PAPER

Load Balancing Scheme on the Basis of Huffman Coding for P2P Information Retrieval

Hisashi KURASAWA[†a)], **Atsuhiro TAKASU**[††b)], ***and*** **Jun ADACHI**[††c)], ***Members***

SUMMARY Although a distributed index on a distributed hash table (DHT) enables efficient document query processing in Peer-to-Peer information retrieval (P2P IR), the index costs a lot to construct and it tends to be an unfair management because of the unbalanced term frequency distribution. We devised a new distributed index, named Huffman-DHT, for P2P IR. The new index uses an algorithm similar to Huffman coding with a modification to the DHT structure based on the term distribution. In a Huffman-DHT, a frequent term is assigned to a short ID and allocated a large space in the node ID space in DHT. Throuth ID management, the Huffman-DHT balances the index registration accesses among peers and reduces load concentrations. Huffman-DHT is the first approach to adapt concepts of coding theory and term frequency distribution to load balancing. We evaluated this approach in experiments using a document collection and assessed its load balancing capabilities in P2P IR. The experimental results indicated that it is most effective when the P2P system consists of about 30,000 nodes and contains many documents. Moreover, we proved that we can construct a Huffman-DHT easily by estimating the probability distribution of the term occurrence from a small number of sample documents.

key words: *peer-to-peer, information retrieval, load balancing, Huffman-coding*

1. Introduction

Peer-to-Peer (P2P) systems have low management cost, high scalability, and good dependability. Furthermore, they are suitable for developing and maintaining a system on a large amount of distributed computation resources that are dynamically reconfigured. However, information retrieval (IR) and information management in P2P systems are difficult because documents are distributed on a P2P network. Although early studies developed efficient query routing schemes, such as the depth-first search [1] and distributed hash table (DHT) [2]–[4], they are not suitable when there is a small amount of network traffic or a need for a precise search. The search indices of the early schemes mainly manage data locations, and don't provide term information, which is important for judging the relevance of documents. Recent studies on query routing schemes have tried to adopt the existing IR techniques for centralized systems to P2P systems.

Modern IR methods use the occurrence of term information in document collection, such as term frequency (TF) that denotes the number of term occurrences in a document and document frequency (DF) that denotes the number of documents including the term, to measure the relevance between queries and documents [5]. The calculation of term information is not an easy task in P2P systems. Thus, early P2P IR systems did not fully use term information. As a result, they gathered documents containing less relevant documents, and this caused heavy network traffic loads.

Recently, several distributed indexing methods for P2P IR systems have been proposed [6]–[9]. The problem with distributed indices is that they cost a lot to construct and have an unfair index management cost. The information about each term is usually maintained by a specific node. When processing a query or inserting a new document into a P2P IR system, we need to access the nodes that keep the term information in the queries or documents. A distributed hash table (DHT) is often used for this purpose. It enables us to access the target node in $O(\log n)$ hops for the number n of nodes in the P2P network. To register a document to the index, we need to access all the nodes that keep the term information included in the document. As a result, the construction of the index increases the traffic. Moreover, it is empirically known that the term occurrence probability obeys an exponential distribution, that is, a small number of terms appear frequently whereas many other terms appear with lower probability in documents. The nodes that manage indices of frequent terms receive more registration and reference accesses than nodes with less frequent terms; hence, the index management cost is unfair among nodes.

To overcome these problems, we proposed a new distributed index [10]. Today's distributed indexing methods assign terms to nodes and access them in $O(\log n)$ hops, regardless of the term probability. Our method allocates term indices to nodes on the basis of the term distribution, and also finds a node with less hops for more frequently appearing terms. To achieve this, we modify the node access scheme of the DHT by using a technique from coding theory. The proposed method uses an algorithm similar to Huffman coding, so we call the resultant scheme *Huffman-DHT*. We show through simulations that the proposed method reduces the average number of hops for index accesses in the construction process, and consequently, balances the traffic load for index construction and management.

Manuscript received November 10, 2008.
Manuscript revised June 8, 2009.

[†]The author is with the Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, 101–8430 Japan.
[††]The authors are with National Institute of Informatics, Tokyo, 101–8430 Japan.
a) E-mail: kurasawa@nii.ac.jp
b) E-mail: takasu@nii.ac.jp
c) E-mail: adachi@nii.ac.jp

DOI: 10.1587/transinf.E92.D.2064

The rest of this paper is organized as follows. We briefly survey P2P IR systems in Sect. 2, and overview the indexing schemes for P2P IR systems in Sect. 3. In Sect. 4, we describe the new peer searching method that balances the load of peers. Section 5 presents our experimental results. Section 6 concludes this paper and outlines future research directions.

2. Related Work

We explain the related workon searching and load balancing in P2P IR. For realizing a precise search, the P2P IR studies construct a global index and manage information about the occurrence of terms in shared documents. In P2P IR systems using unstructured P2P networks, a node floods a global index with a gossip algorithm and shares information concerning the peers' document collections [6]. Since the index is shared by all the nodes, the index cannot be updated frequently. On the other hand, in P2P IR systems using structured P2P networks, a node makes a global index using a DHT [7]–[9]. In structured P2P IR systems, a term is assigned to the appropriate node, whose hash value is near the hash value of the term. We call the node *an index node* for the term. Every node maintains the part of the global index related to the assigned terms on the DHT. Two types of global indices have already been proposed. One is a term-to-peer index where the terms are indexed by a peer's collection [7]. As each node maintains a peer list for the assigned terms, the term-to-peer index cannot distinguish between the documents in the same peer's collection and has worse search performance than a centralized index [11]. Many peer selection strategies have been proposed for more efficient searching, such as CORI [12], the overlap awareness strategy [13], global document frequency estimation [14], and term co-occurrences [11], [15]. The other type of index is a term-to-document index where the terms are indexed by document [8], [9]. We show how to create the term-to-document index in Sect. 3. The term-to-document index can directly refer to the term frequency of a specific document without accessing the document, and its search performance is as good as that of a centralized index. However, the indexing cost is heavy. A query-driven index can reduce the cost by indexing only those terms that have occurred in queries. In the index, the terms that are not indexed must be searched via a broadcast, and because of this, the search efficiency of the query-driven index is inferior to that of systems using a term-to-document index that has the information of all the terms in the documents.

In structured P2P IR systems using ordinary DHT, terms are assigned to nodes on the basis of the hash function. Owing to this node ID assignment, the nodes responsible for a frequent term get heavier loads because they are frequently accessed when documents get registered in the index. Thus, structured P2P IR systems needs load balancing techniques. Those load balancing techniques include the virtual server [2], the local and random probes algorithms [16], pair-wise interactions of peers [4], and proximity-aware balancing [17]. These techniques can improve the node ID assignment in a DHT and can be used in any P2P applications. However, they impose an extra traffic load to modify the IDs.

Search efficiency depends on the quality of the global index. Although registering a document in the index costs more in structured P2P IR systems than in unstructured systems, structured systems can retrieve relevant documents with higher precision. In particular, the systems using a term-to-document index have a better search performance. An ideal P2P IR system can make a precise search for documents, and constructs an index at a lower cost. To achieve this goal, we used the term distribution and refined the DHT structure to overcome the inefficient index construction drawback.

3. Background

3.1 Indexing and Data Allocation Scheme in P2P IR Systems

In this section we explain the term-to-document indexing scheme in P2P IR systems using structured P2P networks.

The global index consists of inverted indices for each term. An inverted index for a term is held by the appropriate node, whose hash value is similar to the hash value of the term. The inverted index maintains three values for a document, the file name, the term weight in the document, and a pointer to the data of the document. A node registers document information to the indices of each term in the document.

The term weight in the indices is usually measured using the term and document frequencies. For example, Concordia [9] uses a probabilistic model and the term weighting formula proposed by [18], which is a variation of BM25 [19].

When given a query consisting of a set of terms, the system connects directly to the index nodes for each term in the query and refers to each index in the nodes. The system calculates the relevance score of each document to the query, and gathers relevant documents with high scores from the network. Figure 1 shows the indexing and retrieving scheme in P2P IR systems.

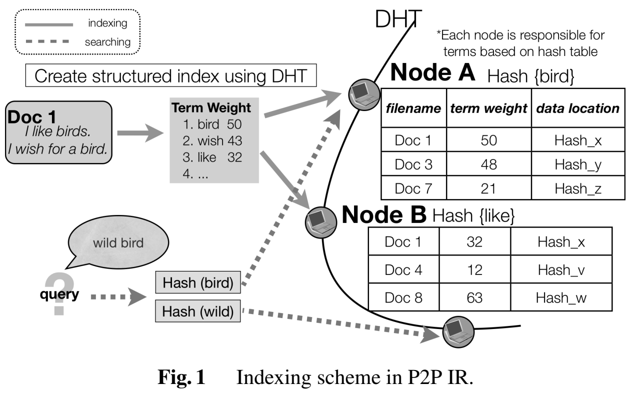

Fig. 1 Indexing scheme in P2P IR.

3.2 Indexing Cost

We compare the term-to-peer index [7] and term-to-document index [8], [9] from the standpoint of indexing cost.

Let us first consider the cost T_1 of a node in P2P IR methods using a term-to-peer index registering a document collection summary for every term to the index nodes on the DHT. Let $f(t)$ be the cost required for calculating the weight of t, and W be the set of terms in the collection. Then, T_1 can be written as:

$$T_1 = \sum_{t \in W} f(t).$$

The number of registrations to the indices in [7] is $|W|$.

On the other hand, the cost T_2 that a node in P2P IR methods using a term-to-document index needs to register each document for every term in its collection to the index nodes on DHT is

$$T_2 = \sum_{d \in C} \left(\sum_{t \in \boldsymbol{d}} f(t) \right),$$

where $\boldsymbol{d}$ denotes a document and C denotes a collection. The number of registrations to the indices in the P2P IR methods is represented as:

$$\sum_{\boldsymbol{d} \in C} |\boldsymbol{d}|,$$

where $|\boldsymbol{d}|$ is the length of a document $\boldsymbol{d}$.

It is clear that the indexing cost in P2P IR methods using a term-to-document index will be lower if a node registers a set of documents including the same term. However, the number of index records is still larger than that of methods using a term-to-peer index. Although a term-to-document index costs a lot, it is useful for IR because it has enough term information to adapt IR algorithms. Therefore, we would like to refine the term-to-document indexing method.

4. Huffman-DHT

4.1 System Overview

The proposed P2P system uses the term-to-document index on top of a structured P2P system to provide the document management and retrieval functions. We can use any structured P2P system that uses a 1-dimension key space such as Kademlia [3] and Chord [2]. We are currently using Kademlia for our implementation.

In the initializing phase, a *trigger node* takes an initial set of document collection and estimates the probability distribution of the word frequencies and constructs the coding tree described in Sect. 4.2. The initially constructed coding tree is copied to all the nodes attached to the P2P network. Each node constructs the index of the corresponding terms described in Fig. 1 in Sect. 3.1. Each node also allocates the documents and their replicas to the appropriate nodes by using the data allocation scheme proposed in [7] and [9]. Then, the P2P system initiates the document management and retrieval services.

In the running phase, a query can be issued from any node in the P2P network by using the procedure described in Sect. 4.4. A new document is also registered by any of the nodes by using the procedure described in Sect. 4.5.

When joining a P2P system, a new node first contacts a node and participates in the network according to the joining procedure of the underlying structured P2P system. Then, the node that is joining the system copies the coding tree from the contacted node, whereas it copies the index for its corresponding term from a neighbor node in the underlying structured P2P system. The corresponding term is determined by using the procedure described in Sect. 4.3. When a node leaves the system, it searches for the nodes that should manage its index, and transfers the index to them.

4.2 Huffman Coding

A Huffman-DHT assigns an ID to each node. Let us denote the ID by a bit vector $b_1 b_2 \cdots b_l$, where b_i $(1 \leq i \leq l)$ is a 0 or 1. For nodes p and q, the distance between p and q is defined as $|id(p) - id(q)|$ where $id(\cdot)$ is the ID of a node. Each node has a list of addresses of the other nodes in the same way as in an ordinary DHT.

We need to construct a coding tree for frequent terms, and we need to know the probability distribution of terms when coding. We can use the document frequency distribution for coding, because the document frequency of a term denotes the number of accesses to index nodes when documents are registered. We estimate the distribution from a sample document set S. Let $df(t, S)$ denote the document frequency of a term t in the sample document set S. We select the top k terms in descending order of $df(t, S)$ as frequent terms, where k is a parameter of the Huffman-DHT. The remaining terms are assigned to the ID calculated by a hash function in the same way as in the distributed index [7], [9]. We estimate the probability of a term t in the frequent terms T by

$$p(t) \equiv \frac{df(t, S)}{\sum_{t \in T} df(t, S)} \ .$$

Then, we construct the coding tree using the Huffman coding algorithm [20]. Figure 2 shows an example of a coding tree. As shown in the figure, we use the binary alphabet for coding.

The reasons we encode only the frequent terms are:

- we can reduce the size of the coding tree that must be shared by all nodes,
- the coding is effective for frequent terms, and
- we can estimate the distribution from a small document sample.

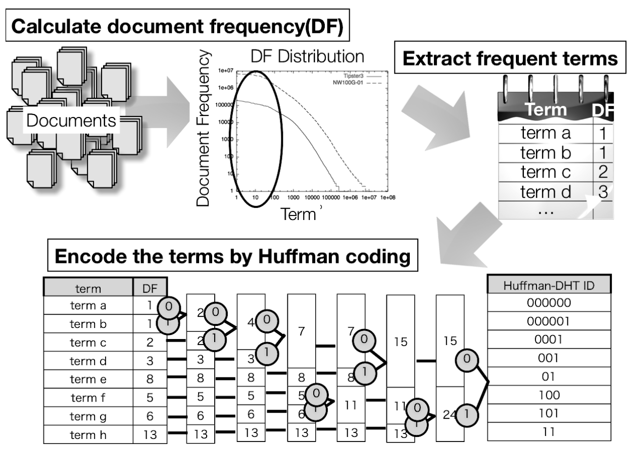

Fig. 2 ID space construction in Huffman-DHT.

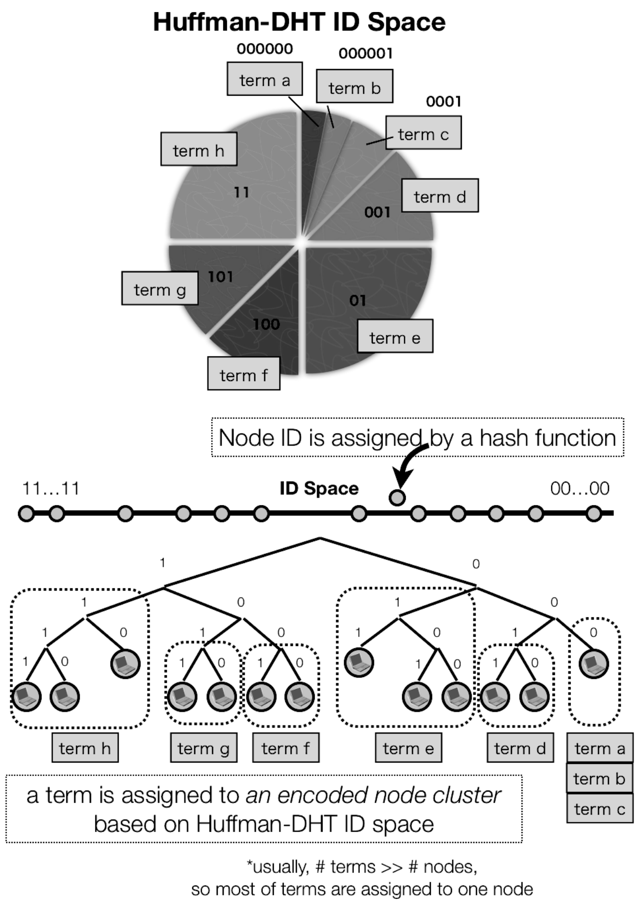
Huffman-DHT ID Space
000000
000001
0001
term a
term b
term c
term d
term e
term f
term g
term h
11
001
01
100
101
Node ID is assigned by a hash function
11…11
ID Space
00…00
a term is assigned to an encoded node cluster based on Huffman-DHT ID space
*usually, # terms >> # nodes, so most of terms are assigned to one node

Fig. 3 Node assignment process.

4.3 Node Assignment

For a frequent term t, let $c_1 \cdots c_m$ be its Huffman code. If the code length m is longer than the bit vector length l of the node IDs, the code is truncated to l bits, and the term is assigned to a node whose ID is closest to the truncated code $c_1 \cdots c_l$. Otherwise, a term is assigned to a set of nodes whose IDs are between $\underbrace{c_1 \cdots c_m 0 \cdots 0}_{l}$ and $\underbrace{c_1 \cdots c_m 1 \cdots 1}_{l}$. We refer to this node set as an *encoded node cluster* for t. If t is not in the frequent terms, it is assigned to the encoded node cluster for the frequent term whose ID is the closest to its ID. Note that a more frequent term is encoded with a shorter code by Huffman coding, and therefore, it is assigned to a larger encoded node cluster. Figure 3 shows the assignment of frequent terms and the node assignment in Fig. 2.

4.4 Node Search

Suppose a node, referred to as a *query node*, searches for an index node of a term t.

If t is a frequent term, the query node encodes it by using the coding tree. Otherwise, the node finds a term t' in a list of the frequent terms whose hash number is nearest that of t, and assigns t as the code of t'. The query node recursively performs the following steps:

1. If the query node knows the address of a node in the encoded node cluster for t, it returns the address.
2. Otherwise, send the query to the node in the address list whose ID is the closest to one in the encoded node cluster.

The node search method differs from that in an ordinary DHT where Huffman-DHT finishes searching when it reaches any one node in the encoded node cluster. We can find an index node in fewer hops than it would take with an ordinary DHT. Moreover, we can find index nodes of a more frequently appearing term with fewer hops, because the encoded node cluster is larger for more frequent terms.

In P2P IR systems using an ordinary DHT, the maximum number of hops required for searching the node which manages an objective term t_k is:

$$h_{baseline}(t_k) = O(\log n), \tag{1}$$

where n is the number of nodes in the P2P network. This number is the same as the number of hops required for accessing the objective node using the ordinary DHT.

On the other hand, the maximum number of hops required for retrieving a term in P2P IR systems using a Huffman-DHT is based on its document frequency. According to Zipf's Law, the term occurrence probability of term t_k is defined as:

$$P_{t_k} = \frac{\frac{1}{k^s}}{\sum_{x=1}^{|W|} \frac{1}{x^s}}, \tag{2}$$

where s is the value of the exponent characterizing the distribution. In the Tipster 3 collection, s is about 1. According to the Huffman coding theory, the maximum code length of t_k in a Huffman-DHT created from the top M frequent terms is:

$$max(l_{t_k}) = \begin{cases} -\log P_{t_k} + 1 & (k \le M) \\ -\log P_{t_M} + 1 & (\text{otherwise}). \end{cases} \tag{3}$$

As we described in Sect. 4.3, a term is assigned one of the codes of the top M frequent terms. Therefore, the code length does not exceed that of t_M. The maximum number of hops in P2P IR systems using Huffman-DHT depends on the code length of the object and the number of nodes in the network. When the network is small and the number of nodes in the encoded node cluster of a term is one, the maximum number of hops for retrieving the term is the same as the number of hops required for accessing the objective node using an ordinary DHT. On the other hand, when the network is large and the number of nodes in the encoded node cluster of a term is large, the maximum number of hops for retrieving the term depends on the document frequency of the term. The maximum number of hops for retrieving the nodes which manages t_k is:

$$h_{HD}(t_k) = \begin{cases} min\{-\log P_{t_k} + 1,\ O(\log n)\}, \\ \quad (k \leq M) \\ min\{-\log P_{t_M} + 1,\ O(\log n)\}, \\ \quad (\text{otherwise}) \end{cases} \tag{4}$$

From Eqs. (1) and (4) it is clear that the proposed Huffman-DHT is superior to ordinary DHT in searching a node for an objective term.

4.5 Document Registration

In the proposed Huffman-DHT, the index for each term is copied to multiple nodes. In the current implementation, each index per document represented by 512 bits consisting of 320 bits for document file name, 32 bits for term weight, and 160 bits for document location. Although the index size is small, since the number of nodes can be large for frequent terms, document registration may cause burst traffic if we access all the nodes in an encoded node cluster for a frequent term at one time. Therefore, we adopted a two-phase registration procedure; first the document information is registered into the index of one node in the corresponding encoded node cluster, and then propagates the modification later.

Suppose a node referred to as a *register node* adds document information to the index. In the Huffman-DHT approach, a document is registered into the index using the following steps:

1. For each term t in the document:
   a. The register node searches for an index node of t by using the procedure described in Sect. 4.4,
   b. The register node makes the index node modify the entry of t so that it contains the document information.

For the propagation, we introduce the parameter *QSync*, which controls the propagation delay. When a node receives a registration *QSync* times, it propagates the modifications of the index, i.e., the address list of the registered documents, to all the nodes belonging to the same node cluster. When $QSync = 1$, the registered document information is immediately propagated, i.e., the index is fully synchronized, but this may cause burst traffic. For a large *QSync*, the propagation is delayed, but we can reduce the total traffic for the propagation by merging the *QSync* modifications into one propagation.

The propagation cost increases as the number of nodes in an encoded node cluster increases. Suppose the minimum and the maximum depth of the Huffman-DHT are d_{min} and d_{max}, respectively. When the number of nodes in the P2P system exceeds $2^{d_{min}}$, the Huffman-DHT begins to assign multiple nodes to frequent terms. When the number reaches $2^{d_{max}}$, the largest encoded node cluster size then becomes $2^{d_{max}-d_{min}}$. For a cluster consisting of m nodes, the propagation requires $m-1$ messages [21]. Therefore, the Huffman-DHT requires a large amount of traffic for the propagation of a large cluster. This is the main drawback of the proposed method.

4.6 Load Balancing in Document Registration

In P2P IR systems, a node has a set of inverted indices of assigned terms. The number of registration accesses for a term t_k is in proportion to its document frequency $df(t_k, C)$. In an ordinary DHT, the term ID of t_k is determined by its hash value. Therefore, the term IDs distribute over the hash space uniformly if the number of terms is large enough. The size of the space allocated to t_k is given by:

$$Space_{DHT}(t_k) = \frac{1}{|W|}. \tag{5}$$

The density of the registration access cost is:

$$\begin{aligned} d_{DHT}(ID_{t_k}) &= \frac{df(t_k, C)}{Space_{DHT}(t_k)} \\ &= |W| \cdot P_{t_k} \cdot \sum_{d \in C} |\boldsymbol{d}|, \end{aligned} \tag{6}$$

where P_{t_k} is the appearance probability of term t_k. From Eq. (6), it is clear that the index registration access cost is influenced by not only the node ID assignment but also the term ID assignment.

On the other hand, in the Huffman-DHT, a frequent term is assigned to a short ID, and allocated a large space in the Huffman-DHT ID space. The size of the space allocated to t_k and the density of the index registration access cost in the Huffman-DHT approach is:

$$Space_{HD}(t_k) = \frac{1}{2^{l_{t_k}}} \simeq P_{t_k}. \tag{7}$$

$$d_{HD}(ID_{t_k}) = \frac{df(t_k, D)}{Space_{HD}(t_k)} = \sum_{d \in C} N_{t \in |\boldsymbol{d}|}. \tag{8}$$

From Eq. (8), $d_{HD}(ID_{t_k})$ is a constant related to the document collection. The Huffman-DHT is free from the influence of the term ID assignment. Therefore, the index registration cost in Huffman-DHT is influenced by the balance of the coding tree of the Huffman-DHT and the node ID assignment.

5. Experimental Results

The Huffman-DHT aims at assigning statistically balanced ID groups to terms that have a specific probability distribution. We conducted four simulation experiments to evaluate the Huffman-DHT. The evaluations were performed on the Tipster 3 collection that was used at TREC [22]. The collection consists of 336,310 articles from the San Jose Mercury News (1991), the Associated Press (1990), U.S. Patents (1983-1991), and Information from the Computer Select disks (1991, 1992) copyrighted by Ziff-Davis. We removed the stopwords and converted the remaining words to stems using the Porter stemmer [23].

5.1 Estimation of DF Distribution

First, we examined the document frequency distribution. Figure 4 shows the distribution of the Tipster 3 collection. The collection contains 817,897 terms in the total 53,445,728 words. Figure 5 shows the cumulative distribution of term occurrences. The terms are sorted by document frequency and assigned IDs by the hash function. The sum of the document frequencies of the top 4,580 frequent terms (0.56% terms in the collection) makes up about 80% of the total number of document frequencies. Therefore, by balancing the index construction cost for this small amount of frequent terms, we can distribute the total index construction cost among the nodes equally.

Second, we conducted an experiment on estimating the document frequency distribution over the terms. For the Huffman-DHT, we need to estimate the document frequency distribution and extract frequent terms from a small number of samples of the document collection. We compared the document frequencies distribution estimated from randomly selected sample documents with the distribution estimated from the whole Tipster 3 collection. Figure 6 plots the Kullback-Leibler divergence between the two distributions with respect to the number of sample documents. To handle terms whose document frequency was zero in the sampled documents, we smoothed over the distributions estimated from the sample documents by using the Laplace smoothing technique. As shown in the graph, the divergence converges at a small number of sample documents. Therefore, the DF distribution of frequent terms can be estimated from a small document collection.

5.2 Hops Required for Node Search

We conducted an experiment to find the number of required hops to search, a node assigned to an objective term using the Huffman-DHT. In this experiment we used the top 4,580 most frequent terms in the Tipster 3 collection to create a Huffman-DHT. These terms occupied 80% of the total document frequencies of all the terms. The mininum and the maximum depth of the resultant Huffman-DHT tree are 8 and 15, respectively.

Figure 7 shows the average number of hops needed to search a node with respect to the number of nodes in P2P IR systems. In the graph, the number of nodes is plotted in log scale. The lines represent a P2P IR system using an ordinary DHT and a system using the Huffman-DHT.

As shown in the graph, the Huffman-DHT reduced the

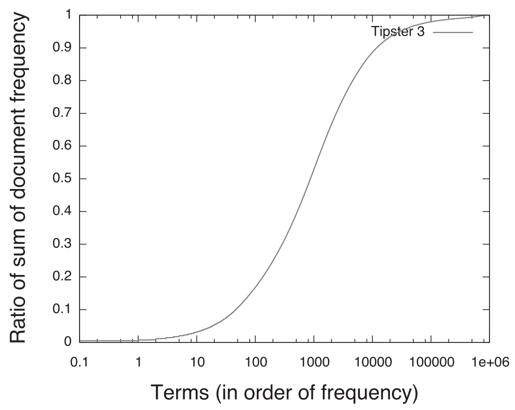

Fig. 5 Cumulative distribution of term occurrences.

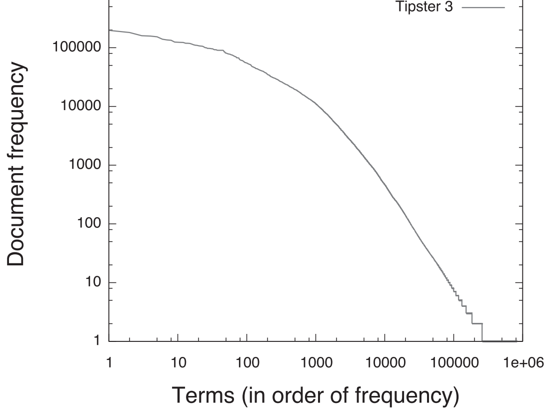

Fig. 4 Document frequency distribution.

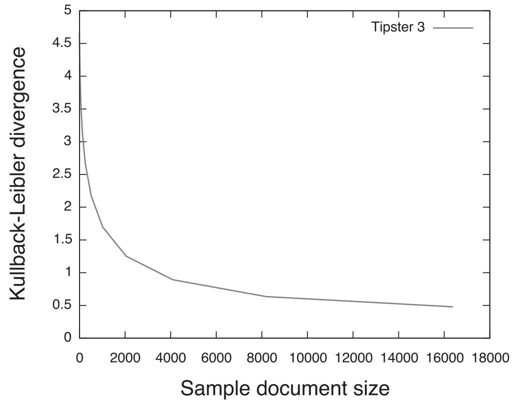

Fig. 6 Similarity between estimation and actual distribution.

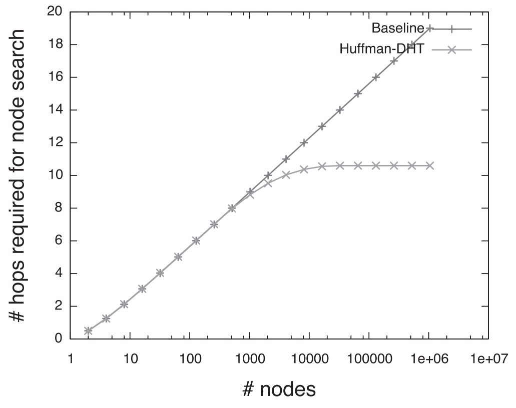

Fig. 7 Average no. of hops for node search.

required number of hops for searching a node as we discussed in Sect. 4.4. The graph also shows that Huffman-DHT is more effective for large P2P systems consist of more than 1,000 nodes. This means the Huffman-DHT is more scalable than an ordinary DHT. In this experiment, the minimum depth d_{min} of the coded tree is 8. Therefore, the effect of the Huffman-DHT begins to appear when the number of nodes exceeds $2^{d_{min}}$ because the path to frequent terms in the tree becomes shorter than that of less frequent terms, and so do the required hops. When the number of nodes exceeds the maximum depth $2^{d_{max}}$ of the tree where $d_{max} = 15$ in this case, the required number of hops for a term becomes exactly the same as the depth of the term in the encoded tree. Therefore, the required number of hops remains the same when there are more than $2^{d_{max}}$ nodes.

5.3 Registration Access Balancing

In an ordinary DHT, the index nodes of frequent terms are accessed frequently for document registration. Hence, these nodes become "hot spots". On the other hand, in the Huffman-DHT, the number of index nodes of a term depends on the term's document frequency. Therefore, the document registration accesses are shared among nodes.

We conducted an experiment on registration access balancing in a Huffman-DHT. The experiment measured the accesses to each node to register a document collection in an ordinary DHT and a Huffman-DHT. Let $a(p)$ denote the number of accesses at node p. The following ratio of the maximum to minimum numbers of accesses is the metric of the load concentration:

$$\frac{\max_{p \in N} a(p)}{\min_{p \in N} a(p)},$$

where N is a set of nodes in the network. Figure 8 plots the load concentration with respect to the number of nodes, where one line represents the load concentration for an ordinary DHT and the other lines represent the load concentrations for the Huffman-DHT for the delay parameter *QSyncs* of 1, 5, 10, and 20. The graph shows that the Huffman-DHT is more effective when a P2P system consists of more than

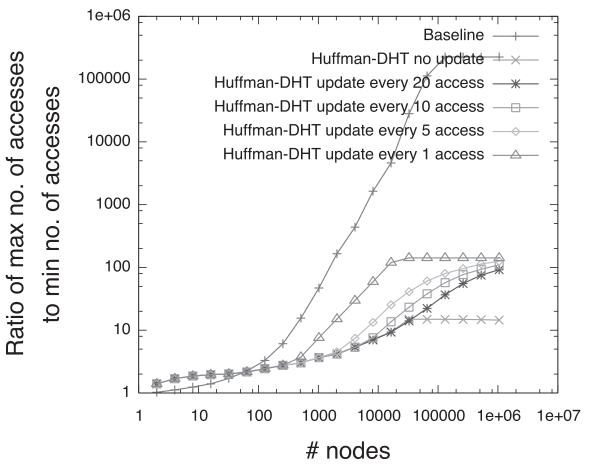

Fig. 8 Registration access balancing.

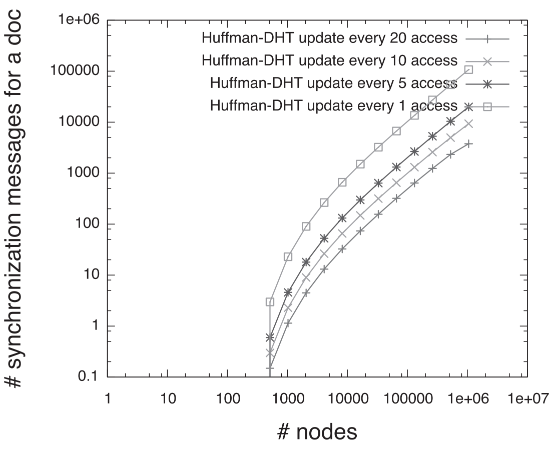

Fig. 9 No. of messages required for synchronization.

100 nodes, even if synchronization messages are needed. In an ordinary DHT, as the number of nodes increases, each node maintains the indices of reduced number of terms. Then, a node including indices for frequent terms are accessed more frequently than a node for less frequent terms. Therefore the load concentration increases as the number of nodes increases. On the other hand, in the Huffman-DHT, the term is assigned to a node depending on its frequency. This mechanism balances the accesses to the nodes in the P2P system.

For the minimum depth d_{min} of the coding tree, when the number of nodes exceeds $2^{d_{min}}$, the Huffman-DHT begins to assign frequent terms to multiple nodes and propagates the updated information of an index. Therefore, the concentration ratios of Huffman-DHT differ from one other depending on the parameter *QSync*.

5.4 Overhead Caused by Synchronization

As we described in Sect. 4.5, Huffman-DHT needs to propagate the updated index when the number of nodes exceeds $2^{d_{min}}$ where d_{min} denotes the minimum depth of the encoded tree. We registered documents in the corpus and measured the number of updated indices for synchronization for various node sizes and the parameter *QSync*. Let us call an updated index sent for synchronization *a message*. Figure 9 shows the average messages sent to register one document with respect to the number of nodes in the system. As shown in the figure, the system begins to send messages when the number of nodes is $2^{d_{min}}$ and number of messages increases as the number of nodes increases.

6. Conclusion

This paper proposes the Huffman-DHT as a means of balancing the index registration accesses among peers and reducing the load concentration for P2P IR systems. The Huffman-DHT assigns the index of a more frequently appearing term to a larger set of nodes by using the Huffman coding algorithm. We showed through simulations that the Huffman-DHT

- reduces the average number of hops to access a node with the index of the objective term, and
- equally distributes the index management costs among nodes

compared to an ordinary DHT in P2P IR.

A Huffman-DHT requires the probability distribution of the term occurrence and all the nodes keep a copy of the distribution. We proved that we can estimate the probability distribution from a small number of sample documents and this is sufficient for nodes to share the probabilities for a small set of frequently appearing terms.

The drawback of Huffman-DHT is its overhead in synchronizing the index update among replicas. We found from the experimental results that Huffman-DHT is effective for mid-size P2P system.

In order for Huffman-DHT to be effective for larger P2Ps, we need to improve the index synchronization methods. We plan to improve the method in two directions. One direction is to avoid the burst traffic caused by update propagation. The current system introduces a delay parameter to reduce the concentration of update traffic all at one time. We want to develop a mechanism to send an updated index when the network is less crowded. Another direction is to reduce the update messages.

We need to construct an encoded tree in the Huffman-DHT first. If the term frequency distribution is not stable enough for document registration, we need to modify the encoded tree. We showed that the distribution can be estimated from a small number of documents, but we need a more efficient encoded tree modification method. This is another future work of ours.

The other problem is the query processing cost. The Huffman-DHT was constructed according to the probability distribution of the terms in the documents. The probability distribution of the query terms may be different from the distribution of the terms in the documents. We need to evaluate the effect of the probability distribution discrepancies on the query processing efficiency. We plan to conduct experiments on this issue in near future.

References

[1] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, “Freenet: A distributed anonymous information storage and retrieval system,” Lecture Notes in Computer Science, vol.2009, pp.46–66, 2001.

[2] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for Internet applications,” Proc. 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM ’01), pp.149–160, 2001.

[3] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the XOR metric,” Proc. IPTPS’02, 2002.

[4] K. Aberer, P. Cudre-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt, “P-Grid: A self-organizing structured P2P system,” SIGMOD Record, 2003.

[5] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, ACM Press, 1999.

[6] F.M. Cuenca-Acuna, C. Peery, R.P. Martin, and T.D. Nguyen, “PlanetP: Using gossiping to build content addressable peer-to-peer information sharing communities,” Proc. 12th International Symposium on High Performance Distributed Computing (HPDC ’03), 2003.

[7] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer, “MINERVA: Collaborative P2P search,” Proc. 31st International Conference on Very Large Data Bases (VLDB ’05), 2005.

[8] T. Suel, C. Mathur, J. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram, “ODISSEA: A peer-to-peer architecture for scalable Web search and information retrieval,” Proc. 6th International Workshop on the Web and Databases (WebDB’03), 2003.

[9] H. Kurasawa, H. Wakaki, A. Takasu, and J. Adachi, “Data allocation scheme based on term weight for P2P information retrieval,” Proc. 9th Annual ACM International Workshop on Web Information and Data Management (WIDM’07), 2007.

[10] H. Kurasawa, A. Takasu, and J. Adachi, “Huffman-DHT: Index structure refinement scheme for P2P information retrieval,” Proc. 2008 International Symposium on Applications and the Internet (SAINT’08), 2008.

[11] S. Michel, M. Bender, and N. Ntarmos, “Discovering and exploiting keyword and attribute-value co-occurrences to improve P2P routing indices,” Proc. ACM 15th Conference on Information and Knowledge Management (CIKM’06), 2006.

[12] J. Callan, “Distributed information retrieval,” in Advances in Information Retrieval, Kluwer Academic Publishers, pp.127–150, 2000.

[13] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer, “Improving collection selection with overlap awareness in P2P search engines,” Proc. 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’05), pp.67–74, 2005.

[14] M. Bender, S. Michel, P. Triantafillou, and G. Weikum, “Global document frequency estimation in peer-to-peer Web search,” Proc. 9th International Workshop on the Web and Databases (WebDB’06), 2006.

[15] M. Bender, S. Michel, and P. Triantafillou, “P2P content search: Give the Web back to the people,” Proc. 5th International Workshop on Peer-to-Peer Systems (IPTPS’06), 2006.

[16] K. Kenthapadi and G.S. Manku, “Decentralized algorithms using both local and random probes for P2P load balancing,” Proc. Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA’05), 2005.

[17] Y. Zhu and Y. Hu, “Efficient, proximity-aware load balancing for DHT-based P2P systems,” IEEE Trans. Parallel Distrib. Syst., vol.16, no.4, pp.349–361, 2005.

[18] H. Fang, T. Tao, and C. Zhai, “A formal study of information re-

trieval heuristics," Proc. 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04), pp.49–56, 2004.
[19] S.E. Robertson, S. Walker, S. Jones, M.M.H. Beaulieu, and M. Gatford, "Okapi at TREC-3," Proc. TREC-3, pp.109–126, 1994.
[20] D.A. Huffman, "A method for the construction of minimum-redundancy codes," Proc. Institute of Radio Enginners (IRE), pp.1098–1101, 1952.
[21] S. El-Ansary, L.O. Alima, P. Brand, and S. Haridi, "Efficient broadcast in structured P2P networks," Proc. 2nd International Workshop on Peer-To-Peer Systems (IPTPS'03), 2003.
[22] Text REtrieval Conference (TREC), http://trec.nist.gov/
[23] Porter stemmer, http://www.tartarus.org/martin/PorterStemmer/

Hisashi Kurasawa received the B.E. and M.E. in Information Science and Technology from the University of Tokyo, Tokyo, Japan in 2006 and 2008. He is currently a Ph.D. candidate in the Graduate School of Information Science and Technology at the University of Tokyo. His research interests are distributed information retrieval systems and context-aware computing. He is a student member of IPSJ, and DBSJ.

Atsuhiro Takasu received B.E., M.E. and Dr. Eng. from the University of Tokyo in 1984, 1986 and 1989, respectively. He is a professor of National Institute of Informatics, Japan. His research interests are database systems and machine learning. He is a member of ACM, IEEE, IPSJ, DBSJ and JSAI.

Jun Adachi is Professor in the Digital Content and Media Sciences Research Division, National Institute of Informatics (NII), Japan. He is also the Director of the Cyber Sceince Infrastructure Development Department of NII. His professional career has largely been spent in research and development of sholarly information systems, such as NACSIS-CAT and NII-ELS. He is also an adjunct professor of the Graduate School of Information Science and Technology (Department of Information and Communication Engineering), University of Tokyo. His research interests are information retrieval, text mining, digital library systems, and distributed information systems. Adachi received his BE, ME and Doctor of Engineering in Electrical Engineering from the University of Tokyo in 1976, 1978, and 1981, respectively. He is a member of IPSJ, IEEE, and ACM.