Advances in Computer, Signals and Systems (2025)
Clausius Scientific Press, Canada
DOI: 10.23977/acss.2025.090104
ISSN 2371-8838 Vol. 9 Num. 1

Theoretical Analysis of Distributed Systems and Their Scalability

Chen Yang

Xiamen University Tan Kah Kee College, Zhangzhou Fujian, 363123, China

Keywords: Internet; Big Data; Distributed System; Scalability

Abstract: With the rapid development of the Internet and big data, distributed systems play an increasingly important role in various fields. This paper first introduces the basic theory of distributed systems, including definitions, characteristics and computational models, and then explores the design architecture of distributed systems, focusing on analysing different architectural patterns and protocols. This paper also discusses in depth the scalability of distributed systems, analyses the characteristics of horizontal and vertical scaling and the bottlenecks faced, and proposes a design strategy for scalability. The results show that a reasonable architecture and optimisation strategy is the key to achieving an efficient distributed system.

1. Introduction

A distributed system is a system structure in which multiple computing nodes work together to accomplish a certain task. With the rapid development of information technology, distributed systems have been widely used in the fields of big data processing, cloud computing, and Internet of Things. Its advantages in achieving high availability, scalability and fault tolerance make distributed systems an important part of modern computing architecture[1]. However, the complexity of distributed systems increases as the scale of the system expands, especially the challenges in performance optimisation, data consistency and scalability are gradually highlighted[2]. Therefore, it is of great theoretical significance and practical value to study the basic theory, architecture design, and scalability of distributed systems. In this paper, we will analyse the basic theory, architecture design, scalability and performance optimization of distributed systems in order to provide guidance for researchers and developers in related fields[3].

2. Basic Theory of Distributed Systems

2.1 Distributed System Definition and Characteristics

A distributed system consists of multiple computing nodes that work together over a network to complete tasks. These nodes can be spread out and each has its own resources, communicating through the network. There's no central control; nodes coordinate via message passing[4]. Key features are transparency, scalability, fault tolerance, and high availability. Transparency hides system details from users, scalability allows adding nodes without performance loss, fault tolerance maintains operation with redundant nodes, and high availability ensures continuous service even with component failures[5].

2.2 Distributed Computing Model

Distributed computing model is a theoretical framework that describes how the nodes in a distributed system perform collaborative computation. Common distributed computing models include the shared memory model, the message passing model, and the stream-based computing model. The shared memory model assumes that all nodes of the system can access the same shared memory and exchange data by reading and writing the shared memory; while the message passing model implements communication between nodes by sending and receiving messages, and the nodes do not share the memory directly with each other; and the stream-based computing model focuses on the delivery and processing of data flow, and the nodes carry out collaborative computation through streaming data transfer. The choice of distributed computing model directly affects the efficiency, scalability and fault tolerance of the system, and different application scenarios require different computing models to optimise performance[6]. For example, in scenarios requiring high throughput and low latency, a message-passing model may be more suitable, while a shared memory model or a streaming computing model may be more advantageous when dealing with large-scale datasets.

3. Design and Architecture of Distributed Systems

3.1 Architecture Patterns

Distributed system architecture patterns dictate how work is organized and coordinated across multiple nodes. Key patterns include client-server, peer-to-peer (P2P), and microservice architectures. In client-server, clients request services and servers process these requests, typically handling data storage and processing, with clients focusing on data display and interaction. P2P networks allow nodes to function as both clients and servers without a master-slave hierarchy, promoting equal node status. Microservice architecture decomposes large systems into small, independent services that communicate via APIs, enhancing flexibility and maintainability through modular design. The choice of architecture impacts system scalability, flexibility, and fault tolerance.

3.2 Distributed Protocols

Distributed protocols are rules that define how nodes in a distributed system collaborate and communicate. Common distributed protocols include consistency protocols, consensus protocols, arbitration protocols and synchronisation protocols. Consistency protocols such as Paxos and Raft protocols are used to ensure that multiple nodes in a distributed system agree, especially in distributed transactions or distributed databases, where it is critical to ensure data consistency. Consensus protocols are used to solve the problem of how nodes in a distributed system agree on a decision, and are often used in fault recovery and election processes in distributed systems. Arbitration protocols, on the other hand, are used to decide priority or contention issues between multiple nodes, while synchronisation protocols are used to ensure the order of operations between multiple nodes to ensure correctness and consistency of the system. The design of distributed protocols needs to take into account the fault tolerance, scalability and performance of the system, and a reasonable protocol design can greatly improve the reliability and efficiency of the system.

3.3 Data consistency and distributed transactions

Data consistency in distributed systems ensures that data copies across multiple nodes are identical at all times, despite challenges like network latency and node failures. To address this, systems use various consistency models and distributed transaction protocols like 2PC and 3PC to maintain transactional consistency and atomicity across nodes. However, these implementations can be performance-intensive, necessitating a balance between consistency and performance. In large-scale systems, an eventual consistency model is often adopted to enhance performance, allowing temporary inconsistencies but ensuring eventual data consistency.

4. Scalability of Distributed Systems

4.1 Overview of Scalability

Scalability is a core characteristic of distributed systems, which refers to the ability of a system to maintain or improve its performance by adding resources (e.g., compute nodes, storage devices, etc.) in the face of increased load. Scalability of distributed systems is usually implemented in two ways: horizontal scaling and vertical scaling. Horizontal scaling is achieved by adding more compute nodes to expand the system, usually through clustering or distributed computing frameworks; vertical scaling is achieved by increasing the resources of individual nodes, such as CPU, memory, storage, etc. to expand the system. The implementation of scalability not only depends on the increase of hardware resources, but is also affected by the system architecture design, protocol selection, load balancing and other factors. With the increasing scale of distributed systems, how to design efficient scaling strategies has become an important topic in distributed system research. A reasonable scaling mechanism can ensure system performance while avoiding system collapse caused by resource bottlenecks or scalability limitations.

4.2 Horizontal and Vertical Scaling

Horizontal scaling and vertical scaling are two basic ways to scale a distributed system. Horizontal scaling (also known as horizontal scaling) refers to expanding the processing power of a system by adding more computing nodes or servers. This type of scaling can achieve performance improvements through distributed computing and storage, and the processing power of the system can increase almost linearly as the number of nodes increases. Horizontal scaling is common in distributed databases, cloud computing platforms, and other fields. Vertical scaling (also known as vertical scaling) is to expand the capacity of the system by increasing the resources of a single node, which is usually manifested in upgrading the server's CPU, memory, disk and other hardware resources. Vertical scaling is suitable for scenarios where the performance of a single node needs to be improved, but its scalability is limited by the maximum configuration of the hardware. In contrast, horizontal scaling is more flexible and scalable, and can better cope with the increasing system load.

4.3 Scalability Bottlenecks and Challenges

During the scaling process of distributed systems, several bottlenecks and challenges are usually encountered. The first one is the data consistency and coordination problem; as the number of nodes increases, ensuring data consistency across nodes becomes more complex. How to maintain consistency and avoid data conflicts through effective protocols and algorithms is an urgent problem in distributed systems. Second, network bandwidth and latency are also one of the major challenges to scalability. As the number of nodes increases, the communication load between nodes also increases, and network latency and bandwidth constraints may lead to system performance degradation. In addition, issues such as load balancing, fault recovery, and dynamic joining and exiting of nodes in distributed systems may also affect the scalability of the system. In order to overcome these bottlenecks, these challenges need to be fully considered during system design and appropriate optimisation strategies need to be adopted.

4.4 Scalability Design Strategies

Designing a scalable distributed system requires full consideration of several aspects of the system, such as load balancing, fault tolerance, and data consistency. In terms of load balancing, the system needs to dynamically adjust the task allocation according to the computing and storage capacity of each node to ensure the optimal use of resources. In terms of fault tolerance, redundant backup mechanisms are designed to ensure that even if some nodes fail, the system can continue to run without affecting the user experience. In terms of data consistency, appropriate consistency models (e.g., eventual consistency) and distributed transaction protocols are used to ensure data correctness and integrity. In addition, more flexible scaling strategies can be achieved through horizontal scaling, which allows nodes to dynamically join or exit to cope with fluctuations in system load.

5. Conclusion

This paper provides an in-depth analysis of the basic theory, design architecture, scalability and performance optimisation strategies of distributed systems. It is shown that distributed systems have significant advantages in handling large-scale data and highly concurrent tasks, but also face challenges such as data consistency, load balancing, and system scalability. Through reasonable architectural design, protocol selection, and scaling strategies, distributed systems can effectively cope with these problems and provide efficient and reliable services.

Reference

[1] ALBERTO M, GUGLIELMO F, FEDERICO T. 5G-Enabled PMU-Based Distributed Measurement Systems: Network Infrastructure Optimization and Scalability Analysis [J]. IEEE Transactions on Instrumentation and Measurement, 2024.
[2] LIMA H D, LIMA L A D P, CALSAVARA A, et al. Beyond scalability: Swarm intelligence affected by magnetic fields in distributed tuple spaces [J]. Journal of Parallel and Distributed Computing, 2019.
[3] YEO S, BAE M, JEONG M, et al. Crossover-SGD: A gossip-based communication in distributed deep learning for alleviating large mini-batch problem and enhancing scalability [J]. Concurrency and Computation: Practice and Experience, 2022.
[4] MAGHSOUDLOO M, KHOSHAVI N. Elastic HDFS: interconnected distributed architecture for availability–scalability enhancement of large-scale cloud storages [J]. The Journal of Supercomputing, 2019.
[5] LIU Y, REN N, OU J. Hydrodynamic analysis of a hybrid modular floating structure system and its expansibility [J]. Ships and Offshore Structures, 2021.
[6] ZHAO H, ZHANG X, WANG Y, et al. Improving the Scalability of Distributed Network Emulations: An Algorithmic Perspective [J]. IEEE Transactions on Network and Service Management, 2023.