Michał GOZDALIK
WEST POMERANIAN UNIVERSITY OF TECHNOLOGY IN SZCZECIN

A fuzzy model in speedup prediction process for parallel applications written in OpenMP

Mgr inż. Michał GOZDALIK

A PhD student at the West Pomeranian University of Technology In Szczecin. Currently occupied with creating a tool which allows generating the parallel code in C, with the consent of OpenMP standard, which could take the most possible advantages of multi-processor machines.

e-mail: mgozdalik@wi.ps.pl

Abstract

A common method to establish code parallelization quality is measuring the program execution time to calculate speedup and efficiency. Generally, parallel and sequential programs must be executed and execution time need to be captured to affirm quality parameters. However, having a good profiling tool, it is easier to designate parameters such as a bus utilization ratio, rather than the measuring program execution time. Having a piece of information about processor and memory ratios, it is possible to estimate quality parameters with satisfying results. In this paper an example solution of the effectiveness prediction process of parallel programs written in OpenMP is provided. As an approach, a fuzzy model was designed and results for a matrix multiplication program are presented. The fuzzy model and a modus operandi are described. Nevertheless, parameters for estimating the efficiency and speedup were implemented using Intel processors event calculation. These parameters are input values of the fuzzy model presented in this paper. According to processor events, the input parameters where divided into two groups. Each group represents one of a submodel in the whole fuzzy model. It provides possibility to measure only some of processor events to estimate the program efficiency. More details on these parameters are included in separate paragraphs.

Keywords: OpenMP, fuzzy logic, shared memory programming.

Rozmyty model predykcji efektywności aplikacji równoległych w standardzie OpenMP

Streszczenie

W artykule przedstawiony został problem dotyczący określenia jakości wygenerowanego kodu równoległego. Mierzenie czasu wykonania programu celem wyznaczenia przyspieszenia jest nieefektywne, a w niektórych przypadkach wręcz niewykonalne. Posiadając narzędzie profilujące dedykowane dla danego typu procesora, możliwe jest stworzenie modelu, który estymował by efektywność wykonywanego programu na podstawie parametrów pamięci cache poziomu drugiego oraz procesora. Dzięki takiemu rozwiązaniu możliwe jest określenie jakości wygenerowanego kodu i podjęcie na tej podstawie decyzji czy warto dalej optymalizować wygenerowany kod. Celem wykonania pomiaru parametrów pamięci i procesora wystarczy wykonywać program przez określony wycinek czasu nie czekając na jego zakończenie. Nie ma również konieczności ingerowania w kod źródłowy programu. Niniejszy artykuł prezentuje model rozmyty estymujący efektywność wygenerowanego kodu źródłowego w standardzie OpenMP.

Słowa kluczowe: OpenMP, programowanie równoległe, automatyczna generacja kodu.

1. Introduction

Parallel programs were intended to work as fast as possible under a specified runtime environment. Multi core processors were designed to satisfy consumers' needs for efficient computation platforms which allow parallel programs to run efficiently.

To prove the effectiveness of parallel program optimisation process, scales and measures like the speedup and efficiency are used. Despite being very precise, those measures are highly time consuming to apply. A program must be executed and execution time must be established. In most cases there is a need to interfere with the program source code to provide time measuring commands, which can be hard to achieve for those programs whose source code was not included.

There are also other methods to establish the program speedup and efficiency. Some of them base on a process of analyzing the assembler code and others use neural networks. In this paper a fuzzy model is constructed to estimate an effectiveness parameter. Having a specialised profiling tool like the Intel VTune Analyzer, the speedup and efficiency can be designated from processors events and ratios. Obtaining the detailed information about hardware states during program execution, enables predicting the speedup and efficiency values. The rest of this paper is dedicated to a fuzzy model which can establish the program efficiency values, according to the Intel processors ratios and events. This process is less time consuming than that based on measuring the program execution time. There is also no need to interfere with a source code.

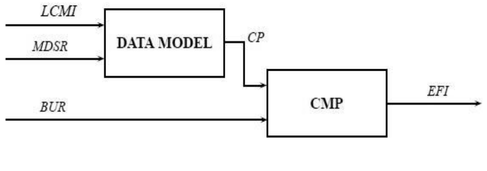
2. A fuzzy model

The model consists of two submodels. The first one is called the Data Model, because all of its input parameters are connected with data flow. The output parameter was named the Cache Parameter. The name indicates the meaning of this ratio. In general, the Data Model defines the cache memory status during a parallel program execution. This status is evaluated as the Cache Parameter and contains information about how efficiently a piece of data was transferred between processor registers and cache memory.

The second one is called the Common Mapping Part due to information gathering via this submodel. The output parameter estimates the efficiency of a parallel program under test. An efficiency parameter is known in parallel computing as one of quality measurements of a parallel program.

All input parameters are able to be measured using the Intel Vtune Profiler and are used to measure the efficiency of Intel multi core processors made in 45 nanometers technology.

Fig. 1 illustrates the whole fuzzy model.



Rys. 1. Model systemu rozmytego estymujący efektywność programu
Fig. 1. A fuzzy model estimating the program efficiency

3. A Data Model

This submodel consist of three parameters. Two of them are input parameters and are described below.

- L2 Cache Miss Impact (LCMI) — This ratio provides information about the absence of data in the second level cache memory. This memory is placed between the first level cache memory and the random access memory. In a memory hierarchy this memory is one of commonly used by Intel processors to store data and provide high impact for parallel program efficiency. The perfect value of this ratio is 0, which means that no time was spent by the program to achieve data from random access memory, because all needed data had been already stored in the second level cache memory. To determine the L2 Cache Miss Impact parameter, a MEM_LOAD_RETIRED.L2 LINE_MISS and INST_RETIRED.ANY events need to be measured. It can be done by the Intel VTune Analyzer. LCMI is a fraction of MEM_LOAD_RETIRED.L2 LINE_MISS and INST_RETIRED.ANY. It is worth noticing that the access time to the second level cache memory is five to seven times shorter than the time needed to reach the random access memory.

- Modified Data Sharing Ratio (MDSR) — This ratio provides information about problems with accessing data stored in the memory which is shared between many threads. The perfect value for this ratio is 0, which means that none of threads which a program executes need to wait for access to data stored in a shared memory. To determine a Modified Data Sharing Ratio, EXT_SNOOP.ALL_AGENTS.HITM and INST_RETIRED.ANY events need to be measured. It can be done by the Intel VTune Analyzer. MDSR is presented as a fraction of EXT_SNOOP.ALL_AGENTS.HITM and INST_RETIRED.ANY events.

The last one is an output parameter which is called the Cache Parameter. This ratio was described in Section 2. It is also an input parameter for a submodel called the Common Mapping Part.

4. Common Mapping Part

This submodel consists of three parameters. Two of them are input ratios and one of them is an output parameter. A Cache Parameter is one of input ratio and is described in Section 2. The Bus Utilization Ratio as an input parameter provides information about the time needed to gain access to the Front Side Bus by a parallel program. The perfect value for this ratio is 0, which means that a parallel program spent no time waiting to gain access to the Front Side Bus. An output parameter in this submodel is also an output parameter of the whole fuzzy model and provides information about the program efficiency. This parameter is strictly connected with the speedup. As a matter of fact, it is the speedup divided by the number of processors used for the execution of a program. The following efficiency is provided

$$\delta_A(n,p)=\frac{S(n,p)}{p} \tag{1}$$

$\delta_A\,(n, p)$ is the efficiency of a code whose size is n performed on p processors. $S(n,\ p)$ is the speedup counted by the following formula.

$$\delta_A(n,p)=\frac{T(n,1)}{T(n,p)} \tag{2}$$

As can be seen, this parameter can provide information about what was the generated code quality according to the execution time. Such information is crucial when there is the need to establish procedures which state whether a parallelization process is efficient or not.

It is worth noticing that the efficiency used in the fuzzy model as an output, can be easily transformed to the efficiency described in this section as the speedup divided by the number of processors. To change output from the model to efficiency, the formula given below must be used.

$$\delta_A(n,p)=1-\text{efficiency from model} \tag{3}$$

To determine the Bus Utilization Ratio, BUS_TRANS_ANY. ALL_AGENTS and CPU_CLK_UNHALTED.BUS events need to be measured. It can be done by the Intel VTune Analyzer. BUR is presented as the fraction of BUS_TRANS_ANY.ALL_AGENTS and CPU_CLK_UNHALTED.BUS events.

5. Process of fuzzyfication

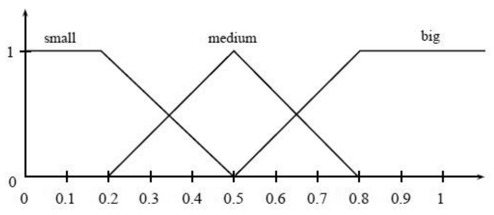
All parameters introduced into the model are discrete. Fuzzy models reflect the way people's mind works --- rather in a continuous not discrete way. Moreover, people better understand the meaning of words than of values. The fuzzyfication process enables transforming discrete values into continues values, described in a linguistic form, which allows a fuzzy model to calculate on input data. For all input parameters used in the fuzzy model, linguistic values are assigned. All parameters are anti stimulants, what means that the lower the value the better efficiency will be achieved. All parameters were normalised to <0,1>. The chosen linguistic forms are shown below.

- L2 Cache Miss Impact (LCMI) — small, medium, big
- Modified Data Sharing Ratio (MDSR) — high, medium, low
- Bus Utilization Ratio (BUR) — high, medium, low
- Cache Parameter CP — very good, good, bad
- Efficiency — very good, good, medium, bad, very bad

In Intel white papers provided on http://www.intel.com there are some clues about what is the meaning of the values combined with fuzzy model parameters. According to these clues, values of parameters obtained from the fuzzyfication process are described in the next sections. In the figures, x axes indicate the parameter value measured from a running program. The y axes provide information about how adequate a linguistic form is according to the measured value in percentage, which is called the level of an appurtenance. For example, if the measured value was 0,3 on x axis and the plot indicates value 0,5 on y axis, for a linguistic form *small* and 0,5 on y axis for linguistic form *medium*, that means 0,3 is half *medium* half *small*.

6. A fuzzyfied L2 Cache Miss Impact Parameter

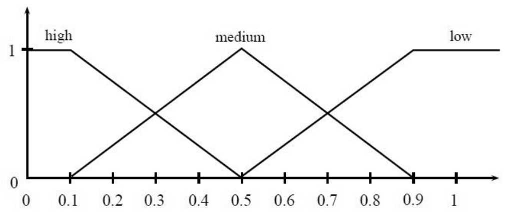
In Fig. 2, the linguistic values for a L2 Cache Miss Impact parameter are presented. The value *small* range is between 0 and 0,5. In the range from 0 to 0,2 the L2 Cache Miss Impact parameter is assigned to the linguistic form *small* in a 100%, which is indicated by value 1. From 0,2 to 0,5 the L2 Cache Miss Impact parameter is not purely *small* but also *medium*. The range of a linguistic *medium* value is from 0,2 to 0,8. The purely *medium* value is assigned when the L2 Cache Miss Impact parameter is equal to 0,5. Accordingly, *big* value is achieved when the L2 Cache Miss Impact parameter is greater than 0,5.



Rys. 2. Fuzzyfikacja parametru LCMI
Fig. 2. Fuzzyfied LCMI Parameter

7. A fuzzyfied Modified Data Sharing Ratio

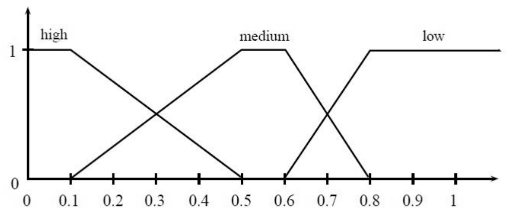
In Fig. 3 the linguistic values for the Modified Data Sharing Ratio are presented. From 0 to 0,5 the linguistic value *high* was assigned with an adequate level of an appurtenance. From 0,1 to 0,9 the linguistic value *medium* was assigned. If the Modified Data Sharing Ratio is higher than 0,5, the linguistic value *low* is assigned with an adequate level of appurtenance.



Rys. 3. Fuzzyfikacja parametru MDSR
Fig. 3. Fuzzyfied MDSR Parameter

8. A fuzzyfied Bus Utilization Ratio

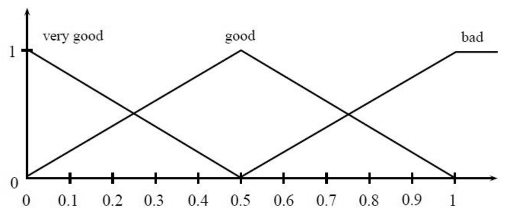
In figure 4, the linguistic values for the Bus Utilization Ratio are presented. From 0 to 0,5 the linguistic value *high* was assigned with an adequate level of an appurtenance. From 0,1 to 0,8 the linguistic value *medium* was assigned. If the Bus Utilization Ratio is higher than 0,6 the linguistic value *low* is assigned with an adequate level of an appurtenance.



Rys. 4. Fuzzyfikacja parametru BUR
Fig. 4. Fuzzyfied BUR Parameter

9. A fuzzyfied Cache Parameter

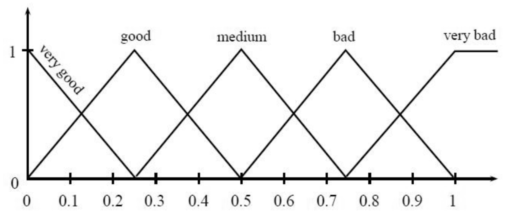
In figure 5, the linguistic values for the Cache Parameter are presented. From 0 to 0,5 the linguistic value *very good* was assigned with an adequate level of an appurtenance. From 0 to 1 the linguistic value *good* was assigned. Whether the Cache Ratio is higher than 0,5 the linguistic value *bad* is assigned with an adequate level of appurtenance.



Rys. 5. Fuzzyfikacja parametru CP
Fig. 5. Fuzzyfied Cache Parameter

10. A fuzzyfied efficiency parameter

In Fig. 6 the linguistic values for Efficiency are presented. From 0 to 0,25 the linguistic value *very good* was assigned with an adequate level of appurtenance. From 0 to 0,5 the linguistic value *good* was applied. In the range from 0,25 to 0,75 the linguistic value *medium* was allocated. From 0,5 to 1 the linguistic value *bad* was assigned. If the efficiency is higher than 0,75, the linguistic value *very bad* was assigned with an adequate level of appurtenance.



Rys. 6. Fuzzyfikacja parametru EFI
Fig. 6. Fuzzyfied Efficiency Parameter

11. A base of model rules

The base of model rules provides information about how the fuzzy model works in general. There are eighteen rules the model depends on. Nine of theme are adequate for the Data Model submodel and other nine are part of the Common Mapping Model submodel. These rules determine how the output parameters react to the input ones. Below all rules for the Data Model are given

- IF the Modified Data Sharing Ratio is high AND the L2 Cache Miss Impact parameter is small THEN the Cache Parameter is very good
- IF the Modified Data Sharing Ratio is medium AND the L2 Cache Miss Impact parameter is small THEN the Cache Parameter is very good
- IF the Modified Data Sharing Ratio is low AND the L2 Cache Miss Impact parameter is small THEN the Cache Parameter is good
- IF the Modified Data Sharing Ratio is high AND the L2 Cache Miss Impact parameter is medium THEN the Cache Parameter is very good
- IF the Modified Data Sharing Ratio is medium AND the L2 Cache Miss Impact parameter is medium THEN the Cache Parameter is good
- IF the Modified Data Sharing Ratio is low AND the L2 Cache Miss Impact parameter is medium THEN the Cache Parameter is bad
- IF the Modified Data Sharing Ratio is high AND the L2 Cache Miss Impact parameter is big THEN the Cache Parameter is good
- IF the Modified Data Sharing Ratio is medium AND the L2 Cache Miss Impact parameter is big THEN the Cache Parameter is bad
- IF the Modified Data Sharing Ratio is low AND the L2 Cache Miss Impact parameter is big THEN the Cache Parameter is bad

There are also nine rules which indicate how the output parameter depends on the input ones. These rules are stated below.

- IF the Bus Utilization Parameter is high AND the Cache Parameter is very good THEN the Efficiency parameter is very good
- IF the Bus Utilization Parameter is medium AND the Cache Parameter is very good THEN the Efficiency parameter is good
- IF the Bus Utilization Parameter is low AND the Cache Parameter is very good THEN the Efficiency parameter is medium
- IF the Bus Utilization Parameter is high AND the Cache Parameter is good THEN the Efficiency parameter is good
- IF the Bus Utilization Parameter is medium AND the Cache Parameter is good THEN the Efficiency parameter is medium
- IF the Bus Utilization Parameter is low AND the Cache Parameter is good THEN the Efficiency parameter is bad
- IF the Bus Utilization Parameter is high AND the Cache Parameter is bad THEN the Efficiency parameter is medium
- IF the Bus Utilization Parameter is medium AND the Cache Parameter is bad THEN the Efficiency parameter is bad
- IF the Bus Utilization Parameter is low AND the Cache Parameter is bad THEN the Efficiency parameter is very bad

12. Example

As an example, a matrix multiplication application was chosen. The source code of the program main part is shown below.

```
#pragma   omp   parallel   for   private(i,j,k,suma)
shared(tab1,tab2)
for (i=0;i<WYM;i=i+1)
{
  for (j=0;j<WYM;j=j+1)
  {
    suma = 0;
    for (k=0;k<WYM;k=k+1)
    {
      suma=suma+tab1[i][k]*tab2[k][j];
    }
    wynik[i][j] = suma;
  }}
```

Time measurements were made using the gettimeofday function which is implemented in the C language. This function can establish an accurate time which was spent by processors on executing the program specific commands. The time was counted in processor ticks, which is the smallest amount of time needed to execute a single processor instruction. To shorten the distortion problem, all tests where repeated five times and the mean value was the one which was used to calculate the speedup and efficiency. Moreover, measures were made for different matrice dimensions. In Tables 3 and 4 the values in the dimension column are the amount of rows in multiplied matrices. For example, the value 100 informs that two square matrices were multiplied together.

The measurements were made on a server machine with the Intel Quad-core Q9550 BOX processor. Twelve mega bytes of a second level cache memory was available for all cores.

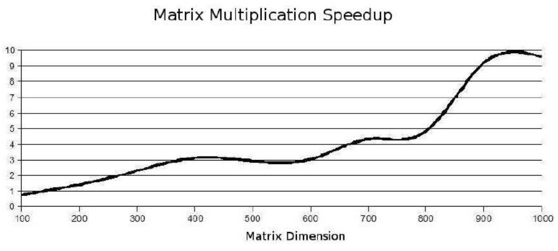
Tab.1. Pomiary czasu wykonania program mnożącego macierze w sposób sekwencyjny
Tab. 1 The results of measuring time for matrix multiplication program executed sequentially

Dimension	*SEQUENTIAL*					
	1	*2*	*3*	*4*	*5*	*Mean*
100	10938	10916	11098	10963	11020	10987
200	87734	87827	87348	87889	88121	87784
300	303379	303204	301894	301808	303195	302696
400	912171	914901	911732	908634	916598	912807
500	1932870	1932844	1935962	1932502	1932368	1933309
600	3442602	3416323	3453511	3403122	3439185	3430949
700	6009118	6046403	6032484	6040126	6004998	6026626
800	10118950	10335589	10233387	10195359	10214222	10219501
900	15652762	15792388	15748217	15683634	15641455	15703691
1000	22152380	22161050	22185937	22187276	22166987	22170726

Tab. 2. Pomiary czasu wykonania program mnożącego macierze w sposób równoległy
Tab. 2. The results of measuring time for matrix multiplication program executed in parallel

Dimension	*PARALLEL*					
	1	*2*	*3*	*4*	*5*	*Mean*
100	6478	11353	9415	7106	11791	9229
200	35571	34926	41054	32501	52312	39273
300	71729	64807	70822	76377	77931	72333
400	164675	167911	146910	158972	148333	157360
500	316111	283079	295223	291340	294995	296150
600	501881	505710	508453	505136	499647	504165
700	895087	861511	874122	861895	867714	872066
800	1473288	1450844	1447694	1452331	1452123	1455256
900	2238686	2196408	2218995	2245063	2225694	2224969
1000	3111751	3123255	3107759	3113325	3125927	3116403

Having measured the time, it is possible to calculate the speedup and efficiency using formulas (2) and (1), accordingly. Fig. 7 shows the speedup achieved during the tests.



Rys. 7. Przyspieszenie programu mnożącego macierze
Fig. 7. The speedup calculated for matrix multiplication program

To verify the fuzzy model, the input parameters were designated from Intel© Vtune® Performance Analyzer for matrices dimensions 800x800. Beneath the measured results are presented

- MEM_LOAD_RETIRED.L2 LINE_MISS = 245
- INST_RETIRED.ANY = 7788
- EXT_SNOOP.ALL_AGENTS.HITM = 108
- BUS_TRANS_ANY.ALL_AGENTS = 27116
- CPU_CLK_UNHALTED.BUS = 37648

The above results enable calculating the input parameters. In Section 3 the formulas are presented. The results are as follows

- LCMI = 0,031
- MDSR = 0,01
- BUR = 0,44

To calculate CP parameter which is the output from the Data Model subsystem, MAXPROD schema was used. MAXPROD operator calculates products from ranges. As a result, the maximum value is taken as CP parameter which became the input to CMP subsystem.

The implementation of the fuzzy model described in this paper is performed in Matlab. Providing all input parameters to the implemented model, the value 0,2112 is obtained, which means that the estimated efficiency is 0,7888 according to Eq. 3 in Section 4. The real efficiency measured was about 1, so the estimation misled the real efficiency with deviation of about 21,12%.

13. References

[1] Ruud Van Der Pas Barbara Chapman, Gabriele Jost. Using OpenMP . The MIT Press, 2007.

[2] Omega Project Do cumentation. http://www.cs.umd.edu/projects/omega/. Valid at June 25, 2009. Omega project website.

[3] Intel white pages. http://www.intel.com. Intel website.

[4] Fujitsu Systems Europe Ltd., http://www.compunity.org/compilers/fujitsu/Workbench.pdf. Parallel Navi Workbench: an OpenMP Development Enviroment for a Wide-Area Network Url valid at June 25, 2009.

[5] Mostafa Abd-El-Barr Hesham El-Rewini. Advanced computer architecture and parallel processing . Wiley Interscience, 2005.

[6] OpenMP Architecture Review Board, http://www.openmp.org/blog/specifications/cspec20.pdf. OpenMP C and C++ Application Program Interface, 2.0 edition, Marc 2002. Url valid at June 25, 2009.

[7] OpenMP Architecture Review Board, http://www.openmp.org/drupal/mp-documents/cspec30_draft.pdf. OpenMP Application Program Interface Draft 3.0 , 3.0 edition, October 2007. Url valid at June 25, 2009.

[8] Marek Pałkowski. http://detox.wi.ps.pl/SFS_Project. Valid at June 25, 2009. ESyS project website.

[9] Dave Kohr Rohid Chandra, Leonardo Dagum. Parallel programming in OpenMP . Morgan Kaufmann Publishers, 2001.

[10] University of Electro-Communications, Japan, http://www.compunity.org/events/ewomp03/omptalks/Monday/Session2/T08p.pdf. Interactive Parallelizing Assistance Tool for OpenMP:iPat/OMP . Url valid at June 25, 2009.

[11] Addison Wesley. An introduction to parallel computing . Addison Wesley, 2003.

otrzymano / received: 21.09.2010
przyjęto do druku / accepted: 01.11.2010 *artykuł recenzowany*