CORVINUS UNIVERSITY of BUDAPEST

Doctoral School of Business Informatics

Ph.D. Thesis Summary

László Mohácsi

Business Computing and Parallel Architectures

Supervisors:

Dr. József Abaffy, DSc
Dr. Erzsébet Kovács, CSc

Budapest, 2014

Department of Computer Science

Ph.D. Thesis Summary

Mohácsi László

Business Computing and Parallel Architectures

Supervisors:

Dr. József Abaffy, DSc
Dr. Erzsébet Kovács, CSc

Contents

1. Background and overview of the research

Over the last few years the rate of improvement in the operation execution speed of processor cores has slowed down. (The outline of a promising result in basic research that could bring rapid development cannot be seen – apart from the graphene research which is currently very far from production stage.) The breakthrough cannot be expected in the near future. Therefore massive improvement in the execution speed of programs designed for a single core CPU cannot be expected from the perfection of the hardware manufacturing technology. Increase in execution speed at order of magnitude can only be achieved using parallel architectures to carry out calculations parallel in time. Software development and algorithm design for parallel architectures require a different approach than traditional algorithm design. To utilize parallel architectures tasks have to be split into subtasks that can be carried out parallel in time by individual computing units. Depending on how the results of subtasks are related to each other the performance varies. Some algorithms can be adopted for parallel execution with minor modification, but in many cases major modification or a completely new approach is necessary.

Market pressure encourages hardware manufacturers to design architectures with more computing units on a single chip, but at the same time the speed of the individual computing units increases slowly over time.

Carrying out calculations in parallel is not a new idea. Working on the Manhattan Project Richard Feynman organized calculations in a way that makes them suitable for parallel executions back in 1944. The problem was the calculation of energy release patterns of various ignition bomb layouts using an IBM program-controlled calculator. Gene Amdahl in 1960 recognized that despite the increased number of active parallel data processing units the execution speed is limited by parts of the program where results are based on each other.

Since the 1960-s there is a continuing effort to build and utilize computers with several or a great number of processors. These attempts culminate from time to time in well-established and widely used computers or ideas, sometimes they simply fade away. A recent idea is to use the graphics card in the computers to execute numerical computations, since they have a large number of processing units. Flynn's taxonomy classifying the basic parallel approaches uses the expression of the *Single Instruction Single Data* (SISD) for describing the usual one-processor computer. The main computing unit of a graphics card, which contains hundreds of processors for image rendering, is called GPU. When used for parallel computation the GPU can be called *Single Instruction Multiple Data* (SIMD) computer, since these hundreds of processors execute the same instruction on different data sets. CPUs with multiple cores fall into the *Multiple Instruction Multiple Data* (MIMD) category since cores can carry out calculations independently of each other on different sets of data. (The remaining class of this taxonomy is MISD.)

The different architectures and their hardware implementation are not equally suitable for each problem. In many cases the algorithm has to be adopted to the underlying hardware architecture.

The thesis gives the parallel adaptation of three algorithms of relevance in economic calculations. It mainly focuses on general purpose multi-core CPU and GPU architectures since they are widely available for researchers.

The first chapter of the thesis describes and compares different general purpose parallel architectures. It gives an overview on the strengths and weeknesseses of different architectures and gives and explains the differences through computational problems. The extract of this chapter was published in GIKOF Journal issued by the John von Neumann Computer Society in 2013.

The second chapter introduces an implementation of the ABS algorithm for solving systems of linear equations on a massively parallel GPU architecture. The topic is actual since the excellent stability properties of the ABS algorithm were proved in 2013. One of the leading GPU manufacturer is the NVidia Corporation, their architecture is called CUDA. Features of the CUDA architecture are explained in depth through the ABS algorithm in this chapter.

Based on the third chapter our article “A parallel implementation of an O*(n4) volume algorithm” that was published in the Central European Journal of Operations Research in 2014. My co-author was István Deák. The thesis includes a more detailed version of published materials with deeper explanations and execution results that couldn’t fit into the article due to volume limitations. By the parallel implementations deeper understanding of the algorithm’s behavior was made possible even in higher dimensions.

Our results were first introduced at the 30th Conference on Operations Research organized by the Hungarian Operations Research Society in 2014.

I have chosen the demographic forecasts and the estimation of future pension as the topic of the fourth chapter. Two approaches are described: the cohort-component method and the microsimulation method. The chapter focuses on the microsimulation method since in case of estimating the future pension the macro approach is not suitable. Apart from the number and age of the pensioners their marital status, education and income also carry significant information. A custom microsimulation framework utilizing parallel programing technologies is also introduced. To illustrate the capabilities of the microsimulation framework I have implemented a birth and death forecast in detail. The large number of records to process and the similar steps to be carried out on each record many times make the microsimulation method perfectly suitable for parallel execution.

2. Research method

Statements and explanations of the first chapter are based on many years of practical experience as an electric engineer and a software developer.

I have used the Visual Studio development environment with the CUDA-C and the built-in C# compiler to build software in the thesis.

The classical waterfall software development methodology was applied during the implementation of the ABS method. In the first stage the basic matrix and vector operations were implemented optimized for the data transfer between different memory locations. The implemented functions have been validated by unit tests. The module test validating the implemented algorithm as a whole used random generated systems of equations with known solutions.

The development of the volume calculation algorithm followed the spiral methodology. Newer and newer iterations were necessary to optimize runtime and to filter out defects as a result of double precision variable rounding errors in increasingly higher dimensions.

Debugging Monte Carlo simulations based on random numbers in a parallel environment can be challenging, since there can be thousands of parallel threads in nondeterministic state. Debugging tools can spot the error but they can't trace back the previous states of the thread. So it can be difficult to figure out how the thread got into an undesired state. For the testing of the correctness of the algorithm various visualization tools have also been implanted to plot the spread of different variables in time and space. A separate application have been developed to carry out many calculations for the same problem in a row and to plot the spread of the results.

The volume algorithm uses a Mersenne-Twister random number generator implementation by NVidia. It’s important to point out that a vast amount of computing power is used by the random generator functions. The open source AlgLib library was ported to CUDA-C to calculate functions like chi-square or inverse chi.

The variance reduction methods introduced in the third chapter are based on orthogonal vector systems. In the first version the Gram–Schmidt method was applied for orthogonalization that was replaced by the Householder method for stability reasons.

The development of the Microsimulation Framework in the fourth chapter followed the spiral methodology in .NET environment using the C# language. The random number generator of the .NET framework is based on Knuth’s subtractive algorithm and is not thread safe. To solve this problem, several approaches are compared in terms of execution time – implementing classic locks seemed to be optimal. As in the case of the volume algorithm the majority of the computing power is used by the random generator. For querying the datasets of the entire population at the end of the simulation the task-parallel LINQ query technology is used.

3. Result of the research, contribution of the thesis

Theses related to the ABS algorithm:

T1. The observed stability of the modified Huang version of the ABS algorithm confirms the statement of Attila Gáti's thesis on the stability of the algorithm.

T2. The modified Huang class of the ABS algorithm is stable and the generated p_i vectors are orthogonal in case of being launched using $H_1=I$ matrix. Due to its low memory consumption, the algorithm suits the GPU architecture well.

T3. Parlett and Kahan have proved that their "twice is enough" technique developed for the classical Gram-Schmidt method increases the precision of the solution of the linear equation dramatically. This technique can also be applied to the modified Huang method that creates orthogonal vectors as well. The applied reprojection with the ABS matrix serves the same purpose. We have confirmed that the technique works for large (8000) dimensions and dense coefficient matrices as well. The results are underlined by tests.

In case of a dense coefficient matrix linear equation of 4000 variables the execution time was around 100 seconds using a GeForce GTX 570 graphic card.

Thesis related to the Lovász-Vempala volume algorithm:

T4. The LVD algorithm uses one point thread only. To achieve (quasi) independence it doesn't use all of the consecutive points for volume calculations. After each integral a few points are left out of the integration. These points are only used to step the point thread further. In the PLVDM implementation no points are left out, because the number of points used was so great that even if we had left out some points, sooner or later we would have got a point at (or very near to) the dropped points. The computer experimentation supported this claim. This way the PLVDM implementation of the algorithm can achieve increment in performance besides the number of the processing units. The methods introduced for variance reduction didn't meet out expectations.

Besides the volume algorithm I have developed utilities to visualize the spread of points in time and space. The visualization plays an important role in bug tracking and the deeper understanding of certain properties and behavior of the algorithm. Due to the rounding error of the applied double precision numbers the implementation described in the thesis is limited to 20 dimensions. More precise variables could break this limit, but it cannot be achieved effectively using GPUs. The implementation makes the volume calculation possible up to 20 dimension on a PC equipped with a CUDA capable GPU. The parallel implementation introduced in the thesis made experimentation possible in higher dimensions for the first time. It helps understanding the properties of the algorithm.

Theses related to the microsimulation framework:

T5. The long execution time of ordinary microsimulation service systems is hard to tolerate for the users – the well-known algorithms had to be speeded up. Using parallel programming techniques the execution time was cut down to an acceptable level.

T6. The goal is to make a microsimulation framework be suitable for various simulations. The introduced Microsimulation Framework can be parameterized both in terms of the structure of the dataset to be processed and the micromodule design and execution order.

Using our Microsimulation Framework 50 year forecasts were made using the research dataset provided by Hungarian Central Statistical Office. The execution time was cut down to 2 minutes using an ordinary PC by parallel programing techniques. To demonstrate the capabilities of the Microsimulation Framework demographic forecasts were made with the probabilities of death and birth taken into account. Using the query module of the framework demographic trees and tables were created to check the correctness of the application.

The capabilities of Microsimulation Framework are demonstrated by demographic forecasts as an example. Demographic forecasts play an important role in future pension calculations to give an estimation on the number of tax payers and future pensioners. The Microsimulation Framework is capable of calculating other factors as well. It’s customizable and can be used for the development, maintenance and testing of various simulation models.

4. Summary

Many decision supporting calculations in the economic life require massive computing workload. Monte Carlo simulations based on random numbers or modelling the dependence between random variables are good examples of computation intensive tasks, where precision and reliability of the result may depend on the invested computing work. In these cases the amount of time available is the limiting factor.

Since the speed of individual computing units are close to the physical limits progress at an order of magnitude can only be achieved by utilizing more computing units at the same time. By now processors in almost every desktop PC and mobile phone are capable of parallel program execution. A challenging task for professionals of various disciplines is to find and implement algorithms that can take advantage of this possibility.

The first step is choosing the architecture that suits the problem the best. In many cases a custom hardware design would yield the best results, but due to financial and time-to-market reasons custom hardware solutions are rare. (FPGAs can be applied as hybrid solution to build custom logical networks programmatically to accelerate calculations, but the work with FPGAs needs more like an electric engineer’s approach besides significant time and effort.) The thesis focuses on General Purpose GPUs and multicore CPUs since they are widely available to the researchers. The various architectures provide a good performance in different types of tasks.

The experience gained by the study also confirmed that creating and implementing algorithms for a parallel architecture is not only professional software development work, but also mathematical and engineering task requiring serious planning, that needs a lot of intuitive ideas as well. There is no straightforward methodology to transform single threaded algorithms to parallel architectures.

In many cases business calculations depend on the solution of linear equations with a large number of variables. Solving a large linear equation require significant computing power. The error propagation and the instability has to be taken into account as well. Operations like matrix multiplication move a lot of data in the memory but the basic arithmetic operations carried out on the data are relative simple. Therefore a GPU with a wide data bus is suitable for the problem. The modified Huang version of the ABS linear equation solver algorithm suits the CUDA architecture well due to its data movement pattern and low memory consumption. The execution time needed to solve an equation of 4096 variables was pushed to approximately 100 seconds on a GeForce GTX 570 GPU. The superior stability properties of the algorithm were also confirmed.

It’s very hard to give a precise estimate on the performance of a specified hardware for a given problem. This is especially true for the CUDA architecture, where groups of arithmetic units share a common control unit. Therefore even one thread entering a conditional branch blocks the execution of all the threads of the control unit. As the development progresses the number of conditional braches in the program can increase. This can lead to the slow erosion of the promising advantage of the GPU measured during the test project. In case of the Lovász-Vempala volume algorithm we had to go back time to time to revise the code for better performance. In some cases the original algorithm had to be modified. In higher dimensions double-precision rounding errors made workarounds necessary.

The Microsimulation Framework is designed for a multicore CPU. Threads running on different cores use locks to control access to common variables.

The microsimulation methodology follows each entity individually along the simulation steps. In our demographic simulations as an example the changes in the population are simulated individually person by person in one year simulation steps using simple algorithms. The algorithms make decisions on birth, death, marriage, divorce etc. using parameter tables with the corresponding probabilities. The decision making algorithms of the simulation step are relatively simple while the Simulation Framework handling data of the entire population and running simulation steps is rather complicated. The system is built up in a kind of "program in a program" manner. The complicated simulation framework runs the less-complicated simulation steps that can be easily modified by non-IT professionals as well. The framework is built on parallel programing technologies to make the execution time acceptable for the users.

There are profiling tools available to measure the utilization of different parts of the CPU or GPU. This way bottlenecks can be identified and the algorithm or program source code can be modified accordingly.

In order to design efficient algorithms for parallel architectures we must have a deep understanding of the underlying architecture. Similarly, architectural insight is generally necessary for the implementation.

5. References

Abaffy, J. (1979): A lineáris egyenletrendszerek általános megoldásának egy direkt módszerosztálya. Alkalmazott Matematikai Lapok, 5, 223-240

Abaffy, J./Spedicato, E./Broyden, C.G (1984): A Class of Direct Methods for Linear Equations. Numerische Mathematik, 45, 361-376

Abaffy, J./Spedicato, E. (1989): ABS projection algorithms: mathematical techniques for linear and nonlinear equations.

Csicsman, J. (1987): A mikroszimulációs rendszer számítástechnikai hátterének kialakítása., (KSH) A Háztartási Mikroszimulációs Rendszer munkálatai, Ts-3/8/8 tanulmánysorozat, 1. kötet

Csicsman, J. (2001): A BME PIKK Mikroszimulációs projektjének cé-lkitűzései és meggondolásai., (V. Pénzinformatikai Konferencia, Budapesti Műszaki és Gazdaságtudományi Egyetem, 2001. Október 15-16.)

Csicsman, J./Fényes, C. (2003): A Mikroszimulációs Szolgáltató Rendszer fejlesztése. Alma Mater sorozat - Üzlet, folyamat, monitoring 91

Csicsman, J./László, A. (2012): Microsimulation Service System. Hungarian Electronic Journal of Sciences

Deák, I. (1979): Comparison of methods for generating uniformly distributed random points in and on a hyperspere. Problems of Control and Information Theory, No. 8, 105-113

Deák, I. (1990): Random number generators and simulation, in: Mathematical Methods of Operations Research (series editor A. Prékopa). Budapest: Akadémiai Kiadó

Deák, I. (2002): Probabilities of simple n-dimensional sets in case of normal distribution. IIE Transactions (Operations Engineering), No. 34, 1-18

Deák, I. (2011): E-ciency of Monte Carlo computations in very high dimensional spaces. Central European Journal of Operations Research, 19, 177-189

Dyer, M./Frieze, M./Kannan, R. (1991): A random polynomial-time algorithm for approximating the volume of convex bodies. Journal of the Association for Computing Machinery, 38, 1-017

Gáti, A. (2013): Automatic roundoff error analysis of numerical algorithms. Ph.D thesis, Apllied Informatics Doctoral School, Óbuda University

Goetz, B./Peierls, T. (2006): Java Concurrency in Practice. Addison-Wesley 92

Hablicsek, L. (2007): Társadalmi-demográfiai előreszámítások a nyugdíjrendszer átalakításának modellezéséhez., Jelentés a Nyugdíj és Időskor Kerekasztal számára

Methuen Haque, Imran S/Pande, Vijay S (2010): Hard data on soft errors: A large-scale assessment of real-world error rates in gpgpu. In Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing. IEEE, 691-696

Kannan, R./Lovász L./Simonovits, M. (1997): Random walks and an O*(n5) volume algorithm for convex bodies. Random Structures and Algorithms, 11, 1-50

Kiss, T./Csata, I. (2007): A magyar népesség előreszámításának lehetőségei Erdélyben. Demográfia No. 7

Klevmarken, N. A./Olovsson, P. (1996): Direct and behavioral effects of income tax changes: simulations with the Swedish model MICROHUS. Amsterdam: Elsevier Science Publishers

Kovács, E. (2010): A nyugdíjreform demográfiai korlátai. Hitelintézeti Szemle, 2, 128-149

Lovász, L./Deák, I. (2012): Computational results of an O*(n4) volume algorithm. European Journal of Operational Research, 216, 152-161

Lovász, L./Simonovits, M. (1992): On the randomized complexity of volume and diameter. In 33rd IEEE Annual Symp. on Foundations of Comp. Sci., 482-491

Lovász, L./Simonovits, M. (1993): Random walks in a convex body and an improved volume algorithm. Random Structures and Algorithms, No. 4, 359-412

Lovász, L./Vempala, S. (2003): Simulated annealing in convex bodies and an O*(n4) volume algorithm. In Proc. of FOCS., 650-659 93

Lovász, L./Vempala, S. (2006): Simulated annealing in convex bodies and an O*(n4) volume algorithm. J. of Computer and System Sciences, 72, 392-417

O'Donoghue, Cathal (2001): Dynamic Microsimulation: A Methodological Survey. Brazilian Electronic Journal of Economics 4 No. 2

Parlett, B.N. (1980): The symmetric Eigenvalue Problem, Englewood Cliffs, N. J. Prentice-Hall

Romeijn, E./Smith, R.L. (1994): Simulated annealing for constrained glob-al optimization. J. of Global Optimization, 5, 101-126

Sanders, J./Kandort, E. (2010): CUDA by example: an introduction to generalpurpose GPU programming. Addison-Wesley Professional , 312 pages

Simonovits, M. (2003): How to compute the volume in high dimensions. Math. Programming Ser. B. 97, 337-374

Smith, R.L. (1996): The hit and run sampler: a globally reaching Markov chain sampler for generating arbitrary multivariate distribution. In Proc. 28th Conference on Winter Simulation., 260-264 94

Zafír, M. (1987): A háztartási mikroszimuláció. Koncepció, rend-szerleírás., (KSH) A Háztartási Mikroszimulációs Rendszer munkálatai, Ts-3/8/8 tanulmánysorozat, 1. kötet

6. References of the author

Csetényi, A./ Mohácsi, L./ Várallyai, L. (2007): Szoftverfejlesztés. HEFOP, Debrecen, ISBN : 978-963-9732-56-8 p. 157

Mohácsi, L./ Forgács, A. (2014): Gazdasági számítások párhuzamos architektúrákon. GIKOF Journal, HU ISSN 1588-9130, pp. 6-14.

Mohácsi, L./Rétallér, O. (2013): A mechanical approach of multivariate density function approximation. Proceedings of the International Conference on Modeling and Applied Simulation, ISBN 978-88-97999-17-1, pp. 179-184.

Mohácsi, L. / Deák, I. (2014): A parallel implementation of an O*(n4) volume algorithm. Central European Journal of Operations Research, DOI :10.1007/s10100-014-0354-7.