Dynamic Load Balancing in Skip Graph

TAKESHI MORIAI
Saitama University
Department of Computer Science
Saitama 338-8570
JAPAN
mt@g.yolab.jp

ANDRII ZHYGMANOVSKYI
Saitama University
Department of Computer Science
Saitama 338-8570
JAPAN
andrew@g.yolab.jp

NORIKO MATSUMOTO
Saitama University
Department of Computer Science
Saitama 338-8570
JAPAN
mats@g.yolab.jp

NORIHIKO YOSHIDA
Saitama University
Department of Computer Science
Saitama 338-8570
JAPAN
yo@g.yolab.jp

Abstract: In this paper, we propose a Skip Graph adapting to the increase or decrease of contents. Skip Graph is one form of P2P networks that can perform flexible content search efficiently. However, it has no mechanism to cope with overload to a peer when the amount of its contents overwhelms. Meanwhile, there is another form of P2P networks named P-Ring. It has also a mechanism for flexible content search as well as dynamic load balancing. Therefore, we propose applying a dynamic load balancing method inspired from P-Ring to Skip Graph. In our proposed method, the network of Skip Graph is dynamically reorganized, and the contents are dynamically transferred among peers according to the increase or decrease of contents in a peer. We also present some simulation-based experiments to confirm the usefulness and efficiency of our proposed method.

Key–Words: Peer-to-peer, Load-balancing

1 Introduction

A P2P (peer to peer) network is attracting much attention as a distributed contents-sharing system. A client-server system is the most pervasive structure because of its simple construction and management on its centralized server. However, the server can be a single point of failure, so that decentralized contents-sharing, namely P2P, has been studied actively from the point of view of availability and load balancing. In a P2P network, each peer has roles both as a server and a client, and each content is maintained on a different peer, therefore P2P can work even if some of peers disappear. A large number of efficient methods for P2P content queries have been proposed, especially range search is one of the fascinating topic.

Skip Graph [1] is one of the effective methods to achieve the range search. This method is an overlay network application of Skip List [2] which is a data structure for efficient data access. This approach has advantages that it can perform the range search and achieves join/leaving operations at small costs. However, as the number of contents increases, some particular peers may get overwhelmed by high load. In order to address this issue, we propose a scalable dynamic load balancing method for Skip Graph, and show the efficiency through some simulation experiments.

In the rest of this paper, we show some related works and P2P networks in Section 2, and we explain the purpose of our study in Section 3. Then, we make a detailed description of our proposed method in Section 4. Finally, we present some simulation-based experiments and their results to confirm the usefulness of the proposed method in Section 5.

2 Related Works

P2P networks is roughly classified into unstructured P2P networks and structured P2P networks. For example, Gnutella [4] and Freenet [5] are in the former. they are easy to construct, however they have a drawback that search messages are overflowed, because they spread query packets in a way of bucket brigade, called Flooding. On the other hand, Chord [6], CAN [7], Kademlia [8], Tapestry [9] and Pastry [10], for example, are in the latter. They use DHT (Distributed Hash Table) in both phase of network construction

and search. This approach can search efficiently using hash tables, although there are two problems. Firstly, each peer cannot perform flexible search. Secondly, management of networks incurs a large cost in general, and it is very difficult to construct a network.

Consequently, some attempts to enable range search without using hash tables have been studied actively. Skip Graph is one of the most acknowledged methods. This approach can access the target contents using range search without hash tables efficiently, and makes nodes to join or leave from a network at a small cost. However, Skip Graph has an issue that the load balancing is out of concern when the number of contents are increased. Meanwhile, there is another proposal named P-Ring [3], which can perform range search on P2P overlay networks and realize dynamic load balancing using content transfer similar to B+tree. Therefore, we aim to address the issue of Skip Graph using the technique of P-Ring.

2.1 Skip Graph

2.1.1 Structure

Figure 1 shows a schematic diagram of Skip Graph. Skip Graph is an overlay network based on Skip List over a P2P network to perform range search. Each peer in the Skip Graph are arranged based on the key order. Here, the keys are comparable value that are assigned to identify themselves uniquely. And each peer has mutual links with peers before it and after it. Here, these peers are called the left peer and right peer, respectively. A peer is given a random integer value in the binary representation which is called a membership vector, and each peer maintains some links at multiple levels based on the matching of membership vector prefix. Namely, linked list at each level contains all the peers which have the matching membership vector prefix.

A Peer in Skip Graph is one-to-one correspondence to key in principle. Multi-Key Skip Graph [11] is proposed in order to solve the problem. In this method, peers with same membership vector are considered as a peer. As a result, a physical peer can have multiple key. However in this paper, we will proceed to the discussion with assuming that a physical peer can have consecutive multiple key.

Skip Graph has been attracted attention in recent years. For example, The Rainbow Skip Graph [12] and Skip B-Trees [13] are proposed. These methods are focused search improvement.

2.1.2 Join and Leave Operation

A new peer knows some introducing peer in the network that will help it to join the network. The new peer makes linked lists based on membership vector at each level and inserts itself in one linked list at each level till it finds itself in a singleton list at the topmost level.

When a peer wants to leave the network, it informs its left peer at each level to update its right peer pointer to point the peer's right peer. It starts at the topmost level and works its way down to level 0.

As described above, peers in Skip Graph notify only to left peer and right peer when a peer performs join operation or leave operation. Therefore, Message cost of join or leave operation in Skip Graph is relatively small.

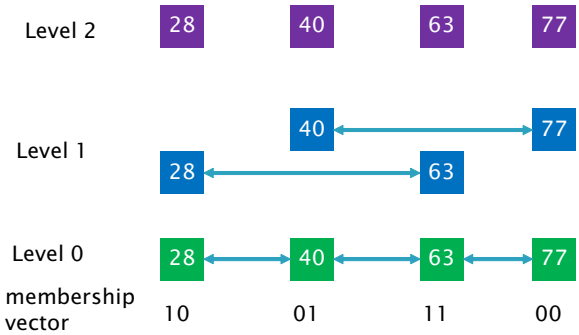

Figure 1: Skip Graph

2.2 P-Ring

A P-Ring network is formed in a circle. Some peers are included in the network, each of which has contents in a certain range respectively, while the other peers are not, which are called helper peers. A newcomer peer joins the network as a helper peer. There are a lower bound and an upper bound for the number of contents in each peer. The upper bound is set twice as the lower bound. When the number of the contents exceeds the upper bound, the peer splits its set of contents and corresponding range, and gives a half to a helper peer. After that, the peer invites the helper peer to join the ring as its right peer. When the number of the contents gets lower than the lower bound, right peer checks whether a redistribution of contents is possible between the peer and itself. If yes, right peer gives some of its contents and the corresponding range to the peer. If redistribution is not possible, right peer gives up its all contents and its range to the peer, and becomes a helper peer. P-Ring does dynamic load balancing in this manner.

For efficient search for contents in P-Ring, each peer has routing table. To create a routing table, they use an tunable value *d* in P-Ring. At the lowest level in routing table, level 1, each peer maintains a list of the *d* successors on the ring, skipping up to *d* - 1 peers at a time. At level 2, we again maintain a list of d successors. However, a successor at level 2 corresponds to the *d* th successor at level 1, skipping up to d^2 - 1 peers at a time. Namely At level *l*, a peer has link to peers that are d^l peers away.

However, each peer in P-Ring needs to update own routing table when a peer joins or leaves. Therefore, there is a problem that message cost by update is great.

3 Load Balancing

A P2P network can distribute the network load to each peer. However, if a single peer has a huge number the contents, the load is concentrated to this peer, and the advantages of the P2P network is lost. Therefore, dynamic load balancing following the transition of the network load is important.

Most studies on Skip Graph focuses on search improvement, however almost none has been done on load balancing because there is an assumption that a peer and a content has one-to-one correspondence. In real deployments, this assumption can not be held, and a peer may have several contents.

Consequently, we have an idea of applying the dynamic load balancing technique in P-Ring to Skip Graph.

4 Proposed Method

For dynamic load balancing in Skip Graph, we introduce the dynamic load balancing technique in P-Ring. The first is the helper peers. A newcomer does not join to the network as in the original Skip Graph, but is included as a helper peer.

The second is the split and merge operations. We introduce an upper limit and a lower limit to the number of contents in a peer. In this paper, the upper limit is double of the lower. If the number of contents exceeds the upper limit, the peer splits the contents in cooperation with a helper peer. If the number of contents gets less than the lower limit, the peer merges the contents in cooperation with its left neighbor peer.

Transfer of contents between peers is not taken into account in the original Skip Graph.

We can perform the split and merge operations in a Skip Graph without destructing its key order property. A receiver peer finds the smallest key in the received contents, and sets it as its own key. Below, their details are presented.

4.1 Split Operation

If the number of contents in a peer exceeds the upper limit, the peer performs the split operation to distribute its load. Figure 2 shows a Skip Graph with helper peers, in which the upper and lower limits for peers are 2 and 4, respectively.

The Figure3 shows an example of the split operation when the content 64 is added to 2. The peer which originally has a key of 63 transfers the second half of its contents to any helper peer. The helper peer which receives the contents performs the join operation to the Skip Graph, assigning the minimum key in the contents as its own key (In this case, 66).

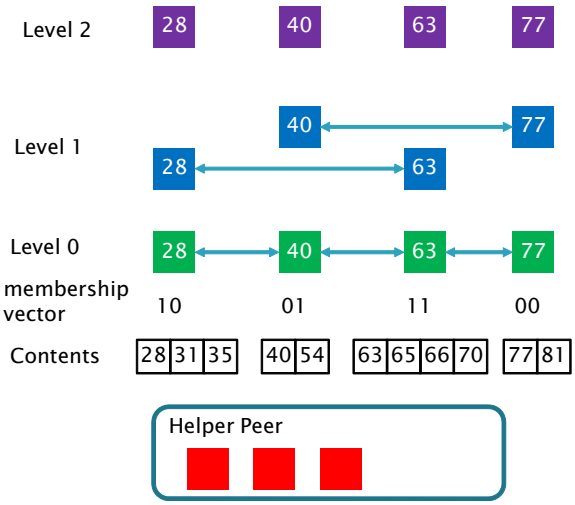

Figure 2: Skip Graph with helper peers.

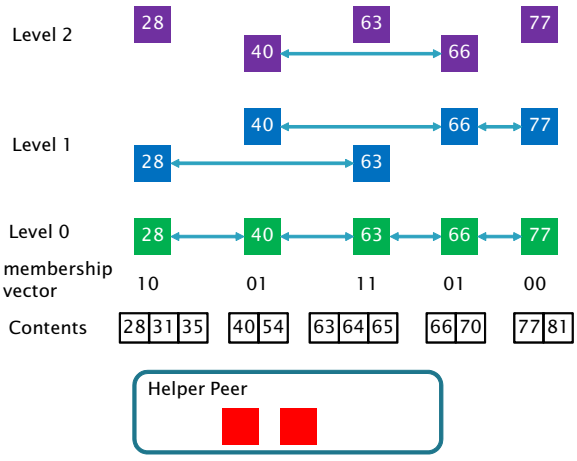

Figure 3: Split operation.

4.2 Merge Operation

If the number of contents in a peer becomes less than the lower limit, the peer performs the merge operation. There are two cases according to the sum of the numbers of contents in the peer and in the left neighbor peer.

(a) If the sum is less than the upper limit
The peer transfers all of its contents to the left peer. Then it leaves from the Skip Graph performing the original leave operation, and becomes a helper peer.

Figure 4 shows an example of the merge operation when content 40 is removed from Figure 2. The sum number of contents in the left peer (its key is 28) and the peer (its key is 40) are 4, which is below the upper limit. Therefore, the peer transfers all the contents to the left peer.

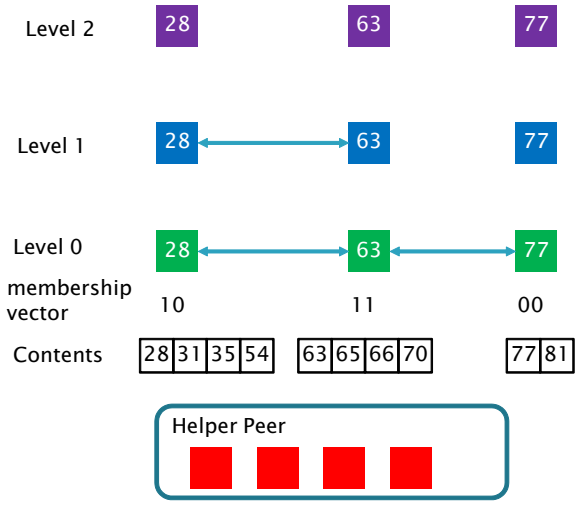

Figure 4: Merge operation (the case 1).

(b) If the sum is more than the upper limit
The left peer performs the split operation first. The peer receives the second half of the contents from the left peer, and sets the smallest key in the received contents as its own key.

Figure 5 shows an example of the merge operation when the content 81 is removed from Figure 4. The sum numbers of contents in the left peer (its key is 63) and the peer (its key is 77) is 5, which exceeds the upper limit. Therefore, the left peer performs the split operation, and transfers the second half of its contents to the peer. The peer sets its key to 66.

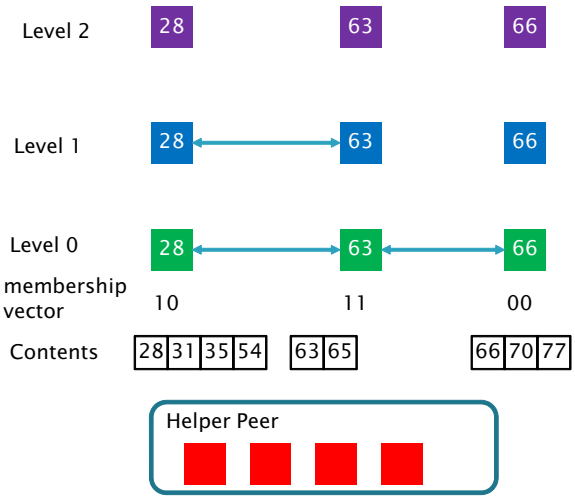

Figure 5: Merge operation (the case 2).

5 Experimental Evaluation

We conducted experiments using a simulator in order to confirm the usefulness of the proposed method. We present the details and results of the experiments below.

5.1 Experiments in detail

We measure some standard deviations of the number of contents on each peer along with the increase of contents, and compare them with the ones on the original Skip Graph. This is to measure the effect of dynamic load balancing. In addition, we compare the network traffic, to measure the cost of load balancing.

As the networks become different each time according to the set of contents to add and its order, we take an average of several simulations. The number of peers is c.a. 10,000, and the number of contents is 75,000, 150,000, 225,000, and 300,000. We do simulations 10 times for each of the four cases, and get their averages. The lower bound of the number of contents in a peer is 5, 10, 15, and 20 in each case. A content has a unique and random key, and is added to the network at some fixed interval.

We conducted two cases, namely addition only, and addition and removal. In the latter, contents are added or removed at a fixed interval. The probability of addition is 0.8, and removal is 0.2. The lower bound for the number of contents in a peer is 10.

5.2 Results

Figure 6 shows the comparison of standard deviations between the original Skip Graph without load balancing, and the network with load balancing following

our proposal. Standard deviations of peer loads are almost one-fourth in our proposed method compared to the original Skip Graph. Figure 7 shows the comparison of the total numbers of network messages between them. In the original Skip Graph, the standard deviation is significantly increased as the contents are added. On the contrary, the standard deviation in our proposed network is suppressed. From Figure 7, we see that the network traffic in our network is greater than the original Skip Graph, However, this overhead is negligible.

From these results, we see that our proposed method achieves dynamic load balancing efficiently as compared to the original Skip Graph. Although the network traffic is slightly increased, we can conclude that our proposed method is efficient in dynamic load balancing.

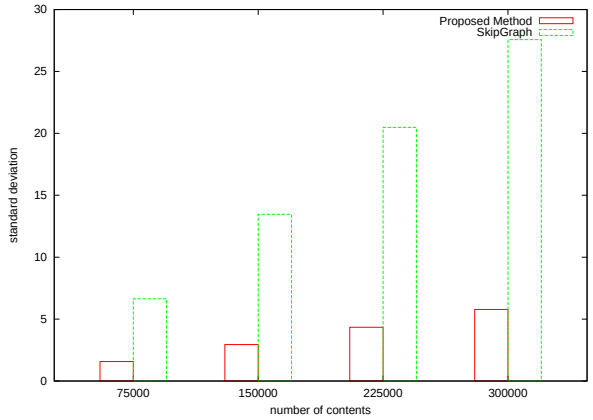

Figure 6: Comparison of standard deviations.

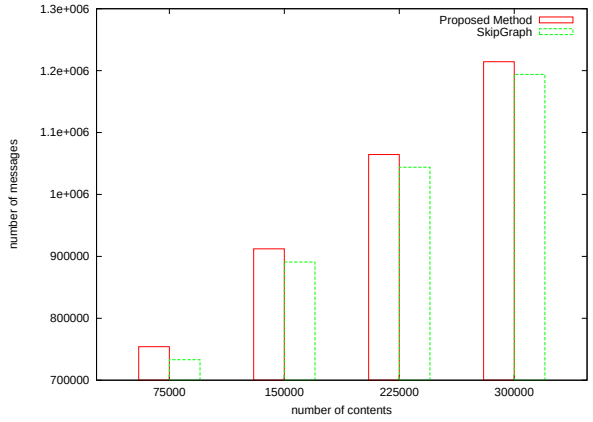

Figure 7: Comparison of the numbers of packets.

Figure 8 shows the transition of the standard deviations in the cases of addition only, and Figure 9 shows the transition of the standard deviations in the cases of addition and removal.

In both the cases, the standard deviations vary significantly at the initial stage because the number of peers is small. However, they get reduced and gradually stable afterward due to the effect of dynamic load balancing.

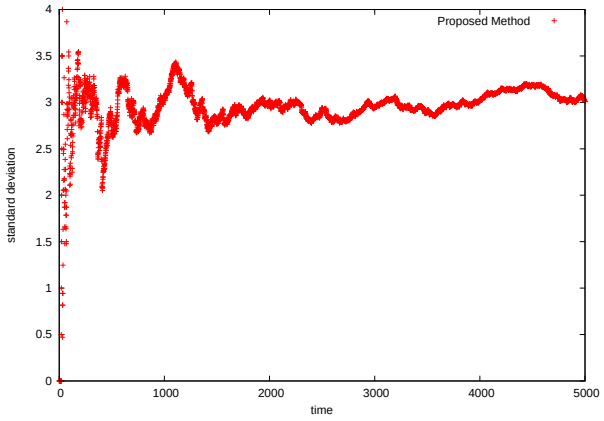

Figure 8: Standard deviation of addition only.

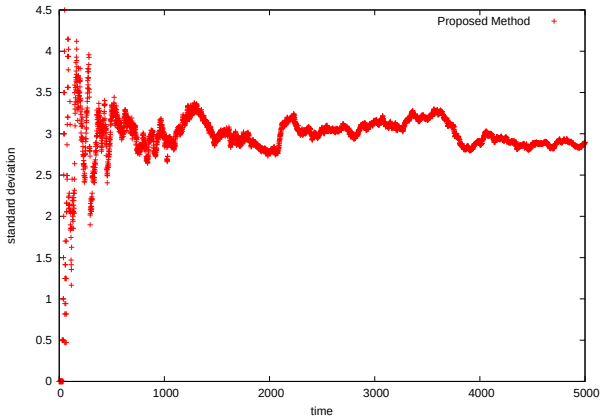

Figure 9: Standard deviation of addition and removal.

6 Conclusion

We proposed a method for dynamic load balancing in Skip Graph by moving contents between peers and by peer joining/leaving in response to the increase or decrease of contents in the network. In order to confirm usefulness of the proposed method, we performed simulations to measure the changes in variability of the number of contents along with adding or removing contents at random to Skip Graph. Standard deviations of peer loads are almost one-fourth in our proposed method compared to the original Skip Graph. From the results, we confirmed that the proposed method is effective.

Load on a P2P network does not depend solely on the number of content. It is affected by the popularities of contents as well. Therefore, we must consider the popularities of contents for more sophisticated load balancing. This issue will be addressed in the future.

References:

[1] J.Aspnes, G.Shah, “Skip Graphs”, ACM Trans. on Algorithms(TALG) Volume 3 Issue 4, November 2007 Article No.37

[2] Pugh, William (June 1990), “Skip lists : a probabilistic alternative to balanced trees”, Communications of the ACM 33 : 668-676

[3] A.Crainiceanu, P.Linda, A.Machanavajjhala, J.Gehrke, J.Shanmugasundaram, “P-Ring : An Efficient and Robust P2P Range Index Structure”, Proc. ACM SIGMOD Conf. 2007:223-234, 2007

[4] Gnutella. http://gnutella.wego.com/.

[5] I.Clarke, O.Sandberg, B.Wiley, and T.W.Hong. “Freenet : A Distributed Anonymous Information Storage and Retrieval System. ” In Workshop on Design Issues in Anonymity and Unobservability, pages 311-320, July 2000. ICSI, Berkeley, CA, USA.

[6] I.Stoica, R.Morris, D.Karger, F.Kaashoek, and H.Balakrishnan,“Chord : A scalable PeerToPeer lookup service for internet applications”, in Proceedings of the ACM SIGCOMM, 2001, pp.149-160.

[7] S.Ratnasamy, P.Francis, M.Handley, R.Karp, and S.Shenker. “A Scalable Content-Addressable Network”, In Proc. of the ACM SIGCOMM 2001 Conference, August 2001.

[8] P.Maymounkov and D.Mazieres. “Kademlia : A Peer-to-peer Information System Based on the XOR Metric. ” In Proceedings of the First International Workshop on Peer-to- Peer Systems (IPTPS ’02), MIT, March 2002.

[9] B.Zhao, J.Kubiatowicz,and A.Joseph, “Tapestry : An infrastructure for fault-tolerant wide-area location and routing”, University of California at Berkeley, Computer Science Department, Tech. Rep. UCB/CSD011141, 2001.

[10] P.Druschel and A.Rowstron, “Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems”, in Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms(Middleware 2001), Nov. 2001.

[11] Y. Konishi, M. Yoshida, Y. Teranishi, K. Harumoto, and S. Shimojo, “A Proposal of a Multikey Extension of Skip Graph”, IPSJ SIG Notes, Vol. 2007 No. 58, pp.25-30, June 2007

[12] M. T. Goodrich, M. J. Nelson, and J. Z. Sun. “The Rainbow Skip Graph : A Fault-Tolerant Constant-Degree Distributed Data Structure”, SODA ’06, 2006

[13] I.Abraham, J.Aspens, J.Yuan, “Skip B-Trees ”, Principles of Ditributed Systems; 9th International Conference, OPODIS 2005; Pisa, Italy; December 2005; Revised Selected Papers, December 2005, pp. 366-380.