e-ISSN 1607-3274 Радіоелектроніка, інформатика, управління. 2021. № 4
p-ISSN 2313-688X Radio Electronics, Computer Science, Control. 2021. № 4

UDC 519.688: 004.75:004.421

ANALYSIS OF THE USE OF MULTITHREADED COMPUTING TECHNOLOGIES TO FACTORIZE OF NUMBERS BY A BINARY ALGORITHM

Prots'ko I. – Dr. Sc., Associate Professor, Department of Automated Control Systems, Lviv National Polytechnic University, Lviv, Ukraine.

Rykmas R. – Software Developer, LtdC "Uniservice", Lviv, Ukraine.

ABSTRACT

Context. Providing high-speed computation by computer systems of factorization of number into prime factors requires the development of effective algorithmic methods using computational technologies. Fast computation of factorization of numbers is used in such applications as, protection of information data, in algorithms of discrete transforms for transition from one to multidimensional computations and others.

Objective. The purpose of the work is to analyze the implementation of technologies of multithreaded computation of factorization of integer value by the binary algorithm of the method of trial divisions using computer systems with multi-core processors and graphics accelerators.

Method. A binary algorithm of trial divisions that uses the remainders of each digit of the binary representation of a number to perform a divisibility check on prime factors of the canonical factorization of number in parallel.

Results. The analysis and comparison of multithreaded computations of software implementations of factorization of number by binary algorithm using hyper-threading, AMP C++, CUDA technologies in computer systems with multi-core processors and graphics accelerators. The results of the process of number factorization for multithreaded computing technologies using the same parallel core function are analyzed.

Conclusions. In the study of realizations of number factorization by the binary algorithm in the multithreaded mode, the technology of hyper-threading calculations using multicore processors is most effectively performed. Heterogeneous computing using AMP C++ or CUDA technologies on computer systems and graphics accelerators requires consideration of GPU microarchitecture features for parallel computing core functions.

KEYWORDS: factorization, prime factors, multithreading, heterogeneous computation, parallel computation, remainder.

ABBREVIATIONS

AMP is an accelerated massive parallelism;
CPU is a central processing unit;
CMT is a chip multi-threading;
CS is a computer systems;
CUDA is a compute unified device architecture;
CTPL is a C++ thread pool library;
FN is a factorization of number;
HT is a hyper-threading;
GPU is a graphics processing unit;
GPGPU is a general-purpose computing on GPU;
IDE is an integrated development environment;
TD is a trial divisions.

NOMECLATURE

i is the number of binary digit ($i = 0,1,\ldots\ k–1$);
k is the bit size of a number of factorization;
M_t is a remainder;
N is an integer number of factorization;
n_i is a binary digit of the number of factorization;
p is a prime number;
p_i is a prime factor;
T is the periodicity of remainders;
s_k is a degree of a prime factor;
t is the number of the vertical ruler of M_t.

INTRODUCTION

The fast execution of factorization, as a process of decomposition of an integer number into prime factors, has demand in many applications. FN is used to move from one-dimensional to multidimensional representation of sequences of the discrete transforms, which is widely used in multidimensional data processing tools [1]. In the Good-Thomas algorithm of calculating the discrete Fourier transform of a composite size N, which uses the Chinese remainder theorem for integers, it is not necessary to perform products on return factors if the size N the sequence of the transform is decomposed into relatively prime factors. The Agarwal-Cooley method converts the calculation of a one-dimensional N-point convolution into a multidimensional one, provided that the size is decomposed into relatively prime factors [2]. FN is used to find one of the parameters of cryptosystems in asymmetric algorithms for encrypting information data [3].

Since the emergence of new approaches in encryption algorithms in the late 70's, a number of FN algorithms have been developed. These include ρ – method, (p–1) – Pollard's method, elliptic curve method, square sieve method, numerical field sieve method and others [4]. Based on these methods, software modules and modifications of classical algorithms for solving various applied problems are developed [5].

The object of study is the process of developing algorithmic and software for FN into prime factors using the technologies of parallel computations.

The subject of the study is multi-threading computations the binary algorithm of the method of TD, which

DOI 10.15588/1607-3274-2021-4-11

performs a check for divisibility by a prime number and its degree using the remainders of each bit of the number.

The purpose of the work is to parallelize the computations of FN according to the binary algorithm of the method of TD using CS with multi-core processors and graphics accelerators for efficient determination the values of prime factors.

1 PROBLEM STATEMENT

Suppose an integer N is given, which we factorize into prime factors in the form

$$N = p_1^{s1} p_2^{s2} \dots p_n^{sn}, \quad (1)$$

where $p_i < p_{i+1}$, i=1, 2,…n.

The simplest solution for factorization is the method of TD. The method is well parallelized using division tests without the remainder of the number N on a set of divisors from the set of prime numbers p_i = {2, 3, 5, 7, 9, 11,…}, i = 1, 2,…. However, the sequential execution of division operations to check for multiplicity in modern universal CS is performed using a division firmware that has the highest weight among arithmetic operations [6]. The problem of increasing the speed of calculation FN is solved in the direction of replacing the test division operation with a set of parallel computations with operations of adding numbers less than p_i and comparing the obtained sum with the corresponding prime number.

To quickly determine the values of simple decomposition factors, the possibility of parallelization of FN by the binary algorithm of the TD method using HT technologies in CS with multi-core processors and graphics accelerators is used.

2 REVIEW OF THE LITERATURE

Algorithmic means of IF, which characterized by regularity, modularity, simplicity and information independence of their components, are effectively implemented by computing means in combination with computing information technologies. The software implementation of algorithms of FN in modern CS is analyzed from the standpoint of the possibility of parallel execution using multithreaded organization [7, 8, 9].

Factorization algorithms based on ρ and (p − 1) methods of Pollard [10], Sherman-Lehman method [11], Shanks method [12], Lenstra elliptic curves method [13] are investigated in the direction of the possibility of parallelization. The main idea of these methods is to choose a random value of a number for a single parallel process, which is close to the element of the factorization. However, the execution of the algorithm by several simultaneous significant flows to find the factors of the number of the factorization does not always give the expected result.

In the known Fermat method, the main idea of the number $N = p_1 p_2$ is to find such pairs of natural numbers that satisfy a certain condition. However, in the case when the factors p_1 and p_2 are close in value to 1 and N, the algorithm will perform worse than the method of TD [14].

In the work [15] the improvement of computations of FN according to the binary algorithm of the TD method due to parallel computations and efficient use of CS computing resources is considered.

Comparative analysis of FN using CMT and CS with graphics accelerators is relevant to improve the software implementation of integer factorization algorithms.

3 MATERIALS AND METHODS

The well-known factorization algorithm is TD, which consists in checking the divisibility of the number N for a sequence of prime numbers, to a value less than or equal to the square root of the number N.

The work [16] describes a binary algorithm of the method of TD, in which the representation of the integer value of the number of the factorization in the binary number system is used to determine the divisibility of a decimal number on a prime number.

$$N = (n_{k-1} 2^{k-1} + n_{k-2} 2^{k-1} + \dots + n_1 2^1 + n_0 2^0). \quad (2)$$

The values of the digits n_i (i = 0,1,… k–1) are equal to 0 or 1.

As a result, the check for the divisibility of N by a prime number p is reduced to the determination modulo p of each weighting factor 2^i (i = 0,1,…, k–1) of the number N, which has the form

$$\begin{aligned} N \bmod p &= (n_{k-1} 2^{k-1} + n_{k-1} 2^{k-2} + \dots + n_0 2^0) \bmod p = \\ &= n_{k-1} (2^{k-1} \bmod p) + n_{k-1} (2^{k-2} \bmod p) + \dots \\ &\quad + n_1 (2^1 \bmod p) + n_0 (2^0 \bmod p). \end{aligned} \quad (3)$$

After accumulating the values of the remainders on the selected weights ($2^i \bmod p$) (i = 0,1,… k–1) for n_i (i = 0,1,…, k–1), equal to 1, compare the accumulated amount with a prime number p. In the case when the accumulated value of remainders is greater than p – again from the previously obtained accumulated value is carried out according to formula (3) the accumulation of remainders. In the case when the accumulated value of the residuals is equal to the number p – the decomposition element p is output and the transition to rechecking the divisibility of the power of this prime number is performed. In the case of comparison, when the accumulated value of the residuals is less than p, we move on to the next value from the sequence of prime numbers. Therefore, we obtain a set of prime factors of the factorization (1) of the number N.

To analyze the parallel organization, consider the work of the binary algorithm of FN using a block diagram (Fig. 1). The structure chart of FN into prime factors Pi consists the following functional blocks: 1 – the multiplexer, 2 – the block of lines of memory of the remainders, 3 – the multi-input adder, 4 – the first comparison unit, 5 – the control unit, 6 – the memory block of prime factors, 7 – the shift register, 8 – the second block of comparison.

The memory block 2 of the periods of the remainders of prime numbers and their powers contains T values of the remainders M_t (t = 0,1,…, T–1) for each of $p_i^{j,}$ which are partially shown in Table 1.

DOI 10.15588/1607-3274-2021-4-11

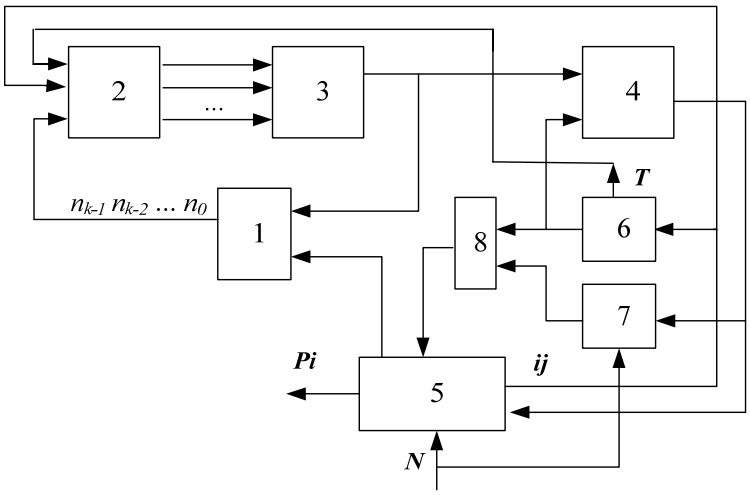

Figure 1 – The structure chart of the FN into factors p_i

Table 1 – Table of the remainders

i j	…M_t	M_9	M_8	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0	mod p_i^j	*T*
11										2	1	mod 3^1	2
12						5	7	8	4	2	1	mod 3^2	6
13	25	26	13	20	10	5	16	8	4	2	1	mod 3^3	18
…	…	…	…	…	…	…	…	…	…	…	…	…	
21								3	4	2	1	mod 5^1	4
22	24	12	6	3	14	7	16	8	4	2	1	mod 5^2	20
23	24	12	6	3	64	32	16	8	4	2	1	mod 5^3	100
…	…	…	…	…	…	…	…	…	…	…	…	…	
31									4	2	1	mod 7^1	3
32	44	22	11	30	15	32	16	8	4	2	1	mod 7^2	21
33	338	169	256	128	64	32	16	8	4	2	1	mod 7^3	147
…	…	…	…	…	…	…	…	…	…	…	…	…	
41		6	3	7	9	10	5	8	4	2	1	mod 11^1	10
42	56	28	14	7	64	32	16	8	4	2	1	mod 11^2	110
…	…	…	…	…	…	…	…	…	…	…	…		
51	10	5	9	11	12	6	3	8	4	2	1	mod 13^1	12
…	…	…	…	…	…	…	…	…	…	…	…	…	

If there is a value for the n_i binary digit (2) of the number *N* equal 1 from the memory block 2 of periods of remainders of prime numbers and their powers, the values of the remainder M_t by *ij* – sequence number of prime numbers and their powers and number *k* are read in parallel for each bit. For example, determining the value of the remainders for the number of binary digit $i = 28$ ($i = 0,1,\ldots k–1$) of the number of the FN, when checking for redivisibility by $p = 5$, that is mod p_i^j = mod 5^2. According to Table 1, this corresponds to the horizontal ruler $ij = 22$ (with the periodicity of remainders $T = 20$) and the vertical ruler M_8, where $t = (28 \bmod 20) = 8$, at the intersection of ruler is the value of the remainder $M_8 = 6$.

The maximum value of the repetition period *T* of the remainders is equal $T_{\max} = (p_i^j - 1)$. The horizontal rulers for $p_i^j = (2^j - 1)$, which are prime Mersenne numbers, have a minimum value of $T_{\min} = j$. Thus, due to the periodic repetition of the remainders, with increasing bit size of the decomposition number *N*, the number *t* of the vertical ruler M_t, $t = 0,1,2,\ldots T–1$, is determined by the formula

$$t = i \bmod T, \ (i = 0,1,\ldots k–1). \qquad (4)$$

The organization of the memory block 2 of the periods of the remainders of prime numbers and their powers allows you to read in parallel the remainders for each of the bits ($n_{k–1}$ $n_{k–2}\ldots$ n_0) of the number.

If there are two values of bits equal to 1 in the number *N*, which are control inputs for the initial multi-input adders 3 of remainders, the value of the sum of remainders ($M_k + M_{k+1}$) will be set at the output of the adders of remainders. If at the control inputs for the initial multi-input adders 3 one of the two values of the bits of the number *N* is 0 another 1, then at output of initial multi-input adders the value of the remainders will be set, which will correspond to the remainders Mk for the 1 bit of the number *N*. For the case when at the control inputs for the initial multi-input adders two values of the bits of the number N are equal to 0, then the output of the initial adder will be set the value of remainder equal to zero. Similarly, with the help of logical operations OR, in the second stage of accumulation of remainders, depending on the values of the result of the operation OR, the corresponding value of the remainder will be set. The largest delay in the parallel accumulation of remainders will be in the case when all binary bits (2) will be equal to 1.

DOI 10.15588/1607-3274-2021-4-11

The multi-input adder 3, in parallel by adding the value of the remainders, from the outputs pass result in first comparison unit 4 for comparison with the value p_i. According to the specifier at the output of first comparison unit 4, if the value is "less" – the prime number p_i or its power is not an element of the canonical factorization and there is a transition to the next ruler *ij*. According to the specifier at the output of unit 4, if the value is "greater" and "equal", one of the two directions of calculations is selected. In the case of "greater", the multiplexer 1 is switched to the output value of the multi-input adder 3, in the case of "equal" – the p_i element of the factorization is memorized and the transition to determining the its degree *j* takes place. For the specifier "equal" at the output of first comparison unit 4 in the shift register 7 provides a shift by one digit to the left of the binary value of the initial number N, thereby reducing the allowable value of the prime number or its power in the analysis of the factorization. The allowable value is compared with a prime number or its power in the second comparison block 8 and in the case of comparison greater than or equal to its output is a sign of completion of the canonical factorization.

The binary algorithm of TD can perform a divisibility check in parallel for each of the prime numbers. In this case, the memory unit 2 contains the values of the remainders only for each of the prime numbers and their powers. Otherwise, it is possible to divide the memory unit 2 into parts for groups of prime numbers and their powers, which corresponds to coarse-grained parallelism.

4 EXPERIMENTS

The computation of FN on prime factors p_i based on binary algorithm of TD is implemented in the IDE Visual C ++ 2019 using technologies HT (Intel) and AMP C ++ (MS), CUDA (NVIDIA) of GPGPU.

Software implementation of parallel computation of FN using multi-threading technology CMT is achieved by creating threads and proportional distribution of their performance between the cores of the microprocessor in the CS. For efficient use of computing resources, a pool of threads is used and the organization of interaction between them is entrusted to the function of the CTPL library [17], which is an add-on to the standard STL library and has the ability to work with system threads.

Software implementations of parallel computation of FN using AMP C ++, CUDA technologies perform a similar algorithm in the kernel function as in CMT technology.

The developed programs are implemented to perform the following actions:

1. Enter the number *N* of factorization.
2. Construction of a sequence of remainders on the prime numbers p_i by the number *T*.
3. Analysis for the divisibility of the number *N* prime number p_i.
4. Output to the file of the factors p_i for number *N*.
5. Go to the next prime p_{i+1} with the constraint of the search set of prime numbers.

DOI 10.15588/1607-3274-2021-4-11

The configuration of the test system is as follows:
CPU: Intel Core i5-9300H (4 cores/8 threads) @2,4 GHz;
GPU: NVIDIA GeForce GTX 1650, 4.0 GB, 896 32-bit core CUDA @1,45GHz;
Memory: 8.0 GB, DDR4-2667 MHz.

The chosen evaluation method involves determining the time spent on computing of FN. The whole process of parallel FN includes both the main part of multithreaded computations, and other actions performed by the program to allocate and free memory, output.

5 RESULTS

Testing the effectiveness of technologies to FN is determined by the computation time. To compare and analyze the efficiency of the use of technologies for computing the FN by the binary algorithm of the method of TD, a number of tests were performed. The time of execution of factoring computations for the developed programs threadsSplitNumber and two programs ampSplitNumber and cudaSplitNumber was determined using GPGPU.

The test results are presented in Table 2, which contains data on the average time (microseconds) of FN in 32-bit mode, which is computed 5000 times.

Table 2– The average time of FN

N	Average time (μs)		
	Threads SplitNumber	Cuda SplitNumber	Amp SplitNumber
15361*14537 = 223302857	126	132	212
15107*18959 = 286413613	83	125	226
30829*22433 = 691586957	75	122	200
29717*52433 = 1558151461	138	129	219

The values of number *N* to FN are chosen to be equal to the product of two prime numbers. Created threads for analysis for divisibility are implemented by computing cores in 32-bit mode. Table 1 shows the time relationship (μsec) for the number of threads 10 in the program threadsSplitNumber [15].

6 DISCUSSION

Almost all modern operating systems support of threads control. The application program implements threads control with the help of special libraries that allow to achieve hardware acceleration of the canonical FN.

Comparison of the results of Table 2 shows the increase in execution time in the binary algorithm of TD in the order of threadsSplitNumber, cudaSplitNumber, ampSplitNumber.

In the program threadsSplitNumber with the optimal number of threads 10 we have the shortest execution time. The threads will be divided equally to process the Table of the remainders (Table 1). Each thread processes the table of the remainders and the binary form of the number *N* to decide on the divisibility of the corresponding prime number and its power. To organize the use of the pool of

threads and the interaction between them, the function of the CTPL library is used [17], which is an add-on to the standard STL library, which has the ability to work with system threads.

CudaSplitNumber uses NVIDIA's CUDA technology [18]. This technology ushered a new era of improved performance for many applications as programming GPUs became simpler: archaic terms such as texels, fragments, and pixels were superseded with threads, vector processing, data caches and shared memory. Accordance GCGPU, every problem that can be parallelized on GPU always passes the following phases:

1. Transferring data from CPU to GPU.
2. Calling kernel for each thread.
3. Transferring data from GPU to CPU.

Time execution of parallel FN (Table 2) includes 2 and 3 phases. Therefore, the main part of multithreaded computing is the actions performed by the program to select, free of memory and transferring data.

The ampSplitNumber program uses technology for GPGPU called C++ AMP. It accelerates the execution of C ++ code by taking advantage of the GPUs present on video cards with DirectX11 support. But unlike CUDA, that are more oriented in C code, C++ AMP looks like STL library. Both languages extend C ++ with special keywords. For CUDA the syntax is "<<< >>>" and C++ AMP is using the keyword *restrict* (*amp*) to run kernels. It is obvious that code written on C++ AMP is less cluttered and hence easier to read than on CUDA. However, the results of running cudaSplitNumber are better than ampSplitNumber (Table 2). This is because the time to phase call a C++ AMP kernel is much longer than CUDA and a C++ AMP technology has some performance lags compare to 5-years old CUDA technology.

Time execution of parallel FN (Table 2) using GPGPU includes 2 and 3 phases. Therefore, the main part of multithreaded computing is the actions performed by the program to select, free of memory and transferring data.

CONCLUSIONS

Virtually any modern software code execution platform, whether a full-fledged operating system or a virtual machine, contains a set of APIs designed to manage threads and create parallel programs. The FN of binary algorithm of TD with use of parallel technologies of multithreaded computations is considered in the work. The software solution is implemented in the IDE Visual C ++ 2019 with the help of special libraries that allow you to perform parallelization using CS with multi-core processors and graphics accelerators.

The scientific novelty lies in the study of the use of parallel technologies for the binary algorithm of TD, which allows to take into account the peculiarities of the organization of multithreaded computations in problems with an average level of parallelism.

The practical significance of the application of the developed software code of FN in CS with multicore processors and graphics accelerators has shown that the efficiency of computations depends not only on hardware resources, but also on the selected technologies of parallelisation. The obtained results are important for improving the organization of efficient computation of FN for information data protection systems, discrete transforms and other applications.

The direction of further research will be the development of software with the expansion of the bits number of binary representation *N*, which will use the microarchitectural features of specific types of GPU memory, the use of atomic operations and other innovations.

DOI 10.15588/1607-3274-2021-4-11

ACKNOWLEDGEMENTS

The work is supported by the state budget scientific research project of Lviv National Polytechnic University "Experimental system of neural network cryptographic protection in real-time data transmission using codes imitating barker codes" (state registration number 0121U109503).

The authors are grateful to Myroslav Mishchuk of the student of project of Lviv National Polytechnic University for participation in testing and discussing the results.

REFERENCES

1. Knuth D. E. The art of computer programming. 3-ed. Vol. 1: Seminumerical Algorithms, Menlo Park. California, Addison-Wesley, 1998, 762 p.
2. McClellan J. H., Rader C. M.. Number Theory in Digital Signal Processing. Englewood Clis, NJ, Prentice-Hall, 1979, 276 p.
3. Stalling W. Cryptography and Network Security: Principles and Practice. 6-ed. Upper Saddle River, NJ, Prentice Hall, 2013, 672 p.
4. Samuel S., Wagstaff Jr. The Joy of Factoring, Providence, RI: American Mathematical Society, 2013, pp. 138–141.
5. Brent R. P. Some parallel algorithms for integer factorisation, *European Conference on Parallel Processing, Toulouse, 1999, Part of the Lecture Notes in Computer Science book series*. Berlin, Springing-Verlag, 1999, Vol. 1685, pp. 1–22.
6. Hindriksen V. How expensive is an operation on a CPU [Electronic resource], Access mode: https://streamcomputing.eu/blog/2012-07-16/how-expensive-is-an-operation-on-a-cpu
7. Mishra M., Chaturvedi U., Saibal K., Pal S. K. A multithreaded bound varying chaotic firefly algorithm for prime factorization, *Advance Computing Conference: 4th IEEE International conference, Gurgaon, India. February 2014: proceedings*. Published by IEEE Computer Society, 2014, pp. 1322–1325. DOI: 10.1109/IAdCC. 2014.6779518
8. Makarenko A. V., Pykhteyev A. V., Yefimov S. S. Parallel'naya realizatsiya i sravnitel'nyy analiz algoritmov faktorizatsii s raspredelennoy pamyat'yu [Electronic resource]. Access mode: http://cyberleninka.ru/article/n/parallelnaya-realizatsiya-i-sravnitelnyy-analiz-algoritmov-faktorizatsii-v-sistemah-s-raspredelyonnoy-pamyatyu
9. Huang L. New prime factorization algorithm and its parallel computing strategy, *Information Technologies in Education and Learning: International Conference (ICITEL 2015), Atlantis, 2015, proceedings*. Published by Atlantis Press, 2015, pp. 1–4.

10. Koundinya A. K., Harish G., Srinath N. K. et al. Performance Analysis of parallel Pollard's Rho algorithm, *International Journal of Computer Science & Information Technology (IJCSIT),* 2011, Vol. 5, No. 2, pp. 157–163. DOI: 10.5121/ijcsit.2013.5214
11. Kim D. K., Choi P., Lee M.-K. et al. Design and analysis of efficient parallel hardware prime generators, *Journal of semiconductor technology and science,* 2016, Vol. 16, No. 5, pp. 564–581. DOI: 10.5573/ JSTS. 2016.16.5.564
12. Gower J. E., Wagstaff S. S. Square form factorization, Mathematics of Computation, 2008, Vol. 77, No. 261, pp. 551–588.
13. Meulenaer de G. Gosset F., Dormale de G. M., Quisquater J. J. Integer factorization based on elliptic curve method: towards better exploitation of reconfigurable hard-ware, *Field-Programmable Custom Computing Machines: IEEE Symposium FCCM, Napa, California, 23–25 April 2007, proceedings*. Published by IEEE Computer Society, 2007, pp. 197–206.
14. Ishmukhametov S. T. Metody faktorizatsii natural'nykh chisel: uchebnoye posobiye. Kazan', Kazan'skiy universitet, 2011, 190 p.
15. Prots'ko I. O., Gryschuk O. V. Computation factorization of number at chip multithreading mode, *Radio, Electronics, Computer Science, Control,* 2019, No. 3, pp. 117–122. DOI 10.15588/1607-3274-2019-3-13
16. Prots'ko I.O. Binarno-bitovi alhorytmy: prohramuvannya i zastosuvannya. L'viv, Triada plyus, 2020, 120 p.
17. Library CTPL [Electronic resource]. Access mode: https://github.com/vit-vit/CTPL.
18. Kirk D. B., Hwu Wen-mei W. Programming Massively Parallel Processors: A Hands-on Approach, Second Edition, Waltham. Morgan Kaufmann, 2012, 518 p.
19. C++ AMP: Language and Programming Model, Version 1.0, August 2012 [Electronic resource]. Access mode: https://download.microsoft.com

Received 21.06.2021.
Accepted 01.10.2021.

УДК 519.688: 004.75:004.421

АНАЛІЗ ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ БАГАТОПОТОКОВИХ ОБЧИСЛЕНЬ ДЛЯ ФАКТОРИЗАЦІЇ ЧИСЕЛ ЗА БІНАРНИМ АЛГОРИТМОМ

Процько І. О. – д-р техн. наук, доцент кафедри автоматизованих систем управління Національного університету «Львівська політехніка», Львів, Україна.
Рикмас Р. В. – розробник, ТзОВ «Uniservice», Львів, Україна.

АНОТАЦІЯ

Актуальність. Забезпечення високої швидкодії обчислення комп'ютерними системами розкладу цілочисельного значення на прості множники вимагає розробки ефективних алгоритмічних методів з використанням обчислювальних технологій. Швидке обчислення факторизації чисел використовується в таких застосуваннях, як захист інформаційних даних, в алгоритмах дискретних перетворень для переходу від одного до багатовимірних обчислень та інших.

Метою роботи є аналіз впровадження технологій багатопотокового обчислення факторизації цілочисельного значення за бінарним алгоритмом методу пробних ділень з використанням комп'ютерних систем з багатоядерними процесорами та графічними прискорювачами.

Метод. Бінарний алгоритм пробних ділень, що використовує залишки кожного розряду двійкового подання числа, для здійснення паралельної перевірки подільності на прості множники для канонічного розкладання числа.

Результати. Проведено аналіз та порівняння програмних реалізацій багатопотокових обчислень факторизації числа за двійковим алгоритмом із використанням технологій гіперпоточності, AMP C++, CUDA в комп'ютерних системах з багатоядерними процесорами та графічними прискорювачами. Проаналізовано результати процесу факторизації чисел для багатопоточних обчислювальних технологій, що використовують однотиповий алгоритм для функції паралельного ядра.

Висновки. При дослідженні реалізації розкладання чисел за бінарним алгоритмом у багатопотоковому режимі найбільш ефективно виконується технологія гіперпоточних обчислень із використанням багатоядерних процесорів. Гетерогенні обчислення за допомогою технологій AMP C++ або CUDA на комп'ютерних системах та графічних прискорювачах вимагають врахування особливостей мікроархітектури графічного процесора для паралельного виконання функцій ядра.

КЛЮЧОВІ СЛОВА: розкладання на множники, прості множники, багатопотоковість, гетерогенні обчислення, паралельні обчислення, залишок.

УДК 519.688: 004.75:004.421

АНАЛИЗ ПРИМЕНЕНИЯ ТЕХНОЛОГИЙ МНОГОПОТОЧНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ФАКТОРИЗАЦИИ ЧИСЕЛ ЗА БИНАРНЫМ АЛГОРИТМОМ

Процько И. Е. – д-р техн. наук, доцент кафедры автоматизированных систем управления Национального университета «Львовская политехника», Украина.
Рыкмас Р. В. – разработчик, ТзОВ «Uniservice», Львов, Украина.

АННОТАЦИЯ

Актуальность. Обеспечение высокого быстродействия вычисления компьютерными системами разложения целочисленного значения на простые множители требует разработки эффективных алгоритмических методов с использованием вычислительных технологий. Быстрое вычисление факторизации чисел используется в таких приложениях, как защита ин-

DOI 10.15588/1607-3274-2021-4-11

формационных данных, в алгоритмах дискретных преобразований для перехода от одного к многомерным вычислений и других.

Целью работы является анализ внедрения технологий многопоточного вычисления факторизации целочисленного значения за бинарным алгоритмом метода пробных делений с использованием компьютерных систем с многоядерными процессорами и графическими ускорителями.

Метод. Бинарный алгоритм пробных делений, что использует остатки каждого разряда двоичного представления числа, для осуществления параллельной проверки делимости на простые множители для канонического разложения числа.

Результаты. Проведен анализ и сравнение программных реализаций многопоточных вычислений факторизации числа за двоичным алгоритмом с использованием технологий гиперпоточности, AMP C++, CUDA в компьютерных системах с многоядерными процессорами и графическими ускорителями. Проанализированы результаты процесса факторизации чисел для многопоточных вычислительных технологий, использующих однотипные алгоритм для функции параллельного ядра.

Выводы. При исследовании реализации разложения чисел за бинарным алгоритмом в многопоточном режиме наиболее эффективно выполняется технология гиперпоточних вычислений с использованием многоядерных процессоров. Гетерогенные вычисления с помощью технологий AMP C++ или CUDA на компьютерных системах и графических ускорителях требуют учета особенностей микроархитектуры графического процессора для выполнения параллельных вычислений функций ядра.

КЛЮЧЕВЫЕ СЛОВА: разложение на множители, простые множители, многопоточность, гетерогенные вычисления, параллельные вычисления, остаток.

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Knuth D. E. The art of computer programming. 3-ed. / D. E. Knuth. – Vol. 1: Seminumerical Algorithms. – Menlo Park, California : Addison-Wesley, 1998. – 762 p.
2. McClellan J. H. Number Theory in Digital Signal Processing /J. H. McClellan, C. M. Rader. – Englewood Clis, NJ: Prentice-Hall, 1979. – 276 p.
3. Stalling W. Cryptography and Network Security: Principles and Practice. 6-ed./ W. Stalling. – Upper Saddle River, NJ : Prentice Hall, 2013. – 672 p.
4. Samuel S. The Joy of Factoring / S. Samuel, Jr. Wagstaff. – Providence, RI: American Mathematical Society, 2013. – P. 138–141.
5. Brent R. P. Some parallel algorithms for integer factorisation / R. P. Brent // European Conference on Parallel Processing, Toulouse, 1999: Part of the Lecture Notes in Computer Science book series, Berlin : Springing-Verlag, 1999. – Vol. 1685. – P. 1–22.
6. Hindriksen V. How expensive is an operation on a CPU [Electronic resource] / V. Hindriksen. – Access mode: https://streamcomputing.eu/blog/2012-07-16/how-expensive-is-an-operation-on-a-cpu
7. A multithreaded bound varying chaotic firefly algorithm for prime factorization / [M. Mishra, U. Chaturvedi, K. Saibal, S. K. Pal] // Advance Computing Conference: 4th IEEE International conference, Gurgaon, India. February 2014: proceedings. – Published by IEEE Computer Society, 2014. – P. 1322–1325. DOI: 10.1109/IAdCC. 2014.6779518
8. Макаренко А. В. Параллельная реализация и сравнительный анализ алгоритмов факторизации с распределенной памятью [Електронний ресурс] / А. В. Макаренко, А. В. Пыхтеев, С. С. Ефимов. – Режим доступу: http://cyberleninka.ru/article/n/parallelnaya-realizatsiya-i-sravnitelnyy-analiz-algoritmov-faktori-zatsii-v-sistemah-s-raspredelyonnoy-pamyatyu
9. Huang L. New prime factorization algorithm and its parallel computing strategy / L. Huang // Information Technologies in Education and Learning: International Conference (ICITEL 2015), Atlantis, 2015: proceedings. – Published by Atlantis Press, 2015. – P. 1–4.
10. Koundinya A. K. Performance Analysis of parallel Pollard's Rho algorithm / A. K. Koundinya, G. Harish, N. K. Srinath et al. // International Journal of Computer Science & Information Technology (IJCSIT). – 2011. – Vol. 5, No. 2. – P. 157–163. DOI: 10.5121/ijcsit.2013.5214
11. Kim D. K. Design and analysis of efficient parallel hardware prime generators / [D. K. Kim, P. Choi, M.-K. Lee et al.] // Journal of semiconductor technology and science. – 2016. – Vol. 16, No. 5. – P. 564–581. DOI: 10.5573/ JSTS. 2016.16.5.564
12. Gower J. E. Square form factorization / J. E. Gower, S. S. Wagstaff // Mathematics of Computation. – 2008. – Vol. 77, No. 261. – P. 551–588.
13. Meulenaer G. Integer factorization based on elliptic curve method: towards better exploitation of reconfigurable hardware / [G. de Meulenaer, F. Gosset, G. M. de Dormale, J. J. Quisquater] // Field-Programmable Custom Computing Machines: IEEE Symposium FCCM, Napa, California, 23–25 April 2007: proceedings. – Published by IEEE Computer Society, 2007. – P. 197–206.
14. Ишмухаметов Ш. Т. Методы факторизации натуральных чисел: учебное пособие / Ш. Т. Ишмухаметов. – Казань : Казанький университет, 2011. – 190 с.
15. Процько І. О. Обчислення факторизації числа в мультипотоковому режимі на кристалі / І. О. Процько, О. В. Грищук // Радіоелектроніка, інформатика, управління. – 2019. – № 3. – С. 117–122. DOI 10.15588/1607-3274-2019-3-13
16. Процько І. О. Бінарно-бітові алгоритми: програмування і застосування : навчальний посібник / І. О. Процько. – Львів : Тріада плюс, 2020. – 120 с.
17. Бібліотека CTPL [Електронний ресурс]. – Режим доступу: https://github.com/vit-vit/CTPL
18. Kirk D. B. Programming Massively Parallel Processors: A Hands-on Approach, Second Edition / David B. Kirk, Wenmei W. Hwu. – Waltham : Morgan Kaufmann, 2012. – 518 p.
19. C++ AMP : Language and Programming Model — Version 1.0 : August 2012 [Електронний ресурс]. – Режим доступу: https://download.microsoft.com

DOI 10.15588/1607-3274-2021-4-11