A Study on Adaptive Fault Detection Algorithm for Distributed Storage System

MA Weijun, WANG Qiang, HE Xiaohui
Institute of Field Engineering
PLA University of Science and Technology
Nanjing, China

SHU Xiaocun, MA Qiang
Institute of Meteorology
PLA University of Science and Technology
Nanjing, China

***Abstract*—In order to establish an efficient and reliable fault detection mechanism for the distributed storage system, a mixed status piggyback adaptive fault detection algorithm was presented. In addition, DFDF, a fault detection framework for the distributed storage system was also put forward and it was able to provide load balancing and high reliability by administrating the operational mode of mutual backup and concurrent working for nodes. According to detection delay calculation based on load predictions and the adaptive adjustment of timeout parameters, a fault detection adaptive algorithm known as MSP-AFD was proposed. Experiments proved that detection performance of MSP-AFD algorithm was superior to the traditional NFD-E algorithm in various load conditions so that both performance of fault detection and self-adaptability was improved.**

Keywords—Fault Detection; Distributed Storage System; Load Prediction; Timeout Allowance; Timeout Time Limit; Detection Period

I. INTRODUCTION

Fault detection has always been the research focus and challenge in the field of distributed system. With the development of distributed applications as well as the increase in both system scale and complexity, researches specific to fault detection begins to gradually develop towards self-adaptability, high efficiency (low latency) and low cost, etc. of a distributed environment. In terms of the fault detection algorithm, the most representative one is the algorithm for adaptive network status fault detection which is put forward by Chen [1]. According to such an algorithm, heartbeat message arrival time can be predicted in line with changes in network status; moreover, QoS concept for fault detection is also presented here for the first time. Bertier [2] combines this method with the prediction algorithm [3] which is put forward by Jacoboson to further propose a new adaptive fault detection algorithm. Despite of shorter detection delay, the detection accuracy of such a new algorithm is inferior to the method put forward by Chen. Fault detection unit φ is come up with by Hayashibara [4] with an attempt to hand over the right to interpret fault detection results to the upper application so as to adapt to QoS demands of different applications.

Fund Program: "A Study on Trust Routing Protocol of Policy-driven Ad hoc Network", National Natural Science Foundation of China (61070174)

In recent years, researches on fault detection mainly focus on fields of sensor network, grid and software system, etc.. Related representative literatures are presented below. In [5], a fault detection and identification algorithm based on Kalman filtering is raised to expand faults into three categories, such as stop, viscosity and shift. An enhanced distributed fault detection (DFD) method is presented in [6], which is able to provide high accuracy in cases of a small number of sensor network nodes with a high error rate. In [7], a method that can be adopted to acquire the spatiotemporal correlation among sensors is also proposed; then, sensor fault detection is performed by such a learning mode and the distributed computing technique; as a result, this method exhibits some advantages even in a harsh environment. In [8], a new grid-based architecture is presented. This architecture can be utilized to decompose a grid into several cell grids for fault detection and recovery; moreover, power consumptions of detections and recovery can be both reduced by means of such a grid-based structure. Fault detection and isolation technique in a distributed time delay system is studied in [9] to put forward concept and construction algorithm relevant with ontology with a limited unobservable state in this system; this technique is provided with high detection efficiency. Reference [10] studies the fault detection problem in complex software systems; by digging information from related clusters, rank sum test based on an entropy measure is adopted to perform fault detection. Such an approach has favorable stability and accuracy. For [11], a rotating machinery fault diagnosis expert system featured with case-based reasoning (CBR) and rule-based reasoning (RBR) is designed and implemented to improve both the efficiency and the pertinence of fault detection and diagnosis in this field. In [12], a Petri net system of mechanical vibration fault diagnosis is designed and 5 Petri net models together with the corresponding fault diagnosis algorithms are also presented so as to bring down fault misdiagnosis rate and the rate of missed diagnosis.

As a complex distributed system, how to detect faults in an accurate and rapid manner is the premise to guarantee that the distributed storage system can operate stably and fault recovery can be carried out timely when it occurs. The existing fault detection mechanism is equipped with certain inadaptability as far as a distributed storage system is concerned and primary expressions are as follows. The most existing fault detection algorithms perform calculations based on time delay of heartbeat messages and main researches focus on prediction and analysis of network delay; however,

impacts of the heartbeat message contents s and the loads of detected nodes on the detection delay are rarely studied.

Therefore, it is necessary to design applicable fault detection algorithms specific to the distributed storage system with an aim to provide a powerful basis for stable operation and fault recovery of this system.

II. Mixed Status Piggyback Adaptive Fault Detection Algorithm

Direct at application features of a distributed storage system and characteristics of nodes, a new fault detection algorithm is presented in this paper, which is known as Mixed Status Piggyback Adaptive Fault Detection Algorithm (MSP-AFD).

The MSP-AFD algorithm adopts a mode of self-detection combining piggyback to merge information from both fault detection and status detection, enhance fault detection function, and reduce detection loads. The information of status detection includes real-time status information of software and hardware in each detected node from the distributed storage system as well as the current applied load information such as the current CPU utilization, memory usage, disk space available, key process status, operating system status, current task load and load duration, etc. During fault detection, such information serves as heartbeat messages that should be sent to detectors who run the fault detection algorithm according to the status information.

Furthermore, the MSP-AFD algorithm is divided into a detecting terminal algorithm and a detected terminal algorithm. Communication mechanism between them is in a form of PULL. In other words, the detecting terminal only sends detection information to the detected terminal when detection is required. As detections are made according to demands, it has favorable adaptability and scalability for distributed storage systems with high complexity and large size.

It is assumed that the detecting terminal S considers denoting detection answer messages from the detected terminal D for n times as $m_1, m_2, m_3, \mathrm{K}, m_n$; among which, mi is expressed by 8 tuples $(CPU_i, RAM_i, DISK_i, P_i, OS_i, IO_i, \Delta t_i, \Delta l_i)$. To be specific, the previous 6 tuples refer to current system real-time status of detected nodes, that is, current CPU utilization, memory usage, disk space available, key process status, operating system status and I/O load; among them, Δt_i refers to computing and processing time delay or computation time of the detected nodes, that is the duration from receiving the detection message to sending this message, because the acquisition of nodes and calculation of status information need to pay a price. With regard to nodes in a distributed storage system, most are I/O data intensive processing nodes; especially when system load is large, the corresponding cost should be given serious considerations. In addition, Δl_i refers to the estimated load processing end time by detected nodes according to the present task processing load; it is also known as load duration; especially for main tasks such as node and proxy node storage, etc., they are nodes for file access, and the end time for tasks can be roughly figured out according to file size and current network bandwidth.

The round-trip delay of detection messages for n times recently is assumed to be $R_1, R_2, R_3, \mathrm{K}, R_n$, detection period to be T, timeout time limit to be $TO_1, TO_2, TO_3, \mathrm{K}, TO_n$, computing & processing delay of detected nodes for n times recently to be $\Delta t_1, \Delta t_2, \Delta t_3, \mathrm{K}, \Delta t_n$, task processing load duration of detected nodes for n times recently to be $\Delta l_1, \Delta l_2, \Delta l_3, \mathrm{K}\Delta l_n$, predicted computing & processing delay value of detected nodes for n times recently to be $\theta_1, \theta_2, \theta_3, \mathrm{K}, \theta_n$, and the predicted round-trip delay value of detection messages for n times recently to be $PR_1, PR_2, PR_3, \mathrm{K}, PR_n$.

According to the round-trip delay of detection messages for n times recently and the load status of detected nodes, the round-trip delay of detection messages for n+1 times can be predicted.

$$PR_{n+1} = \frac{1}{n}\sum_{i=1}^{n}(R_n - \Delta t_i) + \theta_{n+1} \qquad (1)$$

In (1), $\frac{1}{n}\sum_{i=1}^{n}(R_n - \Delta t_i)$ refers to the prediction about round-trip delay of detection messages for $n+1$ times; and θ_{n+1} represents the prediction about detected node calculation time for $n+1$ times.

If make $\overline{\Delta t_n} = \frac{1}{n}\sum_{i=1}^{n}\Delta t_i$, $\varepsilon_n = \Delta t_n - \Delta t_{n-1}$ before the calculation of θ_{n+1}, the computing & processing delay $\Delta t_1, \Delta t_2, \Delta t_3, \mathrm{K}, \Delta t_n$ of detected nodes for n times recently shall be dealt with as follows. If $|\varepsilon_n| \geq \Delta t_{\max} \wedge \Delta l_n < T \wedge \varepsilon_n > 0$, make $\Delta t_n \leftarrow \Delta t_{n-1}$.

Then, (2) is adopted to compute θ_{n+1}.

$$\theta_{n+1} = \begin{cases} \Delta t_n & |\varepsilon_n| \geq \Delta t_{\max} \wedge (\Delta l_n \geq T \vee \varepsilon_n < 0) \\ \overline{\Delta t_n} & else \end{cases} \qquad (2)$$

In (2), $\overline{\Delta t_n}$ refers to the average value of detected node calculation time for n times recently; ε_n refers to the variation in node calculation time for the last time, and $\Delta t_{\max}$ to threshold value of sudden change in detected node calculation time mutation. The calculation of θ_{n+1} can be carried out in 3 different cases.

A. . $|\varepsilon_n| \geq \Delta t_{\max} \wedge (\Delta l_n \geq T \vee \varepsilon_n < 0$

This expression corresponds to two situations. First, a sudden change (abrupt increase) occurs to load of the last time (n times) and its duration exceeds the detection period, which

indicates that load of such a node will be maintained despite of some minor variations during the next detection period. For a storage system, if files are delivered continuously or it is confronted with a large number of data requests and accesses, such two situations may take place. Second, a sudden change (abrupt decrease) occurs to load of the last time (n times) which shows that the load has ended and the detected node has recovered to normal status. Under the above two cases, calculation time of detected nodes for the last time (n times) can be utilized to predict the calculation time for $n+1$ times. Principle for this prediction method can be described in this way. If a sudden change occurs to load and lasts for a long period, it can be deemed that the calculation time during the next detection period is approximately expressed by that within the previous detection period.

B. $|\varepsilon_n| \geq \Delta t_{\max} \wedge \Delta l_n < T \wedge \varepsilon_n > 0$.

That is, a sudden change occurs to load for the last time (n times) and lasts for a short period which is less than a detection period. In this case, this load will be neglected by an algorithm which treats load of the last time ($n-1$ times) as that of this time which is directly adopted to carry out prediction calculation for the next delay. Principle for this prediction method can be described in this way. If a sudden change occurs to load and lasts for a short period, it can be deemed that such a load is accidental. Therefore, it is prominent that the node calculation time of this load is inapplicable to approximately representing calculation time of the next time; in addition, due to being accidental, it fails to stand for a trend or universal phenomenon so as to be inapplicable to the prediction about average value calculations. Concerning such a category of load, the algorithm chooses to neglect them to guarantee the accuracy and efficiency of it.

C. $|\varepsilon_n| < \Delta t_{\max}$.

It means that no sudden changes occur with the load, which indicates that the load is in a normal state. Hence, the average value of node calculation time for n times recently is directly employed to predict the node calculation time of the next period.

Considering timeout time limit settings for $n+1$ times, it can be expressed as follows.

$$TO_{n+1} = PR_{n+1} + \rho_{n+1} \qquad (3)$$

Where, ρ refers to timeout allowance which is set up specifically to network state and QoS demands of fault detection; it is different from NFD-E algorithm, known as ERROR! NO REFERENCE SOURCES FOUND. presented by Chen. The timeout allowance of NFD-E is primarily figured out according to the assumed network state (packet loss probability) and QoS demands of fault detection (T_D^U, T_{MR}^L, T_M^U) ; among them, T_D^U, T_{MR}^L, T_M^U respectively refer to upper bound of detection delay, lower bound of incorrect interval and upper bound of incorrect duration. Under a circumstance of minor changes in network state, the timeout allowance of it can be rather stable. The timeout allowance ρ which is adopted by the MSP-AFD algorithm can be adjusted in real time in line with the practical detection delay results to make the algorithm able to detect faults within the shortest time; as a result, the high efficiency of such an algorithm is ensured. Computing method for ρ is shown below.

Make $$\eta_n = \frac{\sum_{i=1}^{n} \gamma_i \left| PR_i - R_i \right|}{\sum_{i=1}^{n} \gamma_i} \qquad (4)$$

Then

$$\rho_{n+1} = \begin{cases} \max\{\rho_n + p(\eta_n - \rho_n), q\eta_n\} & n > 0 \\ \alpha & else \end{cases} \qquad (5)$$

In (4), η_n refers to the statistics which are made for round-trip delay prediction error of the detection information; it gives expression to the application of this error in a form of weighted average. If the current detection time is closer to the error, such an error hence is able to represent estimation accuracy for current actual network status and node loads to a greater extent, which means that it fits the physical truth to a higher degree. Therefore, this error takes a large proportion in error statistics. In this equation, γ_i denotes weight, the larger i is, the higher the weight will be. Generally, $\gamma_i = i^c$ is acceptable and c is a constant.

In (5), α refers to the timeout allowance of the NFD-E algorithm; at the initial state, ρ is equal to α ; and with the increase in detection times, ρ should be computed again according to detection result of the previous n times by taking detection error and the impacts of both actual network status and detected node loads into a full consideration. Among which, p and q are constants; $\max\{\rho_n + p(\eta_n - \rho_n), q\eta_n\}$ means that a threshold denoted by $q\eta_n$ is set for changes in ρ to prevent the value of ρ from being over low so as to avoid detection error.

By summary, the computing method for timeout time limit of detected nodes is presented below.

$$TO_{n+1} = \frac{1}{n}\sum_{i=1}^{n}(R_n - \Delta t_i) + \theta_{n+1} + \rho_{n+1} \qquad (6)$$

It is clear that MSP-AFD algorithm is able to keep adjusting timeout time limit with an aim to adapt to changes in network state and detected node loads to improve detection efficiency. For the judgment of whether a detected node has collapsed or not, the detection terminal adopts the following strategy.

During detection of $n+1$ times, if it is found to fail to receive answer message from the detected terminal after going

through a timeout time limit TO_{n+1} after sending detection information, detected nodes are deemed to be broken down.

Features of the MSP-AFD algorithm can be described from the following aspects：

Able to merge information of fault detection and status detection to enhance the fault detection function and reduce detection loads.

An approach of load prediction is applied to make statistics for detection delay specific to characteristics of all nodes in such a distributed storage system, with an aim to explore new delay prediction methods for fault detection so as to improve the accuracy of fault detection.

Able to adjust timeout allowance and timeout time limit in a self-adaptive manner according to network status and loads of detected nodes; in comparison with the traditional methods such as fixed timeout allowance and average value prediction, etc., such an algorithm is more flexible and can be utilized to improve both efficiency and self-adaptability of fault detection so that a favorable balance is achieved between accuracy and efficiency of it.

III. EXPERIMENT AND ANALYSIS

For experimental environment, please refer to the physical environment of a distributed storage system; the experimental environment should be constructed on the basis of a wide area network (WAN) and respectively tested by two PCs (denoted by P1 and P2) that are located in Nanjing and Beijing. For P1, detecting terminal software is deployed, while detected terminal software deployed on P2. In order to highlight pertinence of this experiment, P1 is simulated as a storage management node and P2 as a storage node; hence, the storage management node can be tested from of its detection performance and accuracy on storage node status. Considering the practical operating status of the storage node, the following 2 loads are simulated for P2.

A. Load Balancing (NORMAL).

The load is rather stable and change in Δt_i is minor; with regard to the storage node corresponding to such a load, the requested memory capacity is also stable. Moreover, the storage node is able to respond to requests normally.

B. Sudden Sustained Overload (HUGE_PS).

The load substantially increases all of a sudden in a continuous manner and Δt_i goes beyond the control range of TO_{n+1}. With regard to the storage node corresponding to such a load, the requested memory capacity also increases suddenly and exceeds the endurance of storage node. Consequently, overload occurs to the storage node that becomes unable to respond to requests timely; as a result, request delay is rather large.

During the test, the MSP-AFD algorithm is compared to the NFD-E algorithm presented by Chen; algorithm performance and accuracy are investigated by detected nodes under circumstances of different loads.

In the process of this test, the network environment is rather stable. Hence, computational delay of detected node is the main delay factor that is taken into account in this process. Under an initial state, settings for various parameters are presented in Table 1.

TABLE I. TESTING PARAMETER DESCRIPTIONS OF MSP-AFD ALGORITHM

Serial No.	**NOP**[a]	**IOP**[b]	**Set Value**
1	DMSP[c]	T	1000ms
2	TTL[d]	TO_1	600ms
3	SF[e]	n	10
4	CTMT[f]	Δt_{max}	50ms
5	ESWP[g]	c	0.25
6	TAC[h]	p	0.05
7	TAC[h]	q	3
8	TF[i]	100	

[a]NOP: Name of Parameter
[b]IOP: Identification of Parameter
[c]DMSP: Detection Message Send Period
[d]TTL: Timeout Time Limit
[e]SF: Statistics Frequency
[f]CTMT: Computing Time Mutation Threshold
[g]ESWP: Error Statistical Weight Parameter
[h]TAC: Timeout Allowance Constant
[i]TF: Test Frequency

The MSP-AFD algorithm is compared with the representative NFD-E algorithm put forward by Chen in allusion to 2 load cases; the test results are shown in Fig. 1 and Fig. 2.

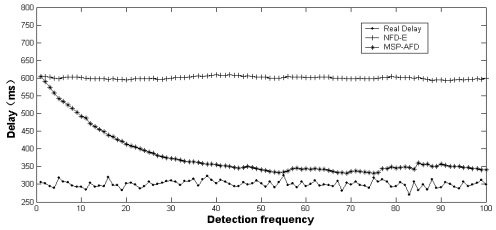

Fig. 1. Timeout value comparison of 2 algorithms for balanced loads

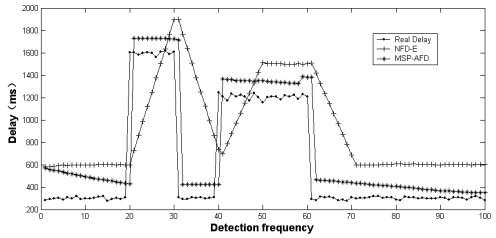

Fig. 2. Timeout Value Comparison of 2 Algorithms for Sudden Sustained Overloads

Seen from test results shown in Figure 1&2, the MSP-AFD algorithm is able to timely adjust timeout allowance according to the actual network status, as well as computational delay and prediction error of detected nodes to

make the timeout time limit (delay) of this algorithm more and more approximate to the virtual duration. However, due to the control by $q\eta_n$ in (5), the timeout time limit will not be over low that is able to cause detection error affecting detection precision. Different from the NFD-E algorithm, the adjustment of MSP-AFD algorithm is correlated to the variation mode of actual loads. To be specific, if a sudden change occurs with loads, it will occur to timeout time limit as well. As the NFD-E algorithm adopts a completely average manner to predict delay, the timeout time limit is adjusted by means of a slow change regardless of variations in loads. In addition, a sudden load has no impacts on the MSP-AFD algorithm, because it is provided with a filtering mechanism of sudden non-sustained loads; as a result, this algorithm is able to eliminate sudden non-sustained load states from prediction algorithms so as to erase the influence of accidental phenomena on algorithms. By contrast, prediction by NFD-E algorithm is carried out by means of average values. Therefore, even if there are accidental non-sustained sudden changes, they may have certain influences on the predicted value.

In Table 2, statistics results further indicate the advantages of MSP-AFD algorithm; and the relevant test objectives and objects are the average delay (D, simplified for performance; unit: ms) and the accuracy (abbreviation: A; unit: %) separately.

TABLE II. TEST RESULTS STATISTICAL DESCRIPTIONS FOR MSP-AFD AND NFD-E ALGORITHMS

TO[a] / TO[b]	NORMAL		HIGH_PS		HUGE_PS		HUGE_NPS	
	D	A	D	A	D	A	D	A
NFD-E	600.6	100	640.6	100	932.8	85	624.7	98
MSP-AFD	380.4	100	441.9	99	767.4	98	424.7	98

[a]TO: Test Objectives [b]TO: Test Objects

IV. CONCLUSIONS

Researches and analysis are performed in this paper, specific to the existing fault detection mechanism; it is found that the straight application of such a mechanism into the distributed storage system is provided with certain inapplicability which is represented by incomprehensive enough study on the existing heartbeat mechanism and low efficiency and precision of fault detection, etc. In view of those problems, a mixed status piggyback adaptive fault detection algorithm (MSP-AFD) is presented in this paper. This algorithm is able to merge information of fault detection and status detection together to enhance fault detection function and reduce detection loads. Direct at features of various nodes in a distributed storage system, statistics is made for detection delay by means of applied load prediction to explore new fault detection approaches to delay prediction so that the accuracy of fault detection can be improved. Moreover, such an algorithm has the capability to self-adaptively adjust timeout allowance and time limit according to network status and the loads of detected nodes. Relative to traditional prediction manners with a fixed timeout allowance and average values, etc., it is more flexible and is able to improve both the performance and the self-adaptability of fault detection.

Experimental data show, detection performances of the MSP-AFD algorithm under circumstances of diverse load states are superior to the conventional NFD-E algorithm. Due to the inherent contradiction between detection performance and detection accuracy, the MSP-AFD algorithm achieves a balance for them by adjusting timeout allowance.

REFERENCES

[1] W.Chen, S.Toueg, MK.Aguilera. On the quality of service of failure detectors [J]. IEEE Trans. on Computers. 2002, 51(5): 561-580.

[2] M.Bertier, O.Mann, P.Sens. Implementation and performance evaluation of an adaptable failure detector[C]. In: Proceedings International Conference on Dependable Systems and Networks. Bethesda, IEEE Computer Society Press. 2002,354-363.

[3] E.Bunrs, M.Gouda, R.Miller. Stabilization and Pseudo-Stabilization[J]. Distributed Computing. 1993, 7(1):35-42.

[4] Naohiro Hayashibara, Adel Cherif, Takuya Katayama. Failure Detectors for Large-Scale Distributed Systems[C]. In: Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems (SRDS'02), Osaka, Japan. Washington, IEEE Computer Society Press. 2002,404-409.

[5] K.Villeza, B.Srinivasanb, R.Rengaswamy, etc. Kalman-based strategies for Fault Detection and Identification (FDI): Extensions and critical evaluation for a buffer tank system[J]. Computers & Chemical Engineering, 11 May 2011, 35(5): 806-816.

[6] Peng Jiang. A New Method for Node Fault Detection in Wireless Sensor Networks[J]. Sensors, 2009, 9(2), 1282-1294.

[7] Oliver Obst. Poster abstract: Distributed fault detection using a recurrent neural network[C].In: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, San Francisco, California, USA,2009, 373-374.

[8] Peng Jiang. A New Method for Node Fault Detection in Wireless Sensor Networks[J]. Sensors, 2009, 9(2), 1282-1294.

[9] N.Meskin, K.Khorasani. Fault Detection and Isolation of Distributed Time-Delay Systems[J]. IEEE Transactions on Automatic Control, 2009, 54(11), 2680-2685.

[10] Miao Jiang, M.A.Munawar, T.Reidemeister,etc. Efficient Fault Detection and Diagnosis in Complex Software Systems with Information-Theoretic Monitoring[J]. IEEE Transactions on Dependable and Secure Computing, 2011, 8(4), 510-522.

[11] Jiang Zhinong, Wang Hui, Wei Zhongqing; Fault Diagnosis Expert System Based on Cases and Rule-based Reasoning [J]. Computer Engineering: 2011, 37(01) 238-240, 243.

[12] Huang Min, Zhang Fang; Implementation of Petri Net Model and System for Vibrant Fault Diagnosis [J]. Computer Engineering: 2011, 37(6) 228-230.

