Constructing a Scalable System for Identifying Arbitrage in Cryptocurrency and Foreign Exchange Markets

Shawn Anderson
shawn_anderson@sfu.ca

Vincent Chiu
vlc4@sfu.ca

Ka Hang Jacky Lok
khlok@sfu.ca

Abstract

Arbitrage is an opportunity for profitability which arises from a discrepancy in the price of an asset across markets. The act of exploiting arbitrage results in the settlement of the markets, and an equilibrium price of the asset. Arbitrage in the domain of Foreign Currency Exchange is well explored and documented [citation], thus opportunity is limited to experienced financial players with access to high power computing and extremely low latency. We explore a new frontier in arbitrage identification, foreign exchange markets in conjunction with cryptocurrency markets. Cryptocurrency markets can be thought of as proxy foreign exchange markets as users are able to convert one fiat currency to another with one or more intermediary cryptocurrency transactions. In comparing foreign exchange rates from traditional markets to cryptocurrency proxy markets, we identify that significant arbitrage opportunity exists. We propose a scalable system that consumes and stores market data, making it possible to observe both current and historical arbitrage opportunities. In addition, our system identifies optimal currency trade paths to yield maximal arbitrage profitability.

1 Introduction

In the emergence of disruptive technologies there is an abundance of uncertainty and volatility. Cryptocurrencies are disrupting traditional financial paradigms at an alarming rate, resulting in ripe opportunities for profit and discovery. In such an uncertain time, data can reveal meaningful underlying patterns to be exploited. Arbitrage is such a pattern that can be discovered by acquiring and analyzing the appropriate data. The purpose of this project is to identify the existence of arbitrage by comparing exchange rates from foreign exchange markets to exchange rates in cryptocurrency markets, using an intermediary cryptocurrency.

In addition to identifying real time arbitrage opportunity, we are interested in capturing market data to be saved for historical analysis. Historical analysis is critical as it provides a window to deeper insights. For example, we hypothesize that high volatility in the price of Bitcoin could be a factor in currency price discrepancies across markets. The importance of historical data necessitates a Big Data approach to this problem. Data is saved in a scalable manner. System architecture is constructed with modularity using Big Data tools.

2 Methodology

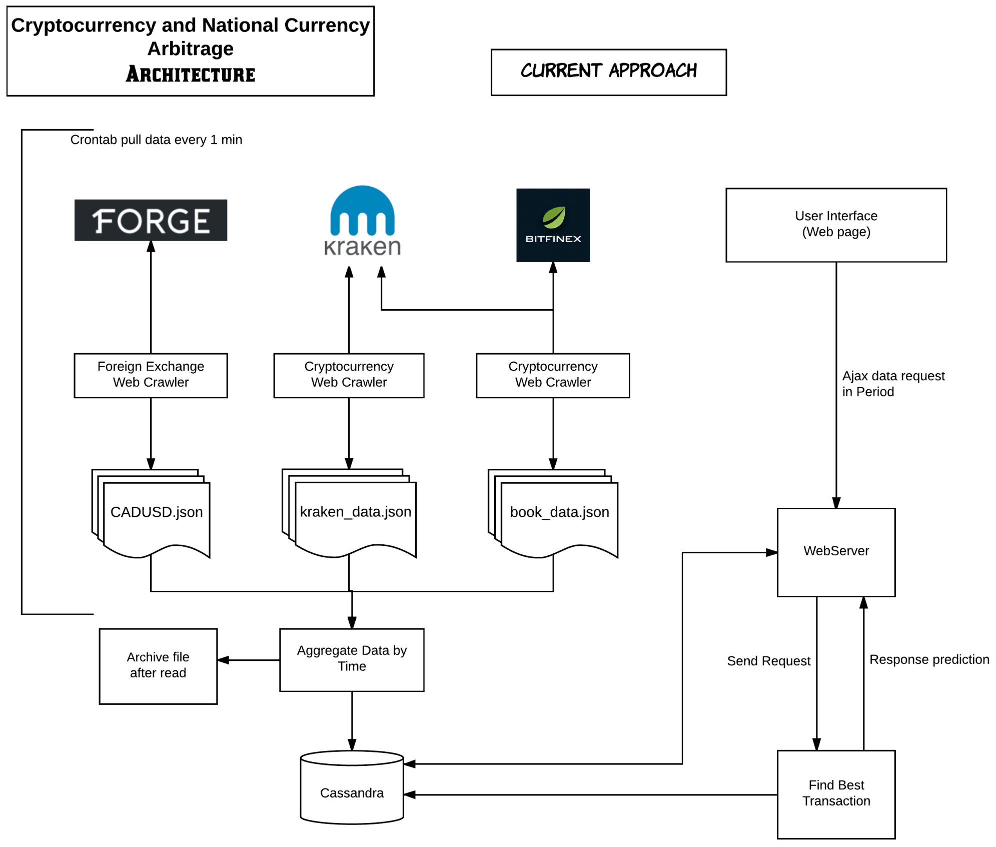

Cryptocurrency and National Currency Arbitrage
ARCHITECTURE
CURRENT APPROACH
Crontab pull data every 1 min
FORGE
kraken
BITFINEX
User Interface (Web page)
Foreign Exchange Web Crawler
Cryptocurrency Web Crawler
Cryptocurrency Web Crawler
Ajax data request in Period
CADUSD.json
kraken_data.json
book_data.json
WebServer
Archive file after read
Aggregate Data by Time
Send Request
Response prediction
Cassandra
Find Best Transaction

Figure 1: High level system architecture

Market data is extracted from one foreign exchange API and two cryptocurrency exchange API's. Data is aggregated and loaded into a Cassandra database in batch processes using Apache Spark. We performed analysis, machine learning, and arbitrage identification using Pyspark. We built a web application using Django, which connects to the Casandra database, allowing us to construct object relational querysets. We expose our data in JSON format via a rest API. We then visualize the data in real time using Java Script with ajax requests and Baidu Echarts [6] for visualization.

A virtual environment was setup using SFU computing resources in order to have a working environment where components of the system can be integrated. Access to a virtual environment allows all developers to work collaboratively in a remote space. Root privileges allow agile customization of software stacks and instant installation of desired software packages. We found that such an environment is imperative to the modern development paradigm of bootstrapping existing systems to iterate quickly, never re-inventing the wheel. In addition to the dedicated virtual environment, we leveraged the instructional computing cluster available for SFU Big Data students to run Apache Spark and Cassandra.

2.1 Extracting Market Data

In order to extract market data from multiple places, three stand-alone programs are constructed to collect data simultaneously. Each program downloads data from a web API, converts raw data into JSON format, then saves it into a pre-defined location. Cryptocurrency data is extracted with the help of a python API wrapper for crypto-exchanges called CCS [3]. Cypto-exchange data is collected from Kraken[4] and Bitfinix[2]. Foreign exchange data is pulled from Forex[1]

We wrote a script which made use of a library called CCS to pull data from the Kraken and Bitfinix API. We also wrote a script to pull foreign exchange rate data from Forex. A Cron job was also used to automate data collection using these scripts.

The cryptocurrency data captures order book information between all trade pairs composed of USD, CAD, Bitcoin (BTC), Etherium (ETH), and various other currencies and cryptocurrencies. The foreign exchange data captures transactional information between US Dollars (USD) and Canadian Dollars (CAD).

2.2 Data Storage

After we extract the data we join the data and save it into Cassandra. Cassandra is a scalable database that will scale for our future plans of accumulating very large datasets. Joining the data beforehand would avoid the query and huge join that we have implemented in run time.

2.3 Analysis

We have constructed processing and analysis programs with Pyspark. One program converts raw market data into arbitrage information. One program applies machine learning in order to predict price changes and future arbitrage opportunities.

We did linear regression and got an r^2 of 0.75 with timestamp as the feature and the asking price of bitcoin in CAD as the target variable. When outliers were removed we got an r^2 of around 0.54. We speculate that this is because the price of bitcoin was monotonically increasing during the time that collected our data. According to efficient market hypothesis, at any given moment in time, the current price is already reflective of all known information about a particular investment asset. Therefore historical prices do not affect future prices and it should not make sense to use time as a feature to predict price. However, due to the monotically increasing nature of Bitcoin's price, we believe that our linear regression model which uses the asking price of bitcoin as the target variable and the timestamp as the feature is justified. We used Spark ML for this. We also did Spark SQL operations on our tables to calculate arbitrage profit by multiplying various exchange rates together.

2.4 Web Application

In order to visualize and interact with our results we chose to use use the Django web framework, which is implemented in Python. Django connects to a database backend in order to expose an object-relational api for data queries. We use django-cassandra-engine[8] as a driver to connect Django to our Cassandra database. From Django we expose rest API nodes for ticker information, order book information, and trade sequence information.

Trade sequence information is computed in real time from a spark job which aggregates data from the database. We are able to achieve this by leaving a spark context session open, running on the cluster. Jobs are queued and executed remotely using spark-celery[7] with a RabbitMQ backend[10].

Visualization is implemented in JavaScript using the Baidu Echarts API[6] for charts. Data is updated in real time via AJAX calls to REST framework URLS. Our team designed the layout for the dashboard using HTML with Bootstrap for CSS. Some styling was adapted from an example by Pablo Garcia [9].

We have charts that show USD to CAD exchange rate vs time, USD to CAD through Bitcoin exchange rate vs time, and a chart which showed these 2 rates superimposed on each other. We have a fourth chart that displays the difference in exchange rates, an indicator of arbitrage magnitude. We also have a real time feed of optimal trade sequences. Trade sequences with profit ratio above 1.0 are profitable, others can be ignored.

3 Results

Significant arbitrage opportunity is observed. However, this model does not take transaction fees into account. Kraken has 0 fees provided that you spend over 10,000,000 on their exchange in the last 30 days. [5]

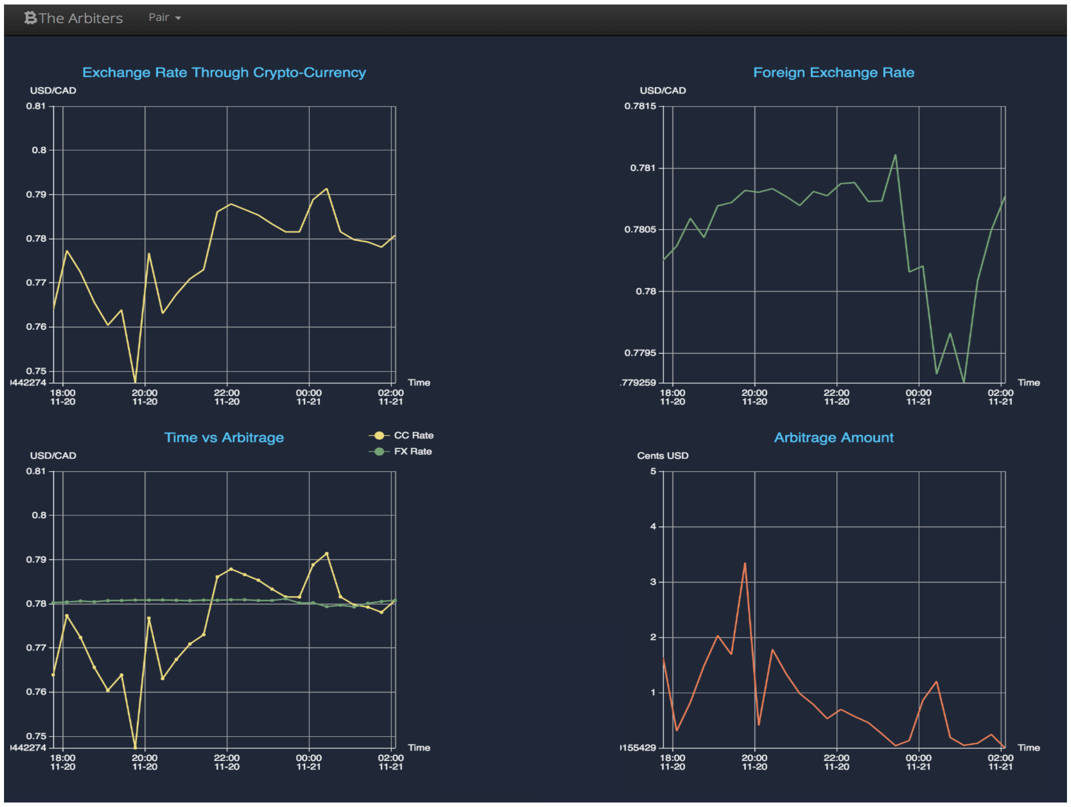
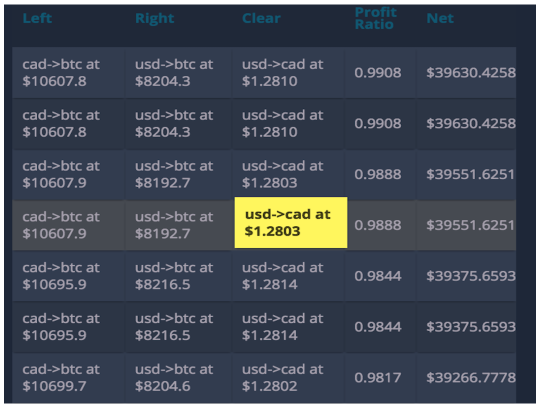


Figure 2: A visualization of exchange rates and architecture



Left	Right	Clear	Profit Ratio	Net
cad->btc at $10607.8	usd->btc at $8204.3	usd->cad at $1.2810	0.9908	$39630.4258
cad->btc at $10607.8	usd->btc at $8204.3	usd->cad at $1.2810	0.9908	$39630.4258
cad->btc at $10607.9	usd->btc at $8192.7	usd->cad at $1.2803	0.9888	$39551.6251
cad->btc at $10607.9	usd->btc at $8192.7	usd->cad at $1.2803	0.9888	$39551.6251
cad->btc at $10695.9	usd->btc at $8216.5	usd->cad at $1.2814	0.9844	$39375.6593
cad->btc at $10695.9	usd->btc at $8216.5	usd->cad at $1.2814	0.9844	$39375.6593
cad->btc at $10699.7	usd->btc at $8204.6	usd->cad at $1.2802	0.9817	$39266.7778

Figure 3: Real time transaction sequences for arbitrage

3.1 Challenges Faced

We found that the bottleneck for progress was generating a shared vision of what we where creating. It was difficult to specify in language what we intended to construct, resulting in a lack of a shared vision. Project specifications became implicit, intuitive, and incongruent. We overcame this challenge by having one team mate take charge, assigning well defined tasks to the other members.

More time was invested in the front-end than should have been. It quickly became overly complicated when we chose to use unnecessary technologies like React and WebPack. A lot of time was spent configuring and trying to work with these tools. Eventually we scrapped them and went with a much simpler front end stack. After choosing to just use Django, with JavaScript and AJAX calls, we progressed quickly.

At first we would like to try RabbitMQ instead of Kafka since we have never used it the class (but was mentioned). But soon after we have installed RabbitMQ server and tested. We found that we need to implement Socket Stream which may consume most our time to develop. Then we switched out focus to Kafka. When implementing Kafka, everything went well until we found that structured spark stream doest not support stream - "Any kind of joins between two streaming Datasets is not yet supported." As the result, we found that we were way behind schedule and we have to give up the DStream join and switch to a more stable and easy way: Schedule job and move file.

3.2 Future Work

Completion of the system architecture involves replacing batch processing with streaming. Data consumption will be increased to more cryptocurrency pairs, deeper order books, and data from more exchanges. With such additional data, we believe that we will be able to optimize arbitrage identification in order to find maximal profitable opportunities. In addition, we would like to implement an automated trade bot using exchange private keys. The bot would automatically seize profitable trade sequences, thus generating revenue.

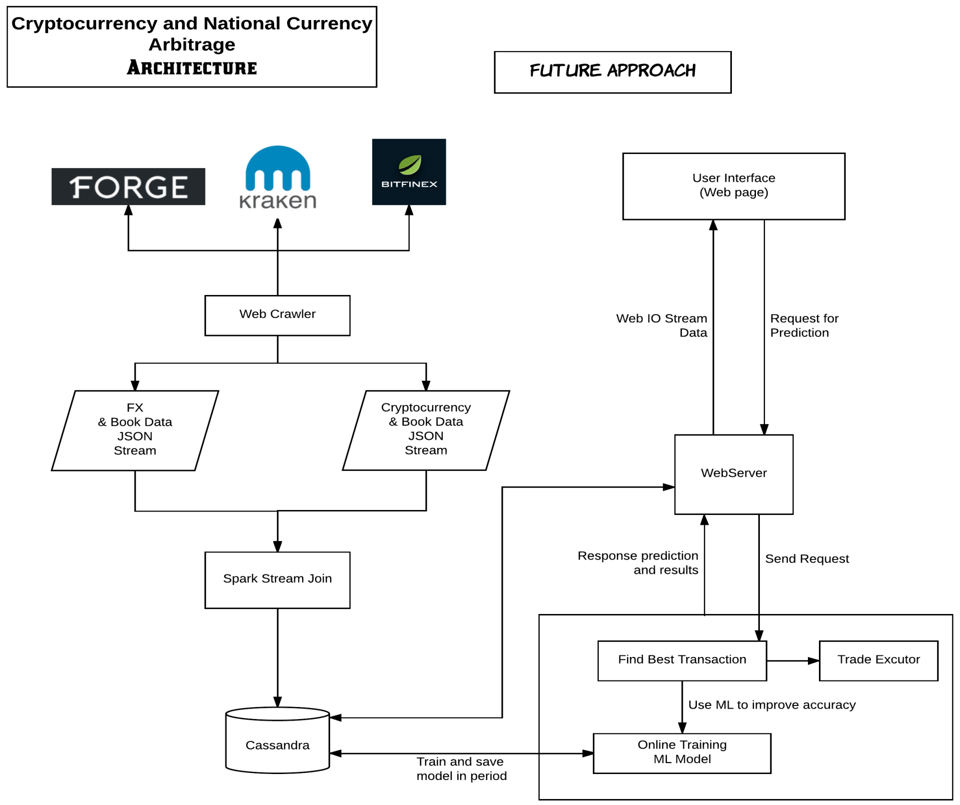

Cryptocurrency and National Currency Arbitrage
ARCHITECTURE
FUTURE APPROACH
FORGE
kraken
BITFINEX
Web Crawler
FX & Book Data JSON Stream
Cryptocurrency & Book Data JSON Stream
Spark Stream Join
Cassandra
User Interface (Web page)
Web IO Stream Data
Request for Prediction
WebServer
Response prediction and results
Send Request
Find Best Transaction
Trade Excutor
Use ML to improve accuracy
Online Training ML Model
Train and save model in period

Figure 4: Future system architecture

4 Conclusion

We have constructed a highly scalable system which is capable of identifying arbitrage of cryptocurrency and foreign exchange markets. We have created a user friendly UI that allows the user to identify the best trade sequence for making money at a given point in time. We conclude that given 0 exchange fees, it it is possible to make money through arbitrage. our system could be scaled up to ingest and process a large amount of data. We envision expanding our project to encompass every cryptocurrency exchange, cryptocurrency pair and foreign market exchange.

Beyond discovering the existence of arbitrage, we are interested in investigating it's behavior over time. We maintain the hypothesis that high arbitrage opportunities, as we have seen are correlated with high volatility in cryptocurrency prices, as we were fortunate to observe during the time the data was collected.

We have experimented with many new technologies during this project, we have learned a lot and had a lot of fun. Thank you for taking the time to read this report.

5 How to Run

```
git clone git@csil-git1.cs.surrey.sfu.ca:BigDataProject/brainstorm.git
cd brainstorm/webserver/
source ../django-react/venv/bin/activate
python manage.py runserver
```

References

[1] 1forge.

[2] Bitfinix.

[3] Crypto currency stocks.

[4] Kraken.

[5] Kraken fees.

[6] Baidu. Baidu echarts.

[7] Greg Baker. Spark celery.

[8] Rafa Furmaski. Django cassandra engine.

[9] Pablo Garca. css.

[10] Pivotal Software. Rabbitmq.