Parallelizing by Smart Tiling

Sergii Sushko
Dept. of Modelling and Econometrics
Pukhov Institute for Modeling in Energy Engineering of NASU
Kyiv, Ukraine
sergii.sushko@gmail.com

Alexander Chemeris
Dept. of Modelling and Econometrics
Pukhov Institute for Modeling in Energy Engineering of NASU
Kyiv, Ukraine
a.a.chemeris@gmail.com

Svetlana Reznikova
Dept. of Modelling and Econometrics
Pukhov Institute for Modeling in Energy Engineering of NASU
Kyiv, Ukraine
svetlana.reznikova@gmail.com

Abstract. **The paper is devoted to the methods of automatic parallelization and software optimization. The authors focus on parallelizing of computational loops. The problem of quickly choosing a partitioning method and determining its parameters is an urgent task. Its solution provides a reduction in software's execution time for computing systems with multiprocessor architecture. To build an automated system for parallelizing programs, the authors propose to use Discrete Particle Swarm Optimization Method as an optimization method which allows to find a local or global minimum of program execution time regarding complicated relationship between tile sizes and execution time. The paper proposes an approach to optimizing the process of partitioning the iterative space of loop operators using the methods of Swarm Intellect. It uses partitioning by rectangular parts but has no fundamental restrictions on its use for other types of partitioning (triangles, parallelograms, rhombuses, etc.). Proposed method is called Smart Tiling Method.**

Keywords: **parallel programs, program parallelization, loop parallelization, particle swarm optimization, tiling, smart tiling**

I. INTRODUCTION

Hardware and software are constantly being improved during entire period of development of computer technologies. At the same time with an extensive type of development also an intensive type of development is used. Considerable attention is paid to more efficient usage of available resources. For example, more efficient usage of resources of data center can have a significant impact on energy consumption. Thereby, an amount of electricity which is required for air condition and as a result an amount of financial costs will reduce. Effective usage of mobile devices allows to work longer without recharging and to use a hardware with better computing capabilities. Effective usage of embedded systems allows to expand of functionality and service capabilities of devices.

Computational efficiency is a multifactorial characteristic. First, it is a measure of ability of hardware and compiler to implement high-level source code [1]. Second, computational efficiency can be defined as a fraction of actual processing performance relatively to peak processing performance available for given hardware. Third, the computational efficiency is based on efficiency of algorithms used by software. These factors can be applied to all computing systems – PCs, servers [2, 3], mobile devices [4], FPGAs [5], controllers, embedded systems [6] and so on.

Computational efficiency is also considered in context of integration of computational hardware. For example, estimates of effectiveness of various options for creating of complex of computing facilities as system with predefined composition of equipment are given in [7]. The author considers such indicator as a coefficient of decrease of real performance which characterizes cost of computing performance for organizing of common work of computers or processors.

$$K_k = \frac{P}{\sum_{i=1}^{N} V_i}, \quad (1)$$

where V_i is effective performance of *i*-th computing device for individual functioning and specific class of problems; P — same parameter for whole system; N – number of computers or processors.

Efficiency of implementation of a certain algorithm or class of algorithms of computational system is almost completely determined by relation of structure of chosen algorithm and by structure of a system. For practical usage of computing systems, a flexibility of logical structure of algorithm and its ability to transform become especially important.

II. THE PROBLEM OF LOOP PARALLELIZATION

In operator of computational loop which consists of n nested loops, a set of distance vectors is approximated by an integer number l from the interval $[1, n] \cup \{\infty\}$, which is defined as the largest integer such that the first l–1 components of distance vectors are zeros. Dependence at level $l \leq n$ means that dependence is found on level l of nested loops, that is, on given

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

iteration of *l*–1 outer loops. In this case, dependence between iteration dependence and such dependences are called loop-carried on l level. If $l = \infty$, then dependence occurs inside of body of loop between two different operators. Such dependencies are called loop-independent. Value l is called level of dependence. Analysis of computational loops, each of which is defined by its own index variable, yields a set of index variables which creates an index vector of nested loops $I = (i_1, i_2 ...., i_n)$.

For any loop in which index of loop *I* changes from *L* to *U* with a step *S*, an iteration number *i* is equal to the value $(I - L + S)/S$, where *I* is index value for this iteration [7].

For *n* nested loops, iteration vector *I* for innermost loop is a vector containing an integer number of iterations for each loop in order of nested loop. In other words, iteration number of a multidimensional nested loop is determined in accordance with the form $I = (i1, i2 \ldots., in)$, where i_k, $1 \le k \le n$, is a loop iteration number for nested level *k*.

Definition. Iteration space is a set of all integer vectors $I = (i_1, i_2 ...., i_n)$ which satisfies the inequality:

$$L_i \le x_i \le \mathrm{U}_i, i = 1..n., \qquad (2)$$

Inequality (2) defines boundaries of loop which limit iteration space by a convex polytope.

Thus, a model for representing of iteration space is defined, which consists of constraints that define boundaries of space, a set of nodes that corresponds to loop iteration, and a set of dependencies between iterations. Such model of iteration space which represents it as a convex polyhedron, is called a polyhedral model [8].

Task of parallelization is to divide iteration space into separate blocks, under which following conditions should be met, if possible:

1) compliance with structure of computing system, for example, take into account architecture and number of processors;

2) ensure of same loading on system processors;

3) minimize connections between blocks of space, ensuring their parallel execution.

Considering variety of approaches for modification of computational loops, tiling method and its modifications should be noted. This method introduces additional loops which break entire iteration space into small blocks so called tiles. Calculations are performed first for each tile. Tile size is chosen in each case individually. This approach improves data locality which leads to more efficient use of cache and internal registers of processor and this can reduce loading on memory bus, faster computations and lower power consumption.

In general, implementation of parallelization and optimization process is performed by converters, algorithms, methods that analyze a program code and convert it to semantically equivalent version which is more efficient by some set of optimization goals. It was shown in [1] some code optimization problems are NP-complete or even such which cannot be solved. In practice, many of them are solved by heuristic methods and give a result in a satisfactory processing time for program code.

Conversion of software to minimize or maximize some goal function, which in general form is represented by expression (3):

$$\underset{x}{argmin}\, f(x) \in \{x \mid \forall y : f(y) \le f(x)\}. \quad (3)$$

Then, considering parallelization and software optimization as a process of directed application or selection of a finite number of methods with a finite set of parameters, it can be described by the expression (4):

$$F_i = argmin(f(\overrightarrow{\boldsymbol{M}_l}, \overrightarrow{\boldsymbol{P}_k})), \qquad (4)$$

where *F* is a target function, which can be one or several parameters requiring improvement – reduction of execution time, power consumption, program memory size, data, etc.; $\boldsymbol{M}$ is a set of possible methods for parallelization and optimization; $\boldsymbol{P}$ is a set of parameter values for these methods.

Expression (4) means that for tiling of iteration space, it is necessary to use several appropriate methods with their own optimization parameters. The choice of methods can be preselected (defined explicitly or set by default) or determined by evaluating a code by using various metrics.

Thus, process of dividing of iteration space into separate tiles is complicated by fact that a choice of a set of specific methods and their specific parameters is not predetermined and requires additional experiments to test effectiveness and their combinations. Through architectural features and limitations, selection of methods and their parameters is unique for each practical case. The authors propose Smart Tiling Method which is based on an optimization method by using swarm intelligence.

III. Smart Tiling Method for Loop Parallelization

The development is based on a typical process that is used in Pluto software package [8]. Its main stages are shown in Fig. 1. This package includes several software modules. These modules are used one by one. First one

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

creates a polyhedral model based on source code of C/C++ languages, then Pluto package itself modifies this polyhedral model according to the specified methods of tiling and parallelization, then other software modules simplify resulting model without changing of lexicographic sequence, and at the last stage program modules create a source code of C or C++ programming languages according to obtained model.

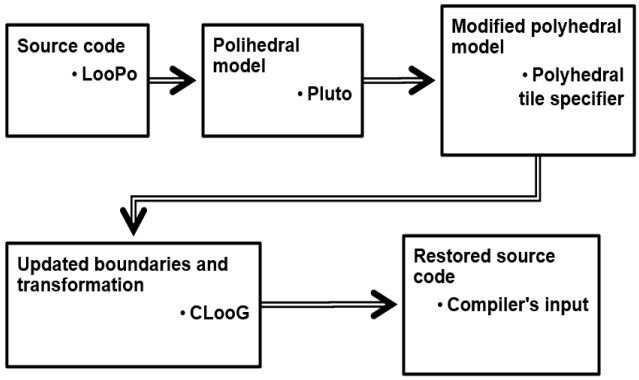

Fig. 1. Parallelization and optimization in Pluto

To efficiently divide of iteration space of loop operators, it is necessary to choose appropriate method that will give the maximum computational performance and find corresponding parameters of this method. For the approach which is shown in Fig. 1 it depends on efficiency of partitioning methods implemented and on experience of software developer. Nevertheless, automation of this process brings its additional effects on inefficiency of transformations. Therefore, it's necessary to investigate and develop some approaches to implementation of parallelization and program optimization.

Process of choosing of parameters for tiling of iterative space is performed on the stage of modifying of polyhedral model. The experiments presented in [9] show complex dependence of computational efficiency on tiling method and size of tiles into which space is partitioned. After determination of minimum of this function a method of tiling and tiles' sizes can be obtained.

The authors, in order to find the minimum of the goal function, proposed to use optimization method of swarm intelligence, namely Particle Swarm Optimization Method [10]. By using of discrete version of the method, the authors propose an intelligent method for partitioning of iterative space of loop operators in software programs. This will reduce searching time for optimal solution when parallelization algorithms for multiprocessor computing systems. The authors performed the experiments for programs written in C/C++, but there are no significant restrictions to use for other languages.

Smart Tiling Method, as shown in Fig. 2, uses Discrete Particle Swarm Optimization Method to find the best tiling block sizes by iterative size choice. Next, effectiveness of chosen tile sizes is evaluated by compiling and measuring of execution time of computer program. According to result of evaluation, parameters of particle swarm are modified. This algorithm is quite flexible and allows to use different versions of tiling methods. There are possible two approaches for tiling – standalone and with parallelization together.

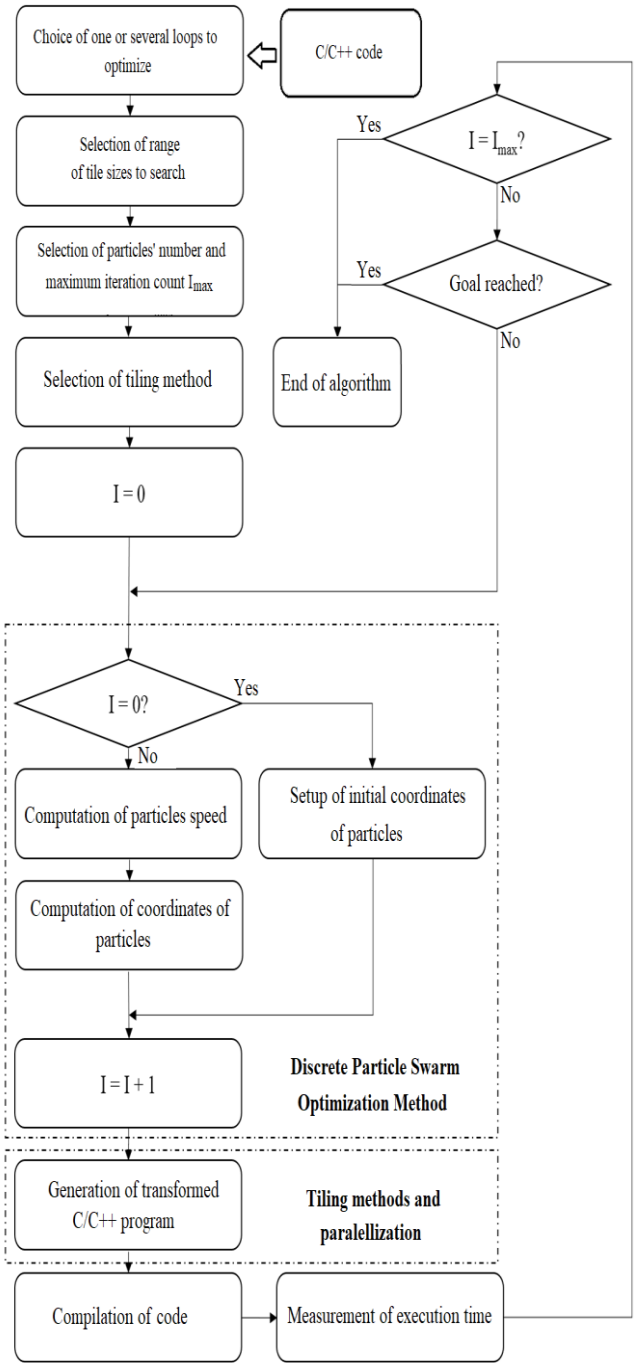

Fig. 2. Smart Tiling Method based on Discrete Particle Swarm Optimization Method

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

IV. The Experiments

For the evaluation of impact of various methods of tiling of iteration space on execution time, a lot of experiments were performed on various hardware platforms. The first experiments were performed on a single board computer Raspberry Pi 3 (quad-core ARM Cortex-A53 CPU, 1.2 GHz, 32KB L1 cache, L2 512KB, 1GB LPDDR2 900 MHz memory). Another series of experiments was verified on the desktop PC with Intel processor (Intel® Core ™ i5-4670K quad-core CPU, 3.4 GHz, 4x64KB L1 cache, L2 4x25.26KB, L3 6MB, 16GB DDR3 1333 MHz memory). The package of functions Polybench 4.1 [11] was chosen as set of the test programs. This package contains about 30 small test programs written in C, in which various classes of algorithms are implemented – operations with vectors, matrices, linear algebra, signal processing. Some experimental results are given in [9].

A series of experiments was performed to test an efficiency of optimizing on execution time of test programs by using of Smart Tiling Method. Smart Tiling Method was used for several tiling methods of Pluto package, namely “tile”, “innerpar”, “parallel”. The obtained data were normalized taking into account the initial execution time of the test programs and are shown in Fig. 3 and 4 as diagrams.

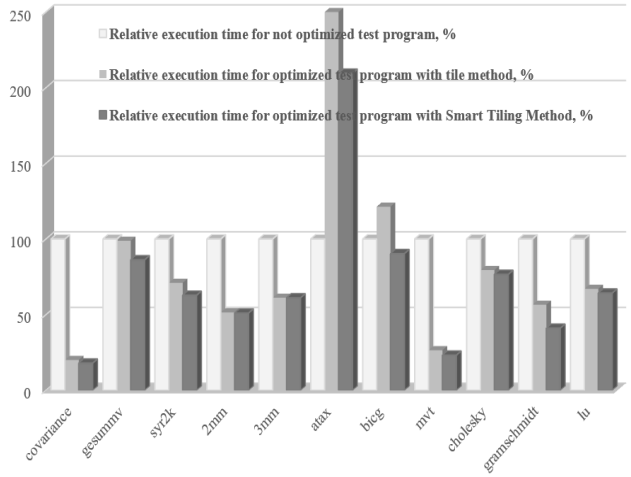

Fig. 3. Relative execution time with “tile” method in comparison with Smart Tiling Method

To accelerate preparation of software for computing systems with multiprocessor architecture, usage of automated parallelization and program optimization systems is an urgent and promising task. Potential effect on microprocessor control systems, IoT systems, embedded systems, mobile devices, and so on should be emphasized.

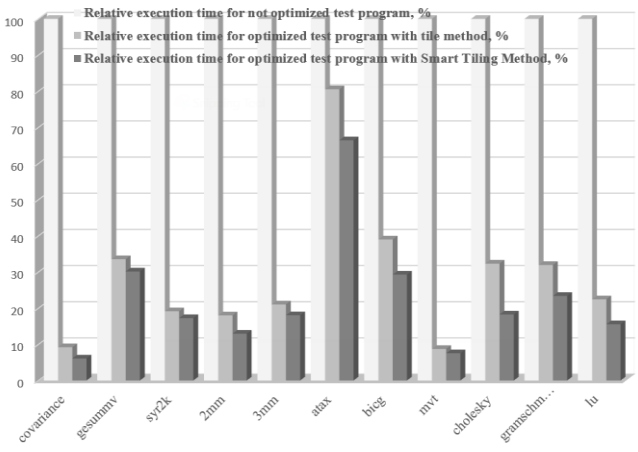

Fig. 4. Relative execution time with “tile”, “innerpar” and “parallel” methods in comparison with Smart Tiling Method

Polyhedral model for representing of iteration space of loop operators is formed by a system of constraints on loop variables and is N-dimensional convex polyhedron. This is of many approaches to analysis and parallelization of program loop operators. It is promising for building an effective system for parallelizing of loop parts of algorithms.

V. Conclusions

Analysis of methods for dividing of iteration space into parts or tiling method shows a complex dependence of program execution time and, thus, computational efficiency on parameters of tiling. Usage of optimization methods to find minimum of goal function makes it possible to speed up a process of program parallelization. For this Smart Tiling Method based on Discrete Particle Swarm Optimization Method is proposed. The method allows to speed up searching a solution by choosing the parameters of tiling method. This allows to improve performance of programs due to better choosing of parameters and thus faster program execution. Experiments show the effectiveness of the method for 2-dimensional case. For certain classes of problems, it’s possible to get up to 15 times improvement in performance.

The developed method for optimizing of partitioning of iteration space of program loop operators, which the authors investigated for the 2-dimensional case of tiling of iteration space into rectangular parts, has no fundamental restrictions on its to use for other types of partition (triangles, parallelograms, rhombuses, etc.).

References

[1] V. V. Voevodin, V. V. Voevodin, Parallel computing, St. Petersburg, BHV-Petersburg, 2002, 608 p. (in rus.)

[2] O. O. Druzhinina, R. N. Kvetny, Increasing the efficiency of web servers with the use of time series forecasting technology based on neural networks, Information technology and computer engineering, 2013, № 1, pp. 15-21. (in ukr.)

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

[3] G. M. Lutskiy, I. S. Raizin, Increasing the efficiency of clusters based on Infiniband, Bulletin of the University "Ukraine", Informatics, computer science and cybernetics, 2011, № 8, pp. 133. (in rus.)

[4] P. Havinga, G. Smit, Low power systems design techniques for mobile computers. Centre for Telematics and Information Technology University of Twente, Enschede, 1997.

[5] I. A. Klimenko, Methods and means to increase the efficiency of information processing in reconfigured computer systems based on FPGA, Dr. Tech. Science thesis: 05.13.05. Kyiv, 2017, 377 p. (in ukr.)

[6] N. V. Borisova, L. V. Shabanova-Kushnarenko, Effective resource management of embedded systems for real-time computing, Information processing systems, 2018, № 1(152), pp. 87-93. (in ukr)

[7] V. K. Dushin, Theoretical foundations of information processes and systems, Moscow, Dashkov and Co., 2003, pp. 348 (in rus).

[8] Uday Bondhugula, Albert Hartono, J. Ramanujam, Ponnuswamy Sadayappan A practical automatic polyhedral parallelizer and locality optimizer, In Conference: PLDI '08: Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation, May 2008, ACM SIGPLAN Notices 43(6), DOI: 10.1145/ 1375581.1375595

[9] S. Sushko, A. Chemeris, Dependency between Tiles' Sizes and Program Execution Time, Proceedings: Reconfigurable Ubiquitous Computing (RUC-2018), 12 October 2018, Dzivnuv, Poland.

[10] A. Chemeris, S. Sushko, Usage of Discrete Particle Swarm Optimization Method for the Searching of Optimal Tile Size, 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T-2019), October 8-11, 2019, Kyiv, P. 202-206.

[11] L. N. Pouchet, PolyBench/C the Polyhedral Benchmark suite http://web.cse.ohio-state.edu/~pouchet/software/polybench/ #description.

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.