Data Allocation Scheme Based on Term Weight for P2P Information Retrieval

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

Hiromi Wakaki
Toshiba R & D Center
1 Komukai Toshiba-cho,
Saiwai-ku
Kawasaki-shi, Yokohama,
JAPAN
hiromi.wakaki@toshiba.co.jp

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

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

ABSTRACT

Many Peer-to-Peer information retrieval systems that use a global index have already been proposed that can retrieve documents relevant to a query. Since documents are allocated to peers regardless of the query, the system needs to connect many peers to gather the relevant documents. We propose a new data allocation scheme for P2P information retrieval that we call Concordia. Concordia uses a node to allocate a document based on the weight of each term in the document to efficiently assemble all the documents relevant to a query from the P2P Network. Moreover, the node encodes the binary data of a document with an erasure code, and Concordia produces an efficient redundancy for counteracting node failures.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed systems*

General Terms

Algorithms, Design, Experimentation

Keywords

Peer-to-Peer information systems, distributed IR, Data allocating

1. INTRODUCTION

The information obtained from networks has been explosively increasing, and search engines are now one of the most important applications on the Web. Almost all search engines are based on centralized systems, such as Google [3]. However, large-scale centralized search systems are expensive and expertise in various areas of system management is needed to maintain their scalability and load balance. Some researchers challenge these problems by using Peer-to-Peer techniques and propose the use of Peer-to-Peer information retrieval (P2P IR) systems.

Information retrieval systems are used to acquire documents relevant to a query. The quality of the search and the gathering of relevant documents are very important issues for P2P IR. In IR, the queries are usually given as a set of terms. Then, the system calculates the relevance between the query terms and the documents in a collection, and presents the documents ranked by the relevance. Relevance is usually measured using the term frequency, which is the number of times the term occurs and the document frequency, which is the number of documents that contain the term. We need a method to retain this frequency information in order to develop a more effective and efficient IR system on a distributed environment like P2P, and because the documents are distributed on a P2P Network. Several indexing methods have recently been proposed that solve this problem [8, 11]. However, the use of this frequency information is limited in current P2P systems. For example, we have to gather ranked documents from publishers or from nodes that keep replicas, which also requires a lot of time. Therefore, we need a scheme that provides an index that maintains the frequency information as well as a function to gather the relevant documents.

In this paper, we propose a scheme that uses a new distributed data allocation scheme for P2P IR, which we call Concordia. Concordia aims at quick gathering of documents relevant to a query from a P2P Network. Concordia allocates a document to the appropriate nodes based on the weight of each term in the document. Concordia provides two kinds of data allocation methods. In particular, Concordia-1 allocates the replica of a shared document to the nodes appropriate to each hash value of the top n high weight terms of the document. By using this scheme, we need fewer requests to gather documents relevant to a query than requests to gather irrelevant ones. In addition, in Concordia-2, we divide the binary data of a document into n chunks that are encoded with an erasure code [26]. Concordia-2 also allocates an appropriate amount of chunks based on the weight of each term in the document to nodes responsible for the term. The document can be assembled from any k out of the n chunks. This means that even if some nodes that store chunks leave the network, the document can be regenerated. As a result, Concordia efficiently gathers documents, that are relevant to a query from a P2P Network, by using a node allocating the encoded binary data of a document, based on the weight of each term in the document. Furthermore, Concordia-2 is able to reduce the document loss probability as compared to Concordia-1. This is because Concordia-2 places chunks encoded with an erasure code, while Concordia-1 places replicas of entire documents. The main contributions of this paper are as follows:

- We propose an indexing method that efficiently retains the term and document frequencies. With this index we can calculate the relevance of queries and documents used in state-of-the-art IR systems.

- We propose a data allocation scheme that quickly gathers relevant documents by accessing indices and documents simultaneously.

The rest of this paper is organized as follows. Section 2 overviews the related works concerning P2P IR and its problems. Section 3 describes Concordia in more detail. Section 4 presents the results from experiments using our scheme. Finally, Section 5 concludes this work and gives a brief outlook on our future work.

2. RELATED WORK

We overview the related works from three aspects: traffic, indexing, and redundancy.

Conventional P2P IR methods search documents by propagating a query to its neighbors [2, 10]. Although these methods adapt to flexible requests, it is difficult to rank all the documents shared in a network by only propagating a query. If a user checks all the documents, the user's system has to connect to all the nodes in the network, and this causes a heavy traffic problem.

Some P2P IR systems solve the heavy traffic problem by indexing documents like centralized IR systems [8, 11, 24]. These P2P IR systems reduce the traffic and response time by sending a request to the nodes that may contain the relevant documents by efficiently referring to the global index. In P2P IR systems using unstructured P2P networks, a node instead of a query floods a global index with a gossip algorithm and shares information about 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) [8, 24]. 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 [8]. 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 [9], an overlap awareness strategy [7], a global document frequency estimation [6], and term co-occurrences [5, 17]. The other is a term-to-document index where the terms are indexed by document [24]. 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 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.

When taking the search response time into consideration, P2P IR systems need a lot of time compared to centralized IR systems, because the indices and documents are distributed on a P2P Network, and we need to connect to many other nodes for referring to indices and gathering documents relevant to a query. P2P IR systems can quickly calculate the relevance scores and rank documents and reduce the search response time by using a term-to-document index. Moreover, some approaches produce smaller traffic and quicker response times by improving the multi-term query execution [27] or a query-driven index [22]. These systems focus on only the efficiency of querying and ranking documents and don't focus on the efficiency of gathering documents. Almost all the existing P2P IR systems using a global index allocate documents regardless of the query, so the systems need to connect to a lot of nodes to gather the relevant documents. The efficiency of gathering data from a P2P Network is improved by using redundant data allocation strategies that use replication [10, 15] or erasures [14]. These strategies allocate shared data based on the request frequency and not based on the term frequency information in a document. Although the response performance of these strategies for popular queries is good, because their allocation schemes are based on the request frequency, the response for unpopular queries needs a lot of time. The ideal P2P IR system can search for documents that are relevant to a query like centralized IR systems by indexing all the documents shared in the network, and can gather the relevant documents quickly by allocating the documents based on the query.

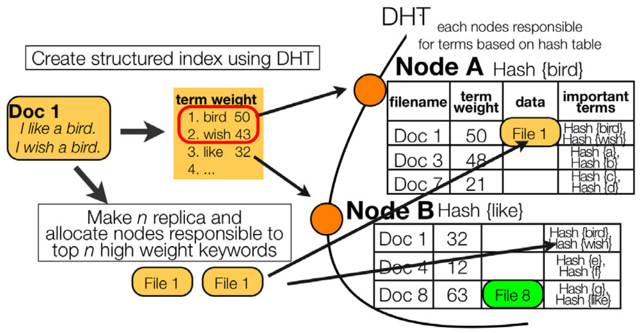
Therefore, we think that a scheme is necessary that uses a new distributed data allocation for P2P IR. With this scheme, we can search for documents just like centralized IR systems, and can efficiently gather documents relevant to not only a popular queries, but also unpopular ones. Moreover, we think that all the documents shared in a network need to be redundant against node failures.

3. ARCHITECTURE OF CONCORDIA

The features of Concordia are as follows.

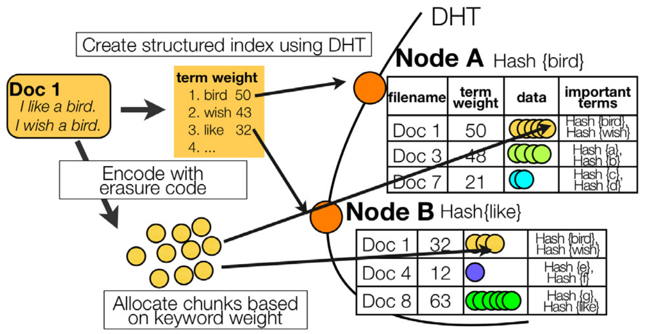
- Concordia can calculate the relevance scores of each document for a query By using the terms indexed by the document using the DHT.

- Concordia efficiently gathers documents relevant to a given query by using a data allocation scheme, where each document is allocated to peers based on the weight of terms included in the document.



DHT
each nodes responsible for terms based on hash table
Create structured index using DHT
Doc 1
I like a bird.
I wish a bird.
term weight
1. bird 50
2. wish 43
3. like 32
4. ...
Make n replica and allocate nodes responsible to top n high weight keywords
File 1
File 1
Node A Hash {bird}
filename
term weight
data
important terms
Doc 1 50 File 1 Hash {bird}, Hash {wish}
Doc 3 48 Hash {a}, Hash {b}
Doc 7 21 Hash {c}, Hash {d}
Node B Hash {like}
Doc 1 32 Hash {bird}, Hash {wish}
Doc 4 12 Hash {e}, Hash {f}
Doc 8 63 File 8 Hash {g}, Hash {like}

Figure 1: Indexing and Data Allocation (Concordia-1)



DHT
Create structured index using DHT
Doc 1
I like a bird.
I wish a bird.
term weight
1. bird 50
2. wish 43
3. like 32
4. ...
Encode with erasure code
Allocate chunks based on keyword weight
Node A Hash {bird}
filename
term weight
data
important terms
Doc 1 50 Hash {bird}, Hash {wish}
Doc 3 48 Hash {a}, Hash {b}
Doc 7 21 Hash {c}, Hash {d}
Node B Hash{like}
Doc 1 32 Hash {bird}, Hash {wish}
Doc 4 12 Hash {e}, Hash {f}
Doc 8 63 Hash {g}, Hash {like}

Figure 2: Indexing and Data Allocation (Concordia-2)

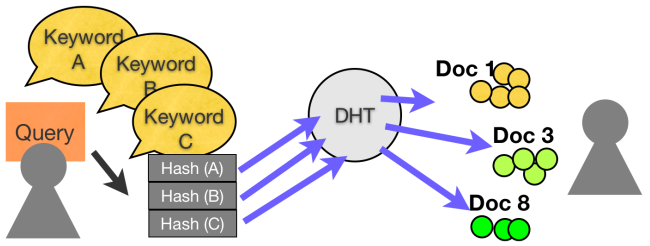


Figure 3: Retrieving and Gathering (Concordia-2)

- Concordia efficiently establishes document redundancy for node failures (Concordia-2) by coding the binary data of a document with an erasure code.

3.1 Overview of Concordia

Concordia is a scheme used for searching and scoring documents in a similar way as centralized IR systems and efficiently gathers documents relevant to a query. Concordia allocates a document based on the weight of each term in the document. Each node manages the indices of its responsible terms and stores documents that weigh high for any one of the terms. In this paper, we don't focus on the query execution like [22, 27], but on the gathering process.

Concordia-1.

In Concordia-1, 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 near the hash value of the term. The inverted index maintains four values for a document, such as the file name, the term weight in the document, the hash value of the important terms in the document, and a pointer of the data of the document. A node registers the term weight and hash values of the important terms in the document (Section 3.2) to the indices that are related to each term in the document. At the same time, the node allocates document replicas to the nodes responsible for each term ranked within the top n weights in the document (Section 3.4). Figure 1 shows the indexing and data allocation in Concordia-1.

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 nodes. If the nodes don't have a relevant document, the user refers to the index for the hash values of the other important terms in the document and gathers the documents from another node that is near any one of the hash values.

Concordia-2.

In Concordia-2, a node creates a global index on the DHT, where the term weights are the same as Concordia-1, but the stored data consists of chunks of encoded and decomposed documents (Section 3.3. The node divides the binary data of a document into n chunks that are encoded with an erasure code. By using the erasure code, the document can be recovered from any k value out of the n chunks. Also the node allocates an appropriate number of chunks based on the weight of each term in the document to the nodes responsible for the term (Section 3.4). That is, the more important the terms of the document a node is responsible for, the more chunks the node stores. We determine the number of chunks that each node stores in the following way; we set the amount of chunks ratio in n chunks that stored with a term index equal to the term weight ratio in the sum of the term weights in the document. The document can still be assembled by using this allocation method, even if some nodes that store chunks leave. Figure 2 shows the indexing and data allocation in Concordia-2.

The system calculates the relevancy between the given query and documents using the index in the same way as Concordia-1. On the other hand, it assembles the chunks of the relevant documents from the nodes. Figure 3 shows the retrieving and gathering of data in Concordia-2. If the user focuses on precision, the user gathers chunks from the highly relevant documents. On the other hand, if the user focuses on recall, the user gathers all the chunks of documents that are stored at the nodes.

After retrieving and gathering data, a document is decoded if the user has over or equal to k chunks of the document. If the number of chunks that the user could get is less

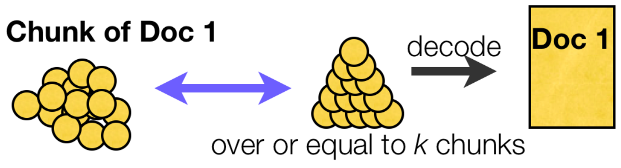


Figure 4: Decoding Documents (Concordia-2)

than k, the user gathers extra chunks. If the nodes don't have enough chunks of a relevant document, the user refers to the index for the hash values of the important terms in the document and gathers the extra chunks of the document from another node that is near any one of the hash values. Figure 4 shows the decoding of documents in Concordia-2.

3.2 Term Weighting Formula in P2P Environment

The term weighting formula is a very important factor for Concordia for accurately searching and efficiently gathering documents. Concordia takes a lot of time making a global index containing the term weights, but reduces the document retrieval time. Many information retrieval methods have been proposed, such as the Vector Space Model and the Probabilistic Model.

We adopted the Probabilistic Model and use the term weighting formula presented in [12], which is a variation of BM25 [20]. For a term t in a document d, the weight of t for d is defined as:

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

where $w(t,d)$ represents the weight of term t appearing in document d, 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:

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

When calculating the term weight defined in Eq. (2), we need to know the total number N of documents, the average document length $avdl$, and the document frequency df, as well as the term frequency tf. 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. We decided to adopt a method that makes use of a global index on the DHT to calculate df, which is similar to related works [8, 25]. Each node registers the document information for every term in the document in its collection to the node responsible for the term on the DHT. As a result, every node maintains an index concerning the responsible term. We can get the number of documents containing the term, that is df, by referring to the index.

The number of documents in collection N and the average document length $avdl$ may be estimated by using the hash sketches [13], such as the distributed hash sketches [18]. However, for our Concordia prototype, we decided that the numbers would be set constants for precisely comparing the relevance scores of the documents using the data allocation scheme in Concordia.

3.3 Data Partitioning with an Erasure Code

In Concordia-2, a node divides the binary data of a document into n chunks that are encoded with an erasure code. The document can be recovered from any k value out of the n chunks by using this erasure code. That is, even if $(n-k)$ chunks are not gathered due to node failures, we can still assemble the document. There are several kinds of erasure codes, such as a scheme using n coordinate pairs on a polynomial of a degree $(k-1)$ [21], and a scheme using simultaneous equations containing k parameters [14]. The decoding method of these schemes is used to solve a set of linear equations.

The (k, L, n) ramp secret sharing scheme [26] is a variation of the secret sharing scheme [21]. In this scheme, the bit-length of each chunk is $\frac{1}{L}$ of the binary data of the document. The scheme is defined as:

$$\begin{aligned} S &= S_0||S_1||S_2|| \cdots ||S_L \\ y &= S_0 + S_1 x + \cdots \\ &\quad + S_L x^L + r_1 x^{L+1} + \cdots + r_{n-L} x^{k-1} \quad mod\ p, \end{aligned}$$

where S is the binary data of a document, $||$ is the bit concatenation operator, and p is a prime number satisfying $p > max(S_i)(0 \le i \le L)$.

The size of a chunk is about $\frac{S}{L}$, and the size of all the chunks is about $\frac{n \cdot S}{k}$.

We use the (k, L, n) ramp secret sharing scheme in Concordia-2. We define $L = k + 1$ to reduce the size of a chunk.

3.4 Data Allocation Scheme Based on the Term Weight

In P2P IR, not only are the searched documents relevant to a query important, but efficiently gathering the documents is also important. If a node that has an index about a term in a query holds a document relevant to the query, the system can reduce the number of requests to the other nodes and can efficiently gather the relevant documents. However, if every node holds the documents that an index in the node refers to, the size of the data shared in the network is very large. We reduce the size using the term weights.

To calculate the relevance given by Equation (2), we need to connect to a set of nodes that are responsible for the query term. If the relevant documents are allocated to the same nodes, we skip connecting to the other nodes when gathering the relevant documents. That is, if the binary data of a document is allocated based on the weight of each term in the document, the more relevant document is easily gathered than any of the other documents.

Therefore, Concordia-1 allocates a replica of a shared document to the nodes appropriate to each hash value of the top n highly weighted terms of the document. In addition, in Concordia-2, a node divides the binary data of a document into n chunks that are encoded with an erasure code. The node also allocates an appropriate amount of chunks based on the weight of each term in the document to the nodes responsible for the term.

3.5 Node and Data Management using DHT

The DHT is a distributed system that provides a lookup service. A single key is assigned to data with a common hash function. A range of key values are assigned to each node, and this node is responsible for the data whose key value is within the range. A node is responsible for maintaining the data with a key that is near the key of the node. The DHT is designed to be scalable, fault tolerant, and self-organizing. The most popular DHTs are Chord [23], Tapestry [28], CAN [19], and Kademlia [16].

In Concordia, a node creates a global index and manages the documents by using the DHT. Each node is responsible for the terms with keys that are near the key of the node. Each node receives information concerning the documents containing any one of its responsible terms from the other

```
<top>
<head> Tipster Topic Description
<num> Number: 101
<dom> Domain: Science and Technology
<title> Topic: Design of the "Star Wars" Anti-missile Defense
System
<desc> Description:
Document will provide information on the proposed configuration,
components, and technology of the U.S.'s "star wars" anti-missile
defense system.
<smry> Summary:
Document will provide information on the proposed configuration,
components, and technology of the U.S.'s "star wars" anti-missile
defense system.
<narr> Narrative:
A relevant document will provide information which aids descrip-
tion of the design and technology to be used in the anti-missile de-
fense system advocated by the Reagan administration, the Strate-
gic Defense Initiative (SDI), also known as "star wars." Any re-
ported changes to original design, or any research results which
might lead to changes of constituent technologies, are also rel-
evant documents. However, reports on political debate over the
SDI, or arms control negotiations which might encompass the SDI,
are NOT relevant to the science and technology focus of this topic,
unless they provide specific information on design and technology.
<con> Concept(s):
1. Strategic Defense Initiative, SDI, star wars, peace shield
2. kinetic energy weapon, kinetic kill, directed energy weapon,
laser, particle beam, ERIS (exoatmospheric reentry-vehicle inter-
ceptor system), phased-array radar, microwave
3. anti-satellite (ASAT) weapon, spaced-based technology, strate-
gic defense technologies
<fac> Factor(s):
<nat> Nationality: U.S.
</nat>
<def> Definition(s):
</top>
```

Figure 5: Sample of TREC-2 ad hoc & TREC-3 routing topics

nodes, and manages the indices of the terms. We adopted the Kademlia DHT into our system. By using Kademlia, a node can contact at most $O(logN)$ nodes to find the binary data of a document or a node in an N-node network, with a high probability.

4. EVALUATION

The purpose of Concordia is to achieve a mechanism to gather number of documents relevant to a query. In addition, Concoridia-2 aims at robust document replication for node failures.

4.1 Documents Gathered with only Connections to Index Nodes

In Concordia, the system first obtains the data from the documents relevant to a query from the nodes responsible for an index of each term in the query. The data in Concordia-1 are replicas of the relevant documents and the data in Concordia-2 are chunks of relevant documents. At that time, the number of requests to the other nodes is equal to the number of terms in the query, that is the maximum distributed environment assumed for the experiment. For our simulation experiment, we evaluated the search results before extra data is gathered from the other nodes, not including the index nodes.

The evaluation is performed on the Tipster 3 collection that was used at TREC [1]. The collection consists of 336,310 documents, and includes material from the San Jose Mercury News (1991), the Associated Press (1990), U.S. Patents

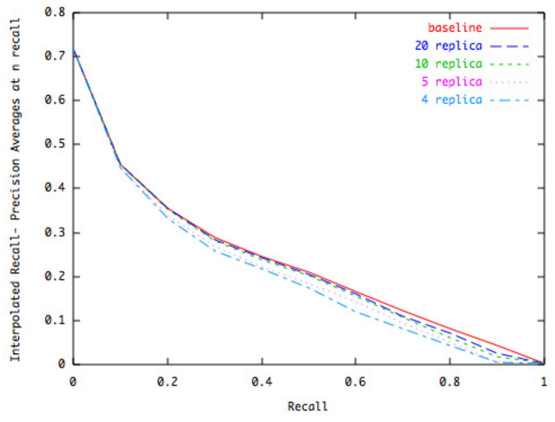


Figure 6: Interpolated Recall- Precision Average (Concordia-1)

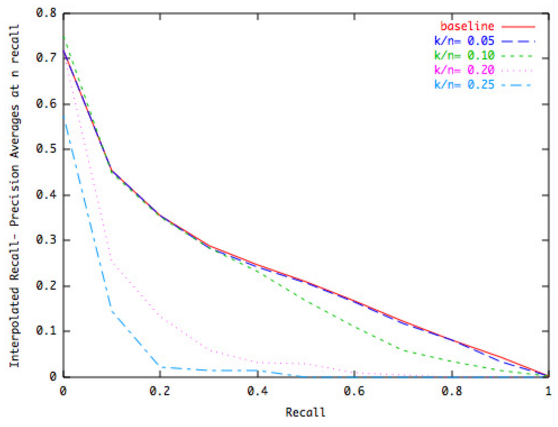


Figure 7: Interpolated Recall- Precision Average (Concordia-2)

(1983-1991), and Information from the Computer Select disks (1991, 1992) copyrighted by Ziff-Davis. We removed the stopwords and stem each remaining term using the Porter stemmer [4]. Then, the term weights in each document are calculated using Eq. (2). We temporarily set k to 100 and α to 0.5. The standard averaged precision for 50 topics in TREC-2 ad hoc & TREC-3 routing topics is evaluated by the TREC Eval tool. Figure 5 shows a sample of the topics. Figure 6 shows the results from an evaluation in Concordia-1, and Figure 7 shows the results in Concordia-2. The baseline in each figure was the standard averaged precision of the search results after gathering all of the relevant documents, i.e., the performance of centralized IR scheme.

In each method, the search result with a higher redundant coding is more precise. In contrast, a search result with a lower redundant coding is less precise, and the system has to gather extra data to retrieve the relevant documents. Moreover, when comparing Concordia-1 with Concordia-2 at the same redundancy, the search result in Concordia-1 was better.

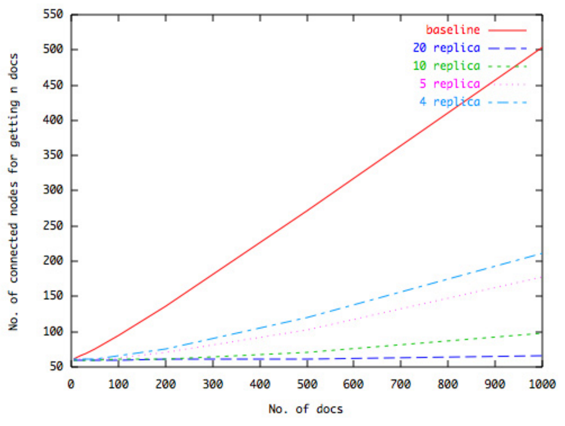


Figure 8: Number of Connected Nodes for Gathering Top n Relevant Docs (Concordia-1)

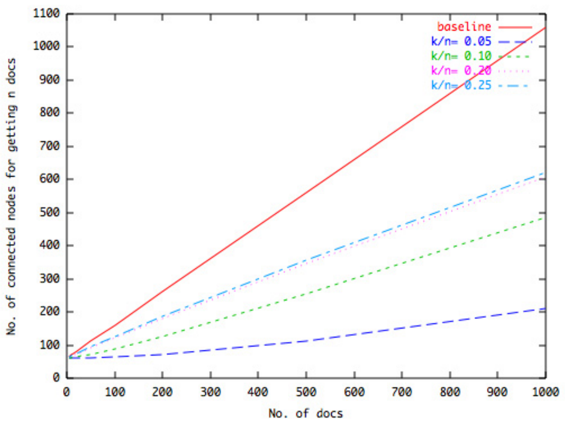


Figure 9: Number of Connected Nodes for Gathering Top n Relevant Docs (Concordia-2)

4.2 Number of Connected Nodes for Gathering Documents

We conducted a simulation experiment on the efficiency of gathering relevant documents in Concordia. The efficiency was evaluated by measuring the number of connected nodes for gathering the top n relevant documents.

The evaluation was performed on the same collection and topics as the experiment in section 4.1. We simply set the number of nodes in the network to the number of terms in the collection. Figure 8 shows the results of the evaluation in Concoredia-1, and Figure 9 shows the results in Concordia-2. The baseline in each figure was a method that randomly allocates the encoded binary data of the documents to the peers.

It is clear from the results in Figs. 8 and 9 that Concordia can gather the relevant documents from less nodes than a random allocation scheme. That is, in Concordia, the system can efficiently gather the documents relevant to a query from a P2P Network by using a node allocating the binary data of a document based on the weight of each term in the document. Moreover, when comparing Concordia-1 with Concordia-2 at the same redundancy, the efficiency of gathering the relevant documents in Concordia-1 is better.

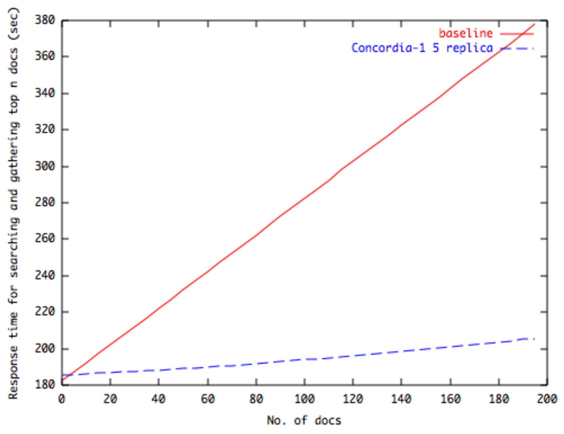


Figure 10: Response Time for Gathering Top n Relevant Docs (Concordia-1)

4.3 Search Response Time

Concordia aims at reducing the response time by simultaneously accessing indices and documents. We conducted an experiment using a large distributed computing environment to evaluate the search response time in Concordia-1.

The evaluation was performed on the same collection as the experiment in section 4.1. The collection consists of 336,310 documents. We used 248 host nodes in four PC Clusters. A Pentium M 1.8 GHz processor and 1 GB main memory computer was used for 127 of the host nodes, and the other host nodes used a Core 2 Duo 2.33 GHz and 4 GB main memory computer. The nodes are connected by a gigabit Ethernet. The experiment used 1,240 peers running as separate processes on the nodes. In the experiment, the system allocated five replicas of a shared document. The search response time was evaluated by measuring the time for gathering the top n relevant documents for 50 topics in TREC-2 ad hoc & TREC-3 routing topics. The average number of terms in the topics was 62.72. Figure 10 shows the average search response time for the 50 topics. The baseline in the figure shows the search response time of another method that allocates five document replicas to peers regardless of the query, which is random peers.

It is clear from the results in Figure 10 that Concordia takes less time to gather documents relevant to a query. This means that Concordia efficiently gathers documents for the user that are relevant to a query from a P2P Network, by using a node allocating a document, based on the weight of each term in the document. It took more than 180 seconds to retrieve relevant documents in each result for this experiment. One reason is that the topics contain a lot of terms. In this Concordia prototype, the system needs to refer to as many indices as there are terms and this takes a lot of time. I think that the search response time could be shorter than for this Concordia prototype if it used query execution techniques like [27, 22].

Table 1: Comparison of Redundancy

	Concordia-1	Concordia-2
Coding method	replica	Erasure code
No. of nodes holding data	$\frac{n}{k}$	$\frac{n}{k} \sim n$
File size (piece)	F	$\frac{F}{k}$
File size (total)	$\frac{F \cdot n}{k}$	$\frac{F \cdot n}{k}$
Document loss probability	$p^{\frac{n}{k}}$	$\sum_{i=0}^{k-1} {}_nC_i p^{n-i} \cdot (1-p)^i \sim p^{\frac{n}{k}}$

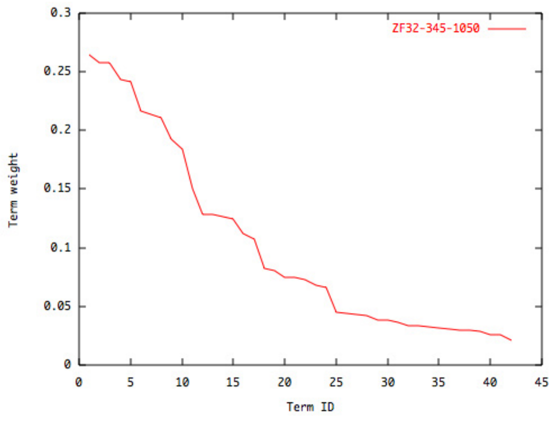


Figure 11: Term Weight Distribution (ZF32-345-1050)

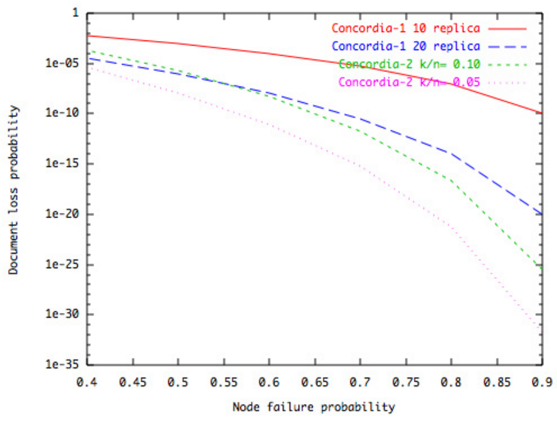


Figure 12: Document Loss Probability (ZF32-345-1050)

4.4 Redundancy Using an Erasure Code

Concordia-2 encodes the binary data of a document with an erasure code, and aims at a stable gathering in a P2P environment where node failures happen. Table 1 shows a comparison between the redundancies in Concordia-1 and Concordia-2. The document loss probability in Concordia-2 is dependent on the distribution of chunks.

We simulated the document loss probability in Concordia. The evaluation was performed on a Tipster 3 collection document. The unique number of the document was ZF32-345-1050. Figure 11 shows the term weight distribution of the document. Figure 12 shows the document loss probabilities in Concordia-1 and Concordia-2.

The document loss probabilities were found to be better in Concordia-2 than in Concordia-1 when comparing the two at the same redundancy. That is, Concordia-2 achieves a more efficient redundancy by using the erasure code.

5. CONCLUSION AND FUTURE WORK

We have proposed Concordia in this paper, which is a scheme that uses a new distributed data allocation in a P2P environment. By using a node allocating a document based on the weight of each term in the document, the system efficiently gathers the documents relevant to a query from a P2P Network. Concordia-1 is designed to efficiently gather relevant documents. Concordia-1 allocates a replica of a shared document to the nodes appropriate to each hash value of the top n high weight terms of the document. In addition, Concordia-2 is designed to perform an efficient redundancy for counteracting node failures. In Concordia-2, a node divides the binary data of a document into n chunks that are encoded with an erasure code. The node also allocates chunks based on the weight of each term in the document to the nodes responsible for the term.

We think that Concordia has two problems. One problem is the storage cost of unpopular documents. A node allocates replicas of documents regardless of the request frequency in Concordia. As a result, the system quickly gathers documents relevant to any query at the sacrifice of the processing cost of document registrations, from the time that the documents are registered to the indices in Concordia. However, the storage cost of unpopular documents is equal to that of popular documents. The total storage cost of unimportant data in Concordia is higher than the existing methods. We should reduce the storage cost while continuing to efficiently gather the relevant documents. The other problem is the load balance in Concordia. In Concordia, a node only allocates data based on the weight of each term in the document. The efficiency of gathering documents would be reduced if the nodes responsible for very popular terms had cheaper network environments or poor computing resources. We should also improve the load balance underlying the physical networks and resources.

Therefore, we are working on improving the data distribution method to resolve these problems. We are also extending Concordia to a method that can perform a multi-term query execution.

6. REFERENCES

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

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

[3] *Google*, http://www.google.com/.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. *Lectured Notes in Computer Science*, 2009:46–66, 2001.

[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] P. Flajolet and G. Martin. Probabilistic Counting Algorithms for Data Base Applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[14] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, 2005.

[15] J. Kangasharju, K. W. Ross, and D. A. Turner. Optimizing File Availability in Peer-to-Peer Content Distribution. In *Proceedings of the 26th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '07)*, 2007.

[16] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Proceedings of IPTPS'02*, 2002.

[17] 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.

[18] N. Ntarmos, P. Triantafillou, and G. Weikum. Counting at Large: Efficient Cardinality Estimation in Internet-Scale Data Networks. In *Proceedings of 22nd International Conference on Data Engineering (ICDE'06)*, 2006.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01)*, pages 161–172, 2001.

[20] 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.

[21] A. Shamir. How to Share a Secret. *Commn. ACM*, 22(11), 1979.

[22] 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.

[23] 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.

[24] 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.

[25] C. Tang and S. Dwarkadas. Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval. In *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[26] H. Yamamoto. On Secret Sharing Systems Using (k,l,n) Threshold Scheme. *IECE Trans*, J68-A(9):945–952, 1985.

[27] J. Zhang and T. Suel. Efficient Query Evaluation on Large Textual Collections in a Peer-to-Peer Environment. In *Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, 2005.

[28] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report USB//CSD-01-1141, U. C. Berkeley Technical Report, 2001.