Design and Implementation of a Distributed Computing System for Scalable Data Processing

HIMANI SIVARAMAN,
Department of Comp. Sc. & Info. Tech.,
Graphic Era Hill University, Dehradun, Uttarakhand, India 248002, himanisivaraman@gmail.com

Abstract:
Even though distributed systems are already widely used, research in this field is still frequently challenging. This is partially explained by the fact that such systems have several sides and that it is inherently difficult to separate them from one another. We give a quick introduction to distributed systems in this paper, outlining what they are, the main objectives of their design, and some of the most prevalent varieties. The paper discuss about design and implementation of a distributed system for scalable data processing.

DOI Number: 10.48047/nq.2020.18.6.NQ20185 **NeuroQuantology 2020; 18(6):67-73**

Introduction:

Distributed computing is a method of processing information or performing calculations by distributing them across multiple computers connected to a network. Rather than relying on a single central computer, a distributed computing system uses a network of computers that work together to achieve a common goal.

In a distributed computing system, tasks are divided into smaller pieces, and each piece is assigned to a different computer. These computers work in parallel to complete their assigned tasks, communicating with each other over the network to coordinate their efforts and share data as needed. By dividing the workload among multiple computers, distributed computing systems can perform computations faster and more efficiently than a single computer could.

Distributed computing systems are used in a variety of applications, including scientific research, financial modelling, and data processing. Some examples of distributed computing systems include cloud computing platforms, peer-to-peer networks, and grid computing systems.

Working of Distributed Computing System :

Distributed computing is a model of computing in which a group of computers work together to solve a complex problem. The goal is to split up the problem into smaller tasks and distribute those tasks among the computers in the network. Here are the basic steps involved in a distributed computing system:

i) Task Partitioning: The problem to be solved is divided into smaller sub-problems that can be solved in parallel.

ii) Task Assignment: Each sub-problem is assigned to a computer in the network. The assignment can be done either in a centralized or decentralized manner.

iii) Task Execution: Each computer independently solves its assigned sub-problem using its own resources.

iv) Result Aggregation: The results from each computer are collected and combined to obtain the final result.

v) Communication: The computers in the network communicate with each other to coordinate the task partitioning, assignment, and result aggregation.

vi) Error Handling: The system should be designed to handle errors that may occur during task execution, such as a computer crashing or a network failure.

There are many different approaches to implementing distributed computing systems, including client-server models, peer-to-peer networks, and cloud computing. Distributed computing can be used for a variety of applications, such as scientific simulations, data processing, and machine learning. It allows for efficient use of resources and can help solve problems that are too large for a single computer to handle.

Different types of Distributed Computing:

Distributed computing systems are a type of computing system that uses multiple computers to work together to accomplish a common goal. There are several types of distributed computing systems:

i) Cluster Computing: Cluster computing involves connecting multiple computers (or nodes) together to form a single system. The nodes in a cluster typically share the same physical network, and work together to perform tasks that require high computational power, such as scientific simulations, data analysis, or machine learning.

ii) Grid Computing: Grid computing is similar to cluster computing, but instead of being limited to a single physical network, it can connect computers across different networks and even different geographic locations. This makes it useful for large-scale distributed projects, such as scientific research or weather modelling.

iii) Cloud Computing: Cloud computing involves the use of remote servers to store, manage, and process data. Cloud computing is often used for hosting web applications, delivering software as a service, and providing on-demand computing resources to businesses and individuals.

iv) Peer-to-Peer (P2P) Computing: P2P computing involves connecting computers directly to each other, rather than through a central server. This allows for decentralized communication and data sharing, and is often used for file sharing and messaging applications.

v) Distributed Data Processing: Distributed data processing involves dividing large datasets into smaller pieces and distributing them across multiple computers for processing. This allows for faster processing times and greater scalability, and is often used in big data analytics and machine learning applications.

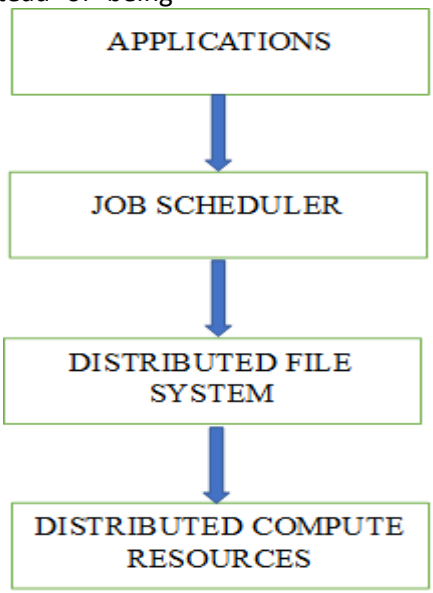

Figure :

The figure 1. shows four main components of a distributed computing system:

i) Application: This is the program or software that runs on the distributed system, and typically includes user-facing features, algorithms, or analytical models.

ii) Job Scheduler: This component is responsible for managing and scheduling the

processing of tasks or jobs across the distributed compute resources. It ensures that each node is used efficiently and that no task is duplicated or missed.

iii) Distributed File System: This component allows the distributed system to share and access files across different nodes, allowing for efficient data processing and analysis.

iv) Distributed Compute Resources: These are the individual computers or nodes that make up the distributed system, and they work together to complete the computational tasks assigned by the job scheduler. Each node may have different processing power, storage capacity, or network speed, and the system must optimize the use of these resources to achieve optimal performance.

Benefits of Distributed Computing System:

Distributed computing systems offer several advantages over traditional centralized computing systems. Here are some of the key advantages:

i) Scalability: Distributed computing systems can easily scale up or down based on the workload and the number of users. This allows the system to handle increasing traffic and data volumes without any performance degradation.

ii) Fault tolerance: Distributed computing systems are designed to handle system failures or crashes without affecting the overall system performance. If one node fails, the other nodes can take over its workload, ensuring that the system remains up and running.

iii) Cost-effective: Distributed computing systems can be built using commodity hardware, which is less expensive than specialized hardware. This reduces the cost of building and maintaining the system.

iv) Improved performance: Distributed computing systems can process large amounts of data faster than traditional centralized systems. This is because the workload is divided across multiple nodes, and each node processes a smaller portion of the data.

v) Decentralization: Distributed computing systems are decentralized, which means that the system does not rely on a single point of failure. This makes the system more resilient and less vulnerable to cyber-attacks or other security threats.

vi) Flexibility: Distributed computing systems can be used for a wide range of applications, from scientific simulations to web hosting and data analytics. This makes the system flexible and adaptable to different business needs and use cases.

Applications of Distributed Computing System:

Distributed systems have a wide range of applications in different fields. Here are some of the most common applications of distributed systems:

i) Cloud Computing: Distributed systems are used in cloud computing platforms to provide on-demand computing resources to businesses and individuals. Cloud computing allows users to access computing resources from anywhere in the world and pay only for what they use.

ii) Big Data Analytics: Distributed systems are used for processing and analyzing large data sets in a timely and efficient manner. Distributed systems can process data across multiple nodes simultaneously, allowing for faster and more accurate analysis.

iii) Internet of Things (IoT): Distributed systems are used in IoT to collect data from sensors and other devices and process that data in real-time. IoT devices are often located in remote locations, and distributed systems allow for efficient processing of that data.

iv) Video Streaming: Distributed systems are used in video streaming services to deliver high-quality video content to users. The video content is stored and processed across multiple nodes, ensuring that the video stream is delivered in real-time without buffering.

v) Distributed Gaming: Distributed systems are used in online gaming platforms to enable players to compete against each other in real-time. Distributed gaming allows for a large number of players to compete against each other simultaneously without any lag or delay.

vi) Distributed Artificial Intelligence (AI): Distributed systems are used in AI applications to train and optimize machine learning models. Distributed AI allows for the processing of large amounts of data and the

training of complex models that would be impossible to train on a single computer.

vii) Distributed Databases: Distributed systems are used in databases to store and manage large amounts of data across multiple nodes. Distributed databases provide high availability and fault tolerance, ensuring that the data is always accessible and up-to-date.

Designing and implementing a distributed computing system for scalable data processing requires careful consideration of various factors. Here are some key steps that can help in designing and implementing such a system:

i) Identify the business requirements: The first step in designing a distributed computing system is to identify the business requirements. This includes understanding the type and volume of data to be processed, the expected processing time, and the desired level of scalability.

ii) Select the appropriate architecture: There are several architectures to choose from when designing a distributed computing system, including client-server, peer-to-peer, and hybrid architectures. The choice of architecture will depend on the specific requirements of the system.

iii) Choose the right tools and technologies: The tools and technologies used in the system should be selected based on the specific requirements of the system. This includes choosing the appropriate programming languages, frameworks, and libraries.

iv) Design the data processing pipeline: The data processing pipeline is the core of the distributed computing system. It is responsible for processing the data in a scalable and efficient manner. The pipeline should be designed to minimize latency, maximize throughput, and provide fault tolerance.

v) Choose the right data storage solution: The data storage solution used in the system should be chosen based on the type and volume of data to be processed. This includes selecting the appropriate database technology, data model, and data storage architecture.

vi) Design the data distribution strategy: The data distribution strategy is responsible for distributing the data across the nodes in the system. This includes choosing the appropriate data partitioning strategy and replication strategy.

vii) Implement the system: The implementation of the system should be done in a way that minimizes the risk of failure and maximizes the reliability of the system. This includes implementing monitoring and logging capabilities, testing the system thoroughly, and ensuring that the system can handle the expected workload.

viii) Deploy and manage the system: Once the system has been implemented, it needs to be deployed and managed in a way that maximizes its availability and reliability. This includes implementing backup and recovery procedures, monitoring the system for performance and availability issues, and scaling the system up or down as needed.

In summary, designing and implementing a distributed computing system for scalable data processing requires careful planning, choosing the appropriate architecture, selecting the right tools and technologies, designing the data processing pipeline, choosing the right data storage solution, designing the data distribution strategy, implementing the system, and deploying and managing the system.

Literature Review:

The paper discusses some of the challenges associated with operating a large-scale distributed file system like HDFS, such as maintaining consistency and ensuring data integrity. The authors also provide some insights into the performance characteristics of HDFS, including benchmarks that demonstrate its ability to handle large-scale data processing workloads [1].

The article then goes on to discuss some of the challenges and trade-offs involved in designing a distributed database system. For example, the author highlights the need to balance consistency and availability, as achieving perfect consistency can be difficult in a distributed system while maintaining high availability is crucial for many applications [2].

The paper discusses some alternative techniques for achieving consistency in distributed systems, including using application-level code to manage consistency

and using consensus algorithms to coordinate updates across multiple nodes [3].

The authors begin by discussing the challenges associated with processing large volumes of data and the need for specialized hardware and software to handle these workloads efficiently. They then introduce the SwissBox architecture, which is designed to provide a flexible, scalable, and cost-effective platform for building data processing appliances [4].

The paper discusses the design and implementation of Megastore, a scalable, highly available storage system that is designed to support interactive online services with strong consistency guarantees. Megastore combines the scalability and fault-tolerance of a NoSQL datastore with the consistency guarantees of a traditional relational database [5].

The paper presents ten rules for designing and implementing high-performance datastores that support simple operations. The authors define "simple operations" as operations that can be executed with a single network message and that involve accessing or updating a single record. Examples of simple operations include inserting, updating, and deleting records, as well as retrieving records by a primary key [6].

In addition to discussing the technical aspects of distributed database systems, the book also covers some of the practical issues involved in deploying and managing distributed databases, such as system administration, performance tuning, and backup and recovery [7].

The book also covers a range of topics related to distributed programming, including distributed communication, remote procedure call (RPC), distributed mutual exclusion and synchronization, distributed transactions, and distributed deadlock [8].

The paper proposes a model for middleware, which is a layer of software that provides common services and abstractions for distributed applications. The model defines four types of middleware services: communication, naming, synchronization, and persistence [9].

The paper describes the architecture of the Horus system, which is based on a group communication model. The system uses a distributed algorithm to replicate data across multiple nodes, ensuring that data is always available even in the event of a failure. The paper also describes the use of virtual synchrony, which is a technique for ensuring that messages are delivered in a consistent order across all nodes in the system [10].

Scalable data processing refers to the ability to process increasing amounts of data with increasing resources in a timely and efficient manner. Some of the characteristics of scalable data processing include:

i) Distributed computing: Scalable data processing often involves distributed computing, which means that the processing workload is spread across multiple machines or nodes.

ii) Horizontal scalability: A system that is horizontally scalable can handle increased workloads by adding more nodes to the system. This allows for greater capacity and can help to reduce bottlenecks.

iii) Fault tolerance: Scalable data processing systems should be designed to handle hardware and software failures. They should have the ability to detect and recover from failures to minimize downtime and ensure data integrity.

iv) Elasticity: The ability to scale up or down quickly in response to changing workloads is an important characteristic of scalable data processing. Elasticity allows the system to add or remove resources as needed to maintain performance.

v) High availability: Scalable data processing systems should be designed to provide high availability, which means that the system should be able to continue operating even if some nodes or components fail.

vi) Data partitioning: Large datasets can be partitioned across multiple nodes to enable parallel processing. This helps to distribute the workload and improve performance.

vii) Data locality: Data locality refers to the ability to store and process data on the same node. This can help to reduce network latency and improve performance.

Overall, scalable data processing systems need to be designed with scalability, fault tolerance, and high availability in mind to handle increasing amounts of data with increasing resources in a timely and efficient manner.

Need of Scalable data processing:

Scalable data processing is needed because the amount of data being generated is increasing exponentially, and traditional approaches to data processing are unable to keep up. With the rise of big data and the Internet of Things (IoT), organizations are faced with ever-increasing amounts of data that need to be processed and analysed. It allows organizations to handle large and complex datasets by distributing the workload across multiple nodes. This approach not only allows for faster processing of data but also provides a cost-effective way to handle large datasets without the need for expensive hardware upgrades. It also provides organizations with the ability to respond to changing business needs and workloads. With scalable data processing, resources can be added or removed as needed to ensure optimal performance and minimize costs.

In addition, scalable data processing enables organizations to extract valuable insights from their data, which can be used to make data-driven decisions and gain a competitive advantage. By analysing large datasets, organizations can identify patterns, trends, and anomalies that would be difficult or impossible to detect using traditional approaches.

Overall, scalable data processing is essential for organizations that need to process and analyse large and complex datasets to gain insights and make data-driven decisions. It provides a way to handle increasing amounts of data with increasing resources in a timely and cost-effective manner.

Table 1. Features of Scalable data processing

Systems	Mechanisms			
	Partitioning of Data	Operations	Schema	Consistency
Volt DB	Horizontal	x	x	Sequential
Megastore	Horizontal	x	KeyValue	Multiple
SQL Azure	Horizontal	x	x	Sequential
Hyder	No	x	Log	Strong
Key-value stores	Horizontal	✓	Key Value	Eventual

Conclusion:

The design and implementation of a distributed computing system for scalable data processing is essential for organizations that need to handle large and complex datasets. The development of such a system requires careful consideration of factors such as distributed computing, horizontal scalability, fault tolerance, elasticity, high availability, data partitioning, and data locality. A well-designed distributed computing system can provide organizations with faster processing times, greater capacity, cost-effective resource management, and the ability to respond quickly to changing workloads. It can also help to ensure data integrity, reduce downtime, and provide high availability. Implementing a distributed computing system requires specialized expertise in areas such as distributed systems, database design, and network architecture. It also requires the use of appropriate tools and technologies, such as NoSQL databases, distributed file systems, and message queues.

Overall, the design and implementation of a distributed computing system for scalable data processing is a complex and challenging task, but it is essential for organizations that need to process and analyse large and complex datasets. By leveraging the power of distributed computing, organizations can gain valuable insights from their data, make data-driven decisions, and gain a competitive advantage in today's data-driven business landscape.

References:

[1] Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop Distributed File System. In 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST) (pp. 1–10). IEEE. http://doi.org/10.1109/MSST.2010.5496972.

[2] Erb, B. (2016). The Challenge of Distributed Database Systems. Retrieved March 23, 2017, from http://berb.github.io/diploma-thesis/community/061_challenge.html.

[3] Helland, P.: Life beyond distributed transactions: an apostate's opinion. In: 3rd Biennial Conf. on Innov. Data Syst. Research (CIDR), Asilomar, CA, USA, pp. 132–141 (2007).

[4] Alonso, G., Kossmann, D., Roscoe, T.: SwissBox: An architecture for data processing appliances. In: 5th Biennial Conf. on Innov. Data Syst. Research (CIDR), Asilomar, CA, USA, pp. 32–37 (2011).

[5] Baker, J., Bond, C., Corbett, J.C., Furman, J.J., Khorlin, A., Larson, J., L´eon, J.M., Li, Y., Lloyd, A., Yushprakh, V.: Megastore: Providing scalable, highly available storage for interactive services. In: 5th Biennial Conf. on Innov. Data Syst. Research (CIDR), Asilomar, CA, USA, pp. 223–234 (2011).

[6] Stonebraker, M., Cattell, R.: Ten rules for scalable performance in "simple operation" datastores. Commun. ACM 54, 72–80 (2011).

[7] Ozsu, M. T., & Valduriez, P. (2011). Principles of Distributed Database Systems – M. Tamer Özsu, Patrick Valduriez – Google Books. Retrieved from https://books.google.com/books?hl=en&lr=&id=TOBaLQMuNV4C&oi=fnd&pg=PR3&dq=Distributed+Database&ots=LqFjgM_P-7&sig=mcmEnxerBLtixHY-0CrzS2hFojc#v=onepage&q=Distributed Database&f=false.

[8] Ben-Ari M (2006) Principles of concurrent and distributed programming, 2nd edn. Prentice Hall, Englewood Cliffs.

[9] Bernstein P (1996) Middleware: a model for distributed system services. Commun ACM 39(2):87–98.

[10] van Renesse R, Birman K, Cooper R, Glade B, Stephenson P (1994) The horus system. In: Birman K, van Renesse R (eds) Reliable and distributed computing with the Isis Toolkit. IEEE Computer Society Press, Los Alamitos, pp 133–147