Sergii SUSHKO, Oleksander CHEMERYS
PUKHOV INSTITUTE FOR MODELING IN ENERGY ENGINEERING, NASU
15 General Naumov St., Kyiv, 03164, Ukraine

Dependency between Tiles' Sizes and Program Execution Time

Abstract

The paper is dedicated to the aspects of software optimization. Optimization problem is described. Tiling and parallelization methods were applied on the test applications. Several tests were performed to estimate influence of the tiles' sizes on the computational time. The obtained results show complicated dependency between tiles' sizes and processing time. Numerical characteristics of the obtained results and the corresponding pictures are presented.

Keywords: software optimization, tiling, parallelization, tile size, code transformation, fast computation.

1. Introduction

An increasing necessity to solve difficult scientific and technical tasks led to fast growth of multiprocessor computer systems (MCS) usage. Their productivity is one of the important parameters taking into account difficulty and size of the tasks to resolve. Fast growth of multiprocessor MCS requires an enhancing of the parallel software development, in particular parallelizing compilers, since development of the parallel program without synchronization issues and dead-end branches is not trivial.

Parallelization of the single thread program is in the mapping of operations' set on data that are placed in memory onto set of processor elements of some particular topology. MCS has to solve load balancing task of the processors to reduce number of the idle processors. This will lead to time reducing of the program execution. Other task of the increasing performance of the computations is a minimization of the data transmission between grids of MCS. Resolving of this task allows effective parallelization of the program.

Typically, the main volume of calculations is concentrated in computational loops. Especially it is very important in parallel processing applications. Hence, loop optimization has a direct impact on the overall application performance. In the case of nested loops, the performance is directly dependent on the degree of granularity of the computations. Transformations such as decomposing and merging of the nested loops (loop fission and loop fusion) can greatly simplify parallelization and improve its efficiency [1].

One of the most effective methods is the division of the loop into separate blocks (loop tiling) [2]. This optimization method consists in splitting the iteration space of the initial loop (which can be performed in several variables) into blocks of smaller size, which allows storing data used in these small blocks in the cache completely for their repeated use during the execution of the block. Partitioning the iteration space of the loop causes the array to be split into smaller blocks that are placed in the cache, which leads to better cache use, fewer misses and reduced cache size requirements [3].

Parallelization and optimization of the loops is an important factor in improving the performance of the applications with parallel processing of the data. Optimizations such as combining, permutation, and deploying loops are typically used to improve the degree of partitioning, load balancing, and efficient localization of the data, and at the same time incur minor costs associated with synchronization and other concurrency costs. The main rule - for parallelization fit the best loops with a large number of iterations. A large number of iterations makes it possible to improve the load balance due to the presence of more tasks for distribution between threads. One should also consider the amount of work performed in the single iteration. Unless otherwise specified, it is supposed that the amount of computation in the each loop iteration (approximately) is the same for all iterations of a given loop.

This paper describes the analysis of the loop tile size influence on the execution time of the program. For the experiments the Polybench software is considered [4]. The experiments were conducted for multi-threads programs but we suppose that the main conclusions are valid and for the distributed multiprocessor architectures.

2. Software optimization process

Optimization of the software is a process of the modification or transformation a source code to achieve better performance (processing time, consumed power, energy efficiency, data or program memory and so on) with the same final result.

Optimization problem for every *i*-th computational block can be defined as follow:

$$F_i = \arg\min\left(f\left(\overrightarrow{M_i}, \overrightarrow{P_i}\right)\right), \tag{1}$$

where F_i is the required parameter of the optimization, M is the vector of the optimization methods, P is the vector of the parameters of the optimization methods.

arg min is defined as:

$$\arg\min\left(f(x)\right) \in \{x | \forall y: f(x) \le f(y)\}. \tag{2}$$

Optimization problem for entire program with N computational blocks can be defined as:

$$F = \arg\min \sum_{i=1}^{N} f(\vec{M}_i, \vec{P}_i)$$

$$F = \arg\min\left(\sum_{i=1}^{N} f(\mathrm{M}_i, \mathrm{P}_i)\right). \tag{3}$$

It means that every particular part of the source code could have some unique set of the methods and its parameters to achieve the best possible productivity.

There are many methods of the optimizations. In terms of time processing it's obvious that better candidates for optimization are located in the computational loops. It's because body of loop is executed many times. Thus, small improvement of the computational loop can lead to the significant effect.

All operations are executed in the integer basis which defined and limited by loop indices. They define dimension and size of iteration space.

Definition. Iteration space is a set of all integer vectors $I = (I_1, I_2, \ldots, I_n)$ that satisfy inequalities

$$L_i \le x_i \le U_i, \tag{4}$$

where i=1..n.

Inequalities (4) define loops' bounds and they restrict the iteration space by the convex polyhedron.

As it was shown in [5] and [6] tiling and parallelization usually make some improvements in terms of time processing and energy efficiency. But these papers don't gives any results how size of the tiles changes efficiency. To verify it were performed some tests.

3. The experiments

Pluto framework [7] was chosen to check influence of the tiles' sizes on the processing time. It includes different tiling methods and parallelization possibility as well. For the experiments were used two option sets: tile and tile plus innerpar plus parallel. First one is a simple tiling method for the single core processing. Second one is a more complicated tiling method plus parallelization that utilizes all available cores by using OpenMP library.

Polybench test [5] container was chosen as a source of the test applications. Totally 17 test application passed the tests in two option sets. For the minimizing measurement error every test has passed 5 times but one maximum and minimum times were excluded. Mean result of 3 remaining values was suggested as a valid value.

Tests were performed on the desktop PC with Intel Core i5-4670K quadcore processor on Ubuntu 14.04 LTS operational system.

Tests were performed in the large range of tiles' sizes. For every pair of the tile sizes execution time was measured. Changing tile sizes one by one were obtained dependencies of the two tile sizes.

Some of the obtained results are shown in Fig. 1-5.

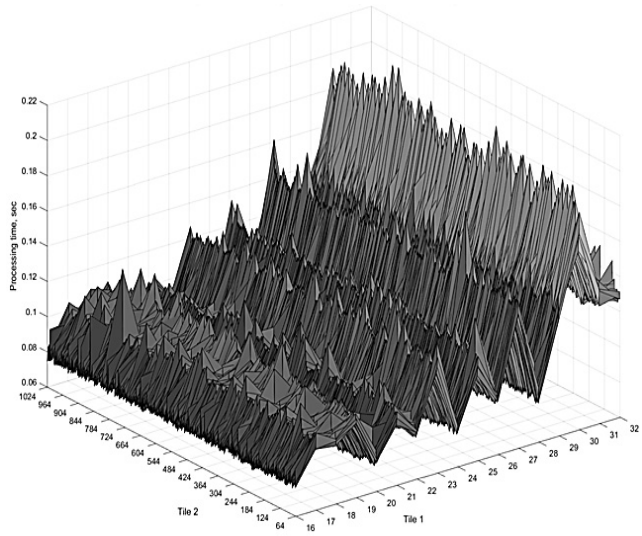

Fig. 3. Processing time for doitgen test application with used Tile, Innerpar and Parallel options.

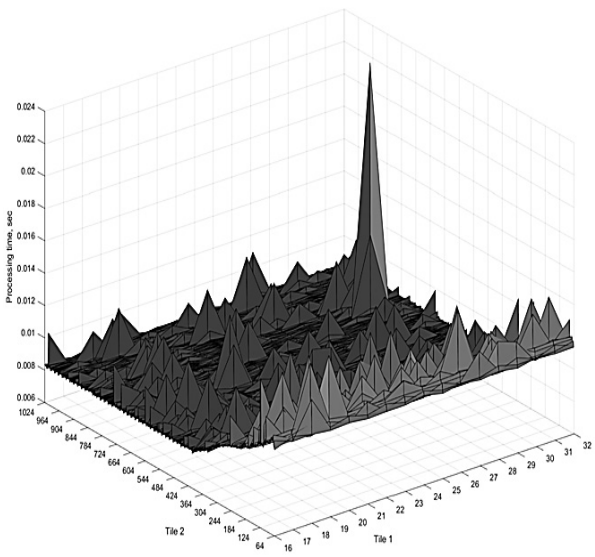

Fig. 1. Processing time for lu test application with used Tile option

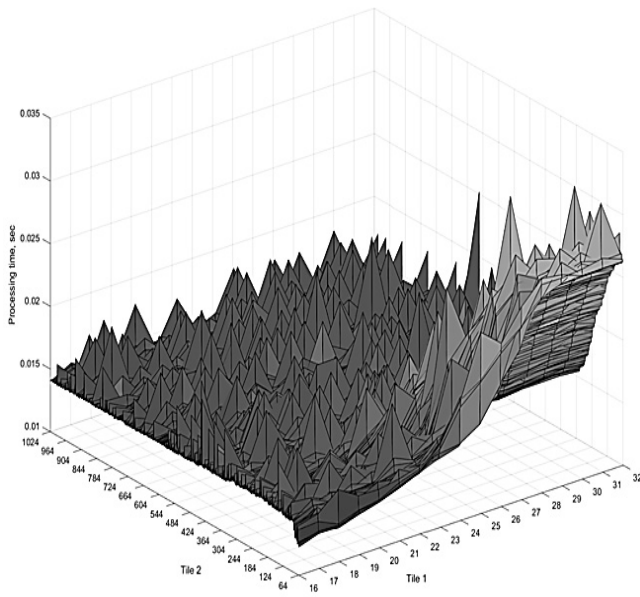

Fig. 4. Processing time for gesum test application with used Tile option

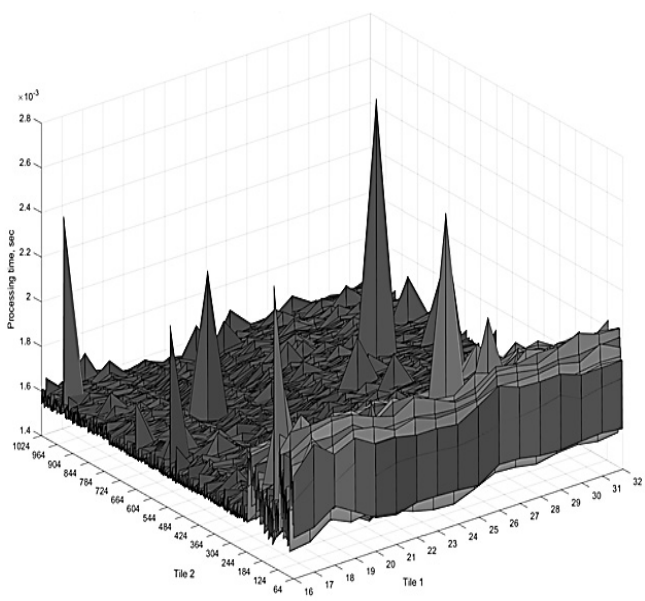

Fig. 2. Processing time for covariance test application with used Tile option

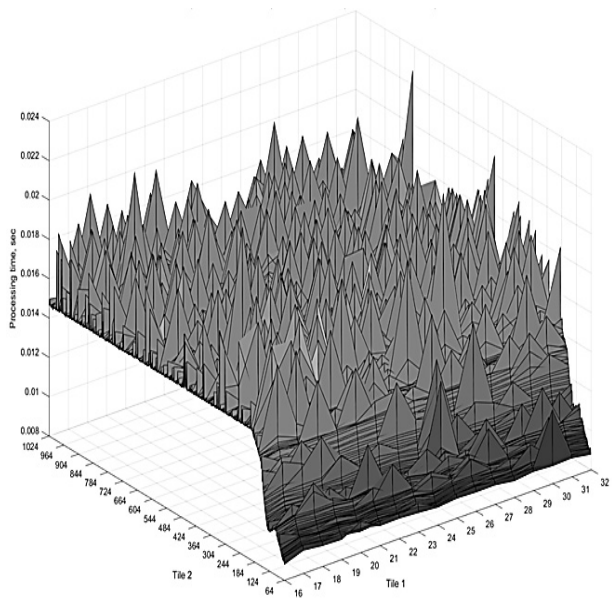

Fig. 5. Processing time for gramsmidt test application with used Tile option

At the horizontal axes both tiles are displayed. Vertical axe is an execution time. The figures show that tiles influence on the execution time in very different way. Sometimes execution time is rather constant in some range of tiles' sizes with small peaks.

Sometimes peaks are very big comparable to the adjacent values as it is shown in the Fig. 6. It could be explained by some internal processes in the operational system that significantly affects on the execution time.

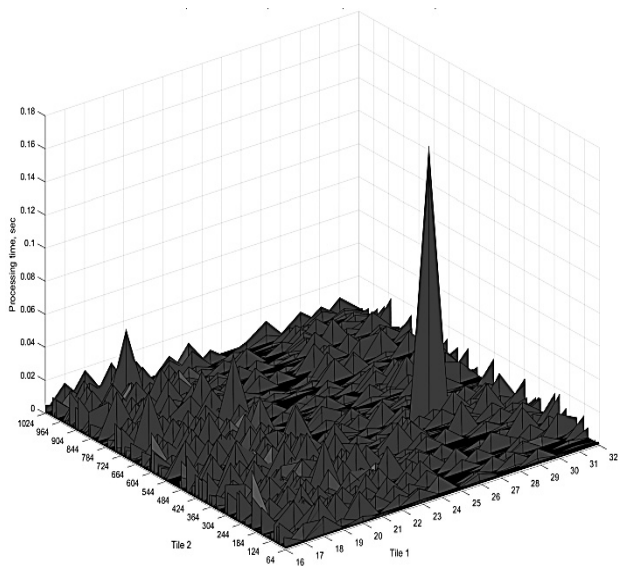

Fig. 6. Processing time for covariance test application with used Tile, Innerpar and Parallel options. Big peak is 254 times larger than the minimum value

For numerical analysis of the obtained results two characteristics were introduced – mean to minimum and maximum to minimum ratios for every test applications and modes. Such estimates allow evaluating common and the worst cases. Results of the obtained ratios are in the table below.

Tab. 1. Results of the measurements

Test program	Tile		Tile + Innerpar + Parallel	
	Mean/Min, times	Max/Min, times	Mean/Min, times	Max/Min, times
2mm	1.0254	2.4929	1.5102	43.7708
3mm	1.0277	1.4658	1.3991	13.7480
atax	1.1341	1.7816	1.4683	32.8580
bicg	1.1716	2.3085	1.4212	46.5144
cholesky	1.0571	1.5511	2.7493	28.6245
covariance	1.0826	1.8730	2.0335	254.5727
doitgen	1.6403	2.2937	1.5256	2.9942
durbin	1.0128	1.5814	1.0157	2.0391
gemm	1.1397	1.7401	1.2237	2.4104
gemver	1.2147	1.5456	1.2490	5.8665
gesummv	1.2520	2.7536	1.2958	13.7526
gramschmidt	1.6527	2.6383	2.0542	16.2835
lu	1.1135	2.9699	2.2271	48.8295
mvt	1.2086	2.4393	1.3390	8.2386
symm	1.0255	1.6794	1.0208	1.5740
syr2k	1.2027	1.9711	1.5365	15.8442
syrk	1.0466	2.2875	1.6282	28.8474

4. Conclusions

As it can be seen on the pictures, tiles make a huge influence on the computational time.

Usually, for simple tiling method with the single processing core difference between mean and minimum values is in the range (5-25)%. This means that in common not precise choice of the tiles' sizes can have lower time execution in that range. Relying on that it is necessary to note that accurate tiles' sizes are keys to further optimization. Also taking into account character of the dependencies it should be pointed out that there are no some particular dependencies. For every test or practical program optimal tiles' sizes cannot be foreseen or computed a priori, they could be achieved only by tests.

For further investigations some methods for the mathematical optimization may be considered. Having the surface of the obtained results, we may use optimization methods to find global minimum for the observed test programs. The genetic methods and the swarm method look as the most perspective.

5. References

[1] Fraboulet A., Kodary K. and Mignotte A.: Loop fusion for memory space optimization, The 14th International Symposium on System Synthesis, 2001, Proceedings, pp. 95 - 100, 2001.

[2] Xue J.: Loop Tiling for Parallelism. Kluwer Academic Publishers. 2000.

[3] Lam M. S., Rothberg E. E. and Wolf M. E.: The cache performance and optimizations of blocked algorithms. In Proceedings of the 4th International Conference on Architectural Support for Programming Languages and Operating Systems, pages 63–74, April 1991.

[4] Pouchet L.N.: The polyhedral benchmark suite [Online]. Available: http://web.cs.ucla.edu/*pouchet/software/polybench/ 22 Sept 2016

[5] Chemeris A., Sushko S.: Issledovanie bystrodeistviya i energopotrlebleniya pri avtomaticheskoy optimizatsii etodai razbieniya na bloki i raspaallelivaniya dlya vychisleniy na platforme x64 – Modeliuvannia ta informatsiyni tekhnologii. №80., pp. 52–60, 2017.

[6] Chemeris A., Lazorenko D., Sushko S.: Influence of Software Optimization on Energy Consumption of Embedded Systems – Green IT Engineering: Components, Networks and Systems Implementation. Springer, 2017.

[7] Uday Bondhugula: Effective Automatic Parallelization and Locality Optimization Using The Polyhedral model. – The Ohio State University, 2010. pp 193.

[8] Pouchet L.N.: The polyhedral benchmark suite [Online]. Available: http://web.cs.ucla.edu/*pouchet/software/polybench/ 22 Sept 2016.

Received: 10.12.2017 *Paper reviewed* *Accepted: 01.02.2018*

Sergii SUSHKO, MSc, eng.

Postgraduate student of Pukhov Institute for Modeling in Energy Engineering, NASU, Kyiv, Ukraine. Author has a great experience of the practical software development and optimization. Area of research: methods of software optimization, automatic source code improvement, energy efficiency of computations, code parallelization and tiling.

e-mail: sergii.sushko@gmail.com

Oleksandr CHEMERYS, MSc, eng.

Deputy director of PIMEE NASU. Strong background of elaborating and constructing of high-performance hardware, elaborating and applying program models for the computer simulation as well as developing original programs for the parallel architectures. Good experience in teaching of the programming languages. Main scientific interests are in 'Green' computing.

e-mail: a.a.chemeris@gmail.com