Piggyback Optimization of XML Data Dissemination

Chee-Yong Chan
National University of Singapore
chancy@comp.nus.edu.sg

Yuan Ni
National University of Singapore
niyuan@comp.nus.edu.sg

1 Introduction

The ubiquity of XML data and the effectiveness of the content-based pub/sub-based paradigm of delivering relevant information has led to a lot of interest in content-based dissemination of XML data. In the pub/sub environment, an overlay network of application-level *routers* (or message brokers) is used to asynchronously forward documents generated by *data publishers* to relevant *data subscribers* (or consumers) based on matching the document contents against the consumers' subscriptions. Each subscriber first needs to pre-register a *subscription*, specifying the type of data that he is interested in receiving, to its local router. A routing protocol is used to set up a *routing table* at each router to record both its local subscriptions as well as the subscriptions from its neighboring routers. For each document received by a router, the router compares the document contents against the subscriptions in its routing table and forwards the document to the matching local subscribers and neighboring routers.

Content-based data dissemination has traditionally been examined in the context of simple subscription specifications, such as matching of keywords or conjunction of basic comparison predicates on attribute values. The recent interest in using the more expressive XPath-based subscriptions for effective dissemination of XML documents has increased the complexity of the content-based subscription matching problem. Thus, there is a even greater need for effective optimization techniques to meet the performance challenges of content-based dissemination of XML data. The existing research efforts have focused on two key optimizations to minimize the number of subscription matchings. The first optimization, which has attracted the most attention, is to exploit efficient index structures (e.g., [2]) to perform selective matching with only a small subset of potentially matching subscriptions. The second optimization uses aggregation algorithms to summarize an initial set of subscriptions into a smaller set of generalized subscriptions (based on subscription containment properties) to reduce the matching overhead [1].

The effectiveness of the existing optimizations are, however, limited to only locally improving the performance of the individual routers. Specifically, the fact that routers are interconnected and related (in terms of the containment relationships of their subscriptions) are not being exploited for more aggressive optimization. In this paper, we propose an orthogonal and more holistic optimization that enables a router to leverage the subscription matching done by previous routers to optimize its own subscription processing. To facilitate such "collaborative" processing, a router is allowed to piggyback some additional useful information (that is derived from its subscription processing) as header annotations to an XML document before forwarding it to a matching router. On receiving an annotated document, a router first pre-processes the header annotations to optimize its own subscription matching. We refer to this approach to optimize performance as *piggyback optimization*.

2 Motivation

Our approach is motivated by the following two observations on content-based data dissemination. First, the overall processing being done at the different routers during the dissemination of a document can be viewed as essentially processing the same data (i.e., XML document) against a sequence of collections of queries (i.e., sets of subscriptions along each path of forwarding routers). Second, the sequence of collections of queries being processed are not independent as they are partially related by the containment property; in particular, the subscriptions at a downstream router are aggregated to a set of subscriptions at each of its upstream routers. Thus, given that the same data is being processed against related sets of queries, we can actually try to leverage some of the subscription matching work done at an upstream router to minimize the processing work of its downstream router and thereby reducing the overall time to deliver a document to relevant subscribers.

To facilitate such "collaborative" processing, an upstream router is allowed to piggyback some additional useful information (that is derived from its subscription processing) as *header annotations* to the XML document being processed before forwarding it to its downstream router. On receiving an annotated XML document, a router first preprocesses the header annotations to optimize its own subscription matching. Our technique, which we termed *piggyback optimization*, is orthogonal to the two existing optimizations, namely, indexing techniques and subscription aggregation algorithms, that are also targeted at reducing the number subscriptions to be matched.

3 Our Approach

The key idea of our piggyback optimization is for an upstream router R_i to help optimize the performance of its downstream router R_j by passing some useful information (in the form of header annotations) along with the document D that R_i is forwarding to R_j. Our design of the annotated information is guided by the following three performance-related requirements. Firstly, it should be concise so that it incurs minimal processing overhead in terms of parsing and transmitting the additional header information. Secondly, it should be efficiently generated so that the computation overhead incurred by the upstream router does not offset any performance gains of its downstream routers. And thirdly, it should be effective in that a downstream router can efficiently preprocess the annotations to optimize its subscription matching performance.

Let us consider the possible sources of useful information that R_i can pass on to R_j. After having matched its own subscriptions against D, R_i has acquired additional information about D and how its subscriptions are related to D. We can classify this knowledge into *positive* and *negative* information: (1) *Positive information* refers to information about (a) subscriptions in that matched D, and (b) patterns / properties that occur in D. (2) *Negative information* refers to information about (a) subscriptions that did not match D, and (b) patterns/properties that did not occur in D.

Based on the above classification, our piggyback optimization utilizes four types of annotations: positive subscription (PS), positive data (PD), negative subscription (NS), and negative data (ND).

PS and NS annotations exploit the relationships between the subscriptions in R_i and R_j. Specifically, in order for R_i to be able to forward relevant data to R_j, the set of subscriptions S_j in R_j are actually aggregated (i.e., summarized) into a smaller and more general set of subscriptions S'_j that is installed in R_i such that R_i will forward a document D to R_j if and only if D matches some subscription in S'_j. Each subscription $s' \in S'_j$ is an aggregation of a subset of subscriptions in S_j. We refer to s' as an *aggregated subscription* and a subscription $s \in S_j$ that is aggregated to s' as an *aggregating subscription*. An aggregated subscription s' that can be transformed to one of its aggregating subscription by substituting some of the wildcards (i.e., $*$ and $//$) in s' with data element paths is called a *simple aggregated subscription*.

In the following, we elaborate on the four types of annotations.

Positive subscription (PS). A PS annotation is of the form (s', B) which specifies an aggregated subscription s' that matches D along with the data element bindings B (obtained from D) for the wildcards in s'. Thus (s', B) essentially represents some data pattern that occurs in D. This information can benefit R_j as follows: if there is an aggregating subscription s in R_j that matches the data pattern represented by (s', B), then this annotation enables R_j to know that D matches s without having to process D; thus, R_j can immediately forward D to the downstream router that is associated with s.

Positive data (PD). A PD annotation is of the form (p, ℓ) where p is some data pattern that occurs in D and ℓ is the position of the last occurrence of p in D. This information can be exploited by R_j as follows: once R_j has processed the portion of D beyond ℓ, R_j can optimize its subscription matching by ignoring all its subscriptions that contain p as these subscriptions are guaranteed not to match the remaining portion of D.

Negative subscription (NS). A NS annotation specifies an aggregated subscription s' that did not match D. R_j can exploit this information to optimize its subscription matching by ignoring the subset of subscriptions that are aggregated to s'.

Negative data (ND). A ND annotation specifies some data pattern p that did not occur in D. Thus, R_j can use p to optimize its subscription matching by ignoring all the subscriptions that contain p.

4 Conclusions

In this paper, we have proposed a novel approach to optimize the performance of content-based dissemination of XML data by piggybacking useful annotations to the document being forwarded so that a downstream router can leverage the processing done by its upstream router to reduce its own processing overhead. The piggyback optimization approach outperforms the conventional method by a factor of 2.

References

[1] C.-Y. Chan, W. Fan, P. Felber, M. Garofalakis, and R. Rastogi. Tree pattern aggregation for scalable XML data dissemination. In *VLDB*, 2002.

[2] C.-Y. Chan, P. Felber, M. Garofalakis, and R. Rastogi. Efficient filtering of XML documents with XPath expressions. *VLDB*, 11(4), 2002.