Massively Parallel Information Retrieval for Wide Area Information Servers

Craig Stanfill
Thinking Machines Corporation
245 First Street
Cambridge MA 02154

***Abstract*. Today most text is stored in electronic form at one point or another in its life. It is then perhaps surprising that computers have made relatively little impact on how it is stored and retrieved, but such is the case. The Wide Area Information Server project (WAIS) seeks to alleviate this problem by making text databases widely accessible via a common user interface and wide area networking. The ultimate success of the effort depends, in part, on the availability of high-performance text search engines. Such engines can be implemented on massively parallel machines, using either signature file or inverted file database structures. Inverted files are particularly appealing, and permit databases as large as 8192 Gigabytes to be searched in under 15 seconds, using currently available hardware**

I. INTRODUCTION

A large proportion of the American workforce is constantly producing or consuming text in various forms — internal memoranda, reports, correspondence, technical papers, electronic mail, etc. — which originates in electronic form. And so it is perhaps surprising that computers have made relatively little impact on how such information is filed and retrieved. Just as they did a hundred years ago, people store their text in hierarchical files, ask co-workers if they know where some data might be found, and run to the librarian for help in accessing external data. Two recent developments — parallel information retrieval systems and wide-area information servers — have the potential to vastly improve on this situation, making essentially all the information in the world instantly available from a desktop computer.

II. CURRENT PRACTICE

The way in which electronic text is stored and organized (or, in most cases, not organized) makes its effective use for purposes other than generating hard-copy very difficult.

Personal data is generally stored in a hierarchical file system, but otherwise unorganized. An individual, looking for an old report or an old piece of electronic mail may have to look through a large number of directories on several machines before the wanted data is found. Information gets lost, and a considerable amount of time is wasted trying to guess where information might be located.

Most companies do not have an over-all structure for managing electronic text. Some electronic mail will be archived. Individuals may maintain their own on-line copies of correspondence they have produced. Engineering projects will have their own design files. Reports prepared by consultants may be filed by the person who requested the report. Files containing presentation graphics will reside on various desktop computers. If an individual wants to find a piece of text which is stored on-line, he must figure out who to ask. On-line systems are rarely if ever available.

Off-site electronic text is better organized but, in practice, not much more accessible. Some databases are elaborately indexed using a tightly controlled vocabulary. Other databases are unindexed, relying on full-text searches. Every database vendor has its own indexing method and its own query syntax. In the majority of cases, the query syntax consists of a mixture of boolean and proximity operators. At the very least, these query interfaces are difficult for non-experts to use effectively; one is always walking a thin line between getting no documents and getting too many documents.These databases are generally available only as dial-up services, and rarely support anything except for teletype-style interfaces. In addition, they have traditionally charged their users by the connect-minute, a practice which strongly discourages the use of these systems for browsing and interactive exploration. As a result of all these factors, end-users are discouraged from signing onto a database and looking for information themselves; if they need on-line text search capability, they will be referred to a librarian. There is a good chance that the librarian will find the necessary information, but this may take several days, and the expense of the process discourages users going to an external information source for most inquiries.

III. WIDE AREA INFORMATION SERVERS

Various technologies exist which can solve these problems. Individual and corporate documents can be made more accessible by storing them in full text databases. Remote access can be simplified by use of wide area networks. The varying query languages of different external databases can be forced into conformity by the establishment of standards. The difficulties associated with using boolean query systems can be avoided by using several non-boolean search methods developed in the information retrieval community. The ease of using the systems can be improved by the development of graphic interfaces. The Wide Area Information Server (WAIS) project [1] seeks to combine these technologies in order to make a wide

Received 29 July 1991.

variety of data available in a desktop environment. This section will describe the major components of the current system (Fig. 1).

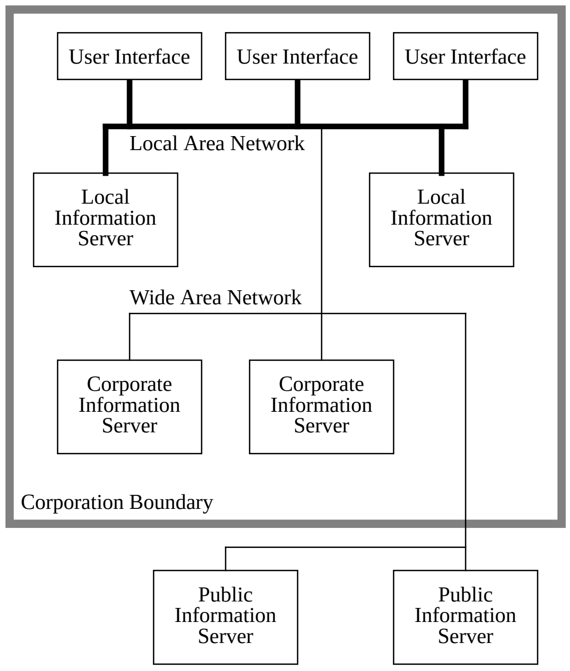

FIG. 1: WAIS ARCHITECTURE

A. Search Methods

The search method used in WAIS is based on several information retrieval techniques developed in the 1970's [2]. Queries consist of short natural language phrases, such as "Corazon Aquino and the Philippine Election." Each phrase is broken into primitive components such as "Corazon Aquino," and "Philippine Election," and each component is assigned a numerical weight with rare (i.e. more specific) terms assigned higher value. Every document in the database is then scored against this query by summing the weights of the query terms it contains. The documents are then ranked from highest to lowest, and the best matches presented to the user. All this is hidden from the user, who knows only that he has asked a question and gotten back some documents which are likely to be relevant. The user may then browse the full text of these documents.

In order to focus the search, the user may indicate to the system that a given document is "relevant" to his search (this is called "relevance feedback" [3]. The system will then examine the full text of that document, break it into words, and use the most important words in the document to construct a new query. This query will be run against the database, and the result presented to the user. In most cases, the results of this search will contain a high proportion of documents on the same topic as the feedback document. Once again, this process is hidden from the user, who only knows that he has indicated to the system that a particular document was interesting, and that the system has presented him with more documents on the same topic.

This sort of search system has several advantages. First and foremost, because queries are expressed in natural language, there is no need for the user to learn an artificial query language. Second, the document ranking scheme, in which documents are presented to the user in order of likely interest, eliminates some of the brittleness of boolean systems; the user will always find something, without being overwhelmed by the thousands of documents which may result from a poorly crafted boolean query. Third, the use of relevance feedback gives a convenient method for focusing a search, enabling the user to be confident that he has found a majority of the documents on a given topic. Finally, the natural language/relevance feedback interface style provides the designer of a query system with a great deal of freedom in the choice of specific information retrieval algorithms. As better methods are developed, they can be incorporated into the system without forcing the users to learn a new query language.

B. Graphic Interfaces

The second major component of the WAIS system is a graphic interface. This interface needs to support natural language querying, relevance feedback, document display, and database selection. Interaction is defined in terms of several kinds of objects (represented by icons) plus actions such as dragging and opening. For example, to initiate a search the user will create a "question", which will have slots for data sources, relevant documents, and a natural language question. Data sources and relevant documents are added to the question by dragging their icons into the appropriate slots. The natural language query is created by typing into the query slot. When the user clicks on a "search" button, the interface will contact all selected data sources, send them a query, and wait for a response. When the response comes, the headlines of the relevant documents will appear in another frame, along with icons standing for the documents. These icons can be opened to examine the full text, or they can be dragged back into the "relevant documents" slot to feed them back into the query. At any time, the user can start a new query (without abandoning the old one) by creating a second question object.

C. Networks

The third major component of the WAIS system is a wide area network plus a protocol enabling clients and servers to exchange information. If a reliable network already exists (e.g. NSF-net), then little needs to be done except for assigning a network address to the server. If a good network connection is not available, then the situation becomes much more difficult. A 9600 Baud telephone connection is generally sufficient if the transmission protocol includes error detection and correction. Once a server has been set up on the network, it is necessary for clients to be able to contact it. This has been done by establishing a database called the directory of servers, which is a WAIS database having descriptions and network addresses for all publicly available WAIS servers, and which all WAIS servers know how to contact. The final component of the WAIS network architecture is the protocol between the servers and the clients. The starting point for this was the Z 39.50 protocol, which was initially set up to allow the interchange of information between libraries. This protocol was inadequate in several ways (it did not support relevance feedback, it did not provide adequate support for the transmission of the full text of documents, etc.), but was fairly easy to extend.

IV. PARALLEL TEXT SERVERS

For WAIS to succeed, it must be possible to build full text retrieval systems which can deliver quick responses (1-2 seconds is ideal) when searching databases having anywhere from a few megabytes of text (e.g. a personal database) to hundreds, thousands, or even tens of thousands of Gigabytes. Some databases can be adequately served by serial machines (e.g. personal or departmental databases having a few megabytes to one gigabyte of data). Other databases — those which are either extremely large (e.g. a corporate text repository) or accessed very heavily (e.g. an external database service) — will require levels of performance which cannot be attained on serial machines. For these databases, parallel text servers are an appropriate technology.

A. Parallel Signature Files

Initial work on parallel text servers represented the database as an overlap encoded signature file [4][5]. In this representation, each document is broken into segments containing (for example) 30 words. A signature file is then created by applying the following procedure to each such segment in the database:

1. A region of memory is allocated and initialized to 0. This will become the signature of the text segment.
2. Several hash functions are applied to each word in the segment. The bits of the signature addressed by these functions are set to 1.
3. A group of consecutive signatures are assigned to each processor.
4. To probe a signature for the presence of a word, the same set of hash functions are applied to it and the corresponding bits of the signature are ANDed together. If the result is 1, then the word is assumed to be present in the segment, even though there is a small possibility that the 1 is a result of coincidence.

Signature files have a number of advantages, namely: 1) the probe operation is very fast, particularly on SIMD machines constructed from bit-serial processors (e.g. the Connection Machine models CM-1 and CM-2 [6], the Goodyear MPP [7] and the DAP [8]; and 2) the signature file is much smaller than the full text.The degree of compression depends on the acceptable error rate in the probe step, with 30% being possible for systems having error rates on the order of 1.2×10^{-6}. The disadvantages of this scheme are 1) that the system can only represent the presence or absence of a word; other information, such as its position and the number of times it occurred will necessarily be lost [9][10]; and 2) that there is no effective strategy for searching databases which are too large to fit in main memory [11].

Nevertheless, such systems are capable of delivering high performance for databases as large as 25 Gigabytes. Main memory systems of this sort are a good choice for heavily accessed databases where the high performance translates into a high throughput.

B. Parallel Inverted Files

For databases which are too large for available memory or for which the number of inquiries per second is not large enough to justify the expense of a large primary memory, secondary storage must be employed. As noted above, this rules out the use of the signature algorithm. The best solution in this case is to adapt the inverted file structure generally used in serial systems. In this sort of system, each document is indexed to produce a set of postings. Each posting contains a word, an identifier for the document it was contained in, and additional information (e.g. a numerical weight indicating how many times it occurred) which may be required by the search strategy being used. These postings are sorted by word, the words are stripped from the postings, and the results written to disk. At the same time, an index is built which permits the postings for a given word to be found on disk. The advantage of this file structure is that a relatively small amount of data needs to be accessed in answering a query. The primary disadvantage of this file structure is that it is fairly complex to build and maintain and, in particular, real-time modifications to the database are difficult to support.

As noted above, WAIS text servers need to evaluate queries consisting of words plus weights indicating the importance of those words. The basic computation is to 1) create a score accumulator (called a mailbox) for each document and zero it; 2) read the postings for each term in the query; and 3) increment the mailboxes for the documents referenced by these postings.

In the simplest parallel inverted file structure, the postings are distributed arbitrarily across the processors and, at query time, sent to the processor containing the appropriate mailbox [12]. In a more complex file structure, called a partitioned posting file, the postings are stored on disk in such a way that they will, when read into the machine's memory, be in the same processor as their mailbox [13]. The later file structure is somewhat faster than the former, as it involves no interprocessor communication, but building this file structure is more difficult.

Benchmarks reveal that, using the partitioned algorithm and an 8192 processor CM-2, it should be possible to evaluate a 10-term query against a 100 Gigabyte database in 1.8 seconds (compute only), vs. 45 seconds for a fast serial machine (a Sun 4/330).

The second major issue in parallel inverted file design is I/O. The Connection Machine supports a disk array called the

DataVault. Each DataVault can store up to 60 Gigabytes of data, which would be sufficient to hold the inverted file for a database as large as 180 Gigabytes. The DataVault has a transfer rate of 25 Megabytes per second. It may be used in two modes: striped mode and independent disk mode. In striped mode, all the disks in the DataVault operate synchronously, yielding a very high transfer rate. The disadvantage of this mode is that it has a latency of 200 milliseconds, so that no more than 5 I/O's per second may be supported. In independent disk mode, each disk may be given a different seek address. In this access mode, the DataVault's latency is 225 milliseconds per 32 I/O's.

According to one model of the retrieval process, an average term from an average query will occur 3000 times per Gigabyte of full text. Each term posting requires approximately four bytes of data. Evaluating a 10-term query against a 100 Gigabyte database will thus require the transfer of 12 Megabytes of data, which should take slightly under 0.5 seconds. A full discussion of the performance characteristics of parallel database servers is beyond the scope of this paper; the interested reader is referred to [13]. Table 1, taken from the referenced paper, summarizes the performance characteristics of various systems having between 1.5 and 8192 Gigabytes of data.

TABLE 1
PERFORMANCE OF CM-2 FOR VARIOUS DATABASE SIZES.
TIMES IN SECONDS, 10-TERM QUERIES.

Size	**Processors**	**DataVaults**	**Time**	**Storage Method**
1.5 GB	4K	0	0.055	Main Memory
3 GB	8K	0	0.055	Main Memory
6 GB	16K	0	0.055	Main Memory
12 GB	32K	0	0.055	Main Memory
24 GB	64K	0	0.055	Main Memory
64 GB	8K	1	1.7	Independent Disk
128 GB	8K	1	2.8	Independent Disk
256 GB	16K	2	3.6	Striped Disk
512 GB	32K	4	3.6	Striped Disk
1024 GB	64K	8	3.6	Striped Disk
2048 GB	64K	16	5.1	Striped Disk
4096 GB	64K	32	8.2	Striped Disk
8192 GB	64K	64	12.4	Striped Disk

V. CONCLUSIONS

This paper has argued 1) that current methods for accessing on-line text are inadequate; 2) that Wide Area Information Servers represent a possible solution to this problem; and 3) that parallel text servers can be built to support any size database likely to be needed in the next decade.

This process is, of course, in its infancy. At this point, it is probably most important that people use and experiment with Wide Area Information Servers. This is currently being done on a research basis at a variety of sites on the Internet. Those who wish to obtain the software described in this paper may do so by contacting Brewster Kahle via e-mail (brewster@-think.com).

REFERENCES

[1] Kahle, B. & Medlar, A. (1991). An information system for corporate users: Wide Area Information Servers. Technical report TMC-199. Cambridge MA: Thinking Machines Corporation.

[2] Salton, G. (Ed.). (1971). *The Smart System --- Experiments in automatic document processing*. Englewood Cliffs, NJ: Prentice-Hall.

[3] Rocchio, J.J., Jr. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), *The Smart System --- Experiments in automatic document processing*. Englewood Cliffs, NJ: Prentice-Hall.

[4] Stanfill, C. & Kahle, B. (1986). Parallel free-text search on the Connection Machine system. *Communications of the ACM, 29*(12), 1229-1239.

[5] Pogue, C. & Willet, P. (1987). Use of text signatures for document retrieval in a highly parallel environment. *Parallel Computing, 4,* 259-268.

[6] Hillis, D. (1985). *The Connection Machine*. Cambridge, MA: MIT Press.

[7] Batcher, K.E. (1980). Design of a massively parallel processor. *IEEE Transactions on Computing, C-29(9),* 836-840.

[8] Flanders, P.M., Hunt, D.J., Reddaway, S.F., & Parkinson, D. (1977). Efficient high speed computing with the distributed array processor. In D.JU. Kuck, D.H. Lawrie, & A.H. Sameh (Eds.), *High speed computing and algorithm organization* (pp. 113-127). New York: Academic Press.

[9] Croft, B. (1988). Implementing ranking strategies using text signatures. *ACM Transactions on Office Information Systems 6*(1), 42-62.

[10] Salton, G. & Buckley, C. (1987). Parallel text search methods. *Communications of the ACM, 31*(2), 202-215.

[11] Stone, H. (1987). Parallel querying of large databases: A case study. *Computer, 20*(10), 11-21.

[12] Stanfill, C., Thau, R., & Waltz, D. (1989, June). A parallel indexed algorithm for information retrieval. Paper presented at the International Conference on Research and Development in Information Retrieval. Cambridge, MA.

[13] Stanfill, C., & Thau, R. (1991). Information retrieval on the Connection Machine: 1 to 8192 gigabytes. *Information Processing and Management 27*(4), 285-310.