Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

Implementation issues for a parallel Psuedo-Search Engine Indexer using MPI and Genetic Programming

R. L. Walker
Computer Science Department, University of California at Los Angeles, USA

Abstract

A parallel Pseudo-Search Engine Indexer has been implemented to study the feasibility of developing a Genetic Programming (GP)/MPI tool for an assessment of Search Engines Indexers. The future improvements to this initial implementation of the GP/MPI tool will incorporate new and enhanced methodologies for GP as well as new developments in the area of Knowledge Discovery in Databases (KDD). Several implementation issues became apparent during this initial phase and will be discussed. The parallel implementation displayed the diversity that exists among the current sets of Web pages that comprise the World Wide Web (WWW). Hierarchy interface issues exist for a set of indexers, a set of Web crawlers, and a set of browser mechanisms. Network issues will provide additional challenges for the efficient execution of Web crawlers. The network issues became apparent in the relevant pages presented by the current search engine browsers. The addition of multimedia traffic to the current as well as the improved Internet will magnify some of the existing implementation issues. These issues provide the motivation to develop mechanisms for handling the exponential growth of WWW documents as this research effort continues. The training sets used in this study provided a small snapshot of the computational effort required to index Web documents accurately and efficiently. Results will be used to develop and implement Web crawler mechanisms capable of presenting the scope of this research effort.

1 Introduction

The Web pages used in this study consisted mostly of Yahoo! Business Headline documents (Yahoo! [21]). These documents were incorporated in the training sets (Koza [7]) since their contents were mostly text. Some of the other documents, ordinary Web pages, contained errors that may occur when a chosen browser displays erroneous data and/or error messages as it parses the selected Web document. A browser produces erroneous results when a Web page designer attempts to create a compact Web page containing a large number of HyperText Markup Language (HTML) related tags, attributes, and values (Musciano & Kennedy [10]). The page designer creates a compact Web page by concatenating several of the HTML related tags on a single line. The problem occurs as the host computer attempts to read each line of text into an internal buffer. A line length greater than the machine's line limit causes the process to crash. Different machines impose different line buffer limits for Web browser software. This technique for compact Web pages attempted to reduce the network load associated with the retrieval and transmission of large documents to/from remote sites.

2 Current Need for Improvements to the Information Retrieval(IR) Systems

The indexers for the early and current search engines used the existing state-of-the-art techniques to retrieve and rank the relevant documents for each request. The coverage of the relevance rankings was not consistent, depending on human and computer constraints. The purposes of improving the early IR systems were twofold. The IR systems (Kraft et al. [9], Stillger & Spiliopoulou [17], Raymer et al. [14]) corrected the precision and recall mechanisms for databases of millions of documents that may be centrally or distributively located throughout the Internet (Ballardi et al. [3]). The growing transmission delays on the Internet impact the efficiency of any remote server. The lack of knowledge about a given indexer (Walker [18]) can also result in several searches over the Internet on the same set of databases with no relevant results.

The goal of precision mechanisms in the IR systems focused on determining the commonality among distinct subsets of documents (Ryu & Eick [15], Oakley [12]). The relevance rating for the appropriateness of a document has been increased in some of the current search engines by coupling the methodologies of their IR systems with Genetic Algorithms(GA) and/or Genetic Programming(GP) (Koza [7]). The improved relevance feedback determined the fitness measure of the requester's database query by computing the relationship between the precision and recall of relevant documents.

The model of an IR system (Kraft et al. [9]) has some characteristics that are similar to those of GP. The model of an IR system consists of:

1. a set of documents or records;
2. a set of index terms (single words or phases); and
3. a fitness function that is used to measure the closeness of a query to retrieving a relevant document.

The components that comprise the GP model (Willis et al. [20]) are divided into two major units: a) the search space and b) a component that controls the quality and speed of the GP system. The search space consists of:

1. the terminal set;
2. the function set; and
3. fitness function.

The quality and speed control component consists of:

1. the algorithm control parameters; and
2. the terminal criterion.

The methodology for Knowledge Discovery in Databases(KDD) (Ryu & Eick [15]) outlined approaches that may be taken by search engines for their IR systems. The conventional approach presented the requester with the query results based on the user's knowledge of the respective IR systems. The typical user normally has limited knowledge about the structural and search methodologies that pertain to the individual search engines. This is a limitation with current search engines. The KDD approach derives queries from the resulting databases built by the search engines. The IR system organizes the database and presents the user with useful information. KDD systems require no in-depth knowledge by the end-users about the incorporated structure and search methodologies. The KDD IR system does require an intelligent tool coupled with the methodology to eliminate repeated queries (Angeline [1]).

3 Implementation Issues for the Pseudo-Search Engine Indexer

Several problems initially hampered the efficient implementation of the indexer for the pseudo-search engine indexer. The indexer development was conducted on a dedicated system that does not have the same memory constraints as most multi-user computer systems. This system, an ideal

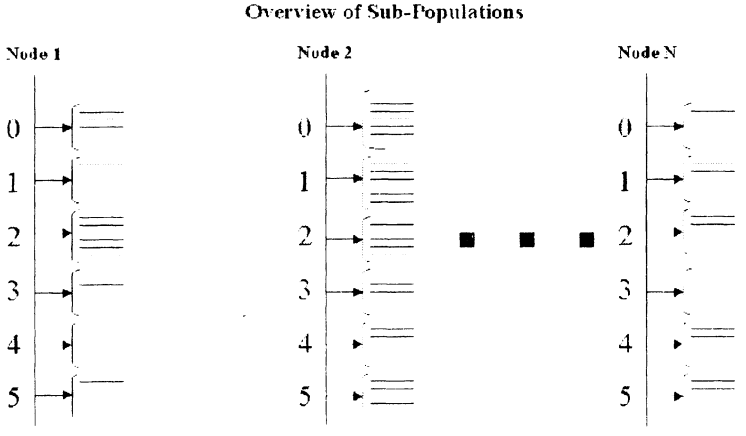

Figure 1: Distribution of Web Pages.

environment, suffered from the same memory constraints but only after indexing over 200 Web pages. The indexer's ideal model might consist of a series of hash tables that can be used for the retrieval of indexed Web pages (see Figure 1). This implementation became impractical as the initial training set was increased in size and the system limitations restrict the allocation of dynamic memory. Memory constraints were documented in the literature (Koza & Andre [8], Oussaidéne et al. [13], Sherrah et al. [16]) as a major limiting factor in the size of the training sets associated with different GP applications.

The performance of the parser for the Pseudo-Search Engine Indexer showed the diversity of approaches (Berners-Lee [4]) among the current Web page designers. This diversity was displayed by the different mixtures of Java applets and CGI files embedded in variations of HTML documents versus standard HTML documents. The page parser was not used to act as a language compiler because parsing exactness was not necessary. The abuse of the basic HTML formats (Musciano & Kennedy [10]) led to the developer suffering more than the user because the information may not

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

Table 1: **Execution times for n processors indexing m WEB pages using a network of workstations.**

	Number of WEB Pages									
nprocs	1	2	4	8	16	32	64	128	256	512
1	3.6	8.9	18.1	32.1	59.6	89.2	174.9	275.1	719.6	1346.1
2	1.5	3.6	7.4	13.1	24.8	37.9	75.1	139.6	313.7	923.6
3	1.5	2.2	4.6	6.6	12.6	19.5	39.3	61.5	161.7	304.5
4	1.6	2.2	4.0	4.8	9.9	13.4	27.1	43.1	112.9	209.0
5	1.7	2.2	2.6	4.3	8.5	11.2	22.2	36.0	91.9	168.1
6	1.5	2.2	2.7	3.7	7.8	10.4	17.4	30.4	77.4	144.0
7	1.5	2.2	2.8	3.2	6.3	7.4	14.9	25.9	67.5	120.7
8	1.5	2.2	2.7	2.8	6.1	7.0	13.6	23.4	61.2	110.0
9	1.5	2.2	2.6	2.8	5.8	6.4	12.9	22.0	57.6	103.0
10	1.6	2.2	2.6	2.8	5.7	7.1	13.2	22.1	57.3	101.7

be conveyed. The user may not have any idea what portions of a Web document are missing unless the user designed or viewed the page in the past. An example of the basic HTML format is

```
<html>
<head>
<title>Title of the HTML document</title>
</head>
<body>
<h3>Title of the HTML document</h3>
Related text for this document.

<!-- Comments to associate with the text. Or the
     script commands associated with the Cascading
     Style Sheets (CSS) model and JavaScript-based
     Style Sheets (JSS) -->

More related text for this document.

</body>
</html>
```

Certain language-dependent HTML tags can be used to consistently display different versions of a page when text may be substituted for an image. The use of the **alt** tag provides one example. Otherwise, some information will be dropped when browsers cannot decipher the designer tag or tag attributes. The worst abuse of a tag occurs when the designer does not adhere to a HTML tag format such as the character string "<!--" for the beginning of a comment, but used "<! --" or some permutation. This added unnecessary words to the document word count when the designer

used the text as comments. The additional words can eventually impact the accuracy of a search engines' *relevancy rating* (Glossbrenner & Glossbrenner [6]). Some of the current search engines employ some form of inverted document frequency (Kraft et al. [9]) to compute their ratings and others may use the results to compute a similarity metric (Wasfi [19]) to implement various information filters.

Problems can also occur when the existing indexer is not updated when new Web page formats (Nielsen et al. [11]) are added by designers for more efficiency and flexibility. The dynamic environment of the Internet will spur several derivations and enhancements of the current HTML formats until they become too limiting, due to network constraints such as throughput and reliability. These enhancements will also lead to the development of more Web pages that may not be executed correctly.

Table 2: **Execution times for n processors indexing m WEB pages using the IBM SP2.**

	Number of WEB Pages									
nprocs	1	2	4	8	16	32	64	128	256	512
1	1.1	2.4	4.6	8.4	30.1	40.7	81.7	73.3	382.0	590.2
2	0.9	2.3	4.7	8.2	42.6	41.1	82.0	128.8	199.5	593.6
3	1.5	3.1	3.4	6.8	21.5	15.0	41.9	63.8	149.1	361.4
4	1.4	2.1	3.1	5.0	12.9	12.6	31.7	42.9	123.2	442.3
5	2.8	2.8	3.9	4.7	12.7	10.7	25.8	35.3	112.4	171.8
6	1.8	2.7	3.5	4.4	10.0	9.6	22.0	30.5	94.8	145.3
7	1.4	2.9	3.8	4.2	10.4	8.7	18.7	27.2	81.0	130.7
8	1.7	3.4	3.4	4.0	8.1	7.4	15.9	23.9	74.6	117.0
9	1.5	2.6	3.6	3.7	8.2	7.4	14.2	21.0	70.8	103.1

4 Experimental Results

One set of results were generated on a cluster of nine Sun workstations. The workstations were labeled as $Node_0, Node_1, \cdots, Node_9$ with $Node_1$ and $Node_9$ the same. A second set of results were generated on an eight node IBM SP2 cluster. The nodes were labeled as $Node_0, Node_1, \cdots, Node_8$ with $Node_0$ and $Node_8$ the same (see Figure 1). This inclusion of a dual client node showed that there will be a performance degradation when a node is used to represent multiple clients. The fitness computations begins after the training set is parsed. These computations will be necessary to index (order) the training subset for each individual node. These computation results will be merged and repeated for several generations (Koza [7]) on the training set as well as on a constantly growing set of Web documents. The effects of multiple client indexer nodes on the GP population of pages becomes more apparent as the number of parsed Web pages increase.

Tables 1 and 2 present the timing measurements associated with each sub-cluster of nodes for powers of two Web pages. $Node_0$ was used as a dedicated server node with the other nodes being treated as its clients when the number of nodes was greater than one. The client nodes sent requests to the server node for Web pages and for access to the output channel. This study suppressed all output for consistency in the timing results. The message passing mechanisms remained in place. The initial approach for channel access appeared to be a mechanism similar to a token ring (Bagrodia [2]). The inconsistency of the file size for Web pages showed that the token would sit in a nodes' message queue as it was indexing a variable size file at its local Web site. Each node was treated as a Web site for storing a distinct set of Web pages.

The MPI implementation used the message tags from the server node to determine which subroutine would be executed by the client node(s) for the information stored in the transmitted message. The client node(s) informed the server node by a general message tag that it was available to index/search another Web page. Currently, the client nodes sit idle until the server node commands the processing of a data stream for a Web page. This study showed that a hierarchy must exist among the nodes in order to eliminate the duplication of Web pages on several nodes simultaneously. The group server concept can ensure that members of its group never parse the same page on multiple machines. The search overhead and fitness measure computations increase with the duplication of an indexed Web page. Duplication also decreases the overall efficiency for a group of Web sites. This became apparent based on the memory resources that vary greatly, derived from Web page contents and size.

The problems that exist for page duplication are similar to the problems that routers (Floyd & Jacobsen [5]) have on a computer networks. The routers use router tables (Ballardi et al. [3]) to provide an efficient mechanism for determining the most effective route for transmitting data between communicating computers. A similar mechanism can be used by the sever node to determine if any client node has received the page that needs to be parsed. This information will be based on the Web page name as opposed to the Web page contents.

5 Conclusion

This preliminary study provides a small snapshot of the computational effort needed to index Web documents accurately and efficiently. The results generated by the development and implementation of the Crawler mechanisms will examine the scope of this research effort. The Pseudo-Indexer results showed that each Web site will have an adequate workload when all the indexing mechanisms coupled with GP are employed. The communication overhead will increase when a mechanism equivalent to cache coherency

is implemented to keep consistency among all of the Web sites for calculating the relevancy ratings for simple and complex queries.

Acknowledgements

The author wishes to thank Walter Karplus and Zhen-Su She for their direction and suggestions. The author acknowledges Elias Houstis, Ahmed K. Elmagarmid, Apostolos Hadjidimos, and Ann Catlin for their support, encouragement, and access to the computer systems at Purdue University. Special thanks to the 1999 UCLA SMARTS students, London Wright-Pegs and Samir Asodia, for their assistance with the figures in this paper.

This work was supported by the Raytheon Fellowship Program. The implementation results associated with the network of workstations were generated on computers located in the Department of Computer Sciences, Purdue University, West Lafayette, IN 47907.

References

[1] Angeline, P.J., Genetic programming: a current snapshot, *Proc. of the 3rd Annual Conf. on Evolutionary Programming*, eds. A.V. Sebald & L.J. Fogel, World Scientific, Singapore, pp. 224-232, 1994.

[2] Bagrodia, R., Process Synchronization: Design and Performance Evaluation of Distributed Algorithms, *IEEE Transactions on Software Engineering*, **15**, pp. 1053-1064, 1989.

[3] Ballardi, T., Francis, P. & Crowcroft, J., Core Based Trees (CBT): An Architecture For Scalable Inter-domain Multicast Routing, *ACM Transactions on Computer Systems*, pp. 85-93, 1993.

[4] Berners-Lee, T., *Information Management: A Proposal*, CERN Technical Report, CERN - European Laboratory for Particle Physics, pp. 1-14, 1989.

[5] Floyd, S. & Jacobsen, V., The Synchronization of Periodic Routing Messages, *IEEE/ACM Transactions on Networking*, pp. 1-28, 1994.

[6] Glossbrenner, A. & Glossbrenner, E., *Search Engines for the World Wide Web*, Peachpit Press, Berkeley, 1998.

[7] Koza, J.R., Survey of genetic algorithms and genetic programming, *Proceedings of WESCON'95*, IEEE Press, New York, pp. 589-594, 1995.

[8] Koza, J.R. & Andre, D., *Parallel Genetic Programming on a Network of Transputers*, Technical Report STAN-CS-TR-95-1542, Stanford University, Department of Computer Science, Palo Alto, 1995.

[9] Kraft, D.H., Petry, F.E., Buckles, B.P., & Sadasivan, T., The use of genetic programming to build queries for information retrieval, *Proc. of the 1st IEEE Conf. on Evolutionary Computation*, IEEE Press, New York, pp. 468-473, 1994.

[10] Musciano, C. & Kennedy, B., *HTML: The Definitive Guide*, O'Reilly and Associates, Inc., Sebastopol, CA, 1997.

[11] Nielsen, H.F., Gettys, J., Baird-Smith, A., Prud'hommeaux, E., Lie, H.W. & C. Lilley, Network performance of HTTP/1.1, CSS1, and PNG, *Proc. of ACM SIGComm '97*, ACM Press, Baltimore, MD, 1997.

[12] Oakley, E.H.N., Genetic programming, the reflection of chaos, and the bootstrap: Towards a useful test for chaos, *Proc. of the Genetic Programming 1996 Conf.*, eds. J.R. Koza, D.E. Goldberg, D.B. Fogel & R.L. Riolo, MIT Press, Cambridge, MA, pp. 175-181, 1996.

[13] Oussaidéne, M., Chopard, B., Pictet, O.V. & Tomassini, M., Parallel genetic programming: An application to trading models evolution, *Proc. of the Genetic Programming 1996 Conf.*, eds. J.R. Koza, D.E. Goldberg, D.B. Fogel & R.L. Riolo, MIT Press, Cambridge, MA, pp. 357-362, 1996.

[14] Raymer, M.L., Punch, W.F., Goodman, E.D. & Kuhn, L.A., Genetic programming for improved data mining - application to the biochemistry of protein interactions, *Proc. of the Genetic Programming 1996 Conf.*, eds. J.R. Koza, D.E. Goldberg, D.B. Fogel & R.L. Riolo, MIT Press, Cambridge, MA, pp. 375-380, 1996.

[15] Ryu, T.-W. & Eick, C.F., MASSON: discovering commonalities in collection of objects using genetic programming, *Proc. of the Genetic Programming 1996 Conf.*, eds. J.R. Koza, D.E. Goldberg, D.B. Fogel & R.L. Riolo, MIT Press, Cambridge, MA, pp. 200-208, 1996.

[16] Sherrah, J., Bogner, R.E. & Bouzerdoum, B., Automatic selection of features for classification using genetic programming, *Proc. of the 1996 Australian New Zealand Conf. on Intelligent Information Systems*, eds. V.L. Narasimhan & L.C. Jain, IEEE, New York, pp. 284-287, 1996.

[17] Stillger, M. & Spiliopoulou, M., Genetic programming in database query optimization, *Proc. of the Genetic Programming 1996 Conf.*, eds. J.R. Koza, D.E. Goldberg, D.B. Fogel & R.L. Riolo, MIT Press, Cambridge, MA, pp. 388-393, 1996.

[18] Walker, R.L., Assessment of the Web using genetic programming, *GECCO-99: Proc. of the Genetic and Evolutionary Conf.*, eds. W.

Banshaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela & R.E. Smith, Morgan Kaufman, San Francisco, pp. 1750-1755, 1999.

[19] Wasfi, A.M.A., Collecting user access patterns for building user profiles and collaborative filtering, *Proc. of the 1999 Int. Conf. on Intelligent User Interfaces*, ed. M. Maybury, ACM Press, Baltimore, MD, pp. 57-64, 1999.

[20] Willis, M.J., Hiden, H.G., Marenbach, P., McKay, B. & Montague, G.A., Genetic programming: an introduction and survey of applications, *Proc. of the 2nd Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications*, IEE Press, London, UK, pp. 314-319, 1997.

[21] Yahoo!, *Yahoo! Web Page*, Yahoo! Inc., Santa Clara, CA, 1998.