International Journal of Grid Distribution Computing
Vol. 8, No.6, (2015), pp.281-288
http://dx.doi.org/10.14257/ijgdc.2015.8.6.27

A Data Transmission Model based on Adaptive Periodic Push Strategy for IaaS Cloud Computing Platforms

Mingwei Lin[1*] and Zhiqiang Yao[1]

[1]*Faculty of Software, Fujian Normal University, Fuzhou 350108, Fujian, China*
linmwcs@163.com

Abstract

In this paper, a data transmission model is proposed to simulate the way of transmitting resource status information in IaaS cloud computing platforms. An adaptive periodic push strategy is also proposed to improve the performance of the data transmission model. The time interval in the adaptive periodic push strategy is dynamically updated by considering the change degree of resource status information. A series of experiments are conducted on a cloud computing testbed and experimental results show that the adaptive periodic push strategy performs better than previous data transmission strategies in terms of the number of data transmissions and data coherence.

Keywords: *Data transmission model, Push strategy, Pull strategy, Cloud computing*

1. Introduction

With the continuous development and mature of cloud computing technology, more and more enterprises choose to deploy their applications on cloud computing platforms in order to improve the hardware resource utilization and reduce IT operation costs [1]. Virtual machines are core components of cloud computing platforms and applications are deployed in virtual machines [2]. Once virtual machines are abnormal, applications will fail to run normally [3]. Resource monitoring can help system administrators to determine whether virtual machines are abnormal or not [4]. Data transmission is an important part in resource monitoring. It is responsible for transmitting resource status information in cloud computing platforms.

A data transmission model is proposed for IaaS cloud computing platforms in this paper. An adaptive periodic push strategy is also proposed, which dynamically updates the time interval by taking the change degree of resource status information into consideration. A series of experiments are conducted on a cloud computing testbed and experimental results show that the adaptive periodic push strategy performs better than previous data transmission strategies in terms of the number of data transmissions and data coherence.

The remainder of this paper is organized as follows. Section 2 briefly reviews existing data transmission strategies. Section 3 shows a data transmission model for IaaS cloud computing platforms. Section 4 presents an adaptive periodic push strategy. Performance evaluation is described in Section 5. Finally, conclusions are drawn in Section 6.

2. Related Work

In large scale distributed networks, there are two basic strategies that are used to transmit data between nodes, which are the push strategy and the pull strategy, respectively.[1]

Mingwei Lin is the corresponding author.

ISSN: 2005-4262 IJGDC

Existing resource monitoring systems also adopt these two basic strategies to transmit resource status information. For example, the Ganglia open source software uses the push strategy to perform data transmission [5]. The Nagios open source software adopts the pull strategy to collect the host and network information of each physical node [6].

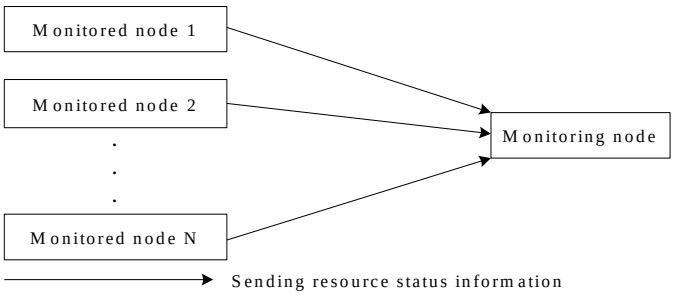

Figure 1. Push Strategy

Figure 1 shows the push strategy. The initiator is the monitored node that pushes resource status information to the monitoring node [7]. There are two different push strategies, which are the periodic push strategy and event-driven push strategy. For the periodic push strategy, monitored nodes periodically push their resource status information to the monitoring node after a time interval. For the event-driven push strategy, monitored nodes push their resource status information when the change degree of resource status information is larger than a predefined threshold. If the time interval or the threshold is too large, the communication overhead will be reduced, but important resource status information will be lost. If the time interval or the threshold is too small, detailed resource status information will be obtained, but the communication overhead will be increased. Therefore, the time interval has a significant influence on the performance of the periodic push strategy and the threshold has a significant influence on the performance of the event-driven push strategy.

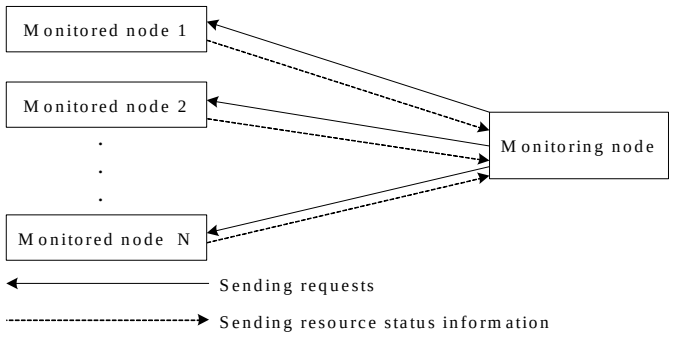

Figure 2. Pull Strategy

Figure 2 shows the pull strategy. The initiator is the monitoring node and it sends requests for obtaining resource status information to monitored nodes and then monitored nodes send resource status information to the monitoring node after receiving the request [8]. There are two pull strategies, which are the periodic pull strategy and the event-driven pull strategy. For the periodic pull strategy, the monitoring node periodically sends the requests to monitored nodes after a time interval and then monitored nodes send resource status information to the monitoring node. For the event-driven pull strategy, the monitoring node sends requests when the change degree of resource status information is larger than a threshold and then monitored nodes send their resource status information to the monitoring node. If the time interval or the threshold is too large, the communication

overhead will be low, but important resource status information will miss. If the time interval or the threshold is too small, detailed resource status information will be obtained, but the communication overhead will be increased.

In order to reduce the communication overhead and improve the data coherence, a number of data transmission strategies have been proposed. Chung *et al*. proposed a time-sensitive mechanism (TSM) to transmit resource status information [9]. The TSM is an periodic push strategy. In the TSM, monitored nodes push resource status information to the monitoring node after a time interval. There are two kinds of time interval, which are the regular time interval and the dynamic time interval. The regular time interval is initialized with a constant value according to the system requirements. However, if the regular time interval is too small, the communication overhead will be increased. If it is too large, important resource status information will be lost and data coherence will be very low. In order to avoid these problems, a dynamic time interval is designed to replace the regular time interval. The dynamic time interval is dynamically updated to the average of time intervals between resource status information changes and the dynamic time interval shows higher data coherence than the regular time interval. Once the resource status information changes, the dynamic time interval will be updated. In the network environments, resource status information changes frequently. In this case, the dynamic time interval will be updated frequently and it will consume massive computing resource.

Sundaresan *et al*. proposed an adaptive polling strategy (APS) to transmit resource status information between the monitored node and the monitoring node [10, 11]. In the APS, the monitoring node periodically sends the request for obtaining resource status information to the monitored node after a time interval and then the monitored node sends its resource status information to the monitoring node after receiving the request. The time interval is initialized with an appropriate value t and it can be dynamically adjusted by using a damping factor d. If the received resource status information is significantly different from the resource status information that is currently maintained by the monitoring node, the time interval is decreased to $t*(1-d)$. If there is not any significant difference, the time interval will be increased to $t*(1+d)$. However, the damping factor d is a constant value. Therefore, the time interval cannot be dynamically adjusted with the change degree of resource status information.

3. Data Transmission Model

In the pull strategy, the monitoring node first sends requests for obtaining resource status information to monitored nodes and then monitored nodes send resource status information to the monitoring node after receiving the request. There are a large number of monitored virtual machines deployed in a cloud computing platform. If the pull strategy is used in the cloud computing platform to transmit resource status information, the monitoring node has to send a large number of requests to monitored virtual machines. It will not only increase the network burden of the cloud computing platform, but also consume massive computing resource of the monitoring node. Therefore, the pull strategy is not suitable for transmitting resource status information in the cloud computing platform. The push strategy often runs on monitored nodes and can be aware of the change of resource status information of monitored nodes. Moreover, it does not incur high communication overhead such as sending massive requests for obtaining resource status information of monitored nodes. Therefore, this paper uses the push strategy as the basic data transmission strategy to transmit resource status information of monitored virtual machines in the cloud computing platform and designs a data transmission model based on the push strategy for cloud computing platforms as shown in Figure 3.

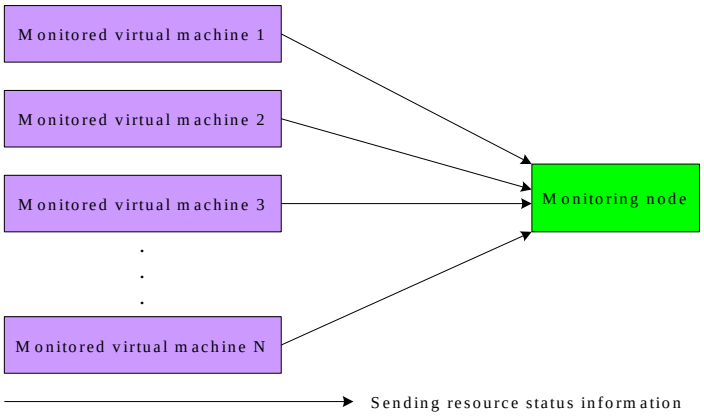

Figure 3. Data Transmission Model

4. Adaptive Periodic Push Strategy

In traditional periodic push strategies, the time interval is a constant. The monitored node sends resource status information to the monitoring node after a time interval. If the resource status information changes infrequently and slightly, a short time interval will send massive useless resource status information and increase the communication overhead. If the resource status information changes frequently and greatly, a long time interval will miss important resource status information and lower the data coherence. In order to achieve a better trade off between the communication overhead and the data coherence, a periodic push strategy called TSM was proposed. In the TSM, the time interval can be dynamically updated as the average of time intervals between resource status information changes. Therefore, TSM can describe the change frequency of the resource status information. However, the dynamic time interval in the TSM does not consider the change degree of the resource status information.

In order to address the problem that dynamic time intervals in existing periodic push strategies do not consider the change degree of the resource status information, this paper proposes an adaptive periodic push strategy for transmitting resource status information in cloud computing platforms. The proposed adaptive periodic push strategy is called APPS and its dynamic time interval considers the change degree of resource status information.

In the APPS strategy, a constant denoted as min_variation is defined to determine whether the resource status information of the monitored virtual machine changes significantly or not.

During the dynamic time interval, if the resource status information of the monitored virtual machine changes and the change degree is larger than min_variation, the dynamic time interval (*DTI*) will be updated as

$$DTI = DTI \times \left(1 - \frac{|A - B|}{max(A, B)}\right) \quad (1)$$

where A is the value of the resource status information after change and B is the value of the resource status information before change.

When the timer expires, the monitored virtual machine sends resource status information to the monitoring node. If the change degree of the resource status information is less than min_variation, the dynamic time interval (*DTI*) will be updated as

$$DTI = DTI \times \left(1 + \frac{1}{e^{|C - P|}}\right) \quad (2)$$

where C is the value of current resource status information of the monitored virtual machine and P is the value of the resource status information that is sent last time.

Figure 4 shows the algorithm pseudo-code of the proposed APPS strategy.


```
Algorithm: Adaptive Periodic Push Strategy

Input:  min_variation; N = 0; T_c presents the current time;
Output: None
1: Push current resource status information;
2: Timer = ∞ ;
3: T_0 = T_c;
4: WHILE(TRUE)
5:      IF The Timer expires
6:          Push current resource status information;
7:          IF |C-P| < min_variation
8:              calculate the new DTI using the equation (2);
9:          ENDIF
10:         Reset Timer to DTI;
11:     ENDIF
12:     IF The resource status information changes
13:         IF N = 0
14:             Push current resource status information;
15:             DTI = T_c - T_0;
16:             N = N + 1;
17:             Reset Timer to DTI;
18:         ELSE IF |A-B| > min_variation
19:             calculate the new DTI using the equation (1);
20:         ENDIF
21:     ENDIF
22:ENDWHILE
```

Figure 4. Algorithm Pseudo-Code of Adaptive Periodic Push Strategy

In existing periodic push strategies, only the TSM considers the change frequency of the resource status information during the time interval. However, the TSM still has two apparent defects. First, once the resource status information changes, the TSM will update the dynamic time interval. If the resource status information changes frequently, but slightly, the TSM will spend a lot of time and computing resource to update the dynamic time interval. Second, the dynamic time interval in the TSM does not consider the change degree of the resource status information. As shown in Figure 4, the APPS strategy uses the equation (1) to update the dynamic time interval only when the change degree of resource status information is larger than min_variation. It effectively reduces unnecessary operations for updating the dynamic time interval. Moreover, the dynamic time interval in the APPS strategy considers the change degree of the resource status information. The larger the change degree of the resource status information is, the shorter the dynamic time interval will be. Therefore, the proposed APPS strategy effectively addresses two defects in the TSM, as well as keeps a high data coherence and reduces the communication overhead.

5. Performance Evaluation

In order to assess the performance of the proposed APPS strategy, a series of experiments are conducted on a private cloud computing platform. The private cloud computing platform consists of two physical nodes, which are a cloud controller and a compute node, respectively. The private cloud computing platform is built by using the Xen [12] and OpenStack open source software [13]. Only one privileged domain runs on the cloud controller, while one privileged domain and three monitored virtual machines run on the compute node.

In order to make the resource status information of monitored virtual machines change dynamically, the RUBis distributed online service benchmark is deployed in each monitored virtual machine [14]. In order to simplify the experiment complexity, only the CPU utilization is selected to represent the resource status information of monitored virtual machines.

An excellent data transmission strategy should have both the low communication overhead and high data coherence. In order to evaluate the effectiveness of the proposed APPS strategy, the number of data transmissions and the data coherence are selected to be evaluation metrics. The number of data transmissions is calculated as the sum of the number of push operations and the number of pull operations. The data coherence is calculated as

$$coh = \frac{\sqrt{\sum_{i=1}^{n} \left(C(i) - R(i)\right)^2}}{n} \tag{3}$$

where $C(i)$ is a point on the graph of the actual measurements and $R(i)$ is a point on the graph of the collected measurements.

The proposed APPS strategy was compared to the APS and TSM strategies. The constant denoted as min_variation is set to 5%.

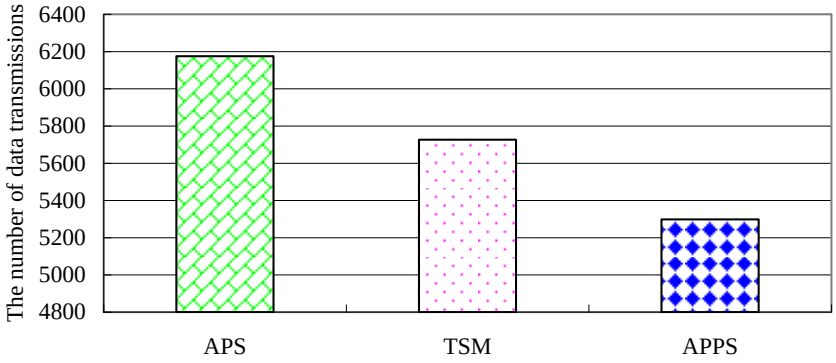

Figure 5. The Number of Data Transmissions

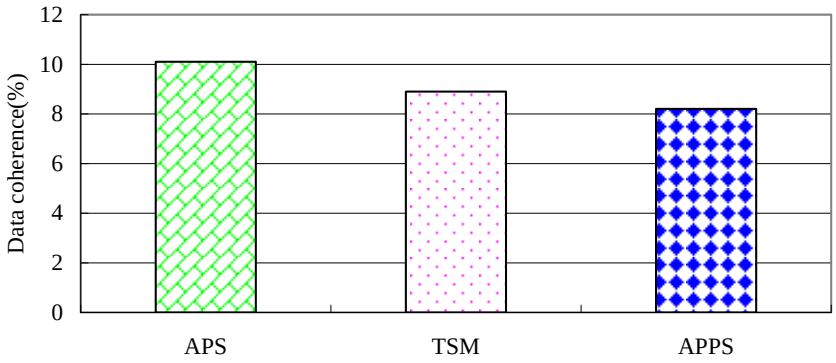

Figure 6. Data Coherence

Figure 5 and Figure 6 shows experimental results of different data transmission strategies in terms of the number of data transmissions and data coherence. The APS strategy is a pull strategy and its dynamic time interval only depends on the damping factor that is a constant. The TSM strategy is a push strategy and its dynamic time interval is dynamically updated as the average of time intervals between the resource status information changes. Therefore, the dynamic time intervals in the APS and TSM strategies cannot be aware of the change degree of resource status information and they may be too small or too large. Our proposed APPS strategy is a push strategy and its time interval is dynamically updated by considering the change degree of resource status information. The dynamic time interval in the proposed APPS strategy will not be too small or too large. Therefore, our proposed APPS strategy is better than other two data transmission strategies in terms of the number of data transmissions and data coherence.

6. Conclusions

A data transmission model that transmits resource status information for cloud computing platforms is proposed in this paper. In order to reduce the communication overhead and improve the data coherence, an adaptive periodic push strategy is also proposed. The time interval in the adaptive periodic push strategy is dynamically updated by considering the change degree of resource status information and then the time interval will not be too small or too large. A series of experiments have been conducted on a private cloud computing platform that is built by using the Xen and OpenStack open source software. Experimental results show that the adaptive periodic push strategy is better than the adaptive polling strategy and the time-sensitive mechanism in terms of the number of data transmissions and the data coherence.

Acknowledgments

The work of this paper is supported by the National Natural Science Foundation of China under grant No.61370078 and No.61402109.

References

[1] A. Atrey, N. Jain, N. Ch S. N Iyengar, "A study on green cloud computing", International Journal of Grid and Distributed Computing, vol. 6, no. 6, (**2013**), pp. 93-102.
[2] H. Goudarzi, M. Pedram, "Energy-efficient virtual machine replication and placement in a cloud computing system", Proceedings of 2012 IEEE 5th International Conference on Cloud Computing, (**2012**), pp. 750-757.
[3] F. Doelitzscher, M. Knahl, C. Reich, N. Clarke, "Anomaly detection in IaaS clouds", Proceedings of IEEE 5th International Conference on Cloud Computing Technology and Science, (**2013**), pp. 387-394.
[4] S. He, M. Ghanem, L. Guo, Y. Guo, "Cloud resource monitoring for intrusion detection", Proceedings of IEEE 5th International Conference on Cloud Computing Technology and Science, (**2013**), pp. 281-284.
[5] K. Bhaduri, K. Das, B. L. Matthews, "Detecting abnormal machine characteristics in cloud infrastructures", Proceedings of 11th IEEE International Conference on Data Mining Workshops, (**2011**), pp. 137-144.
[6] S. A. De Chaves, R. B. Uriarte, C. B. Westphall, "Toward an architecture for monitoring private clouds", IEEE Communications Magazine, vol. 49, no. 12, (**2011**), pp. 130-137.
[7] J. Park, K. Chung, E. Lee, Y. Jeong, H. Yu, "Monitoring service using Markov Chain model in mobile grid environment", Lecture Notes in Computer Science, (**2010**), pp. 193-203.
[8] H. Huang, L. Wang, "P&P: A combined push-pull model for resource monitoring in cloud computing environment", Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing, (**2010**), pp. 260-267.
[9] W. C. Chung, R. S. Chang, "A new mechanism for resource monitoring in Grid computing", Future Generation Computer Systems, vol. 25, no. 1, (**2009**), pp. 1-7.
[10] R. Sundaresan, M. Lauria, T. Kurc, S. Parthasarathy, J. Saltz, "Adaptive polling of grid resource monitors using a slacker coherence model", Proceedings of IEEE International Symposium on High Performance Distributed Computing, (**2003**), pp. 260-269.
[11] R. Sundaresan, T. Kurc, M. Lauria, S. Parthasarathy, J. Saltz, "A slacker coherence protocol for pull-based monitoring of on-line data sources", Proceedings of 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, (**2003**), pp. 250-257.
[12] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, L. Partt, A. Warfield, "Xen and the art of virtualization", Operating Systems Review, vol. 37, no. 5, (**2003**), pp. 164-177.
[13] A. Corradi, M. Fanelli, L. Foschini, "VM consolidation: A real case based on OpenStack cloud", Future Generation Computer Systems, vol. 32, no. 1, (**2014**), pp. 118-127.
[14] E. Cecchet, J. Marguerite, W. Zwaenepoel, "Performance and scalability of EJB applications", Proceedings of the 17th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, (**2002**), pp. 246-261.

Authors

Mingwei Lin, he received his B.S. and Ph.D. degrees from Chongqing University, China, in July 2009. His research interests include NAND flash memory, Linux operating system, and cloud computing. He received the CSC-IBM Chinese Excellent Student Scholarship in 2012.

Zhiqiang Yao, received the PhD degree from Xidian University, China, in 2014. Currently, he is a professor in the Faculty of Software, Fujian Normal University, ACM Professional Membership, Senior Member of China Computer Federation (CCF). His current research interests mainly focus on security in cloud computing, multimedia security.