行政院國家科學委員會專題研究計畫成果報告
嵌入式資訊網系統之高階計算伺服器設計(I)

NSC Project Reports

計畫編號：NSC 89-2213-E-009-062
執行期限：88 **年** 8 **月** 1 **日至** 89 **年** 7 **月** 31 **日**

主持人：鍾崇斌　交通大學資訊工程學系

一、中文摘要

網際網路的快速成長，為資訊擷取系統帶來了新的應用與新的挑戰。在本計畫中，我們探討如何利用壓縮轉置檔進行平行資訊檢索。為了縮短磁碟讀取時間，我們利用文件分群技術以發揮 d-gap **壓縮方法之效益。此外，我們更進一步提出轉置檔切割方法，以利用壓縮轉置檔進行平行資訊擷取。實驗結果顯示，所提之文件分群方法能降低** 16%-23%**的磁碟擷取時間。而轉置檔切割方法，更使得平行資訊擷取能在此基礎上，再獲得幾乎線性的效益成長。本研究的結果，揭示了設計大規模網際網路搜尋引擎的方法。**

關鍵詞： 反置檔案，平行處理，d-gap **壓縮法。**

Abstract

The rapid growth of Internet brings wide variety of applications as well as new system design challenges on information retrieval systems. In this project, we investigate parallel Boolean query processing on compressed inverted file. To reduce disk access time, document clustering techniques are applied to exploit the effectiveness of d-gap compression scheme. Moreover, posting file partitioning schemes are proposed for parallel query processing on the compressed inverted file. Experiment shows that 16%-23% of the disk access time can be reduced by the proposed compression technique. Moreover, with the posting file partitioning scheme, parallel query processing achieves almost linear speed-up based on the enhanced sequential query processing technique. This project shows the way to design a scalable search engine on Internet.

Keywords: inverted file, parallel processing, d-gap compression.

2. Introduction

The rapid growth of Internet brings wide variety of applications as well as new system design challenges on information retrieval systems. Information retrieval systems, widely known as search engines, can be found in various Web applications for searching homepages, research papers, news articles, and product information. However, the problem of information explosion overwhelms the load of CPU and disk for an information retrieval server. The response time to serve a user query scales as the size of the document collection grows. Reducing the query response time is a key issue in designing a scalable information retrieval system. In this project, we investigate inverted file compression and parallel information retrieval techniques to reduce query response time.

3. Inverted File Compression

In an information retrieval system, a user sends a command which contains some query terms to the system and waits for the results indicating documents containing these query terms. The system searches these query terms in the inverted file to know which documents contain them and return these documents' identifications or abstract to the user. Thus, for each distinct term *t*, an inverted file contains its corresponding list, called inverted list (or posting list), of the form

$$< term; f_t; D_1, D_2, D_3, ..., D_{f_t} >,$$

where identification D_i indicates the document that contains term *t* and frequency f_t indicates the total number of documents in which term *t* appears [1].

One obvious fact is that the size of an inverted file will enlarge greatly if the collected documents in the system database increase larger and larger. As a result, the searching time of the inverted file will become unacceptable. Compression is the most compact way to reduce the size of an inverted file. A common compression technique is to sort the document identifications (document IDs) of each inverted list in increasing order, and then replace each document ID by the difference between it and its

predecessor to form a list of *d*-gaps [1, 2, 3]. For example, the inverted list <term; 7; 15, 43, 90, 8, 51, 130, 61> can be sorted to <term; 7; 8, 15, 43, 51, 61, 90, 130> and transformed into d-gap representation as <term; 7; 8, 7, 28, 8, 10, 29, 40>. That is, the sequence of the document IDs is transformed to a sequence of smaller numbers which can be effectively coded by some prefix-free codes, for examples, gamma code, delta code, and Golomb code. The nature of these codes is their variable-length representations in which small numbers are considered more likely, and coded more economically, than large ones [1].

3.1. Method for inverted file compression

The distribution of the gap values in the inverted file is fluctuant. A large gap value may be potentially the same as the original large document ID and the saving of the encoding technique cannot be achieved easily [1]. Most studies have focused on the improvement of encoding technique for *d*-gap, but they have not emphasized on the characteristics of document IDs to reduce the gap values. In fact, when the gaps appear in the inverted list of a term *t*, it means that the documents with the original document IDs all contain the term *t*; that is, there are some *clustering* properties among these documents. The clustering property does exist among documents because if they have the same topics, or belong to a specific domain, their contents usually share a large amount of identical terms. As a result, there is a high probability that the documents which involve the same clustering property may appear in the same inverted list. If we reassign closer IDs to these documents, the gap values will thus be effectively reduced.

In this project, we propose a document ID reassigning mechanism by exploiting the clustering property in documents to reduce the gap values. In this mechanism, we count the number of common terms between two documents and define it as the relation of these two documents. According to the relationship we defined, all the documents will form a weighted graph, called relation graph. We adopt some heuristic algorithms to route the clusters in the relation graph and get a new routing path. Thus, we can reassign IDs to the documents according to this path order. The simulation results show that we can enhance about 10 to 15 percent for the compression rates of inverted files on the traditional *d*-gap technique.

In addition, we also propose an encoding technique to make up the shortcomings of *d*-gap technique, called first-ID-division technique. This technique divides the first document ID in an inverted file by a constant and stores the ID as the quotient and remainder instead of the original ID. Simulation results show that we can further improve about 8 percent of compression rate.

3.2. Evaluation of compression efficiency

To allow practical comparison of various algorithms and techniques, experiments have been performed on some real-life document collections. In this project, we choose “PC” (Paper Collection) and “FBIS” (Foreign Broadcast Information Service) as our experimental collections. The “PC”, which represents the small-scale collection, is a set of papers, including seven subgroups, PC1, PC2, PC3, PC4, PC5, PC6, and PC7, from various proceedings and journals of computer science domain. On the contrary, the “FBIS”, which represents the large-scale collection, is contained in the fifth disk of TREC, an acronym for Text REtrieval Conference. This collection has been distributed to research groups worldwide for researching and evaluating information retrieval experiments.

Figure 1 compares the gap distribution of value 1 to 20 in “PC7” by applying the greedy algorithm for document ID reassignment. The number of gap value 1 is increased largely and the numbers of other gap values are all decreased after reassigning document IDs. Thus, most gaps after reassigning document IDs become smaller than the gaps of the original document IDs. It is because that the greedy algorithm exploits the clustering property among documents and those documents which have closer relation can be assigned closer IDs. If these smaller gaps are encoded by the same encoding technique, which encode smaller values in fewer bits, we will get better compression rate.

Figure 2 shows that the compression rate can be totally improved about 9 percent by gamma code and 6 percent by delta code. The compression rate improvement of FBIS inverted file is less than that of PC7 inverted file mentioned above. The main reason is that the original cluster relation may be broken by the partition of collection. This problem can be considered as future research.

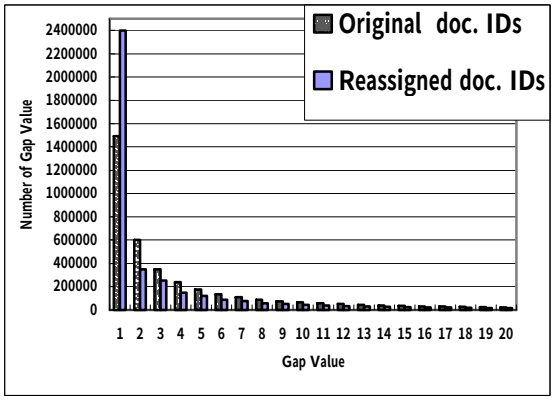


Figure 1. Gap distribution of value 1 to 20 in "PC7"

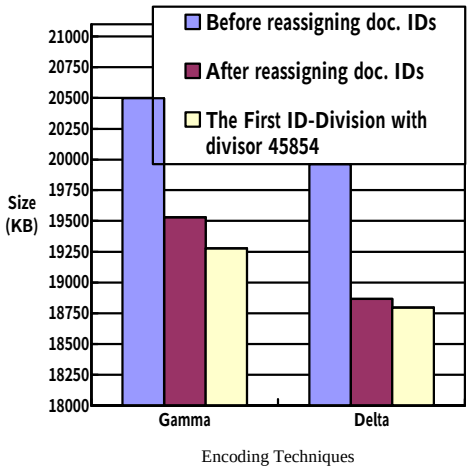


Figure 2. Compression rate improvement of inverted file

4. Posting File Partitioning for Parallel Query Processing

Besides investigating compression techniques to reduce disk access time, we also investigate parallel Boolean query processing on a network of workstations to reduce the query response time. Queries are processed on a cluster of workstations—each has its own CPU, memory, and disks—interconnected by a local area network. The key issue to parallelize query processing is posting file partitioning. In contrast to previous works[4][5], we parallelize both CPU computation and disk accesses. The partitioning avoids transferring posting lists between workstations during parallel query processing. Moreover, the partitioning schemes are designed to balance the load of workstations without affecting the effectiveness of (document clustered) d-gap compression scheme. Experiment shows that almost linear speedup can be achieved without affecting the compression efficiency.

4.1. Parallelism within Boolean query processing

The key idea to parallelize posting list processing is depicted in Figure 3. We use the notation WS_k to denote a workstation with workstation ID k. Each posting list is partitioned by document IDs and each workstation stores a portion of the partitioned posting list. During parallel query processing, each workstation consults only its own locally resident data to establish its own partial answer list.

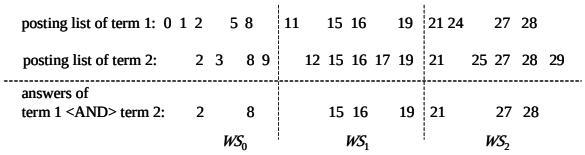


Figure 3. Posting file partitioning method

In general, a posting file partitioning scheme is specified by a mapping A that maps each document ID d to a workstation WS_k. We use the notation $A(d)=k$ to denote that document ID d is mapped to WS_k. If document ID d is mapped to WS_k, then all postings of document d appears and only appears in the local posting file of WS_k.

4.2. Posting File Partitioning Schemes

We now apply the partitioning by document ID principle to produce the partitioned posting file. We assume that the input posting file is d-gap compressed and documents are clustered to reduce gap between document IDs in a posting list, as stated in previous sections. We propose two partitioning schemes and compare the performance of the two schemes.

4.2.1. Consecutive partitioning scheme

A straightforward approach is to let each local posting list be a segment of the input posting list, as shown in Figure 3. Each workstation covers a set of consecutive document IDs. Let D be the number of documents in the entire document collection and M be the number of workstations. The mapping A_{consec} is to map each document ID d to a workstation by:

$$A_{consec}(d) = \lfloor d / \lceil D/M \rceil \rfloor$$

In each workstation, a document can be represented by a local document ID and d-gap compression scheme is applied on the local document IDs presenting the local posting lists. We let the local document IDs in a workstation starts from zero. The rule $LID_{consec}(d)$ to assign local document ID for each document d is:

$$LID_{consec}(d) = d - A(d) * \lceil D/M \rceil$$

A drawback of the consecutive scheme is that it fails to achieve load balancing among workstations with the document clustering assumption. We thus propose the second scheme – the interleaving scheme.

4.2.2. Interleaving partitioning scheme

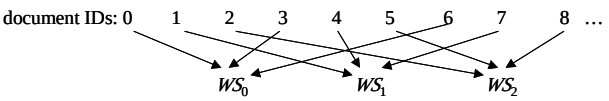
Figure 4 depicts how a posting list is partitioned by the interleaving scheme. As shown in Figure 4(a), each workstation is mapped with a set of interleaved document IDs. Let M be the number of workstations. The rule $A_{intlv}(d)$ is to map each document ID d to a workstation by:

$A_{intlv}(d) = d \bmod M$

The workstation ID that d is mapped to is the remainder of d/M. With the interleaving scheme, postings in a posting list will be distributed to multiple workstations even when the document IDs are in a small range.

However, mapping rule $A_{intlv}(d)$ increases the gap between document IDs in a local posting list. The gap between document IDs in a local posting list is at least M. The effectiveness of d-gap compression scheme on the local posting file will be reduced if documents are represented by the original document IDs. This increases disk space and access time to store and retrieve a local posting list. We notice that , to represent a document in a workstation, only the quotient of d/M is required. We thus take the quotient as the local document ID $LID_{intlv}(d)$ in a workstation.

$$LID_{intlv}(d) = \lfloor d/M \rfloor$$



(a) Mapping document IDs to workstations

posting list: 1, 2, 4, 6, 7, 10, 11, 12, 14, 15

represented using original document ID:	6, 12, 15	1, 4, 7, 10	2, 11, 14
represented using local document ID:	2, 4, 5	0, 1, 2, 3	0, 3, 4
	WS_0	WS_1	WS_2

(b) Partitioning a posting list

Figure 4. Interleaving partitioning scheme

4.3. Performance evaluation

We implement an experimental information retrieval system to evaluate the effectiveness of the proposed partitioning schemes. We follow [3] to generate queries to evaluate the performance. Figure 5 depicts the performance of parallel query processing using the proposed partitioning schemes. The metric is the speed up to sequential query processing:

$$\text{Speed-up} = \frac{Time_{\text{sequential}}}{Time_{\text{parallel}}}$$

We evaluate the speedup when the number of workstations M=2,4,6,8, and 10 are employed. Figure 5 shows that almost linear speedup can be achieved. As expected, interleaving scheme outperforms the consecutive scheme.

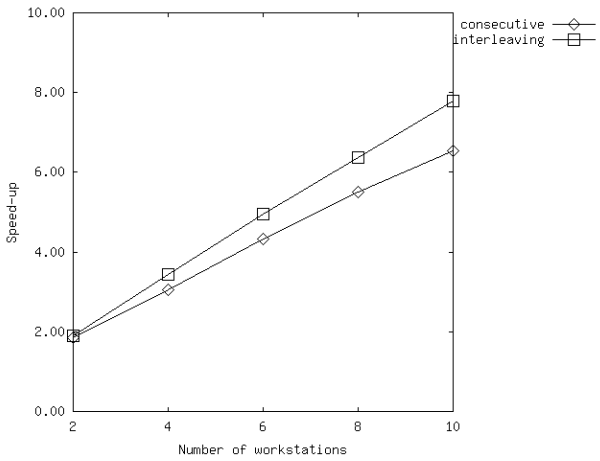


Figure 5. Speed-up of parallel query processing

Reference:

[1] G. Salton and M. J. McGill, *Introdiction to Modern Information Retrieval*, McGraw Hill, 1983.

[2] J. Zobel and A. Moffat, "Adding Compression to A Full-text Retrieval System," *Software Practice and Experience*, Vol. 25, pp. 891-903, August 1995.

[3] A. Moffat and J. Zobel, "Self-indexing inverted files for fast text retrieval," *ACM Trac. Information Systems*, Vol. 14, No. 4, pp. 249-279, Oct. 1996.

[4] B. S. Jeong and E. Omiecinski, "Inverted file partitioning schemes in multiple disk systems," IEEE Transactions on Parallel and Distributed Systems, Vol. 6, No. 2, pp. 142-153, 1995.

[5] B. A. Riberio-Neto, J. P. Kitajima, G. Navarro, "Parallel generation of inverted files for distributed text collections," Proceedings of the 18th International Conference on the Chilean Society of Computer Science, pp. 149-157, 1998

行政院國家科學委員會補助專題研究計畫成果報告

※　※※※※※※※※※※※※※※※※※※※※※※※※※※

※　　　　　　　　　　　　　　　　　　　　　　　　　　　※

※　　**嵌入式資訊網系統之高階計算伺服器設計(I)**　　※

※　　　　　　　　　　　　　　　　　　　　　　　　　　　※

※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：□個別型計畫　　□整合型計畫

計畫編號：NSC 89-2213-E-009-062－

執行期間：88 **年** 08 **月** 01 **日至** 89 **年** 07 **月** 31 **日**

計畫主持人：鍾崇斌

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份

□赴大陸地區出差或研習心得報告一份

□出席國際學術會議心得報告及發表之論文各一份

□國際合作研究計畫國外研究報告書一份

執行單位：交通大學 資訊工程學系

中　華　民　國 89 **年** 10 **月** 31 **日**