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

Siggi Simonarson

January 25, 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 array of such computers that exists today is a decentralized network of machines that communicate with one another over 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. In addition to the framework, an easy to use web interface that would allow future researchers to set up volunteer networks with little knowledge of web technologies will be implemented for accessibility.

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, and others 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 would allow anyone to contribute their computer resources to the cause they wish to support by simply visiting a website without installing any software. Additionally this project seeks to answer whether an easy to use admin interface can be made so that administrators of volunteer computing projects can easily set up, monitor progress, and view results.

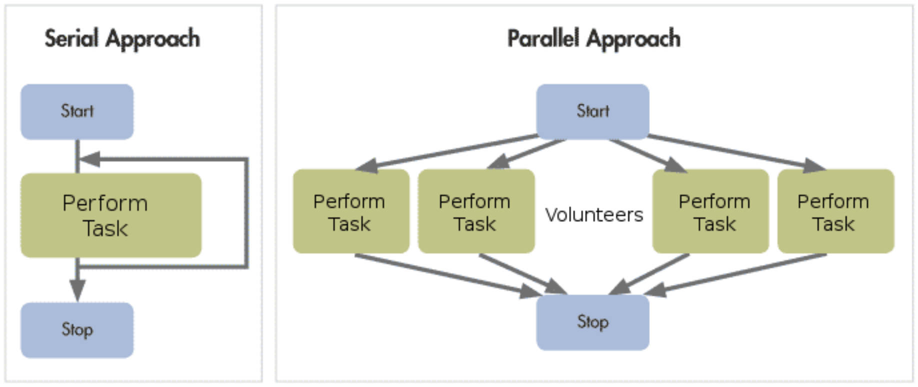


Figure 1: Visualization of serial vs parallel algorithms.

1.2 Introduction

This project aims to create a 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. If possible, that data will then be sent to one of the various volunteer computing projects in existence today.

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. Message passing done in AJAX sent the first two numbers to the volunteer in the initialization function. The worker then added the numbers (just a placeholder function for the cursory model) and sent the results back to the server.

2.2 Data Storage

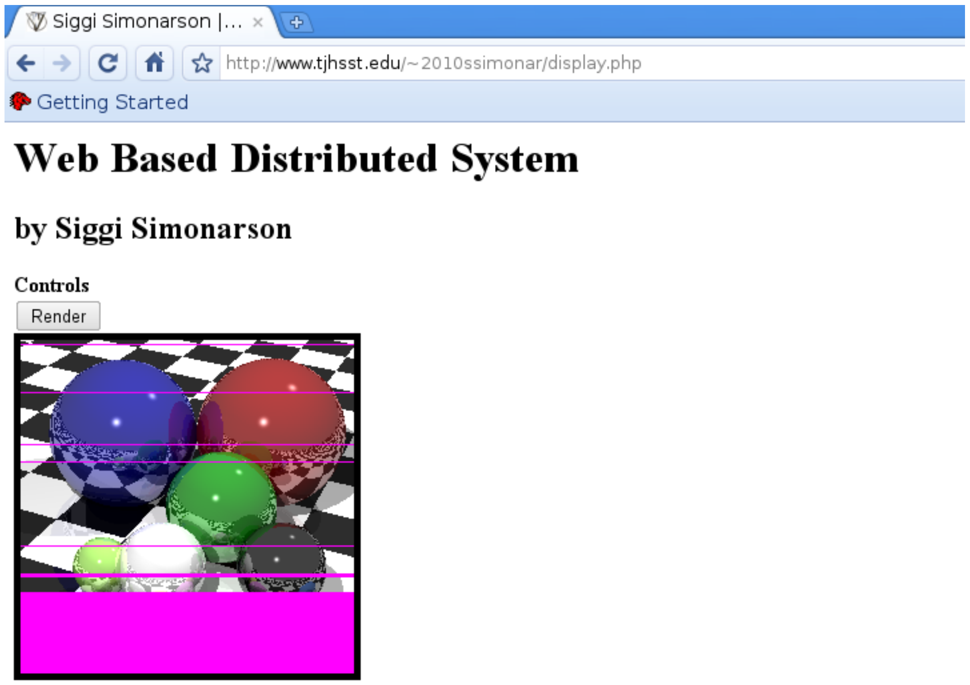
The next step in the development process was to develop a storage infrastructure. Because MySQL isn't as accessible as folder storage, and this framework is aimed at accessibility, a folder storage scheme was implemented. New folders are created for each user to store the problem that they currently 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.

2.3 Implementation

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.

2.4 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. 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 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.



Web Based Distributed System

by Siggi Simonarson

Controls

Figure 2: Screenshot of the display interface.

3 Results

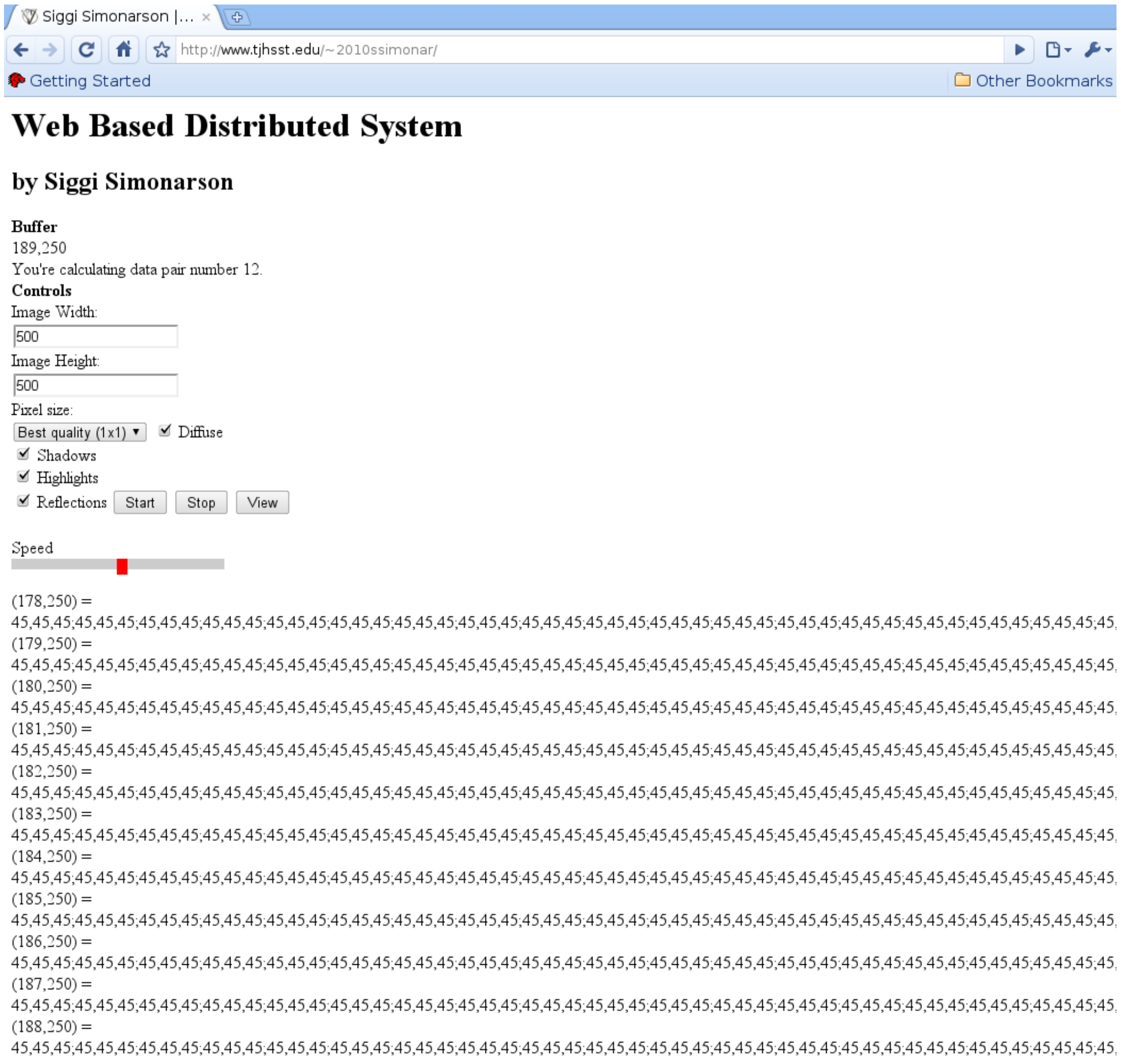
3.1 Current State

At the current state, one implementation of the framework is essentially 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 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 to the largest degree possible. User folders still have to be removed manually.

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.



Web Based Distributed System

by Siggi Simonarson

Buffer
189,250
You're calculating data pair number 12.
Controls
Image Width:
500
Image Height:
500
Pixel size:
Best quality (1x1) ✓ Diffuse
✓ Shadows
✓ Highlights
✓ Reflections

```
(178,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
(179,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
(180,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
(181,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
(182,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
(183,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
(184,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
(185,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
(186,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
(187,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
(188,250) =
45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,45,45;45,
```

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

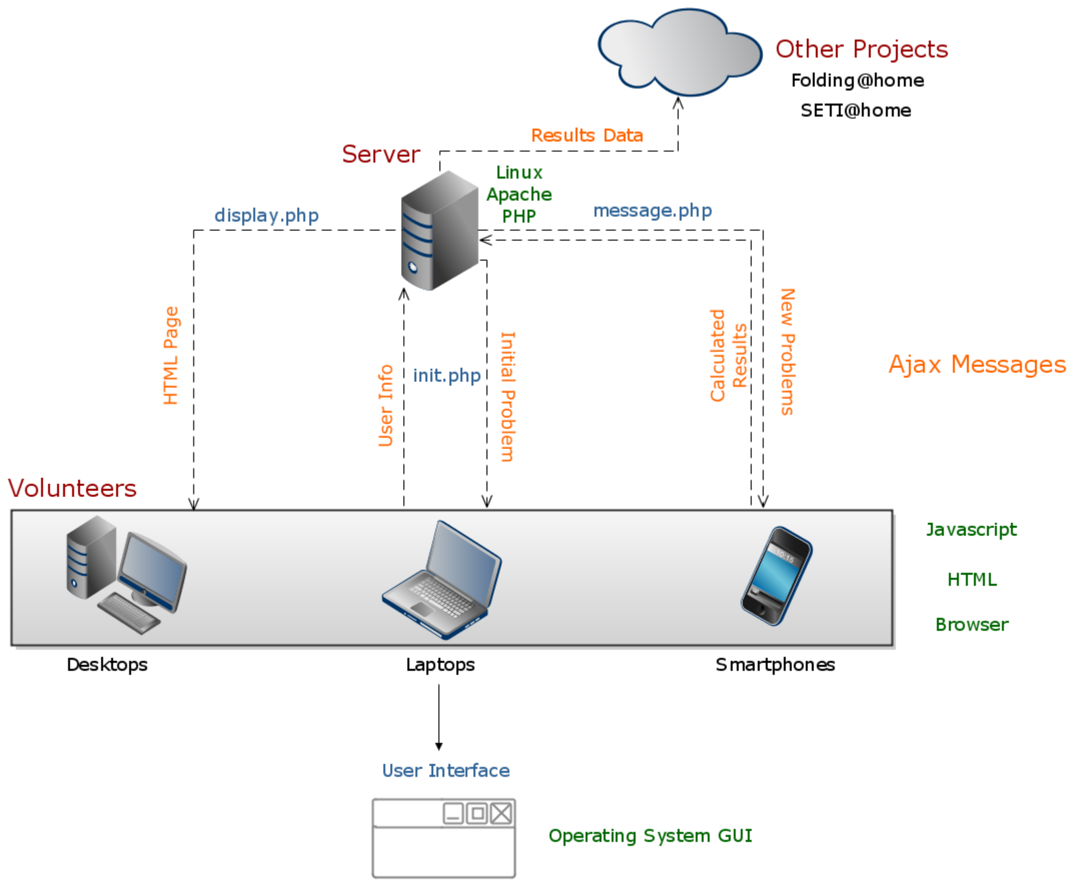


Figure 4: Overview of project.

4 Discussion

The results will be based on the completeness of the framework, judging based on its adaptability, fault-tolerance, security, scalability, and interface. The successfulness of the project, and the answer to the question being researched relies heavily on the speed of Javascript. If a significant speed improvement can be seen with an increase in volunteers, the project will be successful. Even with a slight speed improvement, this will be magnified as Javascript engines in more modern browsers like Google Chrome become more efficient and powerful.

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] L. F. G. Saramenta and S. Hirano, "Sabotage-tolerance mechanisms for volunteer computing systems Volunteer Computing Systems Using Java", *Alteneo de Manila University*, 2002.

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