Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2022.030208
ISSN 2790-0916 Vol. 3, Issue 2: 95-102

Performance Optimization Technology of Fault Tolerance Mechanism in Distributed System Based on Neural Network

Vemparty Velmurungan*

Vellore Institute of Technology, India

**corresponding author*

Keywords: Neural Network, Distributed System, Fault Tolerance Mechanism, Data Restoration

Abstract: In the era of big data, distributed systems(DS) have to deal with more and more data, and the possibility of system hardware failure is higher and higher. Traditional disk arrays can no longer meet the high reliability requirements of large-scale distributed clusters. Data fault tolerance has become an important problem in DSs. In order to ensure the reliability of data processing in DSs and continue to provide users with high-quality services, the performance optimization of fault-tolerant mechanisms has become an important research content in DSs. Aiming at the distributed core storage framework proposed in this paper, the fault-tolerant mechanism of its client module and metadata storage management module is designed and implemented, and the performance of the DS is tested through the performance optimization experiment of fault-tolerant mechanism. The experimental results of Prime-based and neural network-based data inpainting show that the neural network-based data inpainting technology can reduce the network cost of data inpainting. The experimental results of distributed computing delay based on radial basis neural network algorithm, C.T. algorithm and Pedone algorithm show that the delay time of RBF algorithm tends to converge with the increase of MRR.

1. Introduction

In order to improve the reliability of the DS in the process of data processing, it must adopt a certain data fault tolerance technology. Data fault tolerance technology processes the original data to generate redundancy, and places the data on different racks or nodes. These redundant data can directly or indirectly generate the original data and solve the problem of data transmission or storage. Loss problem, in order to ensure the stable operation of the system [1-2].

At present, many scholars have studied the fault-tolerant mechanism of DSs, and have achieved

good results. For example, a DS designed by a scholar adopts the Spark fault-tolerant technology, which is distributed in the system servers, and signals are continuously sent between the servers. When a server fails and another server cannot receive the signal sent by the other party, it knows that the other server has a problem, and automatically runs to the software switching function [3]. A scholar uses the recovery of distributed snapshots. After a fault occurs, the machines in the DS need to retrieve the state from the snapshots stored in the distributed file system. In order to reduce the impact of recorded snapshots on normal execution, the system will provide many optimizations., and this goal is contradictory to fast recovery. The more obvious the optimization effect is, the longer the corresponding recovery time may be [4]. Some studies have proposed to evaluate the total amount of data sent by the DS from the data sender to the data receiver through the network bandwidth cost measurement. Although it can reflect the network occupancy during data repair to a certain extent, it only considers the total amount of data during data repair. The actual network topology is not considered, and the data transmission in the distributed storage system may pass through several layers of switches [5]. From this point of view, the fault-tolerant mechanisms of DSs have their own characteristics and can repair DS failures.

This paper proposes the radial basis neural network algorithm model, and applies the algorithm to the data restoration of DSs; then introduces two fault-tolerant technologies; then designs the distributed core storage system framework, and tests the client module file upload and download performance; finally, the fault-tolerant optimization experiment of DS performance is carried out, which shows the superiority of neural network algorithm.

2. Related Algorithms and Technologies

2.1. Neural Network

The simplest radial basis neural network consists of three levels that act very differently. The first layer is the input layer, which uses input nodes to connect the neural network and the external environment, and the second layer is the hidden layer, which performs nonlinear transformations from input points to hidden points. The third layer is the output layer, which is the sum of the linear weights output by the hidden units [6]. Due to the short training time of RBF, the DS is repaired by using RBF to meet the efficient data processing process of the system.

The RBF function generally uses a Gaussian function:

$$G_i = \exp\left(-\frac{\left\|x - v_i\right\|^2}{\varepsilon_i^2}\right) \tag{1}$$

Among them, G_i is the output of the hidden unit, x is the input value, v_i and ε_i are the center and width of the ith basis function.

The output of the RBF neural network is:

$$y_i = \sum_j^k w_{ij} G_j \tag{2}$$

y_i is the RBF output value, and w_{ij} is the connection weight from the i-th layer to the j-th layer.

2.2. Fault-Tolerant Technology

(1) Fault-tolerant technology of database

Because the database needs to provide reliable data services, it also needs to provide fault tolerance support. Database systems need to support complex transaction models, so the main challenge of its fault tolerance mechanism is to ensure data consistency without affecting high concurrency [7].

SiloR is a fault-tolerant database developed on the basis of the in-memory database Silo. Silo is a well-known in-memory database that utilizes all hardware resources to provide high throughput. SiloR provides complete and efficient fault tolerance support on Silo. The recovery mechanism of SiloR has little impact on the speed of database processing transactions, and can ensure fast recovery [8].

(2) Copy-based fault tolerance technology

It is also a common fault tolerance technology to improve the reliability of DSs by creating replicas. A system with replicas can continue to provide services through replicas when the primary node fails [9]. However, such systems generally also need to design a mechanism to ensure the consistency between the replica node and the master node. The Replicated State Machine Model (RSM) is one approach to this problem [10]. In the RSM model, the commands accepted by the system are deterministic, and the primary server has the same initial state as the backup server. During execution, the primary server accepts and executes commands, and it also sends commands to the backup server. When the backup server completes the command from the primary server it notifies the primary server. Only then will the master server consider the command complete. Under this model, we can guarantee that the state of the replica server is consistent with that of the master server [11-12].

3. Application of Fault Tolerance Mechanism of DS

3.1. Framework Design of Distributed Core Storage System

The core storage module itself is a distributed storage system. In order to achieve high storage efficiency, the system adopts strategies. When it is necessary to locate file metadata or data, it is necessary to obtain the corresponding locating rules from the rule server [13]. As can be seen from Figure 1, the system includes a total of six modules.

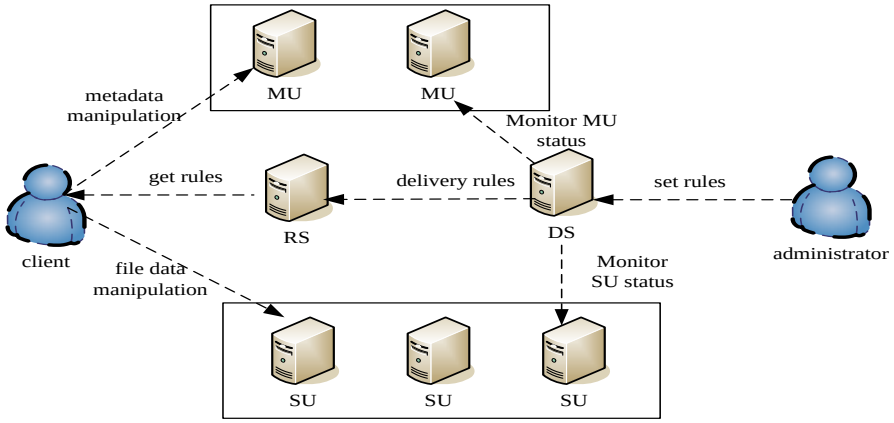

Figure 1. Distributed core storage module framework

Client module (client): Responsible for interacting with users, mainly initiating relevant operation commands to the background core storage in the virtual disk, such as uploading files, downloading files, etc.

Metadata storage management module (MU): responsible for managing the metadata information of files and user information, such as file size, file name, file creation time and file path or user, root directory and files contained in each user number [14].

Data storage management module (SU): responsible for storing data information of files.

Configuration management module (DS): responsible for transmitting rule information to RS, monitoring the activity status information of SU and MU according to WD on each SU and MU, and feeding back the obtained status information to Administrator; executing commands sent by Administrator, such as Start SU or MU.

Administrator module (Administrator): manages the configuration and creation rules of the entire system, mainly interacting with the DS module.

Rule Module (RS): Responsible for storing the rule information of the system, which is used to locate the location of file metadata and the location of data.

3.2. Design and Implementation of Fault Tolerance Mechanism for Clients

In the design of the distributed core storage system structure, the client will use the data rule table and metadata rule table whether uploading or downloading files [15]. Then, according to the two rule tables, determine the SU or MU that finally interacts with the client. Because the rule table will be reused, when the Client logs in, it will actively pull the rule table from the RS, and then cache the two rule tables locally. When the rule table stored by the RS is inconsistent with the rule table cached by the Client, a corresponding strategy must be adopted to solve the problem of inconsistent rule tables. To solve this problem, the Lazy mechanism can be used [16-17].

A rule table that can tolerate client-side caching in a DS is out of date for a certain period of time. However, this inconsistency will eventually be synchronized by the Lazy mechanism to ensure that the rule table cached by the client is up-to-date, and the client can still provide services to ensure consistency, and make the rules stored in the system consistent [18].

3.3. Design and Implementation of Fault Tolerance Mechanism of MU Module

The MU module (metadata storage) management module mainly stores the metadata information of the file. In order to ensure that the metadata information of the file is never lost and the server runs uninterrupted, dual-machine fault tolerance is mainly used [19]. In the MU management module, the dual-system hot backup mode is adopted. The main reason is that the dual-machine hot backup mode can not only meet the fault tolerance requirements of the system, but also is relatively simple to implement. Dual-system hot backup fault tolerance is implemented in the master-slave configuration management module of the MU. The master-slave configuration module is responsible for the establishment and maintenance of the master-slave relationship between a pair of MUs, as well as the master-slave switchover when a fault occurs [20-21].

4. DS Fault Tolerance Mechanism Performance Optimization Test

4.1. Upload and Download Stress Performance Evaluation Test

Test method: The load generator continuously initiates 100, 300, 800, 1000, 2000 connection

requests to upload and download files, where the file size is 5M, and then use the nmon tool to observe and record the write speed of the SU and the usage of the CPU.

Table 1. Upload file performance test results

Number of files	100	300	800	1000	2000
Occupied CPU (%)	12.7	13.3	13.8	14.5	14.7
Write speed (kb/s)	2034	3867	6525	7794	13451

According to Table 1, the increase in the number of uploaded files has little effect on the CPU usage. This happens because the CPU does not need to be involved in the reading and writing process of the underlying disk. And the write speed of the disk increases with the number of files. According to these two performance parameters, the design of SU meets the requirements.

Table 2. Download file performance test results

Number of files	100	300	800	1000	2000
Occupied CPU (%)	15.8	18.3	17.6	16.4	16.2
Read speed (kb/s)	376	749	1233	1153	964

According to Table 1 and Table 2, the CPU occupied by the downloaded file is higher than the CPU occupied by the uploaded file, because the user will use the computer's own data caching mechanism when reading the file. CPU-intensive. The read speed of the disk increases first and then decreases with the increase of the number of files, which is also due to the mechanism of data caching.

4.2. Network Cost Optimization

In order to reduce data repair operations during client read operations, there is a periodic scanner in the DS, which periodically scans the status of each data block and encoding block of each file. Assuming that a disk fails, it is necessary to select the recipient node clockwise along the hash ring and repair the lost data block or encoded block. Data repair not only consumes CPU resources, but also consumes a lot of network bandwidth. Therefore, certain strategies need to be adopted to reduce the network cost.

To compare the Prime-based data inpainting technique with the neural network-based data inpainting technique, we simulated a large-scale distributed cluster. The cluster consists of 5 racks with 8 nodes in each rack and 4 disks per node. First, one million objects are written into the distributed cluster using data layout rules. The size of a single object can be used as an encoding group, and the size of the encoding block in the encoding group is 1MB. Then, 10,000 objects are selected as faulty objects, and each object randomly selects a coding block that is lost and needs to be recovered. Finally, the objects are recovered with Prime repair technology and neural network-based data repair technology, respectively.

The DS can be customized by the system administrator for the length of the coding stripe and the length of the data stripe. Therefore, we compared four erasure coding data coding schemes and used the average network cost of repairing a single object as the evaluation standard. The experimental

results are shown in Figure 2 shown. In this simulated experimental environment, applying erasure coding scheme 1, the data blocks and coding blocks will be distributed in different racks according to the data layout algorithm, so the average network cost of Prime-based data repair and neural network-based data repair is the same. Besides, in the case of erasure codes 2, 3, and 4, the average network cost of neural network-based data inpainting is lower than that of Prime-based data inpainting. It can be seen that the data repair algorithm based on neural network can significantly reduce the network cost of data repair in complex network environment.

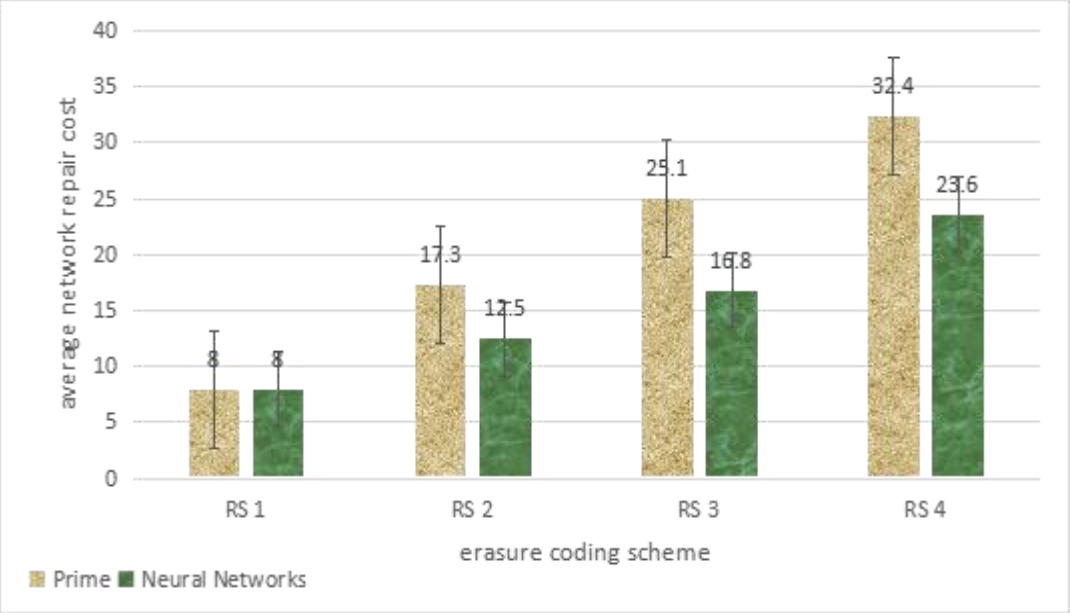

Figure 2. Comparison of average network cost based on Prime repair and neural network repair

4.3. Latency Performance Optimization

In the experiment, when evaluating the performance of the RBF algorithm, the C.T. algorithm and the Pedone algorithm are selected as the reference algorithms. In addition, the experiment introduces a message rescheduling mechanism (Message Reordering), which disrupts the order of messages in the message queue received by the process according to a certain probability (MRR, Message Reordering Rate).

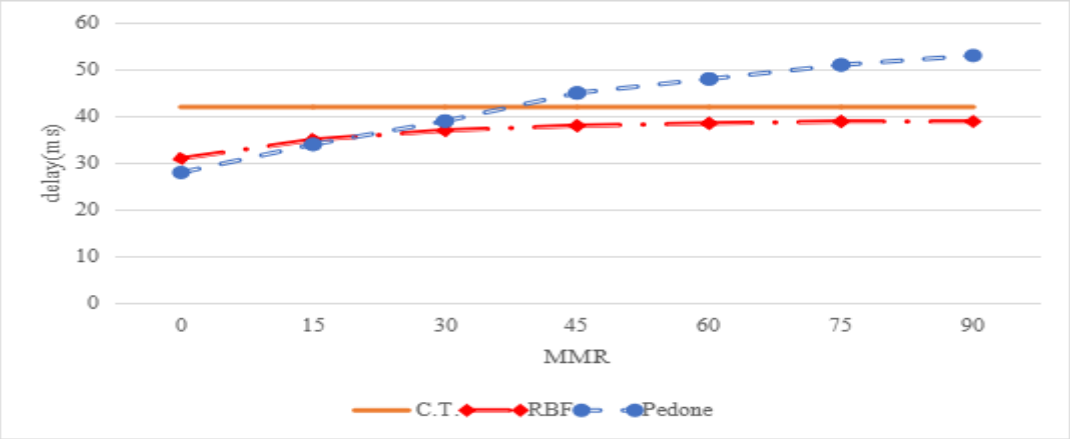

Figure 3. Latency versus MRR

Figure 3 shows the distributed computing delay time of the RBF algorithm, the C.T. algorithm and the Pedone algorithm at different MRR values (the system load is fixed at 400/s). The test results show that the MRR has basically no effect on the C.T. algorithm, but the RBF algorithm and the The delay time of the Pedone algorithm increases with the increase of MRR, but the delay time of the RBF algorithm will converge and will not exceed the C.T. algorithm, which is consistent with the expected results.

5. Conclusion

In this paper, by comparing the data optimization results of DSs based on Prime and neural network, as well as the optimization results of distributed computing delay time based on RBF algorithm, C.T. algorithm and Pedone algorithm, it is found that neural network technology can reduce the network cost of data repair and convergence. The distributed computing delay time ensures the reliability of the DS in the data processing process, and verifies the effective effect of the neural network technology on the fault-tolerant mechanism of the DS.

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] Mehalaine R, Boutekkouk F. A New Intelligent Biologically-Inspired Model for Fault Tolerance in Distributed Embedded Systems. International Journal of Embedded and Real-Time Communication Systems, 2020, 11(3):22-47.

[2] Kasu P, Hamandawana P, Chung T S. DLFT: Data and Layout Aware Fault Tolerance Framework for Big Data Transfer Systems. IEEE Access, 2021, PP(99):1-1.

[3] Kada B, Kalla H. A Fault-Tolerant Scheduling Algorithm Based on Checkpointing and Redundancy for Distributed Real-Time Systems. International journal of DSs and technologies, 2019, 10(3):58-75.

[4] Afshari A, Karrari M, Baghaee H R, et al. Distributed Fault-Tolerant Voltage/Frequency Synchronization in Autonomous AC Microgrids. IEEE Transactions on Power Systems, 2020, 35(5):3774-3789.

[5] Saraswat B K, Suryavanshi R, Yadav D S. A Comparative Study Of Checkpointing Algorithms For Dss. International Journal of Pure and Applied Mathematics, 2019, 118(20):1595-1603.

[6] Moraes M, Gradvohl A. Evaluating the impact of a coordinated checkpointing in distributed data streams processing systems using discrete event simulation. Revista Brasileira de Computa çõo Aplicada, 2020, 12(2):16-27.

[7] Bintoudi A, Zyglakis L, Tsolakis A C, et al. Hybrid multi-agent-based adaptive control scheme for AC microgrids with increased fault-tolerance needs. IET Renewable Power Generation, 2020, 14(1):13-26.

[8] Ekwonwune E N, Ezeoha B U. Scalable Distributed File Sharing System: A Robust Strategy for a Reliable Networked Environment in Tertiary Institutions. International Journal of Communications, Network and System Sciences, 2019, 12(4):49-58.

[9] Youness H, Omar A, Moness M. An Optimized Weighted Average Makespan in Fault-Tolerant Heterogeneous MPSoCs. IEEE Transactions on Parallel and DSs, 2021, PP(99):1-1.

[10] Jeong E, Jeong D, Ha S. Dataflow Model–based Software Synthesis Framework for Parallel and Distributed Embedded Systems. ACM Transactions on Design Automation of Electronic Systems, 2021, 26(5):1-38.

[11] Pashkov V N. Fault-Tolerance Distributed Control Plane for Software Defined Networks. Modeling and Analysis of Information Systems, 2019, 26(1):101-121.

[12] Ali M, Bagchi S. Probabilistic normed load monitoring in large scale DSs using mobile agents. Future Generation Computer Systems, 2019, 96(JUL.):148-167.

[13] Roque A, Jazdi N, Freitas E, et al. A Fault Modeling Based Runtime Diagnostic Mechanism for Vehicular Distributed Control Systems. IEEE Transactions on Intelligent Transportation Systems, 2021, PP(99):1-13.

[14] Rajabi A, Bobba R B. Resilience Against Data Manipulation in Distributed Synchrophasor-Based Mode Estimation. IEEE Transactions on Smart Grid, 2021, PP(99):1-1.

[15] Alvarez I, Ballesteros A, Barranco M, et al. Fault Tolerance in Highly Reliable Ethernet-Based Industrial Systems. Proceedings of the IEEE, 2019, 107(6):977-1010.

[16] Khaldi M, Rebbah M, Meftah B, et al. Fault tolerance for a scientific workflow system in a Cloud computing environment. International Journal of Computers and Applications, 2019, 42(3):1-10.

[17] Singh J, Kaur K. Dynamic Fault Tolerance Job Allocation Mechanism to Conserve Resources in Vehicular Cloud. International Journal Of Computer Sciences And Engineering, 2019, 7(5):538-547.

[18] Rosato A, Panella M, Araneo R, et al. A Neural Network Based Prediction System of Distributed Generation for the Management of Microgrids. IEEE Transactions on Industry Applications, 2019, PP(99):1-1.

[19] Rajkumar R, Sudhamani M V. Image Retrieval System using Residual Neural Network in a Distributed Environment. International Journal of Recent Technology and Engineering, 2020, 8(6):2277-3878.

[20] Bui V H, Hussain A, Kim H M. Double Deep Q-Learning-Based Distributed Operation of Battery Energy Storage System Considering Uncertainties. IEEE Transactions on Smart Grid, 2019, PP(99):1-1.

[21] Hashida S, Tamura K, Sakai T. Classifying Tweets using Convolutional Neural Networks with Multi-Channel Distributed Representation. IAENG Internaitonal journal of computer science, 2019, 46(1):68-75.