing the tradeoffs between consistency, latency, and availability. In addition, research could be conducted to develop new data structures or algorithms that improve the efficiency of replication and erasure coding, especially in large-scale distributed systems. Finally, it would be beneficial to examine how emerging technologies, such as non-volatile memory, may affect the design of distributed storage systems and their capacity to guarantee data consistency and reliability (Arulraj et al., 2017; Li et al., 2012).

Conclusion

This paper has examined the challenges and strategies associated with data consistency and dependability in distributed storage systems. The paper also identified, through a comprehensive literature review, the key mechanisms and approaches used by these systems to maintain data consistency and ensure data reliability. The study further highlighted the tradeoffs and complexities involved in designing and implementing distributed storage systems that can manage data in a distributed environment efficiently. In review of the relevant literature, the paper examined various consistency models, such as eventual consistency, strong consistency, and conflict-free replicated data types (CRDTs), as well as techniques for ensuring data integrity, such as replication, erasure coding, and versioning. The trend analysis demonstrates that significant progress has been made in addressing these challenges, with distributed storage systems continuously evolving to achieve a better balance between consistency, latency, and availability. In the discussion section, the implications of these findings for the design and implementation of distributed storage systems were examined in depth. The paper further identified areas for future research, such as the exploration of novel approaches for balancing the tradeoffs between consistency, latency, and availability, the development of new data structures or algorithms to improve the efficiency of replication and erasure coding, and the investigation of how emerging technologies, such as non-volatile memory, may impact the design of distributed storage systems. The results offer useful insights for researchers, practitioners, and system designers dealing with distributed storage systems and related technologies. By further investigating the areas as identified for future study, it will enhance understanding of this important field and create new solutions to address the growing needs of distributed computing environments.

Cite this article as:
Aikins, M.V. (2023). Distributed storage systems and how they handle data consistency and reliability. *FNAS Journal of Scientific Innovations*, *5*(1), 84-90.

References

Alsberg, P. A., & Day, J. D. (1976, October). A principle for resilient sharing of distributed resources. In *Proceedings of the 2nd international conference on Software engineering* (pp. 562-570).

Armbrust, M., Das, T., Sun, L., Yavuz, B., Zhu, S., Murthy, M., ... & Zaharia, M. (2020). Delta lake: high-performance ACID table storage over cloud object stores. *Proceedings of the VLDB Endowment*, *13*(12), 3411-3424. https://people.eecs.berkeley.edu/~matei/papers/2020/vldb_delta_lake.pdf

Arulraj, J., & Pavlo, A. (2017, May). How to build a non-volatile memory database management system. In *Proceedings of the 2017 ACM International Conference on Management of Data* (pp. 1753-1758).

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

Bernstein, P. A., Hadzilacos, V., & Goodman, N. (1987). *Concurrency control and recovery in database systems* (Vol. 370). Reading: Addison-wesley.

Bhattacharjee, S., Deka, D., & Kalita, J. K. (2017). A survey on distributed storage systems. International Journal of Cloud Computing, 6(2), 115-129.

Brewer, E. A. (2000, July). Towards robust distributed systems. In *PODC* (Vol. 7, No. 10.1145, pp. 343477-343502).

Burckhardt, S. (2014). Principles of eventual consistency. *Foundations and Trends® in Programming Languages*, *1*(1-2), 1-150.

Byers, J. W., Luby, M., Mitzenmacher, M., & Rege, A. (1998). A digital fountain approach to reliable distribution of bulk data. *ACM SIGCOMM Computer Communication Review*, *28*(4), 56-67.

Calder, B., Wang, J., Ogus, A., Nilakantan, N., Skjolsvold, A., McKelvie, S., ... & Rigas, L. (2011, October). Windows azure storage: a highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles* (pp. 143-157).

Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., & Tzoumas, K. (2015). Apache Flink: Stream and batch processing in a single engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 38(4), 28-38.

Castro, M., & Liskov, B. (2002). Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, *20*(4), 398-461.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., ... & Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. *ACM SIGOPS operating systems review*, *41*(6), 205-220.

Ghemawat, S., Gobioff, H., & Leung, S. T. (2003, October). The Google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles* (pp. 29-43).

Gifford, D. K. (1979, December). Weighted voting for replicated data. In *Proceedings of the seventh ACM symposium on Operating systems principles* (pp. 150-162).

Gray, J. (1978). Notes on Database Operating Systems. Operating Systems: An Advanced Course, LNCS, vol. 60.

Hendricks, J., Ganger, G. R., & Reiter, M. K. (2007, August). Verifying distributed erasure-coded data. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing* (pp. 139-146).

Hendricks, J., Ganger, G. R., & Reiter, M. K. (2007). Low-overhead Byzantine fault-tolerant storage. ACM Transactions on Storage (TOS), 3(3), 1-23.

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.

Huang, C., Simitci, H., Xu, Y., Ogus, A., Calder, B., Gopalan, P., ... & Yekhanin, S. (2012). Erasure coding in windows azure storage. In *Presented as part of the 2012 {USENIX} Annual Technical Conference ({USENIX}{ATC} 12)* (pp. 15-26).

Kraska, T., Hentschel, M., Alonso, G., & Kossmann, D. (2009). Consistency rationing in the cloud: Pay only when it matters. *Proceedings of the VLDB Endowment*, *2*(1), 253-264.

Lakshman, A., & Malik, P. (2010). Cassandra: a decentralized structured storage system. *ACM SIGOPS operating systems review*, *44*(2), 35-40.

Lamport, L. (1998). The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, *16*(2), 133-169.

Lamport, L. (2005). Generalized consensus and Paxos. *Technical Report MSR-TR-2005-33, Microsoft Research*.

Li, C., Sivasubramanian, N., & Adve, S. (2012). De-indirection for flash-based SSDs with nameless writes. Proceedings of the 10th USENIX Conference on File and Storage Technologies, 1-14.

Li, H., Ghodsi, A., Zaharia, M., Shenker, S., & Stoica, I. (2014). Tachyon: Reliable, memory speed storage for cluster computing frameworks. Proceedings of the ACM Symposium on Cloud Computing, 6-8.

Li, J., Chen, Z., Srinivasan, A., & Zhou, Y. (2006). C-Miner: Mining block correlations in storage systems. Proceedings of the 3rd USENIX Conference on File and Storage Technologies, 173-186.

Lloyd, W., Freedman, M. J., Kaminsky, M., & Andersen, D. G. (2011, October). Don't settle for eventual: Scalable causal consistency for wide-area storage with COPS. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles* (pp. 401-416).

Mace, J., Roelke, R., & Fonseca, R. (2018). Pivot tracing: Dynamic causal monitoring for distributed systems. *ACM Transactions on Computer Systems (TOCS)*, *35*(4), 1-28.

Ongaro, D., & Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)* (pp. 305-319).

Plank, J. S. (1997). A tutorial on Reed–Solomon coding for fault-tolerance in RAID-like systems. *Software: Practice and Experience*, *27*(9), 995-1012.

Reed, I. S., & Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, *8*(2), 300-304.

Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011). Conflict-free replicated data types. In *Stabilization, Safety, and Security of Distributed Systems: 13th International Symposium, SSS 2011, Grenoble, France, October 10-12, 2011. Proceedings 13* (pp. 386-400). Springer Berlin Heidelberg.

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010, May). The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)* (pp. 1-10). IEEE.

Skeen, D. (1981, April). Nonblocking commit protocols. In *Proceedings of the 1981 ACM SIGMOD international conference on Management of data* (pp. 133-142).

Terry, D. (2013). Replicated data consistency explained through baseball. *Communications of the ACM*, *56*(12), 82-89.

Terry, D. B., Demers, A. J., Petersen, K., Spreitzer, M. J., Theimer, M. M., & Welch, B. B. (1994, September). Session guarantees for weakly consistent replicated data. In *Proceedings of 3rd International Conference on Parallel and Distributed Information Systems* (pp. 140-149). IEEE.

Van Renesse, R., & Schneider, F. B. (2004). Chain replication for supporting high throughput and availability. In Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation, 91-104.

Vogels, W. (2009). Eventually consistent. *Communications of the ACM*, *52*(1), 40-44.

Xin, R. S., Rosen, J., Zaharia, M., Franklin, M. J., Shenker, S., & Stoica, I. (2013, June). Shark: SQL and rich analytics at scale. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of data* (pp. 13-24).

Yu, H., & Vahdat, A. (2000, June). Building replicated internet services using TACT: A toolkit for tunable availability and consistency tradeoffs. In *Proceedings Second International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems. WECWIS 2000* (pp. 75-84). IEEE.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Ghodsi, A, & Stoica, I. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM*, *59*(11), 56-65