Huffman-DHT: Index Structure Refinement Scheme for P2P Information Retrieval

Hisashi Kurasawa
The University of Tokyo
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, JAPAN
kurasawa@nii.ac.jp

Atsuhiro Takasu   Jun Adachi
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, JAPAN
{takasu,adachi}@nii.ac.jp

Abstract

Peer-to-Peer Information Retrieval (P2P IR) systems using a distributed index on a distributed hash table (DHT) can make highly precise searches for documents relevant to a query. However, these systems require a heavy index construction cost, and cause unfair index management costs due to the unbalanced term frequency distribution. We propose a new node access scheme for P2P IR that we call Huffman-DHT. Huffman-DHT uses an algorithm similar to Huffman coding, and modifies the DHT structure based on the term distribution. Huffman-DHT distributes the index construction cost among the nodes equally, and achieves load balancing.

1. Introduction

Information management and retrieval has become a very important part of the rapid increase in digitized and easily accessible information. They are studied from various aspects such as their information processing efficiency in the database research field and their information retrieval (IR) effectiveness in the IR research field. Most of the developed techniques are mainly based on centralized systems. Researchers have recently found peer-to-peer (P2P) network systems attractive as an architecture for information management and retrieval. An information retrieval system on a P2P network (P2P IR) has advantages in their cost, scalability, and dependability. Furthermore, they are suitable for developing and maintaining a system on a large amount of distributed computation resources that are dynamically reconfigured.

Modern IR methods use the occurrence of terms information in document collection, such as the term frequency (TF) that denotes the number of term occurrences in a document and the document frequency (DF) that denotes the number of documents including the term, to measure the relevance between queries and documents [5]. Although the term information can be easily calculated in centralized systems, it is difficult problem in P2P systems. Early P2P systems [2] flooded queries over P2P networks and gathered documents that contained the query terms. They did not fully use the term information, and therefore, the gathered documents contained less relevant documents. They also caused heavy network traffic loads. Recently, several distributed indexing methods for P2P IR systems have been proposed [11, 9, 20, 15]. The indices maintain various information that is useful in IR, including the term information. We can improve the information retrieval effectiveness as well as the query processing efficiency by using these indices.

The problem with the distributed indices is their construction cost. The information of 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, respectively. For this purpose, a distributed hash table (DHT) is often used. It enables us to access the objective node in $O(\log n)$ hops for the number n of nodes in the P2P network. When registering a document into the index, we need to access all the nodes that keep the term information included in the document. Therefore, the construction and maintenance of the index causes heavy traffic.

To balance the index construction cost, we propose a new node access scheme. It is empirically known that 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. However, current distributed indexing methods access nodes responsible to terms in $O(\log n)$ hops, independent of the term probability. Our method finds a node with less hops for more frequently appearing terms. To achieve this, we modify the node access scheme of the DHT using the coding theory technique. The proposed method uses an algorithm similar to Huffman coding, and we call the resul-

978-0-7695-3297-4/08 $25.00 © 2008 IEEE
DOI 10.1109/SAINT.2008.26

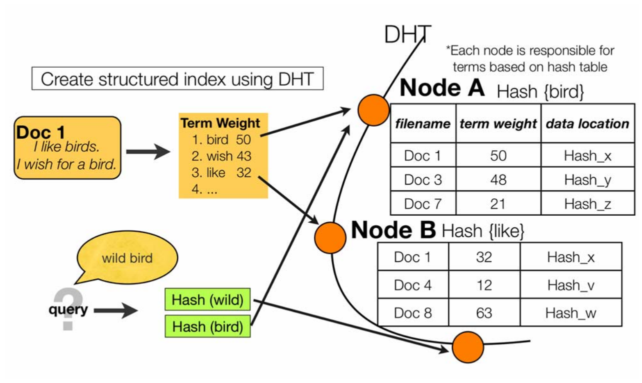
tant scheme *Huffman-DHT*. We show through simulations that the proposed method reduces the average number of hops for a distributed index access, and consequently, balances the traffic load for an index construction.

The rest of this paper is composed as follows. We briefly survey P2P IR systems in Section 2, and overview the indexing schemes for P2P IR systems in Section 3. In Section 4, we introduce a new peer searching method that reduces the network traffic load as well as balances the load of peers. Section 5 presents our experimental results. Finally, Section 6 concludes this paper and addresses some future research directions.

2. Related Work

We start with an overview of the related works using a global index in P2P IR. 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 [11]. On the other hand, in P2P IR systems using structured P2P networks, a node makes a global index using a distributed hash table (DHT) [9, 20, 15]. Every node maintains a part of the global index related to the responsible 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 peer [9]. Each node publishes a document collection summary and sends the summary to the nodes responsible for the terms included in the summary on the DHT. As a result, every node maintains a peer list for the responsible terms. Many peer selection strategies have been proposed for more efficient searching, such as CORI [10], an overlap awareness strategy [8], a global document frequency estimation [7], and term co-occurrences [6, 16]. The other is a term-to-document index where the terms are indexed by document [20, 15]. Every node maintains a document list for the responsible terms on the DHT instead of a peer list. Although making a term-to-document index costs more due to the amount of document insertion when compared to a term-to-peer index, the term-to-document index can directly refer to the term frequency of a specific document.

From the aspect of load balancing, unstructured P2P IR systems are better than structured P2P IR systems, because the nodes in structured P2P IR systems maintain not only their document collection, but also indices of their responsible terms. In structured P2P, the term indices are assigned to nodes using the hash function. The nodes assigned to the indices of frequent terms tend to have heavier loads, because they are frequently accessed in document registration to the index. Many techniques have been proposed for balancing these loads, such as the virtual server [19], the local and random probes algorithms [14], pair-wise interactions of peers[4], and proximity-aware balancing [21]. These



filename	term weight	data location
Doc 1	50	Hash_x
Doc 3	48	Hash_y
Doc 7	21	Hash_z

Doc 1	32	Hash_x
Doc 4	12	Hash_v
Doc 8	63	Hash_w

Figure 1. Indexing Scheme in P2P IR

techniques can improve the node ID assignment in DHT and can be used in any P2P applications. However, they require extra traffic load to suitably modify the IDs.

The search efficiency depends on the quality of the global index. Although structured P2P IR systems require more cost to register a document to the index than unstructured systems, they can retrieve relevant documents with higher precision. In particular, the systems using a term-to-document index have a better search performance. A query-driven index is proposed for ensuring a lower cost for indexing documents in a structured P2P system [18]. In this type of approach, the IR system makes indices for only those terms that have occurred in queries and reduces the number of unused index entries. However, 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 the one in systems using a term-to-document index that has the information of all the terms in the documents. An ideal P2P IR system can make a highly-precise search for documents, and constructs an index at a lower cost. To achieve this goal, we used a term-to-document index 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.

In the P2P IR systems, a node creates a global index on the DHT. 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 for the data of the document. A node registers the document information in the indices that are related to each term in the document.

The term weight in the indices is usually measured using the term and document frequencies. For example, Concordia [15] adopted the Probabilistic Model and uses the term weighting formula presented by [12], which is a variation of BM25 [17]. For a term t in a document d, the weight of t for d is defined as:

$$w(t,d) = \log \frac{N}{df} \cdot \frac{(k+1)\cdot tf}{k\{(1-\alpha)+\alpha \cdot \frac{dl}{avdl}\}+tf}, \quad (1)$$

where k and α are the parameters, N is the number of documents in the collection, tf is the occurrence frequency of t in d, df is the number of documents containing t, dl is the length of d, and $avdl$ is the average document length. The relevance score of query Q is defined as:

$$R(Q,d) = \sum_{t \in Q} w(t,d). \quad (2)$$

In an autonomous and distributed environment like P2P, it is difficult to determine N, $avdl$, and df, because the documents are allocated throughout the network. Many P2P IR systems decided to adopt a method that uses of a global index on the DHT to calculate the df. Nodes can get the number of documents containing the term, that is df, by referring to the index by $O(\log n)$ hops.

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

3.2 Indexing Cost

We compare a term-to-peer index [9] and a term-to-document index [20, 15] from the point of the indexing cost. In P2P IR methods using a term-to-peer index, terms are indexed by a peer's collection. In the term-to-peer index, an index cannot distinguish between the documents in the same peer's collection. On the other hand, in P2P IR methods using a term-to-document index, the terms are indexed by document. That is, an index can recognize the difference between the documents in the same peer's collection. However, the indexing cost of the term-to-document index is heavier than the methods using a term-to-peer index.

Let us first consider the cost T_1 of a node in P2P IR methods using a term-to-peer index registering a collection C_{all} of terms into the index. Suppose $f(t)$ denotes the cost required for calculating the weight of t. Then, T_1 is given as:

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

Let N_{all} denote the number of terms in the collection. Then, the number of registrations to the indices in [9] is represented as:

$$N_{all}.$$

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

$$T_2 = \sum_{i=1}^{N_{doc}} \left(\sum_{t \in C_{D_i}} f(t) \right),$$

where C_{D_i} denotes the set of terms in a document D_i and N_{doc} denotes the number of documents in the collection. Then, the number of registrations to the indices in the P2P IR methods is represented as:

$$\sum_{i=1}^{N_{doc}} N_{D_i},$$

where N_{D_i} is the number of terms in C_{D_i}.

4. Huffman-DHT

4.1. Huffman Coding

In Huffman-DHT, an ID is assigned to each node. In this paper, we denote the ID by a bit vector $b_1 b_2 \cdots b_l$, where b_i $(1 \le i \le l)$ is a 0 or 1. For nodes p and q, the distance between p and q is defined as $|id(p) - id(q)|$ where $id(\cdot)$ denotes 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.

In Huffman-DHT, we construct a coding tree for frequent terms. We need to know the probability distribution of terms when coding. We use the document frequency distribution for coding, because the document frequency of a term denotes the number of accesses to the node responsible to the term when registering documents. 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. Then, we select the top k terms in the descending order of $df(t,S)$ as frequent terms, where k is a parameter of the Huffman-DHT. The remaining terms are assigned to nodes in the same way as in the distributed index [9, 15]. For a term t in the frequent terms T, we estimate its probability by using

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

Then, we construct the coding tree using the Huffman coding algorithm[13]. Figure 2 shows an example of a coding tree. As shown in the figure, we use the binary alphabet for coding.

The reasons why 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 sample of documents.

4.2. Node Assignment

For a frequent term t, let $c_1 \cdots c_m$ be its code. If the code length m is longer than the bit vector length l of the node IDs, then the code is truncated to a l bit, 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 ID is 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. Note that a more frequent term is encoded to a shorter code by Huffman coding, and therefore, it is assigned to a larger encoded node cluster. Figure 3 shows the assignment of the frequent terms and the node assignment in Figure 2.

4.3. Node Search

In Huffman-DHT, all the nodes in a P2P system have a list of the frequent terms and the coding tree described in Section 4.1. Suppose a node referred to as a *query node* searches for a responsible node for a term t.

If term t is a frequent term, the query node encodes t. Otherwise, the node finds term t' in a list of the frequent terms whose hash number is nearest that of term t, and assign 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, return 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.

We can find a responsible node with less hops, because the encoded node cluster is large for a frequent term.

4.4. Document Registration

In Huffman-DHT, a document is registered into the index through the following steps:

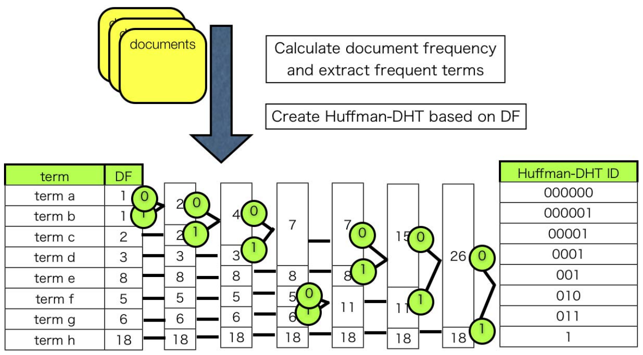


Figure 2. DHT structure refinement process in Huffman-DHT

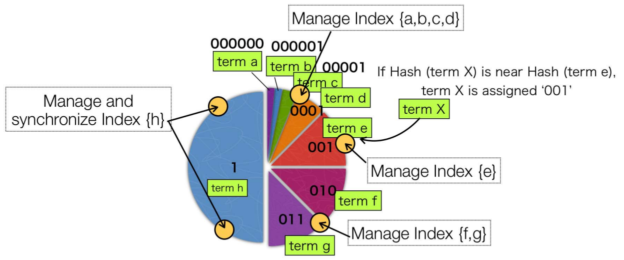


Figure 3. Huffman-DHT ID decision process and node assignment process

1. For each term t in the document:
   (a) Search for a responsible node using the procedure described in section 4.3,
   (b) make the responsible node modify the entry of the term t so that it contains the document information.

Since the node search procedure returns an address of a node in the encoded node cluster, the index update done by the above procedure must be propagated to the other nodes in the same encoded node cluster. This synchronization procedure can be efficiently done because the nodes in the cluster have consecutive IDs and can access each other directly with high probability.

5. Experimental Results

The Huffman-DHT aims at assigning statistically balanced IDs 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 [1]. 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

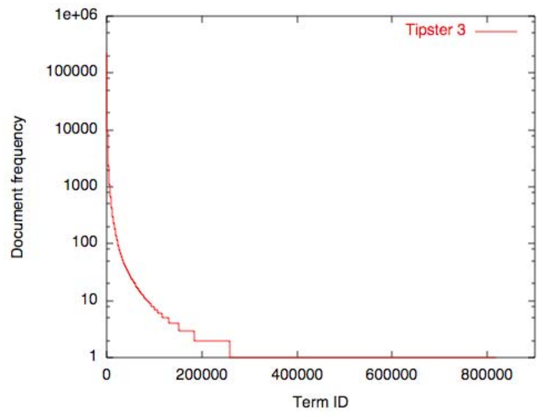


Figure 4. Document Frequency Distribution

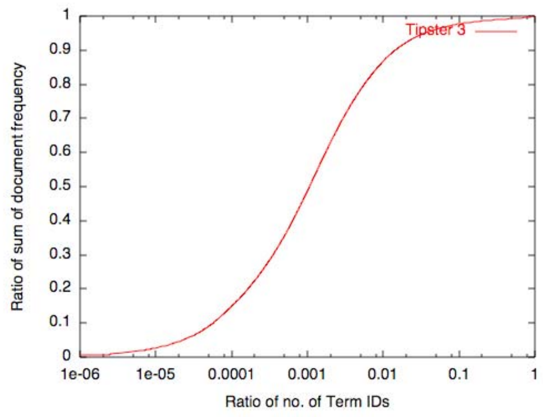


Figure 5. Ratio of Frequent Term Occurrences

Select disks (1991, 1992) copyrighted by Ziff-Davis. We removed the stopwords and converted the remaining words to stems using the Porter stemmer [3].

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, and there are a total number of 53,445,728 term frequencies. Figure 5 shows the document frequencies of terms. 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) occupies 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 the estima-

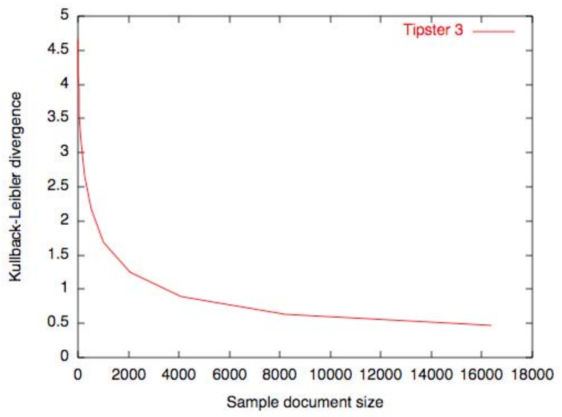


Figure 6. Similarity between Estimation and Actual Distribution

tion of the document frequency distribution over the terms. In Huffman-DHT, we need to estimate the document frequency distribution and extract frequent terms from a small number of samples from the document collection. We compared the document frequencies distribution estimated from randomly selected sample documents to the distribution estimated from the whole Tipster 3 collection. Figure 6 shows the Kullback-Leibler divergence between the two distributions with respect to the number of sample documents. To handle the terms whose document frequency was zero in the sampled documents, we smoothed over the distributions estimated from the sample documents 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. This result indicates that we do not need to frequently update Huffman-DHT.

5.2 Hops Required for Indexing documents

We simulated the number of hops required for indexing documents in 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 depth of the resultant Huffman-DHT tree ranges from 8 to 15. Figure 7 shows the number of hops for indexing documents with respect to the number of nodes in P2P IR systems.

The blue line is the baseline denoting the required hops by a P2P IR system using an ordinary DHT, regardless of the term distribution.

The red line shows the upper bound of required hops calculated by

$$\sum_{t \in T} df(t, D) \times \log n,$$

where T, D, and n denote the number of terms, document collection, and the number of nodes, respectively. This formula is obtained by assuming that we registered each document in the collection one by one to index them and we used $\log n$ hops to search for the node responsible for each term in a document.

The other lines show the required number of hops by the proposed P2P IR system using the Huffman-DHT. All the lines, except the green one, include the number of hops necessary for synchronization.

As shown in the graph, an ordinary DHT (blue line) requires less hops than the upper bound (red line) because we can find the node for a term with less hops if the node is near the registering node. However, this is proportional to the upper bound. On the other hand, the proposed Huffman-DHT successfully reduced the required number of hops for indexing documents if the nodes sent the list of synchronization messages. The extra hops for synchronization within the same encoded node cluster for Huffman-DHT were required, but it reduced the number of hops necessary for searching for the responsible node for a term, It should be noted that Huffman-DHT is effective for large P2P systems that consist of more than 1,000 nodes. The Huffman-DHT is more scalable than an ordinary DHT.

5.3 Load Balancing

In ordinary DHT, the nodes responsible for frequent terms are accessed frequently for document registration. So, these nodes become "hot spots". On the other hand, in Huffman-DHT, the number of nodes responsible for a term is decided according to its document frequency. Therefore, the document registration cost is shared among the nodes.

We conducted an experiment to evaluate the load balancing. In this experiment, we measured the number of accesses to each node to register a document collection in both an ordinary DHT and the Huffman-DHT. Let $a(p)$ denote the number of accesses at a node p. Then, we used the following ratio of the maximum to minimum numbers of access as 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 shows the load concentration with respect to the number of nodes, where the red line represents the load concentration for an ordinary DHT and the other lines represent that for the Huffman-DHT, respectively. As shown in the graph, the Huffman-DHT is more effective, even if synchronization messages are needed when a P2P system consists of more than 100 nodes.

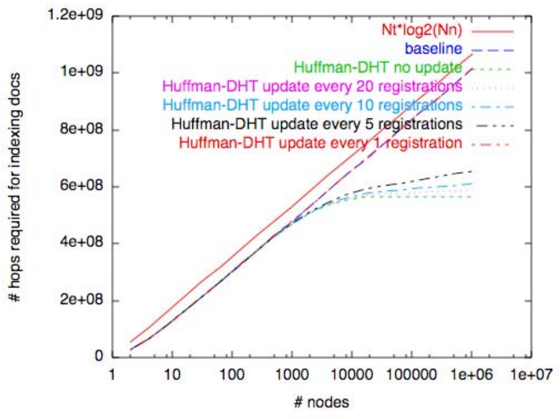


Figure 7. No. of Hops for Indexing Docs

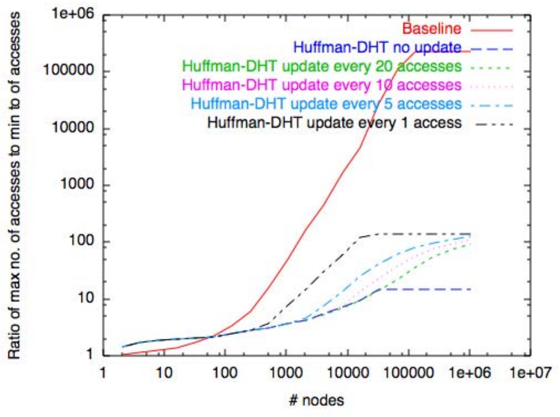


Figure 8. Load Balancing

5.4 Synchronization

In the Huffman-DHT, a node needs to propagate the update of the index to other nodes in the same encoded node cluster. We simulated the cluster size in the Huffman-DHT. Figure 9 shows the maximum size of the clusters in the Huffman-DHT with respect to the number of nodes in the system. We can see from the results that at most 500 nodes need to share the index updates when the P2P system consists of 100,000 nodes.

6. Conclusions

This paper proposed the Huffman-DHT, which improves the load balancing and reduces the distributed index construction cost 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 proved through simulations that the Huffman-DHT reduces the index construction cost, when compared to the

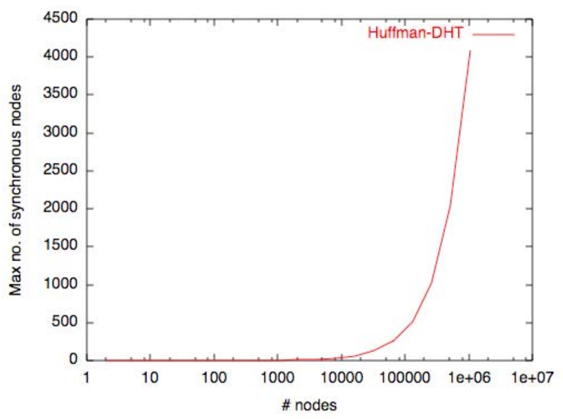


Figure 9. Maximum Node Cluster Size

construction cost of an ordinary DHT. It is most effective when the P2P system consists of about 100,000 nodes and contains many documents. In Huffman-DHT, we need to know the probability distribution of the term occurrence and share the distribution among all the nodes in the probability. We proved that we can estimate the probability distribution from a small number of sample documents and this is sufficient enough to share the probabilities for a small set of frequently appearing terms. 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 the future.

References

[1] *Text REtrieval Conference (TREC)*, http://trec.nist.gov/.

[2] *Gnutella*, http://www.gnutella.com/.

[3] *Porter stemmer*, http://www.tartarus.org/martin/PorterStemmer/.

[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] M. Bender, S. Michel, and P. Triantafillou. P2P Content Search: Give the Web Back to the People. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, 2006.

[7] M. Bender, S. Michel, P. Triantafillou, and G. Weikum. Global Document Frequency Estimation in Peer-to-Peer Web Search. In *Proceedings of 9th International Workshop on the Web and Databases (WebDB'06)*, 2006.

[8] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Improving Collection Selection with Overlap Awareness in P2P Search Engines. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '05)*, pages 67–74, 2005.

[9] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. MINERVA: Collaborative P2P Search. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*, 2005.

[10] J. Callan. Distributed Information Retrieval. *Advances in Information Retrieval, Kluwer Academic Publishers*, pages 127–150, 2000.

[11] 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. In *Proceedings of the 12th International Symposium on High Performance Distributed Computing (HPDC '03)*, 2003.

[12] H. Fang, T. Tao, and C. Zhai. A Formal Study of Information Retrieval Heuristics. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '04)*, pages 49–56, 2004.

[13] D. A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. In *Proceedings of the Institute of Radio Enginners (IRE)*, pages 1098–1101, 1952.

[14] K. Kenthapadi and G. S. Manku. Decentralized Algorithms using both Local and Random Probes for P2P Load Balancing. In *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures (SPAA'05)*, 2005.

[15] H. Kurasawa, H. Wakaki, A. Takasu, and J. Adachi. Data Allocation Scheme Based on Term Weight for P2P Information Retrieval. In *Proceedings of the 9th annual ACM international workshop on Web information and data management (WIDM'07)*, 2007.

[16] S. Michel, M. Bender, and N. Ntarmos. Discovering and Exploiting Keyword and Attribute-Value Co-occurrences to Improve P2P Routing Indices. In *Proceedings of ACM 15th Conference on Information and Knowledge Management (CIKM'06)*, 2006.

[17] S. E. Robertson, S. Walker, S. Jones, M. M. H. Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of TREC-3*, pages 109–126, 1994.

[18] G. Skobeltsyn, T. Luu, I. P. Zarko, M. Rajman, and K. Aberer. Web Text Retrieval with a P2P Query-driven Index. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '07)*, 2007.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01)*, pages 149–160, 2001.

[20] 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. In *Proceedings of the 6th International Workshop on the Web and Databases (WebDB'03)*, 2003.

[21] Y. Zhu and Y. Hu. Efficient, Proximity-Aware Load Balancing for DHT-based P2P Systems. *IEEE Transactions on Parallel and Distributed Systems*, 16(4):349–361, 2005.