2016 IEEE 2nd International Conference on Big Data Security on Cloud, IEEE International Conference on High Performance and Smart Computing, IEEE International Conference on Intelligent Data and Security

Cooperative Monitoring Mechanism for Cluster Based on Message Routing Queue

Guan Yang
College of Computer
Nanjing University of Posts and Telecommunications
Nanjing, China
neo_yang@foxmail.com

Xiaolong Xu
College of Computer
Nanjing University of Posts and Telecommunications
Nanjing, China
xuxl@njupt.edu.cn

***Abstract*—Current cloud computing data centers are usually based on large-scale clusters, including thousands of servers. The monitoring and management of resource is very important to the operation of cloud systems, which usually employ the traditional centralized monitoring mechanisms. The main defects of the centralized management mechanism are the bottleneck of system, the communication traffic and the single point failure. In this paper, we proposed a novel cooperative monitoring mechanism for cluster, which is based on message routing queue. This mechanism utilizes message routing queue to realize the mutual monitoring relationship between data nodes. The experimental results show that the cooperative monitoring mechanism has better performance than the traditional centralized monitoring mechanism in reducing communication traffic and the burden of master monitoring node.**

Keywords—Cloud computing; Cluster; Cooperative monitoring; Message routing queue

I. INTRODUCTION

In the past years, the global cloud services market has increased at an incredible rate. According to the prediction of [1], the global cloud services market will increase at a ratio above 15% each year in the future. And the value of the whole market will reach about 24 billion dollars in 2017. In 2015, the cloud data center of TEAMSUN in ZGC in Beijing began to use. In October 18, 2015, Ali Cloud cooperated with Meraas to construct an online cloud computing data center. In October 28, 2015, Oracle launched the service of cloud computing leasing business. In March 5, 2015, the data center of Ali Cloud in Silicon Valley began to operate. In August 24, 2015, Google invested 170 million dollars to construct cloud data center in Ireland. In 2016, the 12th cloud data center of AWS will begin to operate.

The proper operation of a cloud computing cluster needs a high efficient monitoring system, and plenty of mechanisms have been put forward such as resource monitoring policy based on monitoring agents [2], cloud-adaptive algorithm based on combined push-pull model [3], algorithm based on the publish/subscribe (pub/sub) paradigm [4], self-adaptive push model(SAPM) [5], automatically adaptive distributed monitoring architecture [6], auto window filter algorithm [7], cloud computing service platform based on OpenStack [8], ganglia distributed monitoring system[9], periodically and event-driven push(PEP) monitoring model [10].

Currently, the centralized monitoring mechanism which is used in most cloud systems appears some performance issues as the cloud computing cluster expands. Significant communication traffic would be produced both in the periodic polling method and in the mode where the data nodes report their status periodically to the master monitoring node, leading to the decrease of efficiency of the whole system and the overloaded master monitoring node.

Taking account of the deficiencies of centralized monitoring mode and the low down ratio of recent cloud system, we propose a new cluster cooperative monitoring mechanism which is based on message routing queue. Under the assistance of it, data nodes can detect mutually and report the neighboring node's abnormal circumstances to master monitoring node.

The rest of this paper is organized as follows. Section II discusses the related works. We describe the further explanation and the architecture of message router in cooperative monitoring system in Section III. The specific details of the construction of cooperative monitoring mechanism are introduced in Section IV. Section V presents our simulation results of cooperative monitoring mechanism, and finally we conclude this paper in Section VI.

II. RELATED WORKS

Large numbers of related works have been done in terms of large scale cluster monitoring system during recent years. [2] presents a resource monitoring policy based on monitoring agents, which formulates monitoring agent deployment as a 0-1 programming problem, and uses the advanced quantum genetic algorithm to solve it, this policy has outstanding performance in reducing communication cost, however, it needs a series of complicated steps to abstract the real situation. [3] proposes an improved adaptive algorithm based on the combined push-pull model, an adaptive strategy which uses history data to predict the data from the monitor is added on the producers to reduce the waste of resource, and this algorithm can decrease the resource consumption while exchanging data effectively. [4] presents a novel lighter weight and scalable resource monitoring solution based on the publish/subscribe (pub/sub) paradigm (SQRT-C), and uses effective software engineering principles to make it usable with multiple cloud platforms to

This work was supported jointly sponsored by the National Natural Science Foundation of China under Grant 61472192 and 61472193, and the Natural Science Fund of Higher Education of Jiangsu Province under Grant 14KJB520014.

978-1-5090-2403-2/16 $31.00 © 2016 IEEE
DOI 10.1109/BigDataSecurity-HPSC-IDS.2016.56

improve the system's latency, jitter and scalability. [5] proposes a self-adaptive push model(SAPM) which uses a transportation window to store the collected metrics before being delivered to the monitoring servers. The model decreases the load on a network and achieves a better performance in keeping data coherency between hosts and monitoring servers. But the parameter of window is hard to determine. [6] presents an automatically adaptive distributed monitoring architecture, which offers a monitoring platform-as-a-service to each cloud consumer that allows to customize the monitoring metrics. [7] proposes a monitoring architecture and a monitoring model using auto window filter algorithm, this algorithm has advantages in reducing the network traffic. [8] proposes a cloud computing service platform based on OpenStack which supports service management, auto-scaling, security control and high availability, and implements the resource monitoring subsystem to monitor performance metrics of CPU utilization, memory usage and network I/O of physical and virtual resource. On this platform, the system resource utilization can be improved greatly. [9] proposes the design, implementation, and evaluation of Ganglia along with experience gained through real world deployments on systems of widely varying scale, configurations, and target application domains over the last two and a half years. [10] presents a periodically and event-driven push(PEP) monitoring model to better monitor the virtual resource in cloud computing. This model can simplify the communication between nodes. However, the monitoring information can only be sent to users. [11] analyzes the properties of a monitoring system for the Cloud, and identifies open issues, main challenges and future directions in the field of Cloud monitoring.

III. MESSAGE ROUTER

A. The introduction of Message Router

Message router is actually a kind of message oriented middleware, which provides connection interfaces, a series of information transfer mechanisms and information service queues to supply an efficient cross-platform information transmission for distributed applications. The message router supports two kinds of transmission modes: synchronous transmission and asynchronous transmission, it consists of five parts: interface processing module, message queue, queue management, message channel agent, and security management. Fig.1 shows the structure of message router, interface processing module responds to the service request from process, queue management module adjusts message queues, such as creating or deleting message queue. Message channel agent is responsible for posting messages to the proper message queue according to the feedback from interface processing module, and security management is in charge of the security of the whole transmission process, for instance, encrypts and decrypts the message.

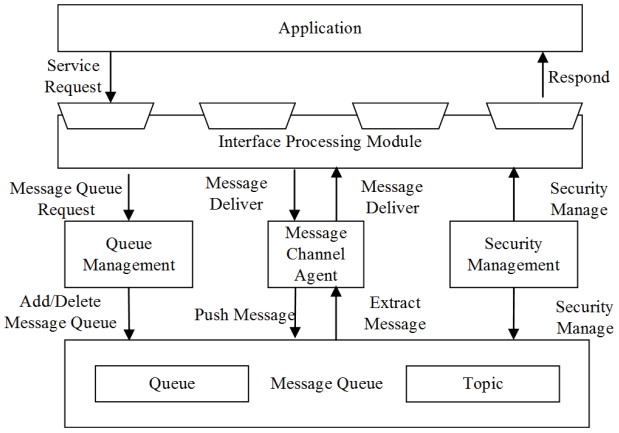

Fig. 1. Structure of Message Router

B. The two work patterns of Message Router

Message router mainly has two kinds of work pattern: point-to-point mode and publish/subscribe mode.

In point-to-point mode, two processes communicate with each other directly by creating logical link on message router. A process sends communication request to message router, and then the message router receives the request, discriminates communication mode, allocates a unique message queue (FIFO Queue) for this communication, and then establishes the connection with the other process. The messages sent by one process would be pushed into the corresponding message queue and the other process is able to take out messages from the queue.

In publish/subscribe mode, a process sends communication request to message router, and then message router receives the request, discriminates communication mode, allocates a message queue (Topic) for this communication and creates a message agent on message router. The process sends messages to message agent and then terminates communication. The message agent would take the responsibility to continue sending the message to processes which subscribe to this topic.

IV. COOPERATIVE MONITORING MECHANISM

A. The introduction of ActiveMQ

In this paper, we select ActiveMQ, which is the most popular message oriented middleware of Apache, as message router to design cooperative monitoring mechanism. It fully supports JMS1.1 and provides two kinds of information transmission mode: Point-to-Point mode and Publish/Subscribe mode. Additionally, it has a similar message structure with JMS, which consists of message header, message body, and message properties.

ActiveMQ supports two kinds of message acknowledgement schemes: transactional session acknowledgement scheme and non-transactional session acknowledgement scheme, and different kinds of communicating protocols such as TCP, NIO, UDP, VM. It also

provides four kinds of message storing methods, including AMQ, KahaDB, JDBC and Memory.

B. Construction of Cooperative Monitoring Relationship

Fig.2 illustrates the construction process of a pair of cooperative monitoring nodes, taking node A and node B as an example, node A is monitored by node B.

First, node A sends communication request to message router, the connection factory of message router receives the request and then allocates a connection for this request. Then A sends the session request which includes the transactional property of the session to the connection. After the connection receives the request, it constructs a corresponding session. Then A sends the information of transmission mode to the session and points out the name of the queue allocated for sending messages. After receiving the request from A, the session creates a homonymous queue. In the end, A negotiates with the queue about specific parameters of messages for communication, for instance, message property, message storing method and so on.

Second, Node B takes the same steps to establish the connection and session. And then B sends the information of transmission mode to the session and points out the name of the queue to receive messages. After receiving the request from B, the session connects the created queue which has the same name to node B. And then B negotiates with the queue about specific parameters of messages.

So far, a pair of cooperative monitoring nodes has been constructed. And the cooperative monitoring relationship between each node and its next node can be constructed successively in the same steps.

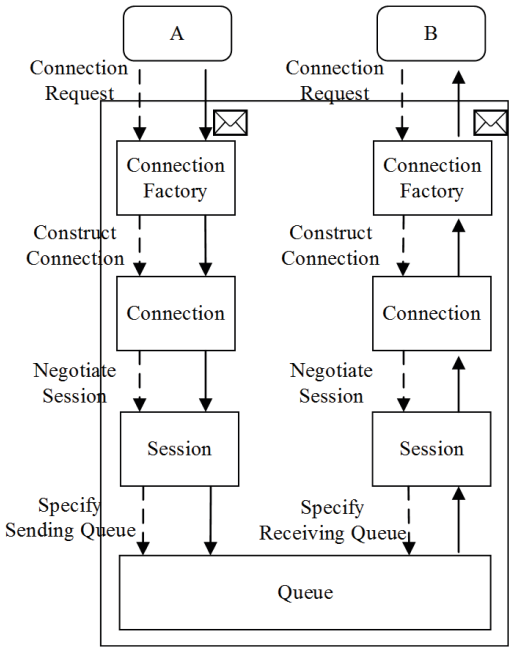

Fig. 2. Construction Process of Cooperative Monitoring Relationship

C. Analysis of Normal Operating State of Cooperative Monitoring System

Fig.3 shows the normal operating state of a cooperative monitoring system which consists of three data nodes (node A, node B, and node C) and a master monitoring node. Each data node has constructed the cooperative monitoring relationship with its next data node in the steps illustrated above. The session is defined as non-transactional session. The acknowledgment scheme is set as AUTO_ACKNOWLEDGE. And the message format is set as Text Message. The master monitoring node is responsible for receiving the reporting message from all monitoring nodes. In normal operating state, each data node pushes the message of its operating status into the allocated queue in ActiveMQ by calling the Send function. And its next node—the monitoring node would extract the message by calling the Receive function, and then discriminates the operating state of the monitored node. If the whole cooperative monitoring system operates normally, messages would only be sent between each pair of cooperative monitoring nodes.

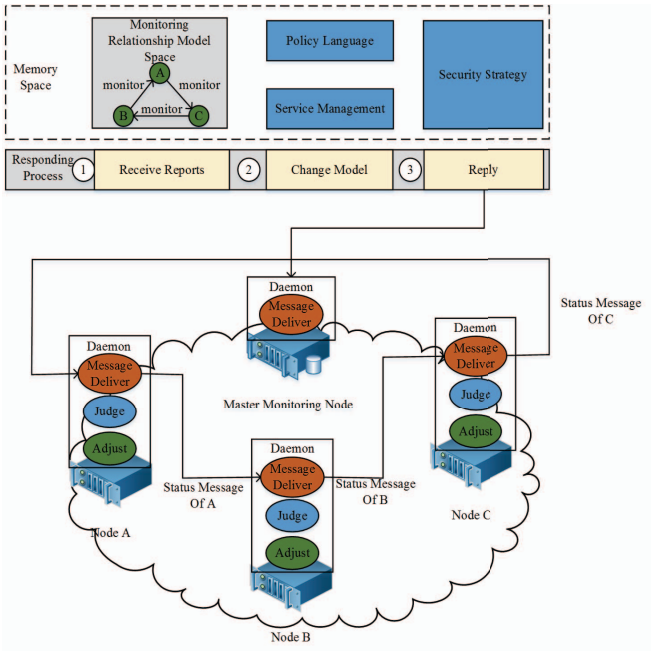

Fig. 3. Normal Operating State of Cooperative Monitoring System

D. Analysis of Operating State of Cooperative Monitoring System(Single Node down)

Fig.4 illustrates the operating state of cooperative monitoring system where one data node shuts down. When data node A shut down, it is unable to send the message of its operating status on time. Under this circumstance, its monitoring node B cannot receive the message from node A in one second.

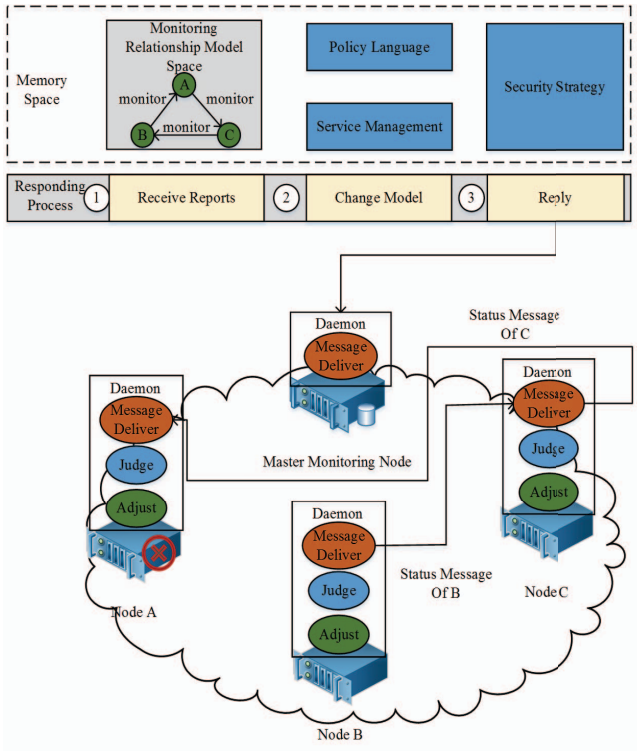

Fig. 4. Operating State of Cooperative Monitoring System (Single Node Down)

Fig.5 shows the processing procedure when one node shuts down.

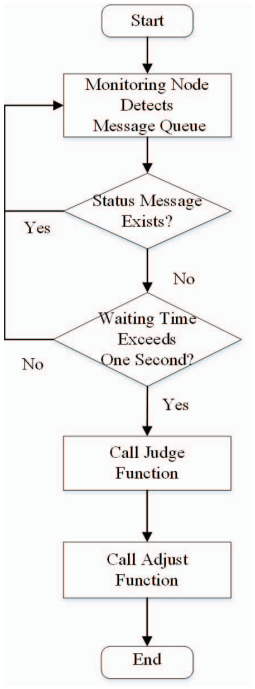

Fig. 5. Flow Chart of Processing Procedure(Single Node Down)

The monitoring node B would call Judge function to conform that node A has shut down, to report the abnormal state of node A to master monitoring node and to inquire master monitoring node about the monitored node of node A. Master monitoring node receives the message from B and then sends the information of the monitored node C back. After receiving the reply, node B calls Adjust function to inform message router to delete the queue which is allocated for A and to apply to message router for receiving the message from C at the same time.

Message router receives the request from B and then reconstructs cooperative monitoring relationship. Fig.6 shows the reconstruction of cooperative monitoring system.

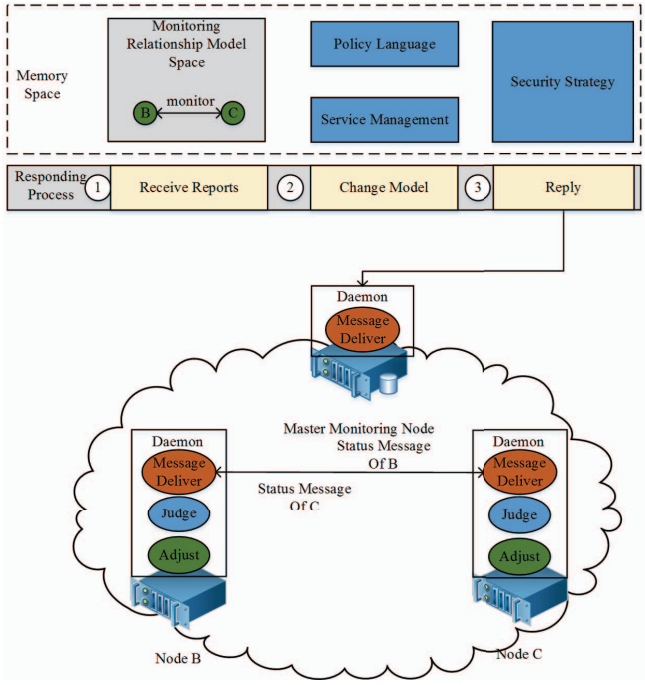

Fig. 6. Reconstruction of Cooperative Monitoring System (Single Node Down)

E. Analysis of Operating State of Cooperative Monitoring System(Multiple Nodes Down)

Fig.7 illustrates the operating state of cooperative monitoring system when multiple data nodes shut down. Taking data node B and C shut down as an example, once node B and node C shut down, they are unable to send the message of their operating status on time. Consequently, though node C is the monitoring node of node B, it cannot report the abnormal state of node B to master monitoring.

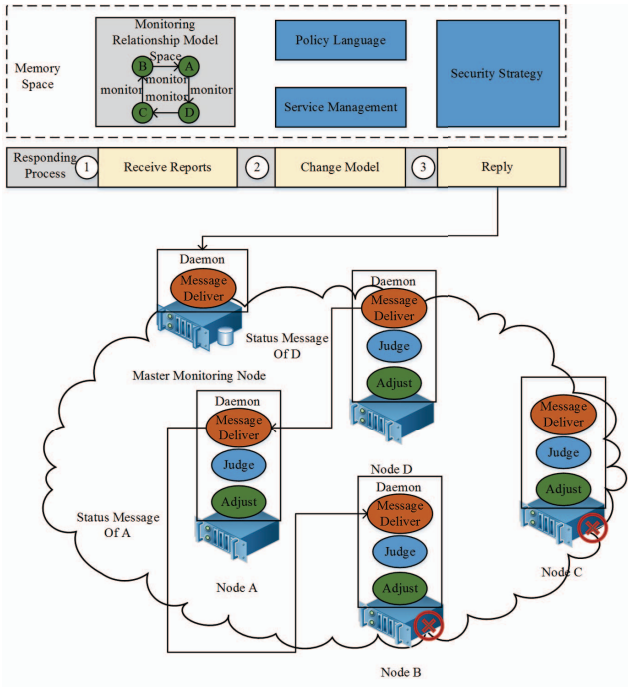

Fig. 7. Operating State of System (Multiple Nodes Down)

However, the monitoring node of C operates normally. On this condition, the monitoring node D cannot receive the message from C in one second, Fig.8 shows the processing procedure when multiple nodes shut down. Node D would call the Judge function to conform that node C has shut down, to report the abnormal state of node C to master monitoring node and to inquire master monitoring node about the monitored node of node C. Master monitoring node receives the message and then sends the information of node B back. After receiving the reply, node D calls Adjust function to inform message router to delete the queue which is allocated for C and to apply to message router for receiving the message from B at the same time.

Message router responds to the request and then reconstructs the cooperative monitoring system. A cooperative monitoring relationship is built between node B and node D. However, since node B has shut down as well, the monitoring node D cannot receive the message from B in one second, node D would call Judge function again to conform that node B has shut down, to report the abnormal state of node B to master monitoring node and to inquire master monitoring node about the monitored node of node B. Master monitoring node receives the message and then sends the information of monitored node A back. After receiving the reply, node D calls Adjust function to inform message router to rebuild the relationship again.

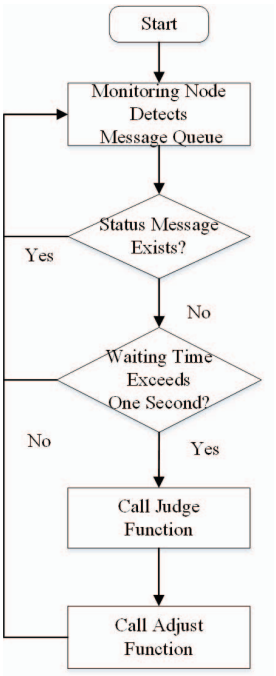

Fig. 8. Flow Chart of Processing Procedure(Multiple Nodes Down)

Message router responds to the request and reconstructs the cooperative monitoring relationship. Fig.9 shows the reconstruction of multiple-nodes-down cooperative monitoring system.

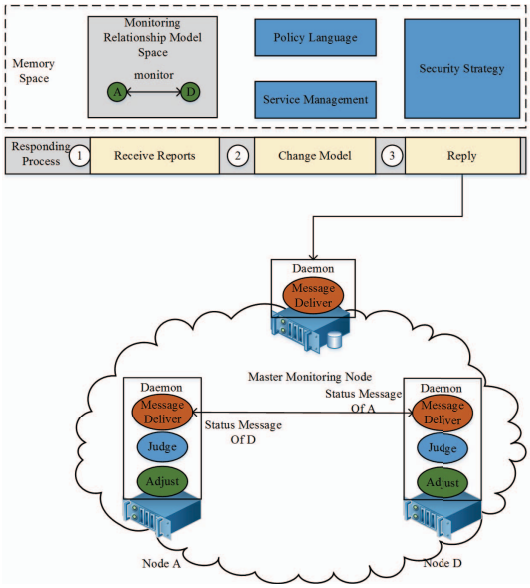

Fig. 9. Reconstruction of Cooperative Monitoring System (Multiple Nodes Down)

V. EXPERIMENTS AND PERFORMANCE ANALYSIS

The main purpose of this part is to test the performance of the new cooperative monitoring mechanism. We compare the

performances of two kinds of monitoring system—new cooperative monitoring mechanism based on message routing queue(CMMRQ) and traditional centralized monitoring mechanism(CMM). The measured aspects include the communication traffic and the required memory of each mechanism. Table.1 illustrates the testing environment of this part. We utilize ActiveMQ to construct a cooperative monitoring system and a centralized monitoring system separately, and then we measured the communication traffic and the required memory of each system in different condition.

TABLE I. TESTING ENVIRONMENT

Experimental Host	HP DL 380 G4 378735-AA1 Intel Xeon 1.60GHz X 2, 8 Core / 4G Memory
Operating System	RedHat Enterprise Advanced Server 4.0
Database	Oracle 10g
Number of Nodes	200 Data Nodes and 1 Master Monitoring Node
Version of ActiveMQ	ActiveMQ 5.2.11.1
Parameter configuration	The length of each message is 1KB. The message is only stored in Memory. Each data node sends one state message to its monitoring node per 100 milliseconds. Monitoring node needs to send three messages to report and fix the cooperative monitoring relationship.

Fig.10 shows the communication traffic and Fig.11 illustrates the required memory of each system respectively when the ratio of data nodes shutting down in the system is 5 %, 10%, and 15%.

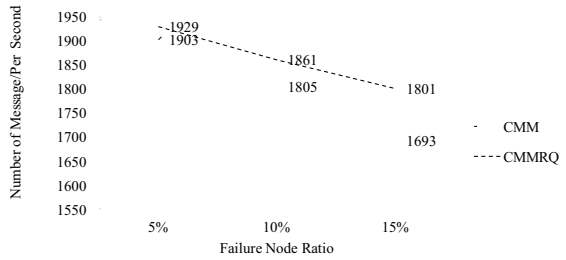

Fig. 10. Communication Traffic of Whole System in Different Failure Node Ratio

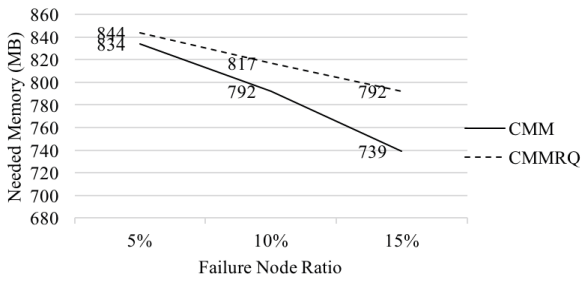

Fig. 11. Required Memory of Whole System in Different Failure Node Ratio

Since after one node shutting down, its monitoring node would send extra messages to report, the communication traffic and required memory of cooperative monitoring system are relatively higher than those in centralized monitoring system. However, as the percentage of failure node increases, the decreasing rates of communication traffic and the required memory in cooperative monitoring become more stable.

Fig.12 and Fig. 13 show the communication traffic and the required memory of the master monitoring node of each system in different failure node ratio.

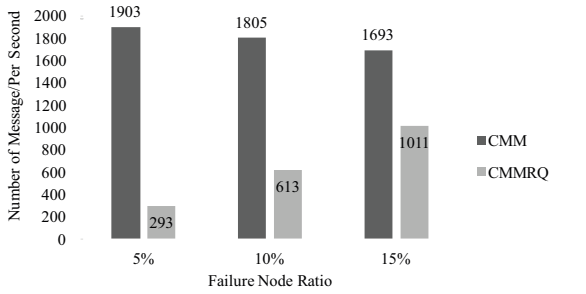

Fig. 12. Communication Traffic of Master Monitoring Node in Different Failure Node Ratio

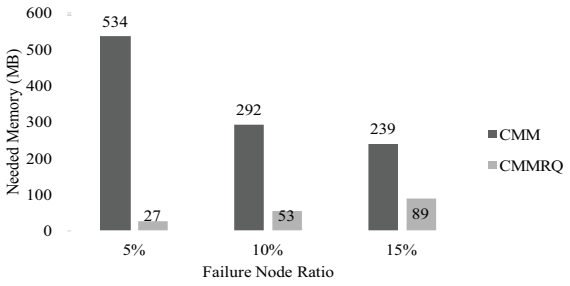

Fig. 13. Required Memory of Master Monitoring Node in Different Failure Node Ratio

Compared with those in centralized monitoring system, the communication traffic and the required memory of the master monitoring node in cooperative monitoring system are much lower. Thus, we can figure out that the cooperative monitoring system can greatly reduce the load of master monitoring node at the cost of a little effect of whole system.

[5] proposes a self-adaptive push model(SAPM) which uses a transportation window to store the collected metrics before being delivered to the monitoring servers. The model is able to decrease the load on a network and achieve a better performance in keeping data coherency between hosts and monitoring servers. We also compare the performance of the two mechanisms, CMMRQ and SAPM (α=1.5, μ=0.5) in decreasing the communication traffic.

Fig.14 shows the communication traffic, and Fig.15 illustrates the required memory of each system respectively when the ratio of data nodes shut down in the system is 5%, 10%, and 15%.

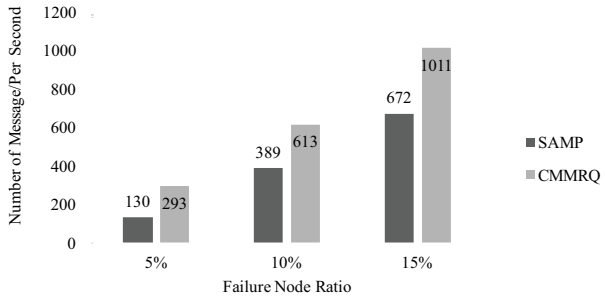

Fig. 14. Communication Traffic of Master Monitoring Node in Different Failure Node Ratio

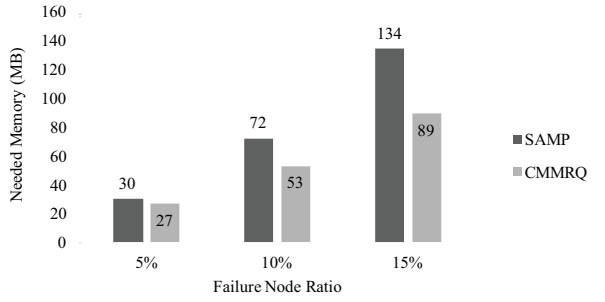

Fig. 15. Required Memory of Master Monitoring Node in Different Failure Node Ratio

Both of the two kinds of monitoring mechanism decrease the load of monitoring node typically. And the communication traffic of cooperative monitoring system is relatively higher than those in centralized monitoring system, since when one node shuts down, its monitoring node would send extra messages to report, However, in SAMP, it needs calculate the value of diff and store the former average value of metrics. As a result, although SAMP has a relatively lower communication traffic in master node, the memory needed is higher than CMMRQ.

Finally, we test the performance of the CMMRQ when the size of cluster increases. Fig.16 shows the communication traffic of CMMRQ with different number of nodes. And the ratio of data nodes shutting down in the system is 5 %.

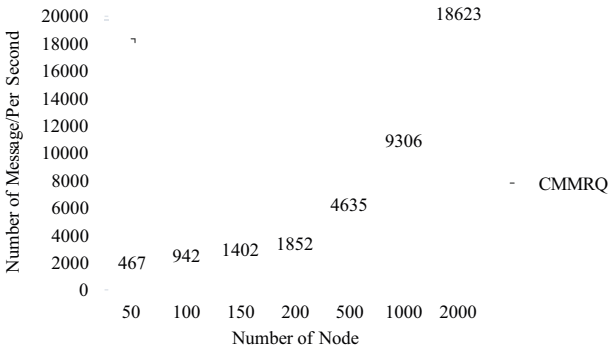

Fig. 16. Communication Traffic of Whole System with Different Number of Nodes

Although CMMRQ significantly decreases the load of the master node, it needs slave nodes to send extra message to reconstruct the cooperative monitoring system. As a result, the communication traffic of the whole system increases. And as the size of the cluster increases, the performance of the whole cluster decreases a little bit.

VI. CONCLUSION

In this paper, we propose a kind of cluster cooperative monitoring mechanism based on message routing queue. In order to reduce the communication traffic and the required memory of master monitoring node, this mechanism utilizes message router to construct the cooperative monitoring relationship between each data node. And this paper not only illustrates the specific process of construction, but also provides different methods to deal with different operating problems. In the end, several simulations have demonstrated the outstanding performances of cooperative monitoring system in reducing the load of master monitoring node.

ACKNOWLEDGMENT

We would like to thank the reviewers for their detailed comments and suggestions throughout the reviewing process that helped us significantly improve the quality of this paper. This work was supported jointly sponsored by the National Natural Science Foundation of China under Grant 61472192 and 61472193, and the Natural Science Fund of Higher Education of Jiangsu Province under Grant 14KJB520014.

REFERENCES

[1] Cloud computing white paper, Telecommunications Research Institute of the Ministry of industry and information technology, 2014.

[2] Yiming Jiang, Julong Lan, and Huiqin Zhou, "Resource Monitoring Policy for Network Virtualization Environment," Journal of Electronics & Information Technology, vol 36, no. 2, pp. 709-714, 2014.

[3] Ying Jiang, Heng Sun, Jiaman Ding, and Yingli Liu, "A Data Transmission Method for Resource Monitoring under Cloud Computing Environment," International Journal of Grid Distribution Computing, vol. 8, no. 2, pp. 15-24, 2015.

[4] Kyoungho An, Subhav Pradhan, Faruk Caglar, and Aniruddha Gokhale, "A Publish/Subscribe Middleware for Dependable and Real-time Resource Monitoring in the Cloud," In SDMCMM'12, 2012.

[5] Kai Lin, Weiqin Tong, Xiaodong Liu, and Liping Zhang, "A self-adaptive mechanism for resource monitoring in cloud computing," In ICSSC'13, pp.243-246, 2013.

[6] Jose M. Alcaraz Calero, Senior Member, IEEE, and Juan Gutierrez Aguado, "MonPaaS: An Adaptive Monitoring Platform as a Service for Cloud Computing Infrastructures and Services," IEEE TRANSACTIONS ON SERVICES COMPUTING, vol.8, no. 1, pp. 66-76, 2015.

[7] Rongheng Lin, Yao Zhao, Budan Wu, and Hua Zou, "An Auto Window Filter Algorithm for Resource Monitoring in Cloud," In 2013 IEEE Sixth International Conference on Cloud Computing, pp. 968-969, 2013.

[8] Lei Xiaojiang and Shang Yanlei, "The Design and Implementation of Resource Monitoring for Cloud Computing Service Platform," In 2013 3rd International Conference on Computer Science and Network Technology, pp. 239-243, 2013.

[9] Massie M L, Chun B N, and Culler D E, "The ganglia distributed monitoring system:design, implementation, and experience," Parallel Computing, vol 30, no.7, pp. 817-640, 2004.

[10] F. Han, J. Peng, and W. Zhang et al, "Virtual resource monitoring in cloud computing," Journal of Shanghai University (English Edition), vol. 15, no. 5, pp. 381‐385, 2011.

[11] Giuseppe Aceto, Alessio Botta, Walter de Donato, and Antonio Pescapè, "Cloud monitoring: A survey," Computer Networks, vol 57, pp. 2093-2115, 2013.

[12] I. Legrand, "MonALISA: an agent based, dynamic service system to monitor, control and optimize distributed systems," Computer Physics Communications, vol 180, no. 12, pp. 2472-2498, 2009.

[13] A. Meera and S. Swamynathan, "Agent based Resource Monitoring system in IaaS Cloud Environment," Procedia Technology, vol 10, pp. 200-207, 2013.