SMPE: Stock Market Prediction on Edge

Bryan Chan, Zi Yi Chen, Qi Zhao
University of Toronto

ABSTRACT

Stock market predictions are usually done on the cloud nowadays, but stock prices fluctuate frequently. Price changes can occur within a second. Hence, market traders often cannot get the up-to-date price predictions in time due to high latency between the computation node and the user. In this paper, we attempt to reduce the latency by leveraging the edge computing technology, offloading the prediction computation process on the edge server instead of the cloud. We built a simple mobile-edge-cloud system, and our evaluations show that with the use of edge nodes, we can reduce the latency between the cloud and the mobile users by more than 35%. Our method also reduces the mobile device's bandwidth usage significantly, making it much more scalable.

KEY WORDS: Edge computing; Latency; Cloud; Stock Market Prediction; Bandwidth

1. INTRODUCTION

Stock market prediction is the act of determining the future value of public stocks or other financial instruments traded on an exchange [17]. Successful predictions of stocks' future prices can yield great profit.

Traditional prediction methods are time consuming and they involve large data sets. The methods include fundamental analysis and technical analysis [18]. Fundamental analysis evaluates a company's past performance as well as the credibility of its accounts and it only focuses on the company's stock itself. However, technical analysis is not concerned with any of the company's fundamentals. It tries to determine the future price of a stock based solely on the trends of the past price [18]. Aside from these two traditional prediction methods, with the advent of the digital computer, stock market prediction has since moved into the machine learning era. Recurrent neural network (RNN) is widely used for stock market prediction, and it usually performs better than the traditional methods [18].

However, training and testing a neural network for stock market prediction are time consuming processes. Training requires days of computations and inference requires seconds of computations due the complexity of the neural networks. Hence, predictions are done on the cloud [19].

This fact conflicts with market traders needing to get price predictions in real time. Also, there is a huge connection latency between the cloud and market traders, which increases the difficulty to obtain the predictions in time. Market traders nowadays tend to take advantages of mobile devices to make purchase decisions [20] but the latency between the cloud and the mobiles over WAN are much higher compared to over LAN [25]. Hence, low latency between mobile and cloud plays a significant role in traders making deals in time.

We attempt to take advantage of edge computing and build a simple mobile-edge-cloud system. We propose to offload the prediction computations onto the edge and establish a faster connection between the mobile device and the edge through a closer geographic proximity. In addition, this architecture also has other potential benefits such as energy consumption and lower bandwidth usage.

The rest of this paper is organized as follows. In section 2, we provide some required background information related to stock market prediction on the edge (SMPE). In section 3, we explain the design and architecture of our system, followed by evaluations in section 4. In section 5, we present our conclusions and some future work.

2. RELATED WORK

In this section, we will present some background knowledge on edge computing and time series analysis.

2.1 Edge Computing

Edge computing is a distributed computing paradigm in which computation is largely or completely performed on distributed device nodes known as smart devices or edge devices as opposed to primarily taking place in a centralized cloud environment [21]. The motivation is to provide server resources and data analysis closer to the sources of data, promising to deliver shorter latency and higher bandwidth. It can benefit applications which require high bandwidth, low latency but without large scale aggregation [22]. In stock market prediction, on one hand, mobile users want to get prediction prices as soon as possible. On the other hand, the historical data is huge and latency between the cloud and the mobile devices are high. Therefore, we think it is useful to leverage edge computing into stock market prediction.

2.2 Time Series Analysis

Time series analysis is a statistical technique that deals with time series data, or trend analysis [23]. With time series analysis, we can obtain an understanding of the underlying forces and structure that produced the observed data, allowing analysts to fit models and proceed to forecasting, monitoring, or feedback and feedforward control [24]. As mentioned previously, we use an RNN, specifically a Long-Short Term Memory model (LSTM) to train a prediction model, which is a common method in time series analysis.

A LSTM model is an RNN composed of LSTM units. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM was proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber [26] and improved in 2000 by Felix Gers' team [27]. As of 2016, major technology companies including Google, Apple, and Microsoft were using LSTM as fundamental components in new products.

3. DESIGN & ARCHITECTURE

In this section, we describe three scenarios where the predictions will most likely occur, and present a network topology that follows the scenarios. Additionally, we provide a description of the prediction model used in order to reflect some issues described in Section 1. At last, we provide a high level overview of the system.

3.1 Scenarios

Three scenarios are compared in Section 4. Traditionally, computations are performed on either cloud or mobile devices. Both scenarios have their respective advantages and disadvantages. Depending on the application usage, developers must decide which device to perform the computations in order to minimize their specified loss. To reduce such burden from the developers, we propose a third scenario where we use an edge node to mitigate the disadvantages in former scenarios, while preserving the advantages and providing flexibility and functionalities.

3.1.1 Cloud

Many deep neural network applications such as Google Translate, Siri, Alexa, Carat, and SnapChat offload the computations to the cloud when connectivity can be established. In this scenario, the prediction model is stored on the cloud and the mobile device will rely on the cloud to perform the prediction. This situation allows the model to have higher complexity since cloud provides “unlimited” computation resources [2]. An extra benefit of using cloud is the ability to reduce the bandwidth required to perform predictions. Based on [3, 4, 5], removing the heavy computations on the mobile device should increase the battery life. However, due to the geographic proximity between the users and cloud, the latency will be high [6, 7, 8]. As mentioned in Section 1, high latency imposes a big challenge where day traders require near real-time predictions to make split-second decisions. We expect the latency between the cloud and mobile device to be the overhead, thereby disallowing day traders to retrieve real-time predictions.

3.1.2 Mobile Device

On the other hand, without connectivity with the cloud, applications mentioned before workaround the heavy computation by caching a simpler neural network model on the mobile device [9]. Moreover, some applications such as Prisma manage to use deep neural network without offloading computations. In this scenario, the prediction model is stored and utilized for inference on the mobile device. This mitigates the latency problem in the cloud scenario where there is a delay between the phone and the cloud. However, power consumption will increase as heavy computations are performed on the mobile device. Additionally, the inputs for inference will be directly retrieved from data sources and this may impose a high bandwidth requirement. We expect the prediction latency to be the overhead in this scenario, in addition to increasing energy consumption. This would impose challenges where day traders need to carry battery chargers and potentially suffer from the prediction delay.

Another challenge arises where the prediction model needs to be less complex without sacrificing prediction accuracy. Although accuracy is not the scope of this paper, it is important for real-life use cases. The simpler model would either sacrifice the accuracy or require massive development time from developers to preserve the accuracy. Recent researches are looking into bringing deep neural networks into mobile devices [10, 11].

3.1.3 Edge

Both previous scenarios suffer from the latency problem. The mobile device scenario may suffer from other aspects such as power consumption and high bandwidth usage. We thereby propose the usage of edge to bring a stronger computational device closer to mobile devices. Utilizing the edge would maintain the benefits of cloud and reduce the latency problem. In this scenario, we assume an edge node to be the Wi-Fi access point of the mobile device. Additionally, we assume that the edge node has better hardware than the mobile devices. An example of an edge node can be a laptop with Wi-Fi hotspot. The prediction model will then be stored on the edge node and will be utilized for inference.

With a lower geographic proximity, we expect the latency between the edge and mobile devices to be lower than the cloud scenario. Moreover, the bandwidth will be similar. On the other hand, the energy consumption of the mobile devices and bandwidth should be lower compared to the mobile device scenario. This scenario should benefit day traders as they can now receive real-time prediction and saving battery life during commute.

3.2 Network Topology

To realize the scenarios in Section 3.1, we constructed a realistic network topology to imitate real-life situations (See Figure 1). All scenarios will have the mobile device connect to a Wi-Fi access point. For simplicity, we assume the access point to be a laptop providing hotspot. The edge will then have a wired connection to the cloud and the API for retrieving the data inputs. In the cloud scenario, the user invokes a request. The packets then traverse through the edge and to the cloud. The cloud retrieves information from the stock price API and performs the inference. The result is then returned through the same path in the opposite direction. In the edge and mobile device scenarios, the user invokes a request. The packet will traverse to the edge. The edge retrieves the information from the API. The result is then returned to the mobile device. If the current scenario is edge, the inference is done prior to returning. Otherwise, the result is simply the API information and inference is done on the mobile device.

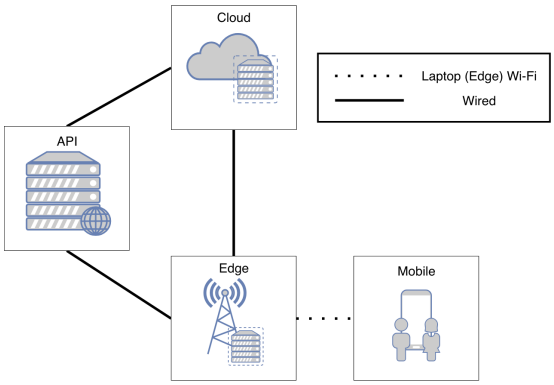

Figure 1: This network topology represents the three scenarios in Section 3.1

3.3 Prediction Model

As mentioned in Section 2, LSTM models are the state of the art for time series analysis. Since we want to compare the mobile device scenario, we only constructed a deep neural net model lightweight enough to be used by the mobile device. Realistically, the model should consist of more complex layers with different functionalities such as Attention and Temporal Convolution [12, 13, 14].

Our model is a three-layered LSTM, followed by a fully connected layer (See Figure 2). The input is 100 consecutive seconds of historical data of a given symbol. Each point consists of features: open, close, high, and low. The LSTM layers consist of 128, 256, and 512 hidden units, respectively. It then uses a fully connected layer to return a float, representing the closing price 10 seconds after the last input time. The gap can be modified depending on the developer.

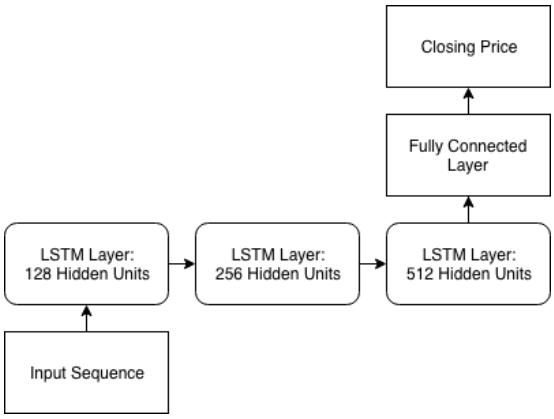

Figure 2: The LSTM model used for stock market prediction. The output is the closing price of 10 seconds after the last input date

3.4 System Implementation

We built a prototype to emulate the scenarios. The LSTM model is built using TensorFlow and is trained offline. This model will be used across all scenarios to keep consistency. We chose Alpha Vantage [1] for our API as it is free. We built a Python Flask RESTful service and dockerized the web service for faster deployment [15, 16]. For convenience, we developed an Android application using Xamarin framework to simulate all three scenarios. It provides an interface for users to see the latency breakdown by invoking a request, as well as the prediction result. We picked Microsoft Azure as our cloud platform as it provides competitive hardware with free of charge. Section 4 will provide more details on the setup: physical locations of the nodes, hardware used, and the results from the Android application.

4. EVALUATION

In this section we first give an overview of our execution environment including hardware specifications, then we present our findings of this architecture with latency and bandwidth analyses.

4.1 Execution Environment

We used the Asus laptop with Intel Core i7 with four cores and 8GB RAM as our edge server for the experiment. To

make the playing field even, we also used 4 cores and 8GM RAM on our Microsoft Azure cloud node. For the mobile device, we used an android mobile phone with Snapdragon 625 processor and 3GB RAM.

4.2 Overall Latency

To measure the overall latency, we performed 20 prediction trials with each of the three different scenarios and took the average and standard deviation. Our results are shown in Figure 3. Predictions on cloud, mobile device, and edge resulted average latencies of 2145 ms, 4258 ms, and 1378 ms respectively. These numbers show that our edge server has reduced the latency by 67% compared to prediction on mobile phone and 36% compared to prediction on the cloud.

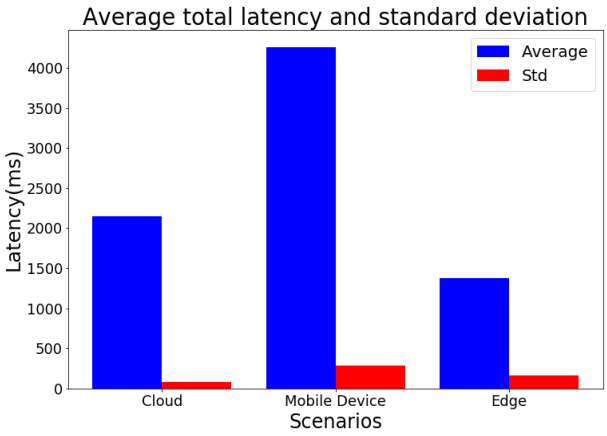

Figure 3: Average total latency and standard deviation over three scenarios. Results are in milliseconds

4.3 Latency Analysis

To further analyze how the edge server help reduced the latency, we splitted our total latency numbers into three components and analyze where the bottleneck occured in our scenarios. The stock API latency is the time it took to retrieve the 100 historical prices from the stock price API. The prediction latency is the time it took to compute the stock prediction. Lastly, the communication latency is the time it took to transmit the prediction from the compute node to the mobile device. As expected, Figure 4 shows our mobile device had the highest stock API latency and prediction latency with its slow Wi-Fi connection and processor speed, but it had the lowest communication latency since the prediction was computed on the mobile device itself. The stock API and prediction latencies are similar between the edge and the cloud servers due to their similar hardware specifications. However, the edge server managed to have a lower communication latency compared to the cloud server by being physical at the edge of the network and closer to the mobile client.

4.4 Bandwidth Analysis

Computing prediction on the mobile device will incur significant bandwidth consumption as the file size of the 100 historical prices from the stock API is 240KB. Stock traders often track hundreds of different stocks simultaneously, which will consume bandwidth in the order of megabytes per second. When predictions are computed on the edge or the cloud, the mobile device only consumes approximately 100 bytes of bandwidth per stock by getting only the predicted prices.

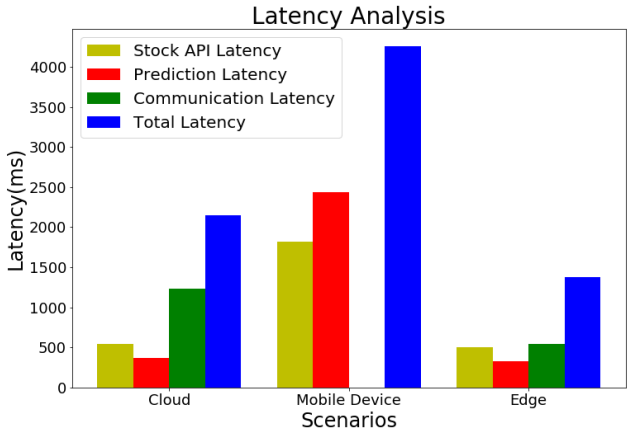

Figure 4: Latency analysis by splitting total latency into stock API latency, prediction latency, and communication latency. Results are in milliseconds

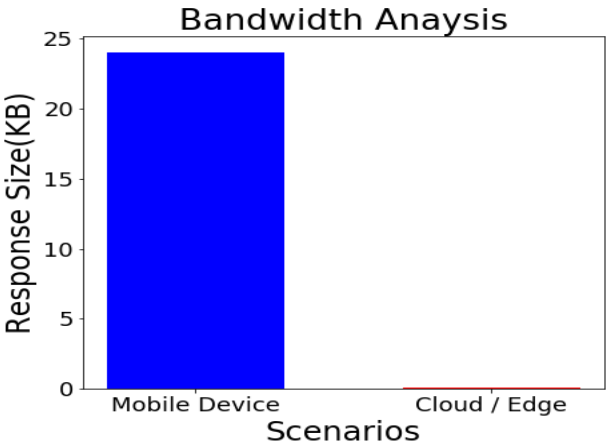

Figure 5: Bandwidth analysis on the response sizes from three scenarios. Results are in KB

5. CONCLUSION & FUTURE WORK

Stock market prediction on edge (SMPE) is an architecture that computes the stock price predictions on the edge server. Our prediction model uses LSTM and takes 100 historical prices as the input to the model. The prediction algorithm is deployed onto cloud and edge servers' Docker containers. We performed experiments on computing predictions on the

mobile device, cloud server and edge servers to show that edge server can reduce latency and bandwidth consumption.

In the future, we plan to explore the stream data processing where users can get real-time update & prediction with the app open. We also intend to experiment with different prediction models to improve the accuracy of our prediction results.

REFERENCE

[1] “ALPHA VANTAGE” www.alphavantage.co/.

[2] Fox, Armando, et al. "Above the Clouds: A Berkeley View of Cloud Computing." Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28.13 (2009): 2009.

[3] Ha, Kiryong, et al. "Towards Wearable Cognitive Assistance." Proceedings of the 12th annual international conference on Mobile systems, applications, and services. ACM, 2014.

[4] Li, Dawei, et al. "Deepcham: Collaborative Edge-Mediated Adaptive Deep Learning for Mobile Object Recognition." Edge Computing (SEC), IEEE/ACM Symposium on. IEEE, 2016.

[5] Qian, Hao, and Daniel Andresen. "Extending mobile device's battery life by offloading computation to cloud." Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems. IEEE Press, 2015.

[6] Claypool, Mark, and David Finkel. "The Effects of Latency on Player Performance in Cloud-Based Games." Network and Systems Support for Games (NetGames), 2014 13th Annual Workshop on. IEEE, 2014.

[7] Zhang, Wuyang, et al. "Towards Efficient Edge Cloud Augmentation for Virtual Reality MMOGs." Proceedings of the Second ACM/IEEE Symposium on Edge Computing. ACM, 2017.

[8] Popescu, Diana Andreea, Noa Zilberman, and Andrew William Moore. "Characterizing the Impact of Network Latency on Cloud-Based Applications’ Performance"(2017)

[9] Good, Otavio. “How Google Translate Squeezes Deep Learning onto a Phone.” Google AI Blog, Google, 29 July 2015,ai.googleblog.com/2015/07/how-google-translate-squeezes-deep.html.

[10] Franklin, Dustin. “NVIDIA® Jetson™ TX1 Supercomputer-on-Module Drives Next Wave of Autonomous Machines.” NVIDIA Developer Blog, NVIDIA, 5 Sept. 2018, devblogs.nvidia.com/NVIDIA-jetson-tx1-supercomputer-on-module-drives-next-wave-of-autonomous-machines/.

[11] McDonough, Tim. “Live from New York, It's Snapdragon 820: Prepare for an Immersive Dive into Mobile Experience.” Qualcomm, Qualcomm, 10 Mar. 2016, www.qualcomm.com/news/onq/2015/11/10/live-new-york-its-snapdragon-820-prepare-immersive-dive-mobile-experience.

[12] Du, Shuyang, Madhulima Pandey, and Cuiqun Xing. "Modeling Approaches for Time Series Forecasting and Anomaly Detection."

[13] Yao, Huaxiu, et al. "Modeling Spatial-Temporal Dynamics for Traffic Prediction." arXiv preprint arXiv:1803.01254 (2018).

[14] Tran, Dat Thanh, et al. "Temporal Attention-Augmented Bilinear Network for Financial Time-Series Data Analysis." IEEE transactions on neural networks and learning systems (2018).

[15] Ma, Lele, Shanhe Yi, and Qun Li. "Efficient Service Handoff Across Edge Servers Via Docker Container Migration." Proceedings of the Second ACM/IEEE Symposium on Edge Computing. ACM, 2017.

[16] Rad, Babak Bashari, Harrison John Bhatti, and Mohammad Ahmadi. "An Introduction to Docker and Analysis of its Performance." International Journal of Computer Science and Network Security (IJCSNS) 17.3 (2017): 228.

[17] Graham, B. The Intelligent Investor HarperCollins; Rev Ed edition, 2003.

[18] Yudong, Zhang , and W. Lenan . "Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network." Expert Systems with Applications 36.5(2009):8849-8854

.[19] Kumar, Jitendra, R. Goomer, and A. K. Singh. "Long Short Term Memory Recurrent Neural Network (LSTM-RNN) Based Workload Forecasting Model For Cloud Datacenters." Procedia Computer Science 125(2018):676-682.

[20] Boateng, Richard . "Mobile phones and micro- trading activities – conceptualizing the link." Info 13.5(2011):48-62.

[21] K. Hong et al., “Mobile fog: A programming model for large-scale applications on the internet of things,” in MCC, 2013.

[22] Zhang, Wuyang, et al. "Towards efficient edge cloud augmentation for virtual reality MMOGs." Proceedings of the Second ACM/IEEE Symposium on Edge Computing. ACM, 2017.

[23] Brockwell, P. J., & Davis, R. A. (1991). Time Series: Theory and Methods (2nd ed.). New York: Springer-Verlag.

[24] NIST/SEMATECH e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/, date.

[25] Dodonov, Evgueni, Rodrigo Fernandes de Mello, and Laurence Tianruo Yang. "A Network Evaluation for LAN, MAN and WAN Grid Environments." International Conference on Embedded and Ubiquitous Computing. Springer, Berlin, Heidelberg, 2005.

[26] Gers, F. A., J. Schmidhuber, and F. Cummins. "Learning to forget: continual prediction with LSTM." Neural Computation 12.10(2000):2451-2471.

[27] Graves, Alex. Long Short-Term Memory. Supervised Sequence Labelling with Recurrent Neural Networks. 2012