Problem Statement

CS 461 - CS Senior Capstone

XZES40 | Group #40

Zixun Lu (luzi), Shuai Peng (pengs), and Elijah Voigt (voigte)

March 18, 2017

Abstract

This document outlines the problem, and proposed solution, for creating a high-throughput and multi-platform XML/XSLT translation server. While XML/XSLT document transformation is not new, existing systems do not meet industry needs for high-volume document transformation; blindly spending resources fully compiling documents at each request, even if they were previously processed. Our proposed solution is to cache parsed documents in memory, in the XML/XSLT transformer and XML parser, to avoid spending computing resources re-compiling documents which have already been processed. The project will be developed with the Apache Xerces-C, Xalan-C, and ICU libraries for XSLT transformation, XML parsing, and Unicode support respectively. As a stretch-goal this project will be design to target multiple operating systems, and create installation packages for each supported platforms.

Problem Definition

Extensible Markup Language/Extensible Stylesheet Language Transformations (XML/XSLT) are common computational tasks requiring the compilation of XML/XSLT documents and the transformation of those into a desired output file. Existing transformation systems accomplish this task by compiling the desired input files, running transformations on them, and producing an output document. These systems unconditionally compile input documents, even if they have been processed before. While this works for a small number of requests it does not scale well as it may spend resources processing documents which have already been used. There is not a suitable XML/XSLT server which is able to detect a reoccurring document and bypass its compilation by pulling the compiled document (or transformation) from a cache.

Proposed Solution

Our solution to this problem (as hinted in the problem statement) is to create an HTTP protocol XML/XSLT transformation server with two layers of caching: one at the XML Parsing layer and another at the XSLT transformation layer. Caching at these two layers would bypass time consuming document compilation and web-requests, speeding up the transformation process greatly.

The server application will accept an XML document and an XML rule-file, and produce an output document. Both the input document and rule-file will be cached after being processed. Some checking will need occur to ensure that new or modified documents are not ignored and that the cache for a given document is up to date.

After initially processing and caching (or retrieving) the documents they will be passed to the XML parser. The parser will fetch and cache (or retrieve) a catalog of additional files needed for document processing. Ideally, given an identical set of input files, minimal processing (verification really) will take place and a cached response will be returned quickly. To create this server application we will use the Apache Xalan-C, Apache Xerces-C, and the Unicode Consortium ICU libraries. [**xalan**, **xerces**, **icu**]

Time permitting we will also design our application to easily target multiple platforms (Linux, BSD, Windows) by having the application interface with an API rather than system-specific interfaces. Furthermore we we hope to package our application for our supported platforms to allow for easy installation.

To demonstrate the transformation server's improved performance our group will show, through the CLI interface and recorded metrics, that there is a performance boost from our work. Time permitting we will also record our server's performance relative to existing XML/XSLT transformation servers.

This proposed solution will have an impact similar to that of any incrementally better tool. The transformation server will not do anything new, but it will make the lives of many individuals easier by speeding up the tedious task of transforming XML/XSLT documents. Nobody will claim that document transformation is exciting, but it is a necessary task. By speeding up the document transformation process we reduce the amount of time people spend doing this chore, allowing people more time to do something else... or allowing them to process more XML/XSLT documents in a day.

Performance Metrics

This application will need to have transformation correctness and a user interface.

The transformation correctness can be verified by taking known input documents with a predictable output document and verifying that they produce the correct results. We will also verify that our caching layers do not find false-positives nor false-negatives. These tests can be largely automated for convenience.

Our user interface can be anything from a native GUI, to a command-line interface, to a website. For simplicity we will target a command-line interface. That said we will be producing an HTTP web-api, so additional interfaces can be built by interacting with our application via common HTTP requests. Time permitting we will target producing a simple web-interface since the users of this system will be individuals who are usually non-technical (e.g., local law enforcement, office managers, etc).