Journal of Theoretical and Applied Information Technology
28th February 2013. Vol. 48 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

LBGR: A LOAD BALANCING P2P FILE STORAGE SYSTEM BASED ON GROUPING AND REPUTATION

[1]SONG GUANGHUA, [2]WU ZHIXING AND [3]YANG BOWEI

[1]School of Aeronautics and Astronautics, Zhejiang University, Hangzhou, CHINA

Email: ghsong@zju.edu.cn , 21024038@zju.edu.cn , boweiy@zju.edu.cn

ABSTRACT

The expansion of the Internet leads to the rapid growth of information, which brings about urgent needs for rapid, efficient and reliable mass storage systems. In this paper, we present LBGR, a load balancing P2P file storage system based on grouping and reputation. LBGR evaluates a node by calculating its reputation value, and adopts a mechanism based on grouping and virtual node to solve the load balance problem which exists in distributed hash table structured systems. In LBGR, groups can balance their loads according to their capacities by migrating virtual nodes. The experimental results show that the presented load balance algorithm works well, and is comparable with the centralized algorithm, while it requires less load information of the system.

Keywords: *Glusterfs; Reputation; Virtual Node; Load Balance; Storage*

1. INTRODUCTION

With the rapid development of the Internet, the demand of storage space is growing. At the same time, there are a lot of free storage and computing resources on the Internet. How to make full use of these idle storage space and computing power to construct a file storage system based on the P2P technologies has become a hot topic in the distributed storage research area.

In this paper, we present LBGR, a novel stable and reliable file storage system based on grouping and reputation. It can be deployed on large-scale and highly dynamic P2P networks.The remainder of the paper is organized as follows: Section 2 discusses the related works. Section 3 introduces the system structure. Our load balancing algorithm is presented in Section 4. We evaluate LBGR in simulations, and the simulation results are discussed in Section 5. Finally, we conclude our study in Section 6.

2. RELATED WORK

Based on the concept of virtual nodes [1], [2, 3] promote load balance algorithms by migrating the virtual nodes. In [4], the load balance algorithm needs to acquire the load information of the whole system, so a large amount of data needs to be updated and exchanged. As the centralized algorithms may introduce single point of failure, [5, 6] try to balance the loads of nodes by the centralized algorithms which rely on some dedicated nodes. With the performance information of neighbor nodes in logic, [7] presents a locality-aware randomized (LAR) algorithm to balance the loads of nodes. Nodes in [8] try to deduce the global load information and then compute the load which nodes must undertake.

The incentive model based on the reputation system and calculation of the weights of nodes are also important for the design of a new system. Based on Hadoop, paper [9] tries to select the most excellent nodes to store files by calculate the weights of the nodes according to their capabilities, storage space, CPU utilization and online time. Paper [10] presents an incentive model through the resource reputation rating algorithm and the reputation incentive rating algorithm, and proves that it works well in P2P reputation system to prevent malicious attacks and incent effective voting.

3. SYSTEM OVERVIEW

3.1 System Structure

Figure 1 depicts the structure of LBGR. Unlike [11] where nodes are grouped by network distance, nodes in LBGR are grouped by their IDs which are computed by the DHT(distributed hash table) algorithm. Group1, Group2, and Group3 are three groups in the system; every group should select a super node by the performance of the nodes. Node1, Node2 and Node3 are selected as super nodes for

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

these three groups respectively. Like GlusterFS, the hash address space in LBGR is equally divided into sections. We denote each section as a virtual node. Each group holds several continuous virtual nodes in the hash address space according to their capacities.

As shown in Figure 1, Group1 holds n1 continuous virtual nodes, denoted as VG_{11},VG_{12},…, and VG_{1n1}. Group2 holds n2 continuous virtual nodes, denoted as VG_{21},VG_{22},…,and VG_{2n2}. Group1 and Group2 are adjacent, so are their virtual nodes. The reputation values of the nodes are calculated according to their performance, and one of the most excellent nodes in each group will be selected as the super node. The function of the super node is presented as follows:

1) to store and update profiles of the groups and their virtual nodes;

2) to store and update the reputation values of the nodes;

3) to store and update the locations of the files.

We denote the ID of the super node as the ID of the group. When a node wants to join the system, it needs to select which group to enter. We define the selecting strategy as follows: the node compares its ID with all groups, and joins the nearest group.

When a node wants to leave the system, the super node just deletes the information about the node. Unlike GlusterFS, through the grouping mechanism, there will not have any significant impact on the file storage system when nodes join or leave the system, and it will be more efficient when groups need to merge or divide.

When peers need to access a file, LBGR calculates the ID of the file using the DHT algorithm, and then figures out which virtual node it belongs to. With the help of the super node, the system will find out which group it belongs to. Then the location information will be returned to the peer who wants to access this file. If peers want to store files, the groups will select a suitable node as the storage node based on reputations of nodes.

3.2 Reputation System

Nodes in P2P networks are heterogeneous; therefore, the reputations of nodes differ greatly. LBGR introduces the reputation system to help the file storage system to be more reliable and stable.We consider that online time, network bandwidth, storage space and node performance are very important factors to measure the reputation value of a node. Each group will update the reputation values of the nodes regularly. The reputation value will be updated as follows:

$$\mathrm{REP}_{n+1} = k \times REP_n + (1-k) \times t \times b \times s \times p \quad (1)$$

Where REPn+1 denotes the reputation value of a node during period n+1, t is the online time of the node, b is the network bandwidth, s is the available storage space, and p is the CPU performance of the node. We define k as a parameter that measures the weights of previous reputation of a node, similar with that in [12].

With the help of the reputation system, a set of nodes with high reputations can be selected as candidates of the super node, as in [13]. The important information on the super node is copied to the candidates. A new super node will be selected from the candidates immediately when the super node leaves the system.

4. LOAD BALANCE ALGORITHM

Due to the drawbacks of the DHT algorithm, traditional DHT structured systems cannot balance the loads of the nodes effectively. In LBGR, we balance the loads according to the capacities of the groups by migrating virtual nodes.

Assume that there are three groups in the system, namely GroupA, GroupB and GroupC, where $ID_A < ID_B < ID_C$. The starting number and the ending number of these three groups about the virtual nodes are listed in table 1.

Table1. Groups And Their Virtual Node Numbers

Group	Starting Number	Ending Number
GroupA	S_a	S_{b-1}
GroupB	S_b	S_{c-1}
GroupC	S_c	S_{d-1}

The load limit of a node is determined by the capability of the node. So the load limit G of a group is calculated as follows:

$$G=\sum_{i=1}^{n} V_i \quad (2)$$

Where n denotes the number of nodes in the group, and V_i represents the load limit of node i in the group.Each group holds a certain number of virtual nodes, so we denote $R=\sum_{i=1}^{m} L_i$ as the real load of a group where m denotes the number of the virtual nodes in the group, and L_i represents the real load of virtual node i in the group.We denote the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

average load utilization rate of GroupA, GroupB and GroupC as μ , which is calculated as follows:

$$\mu = \frac{R_a + R_b + R_c}{G_a + G_b + G_c} \quad (3)$$

Where R_a, R_b *and* R_c represent the real load of GroupA, GroupB and GroupC respectively, and G_a, G_b *and* G_c represent the load limit of GroupA, GroupB and GroupC, respectively.We also denote $\mu_a = \frac{R_a}{G_a}$, $\mu_b = \frac{R_b}{G_b}$, and $\mu_c = \frac{R_c}{G_c}$ as the load utilization rate of GroupA, GroupB and GroupC, respectively.

4.1 Algorithm Sketch

As depicted previously, we denote μ as the average load utilization rate of the node and its two neighbor groups.We define $\alpha = \mu + \Delta$ as the load balance threshold, where Δ is a predefined system parameter. A group will try to balance its load if its load utilization is larger than α .We define $\beta = \mu + \Delta/2$. A group will terminate the load balance algorithm if its load utilization is smaller than β .We define γ as a lower threshold for the load balance algorithm. If the load utilization of a group is smaller than γ , the load balance algorithm will never start. We consider the whole address space as annularity, which means that the virtual node with the maximum address number and the virtual node with the minimum address number are neighbors.

Each group must be mutually exclusive with its neighbor groups when computing their loads by the load balance algorithm. We assume that the utilization of GroupB is larger than α ; and it needs to balance its load. The process that GroupB gets the right to balance its load is presented as follows:

```
typedef enum { normal, overload, loadbalancing } group_state; // three states of groups
void test(int i){// LEFT(i) is the left neighbor group of i
if ( state[i] == overload &&state[LEFT(i)] != loadbalancing &&state[RIGHT(i)] !=
loadbalancing){// RIGHT(i) is the right neighbor group of i
                    state[i] = loadbalancing;
                    V(s[i]);// Each group has a semaphore, the initial value of s[i] is 0
    }
}
void get_rights(int i){
          P(mutex);// the initial value of mutex is 0
          state[i] = overload;
          test(i); // try to get the right
          V(mutex);
          P(s[i]); // blocked if cannot get the right
}
void put_ rights(int i){
          P(mutex);
          state[i]= normal;
          test(LEFTI(i)); //inquire left neighbor if it needs to balance its load
          test(RIGHT(i)); //inquire right neighbor if it needs to balance its load
          V(mutex);
}
void trytobalance(B){
          get_ rights (B);
          loadbalance(B);
          put_ rights (B);
}
```

When GroupB gets the right to balance its load, load balance algorithm will be applied. The load balance algorithm consists of part1 and part 2, which are presented below.

Journal of Theoretical and Applied Information Technology
28th February 2013. Vol. 48 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

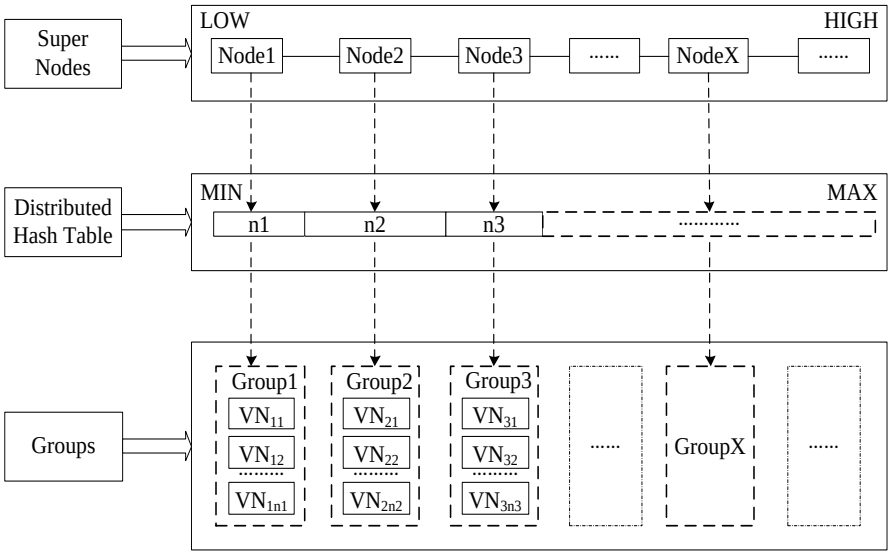

Figure 1 Structure Of LBGR

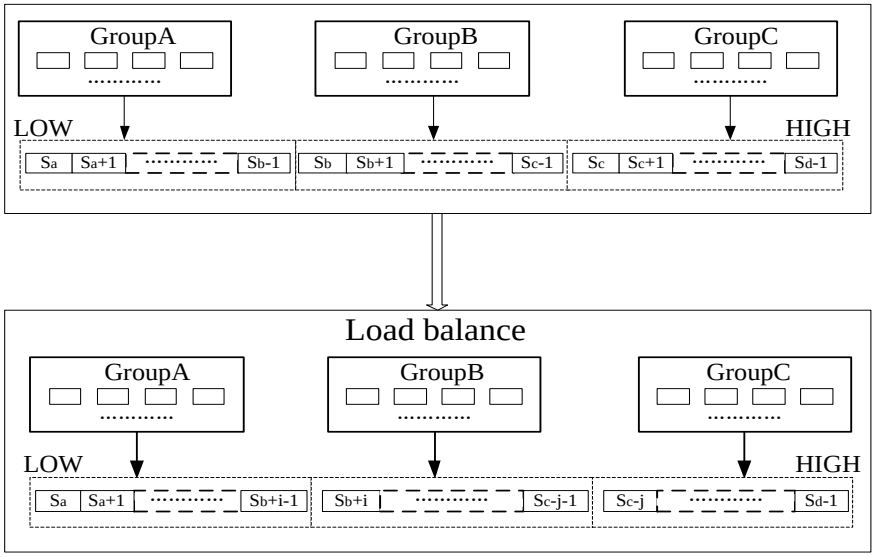

Figure 2 Load Balancing Of LBGR

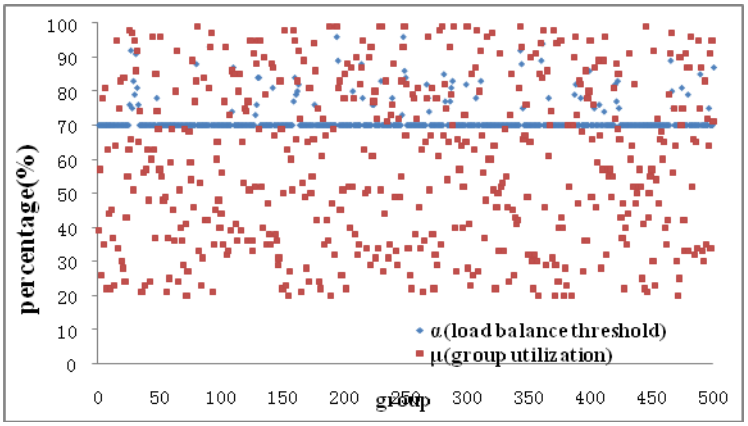

Figure 3 α And μ Before Load Balance

Journal of Theoretical and Applied Information Technology
28th February 2013. Vol. 48 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

As shown in part1 and part2, the value of i means the number of virtual nodes which need to be migrated from GroupB to GroupA, and the value of j is the number of virtual nodes which need to be migrated from GroupB to GroupC. The files that are stored on these virtual nodes also need to be migrated to the target groups. The movement of the virtual nodes is depicted in Figure 2.

As the movement of the virtual nodes may lead to the change of α of some groups which are close to the virtual nodes, LBGR adopts a delayed movement strategy: LBGR will not move those files which need to be migrated immediately, link files will be built in the corresponding groups which link to the actual locations of those files; those files will be migrated after an appointed time when neighbor groups are stable.

Part 1:
if $\mu_a < \mu$
  for ($i = 1; i \leq Sc\text{-}Sb; i{+}{+}$)
    $R_a = R_a + L_{Sb+i-1}$; $\mu_a = \frac{R_a}{C_a}$
    $R_b = R_b - L_{Sb+i-1}$; $\mu_b = \frac{R_b}{C_b}$
    if $\mu_a < \mu$ *and* $\mu_b > \beta$
      continue
    else if $\mu \leq \mu_a \leq \beta$ *or* $\mu_b \leq \beta$
      break
    else if $\mu_a > \beta$
      $i--$
      break
    end if
  end for
end if

Part 2:
if $\mu_a < \mu$ *and* $\mu_b > \beta$
  for ($j = 1; j \leq Sc\text{-}Sb\text{-}I; j{+}{+}$)
    $R_c = R_c + L_{Sc-j}$; $\mu_c = \frac{R_c}{C_c}$
    $R_b = R_b - L_{Sc-j}$; $\mu_b = \frac{R_b}{C_b}$
    if $\mu_c < \mu$ *and* $\mu_b > \beta$
      continue
    else if $\mu \leq \mu_c \leq \beta$ *or* $\mu_b \leq \beta$
      break
    else if $\mu_c > \beta$
      $j--$
      break
    end if
  end for
end if

5. SIMULATIONS

In this section, we simulate 500 groups on LBGR , and 50000 files are deployed by the davies_meyer[14] algorithm. We set $\Delta = 2$ and $\gamma = 70$. The value of α (load balance threshold) and μ (group utilization) which will be calculated by each group are presented in Figure 3. We can figure out that many groups need to balance their loads because of their high loads.

After adopting our load balance algorithm, groups with high loads can balance their loads effectively. Most of the group utilization percentages are under 70, which can be shown in Figure 4.

By taking several factors into account, GlusterFS can balance the loads of subvolumes within a volume, and can eliminate the need for regular tuning of the file system to keep volume load nicely balanced [15]. However, it cannot balance the load between different volumes.

The loads of the subvolumes in GlusterFS can be balanced with the load information of the whole volume. Because the load balance capability of GlusterFS is similar with the centralized algorithm, we compare our load balance algorithm with the centralized algorithm.

We simulate 100 groups and 50000 files on LBGR and set $\Delta = 2$, and set $\gamma = 70$. In the centralized algorithm, we compute the average utilization φ of the system, and then get the estimated load $\omega = \varphi + \Delta$; if $\omega < 70$, we set $\omega < 70$. The experimental results are shown in Figure 5. The result that is balanced by our algorithm is comparable with the result that is balanced by the centralized algorithm while the load information we need is much less than that the centralized algorithm needs.

6. CONCLUSIONS

In this paper, we present LBGR, a novel load balancing distributed file storage system on the P2P network. With the information provided by super nodes, nodes in LBGR can access files accurately and quickly. In this way, LBGR eliminates the impact of metadata which can restrict the scalability of the traditional distributed file systems, and nodes in LBGR are grouped to improve the stability of the system. With the load balance algorithm, groups can balance their loads by migrating virtual nodes

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

effectively. By applying the reputation system, nodes can be reasonably selected as super nodes, candidates of super nodes and storage nodes, to undertake corresponding jobs. The experimental results show that our load balance algorithm works well, and is comparable with the centralized algorithm.

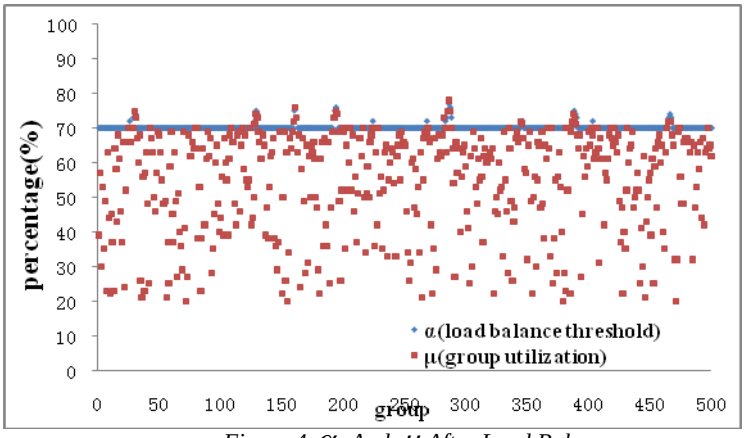

Figure 4 α And μ After Load Balance

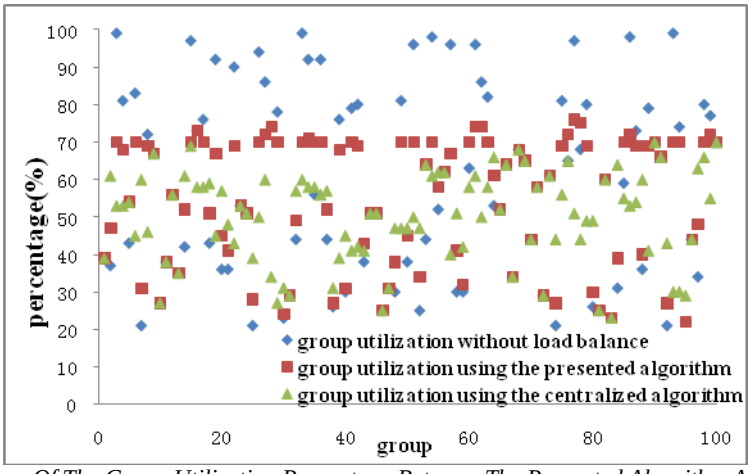

Figure 5 Comparison Of The Group Utilization Percentage Between The Presented Algorithm And The Centralized Algorithm

ACKNOWLEDGEMENTS

This work is supported by the Science and Technology Department of Zhejiang Province, China, under grant No. 2009C14031. We would like to thank the center for engineering and scientific computation, Zhejiang University, for its computational and storage resources.

REFERENCES

[1] Frank Debek, Frans Kaashoek, David Karser, Robert Morris and Ion Stoica, Wide-area Cooperative Storage with CFS, in *Proc.ACM SOSP*, 35(5):202-215,2001.

[2] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, Load Balancing in Structured P2P Systems, Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS '02), pp. 68-79, 2003.

[3] B. Godfrey, K. Lakshminarayanan, S. Surana, R.Karp and I. Stoica, Load Balancing in Dynamic Structured P2P Systems, INFOCOM , pp.2253-2262,2004

[4] Y. Zhu and Y. Hu, Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems, IEEE Trans. Parallel and Distributed Systems, 16(4): 349-361, 2005.

[5] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I.Stoica, Load Balancing in Dynamic Structured peer-to-peer Systems, Performance Evaluation, 63(6):217-240, 2006.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[6] C. Chen and K.C. Tsai, The Server Reassignment Problem for Load Balancing in Structured P2P Systems, IEEE Trans. Parallel and Distributed Systems, 12(2): 234-246, 2008.

[7] H. Shen and C.-Z. Xu, Locality-Aware and Churn-Resilient Load Balancing Algorithms in Structured P2P Networks, IEEE Trans. Parallel and Distributed Systems, 18(6): 849-862, 2007.

[8] H. Hsiao, H. Liao, S. Chen and K. Huang, Load Balance with Imperfect Information in Structured Peer-to-Peer Systems, IEEE Trans. Parallel and Distributed Systems, 22(4): 634 - 649 , 2011.

[9] Song Guang-hua, Chuai jun-na, Yang Bo-wei and Zheng Yao, QDFS: A Quality-Aware Distributed File Storage Service Based on HDFS, Computer Science and Automation Engineering (CSAE), pp.203-207, 2011.

[10] Bowei Yang, Guanghua Song and Yao Zheng, An incentive model for voting based on information-hiding in P2P networks, Journal of Zhejiang University: Science A, 11(12):967-975, 2010.

[11] Yang Lei, Huang Hao, Li Ren-fa, Li Ken-li, Composite P2P Storage System Based on Group Management, computer science, 37(1):64-67, 2010.

[12] Xiaoning Jiang and Lingxiao Ye, Attack-resistant Techniques in P2P Reputation Systems, Networking and Digital Society (ICNDS), pp.390-393, 2010.

[13] Liu Yumei, Yang Shoubao, Chen Wangming, Guo Leitao, Wei Dong, The research of the Reputation-Aware SuperNode Selection Algorithm in P2P system, Journal of the Graduate School of Chinese Academy of Sciences, 25(2):197-203, 2008

[14] J. Black, P. Rogaway and T. Shrimpton, Black-Box Analysis of the Block-Cipher-Based Hash-Function Construction from PGV, In Advances in Cryptology – CRYPTO '02, volume 2442 of Lecture Notes in Computer Science, pp.320-335, 2002.

[15] Information on http://www.gluster.org