Team 1: Final Project Design

Mason Wilde, Jacob Hegna, Gage Benne, Andy Monroe, Grant Jurgensen

Project Name

Lawrence Trading Co.

Project Synopsis

Utilizing data science and Tensorflow machine learning, along with deep market data to automate cryptocurrency trading using principal funding via AWS cloud computing.

Project Description

The field of algorithmic trading is already strongly established within the realm of traditional stock exchanges. Typical strategies rely on extremely quick algorithms and low latency connections to an exchange in order to make short-term trades. Often, algorithms are implemented on a field-programmable gate array (FPGA) due to their speed, and the need to trade as quickly as your competitors to maintain a viable trading strategy. These factors combine to create a high barrier of entry into algorithmic trading on traditional exchanges.

In contrast, algorithmic trading of cryptocurrency is relatively undeveloped, and therefore presents a strong opportunity to enter while competition is thin. Cryptocurrency exchanges are hosted on the internet, which inherently creates a more even playing field than traditional exchanges. Where trades can be made in a matter of nanoseconds on traditional exchanges with direct connections, internet latency will be much higher for all parties. This also means we will need to make strong changes to established techniques in algorithmic trading, as we cannot assume low latency access to the exchange, and we cannot assume quicker access than our competitors.

The end result will be a program which trades cryptocurrencies through an internet exchange based on a machine learning algorithm, trained on market history data we have collected.

Project Milestones

First Semester:

1. Define deliverables: September 28
2. Initial Project Description: October 1
3. Build cryptocurrency trade monitor: October 5
4. Begin containerization with Docker: October 19
5. Project Proposal Report: October 22
6. Project Proposal Video: October 26
7. Begin developing Adversarial Neural Net Algorithm (ANNA): November 2

Second semester:

1. Final Project Design: February 11
2. Deadline for Implementing Pseudo-trading: March 17
3. Revisit & revise models and algorithms based on pseudo-trades: March 24
4. First Day of live trading: April 1
5. *Get rich: April 2*
6. Final Project Video: April 24
7. Quad Charts: May 8

Project Budget

This project will primarily utilize Google's open source machine learning framework: Tensorflow. Open source programming languages coupled alongside the machine learning framework such as Python and C++ mean very little implementation costs. That being said, there will be costs associated with hosting certain components of the project on Amazon Web Services (AWS). We will utilize Amazon Aurora (RDS), a PostgreSQL-compatible relational database, to handle the large amount of market data collected. As the main compute machine, an Amazon EC2 instance will handle pulling market data and populating the database, performing trades, as well as managing data delivered to the machine learning instances Amazon SageMaker (Sage). Making use of Sage will provide us the ability to build, train, and deploy machine learning models quickly in a easy to manage environment. All in all, we are confident our costs will be minimal, but will request reimbursement up to $1,000 for compute costs on AWS. (There is also a stage in which we put our own money in the algorithmic trading platform, we will not include this in the budget.)

Work Plan

Our roles are divided based on interest, capability, and what we each contributed last semester. Following that template, Andy will work primarily on infrastructure: making sure our AWS technologies run smoothly (RDS, EC2, and Sage) and ensuring our stack is fully integrated with Docker. This will ensure that the other developers don't have to worry about library/OS compatibility issues between EC2 and their personal development machines. Gage will be the database administrator, ensuring the databases follow consistent schemas and that the data pipeline between EC2, RDS, and Sage runs smoothly. Gage will also, if time permits, work on a data visualizer for current trading/profits and what data we have stored (price history). Mason is the primary data scientist managing the machine learning models and handling the actual correctness of our ML. Jacob will work on the code that actually executes the trades, in addition to writing the code that communicates with Sage to continuously update the model to respect the new trading information received during the day. Grant will be on flex duty: working initially on integrating more exchange APIs for our data pipeline and ML training, then transitioning to fixing bugs that occur during trade execution. Grant will also likely work with Gage to create a web interface for the data visualization that goes alongside our project (realtime P&L, realtime tracking of current assets owned, visualization of price history and projected future prices based on the model).

Gantt Chart

Task	Start	Finish	Assigned
Project proposal	October 17	October 22	all
Video project proposal preparation	October 24	October 26	all
Collection of raw cryptocurrency market trading data	September 20	February 28	Jacob / all
Cryptocurrency exchange research	October 29	November 9	all
Manual modelling	October 24	November 14	Jacob, Andy / all
Analyzing manual modeling	October 31	November 14	all
Researching market indicators	November 5	November 19	Grant, Mason / all
Tensorflow research	November 19	December 17	all
Tensorflow and general machine learning integration	November 19	February 28	all
Data formatting to support machine learning attempts	December 17	January 21	Grant, Jacob / all
Assisted algorithm development from collected data	January 21	February 18	all
Validation and correction of generated algorithms	January 21	February 18	all
Final Design Document	February 5	February 11	all
Cryptocurrency market integration	February 8	March 15	Gage, Andy / all
Production and validation of generated algorithms from real-time cryptocurrency market	March 11	March 22	Andy / all
Putting money on the line and verifying results	March 22	April 5	all
Project documentation	October 24	April 24	Gage, Mason / all
Public presentation preparation	April 17th	April 24	all
Quad Chart	May 1	May 8	all

Final Project Design

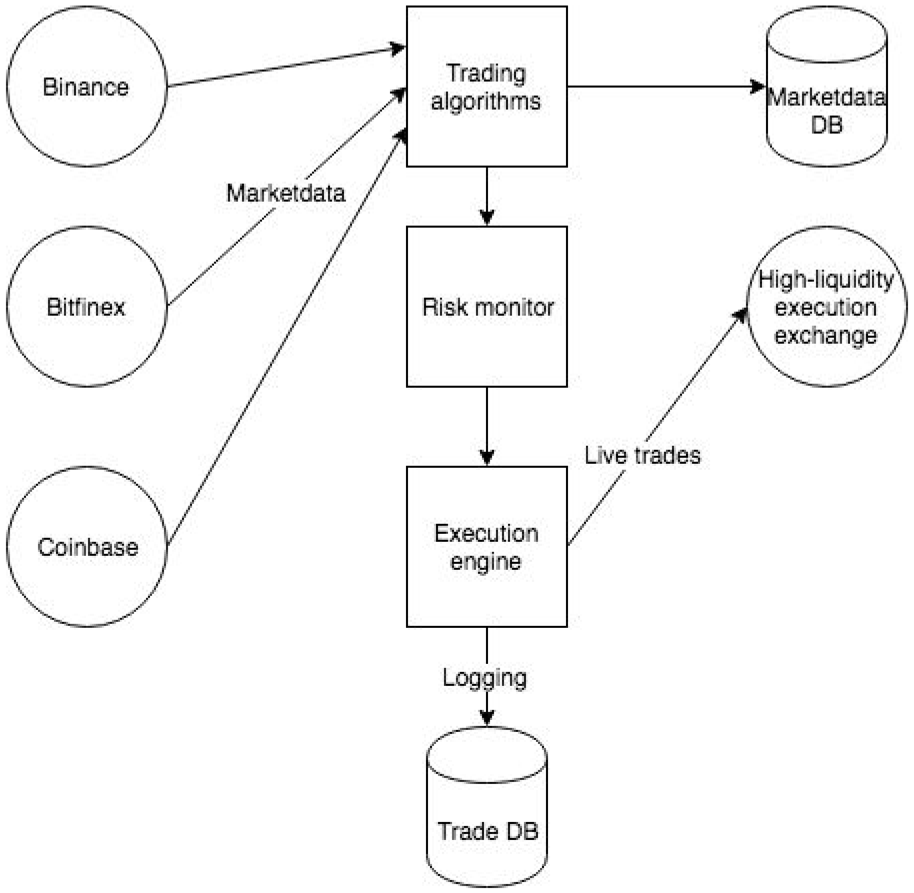
We have set some technical limitations on ourselves in order to keep the project focused and on topic. We have decided to focus strongly on C++ and Python for our languages. C++ was chosen due to its power, efficiency, and popularity, both within the group and industry-wide. This is the language we will use for our computationally significant and time-sensitive components. Python will be used in less computational sections of the project on account of its ease of use, as well as its incredible external libraries such as TensorFlow. Note that TensorFlow and many other Python libraries are written in lower level languages and merely expose a Python calling interface. Likewise, we will use PyBind to establish our interoperability between C++ and Python.

In particular, we will aim to have much of the trading infrastructure be implemented in C++, and do market/strategy research in Python. For example, we will implement:

- Market data feeds/API access
- Trade execution
- Risk (keeping track of the amount of money exposed to various cryptos, with the ability to override trades from the execution engine if they over-expose us. For example, if two different trading strategies each decide to be long on Ripple, the risk platform might stop the second trade because too much money would be dependent on that asset)

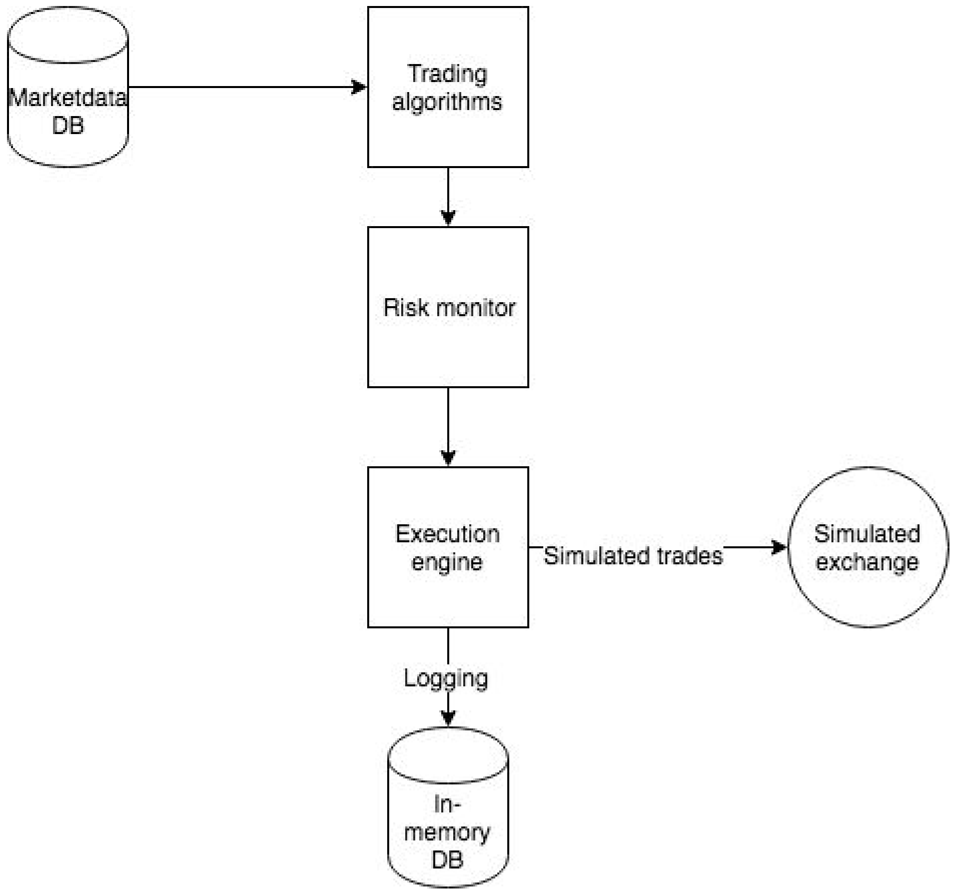
In C++. We will aim to implement things like a simulated trading environment, historical backtesting, and database interaction in Python, as it is easier to get up-and-running in Python and those features depend significantly less on low-latency code.

In live-trading, our infrastructure will resemble:



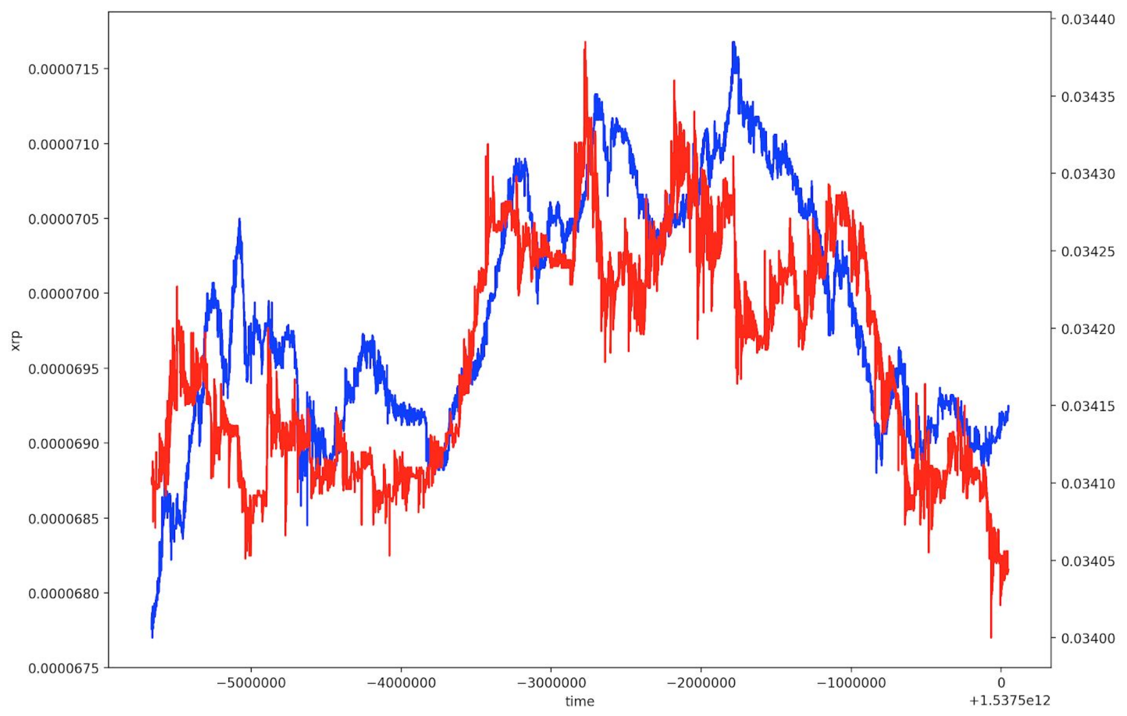
In reality, we will likely have different/more exchanges we listen to for marketdata. We will use (at least) two databases, one which records the trades that our system made, and one that records every piece of market data using a uniform format, so we can simulate trading days later. Through live testing, we will determine which exchange has the deepest liquidity for our trades (this is important, as in a low-liquidity exchange, incoming price data might not match outgoing execution prices).

Our simulation environment will look nearly identical, with minor changes:



The trading algorithms, risk monitor, and execution engine will be identical to our live environment (the only difference is that will instantiate the execution engine and trading strategies with the testing objects, instead of their live counterparts. Dependency injection allows us to abstract away information about the actual exchanges, and instead only worry about the incoming/outgoing data). We will just be replaying historical market data from the database. We will use a simulated exchange that records things like profit/loss, slippage, Sharpe ratio, and volatility of our profits. The historical testing will be not only for training our ML models, but as to do basic regression analysis. A core trading strategy will also be exploiting correlations in different crypto prices.

For example, an early iteration of our code produced this graph (using matplotlib) which shows a price trend within Ethereum and Ripple (adjusted for the scale of the prices):



Which shows the prices are clearly correlated. If we can find a lagging indicator for one of the prices in terms of the other, we can predict price changes and trade accordingly.

In addition, we will attempt to support the Linux and macOS operating systems. This restriction is not self-imposed, but rather a reflection of our development environments and final production environment. The team is working on a combination of Ubuntu and macOS, and we hope to run this software on Amazon Web Services, which supports a variety of Linux distributions. Fortunately, these operating systems are relatively simple to co-develop for on account of their shared heritage and near POSIX compliance.

Finally, we are constrained by the libraries available to us. Thankfully, there is a myriad of powerful open source libraries such as TensorFlow that are available to be used freely. Furthermore, there are libraries such as matplotlib and pybind which allow us to streamlight data analytics and interaction between C++ and Python codebases.

On the business side, we are constrained by our schedule. The project has predefined deadlines, and we are also limited by the availability of the team members and external time commitments. Together, these set the scope of our project, and disallow us from expanding it any further. Many trading teams take years to set up their core infrastructure and testing/learning environments - clearly, we will not be able to have this much flexibility, so we will have to skip some features. For example, many trading teams have a system which automatically takes the recorded market data from that day of trading, updates their trading models overnight, and then executes the new trading models the next day with no programmer interaction. This frees the developers to work on new strategies. In contrast, we will likely have to manually re-train our models on new data, and re-deploy the updated strategies to the trading server.

Our budget will determine where we are able to run our code, the third party software which we can license, and our ability to trade cryptocurrencies. We hope to acquire the funding to deploy and maintain our project on Amazon Web Services. We will likely be able to rely totally on open source and similarly licensed free software. However, we will need to interface with cryptocurrency exchanges, which typically have fees associated with making a trade. Finally, we will require initial funds in order to purchase cryptocurrency for our program to participate in real trades. In order to save money, we will attempt to tune and train our algorithms throughout development by simulating trades. Using real data, we would record trades the algorithm would make in real-time, but without actually trading. We would then go back over our trades and evaluate how successful the algorithm would have been. This is a significant constraint, because we cannot perfectly simulate how a real trade would play out. We will need to make assumptions on latency, and how our trades would actively affect the exchange, if at all.

Ethical and Intellectual Property Issues

Ethically, the primary issue of the project relates to its contributions to society as a whole. At its core, the goal of the project is to maximize profitability for the people using the algorithm to trade cryptocurrencies. In this regard, the project will only directly benefit its creators and will have little to no positive effect on the rest of society. Concerns regarding negative effects to society are also present. Simply put, the cryptocurrency market is a competitive environment, and one party cannot move up without another moving down. Some refer to algorithmic trading as a tax on retail investors, as the bots can get the best deals faster than any human ever could. As the project is contained to cryptocurrencies rather than traditional stocks, this issue is less concerning, but it still warrants mention. Ultimately, the potential negative impacts of the project are minimal, while the benefits are potentially very high, if even for only a small group of people, so we believe it is ethical.

Intellectual property will be slightly complicated for the project. There are two primary areas that will require care to avoid any infringement. First, the trading algorithm itself will need to be fairly unique. There are numerous sources of trading algorithms for the stock market available online in various developer blogs, research articles, and other media. Most of this material is not obviously licensed in any way, but it is also generally only a high level description of the algorithm or technique. As long as no actual code is reused, use of the ideas available online will be safe. Second, the machine learning algorithms used to train and predict information about trades will be a mix of available tools and original code. Luckily, ML tools such as Scikit and Tensorflow are very generous with their licensing, however, it must be verified that any tools we use allow financial gain from their use.

Updated October 22, 2018: Finalized project proposal by adding project budget, work plan, gantt chart, preliminary project design, as well as ethical and intellectual property issues.

Updated February 5, 2019: Prepared proposal for final project design by refining synopsis, simplifying project milestones, clarifying work roles, refining budget to reflect cloud computation usage, as well as updating project design.