A Parallel Document Retrieval Server For The World Wide Web

D. Hawking *P. Bailey*
D. Campbell

Cooperative Research Centre For Advanced Computational Systems
Department of Computer Science
The Australian National University
Canberra ACT 0200 AUSTRALIA

dave@cs.anu.edu.au

Abstract

An architecture is proposed which enables the Parallel Document Retrieval Engine (PADRE), running on a single-user Fujitsu AP1000 multicomputer, to operate as an information server on the World Wide Web. The advantages and disadvantages of a distributed memory parallel machine for this purpose are discussed and the likely applicability to different types of parallel machine is considered. Ideas for a range of types of remote query-generation client are outlined and measurements of query processing speed are reported, shedding some light on potential load handling capacity of this parallel server.

KEYWORDS Text retrieval, information retrieval, parallel computing.

1 Introduction

The Parallel Document Retrieval Engine (PADRE) is implemented on the Fujitsu AP1000, a large-scale, single-user, parallel, distributed memory system. PADRE's operation is based on pattern-matching and supports literal strings, regular expressions, and numeric ranges as patterns. Searches may be confined to particular user-defined components of documents. Results of searches (*match sets*) may be used as operands to set and proximity operators. A PADRE user may select from a rich set of alternative functions for computing the relevance of documents.

It has been demonstrated [4, 7] that PADRE is capable of comfortably handling quantities of data the size of the TREC [2] collection and of achieving good, if not yet spectacular, precision-recall results.

The approach taken to parallelism is that of dividing the document collection into sub-collections which are searched independently by separate processing nodes. Because inter-node communication during searching is required only for computation of global collection statistics and for merging document rankings, the problem can be described as *inherently parallel*, provided that the collection can be divided in such a way that the largest sub-collection is only slightly larger than the average.

Past work on the development of PADRE is described in [10, 3, 4, 5]. It has been shown [4] that search times show near-linear speedup with increasing numbers of processing nodes.

The present paper describes extension of PADRE to operate as a World-Wide Web (WWW) server and re-evaluates the benefits of parallel architectures in this context.

2 Advantages And Disadvantages Of Parallel Machines For Document Retrieval

Four potential benefits to be gained from using a large-scale parallel distributed memory machine for document retrieval are listed below. Unfortunately, it is difficult to achieve all benefits simultaneously. In other words: flexibility is only available at the expense of speed; the ability to handle very large collections is at the expense of lightning response. PADRE provides three different retrieval methods which target one or more of the benefits.

1. Large memory capacity, memory bandwidth and processing power allow more powerful search terms such as regular expressions to be processed quickly over TREC-scale collections. (Full Text Scanning Method - FTS)

2. The same attributes permit extremely fast construction of inverted files and similarly fast searches using them. (Memory Resident Index Method - MRI)

3. Parallel-disk-resident inverted file indexes and dictionaries extends data handling capacity to

Proceedings of the First Australian Document Computing Conference, Melbourne, Australia, March 20–21 1996.

the terabyte level, for a commercially available machine. (Super Dictionary Method - SD)

4. Rapid response to changing collections. (All Methods)

These benefits may be important to a network service provider because either:

1. More valuable services may be provided such as:
   (a) Access to larger collections,
   (b) More powerful queries, or
   (c) Almost instantaneous response to collection updates.
2. In an environment characterised by simple queries and TREC-scale or smaller text bases, A greater number of simultaneous users may be supported.

The only major disadvantage of a large parallel machine is its cost. Generally speaking, use of such a machine for document retrieval is currently only economic in a multi-user time-sharing environment with high-value data.

Another widely perceived disadvantage of parallel systems is increased difficulty of programming in a message passing environment. However, the parallelisation model employed in PADRE is quite simple, and does not require great sophistication in the message passing. Some complexity in the original code was subsequently obviated by a handful of wrappers implementing a slightly higher level of communication abstraction. The development of a POSIX-compliant parallel file system with a simple parallel file model [8] has provided high performance without increasing the complexity of I/O programming. For recent PADRE developments, difficulty in adding new features has been totally dominated by the serial rather than the parallel algorithms.

2.1 Applicability Of Other Styles of Parallel Architecture

Porting of PADRE to other platforms by conversion of AP1000 message passing calls to those of a portable system such as MPI, is expected to be quite straightforward. Accordingly, ports of PADRE to a wide variety of architectures could be considered with the following reservations.

2.1.1 Shared Memory

Shared memory systems generally rely on caching to disguise a low total memory bandwidth. Consequently, performance of the full-text scanning method is not likely to scale with number of processors.

2.1.2 Different Network Topologies

As previously mentioned, PADRE requires little inter-node communication during searches. Significant deterioration in latency and bandwidth could be tolerated, compared with the AP1000, without noticeable deterioration in query processing times.

2.1.3 I/O Systems

Aggregate I/O bandwidth and rate of performing I/O operations are important to PADRE, particularly in the SD Method. A highly parallel computing system without comparable I/O parallelism is unlikely to provide good performance, except when data is memory resident.

2.1.4 Networks of Workstations

A collection of dedicated, homogeneous workstations connected by a network of reasonable aggregate bandwidth and managed as a cluster is essentially indistinguishable from the Fujitsu AP1000 architecture as far as PADRE is concerned. However, most real networks of workstations are not managed as a single entity and are not homogeneous. They are typically subject to irregular and unpredictable contention from other user jobs and fall short of the standards of overall reliability we have experienced from the AP1000 (two hardware failures in four years on a 128-node system with 32 disks). These factors may have considerable impact on quality of service:

1. The probability of downtime of individual workstations due to one-by-one operating system upgrades or hardware/network failures encourages use of time-out protocols which would void any guarantee that *all* documents in the distributed collection had been scanned during processing of a query.
2. Heterogeneity of configuration makes static load balancing more difficult. Variations in background load caused by other applications make static balancing ineffective.
3. Network errors or extreme contention on one or more nodes may cause unacceptable delays in responding to queries.

3 The PADRE As Web Server

3.1 Server Architecture

The key to the PADRE Web-server architecture is a daemon called MUFTI) (Multi User Free Text Intermediary) running on the AP1000 front-end, as shown in figure 1. MUFTI listens on a TCP/IP port for one or more connections from remote (or local) user-interface clients. Communication over these

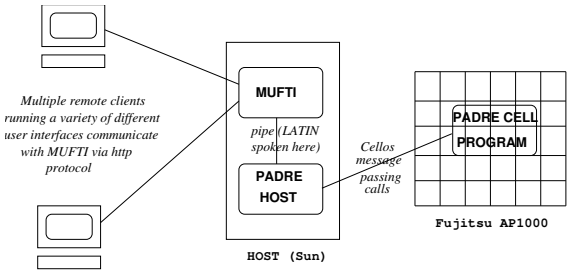

Figure 1: Components of PADRE/MUFTI multi-user architecture.

sockets conforms to the *http* (HyperText Transfer) protocol [9] and conveys PADRE queries in one direction and search results in the other. MUFTI is responsible for starting up PADRE if it is not already running, loading the appropriate text base if it is not already loaded and for terminating PADRE if no queries have been received for a specified time-out period.

Because the PADRE search engine can process only a single request at a time, there is no need for MUFTI to be multi-threaded either. MUFTI will accept connections from multiple clients but causes them to wait until the currently active request is processed.

When a web browser connects to the MUFTI URL (Universal Resource Locator), an HTML page is returned which provides some explanatory information and a form including a scrolling text window into which the search query can be typed. When the query is complete, the browser uses its own standard methods to convert the text into the form of an expanded URL which is transmitted to the server. The server performs minor editing to unpack the PADRE query, submits it to PADRE itself and then filters the output into a HTML generated list of "relevant" documents. The searcher may select one of these for retrieval and display.

This architecture is not tied to any particular web browser and allows for the possibility of multiple styles of query generation client to operate concurrently. Some possibilities are described below.

Document retrieval queries generated by retrieval experts (or by novices with access to query term expansion technology such as thesauri and relevance feedback) are typically quite complex. They are also likely to be subject to a series of interactive modifications as the searcher homes in on the documents they seek. These two characteristics, coupled with the requirement to maximise PADRE throughput, suggest that:

1. Query optimisations performed by the query generation client would be highly desirable. These may include:
   (a) Elimination of redundant operations,
   (b) Naming intermediate result sets for later reference,
   (c) Re-casting or re-ordering the query to restrict the set of documents which must be considered in the later part of the query, and
   (d) Aborting query processing when the number of potentially interesting documents has stabilised at a manageable level.
2. The addition of session management and context saving capabilities to PADRE would be highly desirable. In this model, each separate client connection would be allocated a unique session-id used in all requests. Results of each query would be tagged with the relevant session-id, saved to disk and restored as necessary to reduce the cost of subsequent query modifications.

It may eventually be necessary to introduce micro-scheduling at the sub-query level to improve fairness between multiple simultaneous searchers, even though overall throughput would decrease considerably due to context-switching overheads.

In the initial implementation, there is no microscheduling and no session management.

The basic quanta of work processed by PADRE/MUFTI are of two types:

1. A query which is processed in its entirety and which cannot rely on PADRE to remember any previous state. After processing, results in the form of a list of relevant documents, are sent back to the client and PADRE is reset.
2. A request to return a uniquely specified document (presumably one of the documents listed in response to a query).

3.2 Client-Side Interfaces

3.2.1 PADRE Character Based Interface

The native PADRE query language LATIN (Local ASCII Text INterface) is not well suited to the needs of human researchers. As a person-machine interface, it is verbose, syntactically exacting and allows many gratuitous opportunities for human error. It will be retained, however, because it is convenient to use when the query generation process is machine assisted. Furthermore, direct, batch-style input is very useful for debugging PADRE modifications and for qualifying new versions.

The more user-friendly query interfaces described below must generate queries in LATIN and pass them to PADRE via MUFTI.

In the future, LATIN is likely to be extended to incorporate predicates and conditionals which may be used to modify the form of a query based

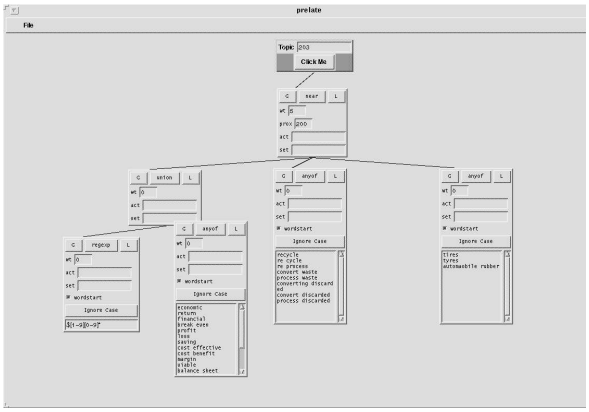

Figure 2: Screen dump of PRELATE representation of a PADRE query relating to the economic impact of recycling tyres.

on initial results, either to increase efficiency or improve precision or recall.

At the time of writing, the only access to the multi-user capabilities of MUFTI is by typing LATIN in the search query field of an interactive form.

3.2.2 PRELATE - A Graphical User Interface

In general, PADRE queries can be regarded as tree-structures in which literals, regular expressions, numeric ranges or lists of alterantive literals form the leaves and set operators or proximity operators form non-terminal nodes. Any node may be tagged with a relevance weighting and an action to be taken after completing the operation.

A prototype of a Tcl/Tk interface (PRELATE - Padre REtrieval LAnguage Topic Editor) capable of generating PADRE queries has recently been developed. It presently operates in off-line mode, but it is also planned to configure it as an *applet* under a TCL/Tk Web browser such as SurfIt! [1]. A sample screen-dump from PRELATE is shown in figure 2.

3.2.3 Automatic Query Generation From Prose Specification

In some institutions, librarians or research assistants take prose descriptions of research topics given to them by researchers and turn them into queries in the language of the information retrieval system in use. For researchers expert in their subject matter but not in the intricacies of computerised information sources or query languages, this style of relevance definition is very convenient and effective.

An interface to PADRE is currently being developed which allows for the automatic dynamic construction and execution of a query from a natural language prose specification.

3.2.4 WAIS/Z39.50

At present these standards are not supported. Provision of such support would not be expected to pose a major problem, but changes would be needed to both PADRE and MUFTI to allow service of requests from WAIS clients.

4 Load Handling Capacity Of A Parallel Web Server

It has not yet been possible to collect experimental data on how many interactive clients can be supported in everyday use by a given PADRE. Indeed, we are not aware of any standard benchmark of multi-user document retrieval performance, though in our Internet-dominated world, such benchmarks would be very useful.

The experiments described below consider only the query processing performance of PADRE itself and do not take into account possible MUFTI or client-server communication bottlenecks. They give a current upper bound on performance of the system.

4.1 Search Engine Performance

Retrieval performance which is adequate when dedicated to a single user becomes unacceptable when shared among a large number. Consequently, speed of the basic search engine is critical to a multi-user PADRE.

Some measurements have been taken recently of the speed of PADRE query processing on an AP1000 configuration comprising 128 25 MHz SPARC processors, each with 16 Mbytes of RAM. Of these processors, 32 have a directly connected SCSI disk. Query processing speeds for the three different methods mentioned above (FTS, MRI and SD) were measured for comparison purposes.

The basic test query (3) required the location of 3 groups of alternative literal patterns anchored at word starts (index points), the computation of a 3-way proximity relation (`znear`) between the groups, the assessment of relevance of all documents based on matches resulting from these operations and the creation of a ranked list of the 20 most relevant documents. There are a total of 20 literal terms.

As can be seen, the basic test query is much simpler than those arising from relevance feedback and thesaurus/fact-book term expansions but is more complex than those typically generated by users of library catalogues or abstract services.

A second test query designed to investigate the reduction in query processing speed caused by use of more sophisticated query terms, namely regular expressions, is shown in figure 4. In it, the set of terms used to locate references to *economic impact*

```
{proximity 1000}
{wsmode start}
{casesensitive 0}
{weight 0}
anyof "economic |economical |profit
|profitable |profits |dollars "
anyof "recycle |recycling |recycles
|reprocess |reprocesses |reprocessing
|conversion |converting |converts "
anyof "glass |paper |plastic |aluminum
|cardboard "
{weight 5}
znear 3
top 20
```

Figure 3: Query 1: The basic test query. Each `anyof` produces a set of matchpoints which becomes an operand to the `znear 3` operator.

```
{proximity 1000}
{wsmode start}
{casesensitive 0}
{weight 0}
anyof "economic |economical |profit
|profitable |profits |dollars "
regexp "\$[1-9][0-9]*"
union 2
anyof "recycle |recycling |recycles
|reprocess |reprocesses |reprocessing
|conversion |converting |converts "
anyof "glass |paper |plastic |aluminum
|cardboard "
{weight 5}
znear 3
top 20
```

Figure 4: Query 2: The basic query augmented by the use of a regular expression. The `union 2` operator replaces the match sets resulting from the `regexp` and the first `anyof` with their union.

are augmented by a regular expression matching dollar amounts.

The test collection comprised approximately 750MB of Associated Press documents from the TREC collection, a total of 242,918 documents.

Results are shown in table 1.

As can be seen, high query processing rates are achievable with the MRI method. Considerable speed must be sacrificed to gain either very large data handling capacity (exemplified by SD method with multiple dictionaries) or powerful query terms FTS method with regexp. Setup times are very small for the FTS method. For the indexed methods, they are very small relative to uni-processor systems using similar structures.

Method	**Setup Time (sec.)**	**Queries/Min.**
FTS	15	15
MRI	464	130
SD (1 dict.)	455	22
SD (3 dict.s)	640	13
FTS/Regex	15	3.6

Table 1: Comparison of the query-processing speed of the different search methods of PADRE, over approximately 750 MB of data on a 128-node Fujitsu AP1000. Speeds are derived from elapsed times observable by the user. Setup times are also elapsed times and include construction of disk-resident structures and loading of data and/or structures from the parallel disks.

Reported setup times include the full cost of building all data structures from scratch. Setup costs for a PADRE run using an existing index would be an order of magnitude less. Setup cost arising from a change to one of the three sub collections in the SD(3) case would be 243 seconds compared with 640 for the full collection. Both data structure building and query processing costs for the superdictionary methods have the potential to fall significantly in response to optimisation of algorithms.

It seems very likely that the number of clients which can be simultaneously handled by PADRE will depend upon the following factors:

1. Whether or not PADRE uses full-text scanning, memory-resident index or super dictionary methods.

2. Using full-text scanning method, the size of the collection.

3. Using super dictionary method, the number of subcollections containing occurrences of the search terms.

4. The complexity distribution for queries. If queries with many terms or queries containing computationally expensive terms such as regular expressions are encountered more than infrequently, long and variable response times may be expected and micro-scheduling would be indicated.

5. The pattern of interaction by researchers. How does the time taken to construct a query and digest its results compare with the PADRE time to process it? How frequently are queries modified in minor ways (taking very little human time) and resubmitted? It is expected that interaction patterns will vary considerably with the type of query interface.

Particularly in a fee-for-service operation, it may be necessary to implement timeouts on searches or to restrict the complexity of queries available to ordinary users in order to ensure reasonable response. Paying customers are unlikely to be impressed by long delays occasioned by someone searching ten gigabytes of documents for anagrams of Jim Morrison!

5 Conclusions

Substantial progress has been made toward rendering PADRE capable of operating as a multi-user server on the Internet or other TCP/IP network. Prototype implementations of various key elements of a successful multi-user system exist and their performance is encouraging. A range of user interface styles seems likely to improve the accessibility of PADRE and to improve the productivity of researchers using it. Further measurements are needed to fully characterise multi-user performance.

In the future we hope to further benchmark (and improve) multi-user performance of PADRE/MUFTI and to complete the implementation of effective and industry-standard query interfaces.

Acknowledgements

The support of the ACSys Co-operative Research Centre and the ANU-Fujitsu CAP project is gratefully acknowledged. We thank Paul Thistlewaite for his input and encouragement and also express our gratitude to NIST in the USA and various copyright holders for access to the TREC data collection.

References

[1] S. Ball, S. Uhler and J. Levy. *SurfIt home page.* `http://pastime.anu.edu.au/SurfIt`.

[2] D.K. Harman. *Proceedings Of The Third Text REtrieval Conference (TREC-3)*, US Department of Commerce, NIST Special Publication 500-225, April 1995.

[3] D.A. Hawking. High Speed Search of Large Text Bases On the Fujitsu Cellular Array Processor. In *Proceedings of the Fourth Australian Supercomputing Conference* pages 83-90, Gold Coast, Australia, December 1991.

[4] D.A. Hawking. PADRE — A Parallel Document Retrieval Engine. In *Proceedings of the Third Fujitsu Parallel Computing Workshop*, paper P2-C, Kawasaki, Japan, November 1994.

[5] D.A. Hawking and P.B. Thistlewaite. Searching For Meaning With The Help Of A PADRE. In *Proceedings Of The Third Text REtrieval Conference (TREC-3)*, US Department of Commerce, NIST Special Publication 500-225, pages 257-267, April 1995.

[6] D. Hawking, P. Bailey, D. Campbell, P. Thistlewaite and A. Tridgell. A PADRE in MUFTI (A Multi User Free Text retrieval Intermediary). In *Proceedings of the Fourth Parallel Computing Workshop* pages 75-84, Imperial College, London, September 1995.

[7] D.A. Hawking and P.B. Thistlewaite. Proximity Operators - So Near And Yet So Far. To appear in *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, US Department of Commerce, NIST.

[8] A. Tridgell and D. Walsh. The HiDIOS Filesystem. In *Proceedings of the Fourth Parallel Computing Workshop*, pp. 53-63. Imperial College, London September 1995.

[9] *World Wide Web Consortium home page.* `http://www.w3.org/`.

[10] *PADRE home page.* `http://cap.anu.edu.au/cap/projects/text_retrieval/`