Evaluation of the Structured Bloom Filters Based on Similarity

Hiroshi Sakuma Fumiaki Sato

Department of Information Science, TohoUniversity
2-2-1 Miyama, Funabashi, 274-8510, Japan
Phone & Fax: +81-47-472-8029
E-mail: fsato@is.sci.toho-u.ac.jp

Abstract

The Bloom filter is a data structure which the characteristics of contents is expressed by a bit pattern. The Bloom filter has become attractive as one of the methods of looking up data in the distributed system. Since two or more keywords can be used in the method based on the Bloom filter, it is more flexible than the distributed hash table (DHT). Especially, the key aspect of the structured Bloom filter is that the number of query forwarding is fixed. In the previous research, the ring of the Bloom filter, which is like Chord ring, was proposed. However, the node in the ring must maintain many filters corresponding to the number of nodes. In this paper, we propose the tree structured Bloom filter and evaluated the performance of the query forwarding. We also propose similarity-based tree management method which is effective to reduce the cost of reconstruction of the tree. To evaluate our method, the size of filter information and the number of average hops of query forwarding in this research are compared to the existing research. The tree reconstruction cost is also evaluated by simulation.

1. Introduction

For looking up data in P2P, the method based on the distributed hash table (DHT) is actively researched [1, 2, 3, 4]. In the method with DHT, the load of the network is smaller than that of pure P2P to which the query is flood. Moreover, the load of the server can be widely distributed in the method compared with hybrid P2P. However, it is difficult with DHT to look up data by two or more key words, and there is a problem of costing for maintenance. On the other hand, the method of looking up data by using the Bloom filter is researched so far [5, 6, 7, 8].

The Bloom filter is a bit string obtained by applying two or more hash functions to the key word, considering the obtained value to be a position of the bit string, and setting the bit at the position to one [6]. The Bloom filter can integrate all filters of contents accumulated on the site by OR operation. And, it can be decided whether the site includes the contents by AND operation of the integrated filters and filters for the query.

The resource on the distributed systems can be efficiently looked up by forwarding the query using the Bloom filter. One of the methods to change the direction where the query is forwarded with the Bloom filter of the networked site is the attenuated Bloom filter [7]. In the attenuated Bloom filter, each node manages a table of the Bloom filters. The node can know the hops from the node to the target node in the network by exchanging the table of Bloom filters with the adjacent node. And it can forward the query to the adequate adjacent node. However, this method cannot fix the upper bound of the number of query forwarding.

Chord algorithm [4] which is the one of the DHT is used for the method for the query forwarding in the Bloom filter [10]. In this method, the number of hops of query forwarding is the same as Chord, and $O(\log_2 N)$. Here, N is a number of nodes. The number of Bloom filters which the node should manage is also $O(\log_2 N)$.

In this research, it proposes a new looking up method of the Bloom filter based on a B-tree. In this method, the number of query forwarding becomes $O(\log_m N)$ and the amount of the managed Bloom filters for each node becomes $O(1)$. The tree management method based on the similarity of Bloom filter is also proposed. Similarity-based approach is effective to reduce the cost of reconstruction of the tree.

The rest of this paper is organized as follows. In the section 2, background and related works are described. Our proposal, Bloom filter management method based on a B-tree is explained in the section 3. In the section 4, similarity-based tree management method is

1550-445X/11 $26.00 © 2011 IEEE
DOI 10.1109/AINA.2011.97

explained. Comparison between the previous method and our method is described in section 5. Section 6 describes conclusions and future works.

2. Related works

2.1. Bloom filter

The Bloom filter is a data structure where it is expressed that certain content exists on the site by using some hash functions and a bit string of constant length.

First of all, the bit string in n bit is prepared to compose the Bloom filter, and all bits are initialized to "0". Next, k hash values of the content are calculated by k hash functions and the content. And, the bit at the position which corresponds to each hash value is changed into "1". To judge the presence of the content by using the Bloom filter, the bit of the Bloom filter at the position corresponding to the hash value of the content which wants to be looked up is examined. It can be judged that the content exists if all the corresponding bits are "1". However, there is a possibility to judge that the content exists by the collision of the hash value though the content does not exist. Oppositely, judging that the content does not exist though the content exists won't happen.

The Bloom filter has the feature of obtaining the Bloom filter which expresses all contents included in each Bloom filter by applying the OR operation to two or more Bloom filters. At this time, the bit length of the Bloom filter does not change regardless of the number of contents. In this research, the key word of contents is expressed by using the Bloom filter. The key word is expressed regardless of the number of key words by the same bit length, and expression of two or more key words of the content becomes possible. Fig. 1 is an example of expressing the key word of contents with the Bloom filter.

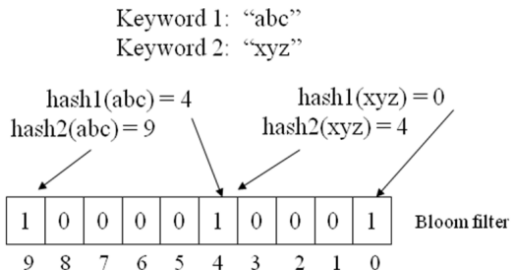

Figure 1. An example of the Bloom filter.

2.2. Bloom filter with the Chord structure

In usual distributed hash table (DHT), a content ID is generated from one or some key words, and the content is added to the node corresponding to the content ID according to the P2P protocol of each DHT. Content ID is generated from a query key word to look up contents, and the contents are looked up by the ID according to the P2P protocol of each DHT. On the other hand, the content ID is not used in this research when contents are added or deleted. You may add it to any node in the network (However, it is preferable that arrangement is distributed because of the load-balancing). The key word of contents is represented in the Bloom filter. The filter is the consolidated Bloom filter according to the network structure of DHT. Therefore, the retrieval by the comparison of Bloom filters based on the P2P protocols is enabled.

Hereafter, it explains details according to Chord which is one of DHTs [8][10].

The node which manages contents generates the Bloom filter from all key words of managed contents. As a result, the Bloom filter which expresses the all key words of contents that the node manages can be created. This will be called Node Bloom filter (henceforth NBf). The key word of contents can be looked up by referring to NBf of each node. However, referring to NBf of all nodes is inefficient. Therefore, to make the looking-up efficiency, the new Bloom filter is created by consolidating the Bloom filters of nodes based on the DHT search path. An efficient looking-up becomes possible as well as DHT because it consolidates the Bloom filter according to the structure of DHT.

In Chord, OR operation is applied to NBf of all nodes in the finger table within the range of each search path. Fig. 2 shows the range of each search path of Node 0, and Node 0 makes a new Bloom filter from NBf of these search paths. The Finger Bloom filter (henceforth FBf) obtained thus contains all key words of each search path. When FBf is made, each node only has to communicate with the node in finger table, and does the following processing. The finger[i] is an i-th node in finger table, and FBf[i] shows FBf of finger[i].

$$\mathrm{FBf}[i] = \mathrm{finger}[i].\mathrm{FBf}[0] + \mathrm{finger}[i].\mathrm{FBf}[1] + \ldots + \mathrm{finger}[i].\mathrm{FBf}[i-1]$$

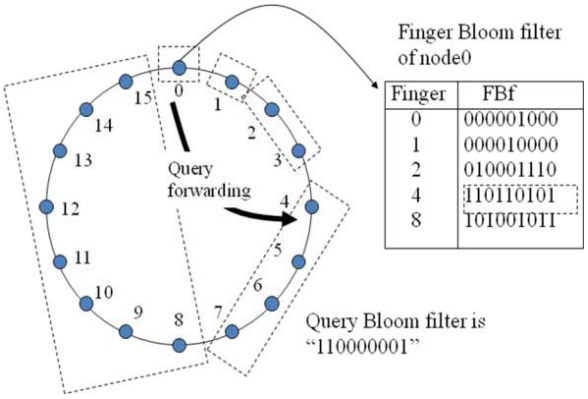

Figure 2. An example of the Bloom filters of Chord structure.

In the looking up data, the query Bloom filter is generated from the query key word, and routing does the query message according to this. The looking-up which uses the search path same as subjacent DHT becomes possible, because each node compares the query Bloom filter and the Bloom filter that each node consolidated according to the structure of DHT and it forwards the query only to the node which has the Bloom filter matching to the query Bloom filter. However, when the query message is transmitted, it is necessary to specify the range of the search so that the search should not overlap.

2.3. Attenuated Bloom filter

Attenuated Bloom filter [7] is the standard Bloom filters combined with *d* layers. A certain node stores information on the presence of the content in the surrounding nodes in the attenuated Bloom filter. In the *(i+1)*th layer of this filter, information on all nodes in *i* hops has been consolidated. In the first layer, information on the site itself is stored.

Table1. Example of Attenuated Bloom filter

Bit number	0	1	2	3	4	5	6	7
Local Info.	0	1	1	0	0	0	0	0
First Hop	0	0	1	0	1	1	1	0
Second Hop	0	1	0	1	1	1	0	1

This attenuated Bloom filter is regularly exchanged by the node which is mutually adjacent. When a certain node receives the attenuated Bloom filter from the node in the vicinity, OR operation is applied to the *i*-th layer of the received filter and *(i+1)*th layer of the node.

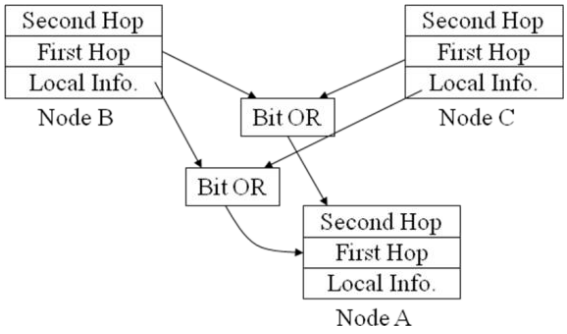

Figure 3. Exchange of the information of the attenuated Bloom filter.

Fig. 3 shows the situation. Therefore, information two hops ahead will be consolidated in the third layer and one hop ahead in the second layer. In this attenuated Bloom filter, the link to which the filter is received is managed. Therefore, whether the searched filter exists can be specified. And, the direction where the nearest filter exists can be specified. The query can be appropriately forwarded from the information.

Because structurizing the network is not required, the maintenance cost of this method is lower than that of the structurizing method. However, there is a problem that the upper bound how many times the query is forwarded is not decided.

3. Bloom filter based on the tree structure

3.1. B-tree structure

In this research, each node (actual node which manages key word information) manages the distributed B-tree. The node in B-tree is called a logical node because it differs from a physical node. ID of a physical node is stored in the leaf node in B-tree. Moreover, the branch information (array of node ID which shows the situation of the branch to the parent node and the child node etc.) and the version number of the information is managed by the internal node of the tree.

Information on this B-tree is managed with a physical node which participates in the network. Each physical node has partial information on B-tree. The consistency between all physical nodes is managed by notifying other nodes change information when the change is caused in B-tree. The physical node manages information on the nodes which belong to the path from the leaf (physical) node to the root node of the tree.

Information of the sibling nodes of the path from the leaf to root node are also managed by the physical node. Therefore, B-tree of the height of about $log_m N$ is composed when N is the number of physical nodes and making it to m-ary tree. Each physical node has information on the number of nodes proportional to $log_m N$.

The root node and the each internal node are assumed to manage the nodes by the ID number of the smallest ID in the internal node information connected below. The physical node with this smallest ID in the internal node information is called as a representative node of the internal node. The representative node has the responsibility of actually managing the internal node information. Therefore the representative node manages the information of the corresponding internal node.

Fig. 4 shows the example of the physical node managed by B-tree and the representative node in that.

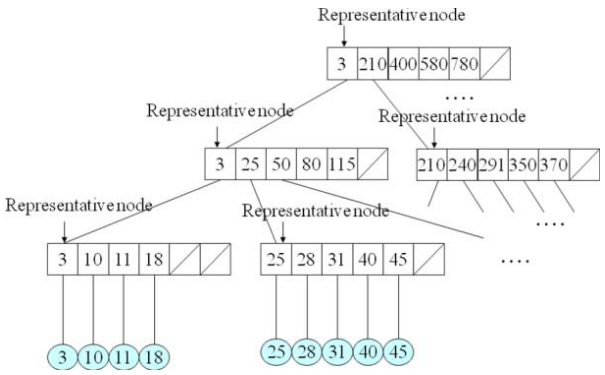

Figure 4. Node management in B-tree.

3.2. Management of the tree

3.2.1. Participation method. A physical node access any of the node which participates already when the node newly participates in B-tree and send the participation request including ID oneself. The accessed node forwards the request to the representative node of the root node. The representative node of the requested root node judges which internal node should be managed based on ID of the request, and forwards the request to the internal node of the subordinate position. In this repetition of the procedure, the participation request reaches the representative node of the lowest internal node which is the appropriate to manage the new node. The ID of the new node is just registered in the array of the internal node when there is room of the array.

The internal node does the division work of the node in B-tree when there is no room in the array. As for the division work, the division of the internal node at the level might cause the division of the higher level. At last, a new root node might be made after the division of the root node. The process can be judged only by information on the representative node of the internal node.

The notification of change information is executed by the representative node of the highest node where the change was caused. The notification is done by the multicast from a higher representative node to the representative node of the subordinate position. In the information update, it is meant the update has been generated concurrently by another process when the version number of the node has already been changed, and discontinues the change process. After the interruption, the participation processing is executed again.

3.2.2. Secession method. When a physical node secedes from B-tree, the secession request is sent to the representative node of the lowest internal node that the node belongs. The representative node just deletes the node ID from the array if the number of the element is not less than $m/2$ when the node ID is deleted from the array of the internal node.

When the number of elements becomes less than $m/2$, the array element is moved from the adjacent sibling node. The adjacent node is amalgamated with the internal node when becoming less than $m/2$ when the array element is moved from the adjacent node. This amalgamation operation goes up to the root node if necessary. According to circumstances, the operation deletes the root node. This process can be judged according to information on the representative node of the internal node.

The notification of change information is executed by the representative node of the highest node where the change was caused as well as the participation process.

3.3. Management of the Bloom filter

In each physical node, a consolidated Bloom filter for the index information is managed. Moreover, a higher internal node has a consolidated Bloom filter which consolidates the Bloom filter of the node of the subordinate position based on the structure of B-tree. In the root node, there is the entire Bloom filter which consolidates the Bloom filter of all nodes.

Information on the consolidated Bloom filter has been distributed to a physical node as well as

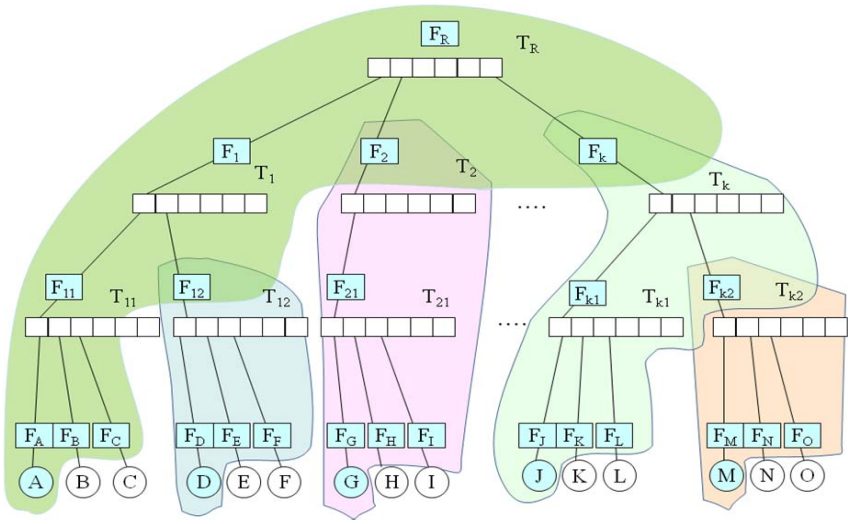

Figure 5. Region of representative node.

information on B-tree. In a word, a representative node has consolidated Bloom filter corresponding to the internal node and Bloom filters of the subordinate filters. Fig. 5 shows information that each representative node should have. TR shows information of the root node. The nodes from T1 to Tk show the first level nodes of the B-tree. The nodes from T11 to T12 show the second level nodes which are the subordinate of the T1. Moreover, FR is information of filter of the root node. The filters from F1 to Fk are information on the consolidated filters in the first level nodes. The filters from F11 to F12 are information on the consolidated filters in the second level nodes. The filters from FA to FC are information on the consolidated filters in the nodes from node A to C.

3.3.1. Management of Bloom filters of leaf node. It notifies the representative node of the highest internal node that the change of consolidated Bloom filter propagates when there is a change in consolidated Bloom filter of the leaf node. The change is notified to the representative node of relating sibling node and subordinate position node by the multicast.

3.3.2. Management of joining node. The representative node of the internal node which accommodates the new node computes the change of consolidated Bloom filter when there is new participation in B-tree. It notifies the representative node of the highest internal node that the change propagates when consolidated Bloom filter is changed, and it propagates to each node by the multicast.

3.3.3. Management of secession of the node. The representative node of the internal node which manages the secession node computes the change of consolidated Bloom filter when there is secession from the B-tree. It notifies the representative node of the highest internal node that the change propagates when consolidated Bloom filter is changed, and it propagates to each node by the multicast.

3.4. Method of looking up data

In the looking up data, first of all, the Bloom filter for the query is made from the key word. The query including the Bloom filter for the retrieval is sent to the node of B-tree somewhere. The query is transferred to the representative of the lowest internal node. The representative node which reaches the query compares the query and Bloom filter which the node maintains. If the matching between the query and the Bloom filter fails, the query is forwarded to a higher representative node. If the matching of the query and the Bloom filter succeeds, the query is forwarded to an appropriate subordinate representative node. If the node is the lowest representative, the query is forwarded to a leaf (physical) node.

Fig. 6 shows an example of forwarding the query. When filter 010101 for the query is sent to node A, it detects that the filter Fk and F12 match the query. The query will be forwarded to node D and node J which is the subordinate representative node of filter Fk and F12.

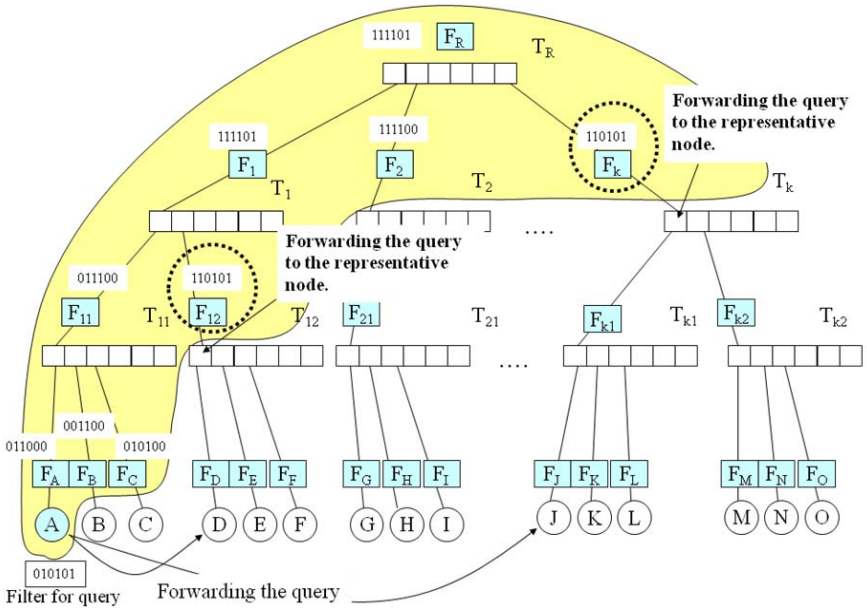

Figure 6. Forwarding the query of A to other representative nodes.

4. Similarity-based tree management

B-tree is generally composed by the value of the node such as key-value. Since B-tree is reconstructed when the node join/leave happens, the cost of reconstruction becomes large if the node join/leave is frequent. Therefore, we propose the tree management method based on the similarity of Bloom filter.

If the similar Bloom filters are grouped and managed as the child node of one internal node, the change of the bit pattern of the internal node is reduced. By the fact, the reconstruction cost is reduced.

4.1. Similarity of Bloom filter

Similarity of the Bloom filter is defined as the number of the bit '1' which the same position of the filter. Therefore, AND operation is applied to the Bloom filters and the number of bit '1' in the result is used for the similarity. If the similarities are the same, the number of bit '0' which is the same position is used. Therefore, the Bloom filters are revered and AND operation is applied. The number of bit '1' in the result shows the second similarity.

The Fig.7 shows an example of similarity of Bloom filter. In the example, a new Bloom filter is compared to the filter X, Y and Z. From the results of AND operation, the similarity of X and Y is 2 and the similarity of Z is 1. The Fig.8 shows an example of the similarity of the bit '0'. From the result of the AND operation, the filter Y is the best similarity.

4.2. Node join

If a new node joins to the tree, the Bloom filter of the new node is compared to the Bloom filters of the tree from root node. The node which has the most similar Bloom filter in the level is selected and it goes down to the next level repeatedly.

The process is repeated to the lowest internal node and the position which the new node joins is selected.

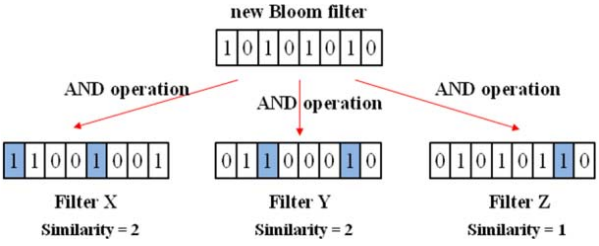

Figure 7. Similarity of Bloom filter.

4.3. Division of the internal node

The maximum number of child node which the internal node of m-ary tree can keep is *m*. Therefore the internal node must be divided if the number of child node exceeds the *m*. In a word, the child nodes of the internal node must be divided to the two groups.

In the grouping process, the similarity of the each child node and the new node is calculated. After that, *(m+1)/2-1* nodes which has the high similarity are selected. One group is composed of the new node and the selected nodes, and other group is composed of the rest.

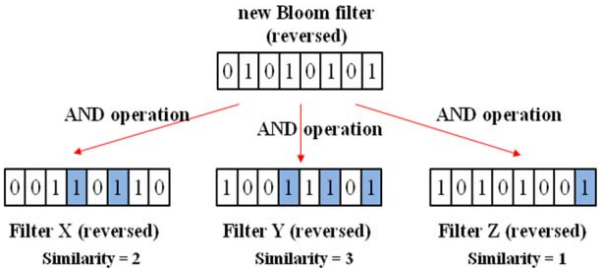

Figure 8. Similarity of reversed Bloom filter.

4.4. Node Leave

The m-ary B-tree must take a child node from the sibling node if the number of child node becomes less than *(m/2)*. In the process, the selection of the child node is base on the similarity of the child node and internal node. The most similar child node moves to the internal node.

5. Evaluation

The number of forwarding of the query message is evaluated. The query will be forwarded from the lowest representative node to root and from the matched node to leaf node. In a word, the forwarding time of the query is proportional to the depth of the tree, which is $O(\log_m N)$.

Moreover, the computational complexity of the work required to change, to add, and to delete the Bloom filter is evaluated. The message to notify the change of the Bloom filter is forwarded from the leaf node to root. Therefore, the complexity of the change of Bloom filter is proportional to the height of the tree, which is $O(\log_m N)$.

Average number of Bloom filters that each node must manage is O(1), because the total Bloom filters which must be managed by representative nodes are proportional to the number of nodes in the B-tree, which is $O(N)$. If it thinks about the average number of the Bloom filter managed by the representative node, it becomes $O(m)$ because the number of representative node is about N/m.

Because the number of forwarding time of the query in [8] is equal to the query forwarding time in Chord, it becomes times of $O(\log_2 N)$. Moreover, the change of the filter is necessary the notification of times of $O(\log_2 N)$ when it notifies the node to which the node with the change relates. Since propagations to the all node can be generated in necessary, the maximum communication cost is proportional to N. Average number of Bloom filters that each node must manage is $O(\log_2 N)$, because finger Bloom filter for node which does not exist is condensable. On the other hand, maximum size of the finger Bloom filter is M, which is the size of the Chord ring.

Table 2 is the summary of the evaluation by comparison with the Chord based method and B-tree based method.

Table 2. Evaluation by comparison.

	Chord based method[10]	B-tree based method(proposed)
Number of times for query forwarding.	$O(\log_2 N)$	$O(\log_m N)$
Cost to change the Bloom filter.	$O(\log_2 N)$	$O(\log_m N)$
Number of filters that each node manages.	$O(\log_2 N)$	O(1) O(m) (*)

(*) in case of the only for representative node.

The average forwarding times of the query is evaluated by the simulation. It is compared to the Chord algorithm. The number of sub-tree of the internal node in B-tree is 5 and 10. Number of nodes is arranged as 1000, 2000, 5000, 10000 and 20000. The average forwarding times of the query is calculated by the 1000 times of experiment. Simulation result is shown as Fig. 9.

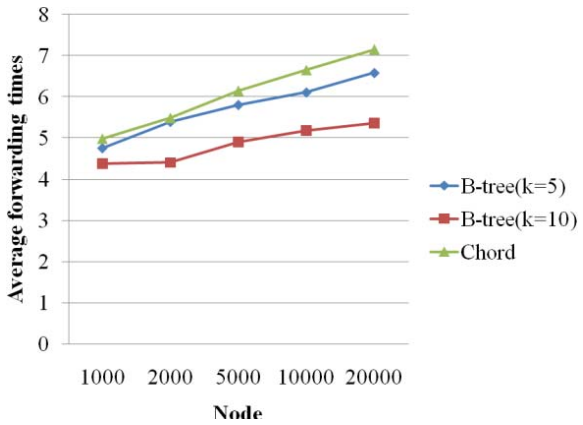

Figure 9. Average forwarding times of the query.

The average forwarding times of the query is reduced about 25% than Chord in the best case. It shows that the performance of the proposal method improves the lookup efficiency.

To evaluate the effectiveness of the tree management method based on the similarity, the number of reconstruction is simulated. The number of nodes is arranged as 10000, 20000, 50000 and 100000. The 2000 join/leave are repeated and the number of reconstruction of the tree is evaluated. The proposed method is compared to the node ID-based tree management method. The result is shown in the Fig.10. From the result, the proposed method reduces 9 to 16% of the reconstruction cost.

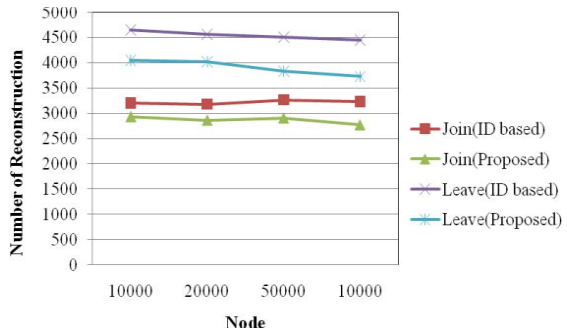

Figure 10. Number of Reconstruction.

6. Conclusions

In this research, it proposed the management method of the Bloom filter with B-tree structure for information retrieval. There is an advantage that the upper bound of the number of forwarding the query is fixed though it takes the cost to organize sites to a logical B-tree composition in this method.

Since a B-tree is used to manage Bloom filters, the height of the tree changes dynamically in proportion to the number of nodes in our proposal. Because the total number of Bloom filters of the internal node is proportional to the number of node in B-tree. Therefore, the number of Bloom filters can be less than that of the Bloom filter with fixed ring structure. This contributes to the speed improvement of the query processing of each node.

We also proposed the similarity-based tree management method. The method was evaluated by the number of reconstruction. The results showed that the proposed method reduce the reconstruction cost. It is effective to the network which the node join/leave is frequent.

In the proposed method, the representative node is used to propagate of maintenance information on the tree and to propagate the query information. The future work is to evaluate how much the concentration of the load on this representative node influences the performance. Furthermore, the method that such a representative node is not fixed will be investigated.

References

[1] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In Proceedings of the ACM SIGCOMM 2001 Technical Conference, San Diego, CA, USA, August 2001.

[2] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pages 329-350, November 2001.

[3] Petar Maymounkov and David Mazieres. Kademlia: A Peer-to-peer Information Systems Based on the XOR Metric . In Proceedings of the IPTPS 2002, Boston, March 2002.

[4] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In Proceedings of the ACM SIGCOMM 2001, San Diego, CA, USA, August 2001.

[5] Andrei Broder, Michael Mitzenmacher, Network Applications of Bloom Filters: A Survey, Internet Mathematics, 2002, pp.636-646

[6] S. Czerwinski, B. Y. Zhao, T. Hodes, A. D. Joseph, and R. Katz: An Architecture for a Secure Service Discovery Service, In Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99), pp. 24—35. New York: ACM Press, 1999.

[7] S. C. Rhea and J. Kubiatowicz.: Probabilistic Location and Routing, In Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Volume 3, pp. 1248—1257. Los Alamitos, CA: IEEE Computer Society, 2002.

[8] J. Kubiatowicz, D. Bindel, P. Eaton, Y. Chen, D. Geels, R. Gummadi, S. Rhea, W. Weimer, C. Wells, H. Weatherspoon, and B. Zhao.: OceanStore: An Architecture for Global-Scale Persistent Storage., ACM SIGPLAN Notices 35:11 (2000), 190—201.

[9] B. Bloom. "Space/Time Tradeoffs in Hash Coding with Allowable Errors."Communications of the ACM 13:7 (1970), 422-426.

[10] Kazuho Sato, Noriko Matsumoto and Norihiko Yoshida, Multi-Keyword Search for DHT P2P Networks, IPSJ/IEICE Forum on Information Technology (FIT)2006, (2006).