Application of Autonomous Monitoring Method Based on Distributed Environment Deployment in Network Fault

Xiaolian Li*, Kunying Li¹, Yu Ding¹, Daiming Wei¹, Xianying Ma¹

¹Computer Application Technology Department, PetroChina Research Institute of Petroleum Exploration & Development, Beijing 100083, China
*smileli@petrochina.com.cn

Abstract. With the rapid development of information technology and the increasing demand for computing, the scale of cloud environment and distributed deployment is becoming larger and larger. Often, due to the outage of a node, a series of chain reaction problems occur, which makes the production environment suffer unpredictable losses. All kinds of uncertainties, randomness, concurrency and diversity exist in the above-mentioned problems. Because of these factors, it brings great troubles to the relevant staff to locate the causes of network failure. In order to solve the problem of network failure in large-scale data network, it is difficult to locate the fault location accurately, and how to give users accurate feedback on the causes of network failure. In this paper, we construct a random forest algorithm, which is based on agent distributed in each device. The system is trained continuously by machine learning, and the samples are classified by decision tree classifier. Each decision tree represents a judgment of the cause of network failure. When an error occurs, the agent collects and processes the device data information according to the algorithm and feeds it back to the controller for aggregation. Then, the analyzer is used to match, judge and find out the corresponding network faults that meet the requirements. The practical application results show that the design model is feasible, which significantly improves the efficiency of problem tracking and the accuracy of problem feedback screening, but also provides strong support for the follow-up intelligent operation and maintenance.

1. Research background
With the increasing of network technology, the speed of hardware development has been unable to keep up with the growing user demand groups. The distributed architecture deployment mode has gradually replaced the use of a single server to deal with massive computing tasks, and has been widely used in website construction, software applications, simulation and so on. At the same time, the distributed system constructed by the huge cluster of devices, whether in the user terminal or in the server side, has brought a series of problems. The detection and removal of network faults are a kind of problems that perplex network operators. Because of its randomness, uncertainty, instability and diversity, it directly affects the quality of service of the network and the difficulty of troubleshooting. When a network failure occurs, the first step is to diagnose the fault, so as to find the source of the problem, such as protocol failure, connectivity failure, configuration, equipment failure, and DDOS attack, then troubleshooting is carried out.

Taking network delay as an example, more serious network delay often makes users feel that the network is interrupted, or the server cannot respond, but where the specific problems need to be eliminated one by one. This is not only time-consuming, but also requires individual detection at the
terminal location, and increases the cost of services. In recent years, some systems with the above-mentioned comprehensive analysis capabilities, such as NetPoirot, Pingmesh, IAT, IxChariot, etc. They have the ability to collect and analyze network failures from the terminal to the server and adopt a distributed structure to test the performance of networks of any size or form[1][2]. On the basis of the former two, machine learning is used to initialize the system in the early stage, and classification algorithm is used to define a set of criteria for eliminating faults, indicating various network faults in training[3]. By adding visual tools (such as obfuscation matrix model) to compare the classification results and actual test values[4], the accuracy and accuracy of each fault model instance are further strengthened. It has been proved that: 1. Server-side high CPU utilization, high I/O data flow and slow data reading. 2. High CPU usage, high I/O data flow and high memory usage in the client. 3. Bandwidth, packet loss and data access delay in the network layer. The above three different dimensions set up mutual network fault examples, which achieved remarkable results in the practical application phase of fault removal.

Through reviewing the existing literatures, it is found that machine learning can solve fault location in large-scale data center environment, helping staff quickly find and analyze problems[5], but the effect of high concurrency, complex network fault and self-learning is not ideal. Based on the above findings, this paper constructs an agent distributed in each device as the basic medium, through machine learning system for continuous training. According to the decision tree classifier, the samples are classified and analyzed by data mining, so that the incentives for each fault are dispersed, and the random combination is made again to generate more fault samples. Finally, when an error occurs, the agent will collect and pre-arrange the device data information according to the algorithm, and feedback it to the controller for summary. Finally, the analyzer will be used to match and find out the corresponding qualified network faults.

2. Agent Service Architecture Based on Distributed Cluster Environment Design

When network failure occurs, the first step is to diagnose the fault, so as to find the source of the problem, such as protocol failure, connectivity failure, configuration, equipment failure, and DDOS attack before troubleshooting [6]. Therefore, three principles should be maintained in this design framework. First of all, the service is lightweight, and the system has been able to keep running. It needs a lightweight solution (from CPU, storage and network overhead, etc.) to avoid the application performance degradation caused by overload. Secondly, service universality, which is dependent on the universality of the system, is the result of the ever-changing application development due to the differences between different network areas or deployment environments. Finally, the positioning accuracy, provide accurate and effective network fault results, the system must identify the causes of the failure, and sort out the problem in which components. Therefore, a lightweight service architecture suitable for distributed environment is designed, as shown in Figure 1.

![Distributed System Diagram](image)

Figure 1. Distributed Service Architecture Diagram.

2.1 Agent

Before the agent is put into formal use, it should record the pre-data first and set the indicators of network transmission. This is an interval value. If the data collected by the agent in the formal use
process exceeds the interval of the indicators we pre-set, the agent will send the investigation request to the controller. Next, the controller collects data and analyzes the source of the fault for each node of the line. Of course, if the controller finds that everything is normal, it will return a message to the machine and modify the upper and lower limits of the interval. The feature of this design is to ensure that each machine can find appropriate network access standards in different network environments, rather than being set as a unified human.

In order to solve the problem of single data acquisition and data collection for different dimensions, the system will include the router in the whole network fault detection conditions and use the remote monitoring mode to collect and analyze the data of router and load balancing.

2.2 Controller
The controller will connect agents, process request data transmitted by agents, issue specific operation instructions to agents, and collect data transmitted by agents to analyzers for data analysis and data mining. The principle of the controller is continuous operation and random sampling, rather than comprehensive data collection for each machine. The reason for this design is that the system realizes the reduction of the operating burden of the controller. When the network area where the agent is located fails, the agent sends out a detection request, and then the controller allocates the reserved resources to deal with emergencies, thus solving the problem that a single machine cannot load when facing massive data requests.

The controller also takes part in data classification, sorting out the preconditions for each failure collected from PC terminals, servers and routing, classifying them, and finally passing them into the analyzer. Supplementary data are also executed. For example, when cross and concurrent faults occur and the first batch of data is less or data need to be collected again, the analyser can not determine the specific fault, the controller will randomly forest according to the set fault, then it will send data request information to the agent again, let the agent expand the scope of data collected, and record new failures into the list.

Finally, the controller is also responsible for initiating the function of training group for system machine learning. Under normal production environment, the isolated training group randomly sends training requests, obtains the specific data information of the agent, and collects the test data of the equipment where the real-time agent is located. Then the device is injected with network fault errors. After the simulation test, the data are compared with the data recorded before.

2.3 Analyzer
The analyzer will receive the data from the controller, analyze and eliminate them according to the pre-set random forest list of faults, and give the corresponding test results. The analyzer not only carries on the machine learning stage according to the fault problem discovered by the developer and tester, but also joins the continuous learning stage in the process of putting into production. The agent collects the preconditions of various new network faults, learning records and aggregating them into a new decision tree in the analyzer ensures that the system accumulates new knowledge continuously.

2.4 Data Base
In this stage of data storage, the design attempts to achieve synchronization of data to each agent's PC terminal by means of mature distributed data storage, which can help the agent to analyze the first local data and collect the preconditions of failure quickly. If a new fault is found, an analyzer can record it. The controller initiates a data synchronization request, and updates the data of the devices where the agents are located.

3. Key technology implementation

3.1 Design of Learning Model for Fault Classification Design
With the complexity of network faults and inadequate collection of problem analysis in the early stage, in order to solve the problem of autonomous learning in the absence of data and a small amount of data, this paper designs a fault analysis model based on neural network design to improve the structure, as shown in Figure 2.

![Training Model](image)

Figure 2. Training model design.

Firstly, a set of training units is set up for continuous training of the system. The controller can send the specific data information of the acquisition agent at random[7]. Then, the test data of the real-time current agent equipment will be collected, and then the network fault error will be injected into the device. After the simulation test, the data will be compared with the data recorded before. Secondly, for each additional fault precondition, the analyzer will automatically combine with other conditions, and then send it to the training unit for simulation test to obtain decision tree samples. The key point is to design the random forest according to the fault preconditions. The fault preconditions refer to the preconditions based on TCP packet, TCP connection, network connection and session. For example, when the network loses packets, first of all, we usually look at whether we can connect to the IP address we access. Secondly, look at the access time, and finally look at the data loss rate to determine whether there is a packet loss phenomenon. Based on the above, we can generate three fault preconditions and record them in each agent. The proxy collects data and analyzes the first data when confirming the network failure; the data analysis here refers to matching the preconditions of the failure, which are also fed back to the controller, rather than returning the whole fault data source to the analyzer for analysis. These fault conditions will be collected in the controller and sent to the analyzer for comprehensive data analysis, and the results are given.

3.2 Data Classification and Mining

In the specific data analysis and mining, this paper uses a centroid-based classifier (CBC) model[8], which has a simple and efficient classification method, and ensures the lightweight and versatility of data analysis services, because of its linearity in the time complexity of training stage. This feature makes it have great advantages in the classification of complex environment with high incidence of network failures. The basic idea of CBC analysis is to calculate the similarity between an unlabelled fault sample and the centroid of each fault class defined and divide the sample into the category with the greatest similarity. Specific formulas are as follows:

\[
C^{(m)} = \frac{1}{C_m} \sum_{X(i) \in Classm} X^{(i)}
\]  

(1)
Calculate the arithmetic mean of all training samples in category $m$ to get the centroid vector $C^{(m)}$.

Where $X^{(i)} \in \text{Class}_m$ represents the number of $i$ in sample list. Molecular $C_m$ represents the total number of samples in category $m$.

In the prediction stage, the similarity between the unlabelled sample $x$ and the center of mass $C^{(m)}$ is calculated by the reciprocal of the Euclidean distance. The formula is as follows:

$$\text{Sim}(x, C^{(m)}) = 1 / \sqrt{\sum_{k=1}^{n} (x_k - c_k^{(m)})^2}$$  \hspace{1cm} (2)

Finally and in order to avoid affecting the overall calculation, some cross-domain category definitions are often filtered out in advance[9], and then the unlabelled sample $x$ is classified into the category of the centroid with the greatest similarity, which is redefined as shown in the following formula:

$$\text{Class}(x) = \text{agr} \ Max_m \ \text{Sim}(x, C^{(|\{j: t_i - m \in d_j\}|)})$$  \hspace{1cm} (3)

Using the formulas given above, the bias values of the unlabelled fault samples and the most similar category values are calculated. The synchronous data function of the analysis manager is sent to the analyzer of the production environment, and the fault samples in the database are updated.

4. Results and Analysis
Firstly, the distributed virtual machine environment is deployed based on windows 2012. Thirty-five classes and 120 fault samples of their subclasses are initialized and imported into the database and analyzer model. Server, client and network communication are monitored respectively. The specific contents are shown in Figure 3.

![Figure 3. Fault List Example.](image)

Then the fault classification process based on centroid is presented as follows:

a) Firstly, the corresponding centroid of each category is obtained.

b) Secondly, the similarity between accidental faults and centroid is calculated.

c) Then, according to the Class $(x)$ calculation method, it is matched to the category corresponding to the center of mass with the highest score.

d) Finally, the results are fed back to the engineers for evaluation, and validated by testing, updated to the fault table, generated a new random tree, and sent to the analyzer of the production environment.
After the proxy server collects samples for a period of time, after the above steps, the test results show that the samples collected in the fault detection data set are unevenly distributed, and most of them are about 95.3% of the normal situation index, while the fault samples only account for 4.7%. The same situation also exists in the specific fault location data set. As shown in Figure 4.

| Fault classification | Number of Sample | Proportion of Sample | Peak of Matching |
|----------------------|------------------|----------------------|-----------------|
| server               | 989              | 45.8%                | 97%             |
| Client               | 367              | 17.3%                | 96%             |
| Network              | 759              | 35.9%                | 84%             |
| Total                | 2115             | 100%                 | -----           |

Figure 4. Fault Detection Data Analysis Diagram.

As shown in Fig. 5, the accuracy decreases with the increase of initial data, but with the increase of time, the original algorithm only considers the initial matching category and the correlation of faults, which will affect the actual evaluation value of the model when new data is introduced. The learning model proposed in this paper takes the similarity between the new fault samples and the defined samples into account in the scoring for a period of time, so as to ensure the continual stability of the model learning. With the increase of time and data correlation, a satisfactory fault analysis list is finally provided for users. Finally, according to the success rate of the engineer's evaluation results, it can be seen that the model has become stable, which shows that the recognition rate of fault samples can meet the needs of daily operation and maintenance, reduce the time of manual judgment, and improve the difficulty of troubleshooting.

5. Conclusion
The large-scale data analysis system established in this paper solves the problems of difficult troubleshooting, high time-consuming and large investment when network faults occur in complex network data environment. The system automatically makes a comprehensive analysis of the overall network environment where the faults occur and gives the optimal results. At the same time, network faults with high concurrency and complexity can also be detected and detected. These two points are very innovative. As part of our future work, we intend to further improve the functions of agents and controllers, and gradually train the system for machine learning, test the effect of data mining under manual intervention, and analyze and screen the optimal results by accumulating knowledge of the system's fault problems.
REFERENCES

[1] B Arzani, S Ciraci, BT Loo, A Schuster, G Outhred. Taking the Blame Game out of Data Centers Operations with NetPoirot. Conference on Acm Sigcomm Conference, 2016: 440-453

[2] C Guo, L Yuan, D Xiang, Y Dang, R Huang. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. Acm Conference on Special Interest Group on Data Communication, 2015, 45 (4):139-152

[3] DJ Rezende, S Mohamed, D Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. EprintArxiv, 2014:1278-1286

[4] M Burgess. Probabilistic anomaly detection in distributed computer networks. Science of Computer Programming, 2006, 60 (1):1-26

[5] LM Surhone, MT Tennoe, SF Henssonow, L Breiman. Random Forest. Machine Learning, 2010, 45 (1):5—32

[6] Q Fu, JG Lou, Y Wang, J Li. Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis. IEEE International Conference on Data Mining, 2009: 149-158

[7] Silver, A Huang, CJ Maddison, A Guez, L Sifre, Mastering the game of Go with deep neural networks and tree search. Nature, 2016 , 529 (7587) :484

[8] Wang Hui, Zuo W, Yuan H. A Classification Method Base on Centroid and Ontology [C]. Journal of Computer Research and Development,2007:6-11.

[9] R Zhu, B Liu, D Niu, Z Li, HV Zhao. Network Latency Estimation for Personal Devices: A Matrix Completion Approach. IEEE/ACM Transactions on Networking, 2017, PP (99):1-14