Łukasz NOZDRZYKOWSKI[1], Magdalena WRÓBEL[2]

Maritime University of Szczecin, Institute of Marine Technologies (1), Maritime University of Szczecin, Institute of Marine Technologies (2)

doi:10.15199/48.2015.12.61

Analysis of the significance of model parameters for program loop execution time estimation

Streszczenie. *W artykule przedstawiono modele szacowania czasów wykonywania się pętli programowych w formie zrównoleglonej oraz przedstawiono analizę istotności parametrów stosowanych do tego szacowania. Analiza istotności pozwala określić trafność doboru poszczególnych parametrów oraz wskazać parametry o niskiej istotności, które można byłoby zredukować.* ***Modele szacowania czasów wykonywania się pętli programowych w formie zrównoleglonej***

Abstract*. Models for estimating execution times of parallel program loops are discussed. The significance of parameters used for such estimation is analyzed. The significance analysis permits to determine the validity of parameters selected for estimation and to identify low significance parameters that may be eliminated.*

Słowa kluczowe: pętle programowe, szacowanie czasu, analiza istotności, teoria zbiorów przybliżonych.
Keywords: program loops, time estimation, significance analysis, rough set theory.

Introduction

The development of information technology systems and growing demand for high computing power have drawn attention of many specialists to multithreaded systems, which employ parallel programming for data processing. The parallelization of operations allows their acceleration, which is of vital importance for program loops, as these often take most of the time. An individual user may take a long time to carry out the parallelization of a sequential application, particularly where data dependency occurs. Another element requiring a lot of time and resources is the process of profiling needed for choosing optimal parameters of parallelization. Parallelization, however, sometimes fails to bring the intended outcome. Hence it is important to determine the rate of accelerating the code execution, especially parallel program loops. To determine the acceleration, we can use models of program loop execution time estimation based on their source code and parameters characterizing the runtime environment.

This article analyzes the significance of parameters of authors' models for the estimation of execution times of program loops in order to prove the correctness of models and parameters adopted in these models. The analysis has been based on the rough set theory and soft reduction of conditional attributes.

Models of program loop execution time

The parallelization of program loops may be a difficult operation due to dependencies occurring inside the loop body or/and dependencies between iterations. To execute a correct parallel loop, we have to honor dependencies by transforming a program loop or by using a program loop transformation that honors dependencies [1]. The following dependencies are distinguished: flow dependency, where a datum is first saved in computer's memory, then it is read out; anti-dependency, where a datum is first read out, then a new piece of information is added to the datum, and output dependency, where a piece of information is added to a datum, then a new item of information is again saved [2]. There are no dependencies in sequential programs because subsequent code lines are executed in a sequence. In parallel programs the sequence of executing the individual code lines can be changed because various fragments are being executed by separate threads [3].

Program loops may have dependencies within a loop body or there may be dependencies between iterations, or the two types of dependencies may co-exist. These dependencies may be honored by using the FAN, PAR and PIPE transformations [4]. The FAN transformation is used when there is no dependency in the program loop or dependencies exist inside the loop body, other types of dependency are not allowed. The PAR transformation is used when there are dependencies between iterations, but no dependencies can occur between instructions in a loop body. The two types of dependencies are allowed in the PIPE transformation. [4,5]

Loop execution time estimation with the FAN, PAR and PIPE transformations

The authors in [5] propose models of program loop execution time estimation for FAN and PAR transformations. For the PIPE transformation, a conversion is proposed to a form compatible with the FAN or PAR transformation.

The time-estimating model conforming with the FAN transformation for multithreaded computers is described by the formula (1).

$$T(n)=\sum_{k=1}^{K}\frac{(r_k l_i z_k)}{l_p n}+w m_d l_i+c_w+t_i \tag{1}$$

where: r - time of single operation execution, w – communication time, l_i – number of iterations (for nested loops all the iterations are summed up), m_d – amount of data required for computations for a thread, z – number of operations within a loop, l_p – number of pipeline stages in the processor, c_w – time of threads synchronization, t_i – time of measurement initiation, k – type of data locality, n – number of threads.

In the model (1) the time of executing a single code instruction was measured for various data localities. The number of locality measurements is determined by the examined program loop. The execution time for a parallel program loop is a sum of the product of individual execution times for an instruction with a given locality multiplied by the number of operations on the variable with that locality, and the number of iterations. This product is divided by the number of threads and number of pipeline stages in the processor. The calculated sum is the time of loop execution on a preset number of threads. This time value should be added to the time of data transmission between the threads (*w*md*li*). The time of transmitting a single datum *w* has been multiplied by the number of data required for the threads m_d and the number of iterations l_i. Finally, the time of synchronization between threads and the time of measurement initiation were added. The parameters r_k, w, c_w and t_i were measured in the testing environment. For a new testing environment, new measurements of these parameters have to be carried out. An example method of measuring the parameters r_k, w, c_w and t_i is shown in Listing 1.

r	t1=omp_get_wtime(); for(i=0;i<n;i++){} t2=omp_get_wtime(); time_for=t2-t1; t3=omp_get_wtime(); for(i=0;i<n;i++){ c1 = a1 + b1; t4=omp_get_wtime(); r= t4-t3- czas_for
w	#pragma omp parallel sections{ #pragma omp section{ t1=omp_get_wtime(); t3=t1; for(j=0; j<i; j++){ buff[j]='a';} t4=omp_get_wtime();} #pragma omp section { while(buff[i-1]!='a'){} t2=omp_get_wtime();}} t5=omp_get_wtime(); for(j=0; j<i; j++){} t6=omp_get_wtime(); for(j=0; j<i; j++){ buff[j]='0';} time_for=t6-t5; time_zap=t4-t3; time_cal=t2-t1; w= czas_cal - czas_zap - czas_for;
cw	omp_set_num_threads(n); #pragma omp parallel{ if(omp_in_parallel()){ i = omp_get_thread_num(); t1[i]=omp_get_wtime();} #pragma omp barrier i = omp_get_thread_num(); t2[i]=omp_get_wtime();}} cw=mean(t2-t1)
ti	t1=omp_get_wtime(); t2=omp_get_wtime(); t=t2-t1;

Listing 1 – the method of determining individual parameters.

The parameters comprised in list. 1 should be determined each time the program loop execution time is being estimated. This is due to the fact that during a program loop tested computer's sub-units have variable loads. The method of determining the parameter r is modified to suit data locality.

Another model for program loop execution time estimation makes use of the PAR transformation. The model has this form (2).

$$(2)\quad T(n)=\max_{i}\sum_{k=1}^{n}\frac{\left(r_{k1}l_i z_{k1}\right)}{l_p n}+w_i m_{di}+c_{wi}+t_i$$

Parameter symbols are identical to those used for the FAN transformation model. In the model (2) the maximum time, meaning the critical path of that algorithm, is chosen. For each path in the algorithm time is estimated similarly to the method for the FAN transformation. Of all time estimates, the maximum time is chosen.

The presented models (1, 2) of program loop execution time estimation have been compared to real measurements of loop executions from the NAS parallel benchmark [6] discussed in [7]. The errors of model-based estimation for selected loops are given in table 1. The complete results of model-based estimation are given in articles [5,7].

The results presented in table 1 show that the average error of the models (1, 2) amounts to 16.425%. The maximum estimation error was 36.63%, while the lowest value of estimation error equaled 0.5%. Table 2 presents the matching of the models to real loop measurements.

Table 1. Errors of estimating by the models (1, 2)

	Models (1), (2)			
Threads num. / Loop name	2	4	6	8
MG_mg_f2p_1	11.75	20.28	23.48	13.58
CG_cg,f2p_6 2D	16.46	4.60	15.50	23.58
MG_mg_f2p_4 2D	12.85	9.59	0.50	1.53
CG_cg,f2p_6	17.85	2.36	29.28	22.08
Seidel	1.96	23.67	24.29	28.20
MG_mg.f2p_4	36.63	35.74	10.57	7.88
Average	16.25	16.04	17.27	16.14

Table 2. Linear correlation of the models (1,2) with the real results

Loop name	Pearson Linear Correlation
MG_mg_f2p_1	0.974175565
CG_cg,f2p_6 2D	0.792099983
MG_mg_f2p_4 2D	0.961035177
CG_cg,f2p_6	0.890530692
Seidel	0.991540911
MG_mg.f2p_4	0.864850576

Pearson linear correlation presented in table 2 is strong and positive, which proves that the models fit well to real times of program loop execution. The average correlation for the models was 0.91, with the lowest value 0.79, while the greatest correlation amounted to 0.99.

Analysis of parameter significance

To analyze the significance of individual parameters which allow estimation of program loop execution time, we have used soft reduction of conditional attributes based on the relative probability of useful rules in the rough set theory [8, 9]. The analysis was based on real measurements performed in the testing environment of program loop execution times. All parameters required in the models estimating the time were determined for that environment.

The use of soft reduction of conditional attributes will allow rejecting those attributes whose removal will not lead to a decreased number of rules generating completely certain rules [10]. The method permits to evaluate the quality of rules based on relative probability of the atomic rules. The probability is expressed by this formula:

$$(3)\qquad P_w=\frac{P}{L}$$

where: P - sum of the probabilities of useful atomic rules, L – number of elementary conditional sets.

An atomic rule, generated for a single elementary set, is said to be useful when its probability is higher than a preset tested threshold, above which the rule can be considered as useful. By specifying decision and conditional attributes and coding them properly, we make an analysis by the reduction of individual conditional attributes and the determination of their significance in generating certain rules. If we use the soft reduction of conditional attributes, the reduction will result in a slight drop of the number of items generating completely certain rules.

The performed significance analysis was based on 32,133 measurement results of the execution time of all program loops from the NAS benchmark [X]. The NAS benchmark includes programs characteristic of irregular access to memory, enhanced communication, or the use of multi-dimensional tables.

Time Time measurements were done for various loads on an eight-threaded processor Intel Core i7-2600 with 8GB DDR3 RAM. The rules analyzed were those with relative probability above 80 %. Additionally, an analysis was made for the probability greater than 90 %.

In the significance analysis of parameters used for estimation of program loop execution times, conditional attributes are parameters of models (1,2). The decision attribute is the time of program loop execution. Attribute coding was done by the method of equal number of samples on the intervals, dividing each of them into 5 intervals. Attribute coding for the runtime environment was done as shown in table 3.

Table 3. Attribute coding.

Attribute	Intervals accepted for coding
Execution time	(0, 0.026⟩, (0.026, 0.073⟩, 0.073, 0.116⟩,(0.073, 0.116⟩, (0.116, ∞)
r	(0, 6.9e-10⟩, (6.9e-10, 7.1e-10⟩, (7.1e-10, 7.3e-10⟩, (7.3e-10, 7.6e-10⟩, (7.6e-10, ∞)
w	(0, 1.8e-8⟩, (1.8e-8, 2.1e-8⟩, (2.1e-8, 2.4e-8⟩, (2.4e-8, 2.75e-8⟩, (2.75e-8, ∞)
l_w	(1, 3⟩, (3,4⟩, (4,5⟩, (5,7⟩, (7,8⟩
l_i	(0,7500000⟩, (7500000,25000000⟩, (25000000, 50000000⟩, (50000000, 80000000⟩, (80000000, ∞)
m_d	(0, 2⟩, (2,3⟩, (3,4⟩, (4,5⟩, (5, ∞)
z	(0, 6⟩, (6,8⟩, (8,9⟩, (9,19⟩, (19, ∞)

We have determined the number of elementary sets of atomic useful rules and the relative probability of all atomic useful rules for an unreduced set of conditional attributes. The analysis results are presented in table 4.

Table 4 – The analysis results for an unreduced set of conditional attributes

Probability Pw	0.8	0.9
Number of elementary sets	3941	3941
Number of atomic useful rules	2819	2615
Relative probability of all atomic useful rules	0.703	0.661

As a result of the analysis performed using the rough sets theory and soft reduction of conditional attributes, we have obtained significance levels of individual conditional attributes, presented in tables 5 and 6, for two levels of probability: 0.8 and 0.9.

Table 5. Analysis of conditional attribute significance at P_w = 0.8.

Conditional attribute	Number of elementary reduced sets	Number of atomic useful reduced rules	Relative probability of all atomic useful reduced rules	Attribute significance
r	960	623	0.63	0.11
w	961	609	0.62	0.12
l_w	943	613	0.63	0.10
l_i	1460	683	0.45	0.36
m_d	2539	1362	0.52	0.26
z	2189	1303	0.58	0.17

It follows from the analysis results that the most significant attribute is the number of iterations l_i, whose significance for Pw = 0.9 amounts to 43 percent. Other parameters of large significance are the amount of data required for computations by a thread m_d (32%) and number of operations within a loop z (18%). The least significant are the time parameters r and w, the time of a single operation, and data transmission, respectively. Nevertheless, at the significance level of 15 % for Pw=0.9 these attributes should not be reduced. The results show that the parameters used in the time execution estimation models have a high significance and should not be reduced.

Table 6. Analysis of conditional attribute significance at P_w = 0.9.

Conditional attribute	Number of elementary reduced sets	Number of atomic useful reduced rules	Relative probability of all atomic useful reduced rules	Attribute significance
r	960	540	0.56	0.15
w	961	549	0.57	0.14
l_w	943	527	0.55	0.16
l_i	1460	547	0.37	0.44
m_d	2539	1145	0.45	0.32
z	2189	1184	0.54	0.18

Summary

In this article, presenting models of program loop execution time estimation based on the source code of the loop and parameters characterizing the program environment, we have proved that the selected parameters for the models are correct. The significance analysis of individual parameters was based on the rough set theory and aimed at identifying parameters that might be eliminated from the models. From the analysis results we conclude that the examined attributes are very useful and should not be reduced. Their selection and high effectiveness in program loop execution time estimation, as herein proved by these authors, confirm that the proposed models are designed correctly.

REFERENCES

[1] Allen R., Kennedy K., Optimizing Compilers for Modern Architectures: A Dependence-based Approach, Morgan Kaufmann, 2001

[2] Pałkowski M., Algorytmy zwiększające ekstrakcję równoległości w pętlach programowych, praca doktorska, Politechnika Szczecińska, 2008

[3] Czech Z, Wprowadzenie do obliczeń równoległych, Wydawnictwo PWN, 2010

[4] Lewis T, Foundations of Parallel Programming: A Machine-Independent Approach, IEEE Computer Society Press, 1992

[5] Wróbel M., Nozdrzykowski Ł.: Model for estimating the execution time of the loop program with limited connection, Logistyka 4/2014, pp.3425-3435

[6] NAS Parallel Benchmarks, http://www.nas.nasa.gov/publications/npb.html (access date: February 2015)

[7] Magdalena Wróbel: Models for estimating the execution time of software loops in parallel and distributed systems, Theory and Engineering of Complex Systems and Dependability, Vol. 365, pp. 533-542

[8]. Z. Pawlak, "Rough Sets", International Journal of Computer and Information Sciences, Vol.11 1982

[9]. J. Ponce., A. Karahoca, Data Mining and Knowledge Discovery in Real Life Applications, i-Tech Education and Publishing, Croatia 2009

[10]. A. Mrózek, L. Płonka, Analiza danych metodą zbiorów przybliżonych. Akademicka Oficyna Wydawnicza PLJ, Wydanie 1, Warszawa 1999

***Authors**: dr inż. Łukasz Nozdrzykowski, Maritime University of Szczecin, Institute of Marine Technologies, ul. Wały Chrobrego 1-2, 70-500 Szczecin, E-mail: l.nozdrzykowski@am.szczecin.pl; mgr inż. Magdalena Wróbel, Maritime University of Szczecin, Institute of Marine Technologies, ul. Wały Chrobrego 1-2, 70-500 Szczecin, E-mail:m.wrobel@am.szczecin.pl;*