CPU and GPU Implementations for High Frequency Trading in Algorithmic Finance

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

***Abstract*— Today algorithmic trading and High Frequency Trading (HFT) account for a dominant part of overall trading volume in financial markets. The trade execution time has grown from daily trading to microseconds and nanoseconds.. A modern GPU allows hundreds of operations to be performed in parallel, leaving the CPU free to execute other jobs. The main objective of this research was to test the possibility and quantify how much higher speedups the use of GPUs can bring in calculations of HFT statistical arbitrage algorithms. In the research MATLAB software was applied for GPU application and computations. The statistical arbitrage- pair trading algorithm was parallelized in order to adapt it to GPU application. The effectiveness was measured according to time CPU and GPU did spent working on historical data using pair trading strategy. In the paper the final results of the research are presented and discussed. The results have proven up to 30% increase in computational speed with the application of statistical arbitrage algorithm in HFT.**

Keywords— high frequency trading; statistical arbitrage; GPU; high performance computing; parallel computing.

I. INTRODUCTION

The computational power requirements have continuously increased in computer science fields such as computational physics, quantitative finance and etc. One of the examples is high-frequency trading (HFT) which is focused on automatic trading decisions making. All decisions to buy or to sell financial instrument are made by computer algorithms without human interaction. The mentioned algorithms analyze the incoming information which is received from the exchange system. Information from exchange system may include new transactions taking place with their transaction prices and volumes, but in some systems also order submission, order modification and order deletion events of other exchange members. If a trading algorithm decides to submit a buy or sell order to the exchange system, then within a few milliseconds this information is sent from exchange member's system to the central exchange server which is responsible for matching offer and demand. The exchange server responds with a confirmation message. [6]

The trade execution time has grown from daily trading to microseconds and even nanoseconds. By 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 hardware and high – speed connections and place their trading platforms close to stock market servers via co-location. One of the hardware invested is GPU. The architectures GPU are a cost effective alternative to traditional parallel processing machines. This change ushers in a new era in computing, which allows any modern personal computer to take advantage of parallel processing capabilities previously available only in specialized systems.[20]

Nowadays, standard computers come with sequential CPUs or with multicore CPUs, which allow a limited number of processes to be executed in parallel. On the other hand, the importance of graphics in most application domains pushed industry into producing ad-hoc Graphical Processing Units (GPUs) to relieve the main CPU from the calculations required for graphics. What is important here is that this hardware is strongly parallel and may operate independent from the main CPU. A modern GPU, like those equipping most computers today, 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. [7]

The increasing pervasivity of parallel architectures like multi-/many-core CPUs and GPUs, parallel programming has become not an alternative but rather a need for increasing the software performance.[2]

Graphics processing units (GPU) offer a new possibility for speeding up large scale simulation of long range interacting systems without sacrificing accuracy. GPU is a powerful device which can process thousands of threads simultaneously with high memory bandwidth. Compared to CPU, GPU is designed with more transistors that are devoted to data processing rather than data caching and flow control. It is suitable for computation-intensive and data-parallel computations needed for high frequency traders that are time sensitive. [5]

Multi-threaded parallel CPU implementations are expected to run faster than the single-threaded counterparts, the overhead of creating, destroying, and synchronizing threads may be very

high. An alternative parallel computing platform is the GPU. Originally, it was developed for graphics applications. Due to their massive parallel processing capabilities, state-of-the-art GPUs are the leading software computing devices for the most parallel and computationally intensive applications such as high frequency trading algorithms. [3]

Our study demonstrates how the use of GPUs can bring impressive speedups in statistical arbitrage trading algorithm, leaving the main CPU free to focus on the remaining aspects of trading strategy. Several vendors have recently started offering toolkits to leverage the power of GPUs for general purpose programming. Unfortunately, they introduce a totally new model of computation, which requires algorithms to be fully re-designed. In this research MATLAB was used for GPU computing which allows to accelerate an application with GPUs more easily than by using C or Fortran. With the MATLAB language it is possible take advantage of the CUDA GPU computing technology without having to learn the intricacies of GPU architectures or low-level GPU computing libraries.

In this paper, we investigate implementations of CPU and GPU the parallel pair trading algorithm. The main aim of this research is to explain the improved designs in detail, and report a performance comparison between CPU and GPU implementations in terms of speed. Improvements suggested in the paper for CPU and GPU implementations are summarized as faster speed due to new memory access patterns, and more flexibility due to a more efficient use of processors, respectively.

In order to take advantage of the CPU and GPU it is necessary to parallelize the calculations. The effectiveness was measured according to time CPU and GPU did spent working on historical data using pair trading strategy. The strategy used was first researched by D. Herlemont on his paper about pairs trading [19]. This trading strategy was used on high frequency data during previous researches. [24][32] However it was not used with GPU. There are a number of functions of this trading algorithm that can be parallelized like pair selection, trading signal detection, trading and profit/loss calculation for each trade. Thus, it had to be modified and parallelize in order to take advantage of GPU. Importantly, not only pairs trading strategies, but also the method of pairs selection is introduced in this research.

Cointegration method was used for trading pairs selection. The pairs selection algorithm is based on using Augmented Dickey Fuller Test, Engle and Grangers 2-step approach and Johansen test. [12] Finally, the comparison of statistical arbitrage trading strategy is given when using CPU and later with GPU.

The rest of the paper is organized as follows: theory and the problem statement are presented in Sections 1 and 2, the methodology, including the pairs trading strategy, pairs selection algorithm, speedup of an trading algorithm is presented in Sections 3 and 4. The results and the summary of the research, followed by conclusions in Section 5.

II. Trading Using 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. Now, however, NVIDIA has come out with a new line of graphics cards – Tesla. [6]

One of NVIDIA GPUs‟ main features is ease of programmability made possible with CUDA – Compute Unified Device Architecture. CUDA provides the means to compile and run code for NVIDIA‟s GPUs. With a low learning curve, CUDA allows developers to tap into enormous computing power of GPUs yielding high performance benefits. [8] 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. Thus, CUDA issues and manages computations on a GPU as a data-parallel computing device. The graphics card architecture used in recent GPU generations is built around a scalable array of streaming multiprocessors. [8] When a program using CUDA extensions and running on the CPU invokes a GPU kernel, which is a synonym for a GPU function, many copies of this kernel – known as threads – are enumerated and distributed to the available multiprocessors, where their execution starts. [6]

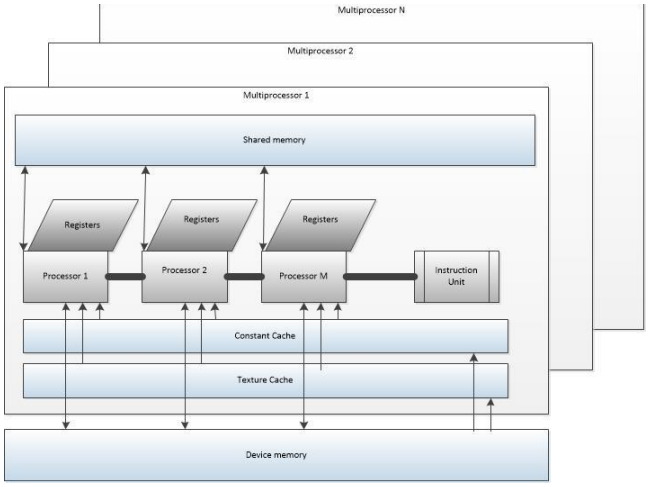

Fig. 1. Visualization of a GPU multiprocessor with on-chip shared memory.Example of a figure caption. *(figure caption)*

As shown in Fig. 3, each multiprocessor of the GPU device contains several local registers per processor, memory which is shared by all scalar processor cores in a multiprocessor. In order to allow for reducing the number of involved multiprocessors, the slower global memory can be used, which is shared among all multiprocessors and is also accessible by the function running in the CPU. Please note, that the GPU's global memory is still roughly 10 times faster than current main memory of personal computers. However, each multiprocessor features only one double-precision processing core and so, the theoretical peak performance is significantly reduced for double-precision operations. [8]

III. Statistical Arbitrage

Correlation is a statistical term that comes from linear regression analysis. This term defines the strength of a relationship between two variables. The main idea of statistical arbitrage or pairs trading is to find the pair of financial instruments that are highly correlated. When a pair is found, a trader must look for the changes in correlation followed by mean – reversion to the trend of financial instruments pair, thereby, creating a profit opportunity. This type of trading needs to identify a relationship between two financial instruments, figure out the direction of their relationship, and execute long and short positions, based on the statistical data presented. Selecting a good pair for trading becomes the most important stage of mean-reversion of the market-neutral statistical arbitrage strategy.[26][34]

A. Pairs Trading Using Cointegration

The cointegration method uses mathematical model, developed by Engle and Granger [17], which have attracted a considerable interest of the economists over the last two decades. Cointegration states that, in some instances, despite two given non-stationary time series, a specific linear combination of the two time series is actually stationary. The two time series move together in a lockstep fashion. The cointegration can be described like this: x_t and y_t are two time series that were non-stationary. If there was parameter γ and the following equation:

$$z_t = y_t - \gamma x_t \quad (1)$$

was a stationary process, then x_t and y_t would be cointegrated. This path-breaking process emerged as a powerful tool for investigating common asset trends in multivariate time series. [25]

B. Data

The microsecond data for this research was provided by Nanotick company. Futures contract data is 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.

After normalization, microsecond futures commodity contracts data consisted of 24957994 records. Upon preparation, the data had to be applied to statistical arbitrage trading strategy.

TABLE I. Microsecond data example for NGF6 contract

Receiving Date	**Receiving Time**	**Symbol**	**Asset**	**Entry Type**	**Entry Price**
20150809	17:00:00.869053009	NGF6	NG	A	3227
20150809	17:00:00.869053009	NGF6	NG	B	3221
20150809	17:00:00.930168164	NGF6	NG	A	3226
20150809	17:00:00.930168164	NGF6	NG	B	3221
20150809	17:00:01.017456320	NGF6	NG	A	3226
20150809	17:00:01.017456320	NGF6	NG	B	3219
20150809	17:00:01.059840559	NGF6	NG	A	3227
20150809	17:00:01.059840559	NGF6	NG	B	3219
20150809	17:00:01.156791713	NGF6	NG	A	3238
20150809	17:00:01.156791713	NGF6	NG	B	3216
20150809	17:00:01.204683812	NGF6	NG	A	3238
20150809	17:00:01.204683812	NGF6	NG	B	3216
20150809	17:00:01.205605232	NGF6	NG	A	3238
20150809	17:00:01.205605232	NGF6	NG	B	3215
20150809	17:00:01.206755867	NGF6	NG	A	3238
20150809	17:00:01.206755867	NGF6	NG	B	3215
20150809	17:00:01.207350519	NGF6	NG	A	3231
20150809	17:00:01.207350519	NGF6	NG	B	3215
20150809	17:00:01.208805474	NGF6	NG	A	3231
20150809	17:00:01.208805474	NGF6	NG	B	3217
20150809	17:00:01.224604710	NGF6	NG	A	3233
20150809	17:00:01.224604710	NGF6	NG	B	3217

The cointegration method uses mathematical model, developed

IV. 3. Methodology

The main purpose of pairs trading is to find two financial instruments that move together. Once the pair of these instruments is found, strategy has to decide when to take long and short positions based on the trading rules. Following the research, six main steps of pairs trading strategy were identified:

1. Selection of the size of the window trading and data normalization;
2. Data normalization;
3. Selection of the correlated pair;
4. Definition of the trading rules;
5. Trading;
6. Assessment of the pairs trading strategy.[16][24][32]

Before selecting trading and data normalization window, strategy has to be trained. Thus, before starting to trade, some data must be used for training. This data may be called out of sample data. All data of microsecond futures commodity contracts had to be divided into training and testing datasets. The method of dividing data into training and testing periods was referred to as the holdout method in statistical classification. [26] When selecting training or out of sample period, it is important to select the right size of this window: if too big window is chosen, strategy may overtrain and it cannot be too small as the strategy will not be able to notice the abnormal behaviour. [30]

Finally, the testing period follows immediately after the training period.

A. Data Normalization

Upon receiving the microsecond data for commodity futures contracts, next step was to normalize these data to be able to implement them in our test environment. First task was to bring time stamp data together. For example, if we have a time stamp of 17:00:00.869053009 in one contract and the time stamp of 17:00:00.825207610 in other futures contract, these two time stamps have to appear in both contracts. In our case, all different time stamps had to appear in all five different futures contacts.

If the contract is filled with a new time stamp, the price for that futures contract is set the same as the last time stamp. It is assumed that the price did not change for that time. In this way, all time stamps of futures contracts are normalized for nanosecond and microsecond data. [24][32]

As all time stamps for all the futures contracts were obtained, it was time to define data out of sample, normalization and trading periods. During this procedure, all parameter were kept the same: out of sample period was 5 minutes, normalization and trading period was kept the same, i.e., 20 seconds for each trading window. One more period was selected, which is for closing the positions, which was 20 seconds as well.

Upon setting and defining the above parameters on the trading strategy, price normalization follows. When normalizing for each price of futures commodity contract *P(i,t)*, we calculate empirical mean *μ(i,t)* and standard deviation *σ(i,t)* for the selected normalization period, and then apply the following equation [30]:

$$p(i,t) = \frac{P(i,t) - \mu(i,t)}{\sigma(i,t)} \quad (2)$$

Value *p(i,t)* is the normalized price of futures commodity contract *i* at time *t*. [30]

B. Pair Selection

One of two main parts of this trading methodology is the pairs selection algorithm which is essentially based on cointegration testing. Cointegration method involves the following steps:

1. Identify futures contract pairs that could potentially be cointegrated;

2. Once the potential pairs are identified, we need to verify the proposed hypothesis that the futures contract pairs are indeed cointegrated based on the information from historical data;

3. Examine the cointegrated pairs to determine whether they can be trade on. [33]

The objective of this phase is to identify the pairs with linear combination exhibiting a significant predictable component that is uncorrelated with underlying movements in the market as a whole. With this aim, we first measure the spread of pair prices for stationarity. In this research, it is done by checking whether the data series are integrated in the same order by using Augmented Dickey Fuller Test (ADF), which is the extended version Dickey Fuller. [12] Having passed the ADF test, cointegration tests are performed on all possible combinations of pairs. To test for cointegration we adopted Engle and Granger 2-step approach and Johansen test. This methodology is based on Caldeira and Moura. [12]

Johansen test determines the number of cointegrating relations and also implements a multivariate extension of the 2-step Engle and Granger procedure. [12]

All of the procedures are implemented on MATLAB. The second part of the algorithm creates trading signals for the detected cointegrating relations based on the predefined investment decision rules.

V. Experimental Setup

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.

During our research CPU used was Intel i5 - 3230M 2,6 GHz with two cores (2 MATLAB worker) and GPU GeForce 710M with 96 CUDA cores. Firstly we did apply the pair trading strategy only two CPU. Using “parfor” function of MATLAB which allows hundreds of operations to be performed in parallel with CPU we did detect calculations that were possible to parallelize. During this stage we did speed up the strategy to maximize its performance by using only CPU.

When it came to GPU we did use gpuArray and arrayfun GPU functions together with parfor, which works on CPU. GpuArray creates array on GPU and arrayfun applys function to each element of array. This method of using gpuArray with arrayfun makes actual evaluation of the function happens on the GPU, not on the CPU. Thus, any required data not already on the GPU is moved to GPU memory, the MATLAB function passed in for evaluation is compiled for the GPU, and then executed on the GPU. All the output arguments return as gpuArray objects. [10][11]

In our experiment we did parallelize pair detection, detecting buy/sell signals, the trading and profit calculation. It was possible to parallelize these functions because every iteration the strategy has it must perform same calculations. In order not to wait for one function to stop we can perform multiple calculations with multiple functions.

VI. Experimental Results

The overall pair trading strategy performance was measured in the profit it did generate. During the experiment we did not use transactions cost, which was kept zero, and the amount invested in each trade was kept the same, which was 10. The profit/loss was measured in percentage in change of overall difference at the end of each trading day. A more detailed information is presented in figure below.

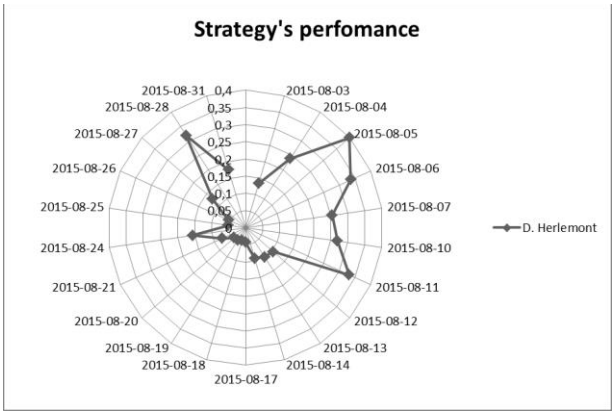

Fig. 2. Strategy performance for each day by the profit it did generate

Figure 2 above shows the daily profits from HFT trading algorithm and confirms the results revealed by High Frequency Trading market leader Virtu Financial, Inc, where only one losing trading day out of 1237 days was generated [14]. The chart in Figure 1 illustrates daily results of an algorithm-based on a statistical arbitrage HFT system. The less profitable days occur because of fewer trades, due to less trade signals, rather than fluctuations or a series of unproductive trades. However, our 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 II. 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	2991,80	2081,60	6096505
2015-08-04	2208,10	1400,50	4579465
2015-08-05	2393,70	1783,10	5793525
2015-08-06	3040,90	2585,3	5595770
2015-08-07	2650,10	2027,1	5586360
2015-08-10	4410,80	3080,70	5732355
2015-08-11	4980,30	3154,50	6249980
2015-08-12	2769,20	2151,20	6758875
2015-08-13	4122,60	3419,00	5666900
2015-08-14	1325,90	1055,80	4227335
2015-08-17	1550,00	1171,10	4879990
2015-08-18	1912,10	1299,50	4364540
2015-08-19	4002,30	3278,70	5666700
2015-08-20	4449,00	3119,43	5411145
2015-08-21	4311,70	3389,10	5946205
2015-08-24	4809,40	4064,00	7710745
2015-08-25	3960,20	3466,10	5105175
2015-08-26	3187,60	2600,40	5119660
2015-08-27	5004,90	4244,20	7963320
2015-08-28	5287,10	4413,10	7721975
2015-08-31	5409,70	4594,10	8613445

From table 2 it is shown how much time in seconds did algorithm spend on each day trading simulation using different hardware CPU (Intel i5 - 3230M 2,6 GHz,2 cores) and GPU (GeForce 710m, 96 CUDA Cores) and how many records it had to process.

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

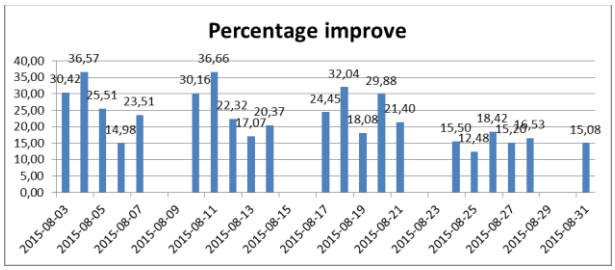

Fig. 3. The improvement of the algorithm when using GPU

As shown in figure above when pair trading algorithm was presented to GPU, the speed of simulation did improve dramatically varying from 12% to 36% improve in overall speed. 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. It is shown that CPU, even with multi-threaded implementation, is not a feasible option for large dense matrices. For the GPU implementation, performance impact of the global memory access patterns on the GPU board and the memory coalescing are emphasized. In our case the bigger the matrix of trades and pairs the more measurable is 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. In our research the possibility to improve the speed of daily trading with microseconds came, when algorithms calculations were parallelized and presented to GPU using gpuArrays and arrayfun in MATLAB, that allows to exploit the GPU at hand.

VII. CONCLUSIONS

Recent technological advances have made trading in the markets fast and mostly done by computers and algorithms. Instead of humans, computers replicate the role of market makers, specialists or liquidity providers but at a much higher rate of speed. The number of derived financial instruments has caused increased opportunities for profits arising from pricing inefficiencies or price move delays between securities. Trading algorithms now work not only with CPU, but with GPU. These factors have been driving forces to test the system based on pair trading in HFT and see how the effectiveness differ when using different hardware. In this paper, high frequency algorithmic pairs trading was developed on the market - neutral statistical arbitrage strategy presented by D. Herlemont. Importantly, all

five futures commodity contracts, used for the proposed pairs trading strategy, belong to same CME group, which is the world's largest options and futures exchange platform. proposed trading strategy used the pairs selection algorithm which consisted of the Augmented Dickey Fuller test. If futures commodity contracts prices pass the Augmented Dickey Fuller test, cointegration tests are performed on all possible combination of pairs. To test for cointegration Engle and Grangers 2-step approach and Johansen test was adopted. Trading strategy was firstly presented to CPU (Intel i5 - 3230M 2,6 GHz1 2 cores) and later to GPU (GeForce 710m, 96 CUDS cores). All trading parameters were kept the same during research. The purpose of this was to measure the effectiveness of hardware and to check how much higher frequency trading evolution and performance improves when it is presented to GPU rather than to only CPU. At the end of the research, when all datasets were implemented to the pairs selection algorithm working with CPU and GPU, the results were gathered. It should be no surprise that when algorithm was presented to GPU it did perform more effective. The speed up of daily improvement of speed did vary from 12% to 36%. 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. The increase could be even more dramatic if algorithm would be presented to even more financial instruments and more trading signals would be created.

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] Ahmed M., Chai A., Ding X., Jiang Y., Sun Y. (2009), "Statistical Arbitrage in High Frequency Trading Based on Limit Order Book Dynamics".

[2] Danelutto M., De Matteis T., Mencagli G., Torquati M. (2015), "Parallelizing High-Frequency Trading Applications by Using C++11 Attributes", *August 2015, IEEE*.

[3] Mustafa U. Torun, Onur Yılmaz, Ali N. Akansu. (2016), "FPGA, GPU, and CPU implementations of Jacobi algorithm for eigenanalysis", *Journal of Parallel and Distributed Computing*, Vol. 96, pp 172-180.

[4] Kozikowski G., Papamanousakis G., Yang J. (2015), "Potential future exposure, modelling and accelerating on GPU and FPGA", *WHPCF 2015 Proceedings of the 8th Workshop on High Performance Computational Finance*, Article No. 4.

[5] Liang Y., Xing, X., Li Y.(2017), "A GPU-based large-scale Monte Carlo simulation method for systems with long-range interactions", *Journal of Computational Physics*, Vol/ 338, pp. 252-268 .

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

[7] 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

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

[9] Napoli C. et al., A cloud-distributed GPU architecture for pattern identification in segmented detectors big-data surveys. The Computer Journal, vol. 59, issue 3 , pp.338-352.

[10] Matlab. (2016), *se.mathworks.com*. [ONLINE] Available at: https://se.mathworks.com/help/distcomp/gpu-computing.html.

[11] Matlab. (2015), *se.mathworks.com*. [ONLINE] Available at: https://se.mathworks.com/discovery/matlab-gpu.html.

[12] Caldeira J. F., Moura G. V. (2013), "Selection of a portfolio of pairs based on cointegration: A statistical arbitrage strategy", *Revista Brasileira de Financas*, Vol. 11(1), pp. 49–80.

[13] Bogoev D., Karam A. (2016), "An Empirical detection of High Frequency Trading Strategies", 6th International Conference of the Financial Engineering and Banking Society. June 10-12, 2016 Melaga.

[14] Cifu D. A. (2014), "FORM S-1, Registration Statement Under The Securities Act Of 1933", Virtu Financial, Inc.

[15] Dickey D., Fuller W. (1979), "Distribution of the Estimator for Autoregressive Time series with a Unit Root", *Journal of the American Statistical Association*, Vol. 74, pp. 427-431.

[16] Driaunys K., Masteika S., Sakalauksas V., Vaitonis M. (2014), "An algorithm-based statistical arbitrage high frequency trading system to forecast prices of natural gas futures", *Transformations in business and economics*. Vol. 13(3), p. 96–109.

[17] Engle, R. F., Granger, C. W. J. (1987), "Co-integration and error correction: Representation, estimation, and testing", *Econometrica*, Vol. 55(2), pp. 251–276.

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

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

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

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

[22] Lau C. A., Xie W., Wu Y. (2016), "Multi – Dimensional Pairs Trading Using Copulas", European Financial Management Association 2016 Annual Meetings June 29-July 2, 2016 Basel, Switzerland.

[23] Madhavaram G. R. (2013), "Statistical Arbitrage Using Pairs Trading With Support Vector Machine Learning", Saint Mary's University.

[24] Masteika S., Vaitonis M. (2015), "Quantitative Research in High Frequency Trading for Natural Gas Futures Market", *Business Information Systems Workshops*, Vol. 228, pp. 29-35.

[25] Miao G. J. (2014), "High Frequency and Dynamic Pairs Trading Based on Statistical Arbitrage Using a Two-Stage Correlation and Cointegration Approach", *International Journal of Economics and Finance*, Vol. 6(3), pp. 96 – 110.

[26] Miao G. J., Clements M. A. (2002), "Digital Signal Processing and Statistical Classification", Artech House, ISBN 1580531350.

[27] Miller R. S., Shorter G. (2016), "High Frequency Trading: Overview of Recent Developments", report, April 4, 2016; Washington D.C

[28] Mushtaq R. (2011), "Augmented Dickey Fuller Test". Available at SSRN: https://ssrn.com/abstract=1911068.

[29] Ohara M. (2015), "High frequency market microstructure", *Journal of Financial Economics*, Vol. 116(2), pp. 257–270.

[30] Perlin M. S. (2009), "Evaluation of Pairs-trading strategy at the Brazilian financial market", *Journal of Derivatives & Hedge Funds*, Vol. 15(2), pp. 122–136.

[31] Vaitonis M. (2017), "Pairs Trading Using HFT in OMX Baltic Market", *Baltic J. Modern Computing*, Vol. 5(1), pp. 37-49.

[32] Vaitonis M., Masteika S. (2016), "Research in High Frequency Trading and Pairs Selection Algorithm with Baltic Region Stocks", In: Dregvaite G., Damasevicius R. Information and Software Technologies. ICIST 2016. *Communications in Computer and Information Science*, Vol 639. Springer.

[33] Vidyamurthy G. (2004), "Pairs Trading – Quantitative Methods and Analysis, New Jersey", John Wiley & Sons, Inc., p.210.

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