High Performance XML/XSLT Transformation Server

Winter 2017 Midterm Progress Report

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

CS 462 | CS Senior Capstone | Winter 2017

June 4, 2017

Abstract

An update on the development of the High Performance XML/XSLT Transformation Server named *XZES40 Transformer*.

Figure 1: Source: Wikimedia Commons [3]

Figure 2: Source: Apache Software Foundation [1]

1 Project Purpose

XZES40 Transformer is an implementation of a standard XML document transformer. This type of application takes two documents as input: an XML document and an XSLT document, and performs an XML document transformation. This is similar to how in an Excel document one might take a series of columns, perform some operation on that data, and set the output to be a new column. Just like Excel, XML document transformation is used to perform operations on data, except that it can be highly automated and done in much larger quantities with programs like ours.

XZES40 Transformer is unique in that it increases performance by both caching already-processed XSLT documents and it performs transformations in parallel. The use of an in-memory cache will cut down considerably on the time required to perform a document transformation. This is because much of the time transforming documents is spent compiling the (commonly re-used) style-sheet. The application will also accept transformation jobs over the internet via a web API (used through a website we will create); this allows users to perform document transformations without installing the application locally.

2 Project Status

2.1 Zixun Lu

Before the winter term started, our teammates Elijah built the structure of project. It helped us to know how to do this in the beginning. Developing this project is hard for me at the beginning. Our team are required to develop our application in Debian Operation system by our client Steven Hathaway. Because each teammate uses different operation system, so we use different tools to develop our application. Shuai and Elijah used vagrant to develop the application. I downloaded the virtual machine to develop my application. We also need to add libraries to our compiler. Even we wrote the design document and technology review in the fall term, I still don't know how to write the real code in this project. I checked the Xalan C++ how to transform an XML and XSLT document. The Xalan C++ is a transformer which transforms the XML and XSLT document to a new stylesheet. The Xalan website gives us some information about the basic function transformation how to do it. Our team finish the basic transformer before the midterm. This transformer doesn't include cache and parallel computation features. This application only can transfer one XML with one XSLT document to a new stylesheet. It can handle parsing XML and XSL objects. Our team also did the benchmark report for application. This benchmark report include two similar applications which those can transfer one XML with one XSLT document. From this benchmark report, our team will set a goal for application. The most important competitor is Xalan because it is also an open source. Our team hopes our application is faster than Xalan. By end of the midterm, our team finish the basic transformation and benchmark report. We have already started the caching part work. Even our progress is slower than our expects, we still have confidence to finish on time.

2.2 Shuai Peng

Because all of our teammate is busy in this term, we are slow than our schedule that we planed in the design document. So far, we just complete the basic function of the XZES40-Transformer. Becasue some personal reason, I did nothing during break, but I read the document of the xalanC++. Elijah list all structure of our program and create a portable development environment during winter break. All of those is helpful for us. Beginning at this term, I finished our basic function, the XML document transformation, individually, however it still has some bugs here. I tried fix bugs, and there still one question bug here. Thus Elijah and me come together and fix the last one question bug, and we finally

get it. The XZES40-Transformer can get one XML file and XSLT file and generate a new XML file. The cache and the parallel computation will be completed during the beta version.

2.3 Elijah C. Voigt

As mentioned above, our group has made unfortunately slow progress on development. I made considerable progress over winter break before the term started. This work included outlining the structure of the code, writing skeleton code to the repository, setting up automated testing, creating a portable development environment with Vagrant [2], and documenting what had been and had yet to be done.

When Winter term began we had not all made similar progress on the application; for one reason or another Zixun Lu and Shuai Peng were not able to do more than light research on some components of the project. As the term picked up I tried to solidify what needed to be done and how to do that. This included helping team members use the development environment, understand the existing code, and outline where components of the project ought to be written. I was unable to meet as frequently as I did in Fall term, but did my best to help Zixun Lu and Shuai Peng get caught up, making myself available remotely if not physically. Despite this help, and because of busy schedules, Zixun Lu and Shuai Peng still had a hard time getting caught up; Shuai less so than Zixun.

We quickly reached a show-stopping bug which, with two out of three developers playing catch-up, halted development for a week. I made the decision after that week to gut the incorrect code and fix the bug by re-designing the problematic portion from the ground up. This fixed the bug and got us to completing the core function of our project: XML document transformation.

Although we are behind schedule we have the core of our project working and it is at least *possible* to complete the project's requirements on time.

3 Remaining Tasks

3.1 Zixun Lu

After we finish the basic XZES40-Transformation, I will help my teammates to do the cache firstly. Because our client Steven Hathaway thinks the cache is the core of application. The cache is a core feature of the XZES40 application. The cache speeds up document transformation by storing previously parsed documents. After we finish the cache function, I will start develop the parallel computation. The application will carry out computations in parallel to further leverage the computing resources available to it and compile documents even faster. At the end, our team finish the whole project, I will develop the CentOS packages. We intend to develop tools for multiple platforms using FPM to create cross-OS packages would be very convenient. Our team hopes each platforms users will use our application. My teammates will develop Windows and BSD packages.

3.2 Shuai Peng

The remaining tasks that I am responsible for is complete the cache function, which is the "plus one" feature for the XZES40-transformer. And I also need create web interface. The web interface is the GUI for user, and it makes people easy to using our application on any browser. I will create Windows package, this is help user to quickly install the XZES40-transformer on any PC.

3.3 Elijah C. Voigt

Having completed basic document transformation, my remaining tasks include turning the application into a daemon (long-running process), exposing it over the web with a CGI script, creating a command-line interface, and creating a FreeBSD package for the application. The latter two tasks being stretch goals, the former two being hard requirements necessary for the beta release of our application.

As a group we *also* have to complete the document caching, parallel document transformation, **benchmarking** (required to grade our project performance), the website, and the Debian Linux package. This is to say nothing of the stretch goals. We have spoken with our sponsor about this and while our requirements have not changed, the priority of them has. This change in priority has been reflected in our updated design document.

4 Problems Encountered

4.1 Zixun Lu

As a Windows operating system user, it is hard to use vagrant in my system. The Windows operating system is not like macOS system. The Windows operating system don't have terminal which can directly run the vagrant. So I decided to use virtual machine to develop the application. It was my first time to use Debian operation system but it is similar with Linux system. There is a problem in doing the benchmark report. We want to set a goal for our application and choose Xalan and Altova to compare with our application. Our application is required to develop in Debian operation system but Altova only works in Windows platform. The Xalan C++ can work in the Debian operation system. I am concerned that the measurement of the data will affect the final outcome. This class required us using Github to connect the each others. Before I took this class, I thought that Github is only for uploading the files. Right now, I understand that Github is not only uploading the files but also the developers can communicate with each other. They can commit and make changes in the code. If you have any questions, you can make a pull request or issues on the Github. The Github also has command line to help develops doing their works in terminal. Our team had difficult on handling to parse XML and XSL objects. We also didn't clear the data flow in the project. When we try to compile and test our projects, there are a lots of bugs in our application. I think the most important reason that we can't fix bugs immediately is my fault. After I took CS261, I didn't write any C++ code for 2 years. My terrible coding ability delay the project progress. I'm struggle to debug the codes. When I tried to look for helps on the Internet, I still didn't get anything. But I won't give up, I will catch up the schedule as soon as possible.

4.2 Shuai Peng

When we are working on the basic document transformation, we face a strange problem. The structure and the implementation of our code should be no any problem. However the compile also say that we have expected unqualified-id bugs for the XalanC++ transformer function. Elijah and me search on the Google, and most of answers said that you should check the syntax of code and some semicolon error. We double check the code, but there is no error such that syntax and semicolon problem. So, we come together and write a new prototype for test XalanC++ transformer, and the prototype is work correctly. Thus, we decided to re-write the transformer.cpp, and make sure this code is only doing one things – transformer document. We change the structure of transformer.cpp from OOP to normal function code, and also make sure the document.cpp pair this file. Finally, we get over this bugs, and our program can transformer document successful.

The another problem is that the documents of the XalanC++ and the XercesC is hard to read and understand. If we want some special function, we can not search it on the API document, the page has bad structure to start search. Thus we just used the demo-code from our sponsor and the usage pattern from the XalanC++ website to start write our code.

Behind the tech problem, our team have to face the time schedule problem, because we did not spend much more time on this project. We have much slow than the schedule that we write last term, and I think we should speed up and spend more source on our project. Elijah suggests that we spend whole day meeting to work our project, I think this is a good idea, so teammate can help each other to finish our project.

4.3 Elijah C. Voigt

Although we purposely scoped our project to be relatively simple and straight forward, we have encountered a lot of stumbling blocks, both technically and non-technically.

Technically we encountered the aforementioned unqualified-id compilation error, this halted development for far too long. Beyond this, members of our team struggle with basic concepts in programming like object oriented programming, data structures, and how best to debug a poorly written program. This can be summed up as *technical debt*.

Non-technically Shuai Peng and Zixun Lu struggled with getting caught up after break, Shuai fairing much better than Zixun. There were valid excuses for this lack of preparedness, but that combined with a lack of willingness to invest an appropriate amount of time in getting caught up, compounded with the technical debt, meant that we have not made nearly as much progress as we need to have made by this point in the term. When we encountered that show-stopping bug mentioned above both team-members did not demonstrate an understanding of the code well enough to either decide to re-write the part of the code that was broken or solve the problem. While this sounds like a technical problem, it is ultimately a problem solving skill I consider non-technical. Everybody invested time in solving the bug, so we all learned something about the project and code, but it still took a very long time and ultimately set us back a ways.

The biggest thing that has bitten us so far is prioritizing work on this project on-par or above that of other class-work. If we as a team spent *much* more time working on the project we would be done by now. As it stands we, and I include myself in this, have not spent enough time on this. I have not spent enough time on the project because I am busy and because I feel as though I have done the lion's share of the work; for work distribution equality I hoped that we would each do equal work, but in practice I will most likely need to do more than my fair share of the work for the sake of the project.

I plan to address this time-prioritization by forcing the group to spend an entire day working on the project instead of delegating jobs and individually working on these. I hoped that we would be able to complete the project asynchronously, checking in weekly, but I was wrong. We will spend a full work day every Sunday from now until expo working on the project. While this may end up being too little too late, it is the only time in our schedule that works, so it's the best we can do.

5 Interesting Code

5.1 Zixun Lu

```
xzes::cli_arguments_t* args = xzes::parse_args( &argc , &argv )
```

We create own logo xzes as namespace. We parse CLI arguments into struct "cli_arguments_t"

```
Int main (int argc, char * argv[])
```

The application parses user input file.

```
xzes::Document::Document( xzes::uri_t file_path )
```

The application stores all data about documents and functions to act on that data.

5.2 Shuai Peng

This is the structure of the XZES40-Transformer code repository.

```
shell> ls
bulid doc examples include lib Makefile README.md src test
```

This is the usage of the XZES40-Transformer.

```
shell> ./main
Usage:
  a.out --xml=input.xml --xsl=style.xslt [--out=output file]
```

This is the result of the XZES40-Transformer. We print the new xml file on screen for easy debug, but it will generate a xml file in the future.

```
shell> ./main --xml=simple.xml --xsl=simple.xsl --out=result.xml
<?xml version="1.0" encode="UTF-8"?><out>Hello</out>
```

5.3 Elijah C. Voigt

While it might not seem like much, this is the core of the XZES40-Transformer application. This function is passed a struct of three arguments, the XML file path, the XSLT file path, and the (optional) output XML file destination. For technical reasons it currently prints the output document to standard output.

```
int xzes::transform_documents( xzes::cli_arguments_t *args )
{
    // Initialize Xalan
    XMLPlatformUtils::Initialize();
    XalanTransformer::initialize();

    // create a xalantransformer
    XalanTransformer theXalanTransformer;

    // Allocate objects on the heap so they can be cached in the non-prototype version.
    // XSLTInputSource  *xml = cache.get(args.xml);
    Document xml(args->xml);
    Document xsl(args->xsl);

    // Perform transformation and capture results
    int theResult = theXalanTransformer.transform( *xml.get_content().obj ,
                                                   *xsl.get_content().obj ,
                                                   std::cout );

    // Terminate xalan
    XMLPlatformUtils::Terminate();
    XalanTransformer::ICUCleanUp();

    return theResult;
}
```

Eventually this will perform document transformation and setup in a separate thread for each requested transformation. That functionality will be the core of the 'Parallel transformation', an anticipated speed-boost of our application.

The Document class used to instantiate the xml and xsl variables are custom written classes. They encapsulates the Xalan-and-Xerces-specific code as well as handling the parsed-document caching. This creates a clean separation of concerns, which (in theory) makes our code modular for current development and more maintainable for future development and maintenance.

6 Relevant Media

We have not completed development enough to have screen-shots and demos to show. We expect to have a functioning website by the end of the term. Usage of the application can be found above in the "Interesting Code" portion of this report.

References

[1] *ASF Press Kit: Apache Software Foundation Logo*. URL: https://www.apache.org/foundation/press/kit/.

[2] *Vagrant: Create and configure lightweight, reproducible, and portable development environments.* URL: https://www.vagrantup.com/doc

[3] *Wikimedia Commons: Oregon State University Logo*. URL: https://commons.wikimedia.org/wiki/File:Oregon_State_University_log