The Journal of Supercomputing, 19, 251–265, 2001

Information Retrieval on an SCI-Based PC Cluster*

SANG-HWA CHUNG, HYUK-CHUL KWON, KWANG RYEL RYU,
YOOJIN CHUNG, HANKOOK JANG AND CHAM-AH CHOI shchung@hyowon.pusan.ac.kr

School of Electrical and Computer Engineering, Pusan National University, Pusan 609-735, Korea

Received August 2000

Abstract. This article presents an efficient parallel information retrieval (IR) system which provides fast information service for the Internet users on low-cost high-performance PC-NOW environment. The IR system is implemented on a PC cluster based on the scalable coherent interface (SCI), a powerful interconnecting mechanism for both shared memory models and message-passing models. In the IR system, the inverted-index file (IIF) is partitioned into pieces using a greedy declustering algorithm and distributed to the cluster nodes to be stored on each node's hard disk. For each incoming user's query with multiple terms, terms are sent to the corresponding nodes which contain the relevant pieces of the IIF to be evaluated in parallel. The IR system is developed using a distributed-shared memory (DSM) programming technique based on the SCI. According to the experiments, the IR system outperforms an MPI-based IR system using Fast Ethernet as an interconnect. Speed-up of up to 5.0 was obtained with an 8-node cluster in processing each query on a 500,000-document IIF.

Keywords: cluster computing, PC cluster, SCI, information retrieval, inverted index file

1. Introduction

As more and more people are accessing the Internet and acquiring a vast amount of information easily, more people consider that the problem of information retrieval (IR) resides no longer in the lack of information, but in how we can choose the right information with speed from a vast amount. Many of us have already experienced that some IR systems provide information service much faster than others. How fast an IR system can respond to users' queries highly depends on the performance of the underlying hardware platform. Therefore, most of the major IR service providers have been urged to spend several hundred thousand dollars to purchase their hardware systems. However, this cost is too high for many small businesses on the Internet.

In this article, as a cost-effective solution for this problem, a PC cluster interconnected by a high-speed network card is suggested as a platform for fast IR service. With the PC cluster, a massive digital library can be efficiently distributed to PC nodes by utilizing local hard disks. Besides, every PC node can act simultaneously as an entry to process multiple users' queries.

*This work was supported by Korea Research Foundation Grant (KRF-98-001-E01171).

It is extremely important to select a network adapter to construct a high-speed system area network (SAN). The Fast Ethernet card or the Myrinet card can be used for a message-passing system. For a distributed shared memory (DSM) system, the scalable coherent interface (SCI) card can be considered. Fast Ethernet developed for LAN is based on complicated protocol software such as TCP/IP, and its bandwidth is not high. The Myrinet [2] card is a high-speed message passing card with a maximum bandwidth of 160 Mbyte/s. However, the network cost is relatively high because Myrinet requires crossbar switches for the network connection. Besides, its message-passing mechanism is based on time-consuming operating system calls. For applications with frequent message-passing, this can lead to performance degradation. To overcome the system call overhead, systems based on user-level interface for message-passing without intervention of operating system have been developed. Representative systems include AM [5] FM [10], and U-Net [1]. Recently, Myrinet is also provided with a new message-passing system called GM [9] which supports user-level OS-bypass network interface access.

The SCI (ANSI/IEEE standard 1596-1992) is designed to provide a low-latency (less than 1 μs) and high bandwidth (up to 1 Gbyte/s) point-to-point interconnect. The SCI interconnect can assume any topology including ring and crossbar. Once fully developed, the SCI can connect up to 64K nodes. Since the SCI supports DSM models that can feature both of NUMA and CC-NUMA variants, it is possible to make transparent remote memory access with memory read/write transactions without using explicit message-passing. The performance of the SCI-based systems has been proven by the commercial CC-NUMA servers such as Sequent NUMAQ 2000 [8] and Data General's Aviion [3].

In this research, the SCI is chosen as an underlying interconnecting mechanism for clustering. The parallel IR system is implemented on an SCI-based PC cluster using a DSM programming technique. In the IR system, the inverted-index file (IIF) is partitioned into pieces using a greedy declustering algorithm and distributed to the cluster nodes to be stored on each node's hard disk. An IIF is the sorted list of terms (or keywords), with each term having links to the documents containing that term. For each incoming user's query of multiple terms, terms are sent to the corresponding nodes which contain the relevant pieces of IIF to be evaluated in parallel. An MPI-based IR system using Fast Ethernet as an interconnect is also constructed for the purpose of comparison.

2. Related works

Although there has been no previous work in developing a parallel IR system under PC cluster environment, some of the relevant research in the area of parallel IR is as follows. Sharma [14] presented a parallel IR system implemented on a hypercube-based machine. The processors are connected in a hypercube and each of the processors has its own hard disk. The documents are clustered into groups of closely related ones and these clusters are equally distributed to all the disks. Upon receiving a user's query, the host computer broadcasts the query to all the processors. Each processor searches the document clusters of its own to retrieve those clusters

relevant to the query and transmits them back to the host. Since those processors that do not have any clusters relevant to the given query will be idling, they are assigned different queries for parallel query processing and for higher utilization. One drawback of this cluster-based method is that its retrieval accuracy is not as good as those of other systems because the result of retrieval is given back in units of clusters rather than individual documents. In addition, the efficiency of parallel processing in this system becomes dependent on the locality, because only those processors having relevant clusters in their disks participate in the retrieval process.

Cringean [4] proposed a Transputer-based system consisting of a triple-chain network. In this system the retrieval is done in two stages . In its first stage, the signature files are scanned to exclude large numbers of documents that are apparently irrelevant. In the second stage, a full-text search is done on the documents by using the Boyer–Moore algorithm to determine whether they are good matches for the query. The documents are all stored in the disk of the root processor, and they are transmitted one-by-one to each of the processors for text searching. The processor that completes the search of a document reports the result back to the root processor and then receives a new document from the root. Since the documents to be searched are all transmitted from the hard disk of the root processor, the overhead of accessing the root hard disk becomes heavy and there occurs a bottleneck at the communication link closest to the root processor.

Stanfill [15] proposed an MPP model which is implemented on the Connection Machine. In his model, the ids of all the documents in the document collection are first distributed to all the processors in an interleaved way, and then the terms in the IIF are distributed to the processors. Let t, d, and $w(t,d)$ be a term in the IIF, the document in which t appears, and the weight value of t in d, respectively. The term t together with the weight value $w(t,d)$ are uploaded to the memory of the node to which the document d is assigned. This guarantees that the score for a document can be computed without any communication with other nodes. However, Stanfill's model does not deal with the issue of hard disk storage or I/O parallelism which is often the more critical factor to the performance of parallel IR systems.

3. PC cluster-based IR system

3.1. Typical IR system on uniprocessor

Figure 1 shows the structure of a typical IR system implemented on a uniprocessor. As shown in the figure, once a user's query with multiple terms is presented to the system, for each query term in turn the IR engine retrieves relevant information from the IIF in the hard disk. When all the information is collected, the IR engine performs necessary IR operations, scores the retrieved documents, ranks them, and sends the IR result back to the user. For the efficient parallelization of the system, it is important to find out the most time-consuming part in executing the IR system. Using the sequential IR system developed previously [11], the system's execution time is analyzed as shown in Figure 2. In the sequential system, the most

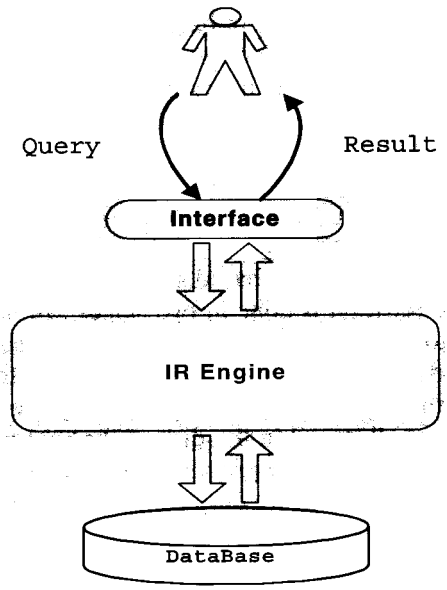

Figure 1. A typical IR system.

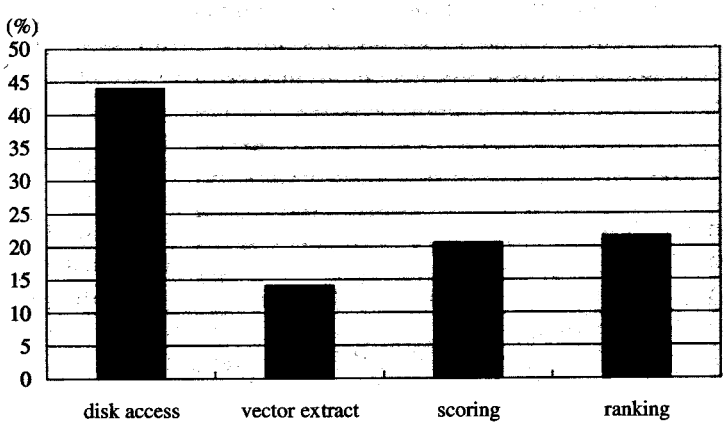

Figure 2. Execution time analysis in the sequential IR system.

time-consuming part is disk access. Thus, it is necessary to parallelize disk access. This can be done by partitioning the IIF into pieces and distributing the pieces to the processing nodes in a PC cluster.

3.2. *Declustering IIF*

Most of the current IR systems use a very large lookup table called an IIF to index relevant documents for given query terms. Given a term t_i, the IIF lookup returns a list $((d_{i1}, w_{i1}), (d_{i2}, w_{i2}), \ldots, (d_{im}, w_{im}))$ which consists of all the documents con-

taining the term tagged with the corresponding weights. The jth pair (d_{ij}, w_{ij}) in the list indicates that d_{ij} is the id of the document containing t_i and w_{ij} is the weight of t_i in d_{ij}. Given a query, all the query terms are looked up from the IIF to retrieve relevant document ids and corresponding term weights. Next, the documents are scored based on the term weight values and then ranked before they are reported back to the user.

Since our IR system processes user's query in parallel on a PC cluster, it is desirable to have the IIF appropriately declustered to the local hard disks of the processing nodes. We can achieve maximum parallelism if the declustering is done in such a way that the disk I/O and the subsequent scoring job are distributed as evenly as possible to all the processing nodes. An easy random declustering method would be just to assign each of the terms (together with its list of document id and weight pairs) in the IIF lexicographically to each of the processing nodes in turn, repeatedly until all the terms are assigned. In this article, we present a simple greedy declustering method which performs better than the random method.

Our greedy declustering method tries to put together in the same node those terms which have low probability of simultaneous occurrence in the same query. If the terms in a query all happen to be stored in the same node, the disk I/O cannot be done in parallel and also the scoring job cannot readily be processed in parallel. For an arbitrary pair of terms in the IIF, how can we predict the probability of their co-occurrence in the same query? We conjecture that this probability has a strong correlation with the probability of their co-occurrence in the same documents. Given a pair of terms t_i and t_j, the probability of their co-occurrence in the same documents $P_{co}(t_i, t_j)$ can be obtained by

$$P_{co}(t_i, t_j) = \frac{N_{ij}}{N}$$

where N_{ij} is the number of documents in which the two terms t_i and t_j co-occur, and N is the number of all the documents in a given document collection. We calculate this probability for each and every pair of terms by preprocessing the whole document collection.

When the size of the document collection is very large, we can limit the calculation of the co-occurrence probabilities only to those terms which are significant. The reason is that about 80% of the terms in a document collection usually exhibit only a single or double occurrences in the whole document collection and they are unlikely to appear in the user queries. Also, since the number of terms in a document collection is known to increase in log scale as the number of documents increases, our method will not have much difficulty in scaling up. As more documents are added to the collection, however, re-calculation of the co-occurrence probabilities would be needed for maintenance. But, this would not happen frequently because the statistical characteristics of a document collection does not change abruptly.

In the first step of our greedy declustering algorithm, all the terms in the IIF are sorted in the decreasing order of the number of documents where each term appears. The higher this number, the more important the term is in the sense that it is quite likely to be included in many queries. This is especially true when the queries are modified by relevance feedback [13]. This type of terms also have a

longer list of documents in the IIF and thus causes heavier disk I/O. Therefore, it is advantageous, whenever possible, to store these terms in different nodes for the enhancement of I/O parallelism. Suppose there are n processing nodes. We assign the first n of the sorted terms to each of the n nodes in turn. For the next n terms, each term is assigned to the node which contains a term with the lowest probability of co-occurrence. From the third pass of the term assignment, a term t_i is assigned to the node k such that the summation

$$\sum_{t_j \in T_k} P_{co}(t_i, t_j)$$

is the lowest, where T_k is the set of all the terms already assigned to the node k. This process repeats until all the terms in the IIF are assigned. When the size of the document collection is large and thus the co-occurrence probability data is available only for those terms which are significant, the remaining terms are declustered by the random method mentioned previously.

3.3. *Parallel IR system model*

The PC cluster-based parallel IR system model is shown in Figure 3. The IR system consists of multiple processing nodes and one of which is an entry node. The participating nodes are PCs with local hard disks and connected by an SCI-based high-speed network. The overall working mechanism of the parallel IR system model can be explained as follows. The entry node accepts a user' query and distributes query terms to processing nodes based on the declustering information described in Section 3.2. Each processing node consults the partitioned IIF using the list of query terms delivered from the entry node, and collects the necessary document list

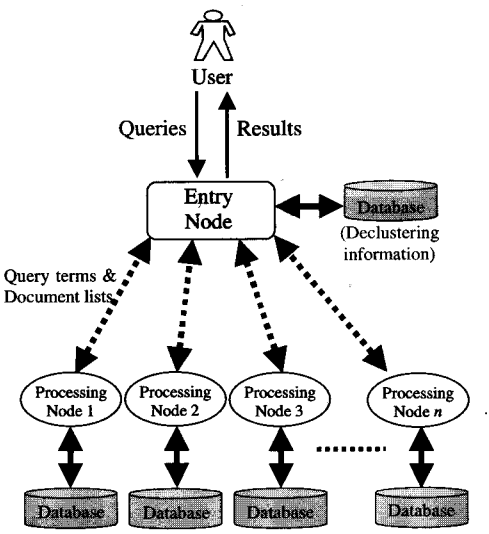

Figure 3. Parallel IR system model.

for each term from the local hard disk. Once all the necessary document lists are collected, they are transmitted to the entry node. The entry node collects the document lists from the participating processing nodes, performs required IR operations and ranks the selected documents according to their scores. Finally, the sorted document list is sent back to the user as an IR result.

The details of IR operations are explained as follows. To measure a similarity between a document and a query, a cosine measure [12] is used in this article. The cosine measure is one of the widely used measures and is developed from the cosine of the angle between the weighted vectors representing the document and the query. Its definition is

$$\text{similarity } (d, q) = \frac{\sum_t (w_{dt} \times w_{qt})}{\sqrt{\sum_t w_{dt}^2} \times \sqrt{\sum_t w_{qt}^2}}$$

where w_{dt} is the weight of term t in document d and w_{qt} is its weight in query q (*this is the inner product* of the weighted document and query vectors, normalized by their lengths). w_{dt} is defined as $f_{dt} \times idf_t$, where f_{dt} is term t's frequency in the document d and idf_t is so called term t's inverse document frequency. The inverse document frequency is given by $\log_2(N/D) + 1$, where N is the total number of documents in the collection and D is the number of documents containing the term. w_{qt} is same as idf_t if term t is in query q and 0 otherwise. Let W_d and W_q be defined as follows.

$$W_d = \sqrt{\sum_t w_{dt}^2}, \quad W_q = \sqrt{\sum_t w_{qt}^2}$$

Since W_q is the unique value for query q and

$$W_d = \sqrt{\sum_t (f_{dt} \times idf_t)^2}$$

calculation of a value

$$\sum_{t \in q} f_{dt} \times idf_t^2 / W_d$$

for each document d is sufficient to rank documents for query q.

The algorithm of the parallel IR system model is described by the flowchart in Figure 4. The first three steps are initialization steps for query processing. In step 1, the entry node calculates idf_t^2 for each term t and W_d for each document d before query processing. In step 2, the entry node sends idf_t^2 for each term t to the processing node containing the corresponding IIF for that term. In step 3, every processing node receives idf_t^2 for each term t in the partitioned IIF from the entry node.

To process a query, steps 4–8 are executed. In step 4, the entry node accepts a user's query and distributes query terms to the corresponding processing nodes based on the declustering information described in the previous subsection. From now on, we call the node that contains the corresponding IIF for at least one query term, an *active node*. The set of query terms in active node i is denoted by q_i.

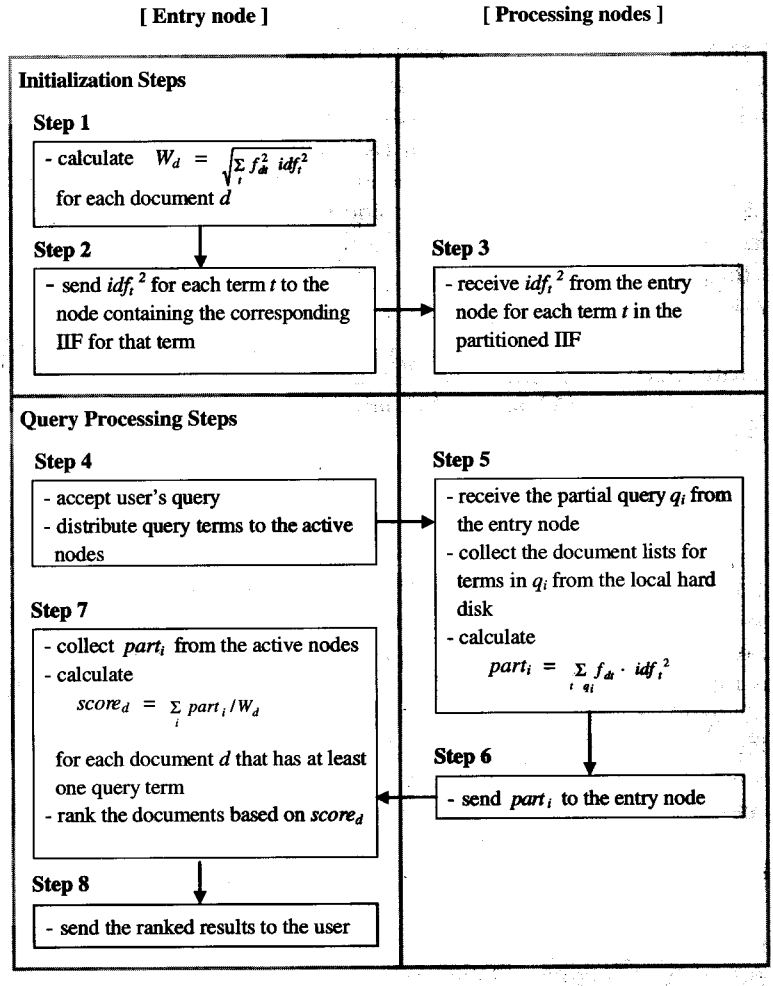

Figure 4. Parallel IR system algorithm.

In step 5, each active node i receives the partial query q_i from the entry node, collects the document lists for terms in q_i from the local hard disk, and calculates

$$\text{part}_i = \sum_{t \in q_i} f_{dt} \times idf_t^2$$

In step 6, each active node i sends $part_i$ to the entry node. In step 7, the entry node collects part_i from the active nodes, calculates

$$\text{score}_d = \sum_i \text{part}_i / W_d$$

for each document d that has at least one query term, and ranks the documents based on score_d. In the final step, the entry node sends the sorted document list back to the user as an IR result.

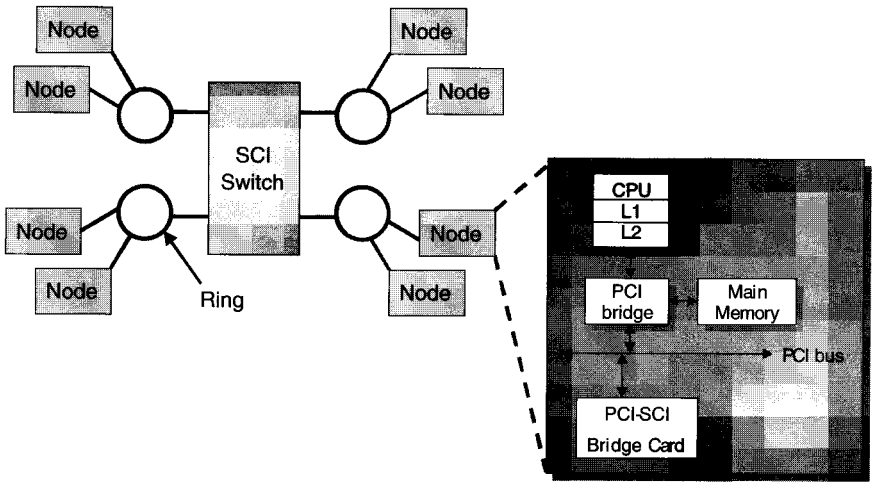

Figure 5. SCI-based 8 node PC cluster system.

3.4. Experimental PC cluster system

In this research, an 8-node SCI-based PC cluster system is constructed as shown in Figure 5. Each node is a 350 MHz Pentium II PC with 128 Mbyte main memory and 4.3 Gbyte SCSI hard disk, and operated by Linux kernel 2.0.36. In the cluster, any PC node can be configured as an entry node. As shown in the figure, each PC node is connected to the SCI network through the Dolphin interconnect solution (DIS)'s PCI–SCI bridge card. There are four rings in the network, and two nodes in each ring. The rings are interconnected by the DIS's 4 × 4 SCI switch. For DSM programming, the DIS's software infrastructure for SCI (SISCI) API [6] is used. With this configuration, the maximum point-to-point bulk transfer rate obtained is 80 Mbyte/s approximately.

For the purpose of comparison, an 8-node Fast Ethernet-based PC cluster system is also constructed as shown in Figure 6. Each PC node has the same configuration as the SCI network's node except that a PCI Fast Ethernet Adapter is used for networking. A switching hub is used to interconnect PC nodes in the cluster. For message-passing programming, MPICH 1.1.1 [7] is used. In this case, the maximum point-to-point bulk transfer rate obtained is 10 Mbyte/s approximately.

3.5. SCI-based DSM programming

The SCI interconnect mechanism supports DSM programming. By using SISCI, a node in the SCI-based PC cluster can establish a mapping between its local memory address space and a remote node's memory address space. Once the mapping is established, the local node can access the remote node's memory directly. In DSM programming, the communication between PC nodes in the cluster is done using remote read and remote write transactions instead of message-passing.

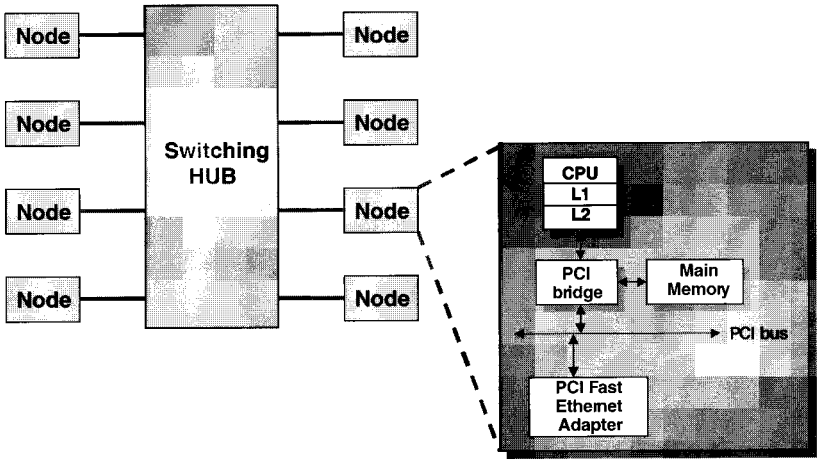

Figure 6. Fast Ethernet-based 8-node PC cluster system.

These remote read/write transactions are implemented using the remote read/write primitives provided by SISCI 1.9.0. For the efficiency of data transfer, it is important not to make unnecessary copies in the transmission path when remote read/write transactions are executed. In this research, several functions are developed on top of SISCI 1.9.0 to make a zero-copy communication mechanism, as explained in Figure 7. The figure shows a remote write transaction from Node B to Node A. The detailed procedure can be explained as follows:

1. The target memory is allocated in the user address space of Node A.
2. The connection between Node B and Node A is established.
3. Data is directly transferred from Node B to Node A without making extra copies.

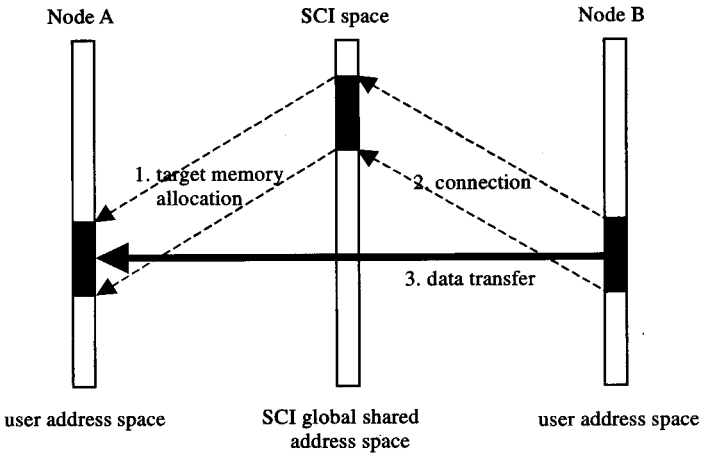

Figure 7. Zero-copy communication mechanism.

When the IR program is actually coded, most of the remote memory transactions are implemented using the remote write function. This is because the remote write function performs about 10 times faster than the remote read function in the current implementation of the DIS's PCI–SCI bridge card.

4. Performance of PC cluster-based IR system

4.1. Performance comparison between SCI-based system and Fast Ethernet-based system

In this experiment, average query processing times are measured for the 8-node SCI-based system, the 8-node Fast Ethernet (FE)-based system, and a single node system. The IIF is constructed from 100,000 documents collected from articles in a newspaper. A user's query consists of 24 terms. Each query is made to contain a rather large number of terms because the queries modified by relevance feedback usually have that many terms. The IIF is randomly declustered to be stored on each processing node's local disk.

As shown in Table 1, the disk access and IR operation times are reduced for both the SCI-based system and the FE-based system in comparison with the single node system. However the SCI-based system has less communication overhead, and performs better than the FE-based system. The performance of the system improves with further optimizations presented in the following subsections.

4.2. Effect of declustering IIF

The greedy declustering method is compared with the random method on a test set consisting of 500 queries each containing 24 terms. To generate the test queries, we randomly sampled 500 documents from a document collection containing 500,000 newspaper articles. From each document, the most important 24 terms are selected to make a query. The importance of a term in a document is judged by the value $f \times idf$, where f is the term's frequency in the document and idf is the inverse document frequency, which were defined previously in Section 3.3. Therefore, a term in a document is considered important if its frequency in that document is high enough but at the same time it does not appear in too many other documents.

Table 2 shows the experimental results comparing the random clustering and the greedy declustering methods using those 500 queries on our 500,000 document col-

Table 1. Query processing times of 8-node SCI-based system and 8-node FE-based system (unit: s)

	SCI-based system	FE-based system	Single-node system
Send query term	0.001	0.021	0
Receive document list	0.035	0.175	0
Disk access	0.097	0.097	0.204
IR operation	0.143	0.143	1.025
Total	0.276	0.436	1.229

Table 2. Comparison of random declustering and greedy declustering (unit: s)

	Random declustering	Greedy declustering	Enhancement ratio (%)
Average query processing time	1.174	1.130	3.7
Average disk access and local IR operation time	0.794	0.749	5.7

lection. The greedy method shows about 3.7% overall enhancement when compared with the random method. Since the two methods show differences only during the disk access and the local IR operations performed at the processing nodes, the time spent for those operations is measured separately. In this measurement, the greedy method shows about 5.7% enhancement, which is more significant than the overall enhancement.

4.3. Performance with various-sized IIF

In this subsection, the performance of the SCI-based parallel IR system is analyzed with the number of documents increased up to 500,000. These documents are collected from a daily newspaper, and 500,000 documents amount to the collection of the daily newspaper articles for seven years. The size of IIF proportionally increases as the number of documents increases. For example, the size of IIF is 300 Mbytes for 100,000 documents, and 1.5 Gbytes for 500,000 documents. The 8-node PC cluster and the greedy declustering method are used for the experiment.

The experimental result is presented in Figure 8. It takes 0.265 s to process a single query with the 100,000 document IIF, while it takes 0.477 s with the 200,000 document IIF and 1.130 s with 500,000 document IIF. As the IIF size increases, the document list for each query term becomes longer, and the time spent for IR operations considerably increases. As a result, the IR operation eventually takes more time than the disk access, and becomes the major source of bottleneck.

4.4. Performance with various number of processing nodes

Figure 9 shows the speed-up of the parallel IR system. The 500,000 document IIF and the greedy declustering method are used for the experiment. The maximum speed-up obtained from the 8-node system when compared with the single node system is 5.0. As shown in the figure, the speed-up of the parallel IR system is saturated rapidly from the 4-node system. As the number of the processing nodes in the system increases, the disk access and IR operation times are reduced because the average number of query terms assigned to each active node decreases. However, the communication time slightly increases as the number of document lists transmitted to the entry node increases. The problem may be alleviated by applying the following idea. Instead of sending all the document lists to the entry nodes, intermediate nodes can be utilized to merge the document lists in advance as shown in

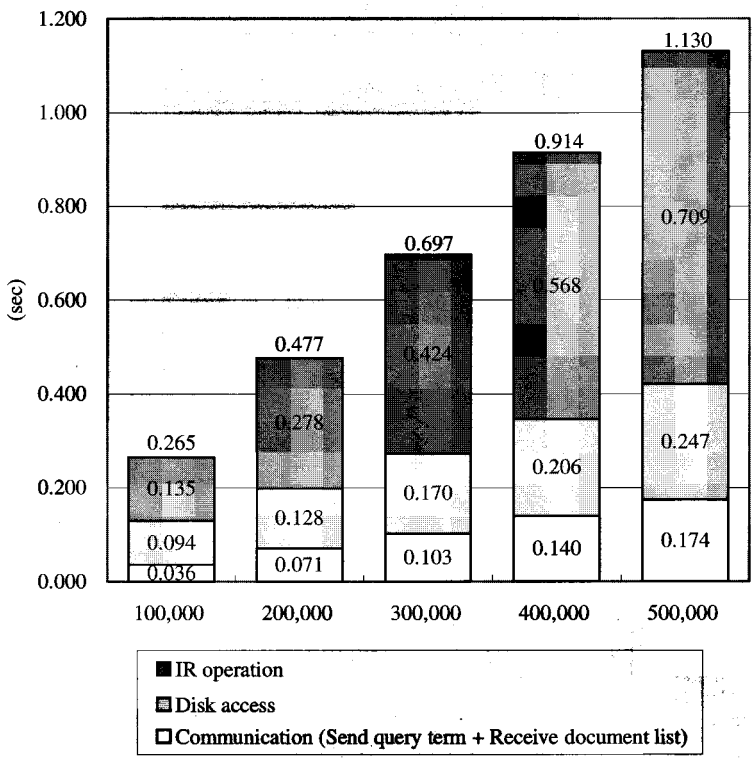

Figure 8. IIF size vs. query processing time.

Figure 10. Thus the entry node finally handles only two document lists. This will help in reducing the communication time. Experiments need to be performed to verify the above idea.

5. Conclusions

In this article, as a cost-effective solution for fast IR service, an SCI-based PC cluster system is proposed. In the parallel IR system developed on the PC cluster, the IIF is partitioned into pieces using a greedy declustering algorithm and distributed to the cluster nodes to be stored on each node's hard disk. For each incoming user's query with multiple terms, terms are sent to the corresponding nodes which contain the relevant pieces of IIF to be evaluated in parallel. The IR system is developed using a DSM programming technique based on SCI. According to the experiments, the IR system outperforms an MPI-based IR system using Fast Ethernet as an interconnect. Speed-up of 5.0 was obtained with the 8-node cluster in processing each query on a 500,000-document IIF.

Currently, the parallel IR system has a single entry node. In the future research, a PC cluster based IR system with multiple entry nodes will be developed. Each processing node in the cluster system can act as an entry node to process multiple

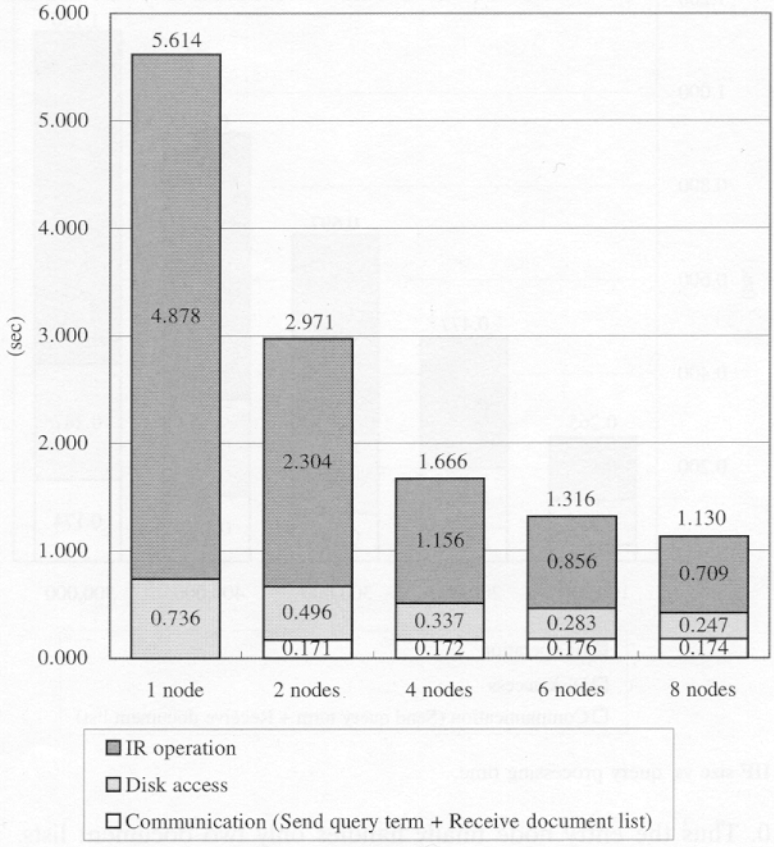

Figure 9. Number of processing nodes vs. query processing time.

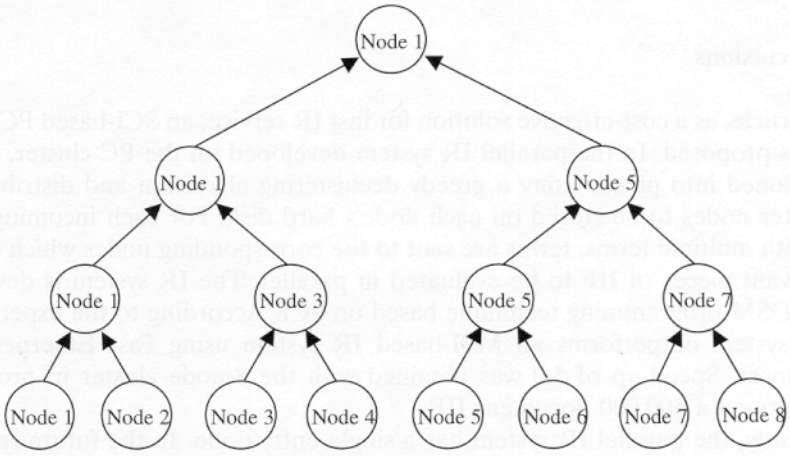

Figure 10. Merging document lists in intermediate nodes.

users's queries simultaneously. This will help in improving both the IR system's utilization and throughput. With more research effort, we hope this model to be evolved as a practical solution for low-cost high-performance IR service on the Internet.

References

1. A. Basu, V. Buch, W. Vogels, and T. von Eicken. U-Net: A user-level network interface for parallel and distributed computing. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, pp. 40–53. Copper Mountain, Colorado, 3–6 December 1995.
2. N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su. Myrinet: a gigabit-per-second local area network. *IEEE Micro*, 15(1):29–36, 1995.
3. R. Clark. SCI interconnect chipset and adapter: building large scale enterprise servers with Pentium Pro SHV nodes. White Paper. Data General Corporation, 1999. Available at http://www.dg.com/about/html/sci_interconnect_chipset_and_a.html.
4. J. K. Cringean, R. England, G. A. Manson, and P. Willett. Network design for the implementation of text searching using a multicomputer. *Information Processing and Management*, 27(4):265–283, 1991.
5. T. von Eicken and D. Culler et al. Active messages: a mechanism for integrated communication and computation. Report No. UCB/CSD 92/# 675. Computer Science Division, University of California, Berkeley, CA 94720, March 1992.
6. F. Giacomini, T. Amundsen, A. Bogaerts, R. Hauser, B. D. Johnsen, H. Kohmann, R. Nordstrøm, and P. Werner. Esprit Project 23174 – Software Infrastructure for SCI (SISCI), Version 2.1.1. White Paper. Dolphin Interconnect Solutions, 1999. Available at http://-www.dolphinics.no/customer/software/documentation/SISCI_API-2_1_1.pdf.
7. W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, 1996.
8. IBM. The IBM NUMA-Q enterprise server architecture. White Paper. IBM, 2000. Available at http://www.sequent.com/whitepapers/numa_arch.html.
9. Myricom, Inc. The GM API. White Paper. Myricom, Inc., 1997. Available at http://www.myri.com/GM/doc/gm_toc.html.
10. S. Pakin, V. Karamcheti, and A. A. Chien. Fast messages (FM): efficient, portable communication for workstation clusters and massively-parallel processors. *IEEE Concurrency*, 5(2):60–73, 1997.
11. S. H. Park and H. C. Kwon. An Improved Relevance Feedback for Korean Information Retrieval System. In *Proceedings of the 16th IASTED International Conference Applied Informatics*, pp. 65–68. IASTED/ACTA Press. Garmisch-Partenkirchen, Germany, 23–25 February, 1998.
12. C. J. van Rijsbergen. *Information Retrieval*, 2nd ed., Butterworths, London, 1979.
13. G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *American Society for Information Science*, 41(4):288–297, 1990.
14. R. Sharma. A generic machine for parallel information retrieval. *Information Processing and Management*, 25(3):223–235, 1989.
15. C. Stanfill and R. Thau. Information retrieval on the connection machine: 1 to 8192 Gigabytes. *Information processing and Management*, 27(4):285–310, 1991.