Hello Professor!

I have been working with Professor Podnar on my thesis proposal, and he had mentioned that you would be willing to review it. If this is the case, please find enclosed a copy of the proposal. If you have any questions or feedback, please feel free to contact me anytime:

- Email: skenny214@gmail.com
- Phone: 203-451-3135

Thanks for your help!

Best,

-Sean Kenny-

[illegible]

Analysis and Implementation of a Web Request Pipelining Framework

Thesis Proposal

For the degree of Master of Science in Computer Science

At Southern Connecticut State University

Sean P. Kenny

December 2011

Thesis Advisor: Dr. Hrvoje Podnar

Major-Field Approval – The advisor and the department chairperson

Advisor	Date
Chairperson	Date

A. Title

Analysis and Implementation of a Web Request Pipelining Framework

B. Statement of Purpose

Modern web sites and web applications provide valuable information and services to an ever growing web audience. One of the most important requirements of web sites and web applications trying to deliver this important content is speed. Increased speed of a website can often determine an end user's perception of overall quality and value of a web based service. Fast and responsive sites will be more likely to achieve higher monthly page views, adoption rates, and overall success.

If speed is recognized as a critical factor for today's web then the current HTTP protocol used to deliver this web content should be examined as the source of a possible bottleneck when trying to deliver it. Web content has become increasingly more dynamic and interactive over the last ten years, and this current standard for web content delivery can be tailored for meeting the current pressures of the modern web.

Engineers at Facebook have authored multiple posts describing an enhanced HTTP request system utilizing a proprietary HTTP request pipeline. These engineers state that through utilizing an HTTP pipeline framework it has become possible to increase the performance of Facebook by two fold. Facebook engineers call their HTTP pipelining implementation, "BigPipe". BigPipe is explained as a component used at Facebook to deliver content as soon as possible, in a sequence that increases perceived load time and speed. [1]

This project intends to analyze the current HTTP based web request and response system utilized by modern web browsers and servers today. Following this analysis, the project will provide a software implementation of an HTTP web request pipelining framework which will attempt to increase speed of web content delivery through the optimization of a? series of phases that occur awk during standard sequence of events for a given HTTP web request. The final intention of this project is to provide a web server extension for Apache based web servers which will fully implement an HTTP request pipelining framework, and allow for a greater level of transparency to web developers whom wish to provide this advanced request pipelining paradigm to their web sites and web application in a more pluggable manner.

great!!

C. Literature Review and Current State-Of-The-Art

Since the introduction of the HTTP, web developers have built an ever growing repertoire of tips and tricks to squeeze the most out of the HTTP based web. These methods include, but are not limited to the following [6]:

Make fewer HTTP requests	Use a content delivery network	Add an Expires header (HTTP 1.1)
Gzip components	Put CSS at the top of the page	Avoid CSS expressions
Reduce DNS lookups	Minify JS	Avoid redirects
Remove duplicate scripts	Turn off ETags	Make AJAX cacheable and small

Might be better to just bullet this list. When I read it, I was reading it as Rows across the table....

Today's web sites and web applications employ many of these techniques to help deliver content to end users in an efficient manner.

lower case

The HTTP protocol is part of the application layer of the OSI model [See figure 1]. When an HTTP request is issued from a web browser for an initial HTML document [illegible] related elements such as images, JavaScript, and Cascading Style Sheet (CSS) are subsequently retrieved using additional HTTP requests. Therefore the rendering of a single web page could involve many HTTP web requests. An HTTP pipeline is a thin layer of application logic on top of HTTP which attempts to optimize the request cycle in a manner that allow pieces of a full web page to load and display independently This added layer is developed as a software library containing both server side and client side application code.

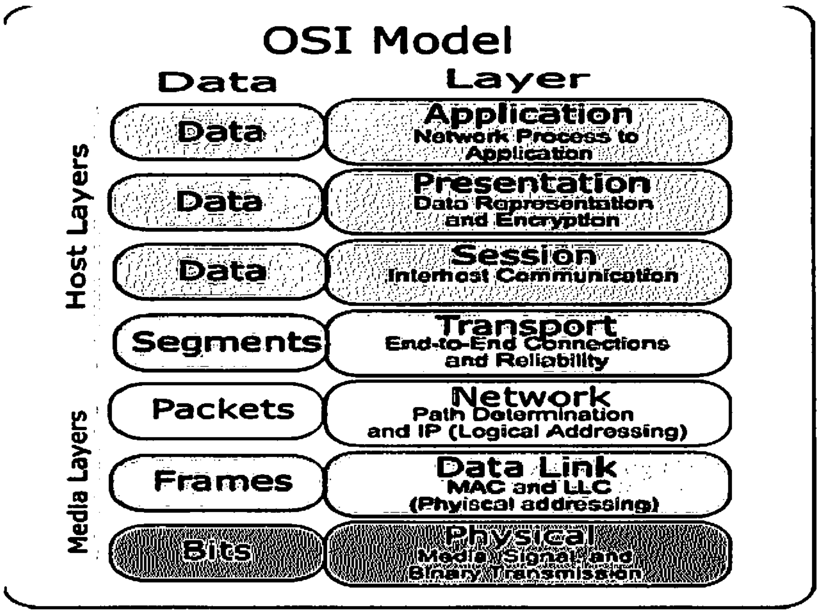

OSI Model
Data
Layer
Host Layers
Media Layers
Data
Application
Network Process to Application
Data
Presentation
Data Representation and Encryption
Data
Session
Interhost Communication
Segments
Transport
End-to-End Connections and Reliability
Packets
Network
Path Determination and IP (Logical Addressing)
Frames
Data Link
MAC and LLC (Physical addressing)
Bits
Physical
Media, Signal and Binary Transmission

Figure 1: Layers of the Open Systems Interconnection model (OSI model)

The resulting framework processes request~~s~~ and deliver~~s~~ responses in an optimized manner taking into account current HTTP limitations that are inherent to the overhead created by the connect and request style of communication. The result of this response optimization through a pipeline is an increase in perceived speed that is accomplished by ~~through~~ displaying fully functional content as quickly as possible – even before the entire document is completely rendered by the web server.

HTTP pipelining is inspired from traditional pipelining technologies utilized by today's modern CPU's, where an instruction's life cycle is broken into multiple stages [See figure 2].

might be helpful to also show figure of instruction life cycle w/o pipelining so one can see affect on time.

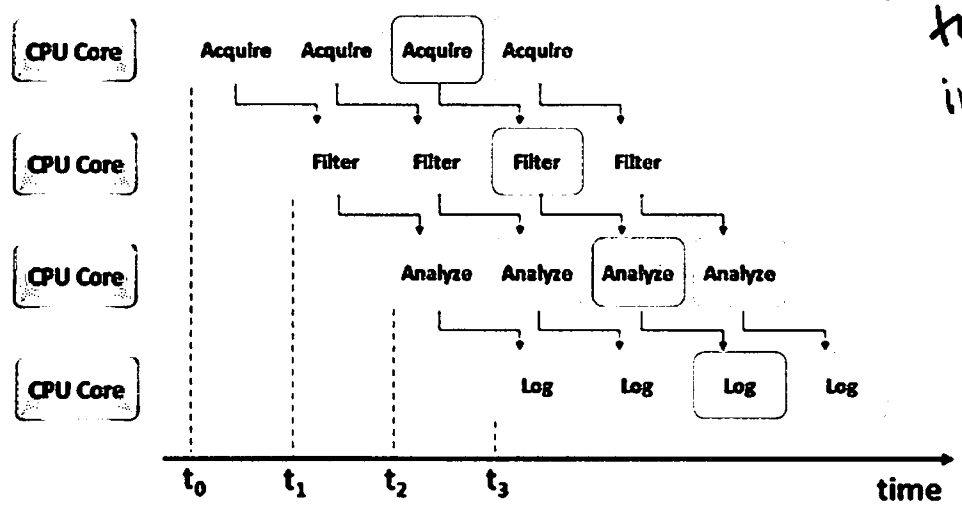


Figure 2: Basic CPU Pipeline

Instead of instructions, HTTP pipelining breaks the page generation process into several stages which include [1]:

1. Request parsing: web server parses and sanity checks the HTTP request.

2. Data fetching: web server fetches data from storage tier.
3. Markup generation: web server generates HTML markup for the response.
4. Network transport: the response is transferred from web server to browser.
5. CSS downloading: browser downloads CSS required by the page.
6. DOM tree construction and CSS styling: browser constructs DOM tree of the document, and then applies CSS rules on it.
7. JavaScript downloading: browser downloads JavaScript resources referenced by the page.
8. JavaScript execution: browser executes JavaScript code of the page

Utilizing an HTTP pipeline, and coupling it with other modern web optimization technologies like content caching, database caching, content compression, and minification~~;~~, web developers should be able to introduce amazingly scalable and advanced web sites that make up the next generation of web content. The main downside to this inherently powerful web technology is that there is currently no out of the box software solution, and private companies such as Facebook have ~~only~~ offered only high level specification details.

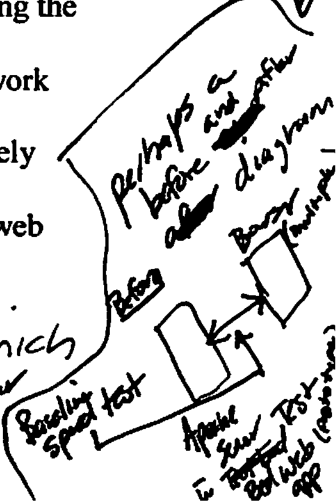
D. Methodology

A prerequisite for all research and development will be to create a prototype web application that will be utilized as the test case for all phases of this project. This web application will have analysis features built into it, which will help facilitate the recording of performance related information with respect~~s~~ to ~~html~~ HTML generation, database requests, and JavaScript execution. This website will attempt to utilize modern web application technologies including CSS, AJAX, PHP, and MySQL database connectivity. Using this web application, analysis of speed, speed up, throughput, bandwidth, network time, generation time, and render time will be made universally available.

Initial research will be conducted to obtain a thorough understanding of how four of today's modern web browsers (Firefox, Chrome, Safari, and Internet Explorer) request web pages from an Apache web server. This stage will reveal details of request sequences including Document Object Model (DOM) instantiation, CSS style sheet requests, JavaScript requests, and image file requests. Once this information is properly mapped, measured, and understood the next step of this research will be to obtain a concrete baseline for speed based on information and samples taken from the previously developed web prototype application.

The next stage of this project will then be to implement ~~an HTTP pipelining~~ specification. This will require a thorough analysis of postings made available by the Facebook engineering team, which broadly outlines the BigPipe system that the team developed between ~~the periods of 2009-2010~~. Since this specification is not in all aspects complete this reverse engineering process will likely involve some assumptions, code analysis, and educated engineering decisions in order to fully implement a functional version of an HTTP pipelining framework.

Once ~~this~~ the initial pipelining framework prototype is complete, testing will commence by converting the previously developed web application to fully utilize the HTTP pipelining framework now available. During this testing phase the developed framework will be iteratively tweaked and improved based on performance information gathered from the test web


perhaps a before and after diagram
Before
Browser (multiple)
Baseline speed test
Apache Server w/ Test Bed Web app (prototype)

Would be helpful to show a block diagram of your system which includes your web app (used as test) (prototype), HTTP pipelining framework & browser

After diagram?

Capped?

application. Once at this stage, the framework will be made available to the open source community as the OpenPipe HTTP pipelining specification. This final version will be analyzed and compared in terms of speed with normal HTTP requests as well as the initial version developed before any improvements were made.

Awesome!

The final stage of this project will look for possible ways to implement this HTTP pipelining paradigm as a pluggable module available to web developers. This would allow web applications and web sites to take advantage of HTTP pipelining capabilities with little to no development and coding considerations. The end goal would be to produce an Apache module which injected HTTP pipelining logic into the request and response cycle, making the process more transparent to traditional web application developers.

E. Contributions

Capped?

The primary goal of this project is to provide an open source framework for web developers whom wish to utilize HTTP web request pipelining for their own web sites and web applications. This open source implementation will be made available as a software library which contains the necessary JavaScript and PHP classes for implementation of customized HTTP request pipelining from the ground up. This open source framework will also provide a standardized implementation from which extensions and improvement in the technology could be built ~~from~~.

In an effort to increase adoption, the project will introduce a pluggable approach to HTTP web request pipelining through the introduction of an Apache module, will allow for a transparent utilization of HTTP request pipelining in most web sites and web applications.

Lastly, this project aims to provide a clear and thorough explanation for why and how HTTP pipelining produces increased speed. This will be achieved through a comprehensive analysis and comparison of current HTTP 1.1 standards to HTTP pipelining techniques.

F. References

[1] "BigPipe: Pipelining web pages for high performance", *Facebook*, June 2010. [Online]
Available: http://www.facebook.com/notes/facebook-engineering/bigpipe-pipelining-web-pages-for-high-performance/389414033919 [Retrieved December 2010]

[2] "Making Facebook 2x Faster", *Facebook*, February 2010. [Online]
Available: http://www.facebook.com/notes/facebook-engineering/bigpipe-pipelining-web-pages-for-high-performance/389414033919#!/note.php?note_id=307069903919 [Retrieved December 2010]

[3] "Part of Hypertext Transfer Protocol -- HTTP/1.1 – 8 - Connections" http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html [Retrieved December 2010]

[4] "Optimizing Page Load Time", *Aaron Hopkins* [Online]
Available: http://www.die.net/musings/page_load_time/ [Retrieved December 2010]

[5] "What is HTTP pipelining", *Darin Fisher* [Online]
Available: http://www.mozilla.org/projects/netlib/http/pipelining-faq.html [Retrieved December 2010]

[6] Steve Souders, *High Performance Websites*, Oreilly Media, September 2007

[7] Steve Souders, *Even Faster Web Sites: Performance Best Practices for Web Developers*, Oreilly Media, June 2009

[8] Henrik Frystyk Nielsen, James Gettys, Anselm Baird-Smith, Eric Prud'hommeaux, Håkon Wium Lie, Chris Lilley, *Network Performance Effects of HTTP/1.1, CSS1, and PNG*, ACM SIGCOMM, volume 27, number 4, October 1997