International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 2 Issue: 2 240 – 245

An Efficient Dynamic XML Data Broadcasting Method in Mobile Wireless Network Using XPATH Queries

M.Chandrapriya[1], R.Ganeshan [2]
PG Scholar[1]
Assistant Professor CSE Department[2]
St.Joseph College of Engineering
Sriperumbudur, Chennai-602105
Chandrapriya08@gmail.com,ganeshramasamy111@gmail.com

***Abstract*—** Wireless mobile computing has become popular. Users communicate in the wireless mobile environment using their mobile devices such as smart phones and laptops while they are moving. In previous system can support only static XML rendered from repositories. It is not efficient for dynamic broadcasting of XML data over the stream. Consider energy conservation of mobile clients when disseminating data in the wireless mobile environment, because they use mobile devices with limited battery-power. structure indexing, lineage encoding, selective tuning algorithms can be used to minimize computation costs and filtering time.

Keywords*—lineage encoding, twig pattern queries*

*****

I. INTRODUCTION

A lightweight and effective encoding scheme, called Lineage Encoding is proposed, to support evaluation of predicates and twig pattern queries over the stream. To consider energy conservation of mobile clients when disseminating data in the wireless mobile environment, because they use mobile devices with limited battery-power (i.e., energy-efficiency).The overall query processing time must also be minimized to provide fast response to the users. The goals of conventional query processing on streamed XML data are to minimize computation costs and filtering time.

II. RELATED WORK

Several researches have been proposed to efficiently process XML twig pattern queries [2], [6].[2] investigated descendant nodes located at higher positions than their ancestor nodes in a stack. Based on this observation, they proposed efficient stack-based join algorithms. TwigStack [6] reduces the amount of the intermediate results and computational cost for merging the intermediate results using a chain of linked stacks that represent partial results to root-to-leaf query path. XR Twig [19] demonstrates superior performance because it skips elements that do not tally with given twig patterns, using an index-based algorithm. In [20], Jiang et al.

use two algorithms, a merge-based algorithm for sorted XML data and an index-based algorithm for indexed XML data, to enhance performance for matching twig queries with the OR-predicate. Kaushik et al.

A number of researches have been proposed to efficiently evaluate a large number of XML queries on a stream of XML documents [3], [7], [11], [15], [16]. XFilter [3] defines a Finite State Machine and converts each XPath query into a Finite State Machine representation. Y Filter [11] proposes a filtering approach based on Nondeterministic Finite Automata. It improves matching performance by processing common query paths only once. Lazy DFA [15] constructs a Deterministic Finite Automaton in a lazy manner, thus, the number of states are reduced. XPush [16] focuses on efficient evaluation of predicates. It avoids evaluation of redundant predicates by constructing a single deterministic pushdown automaton in a lazy manner. GFilter [7] addresses processing of the more complex Generalized Tree-Pattern queries. It achieves polynomial time and space complexity by avoiding redundant predicate evaluation. Conventional XML query processing methods mainly address the problem of efficiency and scalability. None of these approaches focuses on the energy-efficiency issue.Several approaches have been proposed for energy and latency efficient XML query processing in the wireless mobile environment [24]. S-node [24] generates an XML data stream based on the unit for XML broadcasting, called a S-node.

This approach constructs indices based on structural characteristics of an XML document (i.e., tag names and paths of elements). S-node enables mobile clients to skip irrelevant nodes using these indices, providing energy-efficiency. DIX [16] proposes a fully distributed index structure and a clustering strategy for streaming XML data. DIX preserves the location path of an element as a form of bit string, to start query processing without unnecessary waiting time. The performance of query processing is further improved by clustering DIX nodes of the same depth. Path Summary [17] and They improve the performance of XML query processing in terms of both access and tuning times

III. PROPOSED WORK

Figure 1 shows the architecture of the wireless XML broadcasting system in dynamic way. In wireless XML broadcasting, the broadcast server retrieves XML data to be disseminated from the XML repository. The XML stream is continuously disseminated via a broadcast channel. In the client-side, if a query is issued by the mobile client, the mobile client tunes in to the broadcast channel and selectively downloads the required data.In this paper, we use XPath as a query language. The results of an XPath query are selected by a location path. A location path consists of location steps. Processing each location step selects a set of nodes in the document tree that satisfy axis, node test and predicates described.

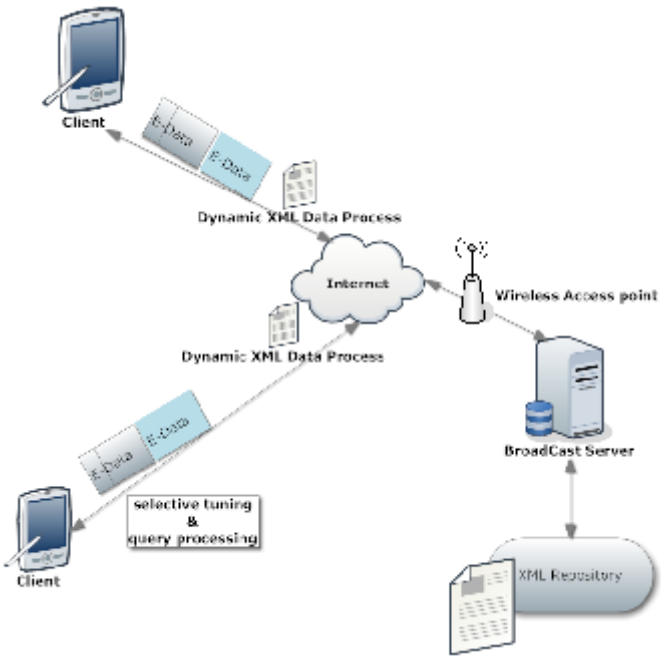


Figure 1: Architecture Of Dynamic Data Processing

A. XML Data & Manipulation

An XML document can be represented as a rooted, ordered, and labeled tree. Elements, attributes, and texts are represented by nodes, and the parent-child relationships are represented by edges in the XML tree. A server retrieves an XML document to be disseminated from the XML repository and it generates wireless XML steam by using SAX (Simple API for XML), which is an event-driven API. SAX invokes content handlers during the parsing of an XML document. Structured Indexing approach integrate multiple elements of the same path into one node, thus, the size of data stream can be reduced by eliminating redundant tag names thereby enabling Twig Pattern Query Processing.

This figure 2 shows the sample XML data and manipulation. The input of the XML file is converted into tree format by using SAX parser and finally produce output as structure indexing.

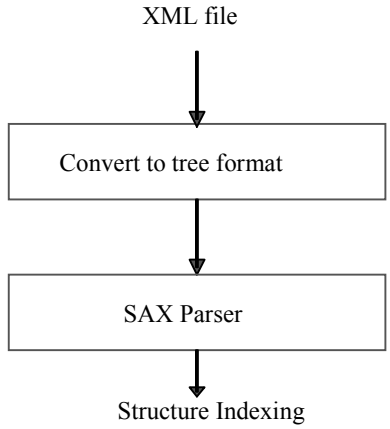


Figure 2: XML data and manipulation

B. Attribute Summarization

The Attribute Value List (AVL) generated in Attribute Summarization with lineage encoded data is the key to process the Twig Pattern Queries in Selective tuning approach in the mobile end .In XML, an element may have multiple attributes, each of which consists of a name and value pair, there is structural characteristic that elements with the same tag name and location path often contain the attributes of the same name. Attribute summarization eliminates repetitive attribute names in a set of elements when generating a stream of G-nodes.

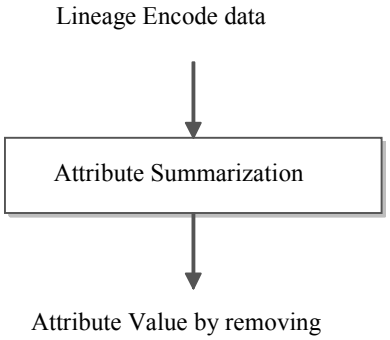


Figure 3: Attribute Summarization

C. G-Node & Xml Dissemination

The G-node structure eliminates structural overheads of XML documents, and enables mobile clients to skip downloading of irrelevant data during query processing.The group descriptor is a collection of indices for selective access of a wireless XML stream. Node name is the tag name of integrated elements, and Location path is an XPath expression of integrated elements from the root node to the element node

in the document tree. Child Index (CI) is a set of addresses that point to the starting positions of child G-nodes in the wireless XML stream. Attribute Index (AI) contains the pairs of attribute name and address to the starting position of the values of the attribute that are stored contiguously in Attribute Value List.

The components of the group descriptor are used to process XML queries in the mobile client efficiently. Specifically, Node name and Location path are used to identify G-nodes. Indices relating to time information such as CI, AI, and TI are used to selectively download the next G-nodes, attribute values, and text. Finally, Lineage Code(V, H) is used to handle axis and predicate conditions in the user's query & Attribute Value List (AVL store attribute values of the elements represented by the G-node, respectively .All the G-Node data's are Broadcasted with the help of a Wi-Fi device which can be received by any android devices in its coverage.

D. Tree Formation & Selective Tuning

Describes how a mobile client can retrieve the data of its interests. Assuming that there is no descendant axis in the user query, query processing algorithms for a simple path query and a twig pattern query are presented. Simple Path Query Processing, Algorithm shows the simple path query processing over the wireless XML stream. Given a query, the mobile client constructs a query tree. Then, it starts to find relevant G-nodes over the wireless XML stream.

The mobile client downloads a group descriptor of the G-node which corresponds to the query node. If the current node is the leaf node, the mobile client downloads. Twig Pattern Query Processing, query over the proposed wireless XML stream. In the Tree traversal phase, the mobile client first constructs a query tree. Then, traversing the query tree in a depth-first manner, it selectively downloads group descriptors of the relevant G-nodes into the nodes in the query tree. Our Selective tuning approach is dynamic and it eases the client to minimize the tuning time and thereby reducing access time also. It dynamically chooses between the Twig Pattern Query and Normal Query and process to render the data. Tuning is optimized with the help of the XPath Query pattern which holds the predicates.

E. Dynamic Data Processing

XML Automation tool is used for customized XML creation enables the server to Broadcast the customized data's as and when needed without relying on the third party for XML files and Implementation support to dynamic customized XML is a major advantage of the wireless streaming in mobile environment. Using XPath as a query language. The results of an XPath query are selected by a location path. A location path consists of location steps. Processing each location step selects a set of nodes in the document tree that satisfy axis, node test and predicates described. A GNode the novel attribute of our system can be added dynamically to the broadcast channel without interrupting the streaming of XML data. This feature enables to dynamically add events in the existing channel. Dynamic addition of GNode ensures the credibility of the Broadcast system efficiently proposed by our approach. AVL tree and Structured Indexing process will be handled that will probably affect the XML document in temporary buffer. Dynamic modification of Attribute value enables to change any data on the broadcast stream whenever needed and is achieved by the Attribute summarization mechanisms and the Structured Indexing of XML data handled in our system.

IV. LINEAGE ENCODING

We propose a novel encoding scheme, called Lineage Encoding, to support queries involving predicates and twig pattern matching. In the proposed scheme, two kinds of lineage codes, i.e., vertical code denoted by Lineage Code(V) and horizontal code denoted by Lineage Code(H), are used to represent parent-child relationships among XML elements in two G-nodes. Thus, the proposed Lineage Code scheme encodes parent-child relationships between two sets of elements in two Gnodes based on light-weight and efficient bit string representation.

Fig. 4 shows an example of Lineage Codes in G-nodecountry, G-nodeprovince, and G-nodecity. Note that Lineage Code(V) of G-node$_{province}$ is defined by 1,011 since the elements integrated in G-node$_{province}$ are mapped to only the first, third, and fourth elements in G-node$_{country}$. Lineage Code(H) of G-node$_{province}$ is (2, 2, 2), where each value denotes the number of child elements in G-node$_{province}$ mapped to the same parent element in G-node$_{country}$ in document order.

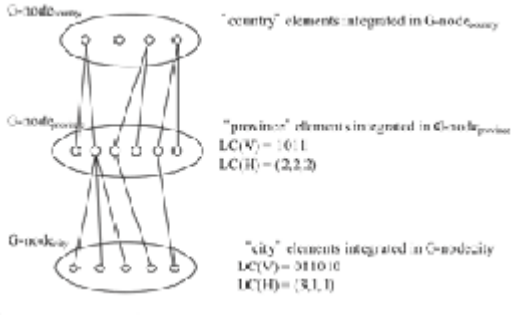


Figure 4.example of lineage encoding

V. TWIG PATTERN QUERY PROCESSING

Twig Pattern Query Processing, query over the proposed wireless XML stream. In the Tree traversal phase, the mobile client first constructs a query tree. Then, traversing the query

tree in a depth-first manner, it selectively downloads group descriptors of the relevant G-nodes into the nodes in the query tree.Twig pattern query processing consists of three phases: Tree traversal phase, Subpaths traversal phase, and Main path. Fig. 7 illustrates exploring steps for (a) twig pattern query with a predicate over the stream and (b) Lineage Codes computation using a query tree. Assume that a mobile client queries "find cities in Aland province ("/mondial/country/province[@name= "Aland"]/city")." The mobile client first constructs a query tree, and finds the relevant G-nodes over the wireless XML stream using CIs. After downloading the group descriptor of G-node$_{province}$, the mobile client selectively downloads the "name" attribute values pointed by an address contained in AI because the current query node contains a predicate condition (i.e., name ¼ "Aland"). After downloading of relevant Gnodes, the mobile client computes the result selection bit string by performing the SelectChildren() function from the root node to the leaf node. Note that the selection bit string of G-node$_{province}$ is 001000 because only the third attribute value in a value list satisfies the predicate. In this example, the mobile client retrieves the fourth element in G-node$_{city}$ because the result selection bit string for the G-node is 00010.

Algorithm 1. Twig Pattern Query Processing

Input: Wireless XML Stream DS, a twig pattern query Q
Output: Result set R satisfying Q

begin
01: result set R ¼ ;// initialization
02: Initialize the selection bit string SB as 1;
03: Initialize Lineage Code of the root G-node as (1, (1));
04: Initialize next node as the address of the root G-node in DS;
05:
06: // Tree traversal phase
07: Construct a query tree T for Q;
08: REPEAT {
09: Tune a group descriptor GD of the G-node indicated by nextNode;
10: IF (current node CN is the leaf node in TÞ THEN
11: Store AVL and TL the node in T;
12: ELSE
13: IF (CN contains predicate conditions PÞ THEN
14: Tune the relevant attribute values and/or text using AI and/or TI;
15: Store the relevant attribute values and/or text into the node in T;
16: END IF
17: Assing the address of the next node in CI to nextNode;
18: END IF
19: } UNTIL (all nodes in T are completely traversed)
20:
21: // subpaths traversal phase
22: Let N be the highest branching node in T;
23: SB$_N$ ¼ GetSelectionBitstringOfðNÞ;
24:
25: // Main path traversal phase
26: Let MP be the main path in T starting from N;
27: P ¼ N;
28: SB$_p$ ¼ SB$_N$;
29: REPEAT {
30: Let C be the child node of P in MP;
31: SB$_c$ ¼ SelectChildrenðC, SB$_p$Þ;
32: P ¼ C;
33: SB$_p$ ¼ SB$_c$;
34: } UNTIL ðC is the leaf node)
35: Select a set R of elements in C using the selection bit string SB$_c$; 36: Return R; end

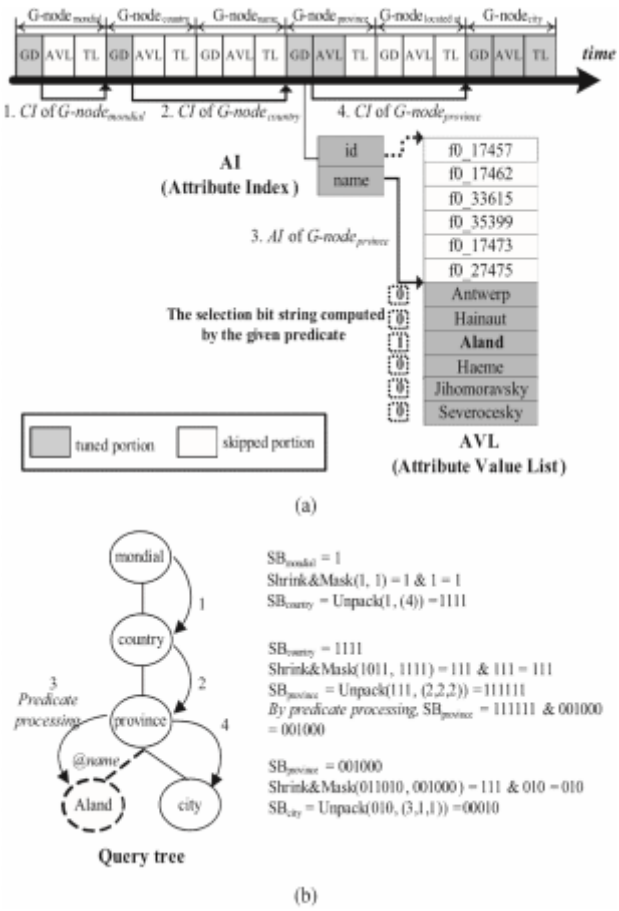


Figure 5 Example of twig pattern query processing. (a) Exploring steps over the wireless XML stream (b) The result selection bit string computation on a query tree.

VI . EXPERIMENTAL RESULTS

Experimental result shows the mobile clients retireve data by using structure indexing, lineage encoding, attribute summarization, twig pattern query processing, and selective tuning.

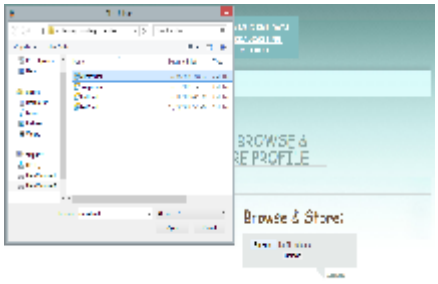
XML MANIPULATION

MAIN PAGE

STRUCTURE INDEXING

LINEAGE ENCODING

ATTRIBUTE SUMMARIZATION



VII. CONCLUSION

Proposed Lineage Encoding to support queries involving predicates and twig pattern matching. And also defined the relevant operators and functions to efficiently process twig pattern matching. The mobile client can retrieve the required data satisfying the given twig pattern by performing bit-wise operations on the Lineage Codes in the relevant G-nodes.And used a real XML data set and a synthetic data set for the correctness of experiments. Demonstrated our scheme is effective and efficient in terms of the access time and tuning time.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, “Broadcast Disks: Data Management for Asymmetric Communication Environments,” Proc. ACM SIGMOD Int’l Conf. Management of Data Conf., pp. 199-210, Mar. 1995.

[2] S. Al-Khalifa, H.V. Jagadish, N. Koudas, J.M. Patel, D. Srivastava, and Y. Wu, “Structural Joins: A Primitive for Efficient XML Query Pattern Matching,” Proc. Int’l Conf. Data Eng. (ICDE), pp. 141-152, Feb. 2002.

[3] M. Altinel and M. Franklin, “Efficient Filtering of XML Documents for Selective Dissemination of Information,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 53-64, 2000.

[4] S. Amer-Yahia, S. Cho, L.V.S. Lakshmanan, and D. Srivastava, “Minimization of Tree Pattern Queries,”

Proc. ACM SIGMOD Int'l Conf. Management of Data Conf., pp. 497-508, 2001.

[5] A. Berglund, S. Boag, D. Chamberlin, M.F. Fernandez, M. Kay, J. Robie, and J. Simeon, "XML Path Language (XPath) 2.0," Technical Report W3C, 2002.

[6] N. Bruno, D. Srivastava, and N. Koudas, "Holistic Twig Joins: Optimal XML Pattern Matching," Proc. ACM SIGMOD Int'l Conf. Management of Data Conf., pp. 310-321, 2002.

[7] S. Chen, H. Li, J. Tatemura, W. Hsiung, D. Agrawal, and K.S.

Candan, "Scalable Filtering of Multiple Generalizaed-Tree-Pattern Queries over XML Streams," IEEE Trans. Knowledge and Data Eng., vol. 20, no. 12, pp. 1627-1640, Dec. 2008.

[8] C. Chung, J. Min, and K. Shim, "APEX: An Adaptive Path Index for XML Data," Proc. ACM SIGMOD Int'l Conf. Management of Data Conf., June 2002.

[9] Y.D. Chung, S. Yoo, and M.H. Kim, "Energy- and Latency-

Efficient Processing of Full- Text Searches on a Wireless Broadcast Stream," IEEE Trans. Knowledge and Data Eng., vol. 22, no. 2, pp. 207-218, Feb. 2010.

[10] B. Cooper, N. Sample, M.J. Franklin, G.R. Hjaltason, and M. Shadmon, "A Fast Index for Semistructured Data," Proc. Int'l Conf. Very Large Data Bases (VLDB), Jan. 2001.

[11] Y. Diao, M. Altinel, M. Franklin, H. Zhang, and P.M. Fischer, "Path Sharing and Predicate Evaluation for High-Performance XML Filtering," ACM Trans. Database Systems, vol. 28, no. 4, pp. 467-516, 2003.

[12] D. Florescu and D. Kossmann, "Storing and Querying XML Data Using an RDBMS," IEEE Data Eng. Bull., vol. 22, no. 3, pp. 27-34, Mar. 1999.

[13] M. Francechet, "XPathMark: An XPath Benchmark for XMark Generated Data," Proc. Third Int'l Conf. Database and XML Technologies (XSYM), 2005.

[14] R. Goldman and J. Widom, "DataGuides: Enable Query Formulation and Optimization in Semistructured Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 436-445, 1997.

[15] T.J. Green, A. Gupta, G. Miklau, M. Onizuka, and D. Suciu, "Processing XML Streams with Deterministic Automata and Stream Index," ACM Trans. Database Systems, vol. 29, no. 4, pp. 752-788, 2004.

[16] A. Gupta and D. Suciu, "Stream Processing of XPath Queries with Predicates," Proc. ACM SIGMOD Int'l Management of Data Conf., pp. 419-430, 2003.

[17] T. Imielinski, S. Viswanathan, and B. Badrinath, "Energy Efficient Indexing on Air," Proc. ACM SIGMOD Int'l Conf. Management of Data Conf., pp. 25-36, 1994.

[18] T. Imielinski, S. Viswanathan, and B. Badrinath, "Data on Air: Organization and Access," IEEE Trans. Knowledge and Data Eng., vol. 9, no. 3, pp. 353-372, June 1997.

[19] H. Jiang, W. Wang, H. Lu, and J. Yu, "Holistic Twig Joins on Indexed XML Documents," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 273-284, 2003.

[20] H. Jiang, H. Lu, and W. Wang, "Efficient Processing of XML Twig Queries with OR-Predicates," Proc. ACM SIGMOD Int'l Management of Data Conf., pp. 59-70, June 2004.

[21] R. Kaushik, P. Bohannon, J.F. Naughton, and H.F. Korth, "Covering Indexes for Branching Path Queries," Proc. ACM SIGMOD Int'l Management of Data Conf., pp. 133-144, June 2002.

[22] R. Kaushik, R. Krishnamurthy, J.F. Naughton, and R. Ramakrishnan, "On the Integration of Structure Indexes and Inverted Lists," Proc. ACM SIGMOD Int'l Management of Data Conf., June 2004.

[23] Nokia N Series, http//www.nseries.com, 2009.

[24] C.-S. Park, C.S. Kim, and Y.D. Chung, "Efficient Stream Organization for Wireless Broadcasting of Xml Data," Proc. Int'l Conf. Asian Computing Science Conf., pp. 223-235, 2005.

[25] J.P. Park, C.-S. Park, and Y.D. Chung, "Attribute Summarization: A Technique for Wireless XML Streaming," Proc. Int'l Conf. Interaction Sciences, pp. 492-496, Dec. 2009.