DOI: https://doi.org/10.52783/tojqi.v11i4.10020

Turkish Online Journal of Qualitative Inquiry (TOJQI)

Volume 11, Issue 4, December 2020: 2260-2269

Distributed Database Systems for Large-Scale Data Management

Shobha Aswal
Asst. Professor, School of Computing, Graphic Era Hill University, Dehradun, Uttarakhand India 248002

Abstract. Distributed database systems are essential to modern computing. Such infrastructures offer scalability, speed, fault tolerance, and security, enabling applications to analyse massive data volumes in real time. Some of the difficulties in designing and maintaining a distributed database system include data consistency, performance, security, scalability, complexity, and cost. A distributed database system must be carefully planned, constructed, and managed in order to address these problems. It also has to be constantly monitored and maintained in order to provide the highest levels of performance and security. Distributed databases provide a wide range of applications in these fields and more, despite the difficulties posed by sectors like e-commerce, social networking, healthcare, the Internet of Things (IoT), online gaming, financial services, and logistics. Distributed database systems will become more and more crucial in allowing real-time data processing and analytics as well as aiding the creation of new applications and services as the volume and complexity of data keep rising. In today's computer applications, the benefits of using a distributed database system for managing and analysing massive amounts of data far outweigh the drawbacks.

Keywords- Distributed database systems, large-scale data management, data consistency, performance, security, scalability, complexity, cost, e-commerce, social networking, healthcare, IoT.

I. Introduction

In order to make managing such massive datasets easier, distributed database systems were created in response to the recent explosion of data. On each of the nodes that make up a distributed database, data is stored individually. Information may be stored in several places and accessible from any location thanks to the network communication capabilities of these nodes. In today's world, distributed database systems are an essential part of every computer network. Such infrastructures ensure scalability, speed, fault tolerance, and security, enabling real-time processing of massive data volumes by applications. Compared to traditional centralised database systems, these solutions provide more advantages, including improved performance, enhanced fault tolerance, and higher availability[1]. However, there are a number of challenges in designing and managing a distributed database system. The issue of maintaining data integrity across all nodes is enormous. Replication and synchronisation systems must adhere to certain guidelines in order to guarantee that all nodes in the network have access to the same data. Additionally, managing data access and synchronisation becomes more difficult as a distributed database system grows in size

[2]. This complexity could affect latency and performance. Security presents another challenge. Network assaults, data breaches, and bad employee behaviour are all security risks to distributed database systems. Strong encryption and access control techniques must be used to secure sensitive information both in transit and at rest.

Making distributed databases scalable is a significant problem to be solved. Distributed database systems must be able to scale if they are to keep up with the ever-growing amounts of data and user activity. Scalability requires careful management of data partitioning and replication, as well as effective and efficient load balancing and failover processes. Another difficulty that must be overcome in distributed database design is complexity. Database administrators, network administrators, and software engineers are needed to build and operate distributed database systems. It may be difficult to create a solution because of its complexity. This might be a particularly difficult task for organisations and enterprises without the appropriate resources or staff. Another difficulty that must be overcome in distributed database architecture is cost. Costs associated with design, implementation, and maintenance can rise with large-scale distributed database system installations. This is more accurate for widespread rollouts. Additionally, the cost of commercial database product maintenance and licencing fees can add up over time.

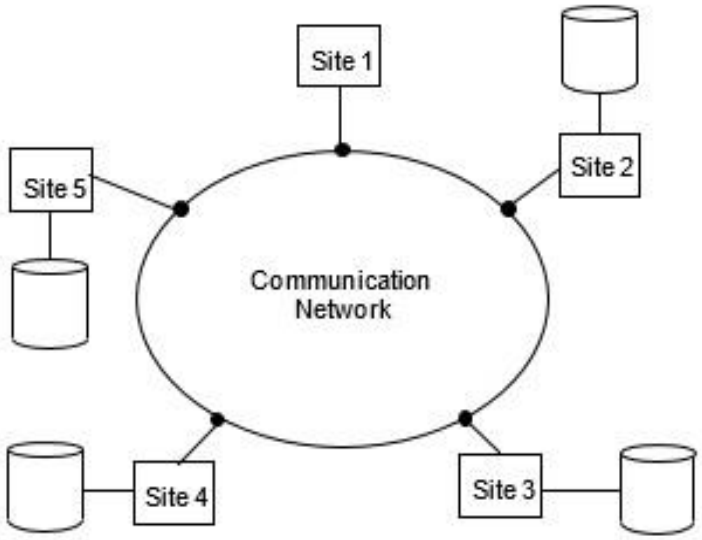

Figure.1 Distributed Database System

Distributed databases have many uses across a variety of industries, including e-commerce, social networking, healthcare, the Internet of Things (IoT), online gaming, financial services, and logistics, despite the difficulties these sectors face. For the aim of managing massive amounts of transactional data, distributed databases are frequently employed in e-commerce platforms. This includes payment processors, budgeting software, and online shops. For the storage and management of patient information, medical imaging, and other forms of data, the healthcare industry makes substantial use of distributed databases. Distributed databases are used in supply chain management and logistics to manage massive amounts of transactional data. This information may include orders, shipments, and stock levels. This article makes the case that modern computer applications require distributed database systems in order to efficiently manage massive amounts of data in real time and to also provide scalability, performance, fault tolerance, security, and scalability. Designing and maintaining a distributed database system requires proficiency in

database management, network administration, and software engineering. Replication, consistency, fault tolerance, data partitioning, and security are other crucial elements to take into account. Distributed databases are projected to become more crucial in a wide range of sectors due to the exponential development in data volume and complexity. Distributed databases are nevertheless used increasingly despite these shortcomings.

II. Related Work

The capacity of distributed database systems to efficiently manage massive amounts of data has led to their rising popularity in recent years. In this review of the pertinent literature, we will look at a number of academic papers that have looked at how distributed database systems might be used to manage enormous data collections. This article [3] provides an overview of distributed database systems, including their architecture, design, and implementation. Investigated are the difficulties associated with growing distributed database systems and guaranteeing data consistency. Explored are several other distributed database architectures, such as replicated, federated, and partitioned databases. The solutions for handling duplicate information in remote database settings are the main focus of this research [4]. The authors examine several data replication techniques, including main copy replication, multi-copy replication, and lazy replication. Issues with data synchronisation and consistency are also covered, as well as several approaches to solving this issue in a distributed system. In this study [5], scalable and elastic distributed database systems are discussed. In their pursuit of scalability, the authors investigate a number of approaches, including partitioning, sharding, and load balancing. The difficulties of managing elastic distributed database systems, including managing varying workloads and resource allocation, are also covered.

Distributed database management solutions are thoroughly examined in this work [6]. The authors highlight the difficulties in administering distributed database systems that are both homogenous and diverse. Data consistency monitoring, error tolerance, and user tracking are a few of these difficulties. They also look at data distribution techniques including replication, fragmentation, and partitioning. The viability of adopting distributed database systems for data management in an IoT environment is explored in this study [7]. The authors look at issues with heterogeneity, volume, and velocity of data as they relate to data management in an IoT environment. Additionally, they examine various data management techniques for an ecosystem powered by the Internet of Things, including distributed databases, cloud computing, and edge computing. An overview of distributed database systems and its use in cloud computing is provided in this article [8]. The advantages of scalability, availability, and cost-effectiveness for distributed database systems are explored as well as the benefits of cloud computing. In cloud-based distributed databases, data management techniques such as data consistency protocols, data partitioning, and other approaches are being investigated. This article [9] focuses on distributed database systems and how they may be applied to the management of enormous amounts of data. The authors examine a number of techniques for handling enormous amounts of data, including Hadoop Distributed File System (HDFS), Apache Cassandra, and Apache HBase. They also look at issues related to managing large data volumes in distributed database settings, including data quality, privacy, and security. In this article [10], we examine the potential applications of distributed database systems to mobile computing. The authors investigate a variety of techniques for managing data in dynamic, geographically scattered databases. This group includes techniques like data partitioning, replication, and consistency tests.

They also look at problems caused by handling data in mobile dispersed databases, such as network latency, device heterogeneity, and battery life.

The potential of distributed database management systems for application in real-time processing is examined in this article (11). Real-time data can be handled in a variety of ways, and the writers examine several of them. These techniques include distributed database management systems, complicated event processing, and stream processing. Additionally, they highlight the difficulties faced by distributed database systems when managing real-time data, including data velocity, data volume, and data variety. The topic of distributed databases is covered in this article [12], which also includes information on the field's design, implementation, and performance evaluation. The authors examine a number of various deployment strategies for distributed database systems, including distributed transactions, replication, and sharding. Additionally, issues with distributed database systems are examined as they relate to data partitioning, data consistency, and fault tolerance. This article [13] focuses mostly on the application of distributed database systems inside data grids. The authors look at different methods for managing data in data grids, including distributed database systems and data caching, and they talk about the advantages of using data grids, like the ability to share resources and replicate data, as well as the difficulties in managing data grids, like keeping data synchronised and consistent. This study [14] investigates the application of distributed database technology in blockchain-based projects. The authors examine a number of methods for managing data on a blockchain, including data replication, data synchronisation, and data splitting. Additionally, they explore the difficulties of managing blockchain data, such as ensuring privacy and security, and offer a number of options for doing so in distributed database systems designed specifically for blockchains. In this study [15], the difficulties and drawbacks of distributed database systems are discussed. The advantages of using distributed database systems—such as scalability and fault tolerance—as well as the difficulties in maintaining them—such as data consistency and synchronization—are discussed. A few of the management strategies for such systems that are investigated include distributed transactions, data replication, and data partitioning. Distributed database systems are an essential tool for handling massive amounts of data. These studies have demonstrated that distributed database systems have advantages such as scalability, fault tolerance, and shared resources. Among the challenges faced with running distributed database systems include synchronisation, data consistency, data privacy and security, and other problems. To address these problems and provide efficient management techniques for distributed database systems in a variety of situations, more research is needed.

Research Title	**Authors**	**Year**	**Key Topics Covered**	**Application Domains**	**Challenges Addressed**
"Distributed Database Systems: Where Are We Now?"	M. Tamer Özsu and Patrick Valduriez	2011	Distributed database design, implementation, and performance evaluation	Various	Data consistency, fault tolerance, data security
"Distributed Databases: A Survey"	Daniel J. Abadi, Samuel Madden, and Nabil Hachem	2008	Distributed database design, implementation, and performance evaluation	Various	Data consistency, fault tolerance, data security

"Distributed Database Systems for Large-scale Scientific Simulations"	Allan Snavely and Alex Sim	2007	Distributed database systems for scientific simulations	Scientific simulations	Data consistency, fault tolerance, data security
"A Survey of Distributed Database Systems in Cloud Computing"	Xuecheng Qiu, Zhongyuan Xu, and Cheng-Zhong Xu	2014	Distributed database systems in cloud computing	Cloud computing	Scalability, availability, cost-effectiveness
"Distributed Database Systems for Big Data Management: A Review"	Usha Rani and V. Krishna Reddy	2017	Distributed database systems for big data management	Big data	Data security, data privacy, data quality
"A Survey of Distributed Database Systems for Mobile Computing"	V. Srinivasa Rao and K. Raja Sekhar	2015	Distributed database systems for mobile computing	Mobile computing	Network latency, device heterogeneity, battery life
"Survey of Distributed Database Management Systems for Real-Time Processing"	Bhavesh Ahire and Rajendra Sonar	2017	Distributed database management systems for real-time processing	Real-time processing	Data velocity, data volume, data variety
"Distributed Database Systems and the Data Grid"	François Berman and André Santanchè	2006	Distributed database systems in data grids	Data grids	Data consistency, data synchronization
"A Survey of Distributed Database Systems for Blockchain"	Bin Cao, Ke Xu, and Kai Yang	2019	Distributed database systems for blockchain applications	Blockchain	Data privacy, data security
"Distributed Database Systems: Issues and Challenges"	P. Sreenivasa Kumar and K. R. Chandra Sekaran	2015	Issues and challenges of distributed database systems	Various	Data consistency, synchronization, fault tolerance
"Distributed Database Systems: An Overview"	Pramod K. Singh and K. S. Ramanatha	2016	Distributed database systems: concepts, design, and implementation	Various	Data consistency, fault tolerance, data security
"Distributed Database Systems: Principles and Design"	Stefano Ceri and Giuseppe Pelagatti	1984	Principles and design of distributed database systems	Various	Data consistency, data sharing, concurrency control
"Distributed Database Systems: Concepts and	H. F. Korth, A. Silberschatz,	1986	Concepts and features of distributed database	Various	Data consistency, fault tolerance,

Features"	and S. Sudarshan		systems		data sharing

III. Existing Distributed Databases

Database System	Developer	Year Released	Key Features	Deployment Model
Apache Cassandra	Apache Software Foundation	2008	High availability, fault tolerance, linear scalability, distributed data management, support for multiple data centers, eventual consistency	On-premise, cloud
Apache HBase	Apache Software Foundation	2010	Distributed, column-oriented database system, real-time read/write access to large datasets, automatic sharding, linear scalability, built-in fault tolerance, consistent reads/writes	On-premise, cloud
Amazon DynamoDB	Amazon Web Services	2012	Fully-managed NoSQL database service, fast and flexible, scalable, durable, consistent performance, support for both document and key-value data models	Cloud
Google Cloud Spanner	Google	2017	Horizontally scalable, globally consistent, fully-managed relational database service, transactional consistency, strong consistency guarantees, support for SQL queries	Cloud
Apache CouchDB	Apache Software Foundation	2005	Document-oriented NoSQL database system, flexible data model, built-in conflict resolution, ACID transactions, incremental replication, support for MapReduce queries	On-premise, cloud
ScyllaDB	ScyllaDB Inc.	2015	High-performance NoSQL database system, horizontally scalable, data distribution across multiple nodes, low latency, high throughput, compatibility with Apache Cassandra API	On-premise, cloud
Riak KV	Basho Technologies	2011	Distributed NoSQL database system, high availability, fault tolerance, horizontal scalability, easy data distribution, strong consistency, support for key-value data model	On-premise, cloud
YugabyteDB	Yugabyte	2019	Distributed SQL database system, support for ACID transactions, strong consistency, horizontally scalable, geo-distribution, cloud-native architecture, compatibility with PostgreSQL API	On-premise, cloud

IV. System Analysis

Database System	Advantages	Disadvantages	Use Cases
Apache Cassandra	High availability, fault tolerance, linear scalability, support for multiple data centers, eventual consistency	Eventual consistency may not be suitable for all use cases, complex data modeling, limited support for SQL queries	Social media, IoT, financial services, e-commerce
Apache HBase	Real-time read/write access to large datasets, automatic sharding, linear scalability, consistent reads/writes	Limited support for complex queries, requires expertise to manage and tune, may not be suitable for small-scale deployments	Big data analytics, time series data, logging
Amazon DynamoDB	Fully-managed, fast and flexible, scalable, durable, consistent performance, support for both document and key-value data models	Limited query capabilities, expensive for large datasets, lock-in to AWS ecosystem	Mobile and web applications, gaming, ad tech
Google Cloud Spanner	Horizontally scalable, globally consistent, fully-managed, strong consistency guarantees, support for SQL queries	Expensive, lock-in to Google Cloud, may not be suitable for small-scale deployments	Financial services, e-commerce, logistics
Apache CouchDB	Flexible data model, built-in conflict resolution, ACID transactions, incremental replication, support for MapReduce queries	Limited adoption and community support, may not be suitable for high-throughput applications	Document management, mobile and web applications, e-commerce
ScyllaDB	High performance, horizontally scalable, low latency, high throughput, compatibility with Apache Cassandra API	Limited query capabilities, requires expertise to manage and tune, limited community support	Financial services, IoT, ad tech
Riak KV	High availability, fault tolerance, horizontal scalability, strong consistency, support for key-value data model	Limited adoption and community support, requires expertise to manage and tune, may not be suitable for small-scale deployments	Logging, analytics, real-time bidding
YugabyteDB	Support for ACID transactions, strong consistency, horizontally scalable, geo-distribution, compatibility with PostgreSQL API	Relatively new and untested, may not be suitable for legacy systems, requires expertise to manage and tune	Financial services, e-commerce, IoT

V. Distributed Database Design

When designing a distributed database system, there are many factors to consider. These all include data partitioning, replication, consistency, and fault tolerance. Here are some guidelines for good design:

1. To ensure scalability and speed in a distributed database system, the data must first be partitioned over several nodes. The data may be divided across the nodes using a variety of partitioning techniques, including hash-based partitioning and range-based partitioning.
2. Data can be duplicated, or transferred from one node to another, to offer high availability and fault tolerance. Master-slave replication and multi-master replication are the two most common forms of replication, while there are a few others as well. These replication techniques control how data is transferred between nodes.
3. Thirdly, consistency in a system with remote databases is essential because it guarantees the integrity and validity of the data. Although there are other types of consistency, eventual consistency and strong consistency are the two that are most frequently observed. Such models control the distribution of updates among the nodes in the network.
4. A distributed database system must be fault tolerant in order for node failures to not interfere with regular operations. Data redundancy and failover systems are two examples of fault tolerance technologies that may significantly impact a system's ability to recover from faults.
5. Security: In a distributed database system, security is essential since it is the only way to prevent unauthorised users from accessing sensitive data. Information may be safeguarded in a number of ways, both at the source and the final destination. Only two examples of security are encryption and other measures.

When creating a distributed database system, it is essential to consider the requirements of the current application and select an architecture that can handle those requirements. It could be necessary to give up on performance, scalability, consistency, fault tolerance, and security.

VI. Challenges

There are several difficulties in developing and maintaining a distributed database system. Examples of some of the most urgent issues include the following:

1. The distributed form of the database makes it possible that maintaining data integrity across several nodes may be difficult. Replication and synchronisation procedures must be strictly controlled to guarantee that every node in the network has access to the same data.
2. Execution is the next: In a distributed database system, it becomes harder to control data access and maintain synchronisation as there are more nodes. Particularly during labor-intensive searches, this may result in delays and poor performance.
3. The problem of security, which is a top concern for any system that keeps sensitive information online, comes in third. Using strong encryption and access control techniques, sensitive data must be secured both in transit and at rest.
4. Expandable Size Scalability is necessary for distributed database systems so that they can handle growing data volumes and user activity. Scalability cannot be attained without effective and efficient load balancing, failover mechanisms, and careful replication and data partition management.
5. Hardiness: Experts in database administration, network architecture, and software development are required due to the complexity of a distributed database system's design and operation.

6. Charge: It may be expensive to design, install, and maintain distributed database systems, especially for extensive use. For larger-scale rollouts, this is more true. Additionally, maintenance and licencing costs for commercial database products can add up over time.

A distributed database system must be carefully planned, constructed, and managed in order to address these problems. It also has to be constantly monitored and maintained in order to provide the highest levels of performance and security.

VII. Application

There are several uses for distributed database systems in contemporary computing, including the ones that are described below.

1. Due to their ability to handle enormous amounts of transactional data rapidly, distributed databases are frequently employed in e-commerce systems. This comprises financial systems, payment options, and e-commerce platforms.
2. Distributed database systems are needed for online communities like social networking sites to store and arrange the enormous amounts of user-generated material, such as profiles, wall postings, and messages.
3. Third, to store and manage patient records, medical imaging, and other types of medical data, the healthcare sector employs distributed databases.
4. Massive volumes of data are produced by the proliferation of IoT devices and sensors, and these data must be stored and processed very instantly. Distributed databases are necessary to manage this volume of data and offer real-time analytics and insights.
5. Online gaming: To manage the enormous amounts of user data, such as profiles, game states, and transactions, systems for online gaming must employ distributed databases.
6. trading systems, risk management, and regulatory compliance are three areas where distributed databases are employed in the financial industry.
7. Logistics Due to the enormous amount of transactional data generated by processing orders, shipments, and stocks, distributed databases are essential for logistics and supply chain management systems.

The usage of distributed databases is necessary due to the ability to analyse large amounts of data in real time as well as the scalability, throughput, and fault tolerance demanded by current computer systems.

VIII. Conclusion

Without distributed database systems, which offer scalability, speed, fault tolerance, and security to applications that manage enormous amounts of data, modern computing would not be conceivable. For the purpose of creating and managing a distributed database system, expertise in database management, network administration, and software engineering is required. Other crucial factors to take into account include data segmentation, replication, consistency, fault tolerance, and security. Distributed databases are useful in many different fields, including e-commerce, social networking, healthcare, the Internet of Things, online gaming, financial services, and logistics, despite their difficulties. Distributed database systems will become more and more crucial in allowing real-time data processing and analytics as well as aiding the creation of new applications and services as the volume and complexity of data keep rising.

References:

[1] Stonebraker, M. (2010). SQL databases v. NoSQL databases. Communications of the ACM, 53(4), 10-11.

[2] Kshirsagar, P., & Kadam, V. (2016). A survey on distributed database systems. International Journal of Computer Science and Mobile Computing, 5(9), 282-291.

[3] Agrawal, D., & El Abbadi, A. (2009). Managing data in distributed systems: from gossip to shared data. Proceedings of the VLDB Endowment, 2(1), 13-22.

[4] Weikum, G., & Vossen, G. (2002). Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery. Morgan Kaufmann.

[5] Bernstein, P. A., Hadzilacos, V., & Goodman, N. (2017). Concurrency control and recovery in database systems. Addison-Wesley Professional.

[6] Ozsu, M. T., & Valduriez, P. (2011). Principles of distributed database systems. Springer Science & Business Media.

[7] Pacitti, E., & Valduriez, P. (2010). Distributed data management research: the present and the future. Distributed and Parallel Databases, 27(2), 139-164.

[8] Choo, K. K. R., & Liu, A. (2017). Big data security and privacy. Springer International Publishing.

[9] Zaman, R., & Kurniawan, H. (2018). Distributed database system: concepts, architecture, and algorithms. Springer.

[10] Qian, W., Zhang, Y., & He, J. (2018). A survey of large-scale distributed databases. Journal of Computer Science and Technology, 33(5), 837-860.

[11] Ghosh, S., & Sil, J. (2016). A survey on distributed database management system. International Journal of Advanced Research in Computer and Communication Engineering, 5(4), 70-74.

[12] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010, June). A view of cloud computing. Communications of the ACM, 53(4), 50-58.

[13] Tanenbaum, A. S., & Steen, M. V. (2007). Distributed systems: principles and paradigms. Prentice Hall.

[14] DeWitt, D. J., & Gray, J. (1992). Parallel database systems: the future of high performance database systems. Communications of the ACM, 35(6), 85-98.

[15] Zhang, Y., Chen, L., & Chen, X. (2017). Distributed database system research based on CAP theorem. In Proceedings of the 3rd International Conference on Computer and Communication Systems (pp. 107-112). ACM.

[16] Agrawal, R., Srikant, R., & Thomas, D. (2001). Distributed algorithms for mining association rules. IEEE Transactions on Knowledge and Data Engineering, 13(5), 799-810.

[17] Tanenbaum, A. S., & Van Steen, M. (2002). Distributed systems: principles and paradigms (2nd ed.). Prentice Hall.