XFLab: A Technique of Query Processing over XML Fragment Stream*

Sangwook Lee, Jin Kim, and Hyunchul Kang

School of Computer Science and Engineering, Chung-Ang University
Seoul, 156-756, Korea
{swlee,jkim}@dblab.cse.cau.ac.kr, hckang@cau.ac.kr

Abstract. We investigate XML query processing in a portable/handheld client device with limited memory in ubiquitous computing environment. Because of memory limitation in the client, the source XML data possibly of large volume is fragmented in the server and streamed in fragments over which query processing is done in the client. The state-of-the-art techniques employ the *hole-filler model* in fragmenting XML data and processing queries over XML fragment stream. In this paper, we propose a new technique where an *XML labeling scheme* is employed instead of the hole-filler model. Through preliminary experiments, we show that our technique outperforms the state-of-the-art techniques both in memory usage and in query processing time.

1 Introduction

In ubiquitous computing environment, there could be a number of heterogeneous portable/handheld client devices deployed. Those include cellular phones, PDAs, and smart cards to name just a few. Despite the rapid advancement of memory technology, it is still usual that memory capacity of such devices is very limited. As such, naïve practice would be that the client sends its query to the server when its source data is of large volume, and the server processes it and ships the result. Such a conventional client-server computing is obviously not scalable.

In this paper, we investigate XML query processing in a client device with limited memory. The source XML data against which the client queries are to be processed could be in large volume. Thus, the whole of it could not be downloaded to the client. Besides, in ubiquitous computing environments, the server data is usually transmitted over a wireless network with limited bandwidth. As such, it is infeasible to transmit a large XML data in its entirety. Rather, it is partitioned into manageable fragments, and they are transmitted as a stream. Their arrival at a client may not be in proper order, and yet the stream query processing over them should return the correct result.

The first proposed system with such capability is XFrag [3]. It employs the *hole-filler model* [1,2] in fragmenting XML data and *pipelined* query processing over XML fragment stream. The hole-filler model is simple and clean in representing

* This work was supported by the Basic Research Program of the Korea Science & Engineering Foundation (grant No. R01-2006-000-10609-0).

R. Cooper and J. Kennedy (Eds.): BNCOD 2007, LNCS 4587, pp. 185–189, 2007.

XML fragmentation. Each XML fragment could contain *holes*, which are supposed to be filled with other XML fragments possibly with other holes. For each hole and its corresponding *filler*, a value is assigned. It is called a hole ID (for the hole) or a filler ID (for the filler). The main inefficiency inherent in the hole-filler model is two-fold: (1) The space overhead for the hole/filler IDs along with accompanying XML tagging created in XML fragmentation process could be very huge considering the typical structure of XML documents on the Web [4]. (2) Given two XML fragments that are an ancestor and a descendant with each other, such a structural relationship could not be identified until all the fragments connecting the two are fully streamed in. Because of this, the processing of the widely used descendant axis (//) of XPath would suffer. Due to these limitations, the amount of memory required for query processing over XML fragment stream at the client would increase.

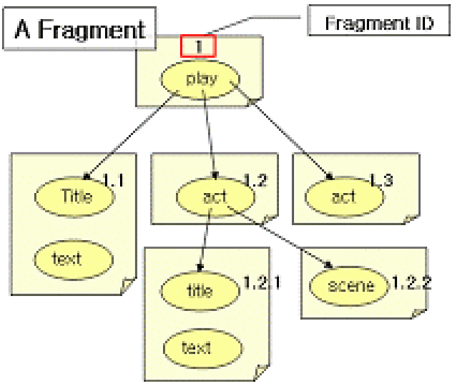

Fig. 1. Example of XML Fragmentation with Labeling

XFPro [5] has improved the query processing pipeline of XFrag. However, it is still based on the hole-filler model, and its improvement is focused on query processing time.

In this paper, we report our on-going efforts for the development of a new technique called *XFLab* where an *XML labeling scheme* replaces the hole-filler model to improve memory efficiency. Section 2 summarizes the salient features of XFLab. Section 3 presents preliminary performance results.

2 XFLab

The XML labeling (or numbering) schemes were devised to represent structural relationship (e.g., parent-child, ancestor-descendant, etc.) among the nodes of XML data modeled as a tree, and to exploit them in query processing. There were many proposed in the literature and most recent and advanced ones include ORDPATH [6] and QED [7]. When an XML document modeled as a tree is fragmented, it could also be represented as a tree, which we call an *XML fragment tree*. Thus, its fragments could be labeled as the nodes of the original XML tree are. Figure 1 shows an example of an XML fragment tree and its labeling with Dewey order encoding.

With our XML fragment labeling, we do not need holes or fillers. Thus, there are no hole/filler IDs. Only the *fragment ID*s (i.e., the labels assigned to fragments) will do. For an XML fragment, there would be only 1 fragment ID needed in our scheme while with the hole-filler model as many hole IDs as the number of holes in a fragment are needed. For a typical structure of XML documents [4], there would be a number of holes in a fragment. As such, our XML fragment labeling considerably reduces the space overhead incurred because of fragmentation. Besides, given any