PosFilter: An Efficient Filtering Technique of XML Documents Based on Postfix Sharing

Jaehoon Kim[1], Youngsoo Kim[2], and Seog Park[1]

[1] Department of Computer Science, Sogang University
1-1 Shinsu-Dong Mapo-Gu Seoul Korea 121-742
jhkimygk@gmail.com, spark@dblab.sogang.ac.kr
[2] Samsung Electronics CO. LTD.,
Maetan-3-Dong Yeongtong-gu Suwon Kyungki-do Korea 443-742
Youngsoo.kim@samsung.com

Abstract. XML message filtering is to evaluate the path matching of a large number of registered path queries over a continuous stream of XML messages in real time. For this purpose, YFilter system has been suggested to exploit the prefix commonalities that exist among path expressions. Sharing such commonality gives the benefit of improving filtering performance through the tremendous reduction in filtering machine size. However, postfix sharing also can be useful for an XML filtering situation. For example, if a stream of XML messages does not have any defined DTD (or XML schema), the XPath queries beginning with the ancestor-descendant axis ('//') can be used often, e.g., '//buyer/name', '//seller/name', and '//name', and such query type is most likely to have the postfix sharing. Therefore, in this paper, we propose a bottom up filtering approach exploiting postfix sharing against the top down approach of YFilter exploiting prefix sharing. Some experimental results show that our method has better performance in the postfix-shared scenario.

1 Introduction

RSS (Really Simple Syndication) is defined as a service for a web content syndication [3]. The current architecture of RSS dissemination is based on a pulling scheme, where the RSS reader of each user periodically pulls RSS files from a contents server and checks the renewal status. However, this architecture tends to lose scalability when the publication of RSS files suddenly explodes. In addition, the renewal status cannot be immediately alerted and unnecessary pulling operation makes the RSS feeder and reader waste resources [2, 9]. In order to resolve this problem, efforts to apply a publish/subscribe scheme to RSS system have been made [7]. In the publish/subscribe scheme, users subscribe their profile to the system and data, generated by a publisher, which matches the registered profile, are delivered to the interested users [8].

Recently, XML message filtering systems have been studied to support the publish/subscribe scheme: XFilter [6], YFilter [11], XMLTK [10], AFilter [5], etc. In such systems, a user profile is represented as XPath query language [12] and a given

R. Cooper and J. Kennedy (Eds.): BNCOD 2007, LNCS 4587, pp. 70–81, 2007.

set of registered XPath queries is continuously evaluated over XML message streams in real time. Among researches above, YFilter has been especially proposed to exploit commonality that exists among path expressions. The commonality is for prefixes sharing. For example, for the registered queries '/a/b', '/a/b/c', and '/a/b/d', the partial path expression '/a/b' is shared. YFilter combines all XPath queries into a single *Non-deterministic Finite Automaton* (NFA) to exploit such prefix commonality. This approach brings about the tremendous reduction in filtering machine size and as a result, it gives the benefits of higher filtering performance and scalability. However, YFilter does not consider postfix sharing, e.g., the path expressions '//b/c' and '//c' are shared for the queries '/a/b/c', '//b/c', and '//c'. Our experimental results showed that if the postfix-shared query pattern appears more often in the total query set, the throughput of YFilter degrades. Thus, in this paper, we introduce a novel bottom-up filtering approach exploiting the common postfixes in the NFA-based scheme, opposed to the top-down approach of YFilter. We name the new method as PosFilter. The key idea of our method is that the state transition in the NFA execution is performed at the end-of-element event to exploit the common postfixes of XPath queries. However, the execution of YFilter is performed at the start-of-element event.

Similar to our concept, AFilter [5] has been recently proposed for the postfix sharing. Moreover, the approach was intended to benefit from prefix commonalities, while simultaneously leveraging postfix commonalities. However, in conclusion, we think that they fail to suggest the adaptable filtering. Because in the reference [5], they do not show how it can be defined the concrete threshold for the unfolding condition (i.e., the switch from postfix sharing to prefix sharing) and how deep is the overhead of calculating the threshold, which is able to significantly affect the overall filtering performance. In addition, AFilter is not an automata-based scheme. It uses its own specific memory organization and a path matching algorithm. Hence, although the scheme of AFilter can give further improvement of path matching speed, we think that the substantial benefits of expressiveness and incremental maintenance provided by the NFA model outweigh the speed improvement as mentioned in the reference [11].

This paper is organized as follows. In Section 2, we review other researches related to our method. Section 3 presents the basic idea of this paper, and Section 4 describes the proposed PosFilter technique. Section 5 presents some experimental results for the throughput comparison of PosFilter, YFilter and AFilter. Section 6 finally concludes this paper.

2 Related Work

XFilter system [6] may be an early study on an automata-based XML filtering. The research point of the system is to use Finite State Machine (FSM) and an inverted index on all the subscribed XPath queries. XFilter converts each XPath query to a FSM and an inverted index (called Query Index) is built over the states of all FSMs. When a start-of-element event is triggered, the related SAX event handler looks up the element name in the Query Index and it is checked whether there are matched queries. Using the Query Index gives the benefit of achieving high performance filtering. The features of XFilter system are the simple construction and maintenance of the filtering machine and the assurance of high performance filtering and scalability.