ABSTRACT

Recent multimedia digital library where various media have to be managed in distributed environments, metadata play an important role. XQuery, a standard language for querying XML, is expected to integrate multimedia contents through metadata written in XML. In this type of query, performance is significantly affected by external document references over network which are currently implemented by HTTP. In this paper, reduction of this HTTP traffic by analyzing XML query and detecting multiple identical reference to the same documents at various expression level, and rewriting that query to equivalent XQuery that incurs minimal HTTP access. This optimization shall maintain scalability of distributed digital library for multimedia contents at adequate level. Preliminary performance evaluation conducted using one of the W3C Use Cases shows that proposed transformation can reduce query execution time by several orders of magnitude for large data sets.

1. INTRODUCTION

Recent digital library integrates various media in distributed environments. Metadata play an important role in management of multimedia contents in DL. By using metadata, it becomes possible not only to query in semantic level to MM contents, but also to personalize MM contents in DL when providing them to users. Another merit especially in distributed environments of using metadata is in usage of network resources. Because ample bandwidth in network cannot be assumed, querying to MM contents in distributed environment cannot always access the contents themselves directly.

Ongoing standardization efforts for metadata for MM contents such as MPEG-7[8] are emerging. Metadata in MPEG-7 are described in XML, and we can see other example that XML are utilized in describing metadata such as Topic Map[9] and Resource Description Format[10] specification.

By using XML in describing metadata in distributed DL, it becomes possible not only to enjoy various techniques developed on the Web, but also to query to metadata in uniform manner such as XQuery[1].

XQuery, the standard language for querying XML is also at the state of working draft at the World Wide Web Consortium. XQuery is expected to realize integration of MM contents through metadata because XQuery is going to be used to query all applications of XML documents including MM metadata for DL. In this paper, we consider XQuery as a query language in distributed DL.

When XQuery is fully incorporated in DL especially in distributed environment, query expression that implements remote media reference is expected to appear frequently as Use Cases at W3C[2] indicates. For example, query that uses remote media reference within FOR expression leads to HTTP accesses in TCP/IP network proportional to the number of total FOR loop counts. Figure 1 shows an example of such query. Nesting of this kind of loop makes this situation worth.

This drawback greatly damage scalability of distributed DL because as the number of metadata increases along with the number of MM contents across distributed DL, above mentioned loop grows substantially, which leads to flood of HTTP traffic.

Our proposal is to eliminate this HTTP traffic by analyzing XML query for MM contents within XQuery processor and detecting multiple identical references to the same documents at various expression levels, and rewrite that query to equivalent XQuery that incurs minimal HTTP access. This optimization shall maintain scalability of distributed DL for MM contents at adequate level.

The rest of this paper is organized as follows: redundant communication elimination optimization for XQuery is proposed in section 2 and preliminary performance evaluation is conducted in section 3. In section 4 we refer to other attempts related to proposed optimization. Section 5 concludes this paper along with future works.
2. QUERY OPTIMIZATION METHOD

2.1 Basic Idea

Basic idea which lies under proposed optimization itself, so called Common Subexpression Elimination, is one of traditional compiler optimization technologies.

Major difference from traditional environments comes from drastic reduction in execution costs.

Within compiler generated code in traditional centralized environment, difference of performance between optimized and unoptimized code are not so large, because cost of recomputation only entails local memory reference.

Enormous difference emerges however, when that optimization is applied to XQuery expressions for distributed MM DL, because remote media reference is affected by bandwidth and communication delay of computer network.

Bandwidth of local memory access is typically 1GBytes per second for PC. On the other hand, based on authors' experience, general distant Internet access environment can not expect more than 10MB/s. To the best of authors' knowledge, there are no preceding research that applies this optimization to XQuery language. This optimization can also significantly improve performance of local computation, because as preliminary performance evaluation conducted in this paper indicates that thrashing can be seen in client computer due to too much network interface activity. Furthermore, less traffic due to our optimization, load on the DL servers' side are reduced, and consequently they can accommodate more requests.

2.2 Transformation Rule

As XQuery is a functional language, there is no need to take side-effects into account. Consequently, rewriting is expected to be simple compared to other languages with side-effects [3].

**Input:** exp such that

\[ e_1 \subseteq exp, \quad e_2 \subseteq exp, \quad e_1 \text{equiv} e_2 \]

where equiv. means \( e_1 \) and \( e_2 \) are expected to be evaluated to the same value.

**Output:**

\[
\text{let } v := e_1 \text{ return } \{ e_1 = v, e_2 = v \}
\]

Part of the definition of the above equiv predicate for XQuery follows

- \( e_1 \) and \( e_2 \) refer the same variable in the same scope, or
- \( e_1 \) and \( e_2 \) are arithmetic expressions with operands \((o_{11}, o_{12})\) and \((o_{21}, o_{22})\) respectively, and both \( \text{equiv}(o_{11}, o_{21}) \) and \( \text{equiv}(o_{12}, o_{22}) \) holds, or
- \( e_1 \) and \( e_2 \) calls the same functions and for each arguments, equiv() holds.

Scope in XQuery can be introduced in FLWR expressions, quantified expressions, and typeswitch expressions[1].

This paper focuses on CSE related to remote document access, that can be detected by the 3rd condition of the above equiv() definition.

<results> {
    for $a in distinct-values(document("http://www.bn.com/bib.xml")//author)
        return <result>
            { $a }
                where some $ba in $b/author satisfies deep-equal($ba,$a)
                return $b/title }
    </result>
} </results>

**Figure 1:** W3C Use Case XQuery 1.9.4 Q4

let $d := document("http://www.bn.com/bib.xml")
return <results> {
    for $a in distinct-values($d//author)
        return <result>
            { $a }


{  for $b in $d/bib/book
    where some $ba in $b/author satisfies deep-equal($ba,$a)
    return $b/title }
</result>
} </results>

Figure 2:  W3C Use Case XQuery with CSE optimization applied

Figure 2 shows removal of redundant HTTP access within for clause by replacing document function call with variable reference bound once at the top-level let clause.

3. PRELIMINARY PERFORMANCE EVALUATION

Figure 3:  Query execution time of Use Case 1.1.9.4 Q4

Figure 3 shows query execution time of XQuery in Figure 1 (original) and that in Figure 2 (CSE). QuiP [4] for Linux is used for XQuery processing. Client machine executes XQuery which in turn make HTTP access to obtain input XML document via document() function call.

According to document attached with QuiP, it works on main-memory and optimization is not available. Although prime target of QuiP may not be the performance, we believe examining difference in performance is useful to assess effect of query rewriting optimization.

Evaluation environment follows:

Client  CPU:  Mobile PentiumIII-800MHz, OS:  Linux 2.4.2, Main Memory:  320MB
server  CPU:  PentiumIII 600MHz, OS:  Linux 2.2.12, Main Memory:  256MB, httpd:  Apache 1.3.9

network  Within same 100Base-T segment, via repeating HUB.
data  Randomly generated XML book data that almost adheres DTD of sample XML data described in[2]. Number of <book> entries was varied from 2 to 1024 logarithmically.

Result

Original query produced HTTP access proportional to the polynomial of the number of book entries in remote XML file, while CSE optimized query accessed remote XML file only once.

Between 16 and 128 books of data source, two queries have the same order of complexity, but CSE accelerates execution by a constant factor of about 4. From 256
elements above however, original query causes thrashing at client machine and CPU usage during query execution was only 5%.

This is because QuiP works on main memory while working set did not fit in main memory.

4. RELATED WORKS
In [5], rewriting rules in terms of algebraic operations can be found.

XQuery implementation Qexo [6] uses Soot framework [7] at back end, where CSE is used as one of its optimizations.

5. CONCLUSION AND FUTURE WORKS
In this paper, opportunity of reducing network access in XQuery by traditional common subexpression elimination optimization had been addressed. Preliminary performance evaluation proved the effectiveness of this optimization.

Experimental result shows cases where contents are text media. We can extend the result to retrieval of MM contents via XQuery access through metadata. That can lead to less remote access and thus more meaningful traffic utilization for MM content access. Furthermore, as XQuery can incorporate user defined predicates or functions that might access MM contents themselves, this optimization can reduce costly execution of such functions that realize less redundant computation and access to MM contents.

However, query translator to perform CSE has not yet been implemented. Problem of finding common subexpression will be equivalent to finding distinct sub-trees in parsed XQuery input, with XQuery language specific aspects being considered. Part of common subexpression detection criteria had been discussed.

According to XQuery semantics, input functions such as \texttt{document()} with the same arguments during the scope of a single query or transformation must return the same result. This means that even remote XML data changes while query is processed, \texttt{document()} function points that data should retain same return value. This requirement means CSE proposed in this paper is not only an optimization but also a mandatory processing.

Acknowledgments
We would like to thank Software AG for allowing us to download their XQuery prototype QuiP which is among very few freely available implementations that support external document reference at this moment.

6. REFERENCES