Bitcoin Price Analyzer with Apache Spark

Saul Crumpton
Randy Deng
Josh Henson
Alberto Li
Lingfeng Zhou

Georgia Institute of Technology
School of Electrical and Computer Engineering

Introduction to Cryptocurrencies

Introduction

The market for Bitcoin in the past few months has gained a lot of attention due to many factors, including its decentralized nature as a currency, and more importantly because of its continuous rise in price. As Bitcoin is starting to become widely adopted by businesses around the world, thousands are jumping on board to both evaluate and analyze its trends in the market. Even more are joining the rush, hoping to simply make a "quick buck". There is currently a huge opportunity for people in any sector to speculate movement regarding the Bitcoin market, and make money while doing so.

The Problem

The price of Bitcoin follows unprecedented patterns in the market. Over the past few months it seems like it has been increasing without bound. As a newly popular commodity, Bitcoin prices are hard to predict as they change every minute. Its characteristics such as a volatile nature and high gains/dips in price or dips are compounded by quick movement and surges of popularity. The Bitcoin Price Analyzer application was created in order to better characterize and analyze the Bitcoin market. Users will be able to more accurately query and analyze historical data in order to make predictions with greater precision regarding the future.

Application Features

Architecture Setup

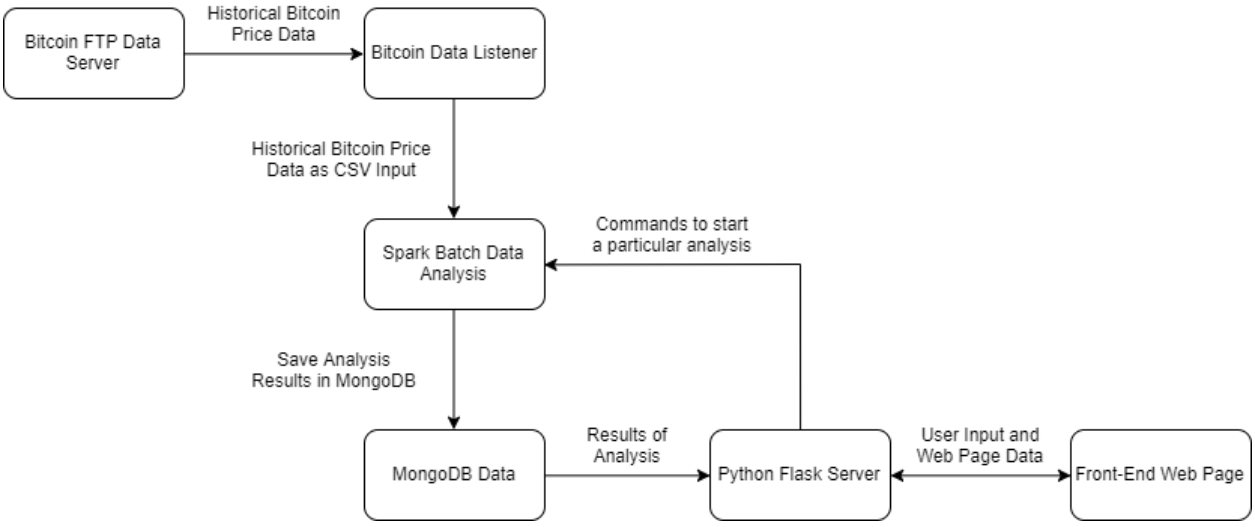

Figure 1. Block diagram of Bitcoin Analyzer setup and process flow of system.

Our architecture uses a Bitcoin data listener to gather both historical and current price data. A user will interact with the front-end application and send inputs to begin data analysis. Parameters are sent to Spark for batch data analysis, which stores the data in a MongoDB database. This data is eventually given to the user in the form of a graph. This architecture is both flexible and allows for proper propagation of data through different interfaces.

Application

Home

Figure 2. Startup home screen of Bitcoin Finance Analyzer web application.

The bitcoin finance analyzer welcomes the user to a home screen with a small introduction of the application and some of the features used to create it. It features a dashboard view and set of analysis tools for the user to have better insights on the market price of bitcoin. The price of bitcoin is viewed in real time on the dashboard view, with live updates to the price every 30 seconds. Also, different types of analysis of of bitcoin prices can be selected on the analysis tools page, where mathematical functions are invoked to carry out parallelized computation of big historical data sets to generate views of the bitcoin prices.

Dashboard and Analysis Pages

Dashboard

The dashboard includes two main graphs. The first being a real-time graph of the price of Bitcoin and the second showing the price of Bitcoin for the past 30 days.

The real-time graph is a six point live graph of the price of Bitcoin. It uses the CoinDesk API to query the current price of Bitcoin from the bitcoin exchange and plots it on a graph. The graph moves dynamically as the price of Bitcoin updates every 30 seconds with a resolution of six points at any given point. Figured below is a screenshot of the real time bitcoin price graph.

Figure 3. Dynamic, real time moving graph of live bitcoin prices featured on the dashboard.

Additionally, a second graph is included on the dashboard that shows the Bitcoin price for the past 30 days. The 30 day Bitcoin market price graph is static compared to the real-time graph as it only updates every day at midnight. Below is a screenshot of the 30 day Bitcoin market graph.

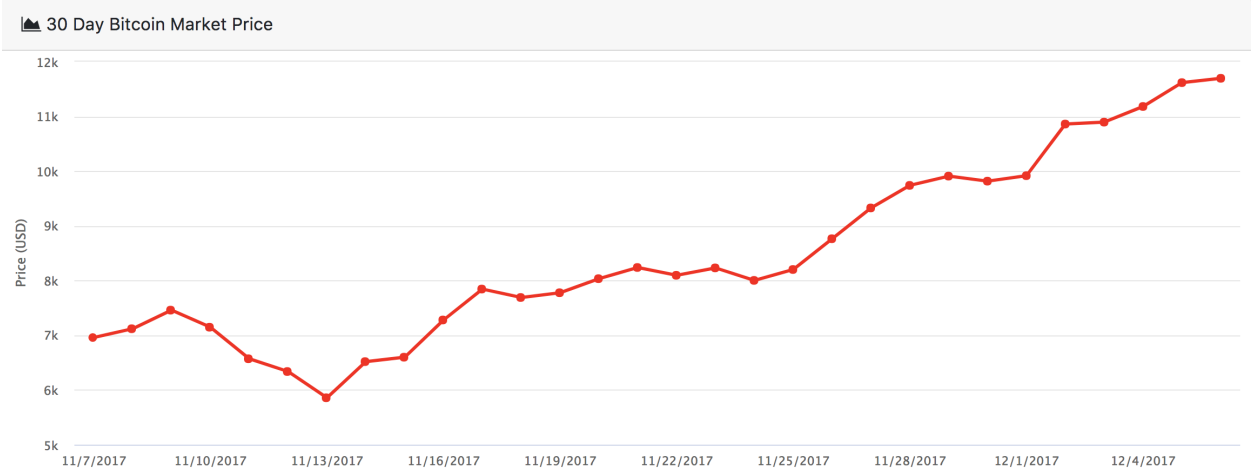

Figure 4. Static, real time graph of 30 day historical bitcoin prices featured on the dashboard.

Analysis / Indicator 1: (SMA)

The first analysis technique used is a simple moving average or SMA. The SMA is a widely used indicator in technical analysis to help investors forecast future price movements in securities using past data. To calculate the SMA corresponding to a given data point or closing point of a security, a number of past closing points are summed together and then averaged with the current closing point. The number of the past points to use is determined by the period or window size of the SMA, which is a configurable parameter for technical analysts to tweak. Not only the is the SMA an important indicator for observing securities, it also serves as the basis for more advanced indicators used in technical analysis such as the moving average convergence divergence (MACD) which uses SMAs of various window sizes to comparing market short term and long term trends to arrive at informed conclusions of buying or selling.

To implement the SMA in code, a simple for-loop could be employed to iterate across the data. This straightforward method, however, is time-inefficient and often would result in unresponsive cloud applications when servicing a large user base. An alternative, time-efficient, and scalable solution involves convolving the input data using a moving average filter through the numpy. This solution is the implemented solution for the Bitcoin Analyzer and alleviates the use of loops. Figured below is a screencapture of the working SMA algorithm integrated into the Bitcoin Analyzer.

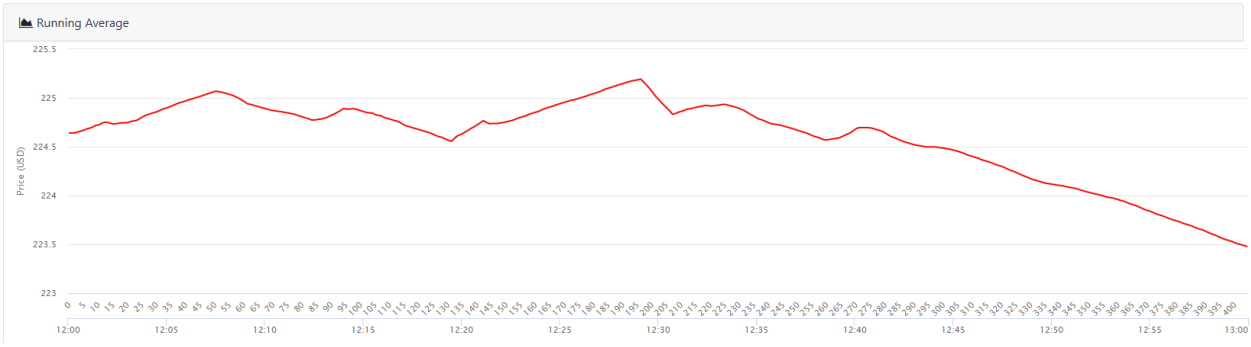

Figure 5. Example of running average graph generated from hourly historical data for a 24 hour period, featured on the analysis and tools page.

Both time and datapoint index values are displayed in the x-axis. The datapoint indices help determine the preset window size for the SMA graph where the window size is determined by the equation [sqrt(N) / 2 + 1] where N is the total index size of the displayed dataset. Analysts can use this equation to observe how the predetermined window about a datapoint changes about index plus and minus window size to get a better grasp of how the price of Bitcoin or BTC is evolving.

Analysis / Indicator 2: (EMA)

An more advanced variant of the SMA is the exponential moving average or EMA. The EMA is similar to the SMA and is also a popular metric for MACD indicators. Unlike a SMA, an EMA is weighted and more recent closing points are given exponentially more weight than past points. This make EMAs much more responsive to change than SMAs. Similar to SMAs, EMAs also have an adjustable period or window size for analysts to tweak and feed into MACD algorithms.

The EMA algorithm implemented for the Bitcoin Analyzer builds upon the SMA solution, using a more complex convolution kernel to account for the exponentially changing weights. The use of convolution through numpy again circumvents the need for inefficient iteration. Figured below is a screencapture of the working EMA algorithm integrated into the Bitcoin Analyzer, applied to the dataset used for the SMA screencapture.

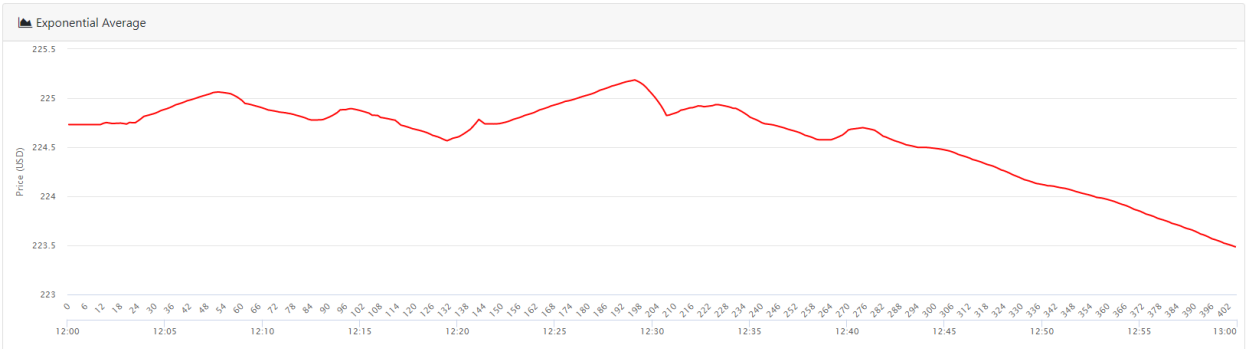

Figure 6. Example of exponential average graph generated from hourly historical data for a 24 hour period, featured on the analysis and tools page.

Due to the relatively small window sizes employed for the SMA and EMA relative to the total number of datapoints inputted into the algorithms, the EMA and SMA curves have similar overall shapes. At local and global maximas and minimas, the shapes of the MA graphs are drastically different. Additionally, at certain slope points, the EMA graph also has a different concavity than that of the SMA which would affect advanced indicators such as MACDs differently.

Analysis / Indicator 3: (MTM)

The final technical analysis indicator available in the Bitcoin Analyzer is the price momentum of bitcoin or MTM. Much like momentum in physics, the price momentum of a security is a commonly used indicator to simulate and forecast the future position or price of a security. If the price momentum of a security has peaked, it is usually a safe and reliable indicator for a buy or sell depending on if the peak was positive or negative peak. Similar to SMA and EMA, each data point or closing point has a corresponding MTM value, and is calculated by adding the price acceleration to the previous MTM value.

Due to the integration required to calculate MTM values, an iterative solution was adopted for the MTM implementation in the Bitcoin Analyzer. For very large datasets, multithreaded integration techniques are often employed to compensate for the computational cost. Figured below is a screencapture of the working MTM algorithm integrated into the Bitcoin Analyzer, applied to the dataset used for the SMA and EMA screencaptures.

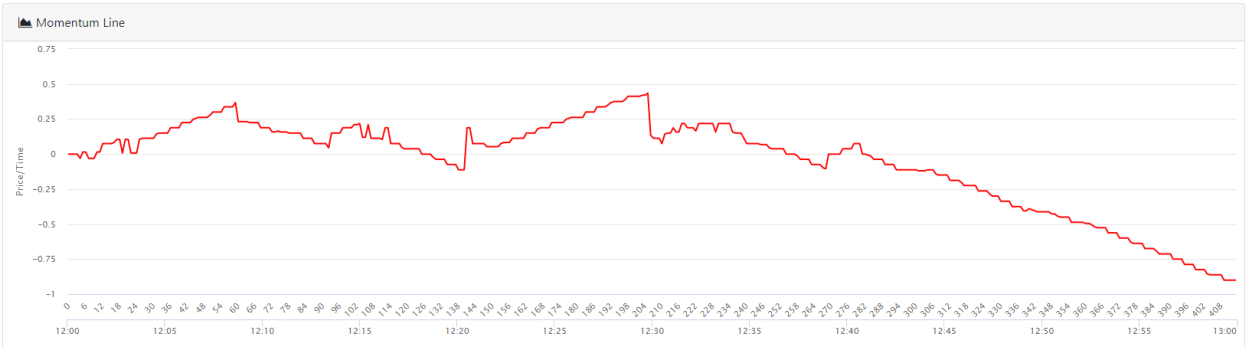

Figure 7. Example of momentum line graph generated from hourly historical data for a 24 hour period, featured on the analysis and tools page.

At the beginning of the dataset, there is a strong and positive momentum which peaks out when the Bitcoin price begins to drop. This drop, however, doesn't cancel out the initial upward momentum until much later in the dataset which means as long as the price momentum is positive, there is an effective netgain. To ensure reliable and high long term returns, the momentum line is commonly studied with MACDs generated from MA indicators to ensure a securities portfolio containing Bitcoin has a net positive momentum.

Python Flask and Spark

Although our team is dealing with a relatively small cloud data set (~700mb unzipped), the sheer amount of data is still too large for conventional manual filtering and transformation. Spark was utilized in order to quickly filter the data and then parallelize some of the workload for separating the data into distinct, time oriented sets. Multiple avenues for getting the data have been developed, allowing for a wide amount of data to be utilized on the front end- all the way up to 5 million data points per analysis. These included outputting back to a plaintext file for large data sets (similar to how the data was received), and pushing to a local MongoDB. The database could be adapted to be on another cloud instance in order to improve the data availability and accessibility. This would be extremely necessary in the case of multiple spark instances being utilized in a cluster; this would be likely in the event that larger data was being done.

Deployment with uWSGI and Nginx

To handle scalability, the web application is deployed using Nginx. Nginx is a web server that can be used for balancing loads from traffic, caching, and reverse proxying to alleviate multiple requests to a web server. The basic functionality of Nginx will be exploited to increase the reliability of the application. More specifically, the load balancing and proxying requests to multiple web servers functions will be used in this application. Since, the computation for analysis of big sets of historical stock market data, employing these measures reduces the stress on a single entity web server.

Below are screenshots of uWSGI and Nginx servers running on an EC2 instance. The configuration files can be found in the root directory of the project and instructions on how to set up the instance can be found in the setup folder.

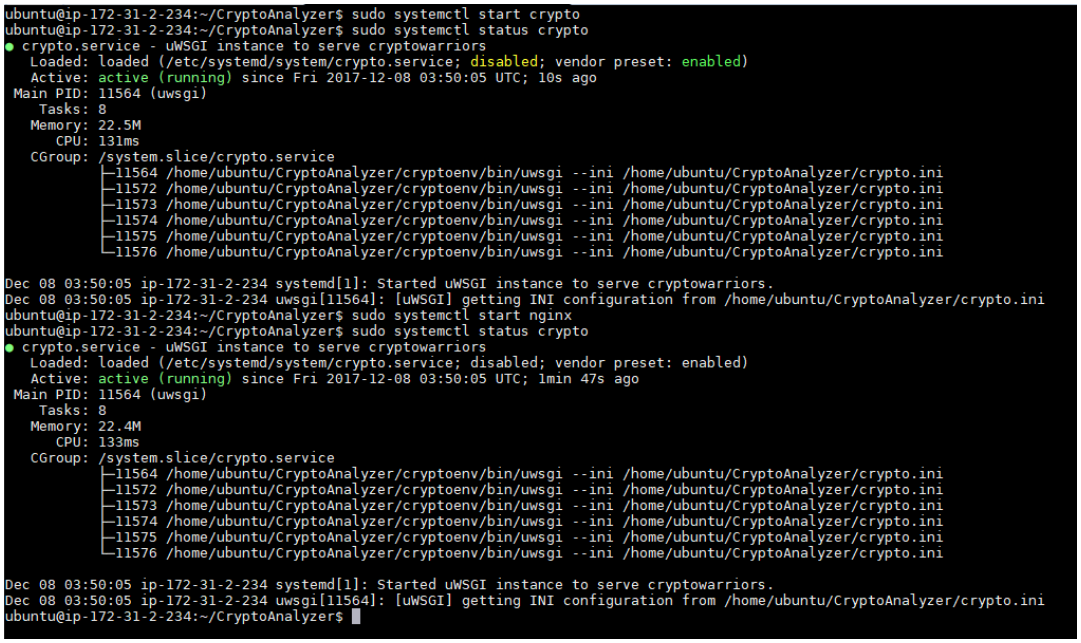

Figure 8. Shows uWSGI and Nginx servers running on the EC2 instance hosting the Bitcoin Price Analyzer application.

Team Member Contributions

Saul Crumpton

- Writing Spark Script for batch data analysis
- Writing shell script for quick setup of EC2 instance
- Filtered data and structured it for pushing to MongoDB

Randy Deng

- Developed Python Flask handling of user GET and POST requests
- Spawn threads to handle Bitcoin data listener
- Conversion of dates to Linux Epoch Time
- Retrieving data from MongoDB and error-handling
- Setup of Nginx and uWSGI

Josh Henson

- Consume data on front-end and display data as graph
- Bootstrap and JQuery setup on front-end
- Home page of web application

Alberto Li

- Functions to handle Bitcoin data listener on Dashboard page
- Create live moving graph of real time bitcoin market price from CoinDesk API
- Dashboard page displaying Bitcoin price graph of last 30 days and current price
- Query live Bitcoin data on front-end using JavaScript

Lingfeng Zhou

- Provisioning technical analysis algorithms
- Researching technical theory behind analysis algorithms
- Analysis algorithm implementation and development
- Added timestamp functionality to analysis frontend
- Debugged dashboard and analysis tool frontend