Apache Log Processing

12.9.14

Matthew Sarver

Michael Shields

John Stanley

ABSTRACT

Modern websites can generate a tremendous amount of valuable log data. System administrators and webmasters are challenged with interpreting this data and using it to provide a higher quality service. However, many tools to do this are either closed source or proprietary, and distinctively lack support for high processing performance with the Nvidia Cuda framework. Our challenge was to develop a framework that allows a future endeavor to add support for the Cuda C API, and insure that future development could focus entirely on tackling the vast power that Cuda offers. This paper shall discuss the various design choices, development concerns, and the future code maintenance of this Senior Design project.

INDEX

INTRODUCTION

Our senior design project was first given to us during the Spring 2014 semester by our client Dr. John Kapenga to use parallel processing to parse and analyze large server log files. Speed and computational efficiency were of absolute importance and our main focus when designing and developing this project. Initially, we were tasked with using the Cuda C API, provided by the Nvidia Corporation, to analyze these log files. Unfortunately, we were unable to obtain firewall log files and later discovered that Apache logs, largely composed of string data, were ill-suited for numerical calculations that the Cuda API focuses on. This meant that we were unable to use the Cuda C API and instead had to rely on other methods of processing and analysis.

Our backend application needed to be lightweight, convenient, accessible, and extensible. This meant that future developers for the backend could easily add new analysis algorithms with little maintenance required on other parts of the program. It had to be well documented and provide useful scripts, so future developers could easily add new C code and quickly compile a new backend. Also it had to be built for future Cuda processing, which would allow for advanced, efficient analysis.

Cuda is a parallel processing framework for Nvidia graphics cards. Nvidia has provided a free API for their GPUs using the C programming language which can solve a variety of intensive problems – and not limited to pixel based computations. Cuda integration was an initial goal of the project, but soon proved to be infeasible due to a lack of support for operating on string data.

Since our goal was to provide an open framework for log analysis, we wanted to research similar products already available. Our focus soon turned to Apache access logs after failing to acquire firewall logs, and so we researched software such as GoAccess and Apache Hadoop. We found that these software products were not very compatible with extension or support with the Cuda C API and elected to develop a framework of our own.

DESIGN DECISIONS

Our application was designed with four main parts, or modules. These modules were each replaceable and extensible, allowing future developers to change what they need without rewriting the entire project.

The C application used three of the four main modules, the Reader, the Parser, and the Analyzer. These parts would need to be developed separately and then connected up to the main core of the application, and each module would provide a miniature API for the core to use.

DESIGN

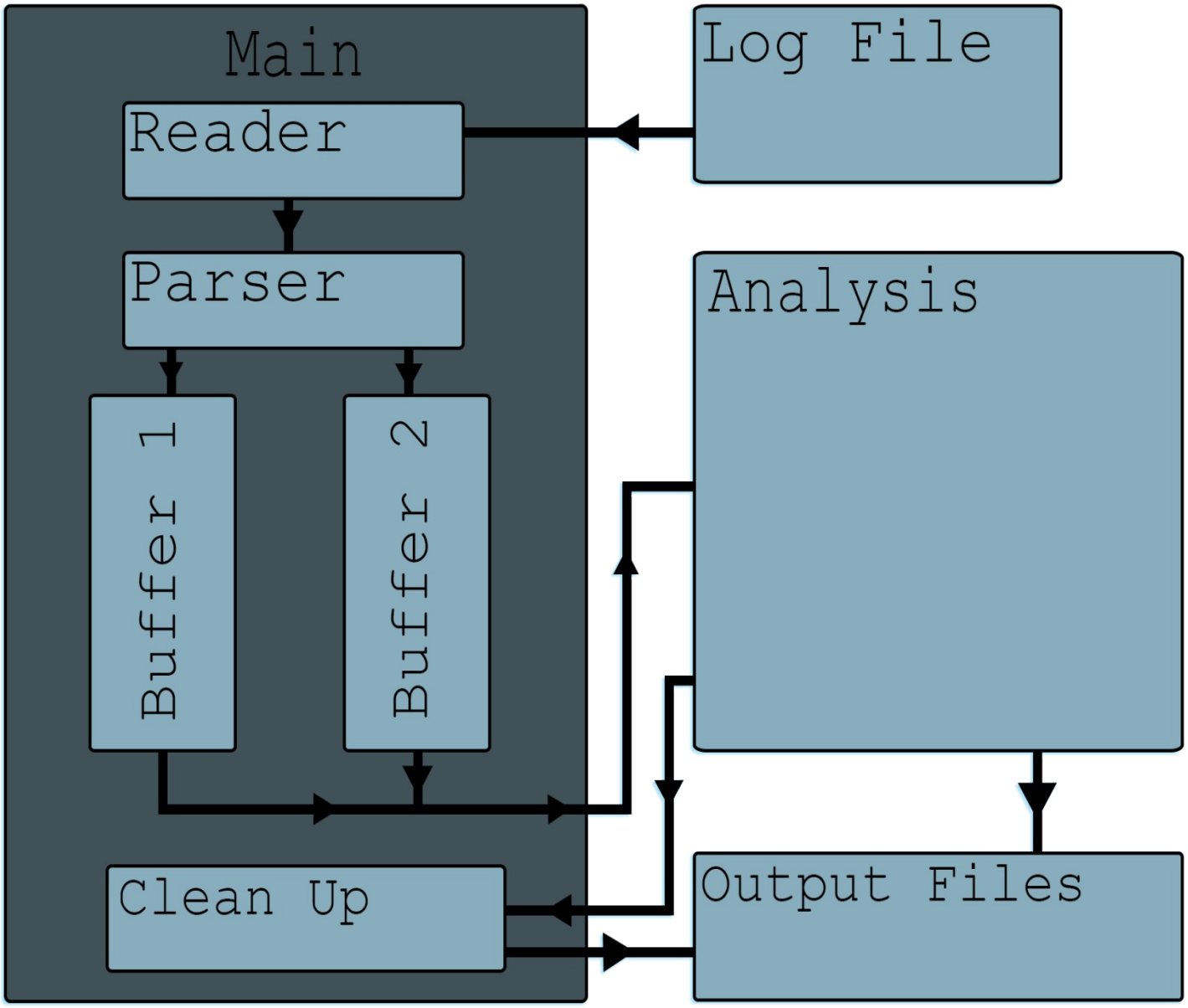
The project is designed into two main parts, the backend C application and the front end web dashboard. Across these two main parts, there are four main modules, the log Reader, the Apache web log Parser, the Analyzer, and a module devoted to the web end. These four main modules are built to be extensible and modifiable by future developers, and rely on few third party dependencies.

The first three modules – the Reader, Parser, and Analyzer – are constructed using the C Programming language, relying on few dependencies and making an emphasis on speed and strong memory management. Furthermore, Nvidia provides a API for Cuda in the C programming language, and unlike it's Python counterpart, it is free to use. Since memory management is a large concern, C provides a vast amount of control over the memory the application can use.

The fourth module emphasized accessibility and responsiveness, so that System Administrators could quickly receive statistics and reports for their web log data. The project also needed to display these statistics in a easy-to-read and friendly manner. The web interface uses two third party libraries, Bootstrap and Highcharts. These third party libraries provided a responsive interface for the web end and various graphs and charts for the system administrator, respectively. The web framework also required a SQL database for basic login support, and this would be achieved through MySQL. Because MySQL is free, well documented and popular with other students at Western Michigan University, it was the natural choice for relational databases.

Development required a few more tools to create the project. Doxygen provided the framework with dynamic code documentation generation, which could then be linked up to the web end for future developers to use easily. Git was used for source version control because it was reliable and convenient, well supported by Linux and Unix development environments, and allowed development to keep the project in a free, open, and public Github repository. Compared to alternatives such as Subversion or TFS, Git was a much better solution for source version control.

DEVELOPMENT AND IMPLEMENTATION



The backend application works as follows: while the Reader has not hit end of file, it reads in a line from the Apache log file and the line is then passed to the Parser, where it is checked against two different regular expressions – one for the Common Log Format, the other for Combined – and then if it passes the check it is then parsed into a C structure containing each relevant field in the line. This structure is now added to one of two Buffers. One Buffer is read and filled while the other Buffer is

run through analysis, and both buffers are considered busy until they are finished with these tasks. Once the analysis is complete, the statistics generated are placed into a text file and the application cleans up the memory used. This statistics text file is then read in by the web end, and properly displays the statistics through the use of Highcharts.

TESTING

Each module of the application was developed and tested separately before being attached to the main application core. All four modules were extensively tested with a variety of input data, insuring that all possible inputs were correctly handled and each function within the framework could fail safely and elegantly. The modules, once linked up to the core, are automated and require little user input, preventing further unexpected input.

The log analysis modules used a variety of log files acquired from the Internet of different sizes and log formats. These logs exhibited many of the anomalies that we were looking for so we would not have to generate our own log files or rely on a third party to provide us with useful log files.

Finally, the web interface was tested with common user input, and properly handles minor security exploits such as SQL injection or session fixation attacks. Because the web interface was constructed with Bootstrap, much of the functionality has been tested by the Bootstrap development team.

MAINTENANCE

When looking forward with this senior design project, there are several improvements that could be made. Many of the algorithms present in the modules for processing, parsing, or analysis could be improved to yield higher efficiency and better performance. Because the application is built to be extensible, future development could add a larger variety of analysis functions, and the web interface could be restructured to increase functionality and take advantage of popular MVC frameworks. The current project is limited to both Common and Combined log formats, and could be extended to take advantage of custom formats or altogether be replaced to handle firewall logs. Finally, the Cuda C API needs to be taken advantage of, and real time analysis could also be supported.

Because development required a radical shift in design goals, the code base may also need to be cleaned up. The core application, however, is well documented and commented to provide maintenance developers with a detailed look into how this application works, making general code maintenance far less painful to do.

Further security measures should be taken with the web interface. A popular MVC framework, such as Laravel or Ruby on Rails, accomplishes this task and provides future developers with a well documented API for modifying the application to handle user authentication and general security.

SUMMARY

In conclusion, the goals of this project were to provide a lightweight and extensible framework for parsing and processing web log files, accessibility and responsiveness for future developers and users alike, and make the core project extensible to handle future development with the Cuda C API. This project reached its goals, with little to no dependencies in the C backend, and provided a responsive web interface through Bootstrap. Despite numerous setbacks, this framework can be expanded upon to provide a complete, more robust, open service.

GLOSSARY

APACHE – A web server application developed by the Apache Software Foundation. Apache is open-source software and is released under the Apache License.

API – Application Programming Interface, a specification that eases the development of software by providing a standardized way to access a particular service

CLIENT - The person requesting the service of the software development team.

CUDA – Compute Unified Device Architecture, a parallel programming architecture for NVIDIA graphics cards. Software programmers are provided with an API for the CUDA architecture in the C Programming Language.

DEVELOPMENT TEAM – The authors of this document, and senior developers for the project, contracted by the client.

GPU – Graphics Processing Unit. Hardware within the computer designed to handle complex floating point calculations or general-purpose processing.

NVIDIA – Nvidia Corporation, an American technology company with headquarters in Santa Clara, California.

RELEASE – A version of the software, made available to the client. Does not imply completeness of the entire project.