12-2-2014

ParWeb: a front-end interface for cluster computing.

Jacob Potter
Western Michigan University, pttr.jcb@gmail.com

WESTERN MICHIGAN UNIVERSITY

Web User Interface Framework for High Performance Cluster Computing

Senior Design Fall 2014

Jacob Potter, Benjamin Johnson, Kyle Chipps

Web User Interface Framework for High Performance Cluster Computing

Abstract

The High Performance Computational Science Laboratory at Western Michigan University operates a cluster of systems for use by students, professors, and professional researchers. Currently users that want to access the cluster, known as "Thor", require knowledge of the Unix/Linux command line. They also require knowledge of operating a piece of software known as TORQUE to correctly achieve results from the High Performance Computing Lab.

The Web User Interface Framework for High Performance Cluster Computing alleviates the need for intimate knowledge of a command line interface, and replaces the current interaction interface with a more user friendly and graphical oriented system. This interface allows users with little knowledge of Unix/Linux, TORQUE, and even computing clusters to run programs specific to their research purposes with ease. The interface allows users to add, delete, and retrieve results in a fluid, easy to navigate environment.

The Web User Interface Framework for High Cluster Computing was designed, developed and implemented using the Extreme Programming philosophy of software life cycles. The client representing Western Michigan University's High Performance Computing Science Lab, Dr. John Kapenga is the main client to which all functional behavior of this interface is described. By request of the High Performance Computing Science Lab the project will be licensed under the GNU General Public License Version 2.

Web User Interface Framework for High Performance Cluster Computing

Table of Contents

Introduction

The Web Interface Framework for High Performance Cluster Computing is an interface for connecting users to the High Performance Computational Science Laboratory at Western Michigan University. Dr. John Kapenga represents the High Performance Computing Lab as our client. The interface itself was built with stability, reliability, and usability as top priorities.

High Performance Computing Lab (HPC) is located on the Parkview Campus of Western Michigan University; however the lab itself is used by students and researchers from many different areas of study. The HPC was created in 2012 with funding from the National Science Foundation. The main goals of the HPC are to assist research primarily in the sciences; such as physics, chemistry and of course computer science. Allowing users from all areas of expertise to access such a resource is crucial to the advancement of all research done in these areas at Western Michigan University.

The HPC itself is comprised of many separate systems called "nodes" that are connected through a network. The HPC uses a mixture of Graphical Processing Units (GPU) and other servers for the cluster. There has been a steep rise in productivity, efficiency and power of GPU's in the last few years compared to their CPU counterparts. As a result the HPC uses GPU's to handle the bulk of its processing power.

The specific setup of Western Michigan University's High Performance Computing Cluster is a set of 22 nodes of computing power. The main portion of the processing power the cluster named "Thor" is comprised of 19 NVidia 2800GT Kepler GPU's, with 1 Terabyte of available temporary storage space to be used by processing. There are other nodes with more specific

purposes, but all nodes using raid arrays and other preventative measures to keep Thor running smoothly.

The HPC uses a variety of technologies to operate its Thor. The main problem with the HPC is user accessibility. Currently users that want to schedule and run jobs on the cluster need a firm understanding of Unix/Linux environments and an understanding of the software used on the cluster, TORQUE, SSH, SCP/SFTP. Current users are forced to learn these new technologies, and run important tests and projects in an unfamiliar and occasionally a frustrating environment. The goal of this Web Interface Framework is to alleviate the burdens from the users who aren't familiar with these environments and software. By building a user friendly environment to use, any student, researcher or professor will be able to access the valuable resources the HPC has to offer, in a quick, safe, and effective manner.

It is important to note that, using a command line interface with Thor, and becoming an expert with TORQUE is the most efficient and most powerful way to run applications in a cluster environment. However it would be acceptable to lose a small amount of performance and functionality to widen the user base exponentially.

Thor uses a job scheduler known as TORQUE. This program handles interfacing between users and clusters. A user normally connects using Secure Shell, and runs commands from a command line. The Web Interface Framework will be communicating with TORQUE for the user. This will allow more user accessibility and a graphical interface for interacting with Thor, and potentially other cluster computing operations.

There are existing projects with similar intentions available through open source, or commercial means. The closest related project available is called FORCE written by Matteo

Ragni. This project was developed for more advanced users of a TORQUE system, but is nonetheless extremely valuable in providing information on how to communicate with a cluster system.

There is also a commercial product eQUEUE; it appears to provide many of the resources and functions the HPC Lab wants, but at a cost. eQUEUE is also proprietary software, meaning the software doesn't adhere to the licensing requirements laid out by the client. eQUEUE will not be discussed further in this report.

The Web User Interface Framework for High Performance Cluster Computing was designed, developed and implemented using an agile programming method known as Extreme Programming. This method of professional software development stresses the importance of communication between client and developing team. It also advocates for shorter release cycles, and a different approach to day to day operations compared to the classic "Design, Implement, Test, Release" software cycle.

Extreme Programming has some unique concepts associated with it, and "stories" are the fundamental basis of Extreme Programming. A "story" is created by a user, known as the client. These stories are functional descriptions of what the client wants software to do. Stories can be modified, added, or deleted at any point in the Extreme Programming philosophy. However with each story follows another concept known as "risk".

Risk is defined in Extreme Programming as a measurement of the likelihood a particular feature is feasible. The risk of a piece of functional behavior is generally not noted by the client, and created by the development team. The goal of both developers and clients is to create software according to the clients need, with the lowest risk possible. Risk is generally implemented on a 1-10 scale. Risk can be reduced through use of another Extreme Programming

principle known as “spikes”.

In Extreme Programming, spikes are small, simple programs that produce a result which can be used to identify risk of a story. These simple programs only purpose is to demonstrate design decisions. The spikes themselves are usually not good enough to add into the final product, but are essential to the overall design of a project.

The resources needed to implement the Web User Interface Framework are all open source and available, free of cost. All of the distributions, libraries and frameworks that are involved with this interface have been chosen to produce an environment that is stable, user friendly and reliable.

Background

Some background information is required before diving into specifics of how this project will be fully implemented, such as the client in question, the particular problem(s) being addressed by this project, background of the system that will utilize this project, and other existing solutions along with their applications and relation to this particular project. The client associated with this project is the High Performance Computational Science Laboratory at Western Michigan University (HPCS), working with Dr. John Kapenga directly. The problem that was addressed by the software was usability of laboratory's computing cluster by users unfamiliar with the system, opening the benefits of high power computing to a larger scientific community by building a framework that can be used on any such system. This is implemented using a web user interface for cluster access and TORQUE scripting to run data through programs written for the system in question. There exist projects that attempt to solve this issue somewhat, each with their own merits and flaws. One such project is FORCE by Matteo Ragni, which is an open source web based user interface for managing 'jobs' on the HPC cluster.

HPCS

The HPCS Lab is used by students, professors and researchers alike and is represented in this project by Dr. John Kapenga, a computer science professor at Western Michigan University, who will supervise the project to Release 1 and beyond. The following passage is the mission statement from the HPCS website and will help give some more background to the goals of this project:

> HPCS houses a high performance computation cluster installed in 2012 with funding from the National Science Foundation, to support interdisciplinary projects in

> computational science and engineering. The cluster provides a unique resource as a hybrid computing environment for distributed computations between nodes, multi-threading within individual nodes and massively parallel computations on graphics processor cores.
>
> Target work on the cluster includes basic computer science research with parallel algorithm development and implementations, and furthermore projects in high energy physics, composite materials research, carbon nanotubes modeling, computational fluid dynamics, flexible body aerodynamics, and chemistry simulations for modeling the nature of chemical bonding and for a computational study of enzymatic catalysis.
>
> High performance computing also permeates many facets of education. Conforming to WMU's mission as a learner-centered university, the HPCS infrastructure can set the stage for developments in high performance computing education and training at the institution, by facilitating training on cutting-edge computing facilities across multiple departments in the CEAS and CAS colleges, and by engaging student participation in the daily operation and maintenance of the system. [1]

In the mission statement, it is stated that the cluster is used across the university by multiple departments and colleges. Some of the research being conducted using the cluster includes Exploiting Multi-Core/Hybrid Parallelism (Elise De Doncker, John Kapenga), Computational Orthopedic Biomechanics (Peter Gustafson), High-Resolution Computational Fluid Dynamics (William W. Liou), Computation of Feynman Loop Integrals by Iterated Adaptive Integration (E.

[1] "HPCS." High Performance Computational Science Laboratory. Western Michigan University, 2012. Web. 18 Mar. 2014. <https://cs.wmich.edu/~hpcs/>.

de Doncker) , Parallel Computation of Flexible Body Aerodynamics (Dewei Qi), Uses of Multiprocessor Cluster Power in the PCI department (Paul Dan Fleming), and Supercomputing for Chemical and Biological Problems (Yirong Mo). The experiments being conducted are calculating large amounts of data which would be extremely difficult and time consuming without the HPCS lab. As is evident, this cluster is used by people of many different fields, educational backgrounds and familiarity with computing in general. This is where the problem lies, how to make the cluster easily usable by the diverse range of possible users.

Problem

High power computing is beneficial not just to those in the Computer Science field, but to those in Biology, Chemistry, Physics, Mathematics, and many, many others. However the majority of possible users of this cluster system are not necessarily familiar with even basic programming, let alone the extremely complex command line tools required for computing the mass amounts of data the system is designed for. This particular problem is solved by having applications written by developers for specific experiments, but this still presents another problem. At the moment, if a user wants to have their data processed by a program on the cluster, there is no simple interface for them to do so. In most cases they must send their data to an administrator to create a TORQUE script to run this data through the software, and then send them the results. This method is rather off-putting to possible users, and if there were a simpler way to do this then the system itself would most likely see more widespread use. This is the problem that this project addresses; a web based user interface makes the task of submitting data to the cluster and getting results much simpler and more user friendly. However, before getting into details about how this solution works, a basic understanding of high performance computing

is required, which will be explained in terms of how it is related to this project and why are they important.

Technical Background

High performance computing is the use of mass amounts of processors to cut the time calculations of large amounts of data quite considerably. The way this is done is using several computers, all with their own individual components (processors, RAM, hard drives, etc.) and linking them together in such a way that they act as one system. This is what a computing cluster is. Each computer in the system is called a computing node, which can be anything from a high powered server to a personal computer. The nodes are connected by a high speed LAN, and uses software to make the cluster behave as one computer. The use of several high performance nodes in a cluster has many uses, but in the context of this project, clusters which take advantage of parallel processing capabilities are the most relevant. In parallel computing, a program makes use of multiple processors to run multiple computations at once, cutting processing time of a program considerably and when combined with a cluster of computers each having several processors, will yield a system that can compute large amounts of data quickly and efficiently. A system with so many resources needs to be balanced well across the cluster, as to not overload a single node. With this in mind, a cluster system is usually managed by software that allocates computational tasks of a specified program into 'jobs' among the nodes of a system. This software is usually controlled by scripts written by a user to run their own programs on the cluster with certain specifications, usually containing some environment variables and some UNIX BASH script. Some common batch control systems in use are OpenPBS and TORQUE.

[2]Today many modern high performance systems make use of GPU's as opposed to traditional CPU's. Now GPU's are designed for graphics, but the way the GPU processes data into graphics is by arithmetic on several numbers at once, which usually correspond to locations on a screen and other relevant data. However, through use of architectures like CUDA, these properties of GPU's and the fact that modern GPU's are becoming quite more powerful than their CPU counterparts, using GPU's for these clusters allows even more data to be computed efficiently. [3]

Thor

The high performance computing cluster used by the laboratory is called Thor, which is implemented with parallel processing across several computing nodes using GPU's for processing data. The specific setup of this cluster isn't very relevant here, but the basic architecture is; the cluster has 22 nodes using NVidia Kepler K20 GPU cards, with each node containing a 1 Terabyte hard drive for temporary storage of computational work, 128-512GB of RAM and 2-4 Sandy-Bridge Xeon Processors (the first two nodes – designated Node 0 and Node 1 – have four processors, Node 0 having 512GB of RAM, and Node 1 utilizing 256GB). The cluster is bound together using an "Ethernet management network". All nodes utilize a network mounted directories on a RAID file server for storage. For user jobs, a main head node (Node 0)

[2] "TORQUE Resource Manager." TORQUE Resource Manager. Adaptive Computing, 2014. Web. 18 Mar. 2014. <http://www.adaptivecomputing.com/products/open-source/torque/>.

[3] Jason Sanders, Edward Kandrot. CUDA by Example: An Introduction to General-Purpose GPU Programming (1st ed.). Addison-Wesley Professional, 2010.

is “used to manage the entire cluster and share programs and data”. Thor uses the TORQUE resource manager for control of batch jobs and distributing tasks among the nodes, and the use of these TORQUE scripts is one of the central tenets of this project which is to create an interface that automatically generates these scripts for multiple users and programs to run on said cluster. [4]

Design Decisions

Now that some basic understanding of the system has been addressed, the solution to the problem in question can be discussed. To start, consider figure 1 below, which gives a basic layout of how a basic user currently utilizes the cluster. The user remotely logs in to the cluster via SSH. The user must then use a command line text editor to create their own TorQUE scripts, sftp/scp to upload their input data(if there is some), and then download their results manually using sftp and scp). This method can present a rather troublesome learning curve as the user would be required to be familiar with these command line tools, of the TorQUE job manager (parameters, etc.), and UNIX/Linux in general.

[4] "Cluster Description." High Performance Computational Science Laboratory. Western Michigan University, 2012. Web. 18 Mar. 2014. <https://cs.wmich.edu/~hpcs/cluster/>.

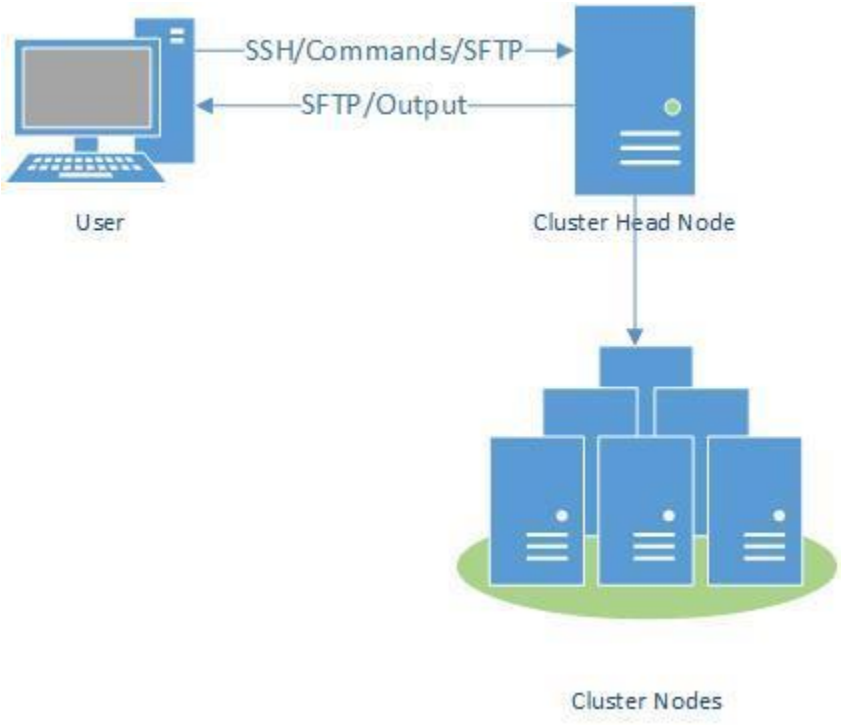

Figure 1

Figure 2 below illustrates how a ParWeb interface allows users to more easily utilize the cluster, eliminating the learning curve. The user logs in to the web – interface, initializes a new job (uploading input data if it exists), and then downloading results. The user has no need to know TorQUE or any command line tools; they just need to be familiar with the interface layout which will be easy to learn for any regular web user. The site saves the data on the web server and automatically generates a TORQUE script and sends it to the cluster. The web server (hosted separate from the cluster for portability) will run the TORQUE script along with the uploaded data on the cluster, which will save the results in a directory on the web server. Part of the TORQUE script generated will contain email information of the user, and will send them an email (if this feature is enabled on TorQUE, handled by sysadmin). However, the user can also

check the status directly in the interface, as well as view the results directly on the site and download results as well.

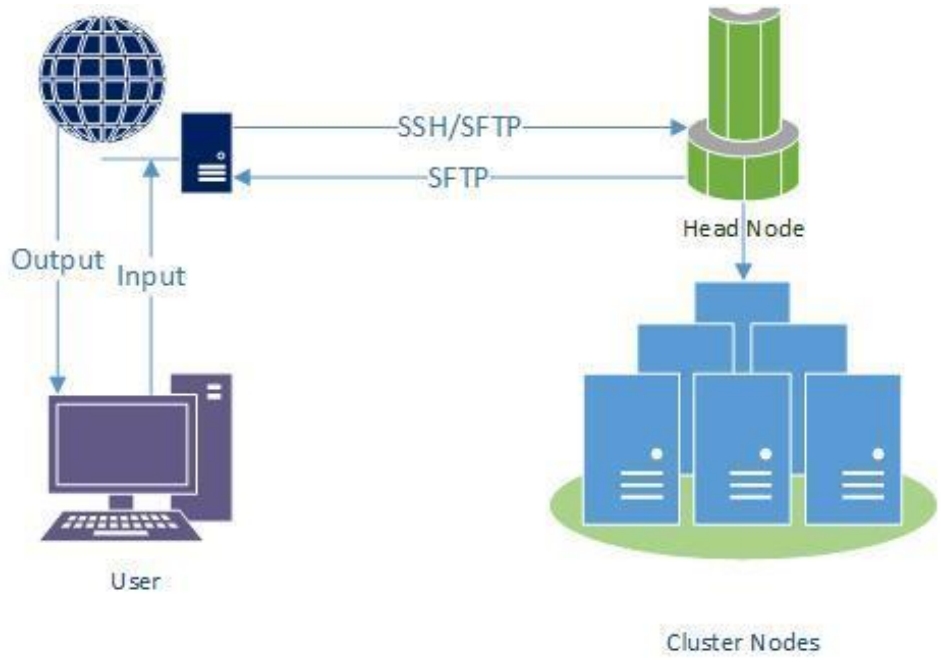

Figure 2

This model cuts a lot of overhead out of the process, and will be useful for users that have little knowledge of TORQUE scripting or even computers in general. This basic user interface is the basis of the first release of this project. Future release hope to have more advanced TorQUE control from the interface. As of now, the framework consists of some basic modules to facilitate communication between the web server and cluster. The workflow of a ParWeb application creation is as follows. There exists a cluster application that users need to access, a flu epidemic simulation for example. Figure 3 below helps illustrate this workflow

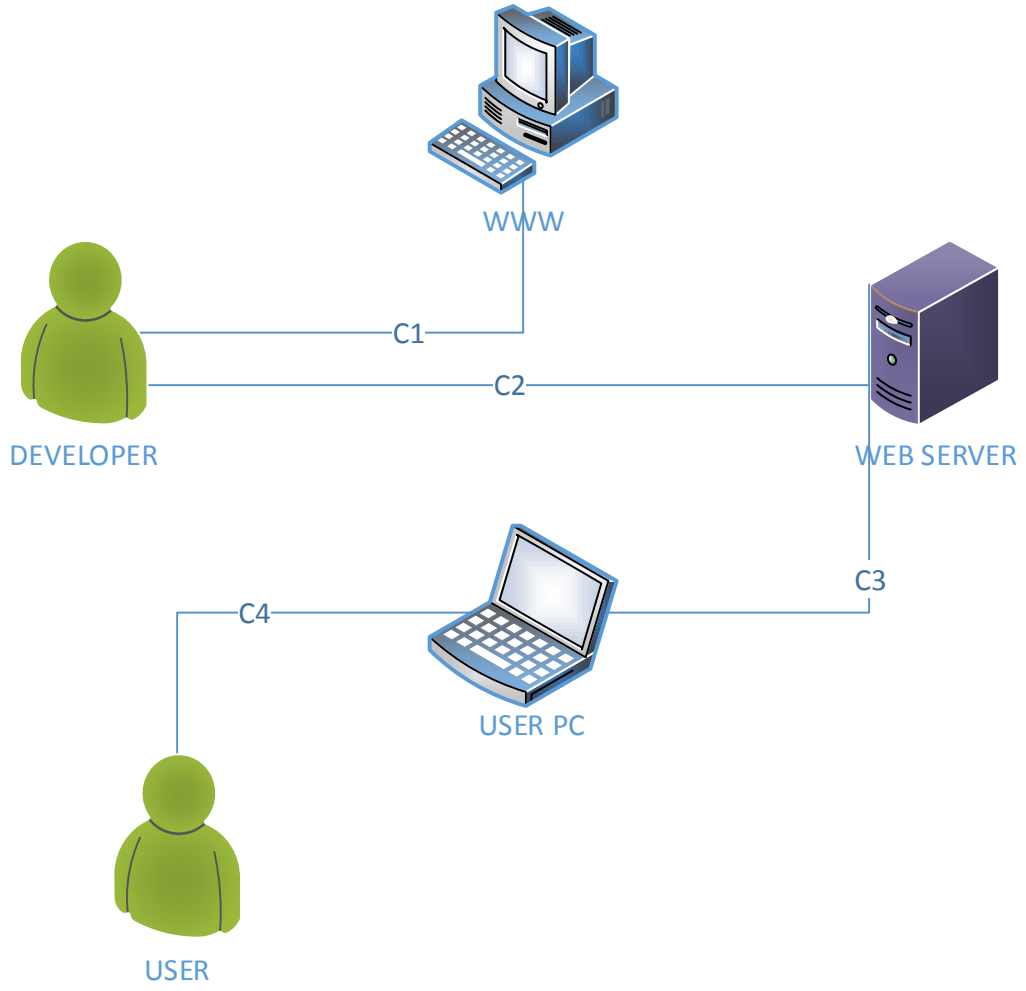

Figure 3

A web developer will develop an interface from ParWeb to utilize the specific cluster application (downloaded from the web-C1). The ParWeb framework at the moment is just a skeleton application that utilizes SSH modules to communicate with the cluster, so a web developer just needs to customize the initial ParWeb application to the cluster application in question (setting cluster address, specifying file locations, TorQUE script contents, etc.). Once the application has been setup, it is deployed to the web server-C2. A user may now access the cluster application

via the web-C3&C4. The next release hopes to take advantage of Rails scaffolding, a simple command to generate a skeleton application automatically.

FORCE Project

Upon initial research, the FORCE Project was found. The project was a good example of interfacing with a TORQUE powered cluster, however it lacked the functionality that was required. The original project functions more as a job manager than an end-user interface geared more towards cluster administrators and advanced users. The interface has options to view the queue and delete jobs, create TORQUE scripts using fields dependent on user entry, the ability for the user to upload scripts already written, and a file directory interface to view and run files/scripts already in the user space. The interface is specific to TORQUE scripts and really isn't customizable to other non-cluster uses. The biggest difference however, is the lack of auto-generation of scripts for basic users, which counts this out as a possible solution to the main problem this project resolves, usability. Without this auto-generation, the goal of making the cluster utilizable to a larger group of users is rendered moot. Below are a few screenshots of the original FORCE project, which will help illustrate some of the similarities between this project and FORCE. [5]

[5] Ragni, Matteo. github.com/MatteoRagni/Force. 18 July 2013. December 2014<https://github.com/MatteoRagni/Force>.

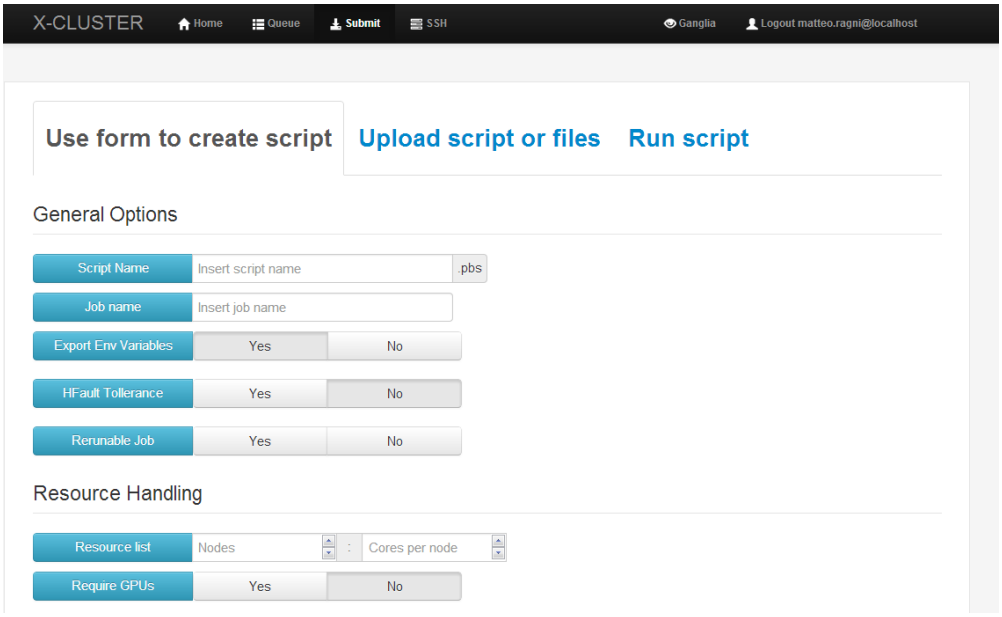

Figure 5 TORQUE Script Generation [v]

The screen shown in Figure 5 demonstrates the fields used to generate the TORQUE scripts in FORCE. Parweb generates these scripts automatically as opposed to this implementation.

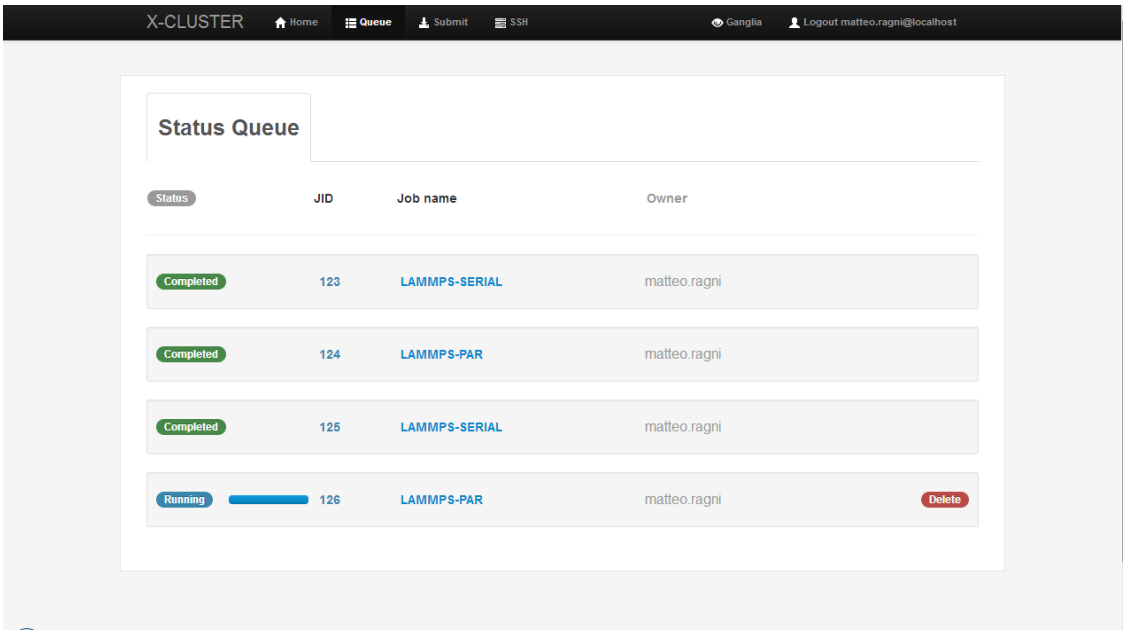

Figure 6 Queue [v]

Figure 6 illustrates the job management available with FORCE. In this project, job deletion control is lost, and users are allowed access only to job information for their own jobs.

Operating Systems

ParWeb needed to be keep as many things as uniform as possible. One such instance is the decision of an operating system that ParWeb will run on. There are many different Operating Systems to choose from and the decision process will be outlined in this section.

There are three main families of operating systems available to use on this project. Those are: Microsoft, Apple and Linux. Choosing a family will determine the type of operating system that will be used in this project. Each family has its strengths and weaknesses.

First up is Microsoft, specifically Windows Server. Virtually everyone is familiar with a windows environment and using Windows may seem like the correct choice at a glance. Getting to some details about the Windows Server Operating System, the GUI is easy to use, and the server manager is easy to navigate and control. Windows Server is used in many enterprise applications. Windows Server has a favorable release cycle, which releases major versions every few years, and in between all the necessary security patches and general small enhancements. However there are drawbacks with Windows Server that do not allow ParWeb to further consider Windows Server for use. The first and main concern is the cost. For what ParWeb would need Windows Server to do, the cost of a Windows Server license is over $500. There are different types of licenses available from Microsoft; however none fit ParWeb for the following reasons. First, Thor and the HPC Lab are almost exclusively Linux based. Implementing a Windows Server in that environment would be more complex than necessary. The latest version of Windows Server has many complaints from home server maintainers, to enterprise systems alike.

Such as features that require .NET 3.5 may not work since Microsoft has moved to .NET 4.5 in Windows Server. This outlines the difficulties in looking for plug-ins or add-ons that may help the project move along. As such the use of a Microsoft based server environment was ruled out almost exclusively because of cost and the environment Thor and the HPC Lab operates in.

Then there is OS X Server edition, an operating system from Apple Inc. which is very similar to the OS X desktop operating system, with added features that allow for server related functions. OS X and OS X Server implement a Unix/Unix Like kernel which is an important plus over Windows Server as it keeps the environment close to uniform. However the main reason to use OS X Server over a Linux based operating system is if the use of Apple Products is in the environment. Linux can do all the things OS X and OS X Server can do outside of clients that don't use Apple Devices. Since ParWeb does not contain "users" in the sense that it would need to administer users on an enterprise like network, with specific file permissions and access to software, the use of OS X Server was ruled out.

That leaves Linux. Linux is used by the HPC Lab and Thor, and by the administrators that maintain Thor and the Lab itself. It is very popular among many developers, both professors and students alike. Linux is assembled under the free and open-source software ideology. Linux is widely regarded as the most efficient for clusters and high performance computing. Many clusters and supercomputers around the world use Linux. The source code for Linux is available for any purpose that a user would need. Typically Linux is packaged with other software into what is known as a "Linux Distribution". There are many and more distributions to choose from; however ParWeb was requested by the client for a specific distribution.

At the request of the client, the web interface framework was built to run on an Ubuntu 14.04 machine. This version was also chosen by the client. Ubuntu is a "debian based" operating system, developed both by open source developers and Canonical Ltd. This operating system fits well with the decision to keep all the software from this project Free and Open Source. Ubuntu is one of the most popular Linux distributions both by "debian based" and overall standards. Versions of Ubuntu are generally updated every six months; however Ubuntu does release LTS (Long Term Support) versions periodically. Ubuntu 14.04 is considered an LTS Operating System. This furthers the notion that the system will be stable operating under Ubuntu 14.04. Unlike its Windows and Apple counterparts there is no cost for this operating system, and it has one of the largest Linux market shares.

In the end the cost, implementation, familiarity, and the uniform of environments were the most important factors in choosing an operating system. Of these a Linux based operating system is a clear winner. Of the thousands of Linux distributions, the client, Dr. John Kapenga and the HPC Lab requested Ubuntu 14.04 LTS to act as a medium between users and Thor.

Web Servers

ParWeb does not make use of an entire operating system. In fact, it mainly uses one key part of a server. That part would be the Web Server. A web server is a system that process requests over HTTP. There are three main web servers available. The two web servers that share the majority of the market share are Apache and Information Internet Services. Nginx is another web server that isn't as popular as the previous two; however was still very much a viable option for ParWeb.

The first web server ParWeb ruled out was Information Internet Services (IIS). This web server is from Microsoft, and shipped with the Windows Server operating system. IIS has been around a long time and is one of the most popular web servers in production today. However to use IIS you must purchase the Windows Server operating system, which rules this out as an option for ParWeb.

Apache is the worlds most used web server. It began in 1995 as a project based on NCSA HTTPd Server. Shortly after its creation, it was the most popular HTTP Server. Apache is shipped with many Linux Distributions as the default web server. This is also the case with Ubuntu 14.04. One of the most important features that Apache holds over IIS is the amount of plug-ins and extensions made available at no cost. The extensions to Apache enhance its ability to be used with numerous amounts of applications and services. With the default web server on Ubuntu being Apache, and the administrators at the HPC Lab suggesting Apache, ParWeb was designed for an Apache Web Server.

Another notable mention in web servers is Nginx. Ngnix shares many same concepts with Apache. It is open-source under the BSD License. It has been known to have a performance advantage over Apache, (and IIS) however it does not have the features and user base that Apache does. However in the future releases of ParWeb, it could be that switching to Nginx would be favorable.

Web Applications Frameworks

ParWeb was not built from a blank screen. ParWeb was built on a web application framework. This design was decided at the beginning with the client and the HPC Lab involved. There are thousands of frameworks available in almost any language today. The languages that

the HPC Lab preferred we use was PHP or Ruby. These restrictions were placed because of the hardware, and software that had been decided. Choosing the best framework for a project can make the project go smoothly. Deciding on a framework that does not give you the features you need would cause more problems than it solves, generally. The important factors in choosing a framework are things like, a strong user base, good documentation, testing suites, integrated unit testing, regular security updates and developer interaction. Good frameworks are actively developed on and easy to use and understand. Of the frameworks in PHP and Ruby, the main two contenders are Laravel and Ruby on Rails.

Laravel

Laravel is a free and open source PHP Framework. Laravel is based on the development model of a MVC Framework. Its source code can be read on GitHub, and of the web applications using PHP Frameworks is one of the most popular. Laravel has a built in packaging system that allows code to be easily added to other applications. Laravel allows for database migration and unit testing within the framework. Laravel is a powerful tool in building web applications by making a lot of the development side of building web applications easy. The syntax is clean and easy to understand, by following some of the extensive documentation that is available for laravel; it is possible to go from nothing to running project in very little time. There are active developers that are updating the code base to Laravel very frequently. Laravel is still fairly new, as the initial release was in 2012. It has grown very quickly, but Laravel's main competition seems to be with the older, but more popular Ruby on Rails.

Ruby on Rails

The Ruby on Rails Framework was initially release in 2005, but in the last few years has seen exponential growth. The Rails Framework is built on Ruby, a programming language. One of the main tenants of Ruby is convention over configuration. Basically this means that the programmer need not spend time configuring files for setup. Ruby and Rails follow this principle very well. Rails, like Laravel utilizes the MVC pattern to organize its application. Rails is RESTful by default and its program structure is similar across any type of project. Rails and Ruby both are very well documented, and both have been on the forefront of development since their beginnings. Rails has very well built testing suites for any application that gets created. This makes it easier to rapidly create good working applications. Rails has specific rules about naming schemes and syntax that allow for a uniform style to be followed in all Rails projects, making it easy to hand to another group of developers and keep active development up. There are negatives to Ruby on Rails; the learning curve of this framework tends to be quite high, because of all the rules about naming of files, databases and classes. Ruby on Rails does not support all types of website hosts (it does support Apache). The scalability of Ruby on Rails is not as good as Laravel as of yet. Scaling up Ruby on Rails applications is possible, but harder to do than other frameworks.

In conclusion the choice between Laravel and Rails is very close. Both offer many strong concepts and tools for web applications. ParWeb would have run on either, with no problems. However in the light of an exponentially expanding user base, immaculate documentation, and easy to understand syntax, Ruby on Rails was chosen for this project.

Capistrano

There is a small amount of work to be done when deploying a Ruby on Rails application. Fortunately there are tools to make this as quick and simple as possible. One of those tools is Capistrano. This tool allows a developer to automate steps during a deploy process such as, copying code from a development machine to a production server, updating system and server configurations, and the restarting server. Capistrano also allows for setting up of databases, manage the application server, and manage Ruby versions, along with a myriad of other features. Capistrano is widely used by the Rails community and is actively developed on.

Phusion Passenger

Getting the code to the server is one part. Actually running the application is another, and that is where Phusion Passenger comes in. Of the two main web servers that Ruby on Rails supports, Apache and Ngnix, there needs to be some modules that allow those services to use Ruby on Rails applications. With a Laravel application, one would install mod_php on the web server, which would allow the application and the web server to interact cleanly. This is exactly what Phusion Passenger does. Passenger allows a Rails application to run effectively on a web server. When Rails applications are running, they run as a persistent process, as starting from scratch takes a large amount of overhead. What Passenger will do is keep track of the Rails stack in such a way that creating additional processes is faster than waiting for a new instance to start from the beginning. Passenger does this by re-using the Rails stack and managing shared memory.

Devise

Devise is a module for Ruby on Rails applications that allow a developer to add authentication to an application. It is comprised of ten modules that handle everything from database encryption and decryption, to IP Address tracking and resetting password functionality. ParWeb uses devise to handle user creation and authentication in a clean and simple manner. Devise is very feature rich and has many uses, however not all the features need to be used, and many can be included for future development. This speaks to the flexible nature of Devise, in that it allows a developer to use only the parts any one application requires, with the possibility of adding more features in later.

SSHKit

SSHKit is a part of the Capistrano package. It is a feature of Capistrano that allows the running of commands on one or more servers. The main communication of ParWeb and Thor uses this feature to pass files, and run jobs on the cluster. With SSHKit, ParWeb can use SSH/SFTP/SCP easily on any server that allows that kind of interaction.

Implementation

Introduction to Extreme programming

Extreme programming is the software development strategy that requires balancing both the customers' requests and the development teams time to build a piece of software that meets the user's needs. Extreme programming works to meet the need of both "business" and "development" where development is the team of programmers and business is the customer that is requesting the applications and features. In any project, communication is vital and extreme programming uses communication as one of the many core building blocks to help verify the success of a project. Another core concept of extreme programming that may differ from a "regular" software development cycle is the use of short release cycles to help reduce the risk involved with implementing a certain feature.

Risk

By assigning a risk value to each feature we can determine a realistic time frame that a requested feature can be solved in. Risk is assigned on a scale from one to ten where a very low risk task would have a value of one and a high risk task would have a value of ten. For example development is given a large file of customers with their name, address, amount of money owed to the company, sorting this file by name alphabetically then sending a bill to each customer starting at the beginning of the list could be a simple task for any part of the development team and could be categorized with a risk value of one to five (assuming the file is small enough). But often times there are risks that are not seen by the business side that are very apparent to the development side for example if business wanted us to sort this file so that they could find the shortest route to take to each customers address and hand deliver the bill this could easily be an

example of a NP-hard problem and a risk value of nine or more would have to be assigned to it once again depending upon file size.

Stories

Stories are given by the business side and they are functional examples of what each feature should do and any requirements that the overall system will need to have. If each feature has an appropriately assigned risk value then this can give both parties a better idea of the time it will take to implement a feature if development is having a problem understanding the story they can always ask business to rewrite it and if it is too big development can break that story into smaller stories. If business has a new feature or change they want to make to an old story they can go ahead and rewrite it at any time. The time unit measurement used for estimating a story are referred to as "ideal engineering hours" which can be defined by the time it would take to implement just that feature with no interruptions. [6]

ParWeb User Stories

1. A feature that is of great importance to the customer is the ability for our framework to work with an apache web server on an Ubuntu Server. This is a very low risk and easy to implement feature since apache is an open source and free of cost web server things like documentation and general information can easily be found on the Internet. The same goes for the Ubuntu operating system so a risk value of three is assigned here. The time it

[6] Kent Beck, Cynthia Andres. Extreme Programming Explained: Embrace 2nd Edition. Addison-Wesley Professional, 2004.

takes to implement this requested feature is only about 3 hours which come from the time it will take to document the configurations of the server and web server.

2. The client has also requested that our framework be very lightweight and that it requires little overhead and server resources along with the use of apache for lightweight and portability the client has asked that we use Ruby on Rails. Although there is no problem in using these technologies there is always going to be a very large risk involved with learning a new programming language. The risk value that was assigned here is a seven since some of the group has experience with similar languages while others have none. The learning of a new programming language could take a whole semester since each member of the development team has mastered at least one programming language this should only take 40 hours for each member to learn.
3. The client has requested that the framework be stateless or RESTful, this means the developer can send a GET, POST, DELETE, or PUT request to a server URL then the server will respond depending upon the verb and URL requested. This will give the person who is developing the application the ability to easily send information using a JSON format and require that the server store no information related to the client. This will be assigned a risk value of nine since the design and implementation of a RESTful API is something that can rise to many opinions and spark much debate. The risk value is also very low since Ruby on Rails is RESTful by default.
4. One functional behavior that our framework will implement is the ability to easily upload, monitor, and view output files from a program that is ran on the cluster itself. Since this is a rather large story we can break it down into fewer easier to work with stories instead.

a. The ability to easily upload data to the cluster comes with risk value of 5 since there are libraries for this openly available in Rails the but making library functions work for what we need will take some time and code. The amount of time allowed for the implementation of this feature is one day or twenty four hours.

b. Being able to view the output files from the web interface will be a little easier to implement and has low risk value of one this should only take about three hours since the aforementioned libraries have very similar functions. A picture of what this may look like can be found in figure 9 below.

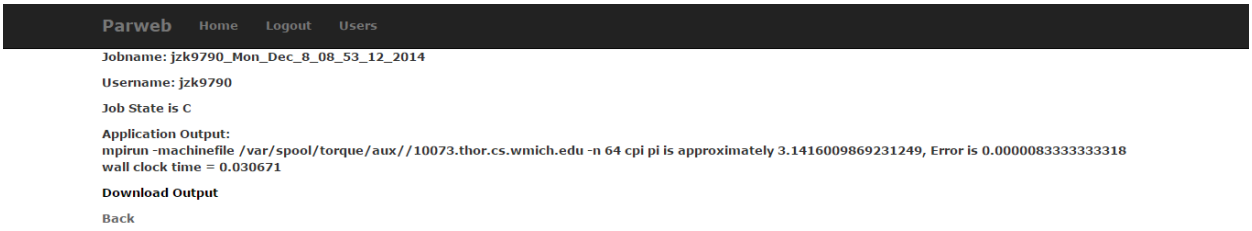

Figure 9 Parweb Output page

Figure 9 shows the user the results of their run job, the user will also have the option of downloading the results in the appropriate file format specified by the cluster application.

5. The client requested that the transferring of any kind of data be done via SSL since the apache web server is being used and enabling SSL on apache can be done through the editing of a configuration file the risk value assigned here is one. Once again the documentation of the configuration files will be needed so time allocated for this story is one hour.
6. Although we decided for release one that using SSH to authenticate will be ok, in our version of release two there will need to be a user database where a access hierarchy can be stored in order to verify what users can run what application with what privileges i.e. if you're a student you will not be allowed to run an application that uses every node while a professor may have this kind of access. Since the implementation of this will be completely from scratch a risk value of six is assigned here and it will take about 35 engineering hours.

Testing

The testing suite in Rails was built along with Rails, not as a side project or as a feature later in development. Testing was always important to the Rails Development team. Every new Rails Application has a testing suite built in. There are three different environments in a Rails application, testing, development, and production. The default suite allows for testing of the controller, view and model, separately so that, each component of the MVC framework is tested independently. To test a Rails application good test data needs to be added to the test database.

To add test data to a Rails testing suite, sample data needs to be added. This sample data is called a "Fixture" in the Rails environment. Fixtures allow for adding data to the testing database with predefined data. Fixtures are written in YAML Ain't Markup Language (YAML), which allows for a human readable format of data that integrates with the testing suite quickly and cleanly.

Unit testing is done to test models. When models are created with the scaffolding feature of Ruby on Rails, it is possible to test those models by adding tests to the file that gets created in the model directory of the testing structure. Run the tests using the "rake test" command and a summary of what passed and failed will be output on the terminal. There are many assert functions available in the Rails testing classes. They are all documented on the Ruby on Rails guide site.

Testing controllers is a little different than testing models. Testing controllers is called functional testing. Some questions that can be answered using these functional tests are: was the web request successful, was the user redirected to the correct page, and was the user

authenticated correctly. The tests themselves are similar in layout to the tests in the model. It is possible to use get, post, put, delete, head, and patch requests in functional tests.

After testing the controller and model, the view can be tested using a variety of methods included in the testing suite such as “assert_template” method. This allows a developer to know what gets sent to the user after an interaction. It is also possible to test the layout of a page using that same method with some parameters specifying the layout.

Security

Like in any web application, the security must be thoroughly looked at before deploying the application to production. It is very important to keep the web application secure for all the users involved. However Ruby on Rails has many built in features that mitigate or completely remove common threats to web applications, threats like SQL Injection, and session hijacking. This section will briefly go over how Rails protects against a variety of attacks.

The first is session hijacking. HTTP is a stateless protocol, and sessions make it stateful. Without states users would have to authenticate on every request. Sessions start by a user authenticating with a server, and once that authentication takes place, the session becomes valid. The session identifier is stored in a cookie that is passed between the user and the server. If an attacker were to gain access to that cookie, the session for the attacker would appear to be as the valid user. There are a few ways for an attacker to gain access to a cookie. An attacker can sniff out packets over an insecure connection, use XSS, cross site scripting attacks, or find a public terminal with a user's cookies still stored on the machine. Rails uses CookieStore by default for session storage, making sessions cookie based. The session hash is saved on the client-side rather than a session id. By adding "reset_session" to the Sessions Controller, Rails will issue a new session identifier after a successful logon, which helps against fixed session identifier attacks.

Another type of attack is called Cross-Site Request Forgery (CSRF). This attack works by adding malicious code or a link in a page that accesses a web application, and also that the web application believes the user to be authenticated. To protect against this attack, Rails uses a required security token that the web application knows but no other sites. By adding

"protect_from_forgery" in the controller, Rails will automatically include a security token in all forms and Ajax requests. If the security token doesn't match, the session gets reset.

Another important attack that is specific to this application is adding executable code in file uploads or downloads. For example if a user intends to download a file from the server, that user, on a vulnerable system could ask for a file outside of the document root folder. By using basic sanitization techniques, this can be avoided. Rails has methods that allow a developer to check if files are in certain directories. It allows a developer to control where users can download or upload files from or to.

Legal

The Web User Interface Framework for High Performance Cluster Computing Project by request of the High Performance Computing Lab of Western Michigan University will be classified under the GNU General Public License Version 2, 1991. The High Performance Computing Lab of Western Michigan University has stated that a Non-Disclosure Agreement is not needed for this particular project. The license itself states the main reason why any project would want to use the GNU General Public License in the Preamble**:**

> The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.
>
> When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things. [7]

The GNU General Public License provides the High Performance Computing Center the freedom to continue to add and remove to the project as they see fit.

[7] Free Software Foundation, Inc. The GNU General Public License version 2 (GPLv2). June 1991. <http://www.gnu.org/licenses/gpl-2.0.html>.

Resources

The Web User Interface Framework for High Performance Cluster Computing has a list of specific resources needed to operate correctly. The following is a list of required modules.

A Debian based GNU/Linux Distribution:

> Debian is an operating system preconfigured with software, most of which is also classified under the GNU General Public License. This operating system is an effort built by a group known as “The Debian Project”. The Debian Operating System's initial release was August 16th, 1993, and builds upon a rigorous definition of specific rules about what this operating system is allowed to contain.
>
> The rules “The Debian Project” adheres to is a set of rules defined in the “Debian Social Contract”. The rules that most pertain to this project are as follows, “Debian will remain 100% free”, “and We will not hide our problems”.
>
> The first rule, pertaining to the cost of Debian is important to this project because this will help the longevity of this project. The second rule is especially useful when reporting bugs and finding work-around for any and all problems related to all the supported libraries and packages that are included with all Debian Distributions.
>
> Overall the Debian system puts stability and reliability as top priorities. These priorities define the goals of this project perfectly. The Web User Interface Framework for High Performance Cluster Computing's top goals is to be stable

and reliable. All of the libraries, modules and distributions reflect those goals. [8]

HTML 5:

Hypertext Markup Language is used for presenting content on a website. From its beginnings in 1990, the HTML project has had the goal of being easy to read, while being efficient for computers and other devices. HTML 5 is the newest release of this language and provides many useful tools such as API's and error handling that previous versions do not have. (This project uses Twitter Bootstrap)

Ruby:

Dynamic, object oriented programming language. Designed in 1995 with the phrase "convention over configuration" at the front. Ruby's goals are about making programming interesting and fun for developers, and at the same time including an easy to learn syntax that is both efficient and clean.

Ruby on Rails:

An open-source web application framework. Based on the MVC pattern of development and released initially in 2005. Ruby on Rails has gained popularity over the last few years over other web application frameworks, like Laravel and other PHP based frameworks. Many large sites like, Github and AirBnB use Ruby on Rails.

OpenSSL:

[8] Debian group, "About Debian." Debian. N.p., 8 12 2013. Web. 18 Mar 2014. <http://www.debian.org/intro/about>.

OpenSSL is an open-source version of SSL. It provides cryptographic functions and validation for web servers.

JQuery 1.9:

JQuery is a JavaScript library for client side scripting in HTML. This library provides the necessary functions needed to be done on the client side. JQuery is supported on Firefox, Chrome, Safari, Opera and Internet Explorer. (Required for Bootstrap.)

TORQUE 2.5.12:

TORQUE is a resource management system for controlling processes on high performance computing clusters. TORQUE was based on the original PBS project which was related to high performance computing job management. TORQUE is also open source and one of the leading job management software packages for clusters available.

Other Software/Packages/Libraries Required:

a. bootstrap.css
b. bootstrap.js
c. devise authentication library
d. Capistrano deployment toolkit
e. Ssh toolkit

Summary

In conclusion, what this project hopes to achieve is simplicity, making the use of a highly valuable asset in the scientific community more available and usable to a broader user base. The framework that will be the final result of this project will be a powerful tool for universities, research facilities, and corporations alike. Its basic functionality can be used for a broad array of applications, and is lightweight, efficient, and simple enough for the best usability. But is this vision possible? Can this project reach its goals and have a stable release? The answer to that question is yes. Whether or not it has all of the wanted features/abilities in the final product is not so concrete. As development goes on the goals and design of this project are more likely to change, and they must as in real life, things change, requirements change, options change. However if the current development plan is followed and stays on schedule, a tangible timeframe appears. The time that is most likely to see a first release depends on two options. The first option on the table now is to just release an end-user interface for the first release instead of a full development framework, leaving the framework to be developed for a later release. If this option is chosen, a release could be seen in a few weeks. However this release would have many flaws as it is still heavily based on the FORCE project and does not have substantial unit testing completed. This gives us the second option, a core framework with limited capability, which would only be useful to an advanced developer, but would allow for more testing. The largest drawback to this option is time, as it would require much redesign and starting some code from scratch, and would take at least 1-3 months. In any case the project is feasible, and with dedication and a lot of man-hours, should see completion within the year.

Glossary

Torque – resource manager providing control over jobs spread across multiple nodes

Node – single system in a cluster of computers

Extreme Programming – agile programming philosophy of software development

Cluster – set of nodes comprised into a single network connected system

GPU – Graphical Processing Units

Thor – Western Michigan University's High Performance Computing Cluster

Unix/Linux – Operating Systems, mainly open source based.

Interface – method of communication between a user and a system

Framework – general purpose software that can be molded to suit custom needs

Np hard - Non-deterministic Polynomial-time hard, informally, at least as hard as the hardest problems in all Non-deterministic Problems"

REST- Representational state transfer

API- Application programming interface

References

SEE FOOTNOTES FOR EXTERNAL REFERENCES