VILNIUS UNIVERSITETY

Mantas
VAITONIS

High Frequency Computerized Trading Strategies Engineering in Financial Markets

SUMMARY OF DOCTORAL DISSERTATION

Technological Sciences,
Informatics Engineering T 007

VILNIUS 2020

The dissertation work was carried out at Vilnius University from 2015 to 2019.

Scientific Supervisor
Assoc. Prof. Saulius Masteika (Vilnius University, Technological Sciences, Informatics Engineering – T 007).

This doctoral dissertation will be defended in a public meeting of the Dissertation Defence Panel:

Chairman
Prof. Dr. Julius Žilinskas (Vilnius University, Technological Sciences, Informatics Engineering – T 007).

Members:
Prof. Dr. Robertas Damaševičius (Kaunas University of Technology, Technological Sciences, Informatics Engineering – T 007),
Prof. Dr. Raimundas Matulevičius (University of Tartu, Natural Sciences, Informatics – N 009),
Prof. Habil. Dr. Leonidas Sakalauskas (Vilnius University, Technological Sciences, Informatics Engineering – T 007),
Prof. Dr. Dmitrij Šešok (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – T 007).

The dissertation shall be defended at a public meeting of the Dissertation Defence Panel at 12:00 p. m. on 25th of September, 2020 in Room 203 of the Institute of Data Science and Digital Technologies of Vilnius University.
Address: Akademijos str. 4, LT-04812 Vilnius, Lithuania.
The summary of the doctoral dissertation was distributed on the 24nd of August 2020.
The text of this dissertation can be accessed at the library of Vilnius University, as well as on the website of Vilnius University: *www.vu.lt/lt/naujienos/ivykiu-kalendorius*

VILNIAUS UNIVERSITETAS

Mantas
VAITONIS

Didelio dažnio kompiuterizuotų prekybos strategijų inžinerija finansinėse rinkose

DAKTARO DISERTACIJOS SANTRAUKA

Technologijos mokslai,
Informatikos inžinerija T 007

VILNIUS 2020

Disertacija rengta 2015– 2019 metais Vilniaus universitete.

Mokslinis vadovas
doc. dr. Saulius Masteika (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – T 007).

Gynimo taryba:

Pirmininkas
prof. dr. Julius Žilinskas (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – T 007).

Nariai:
prof. dr. Robertas Damaševičius (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija – T 007),
prof. dr. Raimundas Matulevičius (Tartu universitetas, gamtos mokslai, informatika – N 009),
prof. habil. dr. Leonidas Sakalauskas (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – T 007),
prof. dr. Dmitrij Šešok (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – T 007).

Disertacija ginama viešame Gynimo tarybos posėdyje 2020 m. rugsėjo mėn. 25 d. 12:00 val. Vilniaus universiteto Duomenų mokslo ir skaitmeninių technologijų instituto 203 auditorijoje.
Adresas: Akademijos g. 4, LT-04812 Vilnius, Lietuva.
Disertacijos santrauka išsiuntinėta 2020 m. rugpjūčio mėn. 24 d.
Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje ir VU interneto svetainėje adresu: *https://www.vu.lt/naujienos/ivykiu-kalendorius*

1. INTRODUCTION

The computational power requirements have continuously increased in computer science fields such as computational physics, quantitative finance, and others. With the increasing requirements of parallel architectures like multi-/many-core CPUs and GPUs, parallel programming has become not an alternative but rather a need. It is becoming one of the major topics in software and parallelism in programs that allows multiple processes to be executed concurrently using separate threads and processing units.

The aforementioned developments in computer technology have changed the way financial instruments are traded. The technology has become a competitive differentiator of the improvement of IT infrastructure that helps to reduce the time for the execution of market and performance trading strategies. This has motivated hardware manufacturers and software developers to create solutions for the financial computing area. Nowadays, a significant part of trades is handled without human intervention, when 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 years and now is called High-Frequency Trading (HFT).

The rapid development of computational power has allowed the number of transactions executed daily in electronic markets to be increased, generating a huge amount of intraday financial data (Dacorogna M. M., et al., 2001). Thus, the interest in HFT algorithms (Aldridge I., 2009; Durbin M., 2010; Zubulake P. and Lee S., 2011) has risen and they have become one of the main tools for quantitative analysts. HFT is an algorithmic way to generate buy-sell orders using quantitative models (Mariano R. S. and Kuen Tse Y., 2008; Lai T. L. and Xing H., 2008; Kantz H. and Schreiber T., 2004). 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 major importance (Kaya O., 2016). In order to make decisions very fast and to decide in a split of a second whether to execute a transaction or not, an electronic trading system must be able to estimate market behaviour.

MiFID II requires trading operators to inspect all algorithmic and HFT systems operating in their territory. They must be tested before they can be operated. Competent authorities also must have methods for verifying the algorithms and huge amounts of data in these systems, which requires developing a unified method for efficient and fast verification of algorithmic trading (AT) and HFT (Bush D., 2016). The research found that it is not generally defined which method of parallelization of large-scale data calculations (code vectorization, multidimensional matrix, or kernel parallelization) is more efficient in HFT and could be used in MiFID II as a tool to verify AT and HFT.

After conducting the literature review, the main problem was identified – HFT strategies and algorithms are not formalized, the effectiveness of their application has not been studied. There is a need for a computerized approach using the knowledge of computer engineering to enable market participants and financial market supervisors to perform the testing of HFT algorithms. In the absence of a general view on HFT, there is no method for testing this type of trading, assessing its impact on markets, and applying it to stock exchanges. The goal of the work is not to improve the methods of parallelization of on-site calculations aimed at increasing their efficiency but to perform applied experimental research in order to develop an engineering solution that would process AT signals at the required speed in the HFT environment.

A method for testing high-frequency statistical arbitrage trading has been proposed and a prototype algorithm has been developed that formalizes the operation and backtesting of this type of strategy. This approach allows the testing of algorithmic and HFT statistical arbitrage trading strategies under the MiFID II Directive.

The suggested method applies code vectorization with multidimensional matrices and kernel parallelization. The combination of these methods has not been tested in HFT. The application of all three above-mentioned methods helps to increase the speed of trading decision making during the experiment, and the trade itself remains profitable. The use of these methods allows to parallelize data normalization, trade pair selection, position opening/closing, deletion of unnecessary trade pairs, and closing of long-held transactions that are performed simultaneously using the SIMD method for the entire data vector.

1.1. Object of the research

The object of this study is algorithmic statistical arbitrage trading systems of high frequency and high-frequency data in electronic financial exchange.

1.2. Research aim

The work aims to develop high-frequency statistical arbitrage trading testing method that makes algorithmic trading decisions faster than the new data from electronic exchanges is received.

The following tasks were set to achieve the aim:

1. To formalize high-frequency algorithmic strategies by setting technological specifications for high-frequency data processing, to test the effectiveness of statistical arbitrage HFT strategies, and to apply the acquired knowledge to achieve the aim of this work.

2. To propose and develop a method for a real-time HFT testing and data processing that would work with high-frequency data and perform algorithmic trading calculations faster than the new data from electronic exchanges is received.

3. To test the developed method by creating a prototype of a HFT testing tool that allows calculations with high-frequency and volume data.

4. To describe the obtained results and observations and to evaluate the possibilities and possible limitations of the developed high-frequency statistical arbitrage trading algorithm testing method.

1.3. Research methods

The theoretical part of the work was based on scientific literature, scientific articles, and Internet sources. The following methods were used in the work: analysis and generalization of scientific literature, observation, numerical modelling, sampling method, synthesis, experiments, correlation analysis, statistical analysis, computer data processing. The systematic review of work and analytical research aims to analyse the existing HFT systems and technologies, to determine their application methods, and to theoretically substantiate the need for the development of high-frequency statistical arbitrage trading algorithm and high-frequency data processing method. The theoretical part of the research explains the need for the developed model and how it can pursue the goals of the thesis as well as describes the chosen technology and architecture of the platform for HFT. The theoretical prototype of high-frequency statistical arbitrage trading algorithm and high-frequency data processing method, their theoretical and practical applicability are described in this work. The research experiment, the data used in it, and the criteria are presented according to which the experiment will be evaluated. After completing the theoretical part, an experiment is performed that ends with the evaluation of the prototype, the chosen technology and architecture of the platform for high frequency trading, and the entire model of high-frequency statistical arbitrage trading system.

1.4. Scientific novelty

Although HFT is often discussed in business publications, it is a relatively underdeveloped branch of science from the perspective

of informatics engineering. There is relatively little information on the application of HFT systems (limitations and risks) and the optimal technological requirements for such a solution. The vast majority of scientific articles dealing with HFT investigate methodologies for selecting optimal strategies, measuring the profitability of HFT strategies, analysing the optimal parameters for this type of trading strategies, when to submit trading transaction information to electronic exchanges, how to measure HFT activity and other that is not related to the optimal configuration, architecture, application method of HFT systems and appropriate technology platform selection (Vaitonis M., 2018, Kearns M. et al., 2010, Anane M. ir Abergel F., 2014, Beckhardt B. et al., 2016). Therefore, further investigations and a more detailed study of HFT systems, their application, architecture, and possible engineering solutions need to be undertaken to implement HFT with high-frequency and volume data.

In this thesis, we aimed at creating and implementing the backtesting method for automated high-frequency statistical arbitrage trading that would be able to analyse a large amount of tick-by-tick data received from the electronic market in a millisecond and nanosecond time-stamp precision. However, as it was discovered in this research, there is no HFT method developed that would be suitable for statistical arbitrage. The majority of the research on algorithmic trading is limited to low frequency (daily, weekly, or even monthly) trading. Although the ideas of low-frequency trading could be applied to HFT, the methods of different frequency implementation differ. Moreover, no system architecture implementations and methods allowing to implement the HFT statistical arbitrage algorithm even for backtesting could be found. Thus, it was the initial motivation to develop a method for automated HFT statistical arbitrage that would work with a large amount of high-frequency data and suggest the hardware for this type of system.

The proposed backtesting method demonstrates how the use of the GPU memory, code vectorization, parallel kernels, and multidimensional matrices brings impressive speedups in the HFT trading and analyzation of the HFD. Such HFT system optimization allows making trading decisions faster than new information is received from an electronic exchange. The speedup occurs due to more compute effective processors of the GPU combined with the memory it has.

The proposed hypothesis was confirmed that the use of code vectorization, when the statistical arbitrage HFT algorithm is transferred to the GPU, the algorithm data is formed as a multidimensional matrix. and the algorithm calculations are performed in parallel by splitting them between individual parallel kernels, enables to make HFT decisions faster than the new data from electronic exchanges is received. A high-frequency statistical arbitrage trade backtesting method has been developed and with the help of this method the trading algorithm has been transferred to the GPU environment by combining code vectorization, multidimensional matrices, and kernel. This method can be used in MiFID II to test these strategies.

1.5. Practical value of the research results

The thesis proposes the technology and the way in which a sophisticated method has been developed that enables high-frequency and high-volume data to be processed in high-frequency statistical arbitrage trading strategies faster than it is obtained from electronic exchanges, thus bypassing other market players. The essence of the solution is not only fast data processing but also a quick and correct trading decision. Due to its commercialization, all HFT system solutions are confidential, thus they require a scientific description of their operation and the explanation of how the methods employed in these systems can be used in other scientific fields. The method proposed in this thesis can be applied to any

platform that works with the GPU CUDA with slight adjustments depending on the platform specifications. The method proposed in this thesis not only can be applied to the real trade, but can be also used as a method for testing HF statistical arbitrage strategies. This approach is recommended by the MiFID II Directive that requires trading operators to inspect all the algorithmic and HFT systems operating in their territory.

Drawing on the previous research, a system can be developed that performs high-frequency calculations with the help of GPU CUDA. These calculations can be applied for making profit in HFT or to increase liquidity in electronic exchanges. Likewise, the same methods can be applied in other fields of science that require quick and correct decision-making, such as artificial intelligence.

1.6. Statements to be defended

The following statements are made in the thesis:

1. The methodology for formalizing high-frequency statistical arbitrage trading strategies, which can be used to determine the optimal window for data normalization and trading in algorithmic trading at milli and nanoseconds, is appropriate for evaluating the effectiveness of strategies.
2. Decisions made by HFT using higher-frequency data are more efficient in algorithmic trading that requires the parallelization of calculations.
3. By combining code vectorization, multidimensional matrices, kernel parallelization and transfer of formalized algorithms to the GPU environment, one can achieve a higher data processing speed than the speed required for the new data from electronic exchanges to be received.

1.7. Approbation of the research

The results of the research were presented at eight scientific conferences:

1. On 24th – 26th June 2015, at the international scientific conference BIS: Business Information Systems Workshops Poznan, Poland, presenting the paper "Quantitative Research in High Frequency Trading for Natural Gas Futures Market".
2. On 3rd–5th December 2015 at the 7th scientific conference DAMSS 2015: Duomenų analizės metodai programų sistemoms, Druskininkai, Lithuania, presenting the paper "High frequency statistical arbitrage strategy engineering and algorithm for pairs trading selection"
3. On 13th–15th October 2016 at the 22nd scientific conference ICIST 2016, Druskininkai, Lithuania, presenting the paper "Research in high frequency trading and pairs selection algorithm with Baltic region stocks".
4. On 1st–3rd December 2016 at the 8th scientific conference DAMSS 2016: Duomenų analizės metodai programų sistemoms, Druskininkai, Lithuania, presenting the paper "Computerized high frequency trading of nanoseconds in futures market".
5. On 12th–14th October 2017 at the 23rd scientific conference ICIST 2017, Druskininkai, Lithuania, presenting the paper "Statistical Arbitrage Trading Strategy in Commodity Futures Market with the Use of Nanoseconds Historical Data".
6. On 30th November – 2nd December 2017 at the 9th international conference DAMSS: Data analysis methods for software systems, Druskininkai, Lithuania, presenting the paper "Research in High Frequency Statistical Arbitrage Strategies Applied to Microsecond and Nanosecond Information".
7. On 27th April 2018 at the 23rd scientific conference IVUS 2018: Information Society and University Studies, Kaunas,

Lithuania, presenting the paper “CPU and GPU Implementations for High Frequency Trading in Algorithmic Finance”.

8. On 29th May 2018 at the 23rd scientific conference SYSTEM 2018: Information Society and University Studies, Gliwice, Poland, presenting the paper “Algorithmic trading and machine learning based on GPU”.

1.8. Outline of the thesis

The thesis consists of six chapters (Introduction, High Frequency Trading and High Performance Computing in Finance, Proposed HFT testing methodology, Experimental setup, Experimental results, and Conclusions), the list of references, and appendices.

In the introduction, the motivation of the work, research objectives, scientific novelty, etc. are presented. Chapter One discusses algorithmic, high-frequency and statistical arbitrage trading; it is explained how they work and when they are used. Chapter Two examines how HFT is applied in high performance computing and what technologies are used to achieve such HFT. Following the selection of GPU CUDA technology for this thesis, Chapter Three introduces statistical arbitrage strategies to be used throughout the research. Chapter Four introduces a high-frequency statistical arbitrage trading method that utilizes GPU CUDA, code vectorization, parallel kernels, and the use of multidimensional matrices. Further chapters provide with the requirements for this research, explain what GPU boards and what high-frequency data are used. The last chapter presents the research findings that are summarized in the conclusions.

2. HIGH FREQUENCY TRADING AND HIGH PERFORMANCE COMPUTING IN FINANCE

Nowadays markets are made electronic with bid and ask queues maintained within a computer by using trading algorithms that employ fundamental and technical analysis for making trading decisions.

HFT has grown rapidly since the moment it was first introduced. In Europe, the share of HFT in total equity boomed from almost zero in 2005 to 40 % in 2010. By 2005, HFT accounted for approximately 20% of the trades in the USA and peaked to 60% in 2009. Then financial crises took place and, by 2014, the share of high-frequency equity markets fell to 35% and 50% of the total market in Europe and the USA (Kaya O., 2016). At the moment the HFT accounts for approximately 55% of the trading volume in the USA equity markets and about 40% in European equity markets (Krauss C., 2015). CFTC found that during the period from October 2012 to October 2014, the algorithmic trading systems were present on at least one side in nearly 80% of the foreign exchange futures trading volume; 67% of the interest rate futures volume; 62% of the equity futures volume; 47% of the metals and energy futures volume; and 38% of the agricultural product futures volume. The algorithmic trading has also grown up to 67% of the trading in 10-year Treasury Futures and 64% of the Eurodollar Futures Markets (Miller R. S. and Shorter G., 2016).

This type of trading is highly commercialized, and the subject of employed strategies as well as optimal configuration for this type of trading for each HFT company is confidential. The information about the optimal HFT implementation and methods to achieve speedups in decision-making would not only help to further develop such trading strategies but could be also applied in other fields of science that require working with big data and high-speed decision-making. Most papers provide with the methodologies for the optimal

trading strategy selection; the way its profitability should be measured; the optimal trading strategies parameters; the best time to send orders; measurement of the HFT activity, etc. However, the information on the optimal way for the implementation of the trading strategies on the platform of choice is still lacking (Kearns M., 2010; Anane M. and Abergel F., 2014; Beckhardt B., et al., 2016; Boehmer E., et al., 2016). In fact, there is little information on the optimal configuration of the high-frequency strategies, i.e., how they should be applied and what method should be used depending on the trading algorithm.

In order to better understand how the HFT is operating, it is necessary to prepare a use case scenario for a high-frequency statistical arbitrage trading system that would make the bases for its exploration. The next section describes the implemented use case scenario in more detail.

2.1. Statistical arbitrage use case scenario in the HFT

The main aim of high-frequency statistical arbitrage trading is to find two financial instruments that work together. Once such a pair is found, it has to be decided when to take long and short positions based on the trading rules. With respect to the research on algorithm formulation, following steps of statistical arbitrage trading strategy have been identified:

- the selection of the size for data normalization and trading window;
- data normalization;
- the selection of the correlated pair;
- the definition of the trading rules;
- the act of trading;
- the assessment of the statistical arbitrage strategy (Vaitonis M. and Masteika S., 2017; Vaitonis M. and Masteika S., 2018).

Firstly, it is necessary to define the trading parameters of data normalization and trading window size. Drawing on the previous research that uses the nanosecond data (Vaitonis M. and Masteika S., 2018), it was determined that 20-second window for data normalization and trading was optimal. However, as it was indicated in the section above, there are difficulties with the high-frequency data since different future contracts send, their data at a different frequency. Therefore, the data was aggregated in equal time slots and averaged out. After these parameters were selected, it was necessary to normalize the future contracts data in order to compare one with the other. Secondly, the pair selection method had to be selected and then the trading rules had to be applied on the found pairs. Finally, the trading signals had to be sent to an electronic exchange. At the end of trading day, it was vital to assess the trading strategies.

2.1.1. Data normalization in HFT

Obviously, the normalization of the high-frequency data is significant for recalculating the prices of the correlated future contracts to a particular unit, thus removing the noise of the price alteration and offering the comparison of the changes in a more qualitative way. Data normalization is performed as follows: for each price of futures contract $p_{i,t}$, empirical mean $\mu_{i,t}$ and standard deviation $\sigma_{i,t}$ are calculated for the selected normalization period, and then the following equation is applied (Vaitonis M. and Masteika S., 2018; Perlin M. S., 2009):

$$P_{i,t} = \frac{p_{i,t} - \mu_{i,t}}{\sigma_{i,t}} \quad (1)$$

P*(i,t)* is the normalized price of futures commodity contract *i* at time *t*.

2.1.2. Pair selection

Before the trading rules are applied, the pairs for future contracts must be found. In order to do it, the pair selection algorithm is used. There are two main pair selection methods: the least squared distance and cointegration. In the research conducted earlier, both methods were used with the microsecond as well as nanosecond data and five commodity futures contracts. It has been confirmed that the statistical arbitrage strategy can be applied in the case of high frequency by applying the methods of least square distance and cointegration pair selection (Driaunys K., et al., 2014; Vaitonis M., 2017; Vaitonis M., and Masteika S., 2016; Vaitonis M., and Masteika S., 2018).

- Least square distance is the mentioned distance method requiring only some calculus and linear algebra to determine which two data elements in the given case correlate with which two futures contracts (Binh D., 2006; Miller S. J. 2006). According to Evan Gatev (Gatev E., et al., 2006), the selection of a pair of assets that minimize the sum of the squared deviation of normalized prices is a simple strategy with a low cost of implementation and, for this reason, it may be the favorite one among the practitioners. The distance method assumes that there is a static linear relationship between the two assets and prices that are equally independent random variables. One of the advantages of this non-parametric model is the absence of mis-specification and mis-estimation but it does not have forecasting power (Binh D., 2006).
- Cointegration allows for the estimation of a long-term relation between the two variables when they have the same integration level; hence two non-stationary time series are cointegrated if their linear combination is stationary (Engle, R. F. and Granger C. W. J., 1987). The cointegration approach fits perfectly with the statistical arbitrage attempting to exploit a short-term deviation from a long-term relation. Short-term deviations are rectified by error corrections that, according to Ganapathy

Vidyamurthy (Vidyamurthy G., 2004), correspond to the adjustment of a single or both time series to reach a long-term relation. It means that, unlike the distance method, cointegration has the ability of forecasting based on past information. In order to find the cointegrated data elements, the following steps have to be taken:

- Identification of futures contract pairs that could potentially be cointegrated;
- Verification of the potential pairs on the basis of the proposed hypothesis that the futures contracts pair is indeed cointegrated due to the information obtained from historical data;
- Examination of the cointegrated pairs to determine whether they can be used for trading (Vidyamurthy G., 2004).

There are two pair selection methods considered in the given research. In order to fully employ the HFT, hardware acceleration should be included in the overall solution. It can be achieved by utilizing specific hardware to gain higher computational results. Different hardware may be used to achieve high-performance computing, such as the CPU, the GPU, or the FPGA. In order to choose the best configuration, it is necessary to know the pros and cons of each mentioned technology.

2.2. Hardware acceleration with the CPU, the GPU and the FPGA

Firstly, the difference between the CPU and the GPU was explored. The modern CPU contains a number of cores optimized for sequential serial processing, whereas a GPU consists of hundreds of ‘smaller’, more ‘efficient’ cores designed for handling multiple tasks simultaneously.

The difference between the CPU and the GPU is that GPU is highly specialized in number crunching, i.e., something that graphics processing demands, as it involves millions, if not billions, of calculations per second. The number of cores that the GPU has depends on the manufacturer. The ability of a GPU with 100+ cores

to process thousands of threads can accelerate some software by 100 times over a CPU alone. What is more, the GPU achieves this acceleration while being more power- and cost-efficient than a CPU (Asaduzzaman A., et al., 2014; Nambia P.P., et al., 2014). Consider:

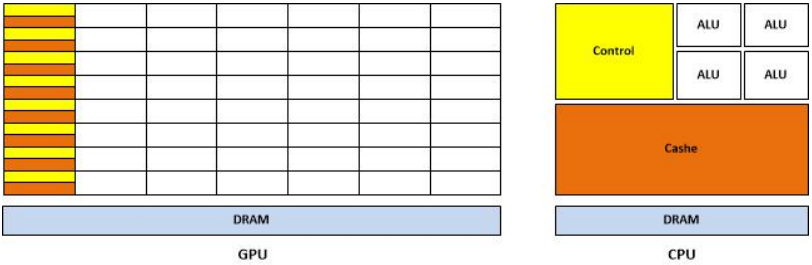

Fig. 1. The GPU and the CPU (Asaduzzaman A., et al., 2014)

A specific system of application delivers high performance with little flexibility, whereas the general-purpose processors provide flexibility with an average performance. As is well known, the general-purpose processors are developed to perform numerous operations; however, they are not suitable for intensive image processing applications due to their serial processing blocks. The GPU devotes much more transistors to data processing than to data caching and flow control. The GPU architecture is highly parallel; it has a large number of cores that are designed for handling multiple tasks simultaneously.

It should be noted here that in embedded systems, the Field Programmable Gate Array (FPGA) has been the leading processor technologies. In other words, the FPGA is a solution for low power consumption, whereas the GPU offers the solution for high expense (Chelva M. S. and Sharanappa V. H., 2016).

Drawing on the previous research (Véstias M. P. and Horácio C. N., 2014; Jones D. H., et al., 2010; Grozea C., et al., 2010; Minhas U. I., et al. 2014), the comparison of the GPU, the FPGA and the multiprocessors can be summarized as given in the table below:

Table 1. The FPGA, the GPU and the multicore processor comparison

FPGA	**Multicore Processor**	**GPU**
Parallel	Sequential	Limited parallel
Extremely fast real time processing	Varies with dependency	Fast real time processing
No good in floating point operations	Versatile	Excellent in floating point operations
Need to interface the software	No need for special interfacing	No need for special interfacing
Comparatively less flexible but with better performance	-	Programming flexibility

Table 1 can be explained as follows: firstly, the power consumption of the FPGA based solution offers significantly lower power consumption than in the case of the GPU ones. Secondly, with regard to time development, in the case of the FPGA, code and cores are added, therefore the simulation and implementation time increases, thus increasing the time of development. In comparison to the GPU case, the development time is much faster because the GPU uses existing libraries with high-level programming languages, such as C/C++, Python, and other. Thirdly, the parallelism should be discussed since the platforms like the FPGA and the GPU are highly parallel; yet, as a rule, they have lower clock frequency than the CPU. Thus, the entire power of the FPGA and the GPU lies in their high number of computational cores. It should be stressed that the code pointing from the CPU to the GPU via the CUDA (Compute Unified Device Architecture) is straightforward, whereas the FPGA implementation is much more difficult because in this frame, it is impossible to directly port an algorithm from the CPU to the FPGA. As for floating point operations, the FPGA seems to suffer due to the fact that it can only work with the fixed-point units, whereas the GPU works perfectly with the floating point. Finally, interfacing is problematic with the FPGA and easy with the GPU because it is difficult to integrate it into any system, and the CUDA in its turn is developed for the hybrid CPU/GPU systems.

It can be summarized that the FPGA is power-efficient and the GPU is cost-efficient: it takes less time to develop the GPU solution, and platform integration is less expensive than in the case of the FPGA. Moreover, the GPU allows easy integration to the common CPU – GPU system. It is so because the FPGA is designed to perform the concurrent fixed-point operations with a close-to-hardware programming approach, whereas the GPU is optimized for the parallel processing of floating-point operations with the use of thousands of small cores and therefore they do not need special interfacing. While working with HFT, the floating-point operations cannot be avoided, since the work is performed with the financial instrument price with floating-point digits. The above factors confirmed that the use of the GPU via the CUDA to further develop the HFT solution would be the optimal choice for hardware application.

2.3. The GPU selection (the CUDA)

As for graphics calculations, they require little control and communication compared to the volume of calculations (Kirk D.B. and Hwu W.M., 2010). The GPU is specifically designed to tackle the problems that can be organized as data-parallel computations with high arithmetic intensity. A typical GPU is organized as an array of highly threaded streaming processors (SPs) distributed among streaming multiprocessors (SMs). It is multiplatform and can be compiled for any of the new Nvidia GPU architectures. The concepts of thread, thread block, and grid are three abstractions often referred to in the CUDA programming paradigm. A thread is each of the many components responsible for executing a given instruction over a single data. According to the SIMT paradigm, multiple threads work in parallel executing the same kernel on a set of data. Threads are divided into thread blocks each of which is run by every SM of the GPU. Threads within a block are able to share data through the SM's shared memory, and they can be synchronized at a

certain point of their execution. Thread blocks are grouped into grids that spread them among all the SMs of the GPU. A thread block can be organized as a one-, two- or three-dimensional array of threads, and the CUDA offers variables with which the index of each thread inside its block can be recovered. By analogy, the grids may be one-, two- or three-dimensional arrays of blocks. The thread blocks within the grids may also be identified by means of indices (Labaki J., et al., 2011).

The data in the CPU global memory takes more time to be processed compared with the data in the GPU global memory. In order to take full advantage of the GPU global memory and available cores, the CUDA was selected to further develop this research.

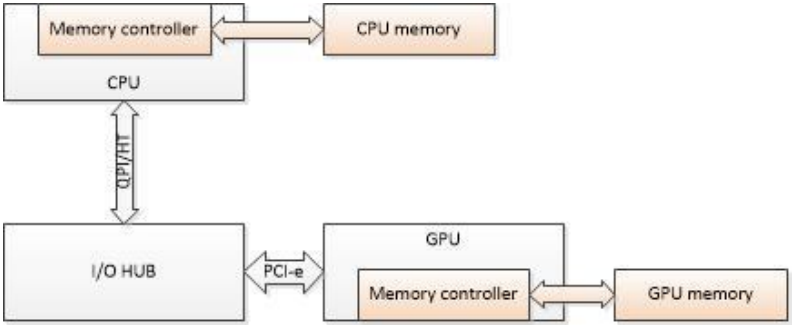

Fig. 2 Connecting a CPU and a single GPU via a motherboard with a PCI-e connector

The ability to use more than one GPU in a system is only possible with the help of a motherboard, as they usually have more than one PCI connector. Also, server-based solutions for many GPUs are usually made by the GPU manufacturers themselves. Proper use of a system with many GPUs depends on how well they are connected to the system. There are two possible ways to connect them. The first one is connecting GPUs to PCI ports as separate devices (D. Foley and J. Danskin, 2017).

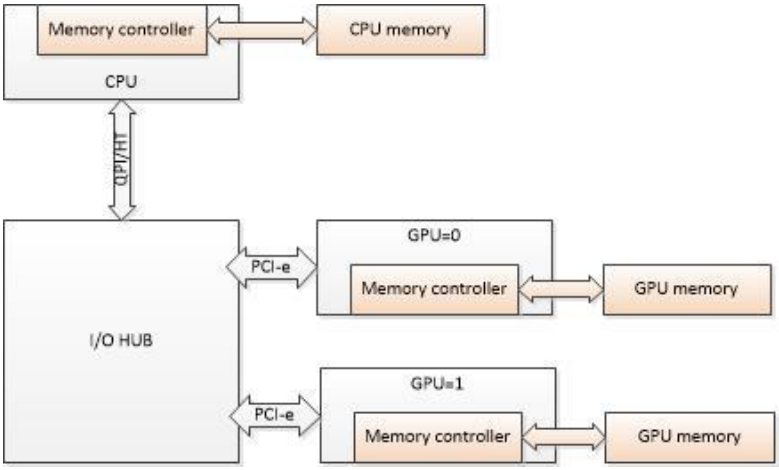

Fig. 3. Connecting a CPU and multiple GPUs via a motherboard with a PCI-e connector

The second option is to have a physical bridge that connects the individual GPUs. The GPUs connected in this way function as one and are more efficient than those connected to individual PCI connectors.

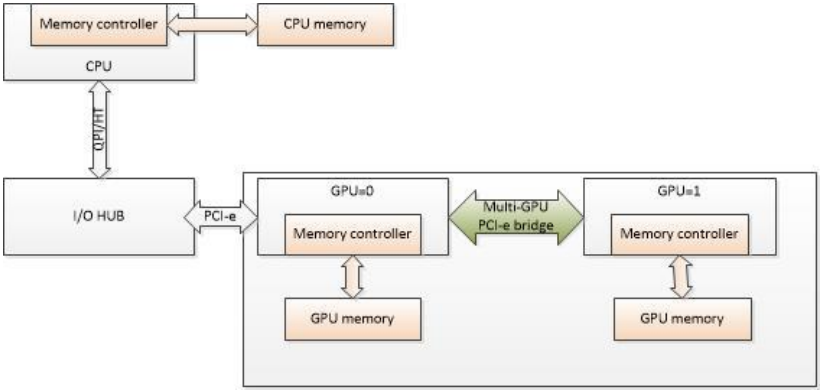

Fig. 4. Connecting a CPU and multiple GPUs via a motherboard with a PCI-e bridge connector

When using this type of connection, the maximum amount of data transferred is 16 GB per second, which is the highest throughput of PCI 3.0. This happens because the whole bridge of multiple GPUs is connected via one PCI slot in the motherboard. It is the weakness of such a system. Therefore, such a system can be used if the applications used require little data transfer between the CPU and the GPU and do not require a high degree of communication between individual GPUs. When large amounts of data transfer between the CPU to the GPU and between the GPU itself are required, CPU and multiple GPUs via a motherboard connection have to be used, which has been selected in this research (D. Foley and J. Danskin, 2017).

2.4. Code vectorization, multidimensional matrices and parallel kernels

Vectorization is the process of transforming an algorithm that operates on the single elements of data into a program that operates on the matrices of data. In computer science, vectorization is the process of converting an algorithm from a scalar implementation, which performs an operation on one pair of operands at a time, to a vector process, when a single instruction can refer to a vector. Such transformation requires converting the control flow of the input program to the data flow by inserting masking instructions and guards. In effect, it adds a form of parallelism to software in which one instruction or operation is applied to the multiple pieces of data (Vaughan C.T., et al., 2018; Li P., et al., 2015).

As a result, the program can be executed on a machine that provides vector instructions, the action is executed faster. In order to perform the mentioned transformation, the developers can either use compiler intrinsic and write a vector code directly or a semi-automatic vectorizing compiler and write annotations in the parts of the program that should be vectorized. The performance or efficiency benefits from vectorization depending on the code structure (Li P., et al., 2015).

When the vectorized algorithm is prepared, it can be moved to the GPU. The MATLAB provides with some of the best practices for using the GPU. Firstly, it is necessary to profile the code in order to identify bottlenecks. Secondly, the code must work on large enough matrices in order to see the benefits of the GPU parallelization (Perino F., 2014).

In order to optimize and accelerate the code, it is necessary to minimize the data transfer between the CPU and the GPU. There are a couple o ways to achieve it:

- Sustained use of supported functionality;
- Creation of variables on the GPU (Asaduzzaman A., et al., 2014).

Once the data transfer between the CPU and the GPU is minimized it is very important to use array indexing and branching in moderation. Every time when loop or index search is invoked in the GPU, the data must be sent to the CPU to make an array work with indexing.

In the given research, the MATLAB was used for the GPU computing, which allowed us to accelerate an application with the GPU. With the MATLAB language, it is possible to get the advantage of the CUDA GPU computing technology without having to learn the intricacies of the GPU architectures or low-level GPU computing libraries.

Applications consist of the interconnection of producer-consumer kernels (Sugerman J., 2009). Parallel kernels are a semantic set of core blocks grouped temporarily, repeated many times, and they can be executed simultaneously without distorting the result. They consist of various software structures that include loops, recursions, or library calls. These kernels make up most of the program execution time (Uhrie R, et al., 2020).

Parallel programming with CUDA involves the direct implementation of the SAXPY routine defined in the BLAS linear algebra library both sequentially and in parallel. Given vectors x and y with n floating-point numbers, it updates y every $\alpha x + y$. Serial

implementation is a simple loop that computes one *y* element in each iteration. The parallel kernel efficiently performs each of these independent iterations in parallel, assigning a separate thread to compute each *y* element (Garland M., et al. 2008).

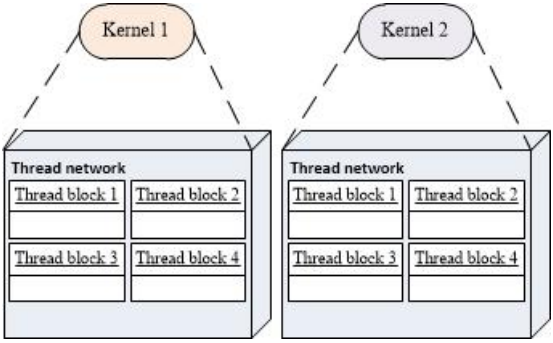

Fig. 5. Parallel kernels

The CUDA program is reorganized into the main program consisting of one or more serial threads running on the CPU and one or more parallel kernels suitable for execution in a parallel processing device, such as the GPU. The kernel runs a scalar sequence program in a set of parallel threads. The programmer can combine these threads into a network of thread blocks. The threads in the thread block can be synchronized with each other and use shared, high-speed GPU on chip memory. The threads from different blocks in the same network can only be coordinated by performing operations on shared GPU memory that is visible to all threads. CUDA requires that the blocks of threads would be independent, which means that the kernel must be executed correctly, regardless of the order in which the blocks are executed even if all blocks are executed consistently arbitrarily, without exception. This constraint due to the interdependence of the thread blocks for the kernel provides an opportunity to expand the computational quantities (Garland M. et al., 2008; Uhrie R. et al., 2020).

If loop is evoked on the GPU, in order for it to work, the CPU has to be invoked, and the data from the GPU global memory has to

be sent to the CPU memory and, at the end of the loop, it would come back to the GPU global memory. It is quite a long root for the data to move and it is extremely time-consuming. One of the ways to avoid loops is to create the 3D/4D matrices at the expense of memory. Thus, in this research, the algorithm was coded into the GPU avoiding loops and creating multidimensional matrices.

For instance, a 3D matrix consists of three axes where x represents futures contract, y represents prices for each futures contract for the given trading windows, and z represents the size of the data passed or the pages of the 3D GPU matrix. In other words, if there is a 2D matrix in which the row represents prices and the column represents futures contract, then in a 3D matrix, each page would consist of these 2D matrices.

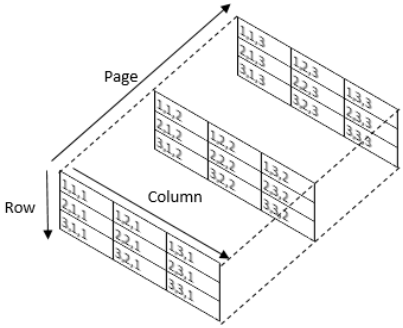

Fig. 6. A 3-dimensional GPU matrix

When creating a 4D matrix, one more dimension is used, where each page consists of a previously explained 3D matrix. A 4D GPU matrix was used only once in this research, i.e., when the cointegrated method for trading pair selection was applied.

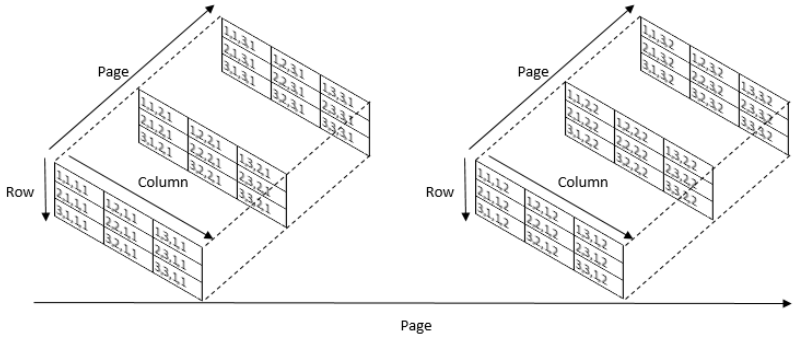

Fig. 7. A 4-dimensional GPU matrix

A 4D GPU matrix is created when a function is used to compute the augmented Dickey–Fuller test. Actually, function compares each futures contract with other ones when prices are already normalized. When a 4D matrix is created, it is of the size of 10x2x30x30, where x is futures contracts, y represents prices for each contract in trading period, z axis refers to the size of the possible pair combinations, and q axis represents the size of all the possible pair combinations to compare different combinations with each other.

2.5. Logical architecture

Each program that runs inside the GPU is divided into many threads. All individual threads execute the same program code only with different data. A thread block is made up of a group of threads that, using shared block memory, communicate with each other and synchronize their work that they can coordinate through shared memory. The thread blocks can be further integrated in a network consisting of a group of thread blocks. Separate procedures are performed by merging and creating threads in a network (Plosk N. and Samaras N., 2016). Each individual CUDA thread can access the data in different memories. All threads have only their own local

memory. All thread blocks have shared memory that is accessible to each thread in that block and that memory is accessible as long as the specified block exists. Also, each thread in the thread block network can access shared global memory of the GPU device.

Using the CUDA model, a programmer can access different memories in GPU that are visible to threads of running procedures. Threads that are grouped into blocks can communicate with each other using shared memory that is only visible to that block. Each thread that is part of the procedure has its own local memory. There is a third type of global memory that is accessible to all threads, regardless of thread blocks. The availability of these memories to threads greatly accelerates the calculation of procedures and the execution of tasks (Horrigue, L. et al., 2018).

One of the software packages that supports GPU and CUDA is MATLAB. In the field of quantitative finance, MATLAB has always been one of the main languages (Jacquier A., 2017). The algorithm for the proposed method of backtesting high-frequency statistical arbitrage strategies was developed using MATLAB, which is also one of the most commonly used backtesting platforms in the fields of finance and HFT (Josephine A. and Fransson L., 2016). Although MATLAB is not well known for its low latency, programming languages such as C ++, C # and Java, but it is more suitable for testing. The developed MATLAB algorithm can be compiled into C language or C executable files, thus achieving low latency (Saikia M. J. et al., 2014).

Multidimensional matrices are used in this paper to accelerate the processing of large amounts of data. The logical architecture of transferring multidimensional matrices to the GPU and calculating them can be described as follows: suppose we have a function that is f (A, B) = bsxfun (@minus, A (3,3,3), B (3,3,3)), where A and B are multidimensional gpuArray arrays. Here, bsxfun is the procedure that calls function A-B. Both gpuArray multidimensional arrays A and B are first created in the GPU's global memory.

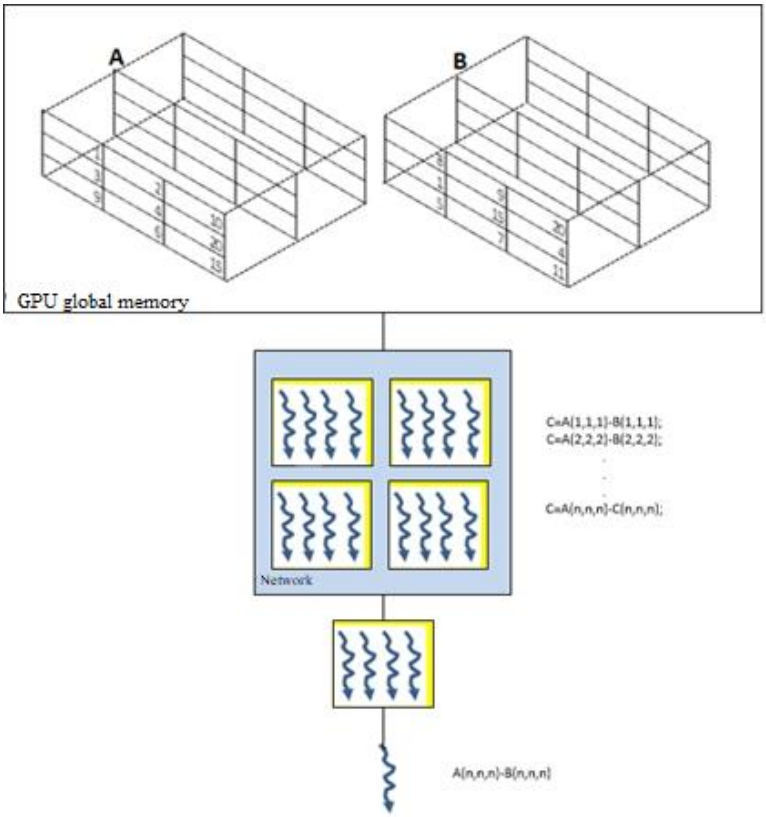

Fig. 8. Procedures and threads in a logical architecture using a GPU

Although arrays are in the format of multidimensional matrices, they are stored as column-major in the GPU itself. Running the procedure function f (A, B) = bsxfun (@minus, A (3,3,3), B (3,3,3)) in MATLAB environment creates thread blocks in which each thread performs the same calculation A (n, n, n) -B (n, n, n). In this way, calculations are performed in parallel simultaneously on all the thread blocks in the threads that are connected in the network.

3. PROPOSED HFT TESTING METHODOLOGY

Before delving into the problem of methodology, it is important to describe the execution model of the CUDA. In this model, the main component is kernel, a function in C programming language, which is executed N times in parallel with N CUDA threads (Klöckner A., et al., 2012). The CUDA threads are often stored in blocks that are called thread blocks. As indicated above, such blocks can be arranged with one (x), two (x, y) or three (x, y, z) dimensional objects. Such GPU CUDA hierarchy offers a convenient framework for handling vector, matrix or multidimensional matrix computation. Furthermore, the CUDA execution model is like the SIMD (Single Instruction Multiple Data) architecture. Every instruction that is executed in a CUDA kernel is an instruction similar to the type of SIMD, which means that it always operates with a vector type of data (Coon B. W. and Lindholm J. E., 2008; Coon B. W. and Lindholm J. E., 2009; Coon B. W., et al., 2010; Coon B. W., et al., 2011).

Taking into account the above-mentioned facts, it becomes clear that the way to move the HFT to the GPU is by vectorizing the algorithm of the statistical arbitrage strategies developed in earlier research and using the multidimensional matrices. In the next section, the proposed method for HFT testing is presented focusing on how to implement the HFT in the GPU by using the global memory, vectorizing the code and implementing multidimensional matrices.

3.1. The developed method for the HFT testing

The main aim of the proposed method is to receive high-frequency data from an electronic market, send it from the CPU memory to the GPU global memory, where calculations are parallelized and processed, the buy/sell decisions are made and the information is sent back to an electronic exchange. The data obtained

from an electronic market that it is stored in the CPU memory is immediately sent to the GPU global memory and trading algorithm or algorithms are evoked. Based on the proposed method (Fig. 9), the backtesting of the high-frequency statistical arbitrage trading strategies is introduced. As mentioned above, for the HFD tick-by-tick data one month of backtesting would be enough; however, in the given research, a three month period was used. It is worth noting that algorithmic trading differs from other types of investments in its ability to provide more reliable expectations about future performance in comparison with past performance, since there is abundant data availability. In this paper, backtesting is carried out by exposing the high-frequency statistical arbitrage trading strategies to a stream of the historical HFD, which leads to a set of trading signals. Each trade has either profit or loss and demonstrates how fast the trading signal was detected and sent to an electronic exchange.

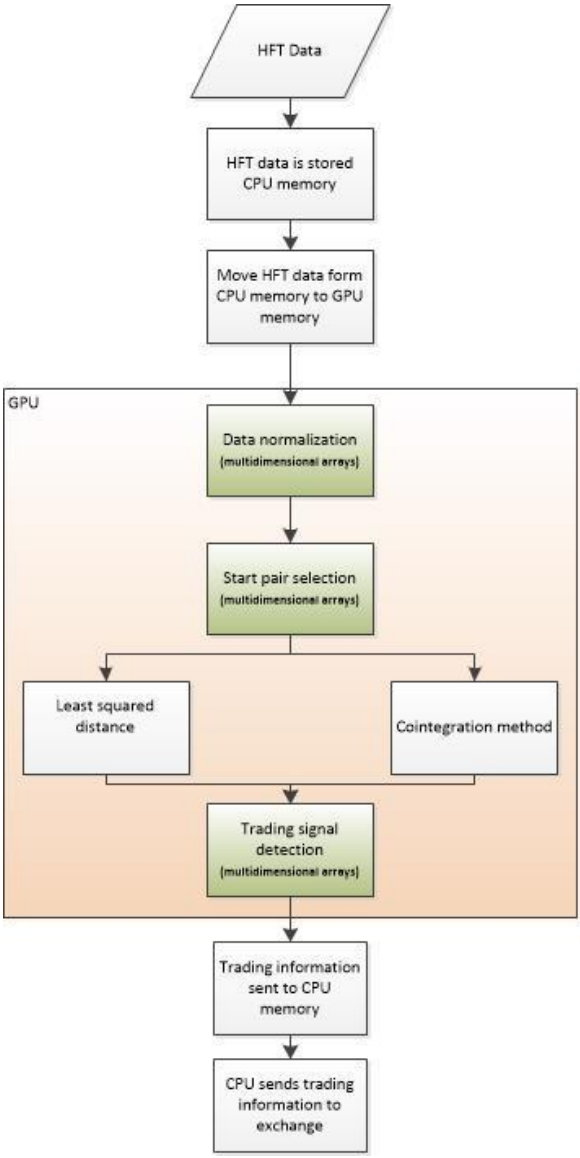

Fig. 9. The proposed method for the HFT on the GPU

After splitting the data using multidimensional matrices, it is necessary to parallelize all the possible calculations in order to

achieve a higher speed of trading decisions. To make better use of GPU cores and memory, kernel parallelization was used to implement code vectorization. By transferring the statistical arbitrage HFT algorithm to the GPU, the algorithm data is formed as multidimensional matrices and the algorithm calculations are performed in parallel by splitting them between separate parallel kernels, allowing HFT decisions to be made faster.

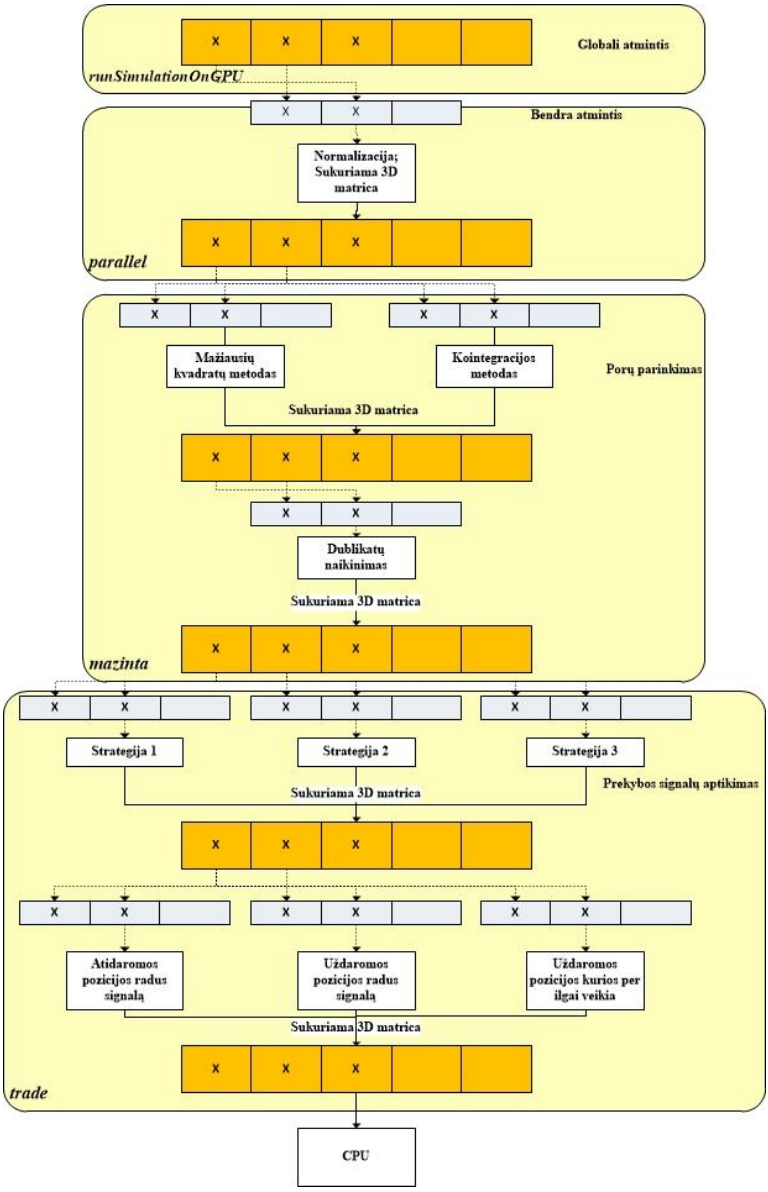

Fig. 10. Parallelization of kernels for HFT statistical arbitrage strategies in the GPU environment

After transferring the resulting high-frequency data stream from electronic exchanges to the GPU global memory, they are formed as multidimensional matrices and distributed to the shared thread block memory. Then data can be normalized in parallel. The obtained normalized data are formed as multidimensional matrices. They are returned to the global memory and moved back to the shared memory of the thread blocks. At this step, it is possible to parallelize two trading pair selection as kernels that can be run simultaneously. These kernels are cointegration and least squares pair selection methods. Returning the data to the global memory removes the trading pair's duplicates. The study used three statistical arbitrage trading strategies the trading signals of which can be selected in parallel. At the final step, positions are closed / opened in parallel, and positions held for too long are closed.

The aim of trading algorithm on the GPU is to parallelize all possible calculations across the CUDA cores at the expense of the GPU global memory. As shown in the Figure 10, the parts that are in the GPU can be parallelized by implementing a multidimensional matrix. Due to the way the algorithm works, the following functions can be parallelized:

- Data normalization;
- Pair selection for the trading period;
- Looking for trading signals;
- Evoking trading and closing positions based on the received data.

At the beginning of the HFT testing method that used in this thesis, data is received from an electronic exchange where the CPU prepares the data that is later transferred to the GPU. The data transfer is performed using the *runSimulationOnGPU* function described in the algorithm, which starts with the two graphics cards and creates GPU variables. The *parallel* function is then launched to parallelize the calculations and transfer the entire trading algorithm to the GPU. In order to perform calculations faster, they need to run inside the GPU. Calculations are performed during each CPU CUDA

kernel cycle. Calculations are not queued but performed simultaneously. With the help of the *parallel* function, the first three-dimensional matrix is created which forms a 130x62000 matrix with futures prices indicating purchase and sale prices. The formation of this matrix triggers the following function that is called *mazinta*; it helps to parallelize the selection of trading pairs, the detection of trading and the closing trades signals. Additional multidimensional matrices are created at the expense of GPU global memory. The first created matrix is for all possible pairs, for each futures contract. In parallel, the search of possible trading pairs is performed using cointegration and least squares methods. However, the cointegration method uses 4D matrices and extends the computation time because some of the data has to be kept in a queue, which does not fully fit in the available GPU memory. Therefore, after conducting the study and developing a method that is not only faster but also more reliable for the selection of pairs, the least squares method was chosen. When the pair selection function *pair* is started, a new 3D matrix of only the selected pairs is created. At the same time, the algorithm eliminates possible duplicates of trading pairs. The *trade* function is then launched, which performs the detection of trading signals on all three trading strategies. The results obtained are stored in a 3D matrix. In parallel, existing positions are checked to see whether a closing signal has occurred or whether the maximum time to keep the positions open has already been reached. The trading algorithm ends when it sends the generated orders to the CPU that sends trading or cancellation orders to electronic exchanges according to the received information.

When the high-frequency data comes from the exchange, it goes through the CPU memory to the GPU global memory where each futures contract prices are first normalized. The normalization process is parallelized so that all contracts prices are normalized at once by implementing the MALTAB built it functions arrayfun and bsxfun.

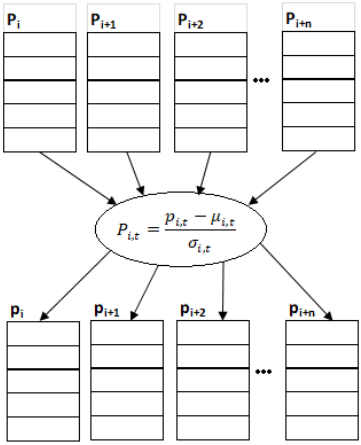

Fig. 11. The high-frequency data normalization

With the help of the mentioned functions, each futures contract price *p* is normalized by performing the normalization function, and normalized prices *P* are used for further calculations.

First, the 2D matrix of data is loaded to the CPU global memory where columns represent the prices for different futures contracts and rows represent different timestamps in nanoseconds for the given day trading period.

The next step after the normalization is to look for all the pairs of futures contracts that could be used for trading signal detection. For this step, only normalized futures contract prices are used, and all the calculations are done on the GPU. The following pair selection methods are implemented:

- Least squared distance method;
- Cointegration method.

When the least square distance method is used for each futures contract, the closest member is searched. For each futures contract, the pair is formed by finding another futures contract with a minimized sum of squared difference *S* in the normalized futures contracts prices (Huck N. and Afawubo K., 2015).

$$S_{i,j=}\sum_t^T (P_{i,t} - P_{j,t})^2 \quad (2)$$

Here $P_{i,t}$ and $P_{i,t}$ are normalized prices of different futures contracts *i* and *j*, on time *t* and *T* is the trading window size (Huck N. and Afawubo K., 2015).

The second method was based on cointegration and used to detect the pairs for trading. Firstly, the algorithm checks whether all the series are integrated in the same order by implementing the ADF. After that, the cointegration tests are performed on all the possible combinations of pairs. Cointegration is tested by using Engle and Grangers 2-step approach and Johansen test (Huck N. and Afawubo K., 2015).

The augmented Dickey Fuller test for a unit root assesses the null hypothesis of a unit root:

$$y_t = c + \phi y_{t-1} \beta_1 \Delta y_{t-1} + \cdots + \beta_p \Delta y_{t-p} + \varepsilon_t \quad (3)$$

Here Δ is the differencing operator that $\Delta yt = yt - yt\text{-}1$. The variable *p* is determined empirically so that the mean zero error term ε_t is serially uncorrelated. The null hypothesis of a unit root is as follows:

$$H_o: \phi = 1 \quad (4)$$

Its alternative is ϕ <1. If the ADF test is passed, then cointegration is tested by applying the Engle and Grangers two-step approach and Johansen test. The Engel and Ganger cointegration test is a two-step approach embracing the following movements: if $P_{1,t}$ and $P_{2,t}$ are the prices of the futures contracts at time *t*, then the first step will require the regression of $P_{1,t}$ against $P_{2,t}$:

$$P_{1,t} - \beta P_{2,t} = \mu - \varepsilon_t \quad (5)$$

Here μ denotes an intercept. The cointegration between the two futures contracts is examined performing the analysis of the order of integration of the residuals ε_t by using the ADF test. The futures contracts are cointegrated on the condition that the residuals of the regression are stationary.

If these tests are passed, the Johansen approach is adopted once again to test whether the futures contract pair might be used for

trading. This approach helps to test the hypothesis of *r*, the unrestricted cointegrating relationships in the unrestricted Vector Autoregressive (VAR) model. The cointegrated pairs with the highest trace statistics are kept as possible pairs. Then a deviation of the relation $P_{1,t}$ - $\beta P_{2,t}$ from its historical mean μ is interpreted as a possible trading pair (Huck N. and Afawubo K., 2015).

Before looking for all the possible pairs, the 3D matrices are created. Each 3D matrix represents a different futures contract and its possible pair. Here the first column represents the futures contract *Pi* that is used for the whole 3D matrix, and the second column represents its possible pair. Thus, each page of the mentioned matrix represents all the possible pairs for the futures contract *Pi* with normalized prices information. It is important to stress that 130 3D matrices are created at this particular step. When all the 3D matrices are prepared, the pair selection is implemented. Once again with the help of the MATLAB built in function pagefun, all the pair selections are parallelized and performed immediately as pagefun function is applied to each page.

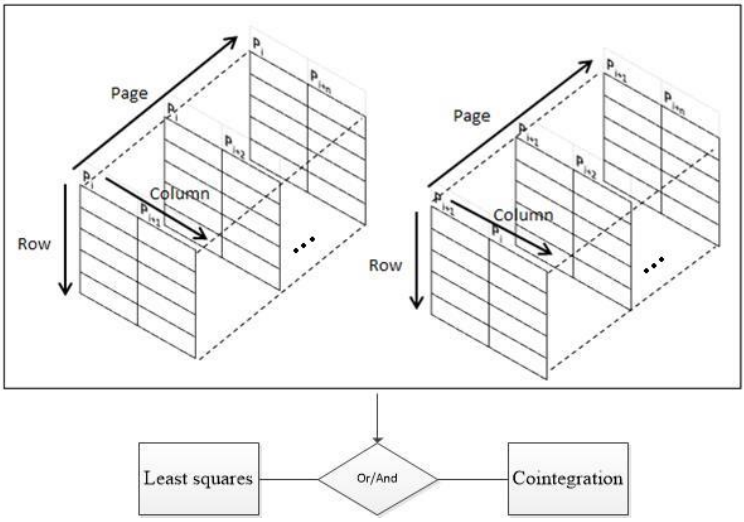

Fig. 12. High frequency data converted to the 3D GPU matrix and passed to selected pairs for trading

As mentioned above, there are two possible methods for pair selection, least square and cointegration. Depending on the parameters, one or both methods could be employed. When both methods are used, the pair is selected if both methods find that it is cointegrated and in a minimal distance.

After all the possible trading pairs are found, a new 3D matrix is created on their basis to perform trading selection on each page in parallel. For the created 3D matrix, each first column represents the futures contract *Pi* and the second column represents its pair *PPi*; rows represent the normalized prices for the given futures contract. Each page of 3D matrix represents a different pair.

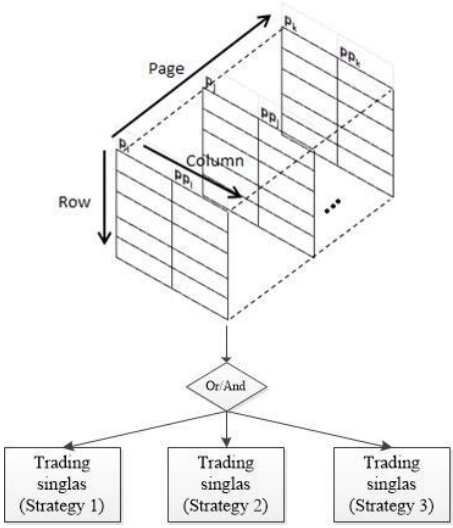

Fig. 13. A 3D GPU matrix with the selected pairs is passed to the trading signal selection part of the algorithm

Once a 3D matrix of the pairs is prepared, the trading signal search is executed. Depending on the strategy, the trading signal search may be different. On the basis of the parameters, it is possible to use one, two or all three trading signal detection strategies. Then the data is checked in order to see if the same trading signal has not been already sent or if it is already closed and could be reopened.

When the trading signal is detected, the information from the GPU memory is sent to the CPU memory and then goes to an electronic exchange. The pair is used only by one trading strategy till the closing signal or the period for keeping the position open is reached.

Finally, when all the aspects of the suggested statistical arbitrage for HFT are explained and implemented in the given research, the experimental setup is executed with backtesting and the speed of the trading decisions is measured.

4. EXPERIMENTAL SETUP

It should be pointed out here that the research is based on 130 different futures contracts that were provided by the NANOTICK company and contains tick-by-tick data of the contracts from the ME group embracing the NYMEX, the COMEX and the CBOT. The data is used from the period between the 30th of April 2008 and the 3rd of August 2018 that encompasses 69 trading days. All the information about these 130 futures contracts is given in nanosecond timestamp precision. Once the data was received, the average data sending time was measured. The average time of the data sent from this electronic market was 32,27 microseconds. This means that new information about one of the provided futures contract is received every 32,27 microseconds. The information from the electronic market necessary for the given research is the buy and the sell prices of the futures contracts and the timestamp.

Considering the fact that new information from the electronic market provided by the NANOTICK Company comes on average every 32,27 microsecond, the algorithm will have to make the buy/sell decision during the indicated time or even faster in order to achieve HFT and beat other market participants.

4.1. Requirements for solution implementation

The proposed statistical trading algorithm consists of several parts that can be executed in parallel. The parallelization is limited by the nature of the trading algorithm and the hardware in place. In the given experiment, NVIDIA GeForce GTX 1060 6GB and NVIDIA GeForce GTX 1070 Ti 8GB have been used.

The performance of a supercomputer is measured by the number of floating-point operations per second (FLOPS) that the machine is able to perform. Thus, it was interesting to calculate FLOPS of the used system. In order to determine how well the algorithm was utilizing the GPU, the calculation of the number of FLOPS was made which was 13,276 FLOPS.

4.2. High frequency data in the research

The NANOTICK Company provided with 130 different future contracts, encompassing 18 agriculture, 15 energy, 21 equities, 37 fx, 29 interest rate and 10 metals futures contracts. Each contract consists of data with nanosecond precision timestamps.

Table 2. Future contracts used in the research

Agri-culture	Energy	Equit.	Equit.	FX	FX	FX	Inter. rate	Inter. rate	Metals
ZS	CL	ES	RSV	6E	SEK	ACD	GE	TUL	HG
ZM	RB	NQ	TPY	6J	EAD	ENK	ZQ	TAF	GC
KE	HO	YM	SDA	6A	EPZ	_	ZN	FIT	SI
ZL	BZ	RTY	_	RP	ZAR	_	ZB	FIX	MGC
LE	NG	NKD	_	6C	NOK	_	UB	NON	QO
ZW	QM	NIY	_	RY	PSF	_	NBY	NCB	PL
ZC	HH	BTC	_	ECD	J7	_	ZF	TUF	SIL
SOM	MB	EMD	_	M6E	ANE	_	FYN	TFY	ZNC
HE	NOB	IBV	_	6N	6Z	_	TUB	NUB	QC

XK	FOB	FT1	_	6S	MJY	_	FYT	ZT	PA
GF	BOB	XAY	_	AJY	6L	_	TN	N1U	_
XC	HP	XAK	_	E7	M6A	_	TUT	_	_
GD	NBP	XAE	_	6M	PLN	_	TFY	_	_
XW	TTE	FTU	_	PJY	SIR	_	BUB	_	_
DC	QG	XAV	_	RF	MCD	_	NOL	_	_
CSC	_	RS1	_	M6B	MSF	_	FOL	_	_
CJ	_	RSG	_	CNH	ESK	_	TEX	_	_
HET	_	XAI	_	SEK	MIR	_	TUX	_	_

The main problem related to the high-frequency data is caused by the discrepancies between the timestamps of the correlated contracts. It occurred that the timestamps for trades or bid/ask changes during the same trading second differ. For instance, if the timestamp of the quote for the contract A changes at 16:45:00.024827526, in the contract B, one can notice quote changes at 16:45:00.027226312 and no activity at 16:45:00.024827526. This requires comparing the timestamp sequences of the correlated futures contracts. If the timestamps differ, the contract that lacks the timestamp is filled in with a missing timestamp and the previous bid/ask or trade prices. As a result, not only the timestamps of the correlated contracts are brought together but also the prices are kept accurate.

All the information vital for this research is shown in the table below:

Table 3. Data set example

Receiving Date	**ReceivingT ime**	**ID**	**Symbol**	**Asset**	**Security Group**	**Entry Type**	**Entry Price**
20180604	16:45:00.02477685	15144	GE:PS M1-M2	GE	GE	A	0,75
20180604	16:45:00.02479496	15144	GE:PS M1-M2	GE	GE	A	0,5
20180604	16:45:00.02481003	15144	GE:PS M1-M2	GE	GE	A	0,25
20180604	16:45:00.02482752	15144	GE:PS	GE	GE	A	0,25

			M1-M2				
20180604	16:45:00.02482752	15144	GE:PS M1-M2	GE	GE	B	0,25
20180604	16:45:00.02493238	15144	GE:PS M1-M2	GE	GE	A	0,25
20180604	16:45:00.02493238	15144	GE:PS M1-M2	GE	GE	B	0,25
20180604	16:45:00.02494946	15144	GE:PS M1-M2	GE	GE	A	0,25
20180604	16:45:00.02494946	15144	GE:PS M1-M2	GE	GE	B	0,25

As shown in the table above, there are different entry types, A and B, where A represents ask price and B represents bid price. The trading algorithm used in this research shortens the futures contract with the bid price and takes a long position of the futures contract with the ask price. Since all the data is of high frequency, all the market orders are made in high-frequency mode, which reduces the risk of the order execution slippage to a minimum.

4.3. Tested trading strategies for the HFT

The statistical arbitrage strategies that are applied in this research have already been adopted in previous studies that measured their performance and profitability (Vaitonis M. and Masteika S., 2018; Vaitonis M., 2017; Vaitonis M. and Masteika S., 2016; Vaitonis M. and Masteika S., 2018; Vaitonis M., 2018). It was concluded that such "end-of-day" strategies can be implemented on high-frequency data and hence used in this research. Although these strategies are based on the same methodology, their trading signal detection methods differ. Statistical arbitrage strategy suggested by J. Caldeira and G. V. Moura (Caldeira J. and Moura G. V., 2013) detects trading signals by calculating at first the difference ε_t between the pair for normalized price of the futures contract:

$$\varepsilon_t = P_{i,t} - p_{i,t}. \quad (6)$$

In the formula (2), ε_t indicates the difference between the futures contracts $P_{i,t}$ and its pair $p_{i,t}$ at time t. Later on the threshold z_t is found:

$$z_t = \varepsilon_t - \mu_t \sigma_t \quad (7)$$

Here μ_t is the mean and σ_t is the standard deviation of the found pair of the futures contracts for former trading window. When the threshold z_t is calculated, the strategy may start searching for the trading signals that are detected using the following logic (Caldeira J. and Moura G. V., 2013):

Open long position, if $z_t < -2\sigma$;

Open short position, if $z_t > 2\sigma$;

Close short position, if $z_t < 0.75\sigma$;

Close long position, if $z_t > -0.50\sigma$ (Caldeira J. and Moura G. V., 2013).

The second strategy is based on the research carried out by M. S. Perlini (Perlin M. S., 2009). Similarly to the former strategy, it is necessary to calculate the difference ε_t between the two normalized prices of the futures contracts and measure it against the threshold *d*, which is defined by the trader before starting the process of trading. When ε_t is calculated, long and short positions are established.

The trading positions are kept open till the spread becomes smaller than the threshold *d*. If there are two commodity futures contracts *A* and *B*, one pair for the selected trading window would be $A_{i,t}$ and $B_{i,t}$ and the very trading would look as follows:

while d > spread;

if $A_{i,t} > B_{i.t}$, *then short* $A_{i,t}$ *and long* $B_{i,t}$;

in other case, long $A_{i,t}$ *and short* $B_{i,t}$ (Perlin M. S., 2009).

The basic idea of the described statistical arbitrage strategy is to follow the distance between the pair of the correlated futures contracts normalized prices and wait till the threshold *d* is reached. In such a case, there is a probability that the prices are going to converge in the future, and this opportunity can be exploited to gain the profit (Perlin M. S., 2009).

The third employed statistical arbitrage strategy is based on D. Herlemont's research offering the theoretical statistical arbitrage model and focusing on its efficiency (Herlemont D., 2013).

While applying this trading strategy, mean μ_t and standard deviation σ_t for the given trading window of the differences of normalized futures contract prices have to be calculated. When the difference of prices of found futures contract pair *A* and *B,* price is $<2*\sigma_t$:

If $A_t > B_t$*, then open short position with* A_t *and long position with* B_t*;*

If $A_t < B_t$*, then open long position with* A_t *and short position with* B_t (Herlemont D., 2013).

All the positions should be closed when the differences between the normalized prices of A_t and B_t pair are $< \mu_t$, or the maximum period of keeping the positions open is reached (Herlemont D., 2013).

The data is not applied to the real market, but is used in a simulated market. In order to find out whether profit or loss is generated at the end of the research, all the arrays for each statistical arbitrage strategy with the real prices for the opening and closing long or short positions are considered to measure the result obtained by the statistical arbitrage strategy (Vaitonis M. and Masteika S., 2018; Perlin M. S., 2009):

$$PL=\Sigma(ls_i-lb_i)-K \quad (8)$$

$$PS=\Sigma(ss_i-sb_i)-K \quad (9)$$

The variable *PL* represents the profit from a long position, and the variable *PS* marks the profit from a short position, while whereas the variable *i* represents the trade that the profit/loss is calculated for. The profit from a long position is the difference between the futures commodity contract *i* sell – *ls* and buy – *lb* values minus *K*, which isis the transaction cost representing the number of contracts. The profit/loss for short positions is found in a similar manner; , where it is equal to the difference between the futures commodity contract *i* sell – *ss* and buy – *sb*, minus *K*. Finally, at the end of each algorithm, the total profit is calculated (Vaitonis M. and Masteika S., 2018; Perlin M. S., 2009):

$$TP=PL+PS \quad (10)$$

It is important to stress that the given research is not based on the profitability aspect for each trading strategy but on how fast each of the trading strategies can generate and send the signal to the market.

5. EXPERIMENTAL RESULTS

The previous research attempted to confirm that the use of HFT brings better results compared to the profitability offered by each strategy (Vaitonis M. and Masteika S., 2018; Vaitonis M., 2017). The very fact that all the trading strategies can bring profit to a trader demonstrates the necessity to create a new method for making trading decisions with the use of the HFT mode enabling to send trading decisions to an electronic exchange faster than new information is received.

As the research focus on the very speed of data processing, it is measured which way of pair selection was less time-consuming. In order to find it out, two weeks trading period from the 30th of April 2018 to the 11th of May 2018 was tested. The average time showed how much time was required to process the prices data from the electronic exchange and send it back in seconds. Consider the results given below:

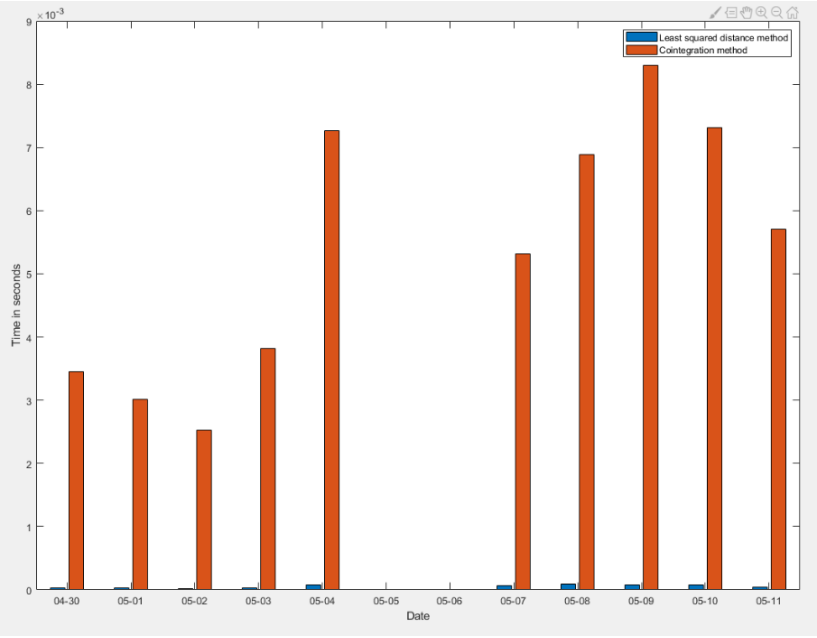

Fig. 14. Time comparison based on the average response time for each time data received from the electronic exchange each trading day worked with the use of the least squared distance and cointegration methods

It is hardly surprising that the least squared distance method was less time-consuming. Even though both the cointegration and least squared distance pair detection methods were parallelized, the former included the measurement of more parameters and the calculation of a significantly higher number of variables. In addition, the use of the four-dimensional matrices which had to be converted to the three-dimensional matrices for the GPU due to its architecture, could not reach a better performance with the cointegration method. The aim was to beat the time of sending data from the exchange in 32,27 microsecond.

Table 4. The average response time on each trading day was calculated by using the least squared distance and cointegration methods for selecting the trading pairs

Date	Cointegration (μs)	Least squared distance (μs)
2018.04.30	345,62	0,9446
2018.05.01	411,8	0,3017
2018.05.02	253,12	0,22312
2018.05.03	381,86	0,3018
2018.05.04	726,42	0,8264
2018.05.05	0	0
2018.05.06	0	0
2018.05.07	531,42	0,67142
2018.05.08	688,87	0,88587
2018.05.09	830,38	0,80938
2018.05.10	731,56	0,80156
2018.05.11	571,07	0,40101

Since the cointegration method needs more time to analyze the given data with the hardware used in this research compared to the data obtained from the electronic exchange, it was refused. Only the least squared distance method was implemented. However, the calculations of the Sharpe Ratio for each method were done in order to find out which of them is more suitable.

The Sharpe Ratio is calculated by using the formula $S=\frac{\mu_{PnL}}{\sigma_{PnL}}$ where μ_{PnL} is the mean of profit and loss and σ_{PnL} is the standard deviation from *PnL*. The same procedure was followed adopting the strategy with the use of nanosecond and microsecond data.

Table 5. The Sharpe Ratio for each pairs selection method

Pair selection method	Average	Standart Deviation	Sharp Ratio
Least square distance	2060,55	2039,217237	1,010461819
Cointegration	951,514	1332,581485	0,714038351

To summarize, the higher the Sharpe Ratio, the better strategy it marks. The table above demonstrates that the least squared distance method is not only able to respond faster than the market provides the information but also the Sharpe Ratio is higher, which reveals that this method results in a better overall performance when compared to the response time. Thus, in order to work with the given high-frequency data without further modification of the hardware used in this research, only the least squired method could be effective.

On average, each trading day consists of 536909612 timestamps with 130 futures contracts bid/ask prices, resulting in the total of 69798249560 records per day to be analysed. Depending on the liquidity of the day, the number may vary. In the graph below, more detailed information of the number of timestamps processed each day is provided:

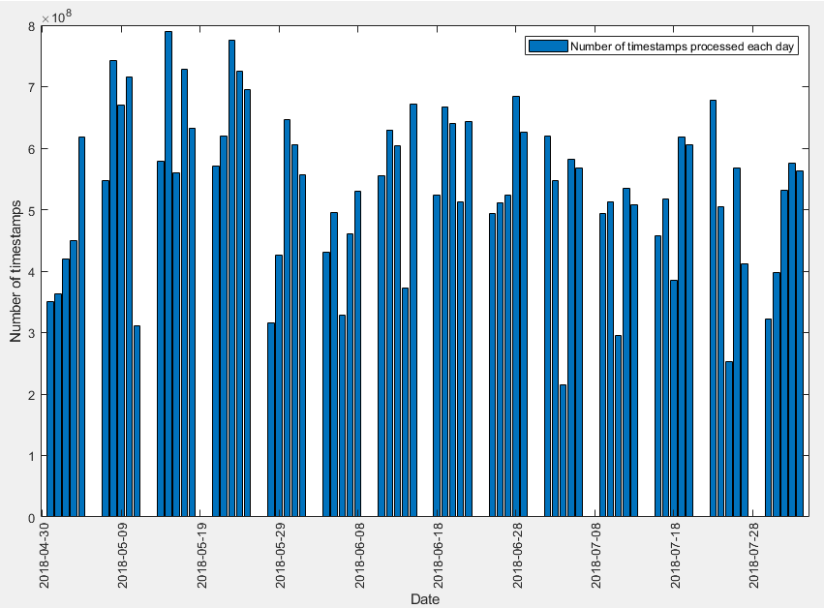

Fig. 15. The number of timestamps for each trading day

Actually, it took 711,11 nanoseconds on average to process each timestamp in 62000 tick-by-tick trading window. It is necessary to compare this result with the average response time for trading strategy given in the graph below. It comes as no surprise that on the days when more timestamps occured, it took a bit more time fo algorithm to process the given data. Nevertheless, the response time was still faster than the time needed for the information to come from the electronic exchange. This can be explained by the fact that in 20 second or 62000 tick-by-tick trading window, more information had to be processed.

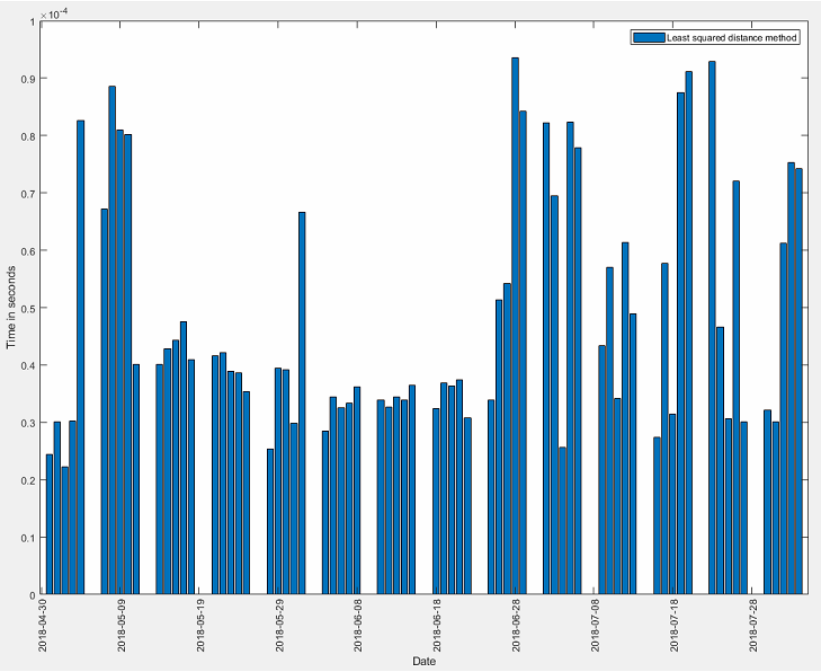

Fig. 16. High frequency trading average response time for every trading day

The implicated parallelization methods help to improve the speed of data normalization, pair selection, and trading/closing signal detection. Since all the data from the electronic exchange is received with the nanosecond precision, it has to be processed and sent back to the exchange at the same speed or even faster.

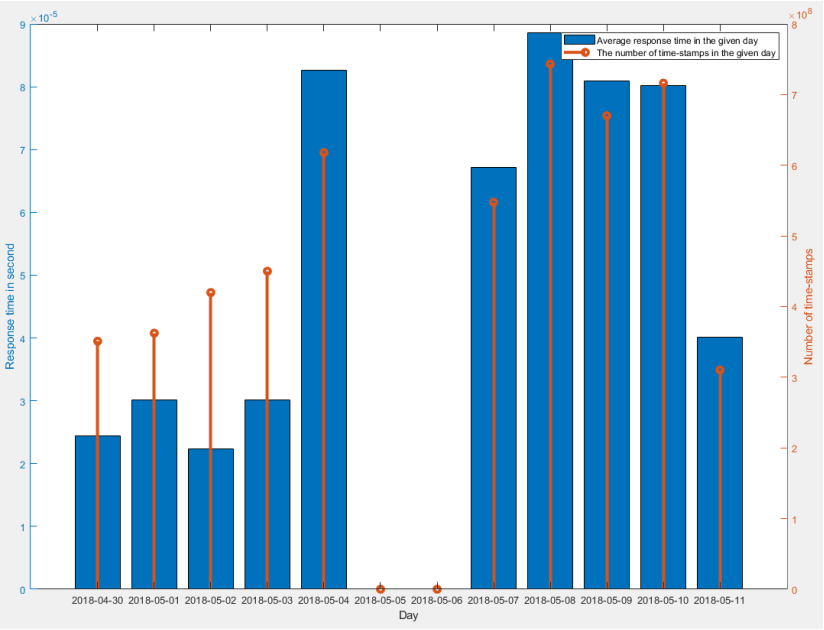

Fig. 17. High frequency trading time and the number of timestamps for a given day

Looking a bit closer at the trading period from the 30th of April 2008 to the 3rd of August 2018 provides us with a better understanding of the trading response time and the number of timestamps for a given day. There is a clear correlation between these two variables, hence the higher number of the response time, the more timestamps occur. It is reasonable that the days with a higher number of the timestamps would require more calculations to be made by the trading algorithm, which happens due to more changes and variations in the 62000 tick-by-tick data trading window from the electronic market of the future markets. Such alterations include the modifications in a high number of pairs, the creation of more trading signals, and more opening and closing position. Thus, more buy/sell signal recalculation will have to be made, due to the higher intensity of the swapping of the trading pairs.

Overall, the proposed methodology of HFT significantly increased the response time of the algorithm with regard to the typical solutions. As a result, it improved the response time by 83,7 %. As mentioned earlier, the previous research tested the same algorithm moved to the GPU without the multidimensional matrices, no code vectorization, no parallel kernels and with only five futures contracts that managed to give back information to the electronic exchange in 849,49 microseconds. The proposed backtesting method with the multidimensional matrices, code vectorization, and parallel kernels when moved to the GPU with CUDA did improve the response time almost 10 times and managed to send response in 711,11 nanosecond.

6. CONCLUSIONS

1. The thesis identifies the HFT strategies with the largest market share. In the work, a market-leading statistical arbitrage strategy that ensures the interconnection of correlating financial instruments and the mutual integrity of financial markets is formalized. Experimental studies have shown that daily closing price statistical arbitrage strategies can be effectively applied in HFT. After the computerization of the HFT strategies for statistical arbitrage, the need for parallel calculations in the proposed method to perform HFT tests with historical data was identified. A comparative analysis of the FPGA and the GPU application that addressed the challenges identified the advantage of GPU in terms of its floating-point number operator and the economy of the solution.

2. While developing the method, the parts of the HFT algorithm that are subject to parallelization were identified: HF data normalization; selection and cancellation of the trading pairs; HFT trade signal identification; trading position opening/closing. The algorithm has been transferred to the GPU. The use of code vectorization for the algorithm helped to achieve the average trading

decision speed of 2.77 milliseconds but it was insufficient in the high-frequency data environment.

3. Multidimensional matrices for the GPU calculations have been applied to increase the average HFT decision-making speed along with code vectorization. The research found that using these computational parallelization methods, the speed of trading decisions improved to 76.88 microseconds, but this was an insufficient speed compared to the speed of the data received from the electronic exchanges, which is equal to 32.27 microseconds.

4. In order to make better use of the GPU cores and memory, kernel parallelization was implemented. Combined with code vectorization and multidimensional matrices, the proposed method achieved the HFT speed that is higher than the speed of high-frequency data received from the electronic exchanges. The experiment showed that the number of loops for computations was reduced using the proposed method, so the decision speed was increased up to 389.63 times compared to the original method and was equal to 711,1 nanoseconds, which is 45.37 times faster than the speed of the data received from the stock exchanges.

LIST OF PUBLICATIONS ON THE TOPIC OF DISSERTATION

1. Masteika S., Vaitonis M., Quantitative Research in High Frequency Trading for Natural Gas Futures Market, Business Information Systems Workshops, Springer International Publishing, Vol. 228,p. 29-35, 2015.

2. Vaitonis M., Masteika S. (2016). High frequency statistical arbitrage strategy engineering and algorithm for pairs trading selection. 7th International Workshop on Data Analysis Methods for Software Systems [abstracts book], Druskininkai, Lithuania, December 3-5, 2015. ISBN 978-9986-680-58-1. p. 51.

3. Vaitonis M., Masteika S. (2016). Research in high frequency trading and pairs selection algorithm with Baltic region

stocks. Information and Software Technologies. 22nd International Conference, ICIST 2016, Druskininkai, Lithuania, October 13-15, 2016, Proceedings. ISBN 978-3-319-46254-7, p.p. 208 – 217.

4. Vaitonis M., (2017). Pairs Trading Using HFT in OMX Baltic Market. Baltic J. Modern Computing, Vol. 5(2017), No. 1, 37-49

5. Vaitonis M., Masteika S. (2017) „Statistical Arbitrage Trading Strategy in Commodity Futures Market with the Use of Nanoseconds Historical Data“,Information and Software Technologies: 23rd International Conference, ICIST 2017, Druskininkai, Lithuania, October 12–14, 2017, Proceedings. R. Damaševičius and V. Mikašytė (Eds.): ICIST 2017, CCIS 756, pp. 303–313, ISBN 978-3-319-67642-5

6. Vaitonis M., Masteika S. (2018). Experimental Comparison of HFT Pair Trading Strategies using Microsecond and Nanosecond Future Commodity Contracts Data. Baltic J. Modern Computing, Vol. 6(2018), No. 2, 195-216, ISSN 2255-8950

REFERENCE

Aldridge I. (2013), High-Frequency Trading: A Practical Guide to Algorithmic Strategies and Trading Systems. John Wiley & Sons, 368 p. ISBN: 978-0-470-57977-0.

Aldrige, I. (2010), High-Frequency Trading - A Practical Guide to Algorithmic Strategies and Trading Systems, John Wiley and Sons, Inc, Hoboken, New Jersey.

Anane M. and Abergel F., (2014). Optimal high frequency strategy in an omniscient order book.

Asaduzzaman A., Gummadi D. and Yip C. M., (2014). A talented CPU-to-GPU memory mapping technique, *SOUTHEASTCON 2014*, Lexington, KY, pp. 1-6.

Beckhardt B., Frankl D.E., Lu C. and Wang, M.I. (2016). A Survey of High-Frequency Trading Strategies.

Binh D., Faff R. and Hamza K., (2006). A new approach to modeling and estimation for statistical arbitrage. *In Proceedings of 2006 Financial Management Association European Conference*, pp. 87–99.

Busch D. (2016) MiFID II: regulating high frequency trading, other forms of algorithmic trading and direct electronic market access, *Law and Financial Markets Review*, Vol. 10, No. 2, pp.72-82.

Busch D., (2016) MiFID II: regulating high frequency trading, other forms of algorithmic trading and direct electronic market access, *Law and Financial Markets Review*, Vol. 10, No. 2, pp. 72-82.

Caldeira J., Moura G. V. (2013). Selection of a portfolio of pairs based on cointegration: A statistical arbitrage strategy, *Revista Brasileira de Financas*, Vol. 11, No. 1, pp. 49–80.

Chelva M. S. and Sharanappa V. H., (2016). A Performance Study of GPU, FPGA, DSP and Multicore Processors For Embedded Vision Systems.

Coon B. W. and Lindholm J. E. (2008). System and method for managing divergent threads in a SIMD architecture. US Patent 7,353,369.

Coon B. W. and Lindholm J. E. (2009). System and method for managing divergent threads using synchronization tokens and program instructions that include setsynchronization bits. US Patent 7,543,136.

Coon B. W., Lindholm J. E., Mills P. C. and Nickolls J. R. (2010). Processing an indirect branch instruction in a SIMD architecture. US Patent 7,761,697.

Coon B. W., Nickolls J. R., Lindholm J. E. and S. Tzvetkov D. (2011). Structured programming control flow in a SIMD architecture. US Patent 7,877,585.

Dacorogna M. M., Gencay R., Muller U., Olsen R. B., and Olsen O. V., (2001). An introduction to high frequency finance.

International Review of Economics & Finance, Elsevier, Vol. 12, No. 4, pp. 525-529.

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.

Driaunys K., Masteika S., Sakalauskas V., Vaitonis M. (2014). An algorithm-based statistical arbitrage high frequency trading system to forcast prices of natural gas futures. *Transformations in business and economics*, Vol. 13, No. 3, p. 96–109.

Durbin M., (2010). All About High-Frequency Trading. McGraw-Hill, New York.

Engle, R. F. and Granger C. W. J., (1987). Co-integration and error correction: Representation, estimation, and testing, *Econometrica*, Vol. 55, No. 2, pp. 251–276..

Foley D. and Danskin J., (2017). Ultra-performance Pascal GPU and NVLink interconnect. *In: IEEE Micro*, Vol. 37, No. 2, pp. 7–17.

Garland M., et al. (2008). Parallel Computing Experiences with CUDA, *IEEE Micro*, Vol. 28, No. 4, pp. 13-27.

Gatev E., Goetzmann W. N., Rouwenhorst K. G. (2006). Pairs Trading: Performance of a Relative-Value Arbitrage Rule. *The Review of Financial Studies*, Vol. 19, No. 3, p. 797-827..

Grozea C., Bankovic Z. and Laskov P., (2010). FPGA vs. Multi-core CPUs vs. GPUs: Hands-On Experience with a Sorting Application, Facing the Multicore-Challenge.

He F., Ren Y., Chen Y. and Cao Q. (2015). GPU Accelerate RD Algorithm Based on MATLAB, *International Journal of Advanced Research in Computer Science & Technology*, Vol. 3, No. 4, pp. 84-86.

Herlemont D. (2013), "Pairs Trading, Convergence Trading, Cointegration", *Quantitative Finance*, Vol. 12, No. 9.

Horrigue, L., Ghodhbane, R., Saidani, T. and Atri M. (2018). GPU acceleration of image processing algorithm based on Matlab CUDA. *International Journal of Computer and Network Security*, Vol. 18, pp. 91–99.

Huck N. and Afawubo K., (2015). Statistical arbitrage and selection methods: is cointegration superior?, *Applied Economics*, Vol. 47, No. 6, pp. 599-613

Jacquier A., (2017). Some Notes On Python For Finance, Department of Mathematics, Imperial College Londo, URL https://wwwf.imperial.ac.uk/~ajacquie/IC_IntroPython/IC_IntroPython_Docs/PythonNotes.pdf.

Jones D. H., Powell A., Bouganis C. H. and Cheung P. Y. K., (2010). GPU Versus FPGA for High Productivity Computing, *In Proceedings of the 2010 International Conference on Field Programmable Logic and Applications (FPL '10)*. IEEE Computer Society, Washington, DC, USA.

Josephine A. and Fransson L., (2016). Algorithmic Trading Based on Hidden Markov Models, , URL https://gupea.ub.gu.se/bitstream/2077/44767/1/gupea_2077_44767_1.pdf.

K. Group (1992). OpenGL - The Industry's Foundation for High Performance Graphics. URL https://www.opengl.org/.

K. Group (2009). The open standard for parallel programming of heterogeneous systems. URL https://www.khronos.org/opencl/.

Kaya O. (2016), High – frequency trading. Reaching the limits, *Automated trader magazine*. Vol. 41, pp. 23 – 27.

Kearns M., Kulesza A. and Nevmyvaka Y., (2010). Empirical Limitations on High Frequency Trading Profitability, URL https://ssrn.com/abstract=1678758.

Kirk D.B., Hwu W.M., (2010). Programming Massively Parallel Processors: A Hands-On Approach, Morgan Kaufmann Publishers.

Klöckner A., Pinto N., Lee Y., Catanzaro B., Ivanov P. ir Fasih A.(2012). PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation, *Parallel Computing*, Vol. 38, No. 3, pp. 157–174.

Krauss C. (2015), "Statistical arbitrage pairs trading strategies: Review and outlook", *IWQW Discussion Paper Series*, No. 09/2015.

Labaki J., Ferreira L. O. S. and Mesquita E., (2011). Constant Boundary Elements on graphics hardware: a GPU-CPU complementary implementation. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, Vol. 33, No. 4, pp. 475-482.

Lai T. L. and Xing H., (2008). Statistical Models and Methods for Financial Markets, Springer Texts in Statistics, Springer.

Li P., Zhang Q., Zhao R. and Yu H., (2015). Data layout transformation for structure vectorization on SIMD architectures, *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, Takamatsu,, pp. 1-7.

Mackintosh P., (2019), Time is Relative: Where Trade Speed Matters, and Where It Doesn't, URL https://www.nasdaq.com/articles/time-relative%3A-where-trade-speed-matters-and-where-it-doesnt-2019-05-30.

Mariano R. S. and Kuen Tse Y., (2008). Econometric Forecasting And HighFrequency Data Analysis,Lecture Notes Series, Institute for Mathematical Sciences, National University of Singapore, World Scientific Publishing Company.

Matlab. (2015). se.mathworks.com. URL https://se.mathworks.com/discovery/matlab-gpu.html.

Matlab. (2016), se.mathworks.com. URL https://se.mathworks.com/help/distcomp/gpu-computing.html.

Miller R. S., Shorter G., (2016). High Frequency Trading: Overview of Recent Developments, *Congressional Research Service*, April 4; Washington D.C.

Miller S. J. (2006). The method of least squares. Mathematics Department Brown University.

Minhas U. I., Bayliss S. and Contantinides G. A., (2014). GPU vs FPGA: A Comparative Analysis for Non-standard Precision, *In proceedings of Reconfigurable Computing: Architectures, Tools, and Applications*, pp. 298-305

Nambia P.P., Saveetha V., Sophia S. and Sowbarnika A., (2014). GPU Acceleration Using CUDA Framework, *International*

Journal of Innovative Research in Computer and Communication Engineering, Vol. 2 No. 3, pp. 200-205.

Nvidia (2007). CUDA Toolkit Documentation. URL http://docs.nvidia.com/cuda/cuda-cprogramming-guide/index.html.7

Nvidia (2007). Nvidia CUDA C/C++. URL https://developer.nvidia.com/cuda-toolkit.

Nvidia (2013). Parallel Thread Execution ISA. URL http://docs.nvidia.com/cuda/parallel-threadexecution/index.html.

Nvidia (2018). Nvidia Volta Architecture Whitepaper. URL http://images.nvidia.com/content/volta-architecture/pdf/volta-architecturewhitepaper.pdf.

Nvidia (2019). CUDA toolkit documentation. URL https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

Nvidia. OpenCL Programming Guide for the CUDA Architecture. URL http://www.nvidia.com/content/cudazone/download/OpenCL/NVIDIA_OpenCL_ProgrammingGuide.pdf.

Perino F., (2014). Getting Started with MATLAB. URL https://www.fisicamedica.it/sites/default/files/documenti/Perino.pdf.

Perlin M. S. (2009). Evaluation of Pairs-trading strategy at the Brazilian financial market, *Journal of Derivatives & Hedge Funds*, Vol. 15, No. 2, pp. 122–136.

Saikia M. J., Kanhirodan R., and Vasu R. M., (2014). High-Speed GPU-Based fully three-dimensional diffuse optical tomographic system. *Journal of Biomedical Imaging*, Vol 3.

Sugerman J., Fatahalian K., Boulos S., Akeley K. and Hanrahan P., (2009). Gramps: A programming model for graphics pipelines, *ACM Trans. Graph*., Vol. 28, pp. 4:1–4:11.

Uhrie R., Chaitali C. and John Brunhaver (2020). Automated Parallel Kernel Extraction from Dynamic Application Traces. ArXiv abs/2001.09995.

Vaitonis M. (2017). Statistical arbitrage Using HFT in OMX Baltic Market, *Baltic J. Modern Computing*, Vol. 5, No. 1, pp. 37-49.

Vaitonis M., (2018). CPU and GPU Implementations for High Frequency Trading in Algorithmic Finance, *Proceedings of the International Conference on Information Technologies*, pp. 119 – 124

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.

Vaitonis M., Masteika S. (2017). Statistical arbitrage trading strategy applied on future commodity market using nanosecond information, *ICIST 2017, Communications in Computer and Information Science*, Vol. 756. Springer.

Vaitonis M., Masteika S., (2018), Experimental Comparison of HFT Statistical arbitrage Strategies Using the Data of Microsecond and Nanosecond Future Commodity Contracts, *Baltic J. Modern Computing*, Vol. 6, No. 2, pp. 195-216.

Vaughan C.T., Cook J., Benner R.E., Dinge D.C., Lin P.T., Hughes C., Hoekstra R.J., Hammond S.D., (2018), On the Use of Vectorization in Production Engineering Workload, *Proceedings CUG 2018 Stockholm*, Sweden from 20-24 May.

Véstias M. P. and Horácio C. N., (2014). Trends of CPU, GPU and FPGA for high-performance computing. *24th International Conference on Field Programmable Logic and Applications* (FPL).

Vidyamurthy G. (2004). Pairs Trading – Quantitative Methods and Analysis. John Wiley & Sons, p. 224.

Zubulake, P. and Lee, S. (2011), The High Frequency Game Changer - How Automated Trading Strategies Have Revolutionized the Markets, John Wiley and Sons, Inc, Hoboken, New Jersey.

ABOUT THE AUTHOR

Mantas Vaitonis obtained bachelor's degree in 2011 in the field of Business Informatics at Vilnius University, Kaunas faculty. From

2011 till 2015 he worked as IT administrator at LTD NOVAKOPA. In 2015 received Master of Business Informatics at Vilnius University, Kaunas faculty. From 2015, is working as IT project manager at LTD NOVAKOPA. He was a PhD student at Vilnius University Kaunas Faculty from 2015 to 2019.

DIDELIO DAŽNIO KOMPIUTERIZUOTŲ PREKYBOS STRATEGIJŲ INŽINERIJA FINANSINĖSE RINKOSE

SANTRAUKA

Elektroninėse vertybinių popierių biržose prekyba atliekama investuotojų ir informacinių technologijų specialistų kuriamais ir automatizuotais prekybiniais algoritmais. Iki šios dienos prekybiniai algoritmai generuoja bemaž du trečdalius visų sandorių finansų rinkose. Pastaraisiais metais algoritmų kompiuterizavimas ir būtinybė būti greitesniam už konkurentus, atliekant investicinius sprendimus bei prekybinius sandorius, dienos prekybą perkėlė į nanosekundinių sandorių dažnį. Tobulėjant elektroninių biržų informacinių technologijų infrastruktūrai, greitėjant sandorių atlikimui bei spartėjant atliekamų sandorių dažniui, prekybos finansų rinkų veikla labiau susiduria su informatikos inžinerijos nei finansinių strategijų kūrimo problematika ir uždaviniais. Tampa aktuali ir būtina tiriamoji mokslinės eksperimentinės plėtros veikla, taikant informatikos inžinerijos žinias finansų sektoriaus dalykinėje srityje, kuriant, testuojant ir pasiūlant inovatyvius sprendimus, tinkamus spręsti didelio dažnio prekybos problematiką ir uždavinius. Šiuo darbu siekiama sukurti, pritaikyti ir adaptuoti didelio dažnio statistinio arbitražo prekybos algoritmo testavimo metodą, galintį analizuoti realaus laiko nanosekundžių tikslumo duomenų, gaunamų iš elektroninių biržų, srautą. Siekiant išsiaiškinti, kaip veikia didelio dažnio prekyba elektroninėse biržose, darbe nagrinėti jų šaltiniai ir elektroninės prekybos mokslo raida. Atlikta literatūros analizė

atskleidė, kad trūksta ištestuoto formalizuoto metodo ir technologinio sprendimo, leidžiančio testuoti algoritmines didelio dažnio prekybos – DDP strategijas finansų rinkose. DDP strategijų testavimą numato Europos Sąjungos MiFID II direktyva, kuria siekiama DDP prekybos operatorius įpareigoti testuoti naudojamas algoritmines ir DDP strategijas. Nustatyta, kad nėra sprendimo, kaip testavimą formalizuoti ir atlikti technologiškai, todėl reikia sukurti DDP strategijų testavimo metodą. Siekiant įgyvendinti išsikeltą užduotį, buvo sukurtas DDP statistinio arbitražo strategijų testavimo algoritmo prototipas, kuris remiasi pasiūlytu metodu. Atlikus testavimą, buvo įrodyta, kad taikomas metodas, kuris naudoja GPU atmintį, kodo vektorizavimą, lygiagrečius skaičiavimus, daugiamates matricas ir lygiagrečius branduolius, pasiekia užsibrėžtą tikslą ir geba atlikti duomenų analizę bei priimti prekybos sprendimą greičiau nei atkeliauja nauji duomenys iš elektroninės biržos.

Darbo objektas

Šio darbo tyrimo objektas – didelio dažnio statistinio arbitražo algoritminės prekybos sistemos ir didelio dažnio duomenys elektroninėse finansų biržose.

Darbo tikslas ir uždaviniai

Darbo tikslas – sukurti didelio dažnio statistinio arbitražo prekybos testavimo metodą, priimantį algoritminės prekybos sprendimus greičiau, nei generuojami nauji duomenys elektroninėse biržose.

Darbo tikslui pasiekti iškelti uždaviniai:

1. Formalizuoti didelio dažnio algoritmines strategijas, nustatant technologines specifikacijas didelio dažnio duomenų apdorojimui, ištestuoti statistinio arbitražo DDP strategijų efektyvumą, pritaikyti įgytas žinias šio darbo tikslui pasiekti.

2. Pasiūlyti ir sukurti DDP testavimo ir duomenų apdorojimo realiu laiku metodą, kuris dirbtų su didelio dažnio duomenimis ir

atliktų algoritminės prekybos skaičiavimus sparčiau nei gaunami didelio dažnio duomenys iš elektroninių biržų.

3. Ištestuoti sukurtą metodą, sukuriant DDP testavimo įrankio prototipą, leidžiantį atlikti skaičiavimus su didelio dažnio ir apimties duomenimis.

4. Aprašyti gautus rezultatus ir pastebėjimus, įvertinti sukurto didelio dažnio statistinio arbitražo prekybos algoritmų testavimo metodo taikymo galimybes ir galimus apribojimus.

Tyrimo metodika

Rengiant darbą, remtasi moksline literatūra, moksliniais straipsniais, šaltiniais internete. Darbe taikomi metodai: mokslinės literatūros analizė ir apibendrinimas, stebėjimas, skaitmeninis modeliavimas, atrankos metodas, sintezė, eksperimentai, koreliacinė analizė, statistinė analizė, kompiuterinis duomenų apdorojimas.

Sistemine darbų ir analitine tyrimų apžvalga siekiama išanalizuoti esamas didelio dažnio prekybos sistemas ir technologijas, išsiaiškinti jų taikymo metodus; taip pat siekiama teoriškai pagrįsti kuriamo didelio dažnio statistinio arbitražo prekybos algoritmo ir didelio dažnio duomenų apdorojimo metodo poreikį.

Teorinėje tyrimo dalyje aiškinamas sukurto modelio poreikis, leidžiantis įgyvendinti išsikeltus tikslus, taip pat aprašoma platformos architektūra. Toliau pateikiamas teorinis didelio dažnio statistinio arbitražo prekybos algoritmo prototipas ir didelio dažnio duomenų apdorojimo metodas, nagrinėjamas jų teorinis ir praktinis tinkamumas, pristatomas tyrimo eksperimentas, jame naudojami duomenys ir nurodomos nuostatos, pagal kokius kriterijus bus vertinamas eksperimentas.

Po teorinės dalies aptariamas atliktas eksperimentas, vertinamas sukurtas prototipas, pasirinkta platformos architektūra, argumentuojamas pasirinktas didelio dažnio statistinio arbitražo prekybos sistemos modelis.

Mokslinis naujumas

Nors didelio dažnio prekyba yra neretai aptariama verslo publikacijose, tačiau jos informatikos inžinerijos aspektas mažai nagrinėjamas. Labai mažai informacijos yra apie didelio dažnio prekybos sistemų taikymą (limitacijas ir rizikas) bei optimalius technologinius reikalavimus. Daugumoje mokslinių straipsnių, susijusių su didelio dažnio prekyba, tyrinėjamos optimalių strategijų parinkimo metodologijos, nusakančios, kaip matuoti didelio dažnio prekybos strategijų pelningumą, kokie yra optimalūs prekybos strategijos naudojami parametrai, kada pateikti sandorių informaciją elektroninėms biržoms, kaip matuoti didelio dažnio prekybos aktyvumą, taip pat tai, kas nėra susiję su optimalia didelio dažnio prekybos sistemų konfigūracija, architektūra, metodu, kaip taikyti didelio dažnio prekybos sistemas pasirinktai technologiniai platformai (Vaitonis M., 2018, Kearns M. et al. 2010, Anane M. ir Abergel F., 2014, Beckhardt B. et al., 2016). Todėl svarbu detaliau išnagrinėti didelio dažnio prekybos sistemas, jų taikymą, architektūrą ir galimus inžinerinius sprendimus, kurie padėtų įgyvendinti didelio dažnio prekybą dirbant su didelio dažnio ir apimties duomenimis.

Norint naudoti didelio dažnio prekybos algoritmus, reikia įvertinti, keliose biržose bus prekiaujama, kiek finansinių instrumentų ir skirtingų prekybos strategijų bus naudojama, koks bus naudojamų duomenų dažnumas (sekundiniai, milisekundiniai ir t. t.), koks bus lango, kuriame vyks prekyba, dydis. Priklausomai nuo pasirinktos konfigūracijos, gali paaiškėti, kad, norint įgyvendinti didelio dažnio prekybą, teks lygiagrečiai atlikti tūkstančius skaičiavimų.

Tyrimo metu naudojant statistinio arbitražo strategijas, reikia rasti koreliuotas finansinių instrumentų poras. Tai atlikti, naudojantis didelio dažnio duomenimis, tampa labai sudėtinga. Taip nutinka, nes skirtingi finansiniai instrumentai rinkose juda skirtingu dažnumu, todėl, norint palyginti didelio dažnio prekybos duomenis, juos reikia agreguoti. Dėl prieš tai jau minėtų priežasčių, pereinant į didelio dažnio prekybą su didelio dažnio duomenimis, prekyba vykdoma ne

tam tikru laiku, bet pagal tai, kiek būna gaunama *tick-by-tick* duomenų. Todėl algoritmai seka ne paprastą laiką, bet pasirinkto GPU procesorių ciklų skaičių, t. y. prekybos langas yra ne tam tikras laikas, bet gautų duomenų kiekis ir procesoriaus ciklų skaičius.

Darbe iškelta ir patvirtinta hipotezė – kodo vektorizacijos naudojimas, kai statistinio arbitražo DDP algoritmas perkeliamas į GPU, algoritmo duomenys formuojami kaip daugiamatės matricos ir algoritmo skaičiavimai, vykdomi lygiagrečiai išskaidant juos tarp atskirų lygiagrečių branduolių (angl. *kernel*), leidžia priimti didelio dažnio prekybos sprendimus greičiau nei gaunamas didelio dažnio duomenų srautas iš elektroninių biržų.

Sukurtas didelio dažnio statistinio arbitražo prekybos testavimo metodas, kuriuo šio tipo prekybos algoritmas perkeltas į GPU aplinką, sujungus kodo vektorizaciją, daugiamates matricas ir branduolių lygiagretinimą, yra pakankamas priimti prekybinius sprendimus.

Darbo rezultatų praktinė vertė

Disertacijoje pasiūlyta technologija, būdas ir jais grįstas sukurtas metodas, kuris leidžia apdoroti didelio dažnio ir apimties duomenis didelio dažnio prekybos strategijose greičiau nei gaunami nauji duomenys iš elektroninių biržų. Tai leidžia aplenkti kitus rinkos dalyvius. Sprendimo esmė – ne tik greitas duomenų apdorojimas, bet ir greitas teisingo prekybos sprendimo priėmimas. Atsižvelgus į tai, kad dėl komercializacijos visi didelio dažnio prekybų sistemų sprendimai yra konfidencialūs, būtina moksliškai aprašyti jų veikimo principą ir parodyti, kaip šiose sistemose taikomi metodai gali padėti kitose mokslo srityse. Disertacijoje pasiūlytas metodas gali būti taikomas ne tik realiai prekybai, bet gali būti naudojamas kaip DD statistinio arbitražo strategijų testavimo metodas. Šio metodo reikia dėl MiFID II direktyvos, kuri reikalauja iš prekybos operatorių tikrinti visas algoritminės ir didelio dažnio prekybos sistemas, kurios veikia pas juos.

Atliktų tyrimų pagrindu gali būti kuriama sistema, kuri leistų, naudojantis GPU, atlikti didelio dažnio skaičiavimus. Šie skaičiavimai gali būti pritaikomi didelio dažnio prekyboje siekiant pelno arba elektroninėse biržose didinant likvidumą. Taip pat tie patys metodai gali būti pritaikomi kitose mokslo srityse, kuriose reikia greitai ir teisingai priimti sprendimus, pavyzdžiui dirbtinio intelekto srityje.

Ginamieji teiginiai

Disertacijos ginamieji teiginiai:

1. Didelio dažnio statistinio arbitražo prekybos strategijų formalizavimo metodologija, kuria remiantis galima nustatyti optimalų duomenų normalizavimo ir prekybos langą algoritminėje prekyboje mili ir nanosekundiniu dažniu, yra tinkama įvertinti strategijų efektyvumą.
2. DDP priimami sprendimai, naudojant didesnio dažnio duomenis, yra efektyvesni algoritminėje prekyboje ir nulemia skaičiavimų lygiagretinimo būtinybę.
3. Sujungus kodo vektorizaciją, daugiamates matricas, branduolių lygiagretinimą ir perkeliant formalizuotus algoritmus į GPU, pasiekiama duomenų apdorojimo sparta yra didesnė, nei generuojamas didelio dažnio duomenų srautas elektroninėse biržose.

Darbo aprobavimas

Disertacijos rezultatai pristatyti aštuoniose mokslinėse konferencijose:

1. 2015 metais BIS 2015 tarptautinėje mokslinėje konferencijoje „Business Information Systems Workshops“ Poznanėje, Lenkijoje skaitytas pranešimas „Quantitative Research in High Frequency Trading for Natural Gas Futures Market“.
2. 2015-12-03–2015-12-05 7-ojoje mokslinėje konferencijoje DAMSS 2015 „Duomenų analizės metodai programų sistemoms“ Druskininkuose pristatytas stendinis pranešimas „High frequency

statistical arbitrage strategy engineering and algorithm for pairs trading selection“.

3. 2016-10-13–2016-10-15 ICIST 2016 22-ojoje „Tarptautinėje informacijos ir programų technologijų konferencijoje“ Druskininkuose skaitytas pranešimas „Research in high frequency trading and pairs selection algorithm with Baltic region stocks“.

4. 2016-12-01–2016-12-03 8-ojoje mokslinėje konferencijoje DAMSS 2016 „Duomenų analizės metodai programų sistemoms“ Druskininkuose pristatytas stendinis pranešimas „ Computerized high frequency trading of nanoseconds in futures market“.

5. 2017-10-12–2017-10-14 ICIST 2017 23-ojoje „Tarptautinėje informacijos ir programų technologijų konferencijoje“ Druskininkuose skaitytas pranešimas „Statistical Arbitrage Trading Strategy in Commodity Futures Market with the Use of Nanoseconds Historical Data“.

6. 2017-11-30 – 2017-12-02 9-oje mokslinėje konferencijoje DAMSS 2017 „Duomenų analizės metodai programų sistemoms“, Druskininkuose, skaitytas pranešimas „Research in High Frequency Statistical Arbitrage Strategies Applied to Microsecond and Nanosecond Information“.

7. 2018-04-27 IVUS 2018 23-ojoje tarptautinėje mokslinėje konferencijoje Kaune „Information Society and University Studies“ skaitytas pranešimas „CPU and GPU Implementations for High Frequency Trading in Algorithmic Finance“.

8. 2018-05-29 SYSTEM 2018 23-ojoje tarptautinėje mokslinėje konferencijoje Gliwice, Lenkijoje „Information Society and University Studies“ skaitytas pranešimas „Algorithmic trading and machine learning based on GPU“.

Dizertacijos rezultatai pateikti šešiose mokslinėse publikacijose:

1. Masteika S., Vaitonis M., Quantitative Research in High Frequency Trading for Natural Gas Futures Market, Business Information Systems Workshops, Springer International Publishing, Vol. 228, p. 29–35, 2015 m.
2. Vaitonis M., Masteika S. (2016). High frequency statistical arbitrage strategy engineering and algorithm for pairs trading selection. 7th International Workshop on Data Analysis Methods for Software Systems [abstracts book], Druskininkai, Lithuania, December 3-5, 2015. ISBN 978-9986-680-58-1. p. 51.
3. Vaitonis M., Masteika S. (2016). Research in high frequency trading and pairs selection algorithm with Baltic region stocks. Information and Software Technologies. 22nd International Conference, ICIST 2016, Druskininkai, Lithuania, October 13-15, 2016, Proceedings. ISBN 978-3-319-46254-7, p.p. 208 – 217.
4. Vaitonis M., (2017). Pairs Trading Using HFT in OMX Baltic Market. Baltic J. Modern Computing, Vol. 5(2017), No. 1, 37-49.
5. Vaitonis M., Masteika S. (2017) „Statistical Arbitrage Trading Strategy in Commodity Futures Market with the Use of Nanoseconds Historical Data“,Information and Software Technologies: 23rd International Conference, ICIST 2017, Druskininkai, Lithuania, October 12–14, 2017, Proceedings. R. Damaševičius and V. Mikašytė (Eds.): ICIST 2017, CCIS 756, pp. 303–313, ISBN 978-3-319-67642-5.
6. Vaitonis M., Masteika S. (2018). Experimental Comparison of HFT Pair Trading Strategies using Microsecond and Nanosecond Future Commodity Contracts Data. Baltic J. Modern Computing, Vol. 6(2018), No. 2, 195-216, ISSN 2255-8950.

Bendros išvados

1. Darbe nustatytos didžiausią rinkos dalį užimančios DDP strategijos. Formalizuota rinkoje lyderio pozicijas

užimanti,statistinio arbitražo strategija, užtikrinanti koreliuojančių finansinių instrumentų susietumą, finansinių rinkų tarpusavio integralumą. Atlikus eksperimentinius tyrimus, nustatyta, kad dienos uždarymo kainų statistinio arbitražo strategijos gali būti efektyviai pritaikomos didelio dažnio prekyboje. Kompiuterizavus statistinio arbitražo DDP strategijas, nustatyta lygiagrečių skaičiavimų būtinybė siūlomame metode, siekiant atlikti DDP patikimumo testavimus su istoriniais duomenimis, vykdyti didelio dažnio prekybą realiu laiku. Atlikus palyginamąją FPGA ir GPU taikymo analizę, skirtą spręsti keliamus uždavinius, nustatytas GPU pranašumas dėl slankiojo kablelio skaičiaus operatoriaus, skaičiavimų resursų pakankamumo ir ekonomiškumo.

2. Sudarant metodą, išskirtos DDP algoritmo dalys, kurioms taikytinas lygiagretinimas, t. y. DDP duomenų normalizavimo, prekybinių instrumentų porų parinkimo ir anuliavimo, DDP prekybos signalų identifikavimo, prekybinių pozicijų atidarymo / uždarymo algoritmo dalys perkeltos į GPU skaičiavimus. Algoritmo kodo vektorizavimas parodė, kad vidutinis sprendimų priėmimo greitis lygus 2,77 milisekundėms, tačiau jis yra nepakankamas didelio dažnio duomenų aplinkos prekyboje.

3. Siekiant paspartinti vidutinį DDP sprendimų priėmimo greitį, kartu su kodo vektorizacija pritaikytos daugiamatės matricos GPU skaičiavimams. Atlikus eksperimentą, nustatyta, kad taikant šiuos skaičiavimų lygiagretinimo metodus, prekybos sprendimų greitis pagerėjo iki 76,88 mikrokundžių, tačiau jis buvo nepakankamas, palyginti su gaunamų duomenų greičiu iš elektroninių biržų, kuris lygus 32,27 mikrosekundėms.

4. Norint geriau išnaudoti GPU branduolius ir atmintis, buvo panaudotas branduolių lygiagretinimas. Sujungus jį su kodo vektorizavimu ir daugiamatėmis matricomis, darbe siūlomas metodas pasiekė DDP greitį, kuris yra didesnis už gaunamų didelio dažnio duomenų greitį iš elektroninių biržų. Atliktas eksperimentas atskleidė, kad, taikant pasiūlytą metodą, buvo sumažinta skaičiavimų išsišakojimų, todėl sprendimų greitis padidėjo iki 389,63 kartų,

palyginti su pradiniu metodu, ir tapo lygus 711,11 nanosekundžių. Tai yra 45,37 kartų greičiau nei gaunami duomenys iš biržų.

TRUMPAI APIE AUTORIŲ

Mantas Vaitonis 2011 m. įgijo verslo informatikos bakalauro laipsnį Vilniaus universitete Kauno fakultete. Nuo 2011 iki 2015 m. dirbo IT administratoriumi UAB „NOVAKOPA“. 2015 m. įgijo verslo informatikos magistro laipsnį Vilniaus universitete Kauno fakultete. Nuo 2015 m. dirba UAB „NOVAKOPA“ IT projektų vadovu. 2015–2019 m. jis buvo Vilniaus universiteto Kauno fakulteto doktorantas.

NOTES

NOTES

Mantas Vaitonis
HIGH FREQUENCY COMPUTERIZED TRADING STRATEGIES ENGINEERING IN FINANCIAL MARKETS
Summary of a Doctoral Dissertation
Technological Sciences
Informatics Engeneering (T 007)
Editor Evelina Kazakevičiūtė

Mantas Vaitonis
DIDELIO DAŽNIO KOMPIUTERIZUOTŲ PREKYBOS STRATEGIJŲ INŽINERIJA FINANSINĖSE RINKOSE
Daktaro disertacijos santrauka
Technologijos mokslai
Informatikos inžinerija (T 007)
Redaktorė Gabija Bankauskaitė

Vilniaus universiteto leidykla
Saulėtekio al. 9, LT-10222 Vilnius
El. p. info@leidykla.vu.lt
www.leidykla.vu.lt
Tiražas 35 egz.