Objective

Create a volunteer computing application for stock analysis

Abstract

Web browsers are some of the most frequently used computer applications today. However, a large portion of their data cycles is wasted during idle time. The purpose of this project is to create a browser page that can take advantage of these wasted cycles, using them to analyze stock price trends. Multiple machines will analyze individual stocks and send their results to a central server, which will then determine the most profitable investment strategies from the compiled data.

Following the naming convention of popular volunteer computing projects such as SETI@home, this program will be called Stocks@home. Stocks@home has the potential to have processing power of a supercomputer for a fraction of the cost due to the prevalence of consumer computers and web browser users. The only limit to its computational capabilities is the number of contributing users.

Introduction

- The large demand for processing power is often fulfilled by expensive supercomputers
- Volunteer computing: A central server sends many clients small tasks, receives responses from those clients, and compiles those responses into a useful result
- In this case, each client is a device whose processing power has been donated by a volunteer
- Benefits of volunteer computing:
  - Little upkeep cost, volunteers pay for power costs
  - Little startup cost, volunteers pay for devices
  - Self-sustaining, volunteers maintain their own devices
  - Self-adapting, volunteers naturally upgrade their own devices or upgrade the hardware/software in their devices

Background Info

- Quantitative analysis: Prediction of stock price trends solely using historical pricing data
- MACD analysis: method of quantitative analysis of stock price trends
- Node.js is an open-source, cross-platform runtime environment for server-side applications.
- Asynchronous I/O model: makes use of non-blocking calls to keep programs running efficiently.
- Async functions can be started and set aside until they are complete
- In the meantime, the program moves onto other methods.
- The program continuously moves forward when using async methods

Design

Tasks

Each task is stored in the server, ready to be sent out for completion

Server

The server sends each task to a client, which returns a response back to the server. The server compiles the responses and sends the final result back to the clients.

Sample Code

Analysis

The data were collected on a single device running multiple instances of the Stocks@home web client hosted locally to emulate multiple devices collaborating over the internet.

The time taken for the Stocks@home server and clients to complete stock analysis gradually decreased as more clients were added. However, as more clients were added the test device struggled to provide the resources needed to drive each client. A performance plateau was experienced at four clients. The system’s CPU ran at 100% during all three four-client trials.

Future Research

- Further optimizing code
- Implementing security measures to prevent malicious/fraudulent responses
- Testing over multiple devices and device types
- Testing with greater volumes of data
- Using real-time price data as opposed to archived data

Conclusion

Stocks@home takes less time to complete the stock analysis as more clients are added and is a successful implementation of the volunteer computing design. With further testing and design, it could potentially analyze thousands of stocks everyday with little-to-no upkeep costs.

References
