Browser Based Distributed Computing
TJHSST Senior Research Project
Computer Systems Lab 2009-2010

Siggi Simonarson

June 9, 2010

Abstract

Distributed computing exists to spread computationally intensive problems over a wide array of machines in order to obtain results more quickly than possible on one machine. This approach requires a large number of less powerful machines in place of one more powerful, more expensive, supercomputer necessary with a traditional approach. The largest such array of computers that exists today is a decentralized network of machines that communicate with one another via HTTP through web browsers known as the Internet. This research project seeks to combine these two ideas by harnessing the power of the Internet through widely available HTML and Javascript in browsers with PHP on the side of the server to perform large problems with relative ease. The project hopes that the wide user base available due to use of web technologies will compensate for the speed decrease associated with using Javascript for calculations.

Keywords: Distributed System, Parallel Computing, Volunteer Computing, Browser Based

1 Introduction and Background

1.1 Problem

Modern attempts at Internet based volunteer networks all require the user to download and install some software, either a standalone program, or the Java Runtime Environment to contribute their processor cycles to a computing project. This limits the number of users that can contribute to the project because some users do not have permissions to install software on their computer, are weary of third party software, or are not technologically savvy enough to install or use the software. This project seeks to answer the question, can a framework be created that is more user friendly for both the volunteers, and the organizers of volunteer computing networks, using current web technology that require no installation or technical knowhow on the part of the volunteers? This approach would allow anyone to contribute their computing resources to the cause they wish to support by simply visiting a website without installing any software.

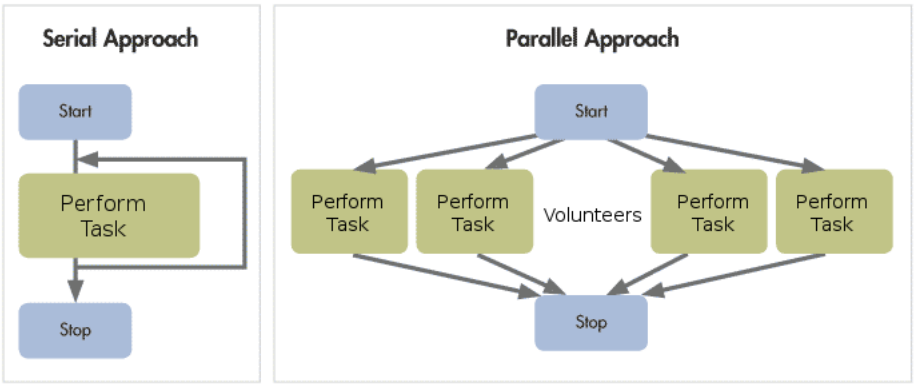

Figure 1: Visualization of serial vs parallel algorithms.

1.2 Introduction

This created framework that allows researchers to harness the tremendous number of nodes available over the Internet for volunteer computing. A web interface for project management will allow for a manager-worker task distribution model to be set up with relatively little knowledge of web technologies, and could run on any operating system. If the framework comes to fruition and is satisfactorily accessible, fault tolerant, secure, and efficient, a sample problem to demonstrate the use of the framework will be implemented. The results of these projects, given an extension to the framework to format the data in the required manner could be sent to one of the various projects already using volunteer networks (Folding@home, SETI@home, etc).

Distributed systems are often used in graphics processing. Modern day graphics cards rely on many processors that can calculate in parallel using programming languages like CUDA to render graphically intensive scenes. Raytracing is an excellent way to utilize a parallel architecture for rendering a scene because each pixel calculation is independent of one another. In this way, each volunteer can receive data points telling them which row to calculate, perform its calculation independent of the other workers, and send the calculated color values for that row back to the server. These results would then be stored, and could be viewed by either workers, the admin, or both depending on the settings enabled by the administrator of the project.

1.3 Previous research

Several research projects have been done in the field of volunteer computing. A project based out of MIT seeks to provide a similar framework to developers using Java instead of Javascript, which they hope to increase speed, but severely limits the number of nodes they can access. With the current increase in the speed of Javascript in modern browsers due to the advent of web based applications, Javascript will become comparably fast. A background in web development is essential to the development of this project. The project will be coded almost entirely in PHP and Javascript using HTML and CSS to display the interface and AJAX for message passing.

Other research has been done in the field to determine how volunteer computing projects can get the most processing cycles out of the resources being donated to them. A project out of the Worcester Polytechnic Institute determined that sending data sets out one at a time increased the efficiency

of resource usage more so than sending the data out in chunks and having the volunteer buffer the data. They also found that devising a more complicated scheme for determining how much data to send at once didn't significantly improve on this efficiency. Due to the findings of this project,a task management scheme will be used in this project that sends out one data set at a time.

Another research project that looked into the most effective method of fault tolerance in Java volunteer networks similar to the one that I'm building. The project determined that instead of the more typical methods of voting, which requires each piece of data to be calculated multiple times, it is more effective to spot check particular pieces of data with already known results, and blacklist users who return incorrect results from that piece. This prevents large numbers of volunteers from sabotaging a project, and reduces the amount of data that needs to be calculated twice. This method of fault tolerance will be implemented, if other areas are satisfactorily completed.

2 Procedure

2.1 Working Model

To begin, a cursory working model of the manager worker interactions between the server and the browser was established to assess the validity of the idea. The test interaction was a simple number addition application. The server would give each worker two numbers to calculate, the volunteer would add the two numbers, and send the results of the calculation back to the server where it would be stored and a new number was calculated. This established the basic message passing between server and worker that is necessary for the more general framework that is being developed.

2.2 Message Passing

Message passing between the browsers (workers) and the server (manager) done in AJAX. A Javascript method called XMLHttpRequest which is built into Javascript that essentially request a webpage in the background while a client is viewing a site. In this case, I would be visiting the message.php site. What makes this function useful, is that arguments can be placed in a url, giving the server some data to work with. For the worker to

send his results back to the server XMLHttpRequest would request message.php with the results as an argument, which would look something like this http://PROJECTURL/message.php?results=RESULTDATA. This server would read in this RESULTDATA, store it and generate new data points for the workers to calculate. These data points would then be displayed at the url XMLHttpRequest has requested, so that data would be usable in the Javascript, which would go to work calculating that data.

2.3 Data Storage

The next step in the development process was to develop a storage infrastructure. Because MySQL isn't as accessible as folder storage, and this framework is aimed at accessibility, a folder storage scheme was implemented. New folders are created for each user to store the problem that they currently calculating, and the number of data pieces calculated. A general storage folder keeps track of the data, results, current user id, and the current piece of data being calculated. PHP functions in the include file allow for easy access of the variables stored in these files.

2.4 Computation

The compute method in the Javascript is a general method that can be used to implement many parallel applications. To test the framework and make generalization easier for more advanced applications. A ray tracing example was implemented in Javascript. When the calculation page is visited, the volunteer runs the initialization function, which sends a message containing its IP address to the server. The server stores this IP address and creates a unique id and folder for this user in the data structure. In the folder is placed a problem, and a new increment value set to 1. The volunteers browser receives this first problem through the AJAX update function and stores it in a buffer. The data that is received is a row number, and the number of pixels in that row. The volunteer then creates an array for storing the calculated pixel values. Using the modified ray tracing engine, the pixel values for this row are calculated and stored in this array. These array values are then serialized, with individual color values separated by commas, and pixel values separated by semicolons. This result is then sent to a message function on the server, which updates the results, the users current problem,

and both the user and global incremental variables. The user is then sent another data set to calculate and the process is repeated.

2.5 Display

If the user wishes to view the results of the project they are working on, they can visit the display page which uses the new HTML canvas object to display the results. Separating the results from the calculations in this way allows users to focus their computing cycles on calculating the problem, and then view the results when all calculations have been completed. The results stored in the data infrastructure are queried from the server and iterated through. The results are first broken up by semicolon, then by commas and stored in arrays for easier access. Color objects included in the ray tracing engine are then created using these values and stored in another array. The engine is then told to render the colors in the array to pixels on the HTML5 canvas. This could be improved upon in the future by having the server do this rendering, and saving the rendered image as a JPEG. This JPEG could then be displayed to each volunteer, significantly reducing the bandwidth required to pass the displayed results back and forth.

3 Results

3.1 Current State

At the current state, one implementation of the framework is complete. Most of the functions are hard coded in and not as general as they will eventually be. When the more general framework is created, the current implementation should be able to be implemented with relative ease. The current state executes the rendering of a ray tracing scene over an array of computers linked through the Internet and web browsers using AJAX as the method of passing messages between the browsers and the users. The interface for the calculation, aside from aesthetic changes is complete for the most part. The start, stop and speed slider will remain, no matter what data is being calculated. The display page however will be easily changed by the framework, while now it is hard coded in.

In its current state, the administration section is not as robust as it will eventually be. The only two options are to view results and purge data. The

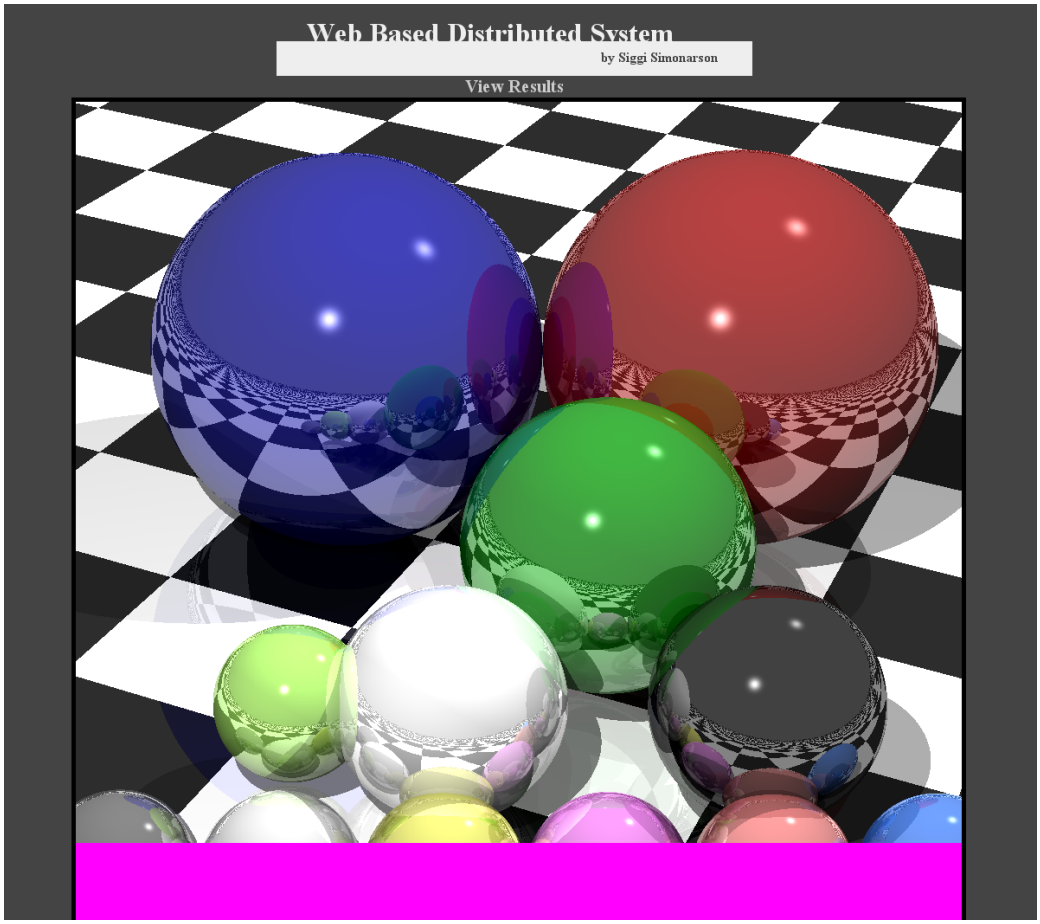

Figure 2: Screenshot of the display interface.

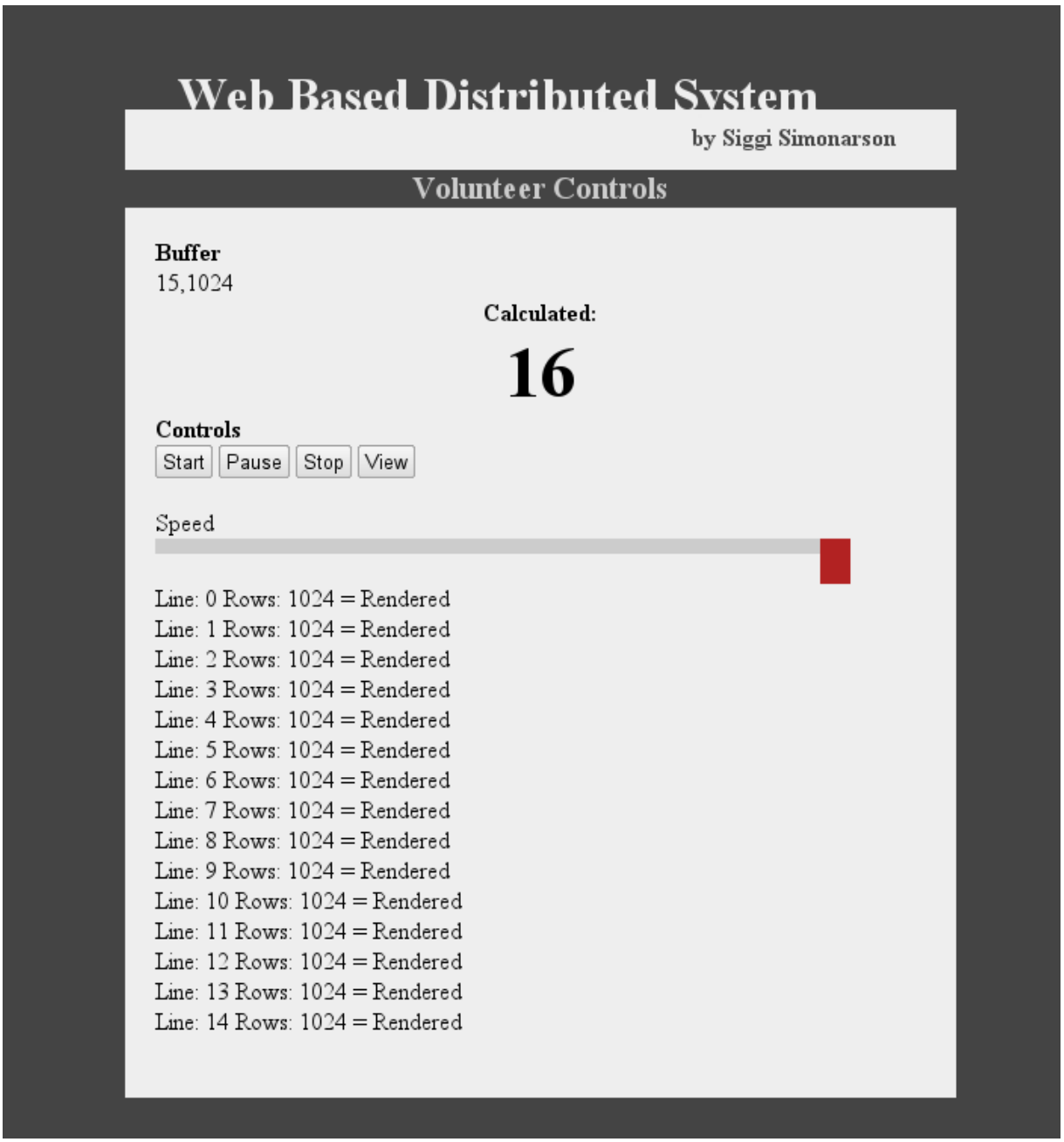

Figure 3: Screenshot of current state of the calculation interface.

view results function takes the data file in which all the results have been calculated, and displays it in a web friendly way that the administrator of the project can easily view. The purge data function came out of necessity. Having to always go back and reset all user values, id values, and results data became rather tedious, so the process was automated. Now to reset the data, the purge page is visited, and a completed project can be completed again.

3.2 Ideal State

Ideally at the end of this research project, the simple addition calculation will be replaced with the ability for the administrator to define a calculation to perform. The data set will also be different, and suited specifically to the project being run using the framework. The contributor interface will be much more user friendly, with the ability to set the speed (limiting processing cycles), stop the calculations, and view more comprehensive statistics of their calculations. The administration section will also be far more robust than the current state. As previously stated, the administrator will be able not only view the results, but add new data, and change the calculations performed on the data. The administrator will also be able to see more comprehensive statistics as to the state of their project.

3.3 Trends

The speed can be seen to decrease as the number of volunteers increase. The increase seems to follows a pattern of exponential decay, which is to be expected because the work is being done by one node in the first case, is split in half in the second case, into thirds in the third case, etc. Extrapolating this data, it appears to approach an asymptote. This is the limit of useful additions of volunteers to the project. This asymptote however could be lowered by adding more servers, improving on the data storage methods, and having faster, more modern browsers.

3.4 Javascript Speeds

Using Ubuntu's computer language benchmarks performed on an Intel Q6500, the speed decrease associated with Javascript can be quantitatively analyzed. The min, median, and max slow downs can be seen in Figure 7. At its cur-

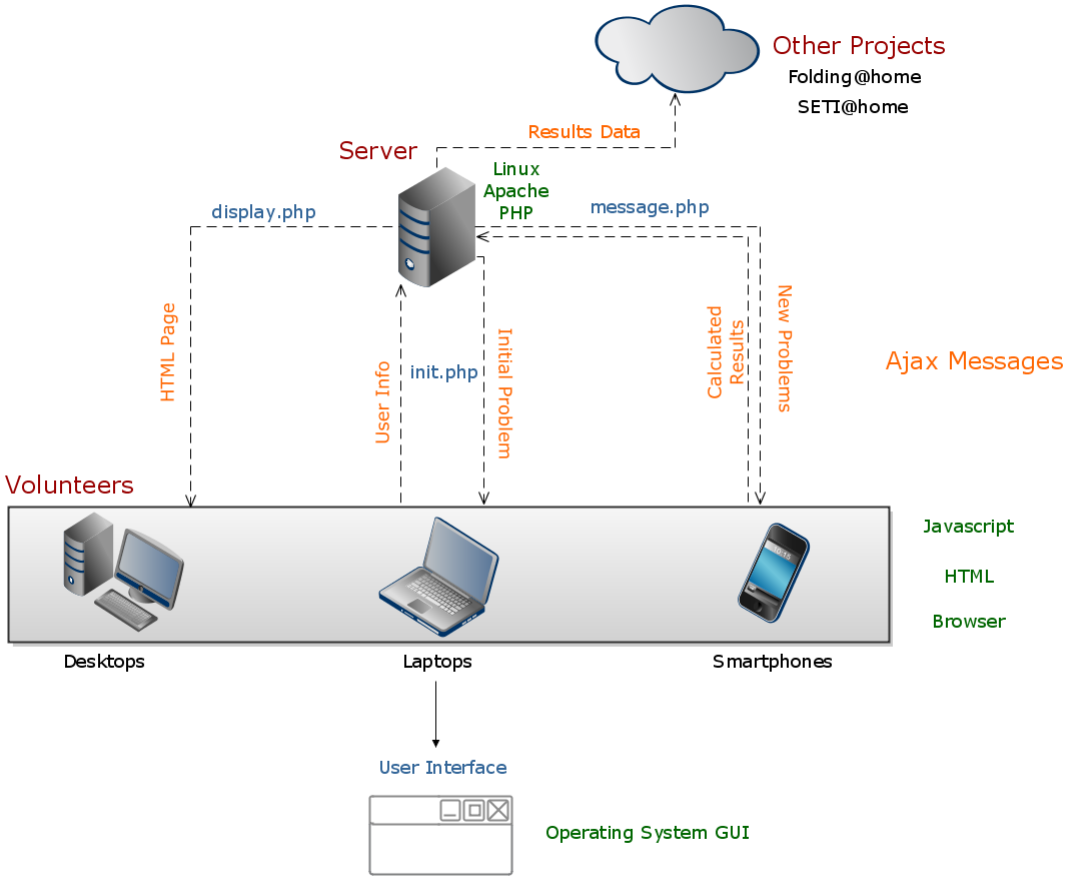

Figure 4: Overview of project.

Size:	Nodes:	Time (s)	Error %	Date
1024*1024	1	210	0	03/12/10
1024*1024	1	202.5	0	03/17/10
1024*1024	1	211	0	03/17/10
1024*1024	2	114	0.10%	03/24/10
1024*1024	2	114.5	0	03/24/10
1024*1024	2	114.6	0.20%	03/24/10
1024*1024	3	86.5	0	03/24/10
1024*1024	3	85.6	0.10%	03/24/10
1024*1024	3	85.7	0	03/24/10
1024*1024	4	63.1	0	03/26/10
1024*1024	4	63.7	0.10%	03/26/10
1024*1024	4	75.7	0	03/26/10
1024*1024	5	54	0	03/26/10
1024*1024	5	59	0	03/26/10
1024*1024	5	53.2	0	03/26/10

Figure 5: Results of time testing with varying numbers of volunteers.

rent speed, Javascript is essentially 7 times slower than native languages like C and C++. This means that you would need 7 times more volunteers in a web based system, than a client based system like BOINC (assuming no asymptote). While a web based system would be more accessible, achieving 7 times as many volunteers would be difficult. For this reason, a speed improvement in Javascript would be necessary in order for this framework to be usable.

4 Discussion

4.1 Conclusion

Given that a framework for parallelization using web technologies was created, it was proved that such a system is feasible. The more important aspect of such a system is whether or not it is practical given the relatively slow speed of scripts that are executed in the browser, such as Javascript. This slow speed was hoped to be made up for by the increased user base gained by using such technologies. Because the decrease in time is approaching an asymptote, a large number of volunteers past a certain point will essentially be useless. For this reason, the viability of the framework is in doubt

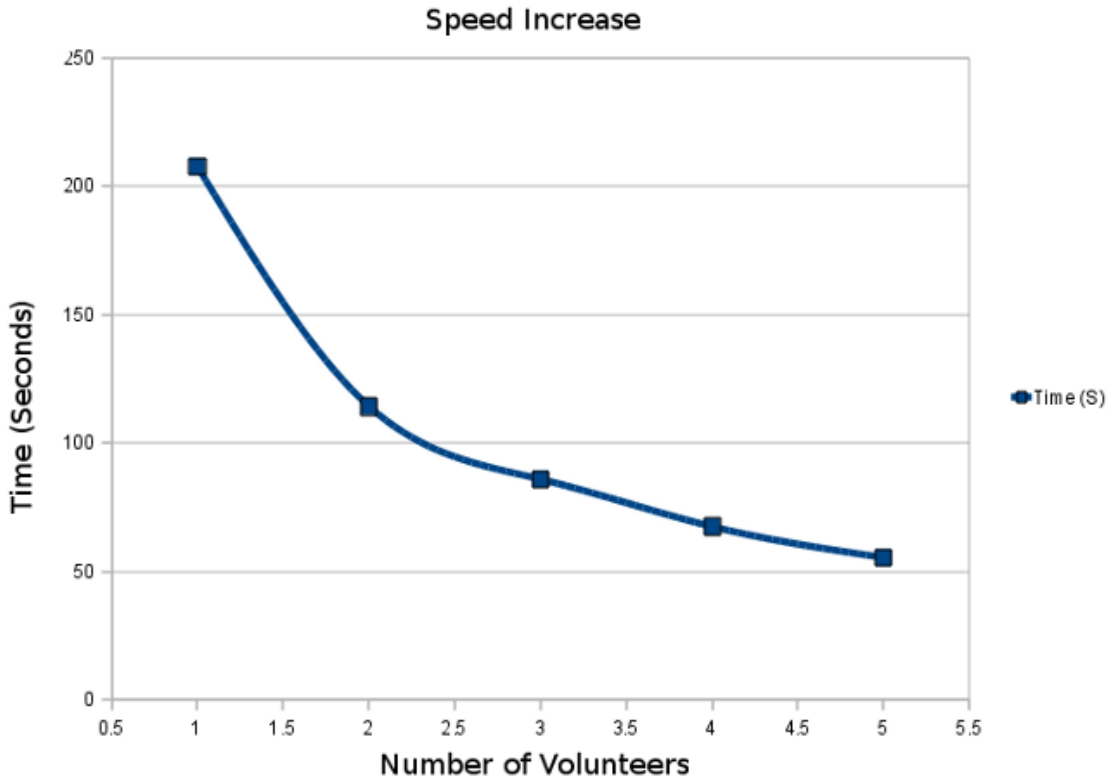

Figure 6: Results of time testing with varying numbers of volunteers.

Javascript Slowdown			
Vs.	**Max**	**Min**	**Median**
C	1/91	1	1/ 7
C++	1/91	3	1/ 8
Java	½	1	1
Python	1/41	24	7

Figure 7: Results of comparative language benchmarks.

with current technologies. As the speed of Javascript engines increase with modern browsers like Firefox and Chrome this will improve.

4.2 Improvements

The framework is satisfactorily complete in terms of working, but there is room for improvement in a wide variety of areas. The framework can still be made more generalized, so that it can be adapted to fit the needs of any parallel project. Fault-tolerance and security are also only implemented in the simplest of forms, and adding such functionality would be essential in a production version of such a system. The system is scalable to some degree, however the the speeds to approach an asymptote, which can be fixed by varying the work package size, number of servers, etc. The interface at the moment is rather rudimentary, as the admin interface is rather sparse, and the front end interface could be made more user friendly.

References

[1] L. F. G. Sarmenta, "Sabotage-tolerance mechanisms for volunteer computing systems", *Future Generation Computer Systems*, 2002.

[2] L. F. G. Saramenta and S. Hirano, "Bayanihan: Building and Studying Web-Based Volunteer Computing Systems Using Java", *Elsevier Preprint*, 1998.

[3] D. Toth and D. Finkel, "Increasing the Amount of Work Completed by Volunteer Computing Projects with Task Distribution Policies", *Elsevier Preprint*, 2008.