Abstract

The large unstructured text collections demand full-text search capabilities from IR systems. Current systems typically allow users only to connect to a single database (or site) at a time. An IR system should be able to provide multiple users with concurrent, efficient access to multiple text collections distributed across remote sites. However, high resource demands of networked IR systems limit their performance as the size and the number of data collection increase. In general, the size of inverted file index for full-text search is either as big as that of original text collection or more. Our goal is to represent a new index management method, which is fast enough in practice while the performance is not limited by the size and number of data collections.

Keywords: IR system, information retrieval, index, distributed system, digital library

1. Introduction

The exponential growth of the internet accelerates the need for new tools to help users cope with the vast amount of information. In order for users to effectively access these collections, information retrieval (IR) systems must provide coordinated, concurrent, and distributed access [1]. Consequently, effective and efficient index management is becoming ever more important and at the same time more difficult.

Centralized index approach [4,5], where a centralized index is maintained as a composition of all local indexes, is one approach to the problem. Without having to query individual collections, the center site knows where everything is, and consequently provides fast access to relevant documents. However, since the index for full-text search is in general at least as big as the original text collection, a centralized index can become very large. Usually the index becomes so large that building and maintaining a central index is not feasible for most distributed IR applications.

Another approach is to have each local site keep its local index and send requests to all sites for every query. This is distributed index approach [2,3]. There is no worry of a central index growing too large since there is none. However, distributed index systems have a number of redundancies which contributes to low overall performance.

Because a distributed IR system does not know which collections have relevant information for a given query\(^1\), it has to forward the query to all of the collections in order not to miss any results. Typically, more than half of the data collections will not have any documents relevant to the user’s request, and will return nothing but a message indicating they have no matched data. This is a big waste of processing time and, more importantly, network bandwidth.

A more serious redundancy can be found when identical results\(^2\) are transferred multiple times over time. Generally, certain terms occur very frequently in user queries. Such situation is intensified when the search domain is subjected to a specific field such as computer, law, and medicine, et al. Under such conditions, it is very likely that the current query has been asked before and has received corresponding results. Nevertheless, an IR system employing the distributed index approach will send the query and transfer the result list repeatedly because it has no information about it.

---

\(^1\) A query in IR context typically consists of a list of keywords connected by logical operators. To simplify our discussion, we will assume that a query consists of a single term.

\(^2\) We consider as results only the location information or links to actual documents. The transfer of the content of a document is a separate process not directly relevant to indexing.
We present in this paper, a new index method that eliminates the shortcomings of both of the above mentioned approaches. The proposed system uses a centralized query-based virtual index (QVI) whose size is significantly smaller than fully replicated centralized index. Forwarding queries to individual sites are kept to minimal thus eliminating redundancies inherent in distributed index approach. Simulation results indicate that the average query response time is close to that of the centralized index approach with the index size being less than 1% of the size of the centralized index.

Overviews of our system and our IR model are provided in section 2. In section 4, we discuss performance characteristics of the proposed system. Future works and concluding remarks are given in section 4.

2. QVI (Query-based Virtual Index) System

![Flow of QVI operation](image)

2.1. Architecture and Operation

Because it is not feasible to use fully replicated centralized index, we adopted the distributed index approach but eliminated the redundancies by keeping a central index only on those terms that have been queried previously. We call this index Query-based Virtual Index (QVI).

Figure 1 provides the architecture of a system using QVI. Search is carried out by the QVI engine that consists of two search managers: QVI manager and DI (Distributed Index) manager. QVI manager is responsible for using and maintaining QVI. A search process begins with Query_to_QVI, which examines whether the current query term exists in QVI. If it does, the result is returned directly from the index. If it does not, which means it is a new term that had never been used in previous queries, the DI manager issues Query_at_Network. The DI manager is responsible for forwarding the query to and receiving results from the distributed data collections. The Response Organizer rearranges the results into a common form used by the QVI engine. QVI manager updates QVI with this new result. Hence, next time the same query arrives, QVI can reply immediately without any remote searches.

The search process can be divided into two function sequences. One is the miss case, where the query term is not in QVI, and the other is the hit case, where the query term is in QVI.

\[
\text{miss : User } \rightarrow \text{Query_to_QVI } \rightarrow \text{Query_at_Network } \\
\quad \rightarrow \text{Organize_Response } \rightarrow \text{User } \tag{1}
\]

\[
\text{hit : User } \rightarrow \text{Query_to_QVI } \rightarrow \text{User } \tag{2}
\]

It is obvious that hit cases are desirable. Fully replicated centralized index systems will only have hit cases at the price of maintaining the large centralized index, whereas distributed index systems can only have miss cases since they keep no central indexes. For our QVI system, hits occur when user query is duplicated. We show that query duplication in actual IR systems is high enough to bring the average response time of our QVI system to that of a fully replicated centralized index system.

2.2. Query Duplication Model

We have studied the query logs of two IR systems in operation in Korea. One is a general Web search service, SYMMANY, operated by HanCom Inc., and the other is KRISTAL-II, a typical IR system for bibliographic search on science and engineering subjects operated by Korea Research & Development Information Center. Interesting common characteristics were observed in the two systems. In SYMMANY, as the number of total query increased from 247,559 to 258,195, the number of distinct query terms increased from 29,755 to 30,598. The number of distinct query terms is only 12% of the total number of user queries. This implies that 88% of total queries have used terms that had been used already. The growth amount shows another characteristic of user queries. The fraction of distinct terms decreased from 12% to 11% as the number of queries grew. Continued observation showed that this fraction converges to some value with small variance. KRISTAL-II showed similar results except that distinct terms were 44% of total queries. In spite of the differences, these systems show that there is a high degree of query duplication in IR systems.

We define a new parameter, \(\text{RepCnt}_{N,K}\), which
represents average repetition count of one query term, as

\[ \text{RepCnt}_{N,K} = \frac{N}{K} \quad (3) \]

where \( N \) is the total number of search trials, and \( K \) is the total number of distinct query terms in \( N \) queries.

For example, if there are 5 distinct query terms in 10 queries, \( \text{RepCnt}_{5,10} \) is 2. This means that on the average a distinct query term is repeated 2 times. A \( \text{RepCnt}_{N,K} \) of 1 means that every query is distinct. On the contrary, a \( \text{RepCnt}_{N,K} \) of \( N \) means that a single term was queried \( N \) times. For SYMMANY, \( \text{RepCnt}_{N,K} \) ranged from 8.1 to 8.4, and for KRISTAL-II, from 3.9 to 4.5. For a QVI system, \( \text{RepCnt}_{N,K} \) is an important indicator of its performance characteristics.

3. Performance Analysis and Simulation Results

We define \( t_{\text{miss}} \) and \( t_{\text{hit}} \) as the response time for the miss case and the hit case, respectively. \( t_{\text{operation}} \) is the time it takes for \( \langle \text{operation} \rangle \) to run. Then, from (1), (2), we have the followings;

\[ t_{\text{miss}} = t_{\text{query.to.qvi}} + t_{\text{query.at.network}} + t_{\text{Organize.Response}} \]
\[ t_{\text{hit}} = t_{\text{query.to.qvi}} \]

In addition, let \( n_{\text{miss}} \) and \( n_{\text{hit}} \) be the number of \textit{misses} and \textit{hits}, respectively, during \( N \) queries. Then,

\[ n_{\text{miss}} = K = N/\text{RepCnt}_{N,K} \quad \text{and} \quad n_{\text{hit}} = N - n_{\text{miss}} = N - N/\text{RepCnt}_{N,K} \]

Now, the average response time for \( N \) queries can be defined as,

\[ T_{\text{avg}}(\text{RepCnt}_{N,K}) = \left( t_{\text{hit}} \times n_{\text{hit}} + t_{\text{miss}} \times n_{\text{miss}} \right) / N \]
\[ = t_{\text{query.to.qvi}} + t_{\text{query.at.network}} + t_{\text{Organize.Response}} / \text{RepCnt}_{N,K} \quad (4) \]

Because both \( t_{\text{query.to.qvi}} \) and \( t_{\text{Organize.Response}} \) are much smaller than \( t_{\text{query.at.network}} \), they can be ignored in most cases. So, the average response time is almost reverse proportional to \( \text{RepCnt}_{N,K} \). When \( \text{RepCnt}_{N,K} \) is 1, all queries are sent to network and the situation is similar to simple distributed index system. However, our observation in section 2 shows that \( \text{RepCnt}_{N,K} \) is usually bigger than 4 in general.

For our simulation, we used three sets of articles (in English) taken from several technical report archive sites. Each set made up 1GB of documents. We used \texttt{glimpse}[8] for creating our index, and implemented the fully replicated centralized index (CI), simple distributed index (DI), and QVI system independently.

We randomly selected 10,000 distinct terms among the index terms extracted from all three sets. Tests sets for seven different values of \( \text{RepCnt}_{N,K} \) were created. For each \( \text{RepCnt}_{N,K} \), we generated random query sequences varying \( N \) from 1,000 to 10,000 by 1,000 (adjusting \( K \) to maintain the given \( \text{RepCnt}_{N,K} \)).

Figure 2 shows change of average response time for each \( \text{RepCnt}_{N,K} \). As expected, the average response time of QVI system decreases as \( \text{RepCnt}_{N,K} \) increases. It decreases rather steeply and smoothes out at about \( \text{RepCnt}_{N,K} = 4 \). For \( \text{RepCnt}_{N,K} \) of 4 or greater, QVI is about 5 times faster than DI and almost as fast as CI.

Figure 3 shows the influence of network traffic to QVI performance. We ran the same simulation at three different times of day. As can be expected, heavy traffic contributes to slower network communication and, thus,
1.3 sec. When 8, difference is less than 0.1 sec. In other words, if $RepCnt_{N,K}$ is sufficiently high, performance is maintained at a satisfactory level regardless of network traffic.

As for the size of QVI, when $RepCnt_{N,K}$ is 4, the index occupied only 1.34Mbytes of storage for the largest query set. This is about 0.2% of the size of the full centralized index. Moreover, this proportion gets even smaller as $RepCnt_{N,K}$ increases.

4. Conclusion

We have proposed a new type of index method which can be used to improve the performance of distributed IR systems. QVI uses a kind of centralized index, and so performs remarkably well compared to simple distributed index system. In order to keep this central index minimal so as to avoid the size problem of the fully replicated centralized index, only those terms that have been used previously in user queries are indexed. At the beginning of service, QVI will have an empty index, and so runs mainly as a distributed index system. However, as time goes on, the index will grow (or mature) and hit cases will increase. It is well known that a small fraction of terms make up most of the user queries. And in such cases, QVI will show performance close to full central index with storage requirements less than 1% of that required by full index.

The average query term repetition count, $RepCnt_{N,K}$, is presented as a parameter characterizing a QVI system’s performance. Our simulation results show that near optimum performance can be achieved for $RepCnt_{N,K}$ values of 4 or greater. The fact that the $RepCnt_{N,K}$ values for the two commercially running systems which we examined fall in this range is indicative that QVI will perform well in actual applications.

We are currently working on the internal index structure for QVI in order to represent richer information that can be extracted from a query and its results. This information will be used to answer in a more intelligent way later queries sharing the same query term. Also, the issue of populating QVI initially (jump starting) is being studied.

References