Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 8 (4): 278-286, 2014
ISSN 1819-4311 / DOI: 10.3923/jse.2014.278.286

DRMST: A Novel Dynamic Replica Management Strategy Based on TOPSIS in Cloud Storage

[1,3]Yang Zhiyong, [1,2]Li Chunlin, [1]Liu Yanpei and [1]AL-Gabri Malek
[1]Department of Computer Science, Wuhan University of Technology, China
[2]State Key Laboratory for Novel Software Technology, Nanjing University, China
[3]Key Laboratory of Fiber Optic Sensing Technology and Information Processing, Ministry of Education, China

Corresponding Author: Yang Zhiyong, Department of Computer Science, Wuhan University of Technology, China

ABSTRACT

The present study proposes a dynamic replica management strategy based on Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) in the cloud storage system. The replica management strategy includes the replica placement algorithm and the replica number strategy. The replica placement algorithm sorts the nodes by TOPSIS according to the node performance and load, the replicas are placed in the nodes with high performance and low load. The replica number strategy designs the reliability model and the availability model to compute the number of replicas, the initial number of the replicas is decided according to the reliability model and the number of the replicas is adjusted dynamically according to the availability model. Experimental results show that the strategy in the study has better performance in response delay and load-balance than the Default Replica Management Strategy in Hadoop (DRMSH) distributed file system.

Key words: Replica management, cloud storage, load balance, TOPSIS

INTRODUCTION

In recent years, huge amounts of data are generated in the fields such as scientific and engineering applications, management of such huge data resources has been a great challenge (Mansouri and Dastghaibyfard, 2012). Data replication is an important technique to manage large data in cloud storage system, which can reduce the bandwidth consumption and reduce the job execution time, the general idea of replication is to place, replicas of data at various locations. There are two key issues in all the data replication algorithms. How many replicas should be created and where replicas should be stored.

How many to replicate? (The number of the replicas): The number of the replicas is decided by the importance and popularity of the data. The data is more important, the reliability requirement is higher. The popularity of data easily leads to node overload, increasing the number of the replicas can reduce the response delay and balance loads in hotspot nodes. So, the number of the replicas is concerned with the reliability and availability.

Where to place the replicas? (Replica location): The replica location is an important factor to affect the system performance. Since different nodes have different processing capabilities in the

heterogeneous cloud storage system and node load affects the availability of the data, therefore, it should be considered comprehensively between the node performance and load to decide the replica location.

In this study, two algorithms are proposed, a novel Dynamical Replica Placement Strategy based on TOPSIS (DRPST) and a Dynamical Replica Number Strategy (DRNS). The DRPST aims to reduce the response time and balance load, which takes into account the node performance and load comprehensively and uses TOPSIS method to sort the nodes, the top n nodes are chose to place the replicas, DRNS designs the reliability model and the availability model, which computes the initial number of the replicas according the reliability model and dynamically adjusts the number of the replicas according to the availability model.

RELATED WORK

The static replication strategy is adopted in HDFS (Shvachko *et al*., 2010), in which the number of the replicas can be assigned by the client and the default is 3. Two replicas are placed in the different data nodes of the same rack nearby the client and the third is placed in a random data node of the other rack. If the number of the replicas is more than 3, the other replicas are placed in random data nodes of the cluster. The static replication keeps stationary replica location and replica number, which can't adapt to the complex network environment.

Li *et al*. (2011) presented a novel cost-effective data reliability management mechanism which proactively checks the availability of replicas for maintaining data reliability. Bansal *et al*. (2011) presented dynamic replication management architecture which achieves high performance and dependability by placing the replicas uniformly over all distant nodes, but these replica management mechanism and architecture doesn't take into account the load balance.

Bsoul *et al*. (2011) proposed a dynamic replication strategy that takes into account the number and frequency of requests, the size of the replica and the last time the replica was requested. Shorfuzzaman *et al*. (2010) proposed Popularity Based Replica Placement strategy in a hierarchical data grid which is guided by file "popularity". It places replicas close to clients to decrease data access. These replica strategies place replicas just according to the "popularity", but they don't take into account the node performance in the heterogeneous cloud storage system.

Zhang *et al*. (2010) presented a replication approach based on swarm intelligence, which was an adaptive and decentralized bottom-to-up method. Mendez and Carballeira (2006), Munoz and Carballeira (2007) and Munoz *et al*. (2010) applied Particle Swarm Optimization and Ant Colony Optimization to replica creation and data grid replica selection. Ma *et al*. (2013) propose replica creation strategy based on quantum evolutionary algorithm in data grid, these algorithms predict according to the history information, but it will take much cost to process the amount of the history information.

Tang and Xu (2005) and Wei *et al*. (2009) proposed QoS-aware replica placement strategy to meet the QoS of every request, but these strategies doesn't pursuit the optimal system performance.

DYNAMIC REPLICA PLACEMENT STRATEGY BASED ON TOPSIS

Replica placement is one of the important issues for cloud storage system, which deal with how to locate and deploy the replicas. The cloud storage system is composed of m independent heterogeneous data nodes, denoted d_1, d_2, d_3,..., d_m, and n replicas of a file will be stored in the different data nodes. To reduce the response delay and balance the load, the nodes with high

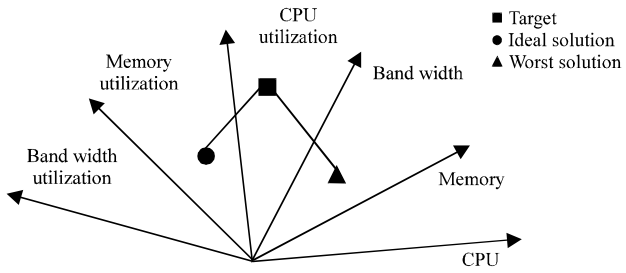

Fig. 1: Principle of TOPSIS

Table 1: Index in TOPSIS

	Index					
Node	CPU	Memory	Bandwidth	CPU utilization	Memory utilization	Bandwidth utilization
1	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
2	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}
3	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}
...	...	...	...	...	...	...
m	x_{m1}	x_{m2}	x_{m3}	x_{m4}	x_{m5}	x_{m6}

performance and low load are chose to store the replicas. As shown in Table 1, high performance is reflected in high CPU frequency, big memory, big bandwidth and low load is reflected in low CPU utilization, low memory utilization and low bandwidth utilization.

Therefore, the replica placement is a classical multiple attribute decision making-question. The TOPSIS method (Hwang and Masud, 1979) is a good solution to the problem, which sorts the objects according to the distance from the target to the ideal solution and the worst solution. If the target is closest to the ideal solution and farthest to the worst solution, then it is best, otherwise it is worst. The ideal solution is that every index value achieves the best value of each index and the worst solution is that every index value achieves the worst value of each index. The principle of TOPSIS is shown in Fig. 1.

The sorting steps of TOPSIS are as follow:

- Generate the decision matrix D and the weight matrix W:

$$D = \begin{bmatrix} x_{11} & x_{12} & \cdots & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & \cdots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{k1} & x_{k2} & \cdots & \cdots & x_{km} \end{bmatrix} \quad W = \begin{bmatrix} w_1 & 0 & \cdots & \cdots & 0 \\ 0 & w_2 & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & w_k \end{bmatrix}$$

where, x_{ij} is the jth index of the node d_i, w_i is the weight of the ith index:

- Calculate the standard decision matrix R:

$$R = (r_{ij})_{k*m} \quad (1)$$

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{k} x_{ij}^2}} \quad 0 < i \le k,\ 0 < j \le m \tag{2}$$

- Calculate the weighted decision matrix V:

$$V = (v_{ij})_{k*m} = W*R \tag{3}$$

- Determine the ideal solution A^+ and the worst solution A^-:

$$A^+ = \{v_1^+, v_2^+, \cdots, v_m^+\},\ A^- = \{v_1^-, v_2^-, \cdots, v_m^-\} \tag{4}$$

$$v_j^+ = \begin{cases} \max v_{ij} & j \in I \\ \min v_{ij} & j \in J \end{cases} \quad v_j^- = \begin{cases} \min v_{ij} & j \in I \\ \max v_{ij} & j \in J \end{cases} \tag{5}$$

Where:

I: Index of CPU, memory and bandwidth.
J: Index of CPU utilization, memory utilization and bandwidth utilization:

- Calculate the distance from target to the ideal solution s^+ and the distance from target to the worst solution s^-:

$$s_j^+ = \sqrt{\sum_{i=1}^{k} (v_{ij} - v_j^+)^2} \quad s_j^- = \sqrt{\sum_{i=1}^{k} (v_{ij} - v_j^-)^2} \quad 1 \le j \le m \tag{6}$$

- At last, calculate the relative distance from target to the ideal solution and the worst solution:

$$d_j = \frac{s_j^+}{s_j^+ + s_j^-} \quad 1 \le j \le m \tag{7}$$

The smaller d_j is, the better the performance of the node and more balance the load of the node. So, the top n nodes with the smallest d-value should be chose to store the replicas.

DYNAMIC NUMBER STRATEGY BASED ON THE RELIABILITY AND AVAILABILITY

Model of reliability: The minimum replica number is computed according to the model of reliability, the reliability value varies from 0-1 according to the file's importance, the more important the file is, the bigger the file's reliability value is.

Assume that a file is divided into m blocks with the same size and every block has n replicas, every replica of the block B_i should be placed in different node N_{ij} ($1 \le j \le n$).

Some definitions are used in the model of the reliability as follow.

Node unavailable = A node is unable to be accessed for the failure of the machine or network
P($\bar{N}$) = The probability of node unavailable

Block unavailable = All nodes that store the same block replicas are unable to be accessed
P($\overline{B}$) = The probability of block unavailable
File unavailable = At least one block of the file is unavailable
P($\overline{F}$) = The probability of file unavailable:

$$\begin{aligned} P(B_i) &= 1 - P(\overline{N_{i1}}) * P(\overline{N_{i2}}) * \ldots * P(\overline{N_{in}}) \\ &= 1 - \prod_{j=1}^{n} P(\overline{N_{ij}}) = 1 - \prod_{j=1}^{n} (1 - P(N_{ij})) \end{aligned} \tag{8}$$

$$\begin{aligned} P(F) &= P(B_1) * P(B_2) * \ldots * P(B_m) \\ &= \prod_{i=1}^{m} P(B_i) = \prod_{i=1}^{m} (1 - \prod_{j=1}^{n} (1 - P(N_{ij}))) \end{aligned} \tag{9}$$

In order to simplify the computation, it is assumed that the online rate of every node is p, the file's availability can be simplified by Eq. 10:

$$\begin{aligned} P(F) &= \prod_{i=1}^{m} (1 - \prod_{j=1}^{n} (1 - P(N_{ij}))) \\ &= \prod_{i=1}^{m} (1 - (1-p)^n) = (1 - (1-p)^n)^m \end{aligned} \tag{10}$$

The file is more important, the requirement to the reliability is higher. The reliability parameter fac should be set according to the importance. The initial number of the replicas can be computed according to the reliability model as follow:

$$P(F) = (1 - (1-p)^n)^m \geq fac \tag{11}$$

$$n \geq \log_{(1-p)} (1 - \sqrt[m]{fac}) \tag{12}$$

where, n is the minimum number of the replicas to ensure the file's reliability.

Model of the availability: The accesses to the data accord with the Zipf's law (Adamic and Huberman, 2002), that is, 80% visits focus on the 20% files, the hot files result in a long response delay. To reduce the response delay and meet the requirement to the availability, increasing the number of the replicas is a good way to the problem. The load is the key factor to affect the response delay, so the model of availability is designed according the average load.

Assume that a file has k replicas and the replicas are placed in the nodes denoted $\{d_1, d_2, \ldots, d_k\}$, because the node load ld_i concerns with CPU utilization C_i, memory utilization M_i and bandwidth utilization B_i, so the node load ld_i is computed by Eq. 13:

$$ld_i = w_1 * C_i + w_2 * M_i + w_3 * B_i \tag{13}$$

where, w_i is the weight value for each factor.

The suitable number of the replicas is computed according to the availability model (Eq. 14):

$$N = \left\lceil \frac{\sum_{i=1}^{n} ld_i}{\beta} \right\rceil \tag{14}$$

where, β is the ideal load to ensure nodes stability, $0.5 \le \beta \le 0.6$.

if $k<N$, it means, the nodes are overload, the availability of the file is not enough to meet the requirement of the clients. So, the additional N-k replicas should be created, the adding replicas will be placed to the nodes with DRPST.

So, DRNS determines the initial number of the replicas according to the reliability model; for the hot files, DRNS dynamically adjusts the number of the replicas according to the availability model.

EXPERIMENT AND ANALYSIS

Hadoop is used as the experiment platform for the open source. Seven computers are used in the experiment, one computer is as the name node, one computer is as the client and the other five computers are as the data nodes. The structure is shown in Fig. 2.

The computers are with the same configuration. CPU: Intel(R) core(TM) i7-3770 3.4 GHz, Memory: 4.0 GB, Hard disk: 320 GB, OS: 64-bit Cent OS 5.6 with Linux 2.6.18.8 kernel. The other parameters are set as $w_1 = 0.3$, $w_2 = 0.3$, $w_3 = 0.4$, $\beta = 0.5$.

Experiment 1 (Relationship between the reliability and the number of the replicas): The reliability of a file is concerned with the number of the blocks m, the number of the replicas n and the online rate p. The file's reliability is calculated by Eq. 8. As shown in Fig. 3, the file's reliability increases with the growth of the number of the replicas. If p is bigger than 0.8, the reliability is usually bigger than 90% when the number of the replicas is bigger than 3, so the initial number of the replicas can be calculated according to the file's importance, the more important the file is, the bigger the file's reliability value and the initial number of the replicas are bigger.

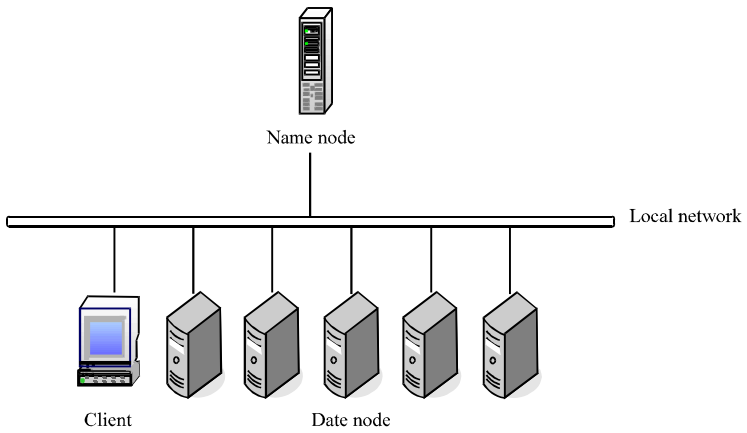

Fig. 2: Structure of the hadoop platform

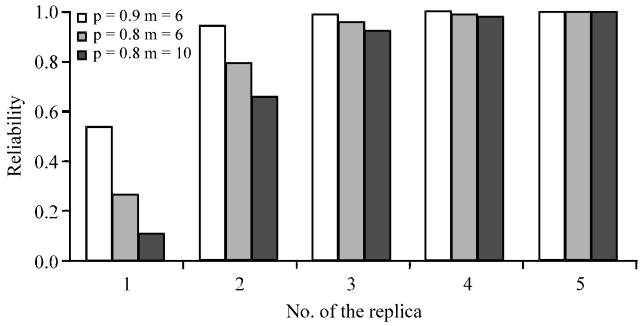

Fig. 3: Reliability vs. the No. of the replicas

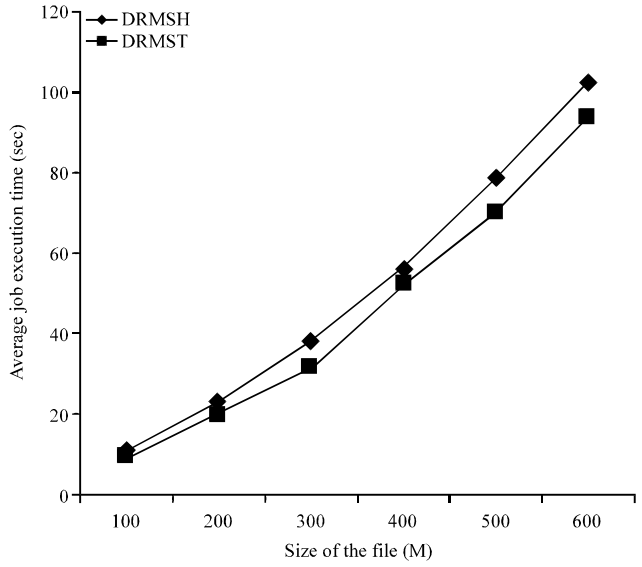

Fig. 4: Delay contrast between DRMST and DRMSH

Experiment 2 (Uploading delay contrast between DRMST and DRMSH): The delay is tested by uploading files from the client and the file size varies from 100-600 M.

As shown in Fig. 4, DRMST spends less time than DRMSH when the client uploads files to the date nodes. Because in the DRMST, the node with high performance and low load has good ability in process, however, DRMSH places replicas to a random node, so the DRMST has shorter delay in uploading the files and the delay is expended with the growth of the file size.

Experiment 3 (Load contrast between DRMST and DRMSH): By uploading files many times in different situation, we record the loads of the replica nodes and calculate the average load in DRMSH and DRMST.

As shown in Fig. 5 because DRMSH places replicas in a random way, DRMSH has bigger fluctuation and value in the average load. Whereas low load is taking into account in the DRMST, so the node is with less load value in most time, DRMST can keeps the load balance better.

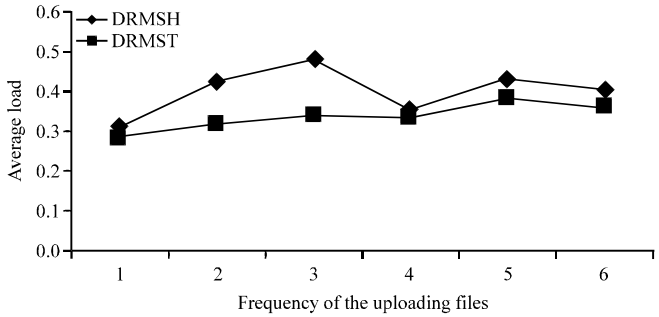

Fig. 5: Load contrast between DRMSH and DRMST

CONCLUSION

The study has researched the replica management technology and proposed an adaptive replica management strategy. The replica placement and the replica number are the most important problems in the dynamic replica management. The DRMST uses the TOPSIS method to sort the nodes according to the performance and load and the top n nodes are chose to place the replicas in sequence, the node with high performance and low load is the preferred target to choose. The reliability and availability of the file increases with the growth of the number of the replicas, DRMST computes the initial number of the replicas according to the reliability and adjusts the number of the replicas according to the availability, the experiment results prove that DRMST is better in response delay and load balance than DRMSH.

ACKNOWLEDGMENTS

This study was supported by the National Natural Science Foundation (NSF) under grants (No.61171075), Special Fund for Fast Sharing of Science Paper in Net Era by CSTD (FSSP) No. 20130143110021, Program for the High-end Talents of Hubei Province, Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 20120143110014 and the Open Fund of the State Key Laboratory of Software Development Environment (SKLSDE-2013KF). Any opinion, findings and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

REFERENCES

Adamic, L.A. and B.A. Huberman, 2002. Zipf's law and the internet. Glottometrics, 3: 143-150.

Bansal, S., S. Sharma and I. Trivedi, 2011. A dynamic replica placement algorithm to enhance multiple failures capability in distributed system. Int. J. Distrib. Parallel Syst., 2: 79-85.

Bsoul, M., A. Al-Khasawneh, E.E. Abdallah and Y. Kilani, 2011. Enhanced fast spread replication strategy for data grid. J. Network Comput. Appl., 34: 575-580.

Hwang, C.L. and A.S.M. Masud, 1979. Multiple Objective Decision Making Methods and Applications: A State of the Art Survey. Springer-Verlag, Berlin, Germany, ISBN-13: 9780387091112, Pages: 351.

Li, W., Y. Yang and D. Yuan, 2011. A novel cost-effective dynamic data replication strategy for reliability in cloud data centres. Proceedings of the IEEE 9th International Conference on Dependable, Autonomic and Secure Computing, December 12-14, 2011, Sydney, Australia, pp: 496-502.

Ma, T., Q. Yan, W. Tian, D. Guan and S. Lee, 2013. Replica creation strategy based on quantum evolutionary algorithm in data gird. Knowledge-Based Syst., 42: 85-96.

Mansouri, N. and G.H. Dastghaibyfard, 2012. A dynamic replica management strategy in data grid. J. Network Comput. Appl., 35: 1297-1303.

Mendez, V. and F.G. Carballeira, 2006. PSO vs. ACO, Data Grid Replication Services Performance Evaluation. In: Frontiers of High Performance Computing and Networking-ISPA 2006 Workshops, Min, G., B. Di Martino, L.T. Yang, M. Guo and G. Runger (Eds.). Springer, Berlin, Heidelberg, pp: 833-843.

Munoz, V.M. and F.G. Carballeira, 2007. PSO-grid data replication service. Proceedings of the 7th International Conference on High Performance Computing for Computational Science, June 10-13, 2006, Rio de Janeiro, Brazil, pp: 656-669.

Munoz, V.M., G.A. Vicente, F.G. Carballeira and J.S. Cairols, 2010. Emergent algorithms for replica location and selection in data grid. Future Gener. Comput. Syst., 26: 934-946.

Shorfuzzaman, M., P. Graham and R. Eskicioglu, 2010. Adaptive popularity-driven replica placement in hierarchical data grids. J. Supercomput., 51: 374-392.

Shvachko, K., H. Kuang, S. Radia and R. Chansler, 2010. The hadoop distributed file system. Proceedings of the 26th IEEE Symposium on Mass Storage Systems and Technologies, May 3-7, 2010, Incline Village, NV., pp: 1-10.

Tang, X. and J. Xu, 2005. QoS-aware replica placement for content distribution. IEEE Trans. Parallel Distrib. Syst., 16: 921-932.

Wei, F., X. Nong and L. Xicheng, 2009. A survey for QoS-aware replica placement problem. J. Comput. Res. Dev., 46: 36-43.

Zhang, P., K. Xie, X. Ma, X. Li and Y. Sun, 2010. A replication strategy based on swarm intelligence in spatial data grid. Proceedings of the 18th International Conference on Geoinformatics, June 18-20, 2010, Beijing, China, pp: 1-5.