XCache - A Semantic Caching System for XML Queries*

Li Chen Elke A. Rundensteiner Song Wang
CS Department, Worcester Polytechnic Institute, Worcester, MA 01609
{lichen|rundenst|songwang}@cs.wpi.edu

1. INTRODUCTION

A wide range of Web applications retrieve desired information from remote XML data sources across the Internet, which is usually costly due to transmission delays for large volumes of data. Therefore we propose to apply the ideas of semantic caching to XML query processing systems [2], in particular the XQuery engine. *Semantic caching* [3] implies view-based query answering and cache management. While it is well studied in the traditional database context, query containment for XQuery is left unexplored due to its complexity coming with the powerful expressiveness of hierarchy, recursion and result construction. We hence have developed the first solution for XQuery processing using cached views.

We exploit the connections between XML and tree automata, and use subtype relations between two regular expression types to tackle the XQuery containment mapping problem. Inspired by XDuce [1], which explores the use of tree-automata-based regular expression types for XML processing, we have designed a containment mapping process to incorporate type inference and subtyping mechanisms provided by XDuce to establish containment mappings between regular-expression-type-based pattern variables of two queries. We have implemented a semantic caching system called XCache (see Figure 1), to realize the proposed containment and rewriting techniques for XQuery.

The main modules of XCache include: (1) **Query Decomposer.** An input query is is decomposed into source-specific subqueries explicitly represented by matching patterns and return structures. (2) **Query Pattern Register.** By registering a few queries into semantic regions, we warm up XCache at its initialization phase. (3) **Query Containment Mapper.** The XDuce subtyper is incorporated into the containment mapper for establishing query containment mappings between variables of a new query and each cached query. (4) **Query Rewriter.** We implement the classical bucket algorithm and further apply heuristics to decide on an "optimal" rewriting plan if several valid ones exist. (5) **Replacement Manager.** We free space for new regions by both complete and partial replacement. (6) **Region Coalescer.** We apply a coalescing strategy to control the region granularity over time.

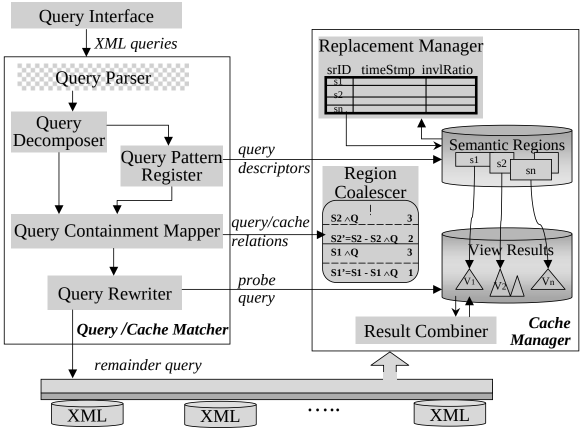

Figure 1: The XCache Architecture

2. DEMO DESCRIPTION

Our XCache prototype is developed in Java JDK 1.3 and the front end is hosted by Apache Web Server and Tomcat servlet engine. We have adapted Kweelt (available from http//cheops.cis.upenn.edu/Kweelt:) for evaluating XQuery and exploited the XDuce system [1] for establishing type inclusion relations between the pattern variables of two XQueries. The demonstration focuses on:

- validating the rewritten queries and results with the results returned from the remote sources directly;
- monitoring system internal decisions and status;
- displaying utilization of cache regions over time;
- analyzing query performance with various workloads.

3. REFERENCES

[1] H. Hosoya and J. Vouillon and B. C. Pierce. Regular Expression Types for XML. In *Int. Conf. on Functional Programming (ICFP)*, pages 11–22, 2000.

[2] L. P. Quan, L. Chen, and E. A. Rundensteiner. Argos: Efficient Refresh in an XQL-Based Web Caching System. In *WebDB 2000*, pages 23–28, May 2000.

[3] S. Dar and M. J. Franklin and B. Jonsson. Semantic Data Caching and Replacement. In *22nd VLDB*, pages 330–341, 1996.

*This work was supported in part by the NSF NYI grant #IRI 94-57609 and IBM fellowship support.

ACM SIGMOD '2002 June 4-6, Madison, Wisconsin, USA
Copyright 2002 ACM 1-58113-497-5/02/06 ...$5.00.