Parallel Match Algorithm to Speed up Evaluation of Path-oriented Queries in Document Databases

Hee-Sook Park *, Woo-Hyun Cho, Hee-Hiong Ngu and Carol Hazlewood

Department of Computer Engineering

Pukyong National University

559-1 Daeyon-dong, Nam-gu, Busan, 608-737, Republic of Korea

E-mail: Hee-Sook Park mailto:bg007@edunet4u.net

Key Words : Path Signature, Parallel Match Algorithm

ABSTRACT

XML is a popular syntax for a semistructured data to exchange and manipulate documents on network. The potential of XML is unlimited and many applications using XML are arising. A path-oriented query frequently is issued toward XML documents. Then, the database management system has to evaluate the query that contains a path and to find the stored location in the database. So, an indexing algorithm for high performance is needed to speed up evaluation process of the path-oriented query. Queries navigate semistructured data and can be accelerated using path expressions. To evaluate the path-oriented query toward XML document, we need to match the path in the query and the paths for all elements in the XML document. The sequential methods to improve performance of evaluation for the path-oriented query have been proposed. An indexing technique that avoids access to non-related elements and can be enhanced by combining the technique of pat-trees with scanning of signatures is proposed. A solution that encodes paths as string and inserts those strings into a special index that is highly optimized for long and complex keys is described.

A method for indexing and storing XML data based on a numbering scheme for elements is introduced. In this paper, we concern the parallel match algorithm to speed up evaluation process of the path-oriented query using path signatures.

```
<Inventory code="J100a12">
 <Maker>
  <Items>
   <name> K1-123 </name>
   <price> 23,000</price>
   <quantity> 400 </quantity>
   <date> July 25th 2003 </date>
  </Items>
 </Maker>
 <warehouse no="kp-001">
  <location> Pusan city</location>
  <manager> Kim Yoo-Sin </manager>
  <date> June 5th 2004 </date>
 </warehouse>
</Inventory>
```

Fig. 1. A sample XML document

Table 1. A path signature file for a sample XML document.

Location	Path of Element	Path Signature(h=7)
1	Inventory	0011001
2	Inventory/Maker	1010110
3	Inventory/Maker/items	0001101
4	Inventory/Maker/items/name	1100101
5	Inventory/Maker/items/price	0000101
6	Inventory/Maker/items/quantity	1100001
7	Inventory/Maker/items/date	0011101
8	Inventory/warehouse	0100101
9	Inventory/warehouse/location	0000110
10	Inventory/warehouse/manager	0010011
11	Inventory/warehouse/date	0010110

First, using the binary match trie that is a binary trie in which the values of successive bits in the binary pattern determine a path of the tree from root downward. The maximum level of binary match tire corresponds to the length of a path signature(h).

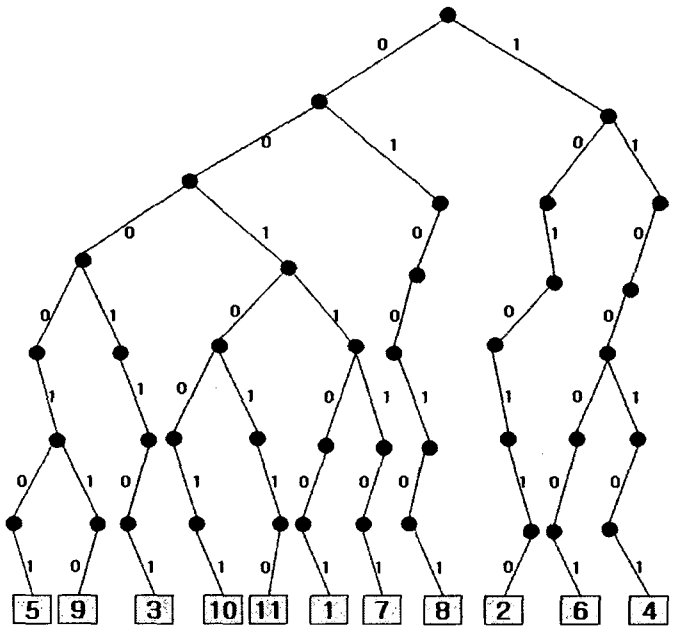

Fig. 2. Binary trie structure of the path signature file.

Next, we transform the binary match trie into the parallel match structure and construct the parallel match algorithm for each processing element. A parallel match structure is a connection of processing elements. Each processing element has one input line, two(left and right) output lines, some storages, and simple comparator. The parallel match structure equals to configuration of the binary match trie. That is, a node corresponds to a processing element, and links to input or output lines. All of processing elements in parallel match structure operate in parallel to perform a matching process.

To match using the parallel match structure, the path signature to be matched should be serially entered into the processing element corresponding to the root node by rightmost bit first. The 'match' signal is attached as the last input.

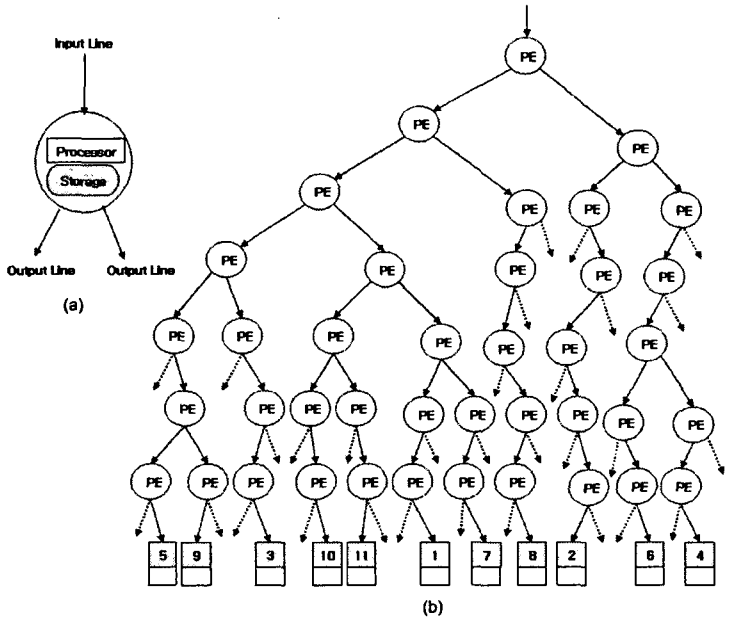

Fig. 3. Each processing element and parallel match structure.

Our proposed parallel match algorithm by parallel match structure consists of two steps for matching path signature. After doing first step, the root node of level 1 in parallel match structure has leftmost bit of path signature to be matched and nodes of level i have i-th bit of path signature. In second step, the 'match' signal is entered into the root node and then the root node generates 'match' or 'nomatch' signal as this node has stored '0' or '1'. If a node has stored '0' signal then generates 'match' signal to left and right nodes, and if a node has stored '1' signal then generates 'nomatch' signal to left node and 'match' signal to right node. After each 'match' or 'nomatch' signal reaches leaf node, each leaf node corresponding to the given path signature to be matched holds 'match' signal but each leaf node not to be matched holds 'nomatch' signal.

For example, if we assume that a path signature value for path-oriented query issued by user has a bit pattern of '1000100'. We can get the location information 2 of the signature value '1010110' and 4 of the signature value '1100101' as shown in Fig. 4.

Finally, we analyze the performance of our parallel match method. The time complexity of our method has the order-of-magnitude similar to the logarithm of N to the base 2, where N is the number of path signatures for an XML document. Also, we simulate our proposed parallel match algorithm to confirm correctness.

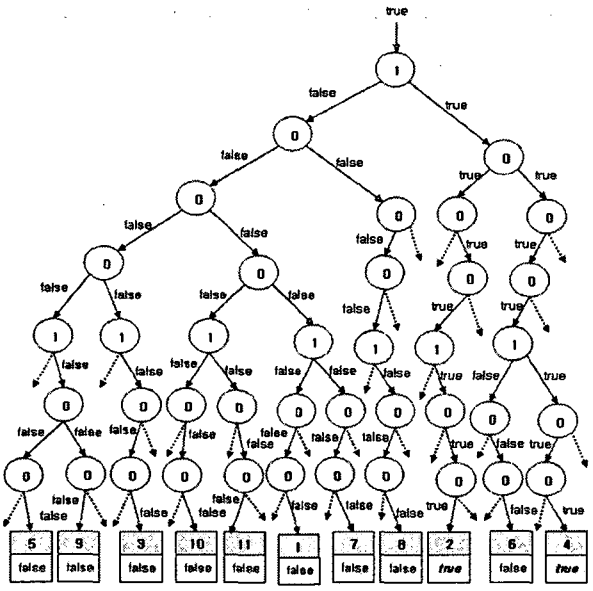

Fig. 4. An example of parallel match processing.

Reference

[1] T. Bray, E. Maler, J. Paoli, C. M. Sperberg McQueen(2000), Extensible Markup Language(XML) 1.0, W3C Recommendation, http://www.w3.org/TR/2000/REC-xml-20001006/.

[2] A. Deutch, M. Fernandez, D. Forescu, A. Levy, D. Suciu(1998), XML-QL: A Query Language for XML, http://www.w3.org/TR/NOTE-xml-ql/.

[3] S. Boag, D. Chanberlin, M. Fernandez, D. Florescu, J. Robie, J. Simeon(2002), XQuery 1.0: An XML Query Language, http://www.w3.org/TR/2002/WD-xquery-20020816/.

[4] D. Eastlake, J. Reagle, D. Solo,(2002), XML-Signature Syntax and Processing, W3C Recommendation, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/.

[5] Yangjun Chen, Gerald Huck(2000), Path Signatures: A Way to Speed up Evaluation of Path-oriented Queries in Document Databases, Proceedings of the International Conference on Web Information System Engineering(WISE 2000), pp. 240-244.

[6] Brian F. Cooper et al(2001), A Fast Index for Semistructured Data, Proceedings of the 27th International Conference on Very Large Data Bases table of contents(VLDB 2001).

[7] Q. Li, B. Moon(2001), Indexing and Querying XML Data for Regular Path Expression, Proceedings of the 27th International Conference on Very Large Data Bases table of contents(VLDB 2001).

[8] C. Faloutsos, Access Methods for Text(1985), ACM Computing Surveys, Vol. 17, No.1, pp. 49-74.

[9] Carlo Zaniolo et al(1997), Advanced Database Systems, Morgan Kaufmann Publishers.

[10] William B. Frakes, Ricardo Baeza-Yates(1992), in: Information Retrieval: Data Structures and Algorithms, Prentice Hall PTR; Facsimile edition.