Algorithmic Trading and Machine Learning Based on GPU

Mantas Vaitonis
Vilnius University Kaunas Faculty
Muitinės street. 8,
LT-44280 Kaunas, Lithuania
mantas.vaitonis@knf.vu.lt

Saulius Masteika
Vilnius University Kaunas Faculty
Muitinės street. 8,
LT-44280 Kaunas, Lithuania
saulius.masteika@knf.vu.lt

Konstantinas Korovkinas
Vilnius University Kaunas Faculty
Muitinės street. 8,
LT-44280 Kaunas, Lithuania
konstantinas.korovkinas@knf.vu.lt

***Abstract*— This paper investigates the speed improvements available when using a graphics processing unit (GPU) for algorithmic trading and machine learning. A modern GPU allows hundreds of operations to be performed in parallel, leaving the CPU free to execute other jobs. Several issues related to implementing algorithmic trading and machine learning on GPU are discussed, including limited programing flexibility, as well as the effect that proper memory layout can have on speed increases when using GPU devices. An empirical research of algorithmic trading on GPU is presented, which showed the advantage of the GPU over CPU system. Moreover the machine learning methods on GPU are presented and the findings of this paper may be applied in future works.**

Keywords— high frequency trading; machine learning; GPU; high performance computing; genetic programming.

I. INTRODUCTION

Nowadays standard computers come with sequential CPUs or with multicore CPUs, which allow a limited number of processes to be executed in parallel. What is important here is that this hardware is strongly parallel and may operate independent from the main CPU. A modern GPU allows hundreds of operations to be performed in parallel, leaving the CPU free to execute other jobs. In particular, GPUs offer hundreds of processing cores, but they can be used simultaneously only to perform data parallel computations. Moreover, GPUs usually have no direct access to the main memory and they do not offer hardware managed caches; two aspects that make memory management a critical factor to be carefully considered [1].

GPU architectures are specialized for computeintensive, memory-intensive, highly parallel computation, and therefore are designed such that more resources are devoted to data processing than caching or control flow. State of the art GPUs provide up to an order of magnitude more peak IEEE single-precision floating-point than their CPU counterparts. Additionally, GPUs have much more aggressive memory subsystems, typically endowed with more than 10x higher memory bandwidth than a CPU. Peak performance is usually impossible to achieve on general purpose applications, yet capturing even a fraction of peak performance yields significant speedup. GPU performance is dependent on finding high degrees of parallelism: a typical computation running on the GPU must express thousands of threads in order to effectively use the hardware capabilities. Algorithms for machine learning applications will need to consider such parallelism in order to utilize many-core processors. Applications which do not express parallelism will not continue improving their performance when run on newer computing platforms at the rates we have enjoyed in the past. Therefore, finding large scale parallelism is important for compute performance in the future. Programming for GPUs is then indicative of the future many-core programming experience [2].

When searching for “GPU back-testing software” almost no results appear. The technology is very difficult to use and implement across a general back-testing.

The problem is the way in which a GPU works and the way in which general purpose back-testing works. Most of these back-testing programs have a language like MQL4, Ninjascript. These languages are used to construct trading systems that the simulator executes by performing some sort of parsing of the scripted code. This approach gives flexibility because researchers can code whichever strategy they can think of with whatever logic and the simulator will be able to handle it. The strategy coded is in essence a function that the simulator then uses to execute code within its back-testing engine. However, when trying to move this type of thinking to the GPU researches go into lots of problems [3].

The work reported in this paper aims to present literature review of GPU benefits on algorithmic trading and machine learning. The overview of the uses of machine learning and algorithmic trading on GPU is presented. Both topics are presented separately and the results will be used for future works in machine learning with high frequency trading on

GPU. The paper also presents high frequency algorithmic trading results when applied on CPU and GPU.

The rest of the paper is organized as follows: theory and the problem statement are presented in Sections 1 and 2. Sections 3, 4, 5 and 6 give an overview of: GPU for hardware acceleration, high frequency trading, GPU in high performance computing and GPU in machine learning The results and the summary of the research, followed by conclusions in Section 7.

II. OBSTACLES USING GPU

GPU is a very limited machine in terms of programming flexibility. It is not possible just to code the system within a script and send it to a GPU back-tester. If researchers want the GPU to perform a trading system simulation they will need to code the entire system and simulator within the same function and have the GPU run that in a batch process.

Introducing things like double loops and random access patterns is hard for the GPU. When writing simulations for a GPU it is necessary to ensure that everything that is random access intensive or conditional intensive is pre-calculated and passed to the GPU. Therefore, something that is "general purpose" starts to become very hard to pre-calculate and interactively build the entire simulator-plus-system code to load it into the GPU and perform the simulations. There are many ways in which GPU technology is currently being used in trading. Traditionally they have been used to execute simulations that are very specific and parallelizable – such as pricing simulations, machine learning training and high frequency trading algorithms.

When looking for something very general the GPU tends to be a hard solution. However, if one is interested in some particular trading problem then there's a big chance that researchers would be able to benefit from it if they are willing to spend the time, energy and money necessary to create a custom GPU implementation [3][4].

III. GPU FOR HARDWARE ACCELERATION

Hardware acceleration is achieved by utilizing specific hardware to gain higher computational results than those provided by general purpose CPU. Most devices intended for intense calculations include Field-Programmable Gate Array (FPGA), IBM's Cell Broadband Engine Architecture (Cell BE or, simply, Cell) and Graphics Processing Units (GPUs). Until recently GPU remained on fringes of HPC (high performance computing) mostly because of the high learning curve caused by the fact that low-level graphics languages were the only way to program the GPUs. However, now NVIDIA has come out with a new line of graphics cards – Tesla [4].

One of NVIDIA GPUs main features is ease of programmability made possible with CUDA – Compute Unified Device Architecture. With a low learning curve, CUDA allows developers to tap into enormous computing power of GPUs yielding high performance benefits [5]. As mentioned in the introduction, we use the compute unified device architecture (CUDA), which allows for implementation of algorithms using MATLAB with CUDA specific extensions [5]. When a program using CUDA extensions and running on the CPU invokes a GPU kernel, many copies of this kernel – known as threads – are enumerated and distributed to the available multiprocessors, where their execution starts [4].

The two main criteria for algorithmic trading are speed – that is the speed with which the same set of computations can be performed on multiple sets of data – and programmability. For this principle, general-purpose hardware – such as Intel Central Processing Unit (CPU) – is not suitable. The CPU is designed to execute commands in a linear fashion, however, the task at hand will benefit most from parallelization as the same calculations are required to be performed on multiple data; this is where parallelization and hardware acceleration come into play.

IV. HIGH FREQUENCY TRADING

The developments in computer technology have changed the way financial instruments are traded. A significant part of trades is handled without human intervention, where trading algorithms make trading decisions. Although the concept of algorithmic trading is not brand new, the speed in which algorithmic trading operates has grown tremendously over the past ten years.

The trade execution time has grown from daily trading to microseconds and even nanoseconds. Due to the increase in speed, a huge number of orders and order cancellations are required. Profit chances for high frequency traders are very time-sensitive and low latency for trade execution is of the main importance. Thus, HFT firms invest in high-speed connections and place their trading platforms close to the stock market servers via co-location [6].

Nowadays, financial markets are fully automated, consisting of algorithmic trading, thus, they are largely dominated by high frequency trading. High frequency trading platforms have replaced the traditional auction-like floor where traders compete on price [7]. The main focus of HFT is to beat the time. The algorithm waits till the trader buys a certain amount of any financial instrument at any given time, then the high frequency traders use this information to change the price it is quoting in the market [8][9][10][11]. The economics and finance academic community consider HFT as beneficial to the market because HFT provides liquidity and, therefore, facilitates the flow of commerce in the capital markets [11].

Given the fact that high frequency trading has to be done in milliseconds or even nanoseconds, all trading must be performed by using supercomputer. In real life, depending on the trade, trading opportunities can last from nanoseconds to minutes or even hours.

Trading strategies, used by high frequency traders, seek for the opportunity to exploit short-lived trading in the markets that would not be possible to find or identify in other way than high-speed processing power of computers. These trading opportunities are very small abnormalities in the pricing of financial instruments that result in extra low profit per trade.

High frequency earns higher profit as it is possible to trade in big volumes. Thus, profit can be generated from these small changes in the prices. One of the advantages of HFT is that it provides liquidity and helps to ensure the efficiency of prices for financial assets [12].

V. GPU IN HIGH PERFORMANCE COMPUTING

High-frequency trading (HFT) is a specialized form of Algorithmic trading, where the execution of computerized trading strategies is characterized by extremely short position-holding periods – just a few seconds or even down to milliseconds. The success of an HFT algorithm depends on its ability to react to a situation faster than others. This has given birth to another variant of HFT called Ultra High Frequency Trading (UHFT). Here, the execution of trades happens in sub-millisecond times. The technology used by UHFT traders is co-location of servers with exchange, direct market access, using parallel processing on GPUs and using special hardware like FPGAs [13].

Consolidated Tape Association(CTA) oversees the collection, processing and dissemination of consolidated quote and trade data at NYSE. Securities Information Processor(SIP), is the technology that enables collecting quote and trade data from the exchanges, consolidating it, and sending it out as a continuous stream of best bids and offers (quotes) and last sales (trades). SIP has to work at enormous speed. On average, NYSE handles average 2 lakh quotes per second out of which 28000 per second get converted into trades. The traders talk to the exchanges using FIX protocol. FIX stands for Financial Information eXchange. The standard is managed by a nonprofit organization called FIX Trading Community. The message consists of ASCII characters and the format is an extension of XML, called FIXML. Recently Citibank has announced that it will provide FIX functionality to NSE in India [13].

There is increasing use of High Performance Computing platforms like GPU multiprocessing and FPGA. D.HFT algorithms. They are fast and parallelizable. They are specifically designed to make money by exploiting tiny, lightningfast price changes in shares[13][14].

A. GPU in high frequency trading

During our research algorithmic trading strategy [8] was used on CPU Intel i5 - 3230M 2,6 GHz with two cores (2 MATLAB worker) and GPU GeForce 710M with 96 CUDA cores. Firstly we applied the pair trading strategy only on CPU and then on CPU together working with GPU.

The nanosecond data used for experiment was provided by Nanotick company. Futures contracts were from ME group which consists of NYMEX, COMEX and CBOT. Nanotick provided five different futures commodity contracts: NG (natural gas), BZ (Brent crude oil), CL (crude oil), HO (NY Harbor ULSD) , RB (RBOB Gasoline). Time period of commodity futures contracts was from 01-08-2015 to 31-08-2015.

During the research pair detection, detection of buy/sell signals, the trading and profit calculation were parallelized when implemented on CPU and GPU [8]. When these functions were parallelized it was no longer necessary to wait for one function to stop and start the other one. The multiple calculations with multiple functions were possible.

The research aim was not to measure the profit of the strategy but to improve the speed of algorithm by using GPU. The same pair trading strategy was applied to CPU and later to CPU working together with GPU. In the table below we can see the amount of records pairs trading algorithm had to process and how much time did it take using CPU and GPU.

TABLE I. CPU and GPU comparison

Date	**Intel i5 - 3230M 2,6 GHz,2 cores (in seconds)**	**GeForce 710m, 96 CUDA Cores (in seconds)**	**Number of records processed**
2015-08-03 till 2015-08-31	74777,4	58378,53	124789970

Table 1 shows trading time of algorithm using different hardware CPU (Intel i5 - 3230M 2,6 GHz,2 cores) and GPU (GeForce 710m, 96 CUDA Cores). The total number of records processed was 124789970 for each simulation.

The more detailed information is presented in figure below where the speedup difference is presented.

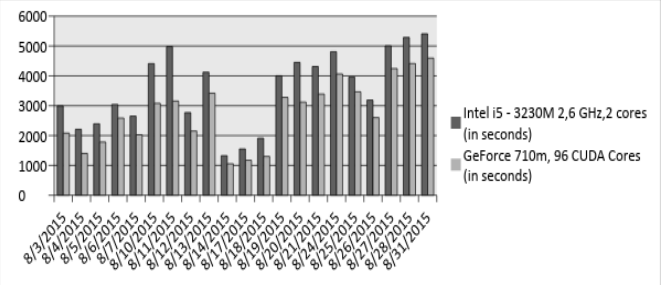

Fig. 1. Comparison of CPU and GPU using HFT in seconds

As shown in figure above the pair trading algorithm speed of simulation did improve varying from 12% to 36% when used on GPU instead of just CPU. The difference of speed for different days occurs due to different number of trades made and different number of trade signals. The more parameters are possible to make parallel and move to GPU, the bigger speedup is possible to achieve. During this experiment bigger the matrix of trades and pairs were used the more measurable was the speed up by GPU. The results show the importance of technical advantages in HFT and how important is to improve the algorithm in order to use the most of the hardware it is presented to.

B. *Stock trading using genetic programming on GPU*

D. McKenney and T. White [14] did present their research on stock trading using genetic programing on GPU. Within this work, genetic programming was used in an attempt to solve the real-world problem of stock trading strategy generation. A GPU device was used to evaluate individuals within the GP population through stack-based interpretation (due to the lack of recursion support on many GPU devices). With a small amount of memory access optimization, a speedup factor of over 600 was reached when compared to a sequential evaluation of the same data running on a 2.4Ghz CPU. The effect of increasing the size of the training set (through the addition of more stocks and longer training periods) was also investigated. It was found that using small training sets resulted in the worst testing results. Furthermore, the best test results were found when using the largest training sets. These results supported the hypothesis that analyzing more stocks over a longer period of time can generate a more general and effective stock trading strategy. The speedup gained using GPU devices for evaluation enable this large training set to be evaluated quickly, while a sequential implementation would make this approach unfeasible. Finally, several areas of improvement for both GP on GPU and stock trading strategy creation using GP were identified. Continuing work and addressing these possible areas of improvement may result in faster evaluation of individuals, as well as a much more profitable trading solution [14].

VI. GPU IN MACHINE LEARNING

The use of GPUs in machine learning is widely used in recent years. The most promising machine learning algorithm is SVM, that can be conveniently adapted to parallel architectures. During the last decade, many works have been done for accelerating the time-consuming training phase in SVM on many-core GPUs. Catanzaro et al. in [2] first proposed the GPUSVM for binary classification problem and achieved speedup of 9-35× over LIBSVM running on a traditional processor. Later Herrero-Lopez et al. in [18] improved Catanzaro's work by adding the support for Multiclass classification. They achieved the speedups in the range of 3-57x for training and 3-112x for classification. Carpenter in [19] presented cuSVM, a software package for high-speed Support Vector Machine (SVM) training and prediction that exploits the massively parallel processing power of Graphics Processors (GPUs). Other authors in papers [15][17][23] also reported that GPU optimization of SVM achieves better performance to compare with CPU. Vaněk et al. in [20]. introduced a novel GPU approach of the support vector machine training: Optimized Hierarchical Decomposition SVM (OHD-SVM). It uses a hierarchical decomposition iterative algorithm that allows using matrix-matrix multiplication to calculate the kernel matrix values. They declared that algorithm is significantly faster than all other implementations for all datasets. The biggest difference was on the largest datasets where they achieved speed-up up to 12 times in comparison with the fastest already published GPU implementation.

Another challenging research area is Deep Learning, which largely involve simple matrix manipulations and are therefore well suited to be implemented on graphic processors. Raina et al. in [21] developed general principles for massively parallelizing unsupervised learning tasks using graphics processors and shown that these principles can be applied to successfully scaling up learning algorithms for both deep belief networks (DBNs) and sparse coding. Their implementation of DBN learning is up to 70 times faster than a dual-core CPU implementation for large models. Dean et al. in [22] presented that training large deep learning models with billions of parameters using 16000 CPU cores could dramatically improve training performance. Krizhevsky et al. in [29] showed that training a large deep convolutional network with 60 million parameters and 650,000 neurons on a large data set was in great performance based on GPU processors [16]. Coates et al. in [24] presented their own system based on Commodity Off-The-Shelf High Performance Computing (COTS HPC) technology: a cluster of GPU servers with Infiniband interconnects and MPI. Their system is able to train 1 billion parameter networks on just 3 machines in a couple of days, and they showed that it can scale to networks with over 11 billion parameters using just 16 machines. They have shown that can comfortably train networks with well over 11 billion parameters—more than 6.5 times as large as the one reported in [22] (the largest previous network), and using fewer than 2% as many machines. Chen et al. in [25] implemented a variant of the deep belief network (DBNs), called folded-DBN, on NVIDA's Tesla K20 GPU. Results showed, that comparing execution time of the fine-tuning process, the GPU implementation results 7 to 11 times speedup over the CPU platform.

Others authors in their researches also approved that proposed models on GPU achieved the better results. Hung and Wang in [26] proposed a GPU-accelerated PSO (GPSO) algorithm that uses the NVIDIA Tesla C1060 GPU to improve the timing efficiency of PSO. Numerical results showed that the GPU architecture fits the PSO framework well by reducing computational timing, achieving high parallel efficiency and finding better optimal solutions by using a large number of particles. Cai et al. in [27] proposed approach to forecast large scale conditional volatility and covariance using neural network on GPU. Tran and Cambria in [28] developed an ensemble application of extreme learning machine (ELM) and GPU for real-time multimodal sentiment analysis that leverages on the power of sentic memes (basic inputs of sentiments that can generate most human emotions). Their proposed multimodal system is shown to achieve an accuracy of 78%. In term of processing speed, their method shows improvements of several orders of magnitude for feature extraction compared to CPU-based counterparts.

VII. CONCLUSIONS

In this article we have presented both the opportunities and challenges of the algorithmic trading and machine learning approach on GPU. The empirical study of algorithmic trading on GPU was presented, which proved the advantage of GPU versus CPU.

High frequency trading and machine learning is new and growing phenomenon. It provides interesting research opportunities in Financial management, market dynamics, FPGA hardware, multicomputing on platforms like CUDA.

Review of works in the area of machine learning based on GPU is also presented in this paper and led to the conclusion that this technique is very promising in classification, forecasting tasks and could be used in big data areas. The systems implemented on GPU is able to process a huge volume of parameters faster than CPU. The findings of this paper may be applied in the future works.

ACKNOWLEDGMENT

We would also like to show our gratitude to the NANOTICK for providing high frequency data in microseconds of 5 commodity futures contracts.

REFERENCES

[1] Margara A., Cugola G. (2011), "High performance content-based matching using GPUs", Proceedings of the 5th ACM international conference on Distributed event-based system, New York, USA

[2] Catanzaro, B., Sundaram, N., & Keutzer, K. (2008, July). Fast support vector machine training and classification on graphics processors. In Proceedings of the 25th international conference on Machine learning (pp. 104-111). ACM.

[3] MechanicalForex. (2016), mechanicalforex.com. [ONLINE] Available at: http://mechanicalforex.com/2016/02/trading-and-the-gpu-wasted-power.html. [Accessed 12 January 2018].

[4] Preis T. (2011), "GPU – computing in econophysics and statistical physics", The European Physical Journal Special Topics, Vol. 194, pp. 87 – 119.

[5] [8] NVIDIA Corporation. (2008) NVIDIA CUDA Compute Unified Device Architecture.

[6] Kaya O. (2016), "High – frequency trading. Reaching the limits", Automated trader magazine. Vol. 41, p. 23 – 27.

[7] Fox M. B., Glosten L. R., Rauterberg G. V. (2015), "The New Stock Market: Sense and Nonsense" , 65 Duke L.J. 191.

[8] Herlemont D. (2013), "Pairs Trading, Convergence Trading, Cointegration", Quantitative Finance, Vol. 12(9).

[9] Zubulake P., Lee S. (2011), "The High frequency game changer: how automated trading strategies have revolutionized the markets", Aite group. Wiley trading.

[10] Brogaard J., Hendershott J. T., Riordan R. (2013), "High frequency trading and price discovery", ECB Lamfalussy fellowship programme/ Working paper series, No 1602, European central bank Press.

[11] Jaramillo C. (2016), "The Revolt against High-Frequency Trading: From Flash Boys, to Class Actions, to IEX", Review of banking & financial law, Vol. 35, pp. 483 – 499.

[12] Kirchner S. (2015), "High frequency trading: Fact and fiction", Policy: A Journal of Public Policy and Ideas, Vol. 31(4), pp. 8-20.

[13] Limaye S. S. (2014)," Electronically aided High frequency trading", International Journal of Engineering Research and Applications, pp. 14 – 18.

[14] McKenny D., White T. (2012), "Stock Trading Strategy Creation Using GP on GPU", Soft Computing, Vol 16(2), pp. 247 – 259.

[15] Salleh N. S. M., Baharim M. F. (2015), "Performance Comparison of Parallel Execution Using GPU and CPU in SVM Training Session". In Advanced Computer Science Applications and Technologies (ACSAT), 2015 4th International Conference on, pp. 214-217.

[16] Li X., Li K., Zhang G., Zheng W. (2015), "Deep Learning and Its Parallelization: Concepts and Instances".

[17] Sopyła K., Drozda P., Górecki P. (2012), "SVM with CUDA accelerated kernels for big sparse problems". In International Conference on Artificial Intelligence and Soft Computing, pp. 439-447.

[18] Herrero-Lopez S., Williams J. R., Sanchez A. (2010), "Parallel multiclass classification using SVMs on GPUs". In Proceedings of the 3rd Workshop on general-purpose computation on graphics processing units, pp. 2-11.

[19] Carpenter A. U. S. T. I. N. (2009), "CUSVM: A CUDA implementation of support vector classification and regression". patternsonscreen. net/cuSVMDesc. pdf, pp. 1-9.

[20] Vaněk J., Michálek J., Psutka J. (2017), "A GPU-Architecture Optimized Hierarchical Decomposition Algorithm for Support Vector Machine Training". IEEE Transactions on Parallel and Distributed Systems, 28(12), pp. 3330-3343.

[21] Raina R., Madhavan A., Ng, A. Y. (2009), "Large-scale deep unsupervised learning using graphics processors". In Proceedings of the 26th annual international conference on machine learning, pp. 873-880.

[22] Dean J., Corrado G., Monga R., Chen K., Devin M., Mao M., ... , Ng A. Y. (2012), "Large scale distributed deep networks". In Advances in neural information processing systems, pp. 1223-1231.

[23] Li Q., Salman R., Kecman V. (2010), "An intelligent system for accelerating parallel SVM classification problems on large datasets using GPU". In Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on, pp. 1131-1135.

[24] Coates A., Huval B., Wang T., Wu D., Catanzaro B., Andrew N. (2013), "Deep learning with COTS HPC systems". In International Conference on Machine Learning, pp. 1337-1345.

[25] Chen Z., Wang J., He H., Huang X. (2014), "A fast deep learning system using GPU". In Circuits and Systems (ISCAS), 2014 IEEE International Symposium on, pp. 1552-1555.

[26] Hung Y., Wang W. (2012), "Accelerating parallel particle swarm optimization via GPU". Optimization Methods and Software, 27(1), pp. 33-51.

[27] Cai X., Lai G. Lin X. (2013), "Forecasting large scale conditional volatility and covariance using neural network on GPU". The Journal of Supercomputing, 63(2), pp.490-507.

[28] Tran H. N., Cambria E. (2018), "Ensemble application of ELM and GPU for real-time multimodal sentiment analysis". Memetic Computing, 10(1), pp. 3-13.

[29] Krizhevsky A., Sutskever I., Hinton G. E. (2012), "Imagenet classification with deep convolutional neural networks". In Advances in neural information processing systems (pp. 1097-1105).

[30] Bonanno, F., Capizzi, G., Sciuto, G. L., Napoli, C., Pappalardo, G., & Tramontana, E. (2014). A novel cloud-distributed toolbox for optimal energy dispatch management from renewables in igss by using wrnn predictors and gpu parallel solutions. In International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), (pp. 1077-1084.

[31] Napoli, C., Pappalardo, G., Tramontana, E., & Zappalà, G. (2014). A cloud-distributed GPU architecture for pattern identification in segmented detectors big-data surveys. The Computer Journal, 59(3), 338-352.