Browser Based Distributed System

Siggi Simonarson
TJHSST Computer Systems Lab

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 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.

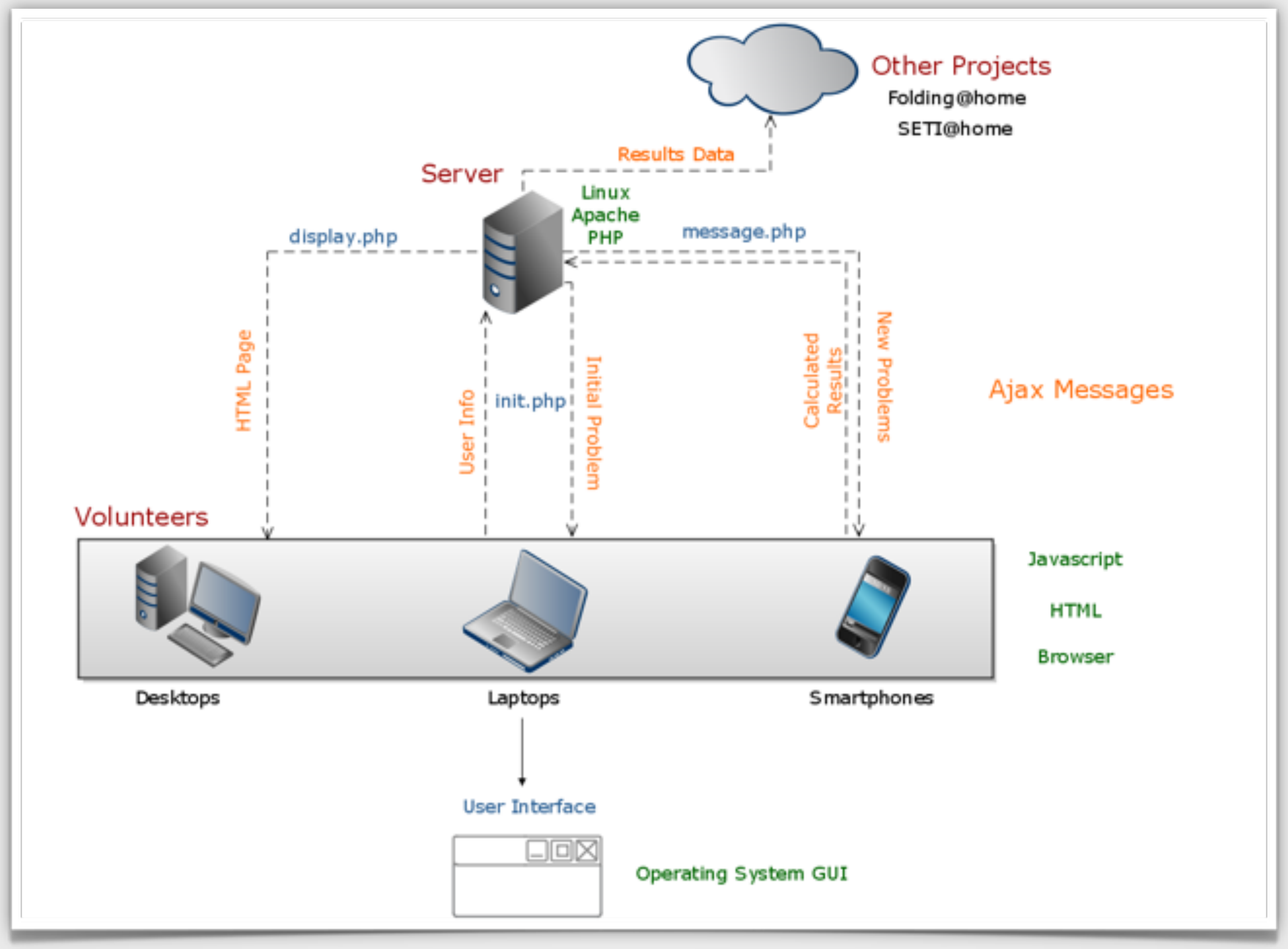
Background

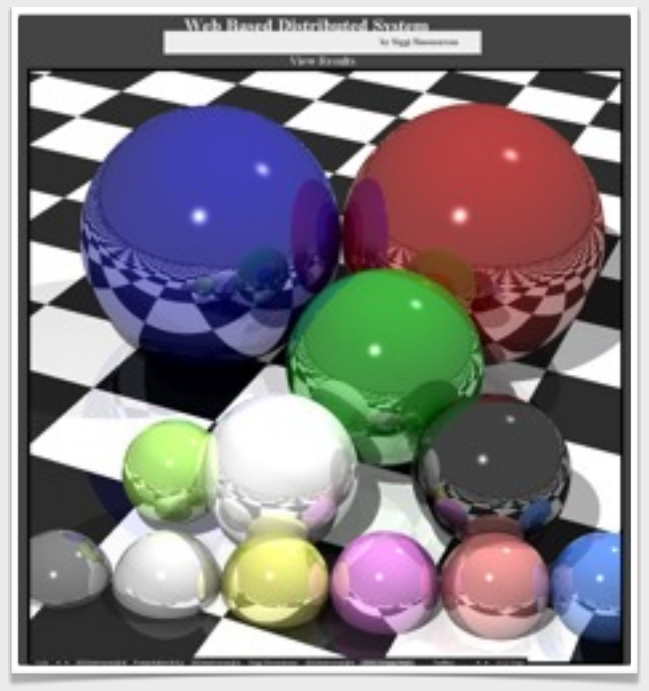
Problem
Modern attempts at Internet based volunteer networks all require the user to download and install some software, either a standalone program, or the Java Runtime Environment to contribute their processor cycles to a computing project. This limits the number of users that can contribute to the 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.

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. 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.

Procedure



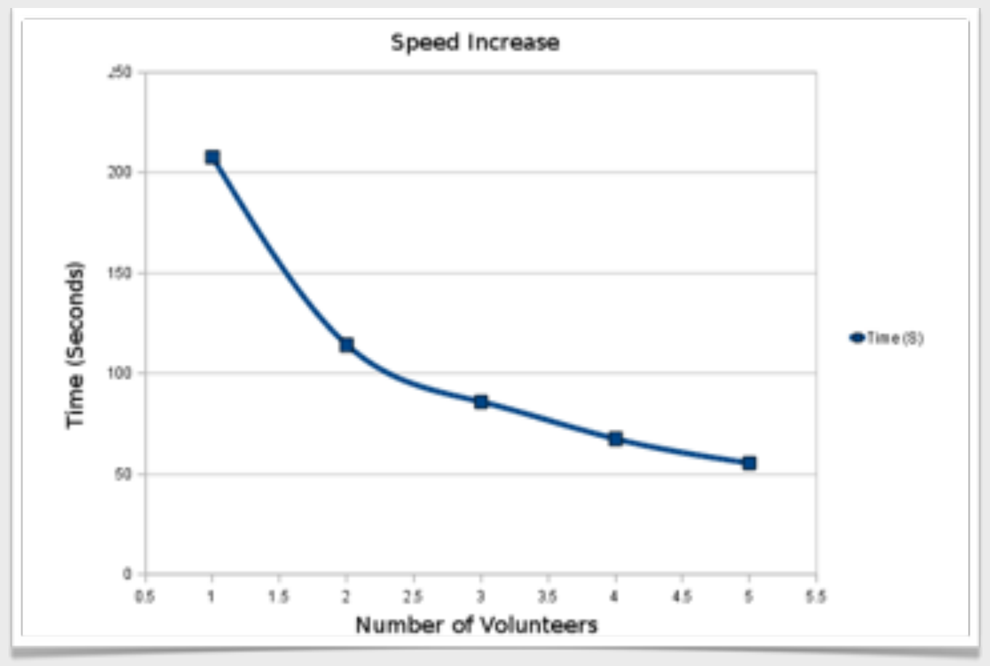


Data Storage
The next step in the development process was to develop a storage infrastructure. Because MySQL isn't as accessible as folder storage, and this framework is aimed at accessibility, a folder storage scheme was implemented. New folders are created for each user to store the problem that they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

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 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.

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.

Results



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. The speed does however seem to be approaching an asymptote at which the speed cannot be increased any more.

Conclusions

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

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”, *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”, *Worcester Polytechnic Institute*, 2008.