PARALLEL QUERY PROCESSING USING WARP EDGED BUSHY TREES IN MULTIMEDIA DATABASES

Lt. S. Santhosh Baboo
Department of Computer Science, D.G Vaishnav College, Arumbakkam, Chennai 106. India.

P. Subashini
Department of Computer Science, Avinashilingam Deemed University, Coimbatore, India.

K. S. Easwarakumar
Department of Computer Science Anna University, Chennai, India.

Keywords: Multimedia databases, Bushy trees, Warp edges, Cost model.

Abstract The paper focuses on parallelization of queries execution on a shared memory parallel database system. In this paper, a new data structures, named Warp edged Bushy trees, is proposed for facilitating compile time optimization. The warp edged bushy tree is a modified version of bushy trees , which provides better response time than bushy trees, during query processing.

1 INTRODUCTION

A database is a repository of data that traditionally contains text, numeric values, boolean values and dates, known as 'printable' objects (Golshani, F. and Dimitrova N., 1998). A multimedia database additionally contains graphical images; video clips and sound files, known as 'presentable' objects. Users may retrieve information from a database without having any knowledge of how or where that data is stored. The paper focus on bushy trees in section 2, warph edged bushy trees in section 3, cost model in section 4, experimental results and conclusion are in section 5 and 6.

There are a number of issues relating to multimedia database management systems and multimedia information servers that are caused by the particular nature of multimedia data. These can be summarized as:

- The development of appropriate data modeling techniques for multimedia information, including video, sound and graphical images,
- Indexing and searching techniques for the retrieval of multimedia data objects,
- Developing efficient storage schemas that support heterogeneous data objects,
- Issues relating to the delivery quality of multimedia objects.

Modern database applications, such as data mining and decision support pose several new challenges to query optimization and processing (Haiyun He and Curtis Dyreson, 2002) (Silberschatz et al, 1996). Parallelism is one of the key technologies to handle these challenges (M. T. Ozsu and P. Valduriez, 1997) but increases the complexity of query optimization (Hasan et al., 1996). The most crucial problem to be solved within query optimization is ordering the operators (D. Taniar and Y. Jiang, 1998). The operators describe the dependencies between the different algebra operators of a query and determines at the same time the degree of inter operator parallelism. The problem mainly focuses on the search for an optimal ordering with respect to cost function.

Rather than navigating blindly into the search space (K.-L. Tan and H.Lu, 1991) (Spiliopoulou et al., 1996) (Leonindes Fegaras, 1998), we propose to consider a subspace of it, called Warp edged Bushy trees. **Warp edged bushy trees** identify a regular in the search space including inter-operator parallelism and face a significantly smaller space rather than the

S. Santhosh Baboo L., Subashini P. and S. Easwarakumar K. (2006).
PARALLEL QUERY PROCESSING USING WARP EDGED BUSHY TREES IN MULTIMEDIA DATABASES.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - DISI*, pages 273-276
DOI: 10.5220/0002487802730276

general one. Introducing warp edges into any tree increases its efficiency.

2 BUSHY TREES

Two major forms are actually distinguished (Zait et al., 1996), Bushy trees and warp edged bushy trees. In bushy trees, two or more join operators lying on different paths of a query-processing tree can be processed at the same time. Almost all query optimizers working on bushy trees have yet considered the complete spawn search space. Fig. 1 shows the bushy trees. Supposing that a hash based join is used and all hash tables of the left input relations are built for a right deep tree, the tuples of the right input can be pipelined through the whole tree (D. Schneider and D. J. DeWitt, 1990). This pipelining can be implemented very efficiently if the entire hash table of the left input relation fits in the main memory.

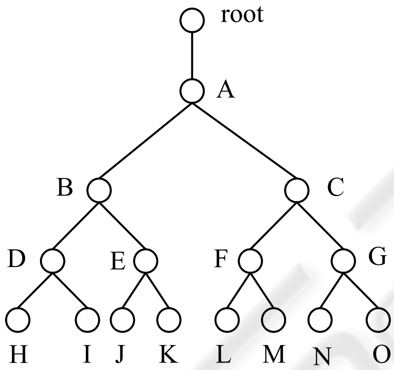

Figure 2: Bushy Trees.

3 WARP EDGED BUSHY TREES

A warp edge is an edge that is something other than a parent to child edge i.e. an edge from an element to a sibling or a grand child. Warp edges can be dynamically generated and stored during query evaluation to improve the efficiency of future queries.

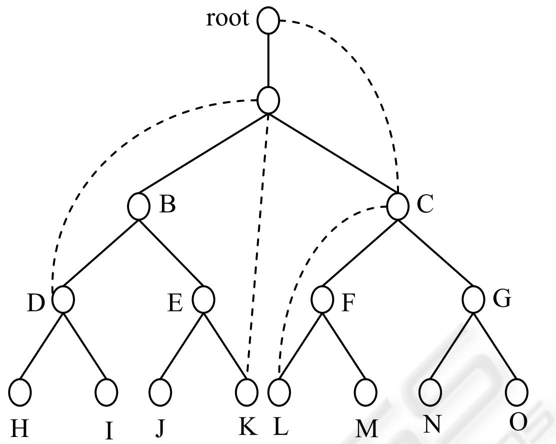

Figure 3: Warp Edged Bushy Tree.

The warped edged bushy tree created is shown in the fig. 3. The warp edges connect the root with each section since the query starts at the document root and terminate at each section. The size of the document is small and sections can be found quickly, but in general the document could contain thousands of sections.

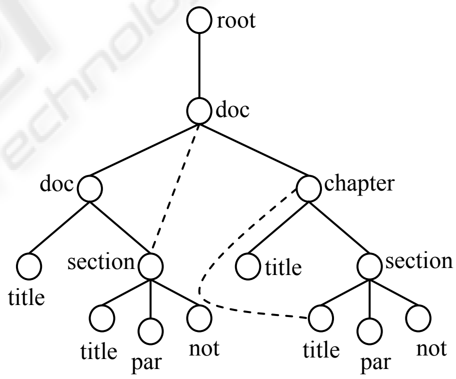

Figure 4: Warp Edged Bushy Query Trees.

A warp edge is an edge in the data model that traverses more than one level in the query tree. Typically warp edges are added as the result of previous queries. The warp edges can be traversed during the evaluation of a query.

4 A COST MODEL FOR BUSHY PARALLELISM

Our cost model, hereafter denoted as "BO", exploits bushy parallelism only. Joins appearing in different subtrees of the QEP are executed independently, as soon as their input streams are available for processing. Thus, the cost of the QEP is the cost of the most expensive branch.

In this execution scenario, the output of a join must be stored in the local disk, from which it is retrieved by its consumer. Hence, the cost of a join consists of the time needed to retrieve its input streams from a local or remote disk, the time for local processing, and the time for storing the output stream locally:

$$Tcost(x) = Tin(x) + Tlocal(x) + Tout(x)$$

The execution cost of the tree rooted at x is the total elapsed time Ecost(x) from the beginning of query execution until the completion of join x3. The cost is equal to the execution time of x and the time required by its slowest producer to complete execution:

$$Ecost(x) = T(x) + \begin{cases} 0, & \text{x is a leaf} \\ Ecost(y), & \text{x has one child, join y} \\ \max(Ecost(y),\ Ecost(z)), & \text{both children of x are joins y,z} \end{cases}$$

Hence, the execution time of a QEP is the time required for its root process r to complete execution, Ecost(r).

Parameters of the experiments: We have studied the cost distribution for 10 query sizes. For each size, we have generated an acyclic query graph. The database parameters are summarized in Table 1. We consider two databases, the Small and the Large one, containing relations with different size ranges. Out database and query settings are close to those used in "portofolio" database experiments, as presented in (Lanzelotte et al., 1993).

The settings of the parallel architecture assumed by the cost models are shown in Table 2. The small size of processor memory was intended to counterbalance the modest size of the database relations, in the sense that main memory should not be adequate to hold all relations.

Table 1: Database and Query parameters.

Relation sizes:	Small database: 1,000 to 10,000 tuples
	Large database: 1,000 to 1,00,000 tuples
Attribute sizes	8 - 20 bytes
Output attributes:	4
Number of joins:	10 - 100

Table 2: Parallel Machine parameters.

Page sizes	: 1024 bytes
Page transfer time – Network	: 1.7 msec (600 Kbytes/sec)
Page transfer time - Local disk	: 8.3 msec
Page transfer time - Remote disk	: 10 msec
Number of Processors	: 100
Processor Memory	: 800 Kbytes

The base relations input to the leaf nodes, all intermediate results not fitting in main memory, and the output streams for the BO model, imply I/O accesses. The I/O cost for processing a very large (intermediate) relation can thus easily become the dominant factor, especially for small queries. Therefore, QEPs processing the same relation can have the same cost, although they may differ in the rest of their structure. For the Small database, the diversity is higher and a wide spectrum of values is covered smoothly. This is due to the lower diversity of relation sizes occurring in this database.

5 RANDOM EXPERIMENT

In this section, we describe a set of experiments on randomly generated data. Our aim is to test the performance of warp edged bushy trees over normal bushy trees.
We conducted the test on randomly generated XML documents. We categorized the test in two ways. First we tested the Time factor of both the tress. Second we tested the Space factor of the trees. We randomly conducted the test on 500 queries.

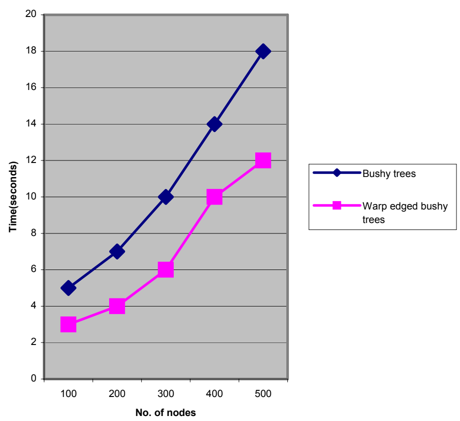

Figure 1: Varying the time factor.

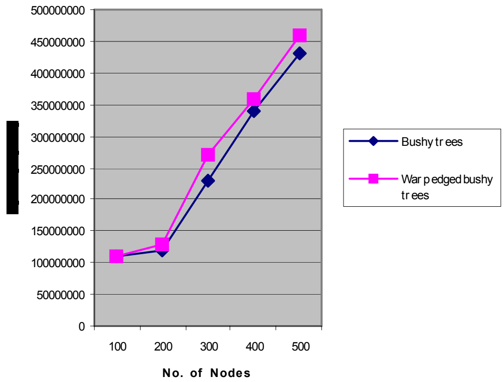

Figure 2: Varying the Space factor.

The first figure shows that warp edged bushy trees take lesser time than bushy trees. It shows that the turnaround time for query evaluation is lesser for warp optimization. The optimization approximately halves the time needed for query evaluation at a modest increase in the amount of space. The second figure shows that warping occupies more space than normal bushy tyrees But the space occupied by such optimized trees are at modest level only.

Overall, the experiments show that while warp edged bushy tress needs a small amount of additional space, it can improve query performance for bushy trees.

6 CONCLUSIONS

The paper emphasized the warp edging optimization on normal bushy trees in multimedia databases. Any query can be done easily using query trees. Result shows that multimedia databases can be represented using bushy trees. Warp edges are dynamically generated on the bushy query trees and stored during query evaluation to improve the efficiency of future queries. The technology needed to such optimization can be implemented as a layer on top of any evaluation engine. Experiments shows that in the evaluation the use of warp edges results in substantial savings of times at a modest increase in space. So the objects stored in image documents can be retrieved based on some query very fastly when we use warp edging in query trees.

REFERENCES

Haiyun He and Curtis Dyreson, Warp-Edge Optimization in Xpath, Springer-Verlag 2002.

Golshani, F. and Dimitrova N. , A Language for Content-Based Video Retrieval, Multimedia Tools and Applications 6, 1998, pp. 289-312.

A. Silberschatz, M. Stonebraker, and J. Ulman. Database research: Achievements and opportunities. Into the 21st century. SIGMOD Record, 25(1):52-63, March 1996.

M. T. Ozsu and P. Valduriez. Distributed and Parallel Database Systems, pp. 1093-1111. CRC Press, 1997.

W. Hasan, D. Florescu, and P. Valduriez. Open issues in parallel query optimization. SIGMOD Record, 25(3): pp. 28-33, September 1996.

D. Taniar and Y. Jiang. A high performance object-oriented distributed parallel database architecture. In HPCN Conference 98, pp. 498-517. Springer Verlag, April 1998.

K.-L. Tan and H.Lu. A Note on the Strategy Space of Multiway Join Query Optimization Problem in Parallel Systems. SIGMOD Record, 20(4):pp. 81-82, December 1991.

M. Spiliopoulou, M. Hatzopoulos, and Y. Contronis. Parallel Optimization of Large Join Queries with Set Operators and Aggregates in a Parallel Environment Supporting Pipeline. IEEE Transactions on Knowledge and Data Engineering, 8(3): pp.429-445, June 1996.

Leonindes Fegaras. A new heuristic for optimizing large queries. In International Database and Expert Systems Applications Conference, pp. 726-735, Vienna, Austria, August 1998. Springer Verlag LNCS 1460.

M. Zait, D. Florescu, and P. Valduriez. Benchmarking the DBS3 Parallel Query Optimizer. IEEE parallel and distributed technology: systems and applications, 4(2): pp. 26-40, 1996.

D. Schneider and D. J. DeWitt. Tradeoffs in processing complex join queries via hashing in multi-processor database machines. In Proceedings of the International Conference on Very Large Databases, pp. 469-490, Melbourne, Australia, August 1990.

Rosana Lanzelotte, Patrick Valduriez, and Mohamed Zait. On the effectiveness of optimization search strategies for parallel execution spaces. In Int. Conf. on very Large Databases, pp. 493-504, Dublin, Ireland, 1993.