Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2020.010405
ISSN 2790-0916 Vol. 1, Issue 4: 33-40

State Consistency Algorithm for Peer to Peer Distributed Systems Based on Data Mining

Seyyed Babak Alavi*

Univ Cincinnati, Cincinnati, OH 45221 USA

**corresponding author*

Keywords: Data Mining, Distributed System, State Consistency, Vector Clock

Abstract: In recent years, with the continuous development of network and information technology, the distributed system has replaced the centralized architecture and become the focus of researchers. Consistency is a basic and complex problem in the implementation of distributed systems. This paper mainly studies the state consistency algorithm of peer-to-peer distributed system based on DM(DM). This paper first summarizes the types of DM algorithms. In order to achieve high throughput, low latency and high availability at the same time as possible, this paper designs a VC algorithm for state machine replication, and this paper makes a quantifiable performance analysis of the consensus protocol from the four aspects of cluster fault tolerance, communication cost and election cost.

1. Introduction

In recent years, the coordinated control problem of multi-agent systems has been widely concerned by relevant researchers. With the development of science and technology and the gradual advancement of social demand, the scale of control system is getting larger and the complexity of control system is getting higher and higher. Under such a trend, the traditional centralized control method is increasingly difficult to meet the needs of control tasks under the new situation [1]. Centralized control relies on the communication between the central node and all other nodes in the network. When the number of nodes in the network is large or the data to be communicated is large, the communication cost of the network and the calculation burden of the central node will be greatly increased. Moreover, the robustness of centralized control is not ideal. In order to improve the above problems in the centralized control system, researchers have proposed the distributed control method, and carried out in-depth and extensive research. Unlike centralized control, distributed control does not require a central node (central controller). The distributed controller designed for each individual only needs to use the information of the individual itself and its neighbor agents with communication connection, that is, the control only uses the local information without the global information of the whole multi-agent system. The tasks of distributed

coordination control of multi-agent system usually include consistency, aggregation, formation and coverage [2-3]. Compared with centralized control, distributed control has the advantages of low communication cost, good scalability and strong robustness. Based on the above advantages, under the background of the rapid development of network computer technology, the distributed coordinated control technology of multi-agent system has been widely used in many fields such as multi robot system collaborative assembly, sensor network source point positioning and multi aircraft / multi mobile robot formation [4].

Consistency is a basic problem in the field of distributed coordinated control of multi-agent systems [5]. Consistency means that each agent can reach agreement on a certain state variable through mutual coordination by designing a reasonable communication protocol, such as position, speed and attitude [6]. In fact, aggregation, formation and coverage can be seen as an extension of the consistency problem. Researchers have invested a lot of time and energy in the unification of multi-agent systems and have achieved a wealth of research results. A scholar has established the consistency control design scheme of the first-order integrator multi-agent system [7]. For continuous time and discrete-time first-order multi-agent systems with different topologies, a team studied the consistency control design problem [8]. Some experts designed a consistency protocol for the second-order integrator multi-agent system under the condition that the relative speed cannot be measured directly and the output is saturated [9]. In addition to the above theoretical research results, researchers have also made fruitful research results in the application design of multi-agent systems. Typical applications include multi Euler Lagrangian systems (such as robotic arms, aircraft, underwater robots), multi nonholonomic systems (such as multi mobile robots) [10].

Although the consistency control based on distributed event triggering has made great progress, there is little research on the state consistency algorithm of peer-to-peer distributed systems using DM.

2. DM Distributed System State Consistency Algorithm

2.1. DM Algorithm Type

Currently, large DM technology includes many different types of mining algorithms, including sorting algorithm, clustering algorithm, the correlation rule algorithm, etc. In practical applications, the selection and use of specific algorithms is mainly determined by the objective objectives to achieve the predefined data analysis and mining results [11-12].

(1) Classification algorithm

Sorting algorithm is a technology that can find the correlation and difference between a large number of sample data by sorting the data set, so that the value of the data can be deeply extracted [13]. Typical classification algorithms mainly include: decision tree algorithm, Bayesian algorithm, rough set algorithm and fuzzy logic algorithm [14].

(2) Clustering algorithm

This algorithm refers to the algorithmic process of grouping similar and similar data objects into a large number of data information, so that similar data information is collected and grouped for DM and calculation [15]. Currently, the clustering algorithm has been applied in many fields such as society, sports, education and so on. For example, in social economy, data such as national income, industrial and agricultural output value and per capita consumption can be better clustered through clustering algorithm to facilitate the evaluation of the overall national economic strength; In physical education, by clustering the height, strength, speed, vital capacity and other physical fitness and physiological indicators of athletes, it can better teach the athletes according to their

aptitude, develop their strengths and avoid their weaknesses, and give better play to the personal potential of athletes. At present, the classical clustering algorithms mainly include K-means, expectation maximization algorithm (EM), single link, complete link, average link and so on [16].

(3) Association rule algorithm

The correlation rule algorithm is mainly used to objectively reflect the internal relationship between a large number of data through a structured mathematical model. Because large data has the characteristics of massive, high-dimensional, heterogeneous, dynamic, spatio-time, diversity, multipurpose, multi-scale and unclear, the inherent correlation between the data is very hidden.

At present, the basic algorithms of association rules mainly include: Apriori algorithm based on Apriori property to generate candidate sets, FP growth algorithm without generating candidate sets, etc.

(4) Evolutionary analysis algorithm

The algorithm mainly mines and analyzes the existing data by means of mathematical modeling, so as to analyze and describe the changes and trends of future data objects, and can be used to guide and predict future decision-making behavior. At present, the evolution analysis algorithm has also been widely used in the big DM analysis of various industries. For example, the enterprise can also use the evolution analysis algorithm to predict the sales volume of its products in the next year, and formulate the purchase plan and production schedule of raw materials, adjust the inventory strategy and reasonably allocate sales tasks according to the prediction results [18].

2.2. Distributed Consistency Algorithm

Distributed systems put forward three main requirements for distributed consistency algorithms: replication in distributed systems has high throughput; Replication across data centers has low latency; The service has high availability.

In order to achieve high throughput, low latency and high availability at the same time as possible, we designed a VC algorithm for state machine replication. VC does not need a leader. All replicas can concurrently propose operation commands at any time. Under normal circumstances, each operation command can be submitted after one round-trip network message communication with most replicas. VC separates the consensus protocol from the ordering and execution of operation commands. All copies can submit operation commands concurrently at any time without determining the order of operation commands. However, vector clock is used to track and record the dependency between them. The subsequent playback stage sorts the submitted operation commands according to dependencies and executes them in order.

Vector clock algorithm is a kind of algorithm used in distributed system to generate partial order value for each event. The vector clock marks an n-dimensional vector clock on each node, so as to distinguish the causal relationship between the states of each node and to ensure the order of operation to a certain extent.

For a distributed system with n nodes, if VI [k] is defined as the time value of node K known to node i, then the vector clock held by node I is:

$$V_i = (V_i[1], V_i[2], ..., V_i[n],), i \in (0,1,2,..., n-1) \quad (1)$$

In the system initialization phase, the vector clocks of all nodes are initialized as:

$$V_i = (0,0,...,0), i \in (0,1,2,..., n-1) \quad (2)$$

When the state of node I is updated, the vector clock value on node I is updated:

$$V_i[i] = V_i[i] + d(d > 0) \tag{3}$$

When node i sends a message to node j, it carries its own vector clock VI. node j will compare and update the received vector clock of node i with its own vector clock:

$$V_i[k] = \max\{V_i[k], V_j[k]\}, k \in (0,1,2,...,n-1) \tag{4}$$

Let the vector clocks of nodes I and j in the system be VI and VJ respectively. If there is VI [k] $\leqslant$ VJ [k] for $\forall$ K $\in$ (0, 1, 2,..., N - 1), then VI and VJ are said to satisfy the causal relationship VI $\rightarrow$ VJ, representing that the state of node j is newer than that of node I. The vector clock algorithm records the sequence of each operation in the distributed system, and then can judge the causal relationship of each node state. The vector clock is used in the distributed consistency algorithm, which can record the order of submitting the operation commands of each replica, and distinguish the causal dependency between the operation commands. Each replica concurrently submits operation commands in disorder, and then executes them according to the causal dependency between them during playback to ensure the consistency of the status of each replica.

The procedure for executing an VC presence is approximately the following. Each copy retains a vector clock. When a copy starts a new presence, it is the only supporter in the new presence. It can skip the preparation phase in Paxos and run the acceptance phase directly. The copy increases its vector clock, and then sends the raised vector clock to at least most of the copies in the suggested message. When the copy receives the proposal message, it updates its own vector clock with the received vector clock, and then, responds to the rapporteur with the updated vector clock in the acceptance message. When the rapporteur receives the download message, he updates his vector clock with the received vector clock. After the proposer receives the acceptance message from most copies, submits the corresponding presence using the current vector clock and then sends the vector clock along with the commit message to all other copies. When the copy receives the commit message, it updates the vector clock with the received vector clock, and then submits the corresponding presence. When a copy is submitted to a presence, will start a playback phase to sort the operation commands generated by the presence and other relevant operation commands according to the dependency recorded by the vector watch and execute them in turn.

3. Simulation Experiment

In order to restore the application scenario that the replica state machine is consistent to a certain extent, the cluster server node is set in the experiment, and the client node that sends requests to the server is also set outside. After starting the operation, the client node will send a request to the cluster node; If the node receiving the request is the cluster leader, it will directly initiate the log resolution; Otherwise, it will reply a message to the client and carry the node ID of the current cluster leader. After receiving this message, the client will resend the request to the cluster leader according to the node ID. This indicates that the client node should know the node ID of the cluster node and the network node. They also need to initially configure the necessary cluster related information before operation.

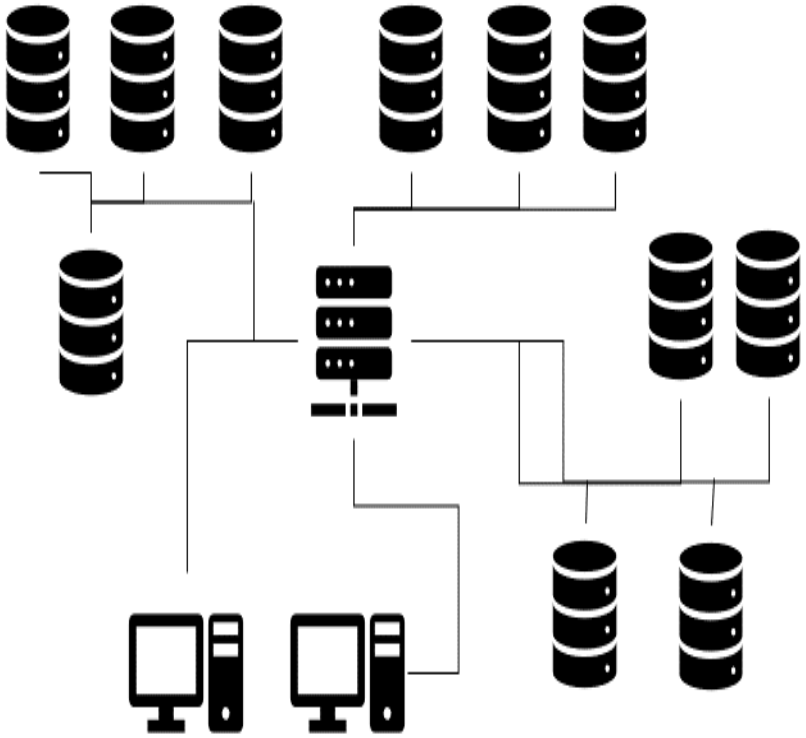

Figure 1. Experimental network, server and client

This paper uses multiple server nodes on the same network segment to construct a distributed experimental environment. In order to make the experimental results more general and test whether the performance of the consensus protocol will be affected by the number of cluster nodes, we conducted experiments on a cluster composed of 5 nodes, 7 nodes and 11 nodes. The cluster architecture of the experiment is shown in Figure 1. At the same time, on the network transmission protocol, we choose to implement the communication between nodes based on TCP protocol; As a widely used communication protocol, TCP has many mature communication modules that can complete the packaging and parsing of messages.

Table 1. Cluster node configuration used in the experiment

Configuration items	Configuration description
CPU	Intel Core i5-9400
Memory size	8GB
Operating system	CentOS 7.0

4. Experimental Result

In the experiment, this paper makes a quantifiable performance analysis of the consensus protocol from three aspects: cluster fault tolerance, communication cost and election cost. The experimental results are as follows.

4.1. Cluster Fault Tolerance

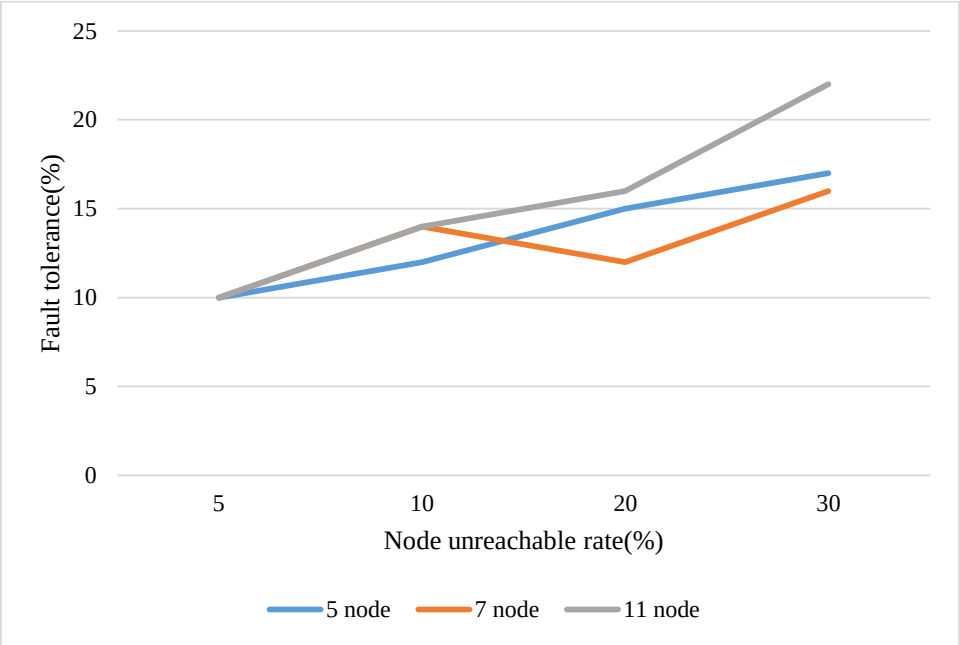

Figure 2. Variation of cluster fault tolerance with unreachable rate

As shown in Figure2, the more the number of nodes in the cluster, the smaller the impact of the unreachable rate y on the cluster, and the cluster with more nodes reflects a higher cluster fault tolerance.

4.2. Communication Cost

Table 2. Communication cost comparison

	5 node	7 node	11 node
VC	6.7	6.1	5.9
Paxos	7.8	8.2	7.7

As shown in Table 2, for the two algorithms in clusters with different node numbers, the communication cost difference of the consensus protocol is small and the value is relatively stable.

4.3. Election Costs

As shown in Figure 3, the leader election process of Paxos may have a greater impact on the services provided by the cluster. Whenever there is a change of leader in the cluster, Paxos will not be able to respond to the request message of the external client for a longer period of time, and the election cost will be higher.

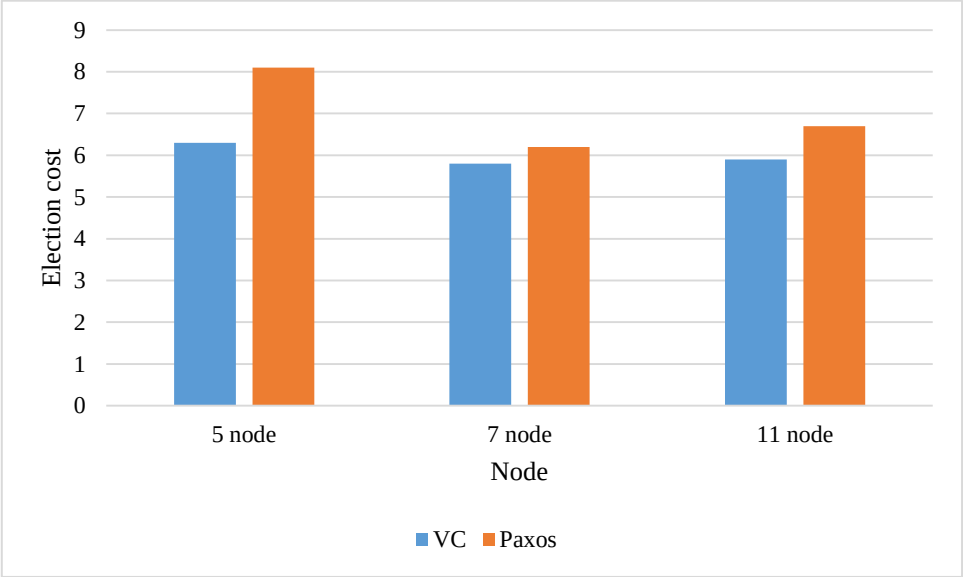

Figure 3. Comparison of election costs between the two agreements

5. Conclusion

In this paper, based on the consensus of replica state machine in distributed environment, the distributed consistency algorithm is discussed and studied. A new consensus protocol VC is designed and implemented. The new protocol VC is designed to ensure the consistency of the replica state machine in the distributed environment with high network and node failure rate. In order to explain the detailed design and the principle of VC protocol in detail, this paper introduces VC protocol in many aspects. The composition of the distributed cluster can be complex and diverse. Although the existing engineering implementations are based on the design premise of low failure rate of the network and nodes, the application scenarios with high failure rate in the future are not excluded. We hope that VC's protocol design can perform better in the distributed environment with high failure rate after gradual improvement.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Baresi L, Ghezzi C, Ma X, et al. Efficient Dynamic Updates of Distributed Components

Through Version Consistency. IEEE Transactions on Software Engineering, 2017, 43(4):340-358.

[2] Alghamdi M I, Alghamdi M, Almushilah M, et al. Recovery Support for Real-time Distributed Editing Systems. Journal of Internet Technology, 2020, 19(4):1119-1129.

[3] Katsaros D. Distributed ledger technology: the science of the blockchain (2nd ed.). Computing reviews, 2018, 59(11):596-597.

[4] Bello A U, Nnakwe M O. An asynchronous inertial algorithm for solving convex feasibility problems with strict pseudo-contractions in Hilbert spaces. Proceedings of the Edinburgh Mathematical Society, 2020, 65(1):229-243.

[5] Nikitin V, Andrade V D, Slyamov A, et al. Distributed Optimization for Nonrigid Nano-Tomography. IEEE Transactions on Computational Imaging, 2020, PP(99):1-1.

[6] Alghamdi M I, Jiang X, Zhang J, et al. Recovery support for real-time distributed editing systems. Journal of Internet Technology, 2018, 19(4):1119-1129.

[7] Hsu T Y, Kshemkalyani A, Shen M. Causal consistency algorithms for partially replicated and fully replicated systems. Future Generation Computer Systems, 2017, 86(SEP.):1118-1133.

[8] Bouyakhf E H, Hammoujan S, Benelallam I. Dynamic vs. static agent ordering in distributed arc consistency. International Journal of Advanced Intelligence Paradigms, 2018, 10(3):266.

[9] [1]Beck, Christopher J. [Lecture Notes in Computer Science] Principles and Practice of Constraint Programming Volume 10416 || Arc Consistency via Linear Programming. 2017, 10.1007/978-3-319-66158-2(Chapter 8):114-128.

[10] Yan P, Choudhury S, Wei R. A Machine Learning Auxiliary Approach for the Distributed Dense RFID Readers Arrangement Algorithm. IEEE Access, 2020, PP(99):1-1.

[11] Alanazi E. Arc Consistency for Constrained Lexicographic Preference Trees. IEEE Access, 2020, PP(99):1-1.

[12] Parise F, Gentile B, Lygeros J. A distributed algorithm for average aggregative games with coupling constraints. IEEE Transactions on Control of Network Systems, 2020, 7(2):770-782.

[13] Arleo A, Didimo W, Liotta G, et al. A Distributed Multilevel Force-Directed Algorithm. IEEE Transactions on Parallel & Distributed Systems, 2019, 30(4):754-765.

[14] Ferrer M, Gonzalez A, Diego M D, et al. Distributed Affine Projection Algorithm Over Acoustically Coupled Sensor Networks. IEEE Transactions on Signal Processing, 2017, 65(24):6423-6434.

[15] Jose L, Ibanez S, Alizadeh M, et al. A Distributed Algorithm to Calculate Max-Min Fair Rates Without Per-Flow State. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2019, 3(2):1-42.

[16] Iranpour E, Sharifian S. A distributed load balancing and admission control algorithm based on Fuzzy type-2 and Game theory for large-scale SaaS cloud architectures. Future Generation Computer Systems, 2018, 86(SEP.):81-98.

[17] Seshadri K, Mercy S S, Manohar S. A distributed parallel algorithm for inferring hierarchical groups from large-scale text corpuses. Concurrency, practice and experience, 2018, 30(11):1-18.

[18] Zayyani H, Sari R, Korki M. A Distributed One-bit Compressed Sensing Algorithm for Nonlinear Sensors with a Cramer-Rao Bound. IEEE Communications Letters, 2017, PP(99):1-1.