Huo, H., Wang, G., & Hui, X., et al. (2006). Efficient query processing for streamed XML fragments.

Originally published in M. L. Lee, K. L. Tan, & V. Wuwongseet (eds.). *Proceedings of the 11th International Conference on Database Systems for Advanced Applications (DASFAA 2006), Singapore, 12–15 April 2006.*
Lecture notes in computer science (Vol. 3882, pp. 468–482). Berlin: Springer.

Available from: http://dx.doi.org/10.1007/11733836_33

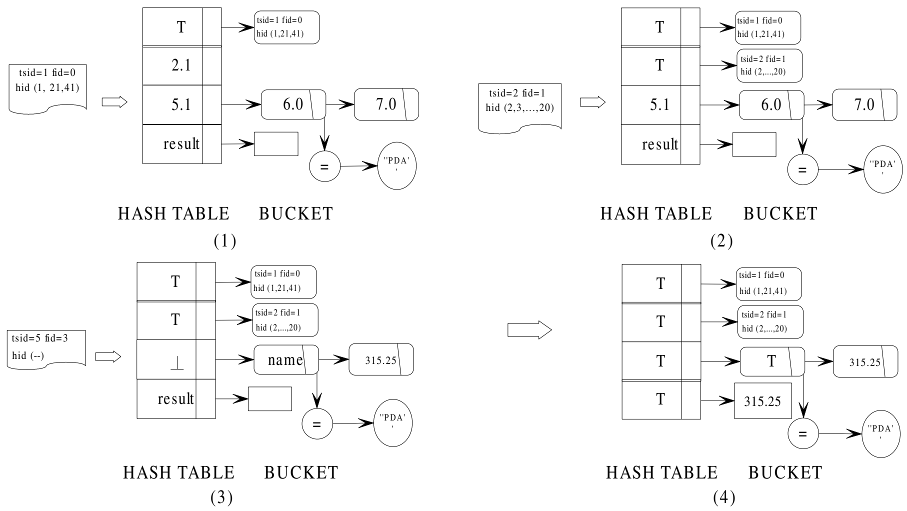

Fig. 10. XFPro Processing Example

```
Algorithm1 FindQueryChild()
 {Input an element node and trigger descendant operators}
  IF (isHashTerminalNode(element)) THEN output element;
   ELSE
    q <- HashBucketFirstnode(QueryNextnode(element));
    WHILE(q!=null)DO
     IF (q.fid==elemnet.fid)
      THEN  q.val=element.val; FindQueryChild(q); q=q.next;
      ELSE
       FOR(p=element.hid;p!=null&&p.hid!=q.fid;p=p.next);
         IF (p.hid==q.fid)
          THEN q.val= element.val; FindQueryChild(q);q = q.next;
         END IF END FOR
     END IF
    END WHILE
  END IF
```

Algorithm 1 and 2 change the corresponding values of the hash table to schedule triggering the descendant operator and inquiring the parent operator.

```
Algorithm2 FindQueryParent()
 {Input an element node and inquire parent operator}
  IF (HashQueryFirstnode(element)) THEN element.val=TRUE;
   ELSE
    q <- HashBucketFirstnode(QueryPrenode(element));
    WHILE(q!=null)DO
```

```
      FOR ( ; q!= null;q = q.next )
       IF (q.fid==element.fid) THEN element.val = q.val;
        ELSE
         FOR (phid=q.hid; phid!= null; phid = phid.next);
          IF(p.hid==element.fid) THEN element.val = q.value;
          END IF  END FOR
       END IF  END FOR
    END WHILE
    element.val=UNDECIDED;
  END IF
```

5 Performance Evaluation

We have implemented the XFPro translator engine in Java, which rewrites XPath expressions into tid-tree based query plans for XML fragments. Our XF-Pro query engine on fragmented XML streams processes the optimized queries directly on the filler fragments before reconstructing the entire XML document.

All experiments are run on a PC with 2.6GHz CPU, 512M of memory and 80G hard disk. The operating system is WindowsXP. The experiments are run on data sets generated by the xmlgen program. We have written an XML fragmenter that fragments an XML document into filler fragments to produce an XML stream, based on the tag structure defining the fragmentation layout.

We have selected three representative queries (Q_1,Q_2 and Q_3) on the generated XML documents and compared the results with the XFrag Processor [4].

```
Query1:doc("book.xml")/book/sections/section/subsection/title
Query2:doc("book.xml")/book/section[difficulty>="default"]/title
Query3:doc("book.xml")/book/title/section[difficulty>="default"]
```

To illustrate the differences in the query execution methods on the filler fragments, consider the *Query 1* that returns the subsection title of the books. Since “section”, “subsection” and “title” are in common filler fragments, according to the fragmentation information in tag structure, our query operates “subsection” and “title” over fragment only when the fragment tsid matches that of the operator. Furthermore, each fragment is only evaluated once and hashed to corresponding item if tsid matches. While in XFrag, each fragment needs to be passed on through the pipeline and evaluated step by step. In this way, our method performs better than XFrag. The results of the experiments are summarized in Figure 11.

From the experimental results, we observe that the XFPro method outperforms the XFrag method mainly on running time, while the memory cost of these two methods makes little difference. That is because both of the methods adopt the policy of keeping the output-related information of the fragments while hash buckets use less links than association table. For the query processing time, the XFPro method saves CPU time by avoiding subsumption operations. Furthermore, the XFrag method has to schedule the operations for each fragment, while the XFPro only changes the corresponding value of the hash table.

Query	File size	Fragmented File Size	Method	Run time	memory
Q1	10Mb	11.04Mb	XFPro	518.27ms	0.36Mb
			XFrag	1875.00ms	0.62Mb
	15Mb	17.56Mb	XFPro	1377.05ms	0.81Mb
			XFrag	3926.50ms	1.35Mb
	20Mb	23.18Mb	XFPro	2121.59ms	1.18Mb
			XFrag	5245.56ms	1.83Mb
Q2	10Mb	11.98Mb	XFPro	3015.92ms	1.87Mb
			XFrag	7329.70ms	2.13Mb
	15Mb	19.20Mb	XFPro	4585.60ms	5.39Mb
			XFrag	11444.55ms	6.95Mb
	20Mb	24.12Mb	XFPro	6727.93ms	6.78Mb
			XFrag	15259.40ms	9.83Mb
Q3	10Mb	11.78Mb	XFPro	3005.86ms	2.08Mb
			XFrag	7239.07ms	2.03Mb
	15Mb	19.38Mb	XFPro	4550.15ms	5.01Mb
			XFrag	11429.71ms	6.64Mb
	20Mb	24.33Mb	XFPro	6674.87ms	6.73Mb
			XFrag	15154.78ms	8.86Mb

Fig. 11. Experimental Results

6 Conclusions

This paper has presented a framework and a set of techniques for processing XPath queries over streamed XML fragments. We present techniques for enabling the transformation from XPath expression to optimized query plan. Our query model of tid tree helps to transform queries on element nodes to queries on XML fragments and analyze "redundant" operations in them. Furthermore, such transformations specify query operations such as "//" and "*" and reduce the query workload. Based on optimized tid tree, we present a scheme to map a tid tree directly into an XML fragment query processor, and thus efficient query execution plan is generated. Our experiments show that our framework performs well on saving processing power and memory space.

Acknowledgments This research was partially supported by the National Natural Science Foundation of China (Grant No. 60473074 and 60573089) and Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP).

References

1. W3C Recommendation: Extensible Markup Language (XML) 1.0 (Second Edition). (2000) http://www.w3.org/TR/REC-xml.
2. W3C Working Draft: XML Path Languages (XPath), ver 2.0. (2001) Tech. Report WD-xpath20-20011220, W3C, 2001, http://www.w3.org/TR/WD-xpath20-20011220.
3. W3C working draft: XQuery 1.0: An XML Query Language. (2001) Technical Report WD-xquery-20010607, World Wide Web Consortium.

4. Bose, S., Fegaras, L.: XFrag: A query processing framework for fragmented XML data. In: Eighth International Workshop on the Web and Databases (WebDB 2005), Baltimore, Maryland (June 16–17,2005)
5. Bose, S., Fegaras, L., Levine, D., Chaluvadi, V.: A query algebra for fragmented XML stream data. In: Proceedings of the 9th International Conference on Data Base Programming Languages(DBPL 2003), Potsdan, Germany (September 6–8, 2003)
6. Fegaras, L., Levine, D., Bose, S., Chaluvadi, V.: Query processing of streamed XML data. In: Eleventh International Conference on Information and Knowledge Management (CIKM 2002), McLean, Virginia, USA (November 4–9, 2002)
7. Chen, J., J.DeWitt, D., Tian, F., Wang, Y.: NiagaraCQ: A scalable continuous query system for internet databases. In Chen, W., Naughton, J.F., Bernstein, P.A., eds.: SIGMOD Conference, Dallas, Texas, USA, ACM (2000) 379–390
8. Florescu, D., Hillery, C., Kossmann, D., Lucas, P., Riccardi, F., Westmann, T., Carey, M.J., Sundararajan, A.: The BEA/XQRL streaming xquery processor. In Freytag, J.C., Lockemann, P.C., Abiteboul, S., Carey, M.J., Selinger, P.G., Heuer, A., eds.: Proceedings of the 29th International Conference on Very Large Data Bases, Berlin, Germany (2003)
9. Koch, C., Scherzinger, S., Schweikardt, N., Stegmaier, B.: FluXQuery: An optimizing xquery processor for streaming XML data. [16] 1309–1312
10. Ludäscher, B., Mukhopadhyay, P., Papakonstantinou, Y.: A transducer-based XML query processor. In Bernstein, P.A., Ioannidis, Y.E., Ramakrishnan, R., Papadias, D., eds.: Proceedings of the 28th International Conference on Very Large Data Bases, Hong Kong SAR, China (2002) 227–238
11. Ludäscher, B., Papakonstantinou, Y., Velikhov, P.: Navigation-driven evaluation of virtual mediated views. In: Proceedings of the 7th International Conference on Extending Data Base Technology(EDBT 2000), Konstanz, Germany (March 27–31, 2000) 150–165
12. Wang, E., et al.: Efficient management of XML contents over wireless environment by Xstream. In: ACM-SAC 2004. (March, 2004) 1122–1127
13. Abiteboul, S., Benjelloun, O., Cautis, B., Manolescu, I., Milo, T., Preda, N.: Lazy evaluation for active XML. In Weikum, G., König, A.C., Deßloch, S., eds.: SIGMOD Conference, Paris, France, ACM (2004) 227–238
14. Huo, H., Hui, X., Wang, G.: Document fragmentation for XML streams based on hole-filler model. In: 2005 China National Computer Conference, Wu Han, China (October 13–15,2005)
15. Bar-Yossef, Z., Fontoura, M., Josifovski, V.: On the memory requirements of xpath evaluation over XML streams. [16]
16. Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B., eds.: (e)Proceedings of the Thirtieth International Conference on Very Large Data Bases. In Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B., eds.: Proceedings of the 30th International Conference on Very Large Data Bases, Toronto, Canada, Morgan Kaufmann (2004)