PARALLEL FACIAL RECOGNITION SYSTEM BASED ON 2DHMM

Janusz Bobulski
Institute of Computer and Information Science,
Czestochowa University of Technology,
73 Dabrowskiego Str., Czestochowa, Poland
januszb@icis.pcz.pl.pl

Abstract. The constantly growing amount of digital data more and more often requires applying increasingly efficient systems for processing them. Increase the performance of individual processors has reached its upper limit therefore we need to build multiprocessor systems. To exploit the potential of such systems, it is necessary to use parallel computing, i.e. creating computer systems based on parallel programming. In practice, most often their used adjusts the parallelization of the data processes, or regulates the parallelization of the query tasks. The system of face recognition that requires high computational power is one of potential application of the computations parallelization, especially for large database sizes. The aim of the research was to develop a parallel system of face recognition based on two-dimensional hidden Markov models. The results show that compared to sequential calculations, the best results were obtained for parallelization of tasks, and acceleration for training mode was 3.3 and for test mode - 2.8.

Keywords: parallelization, face recognition, 2D hidden Markov models, HMM, parallel task.

1 Introduction

The system of face recognition that requires high computational power is one of potential application of the computations parallelization, especially for large database sizes. It is forcing producers of computers to the prospecting of solutions which will be able to satisfy current and prospective needs. The evolution of parallel architectures has an intense influence on all computer systems, starting from smartphones until computing systems of the great scale. In particular, the multi-core became a main solution enabling to sustain the law Moore'a that is of exponential increasing the productivity of the computer what is being carried out above all by increasing the number of cores. A computational productivity and a demand for the electric energy are important factors affecting direction of the development of designs of parallel processors [1, 2]. So computational new architectures are being planned, of which characteristics are taking the liberty of getting the greater computing power at keeping or lowering the demand to the electric power. Increasing the frequency of the clocking was one of directions of the

Advances in Intelligent Systems and Computing, Volume 534, 2017, Pages 258-265

development [3]. Unfortunately, the technology based on silicon in this respect achieved ones of limits. Increasing the number of cores was another trend of the development of processors of the general-purpose. Here a problem with the access to shared sources and the time and the manner of the transport appeared between cores. Functioning of processors producers show that future computational architectures in their nature will be solutions hybrid, finding application in very wide range. Hybrid, and at the same time heterogeneous systems, will be based on integration of two main kinds of components, multi-core processors of the general-purpose and dedicated, massively parallel computational accelerators. The virtue multi-core processors of the general-purpose is the fact that the substantial amount of cores in the uniform integrated circuit lets for increasing productivities is without raising frequencies of the clocking of the core, and consequently, without increasing the generation of the heat, what is being transferred on smaller power consumption. However, dedicated massively parallel computational accelerators, so as GPU overtook CPU processors in the productivity of the floating-point calculation. Applying two basic computational components in hybrid architectures [2, 4] in the form of CPU and computational accelerators is enjoying considerable influence to the productivity growth of numerical calculations. Traditional processors perform well best in carrying programs out with numerous branches, in which relatively it is necessary to process the little amount of data according to the complicated algorithm. However computational accelerators let the high performance getting indeed for the part of application, because are designed in order productively to carry the same set of the operation out on the large amount of data.

2 Parallel computing

Contemporary processors of the general-purpose are ensuring excellent abilities for the parallel processing, letting for creating the application which the productivity of calculations will bring closer oneself to the peak performance of the arrangement. However exploiting the full potential of these processors constitutes the greatest challenge for programmers. The most popular methods of code parallelization are parallel data processing and task parallel processing [5, 6].

The model of data parallel processing is defining calculations as the sequence of instructions marked to a lot of elements put in the memory. The index space connected with the model of the workmanship is defining threads and the way of connecting data with them. The real model of parallel processed data assumes copying one to one between the weft and the element in the memory, on which kernel may parallel be made. Two forms of the realization of the model of data parallel processing are possible: (i) open form, programmer is defining the total number of executed parallel threads as well as a manner of the division into groups is describing by them, (ii) secret form, programmer he determines only a total number of executed parallel threads, however a programming environment is managing the division into groups.

The model of task parallel processing is establishing, that single authority kernel is being made irrespective of the index space. This attempt is analogous to the process of making kernel on the computational individual, where every group of threads contains only one thread. In this model the programmer is defining levels of the parallelism

through: using vector supported data types by the device, performing many tasks simultaneously and workmanship kernels of other computational programming models, drawn up on the base [7].

Both recalled techniques parallelization of code were used for the parallel implementation of 2DHMM.

3 Evaluation of the effectiveness of parallel processing

The evaluation of the productivity of parallel programs requires applying a few certificates what allows for efficiency analysis of complex problems. It is possible to rank among most often used metrics [8, 9]:

- execution time of calculation T_{obl}
- theoretical maximum performance R_{max};
- real performance R;
- speed up the S_p
- the efficiency E_p
- granularity G;
- memory bandwidth and I/O bus B.

The execution time of the parallel program is a basic parameter of the evaluation of the productivity of calculations. This time is being measured from the moment of commencing calculations all the way to the moment, which all performed calculations will end in. In the parallel algorithm it is possible to accept the lead time of calculations:

$$T_{obl} = T_s + T_p + T_c + T_i \tag{1}$$

where T_s is a execution time of the sequential part of the code, T_p is a execution time of the parallel part, T_c - time of the communication, and T_i it is a time of the idleness.

The theoretical maximum performance multiprocessor system can be determined according to the equation:

$$R_{max} = f \cdot N \cdot I \tag{2}$$

where f - frequency of the clocking of the processor, N - number of cores, I - the product of the number of elements of the vector and numbers of operations performed on one element of the vector.

The real performance is used to compare the get productivity with theoretical possibilities of the computer system. It is expressed with number of floating-point operations for a second (flop/s). A F_h number of indeed performed floating-point operations determines the real performance in the given time. This value, determined with formula:

$$R = \frac{F_h}{T_{obl}} \tag{3}$$

and may never exceed the maximum theoretical performance of the computer.

Speedup S of the parallel algorithm on p processors, at the permanent size of the considered problem it is possible to define as:

$$S_p = \frac{T_1}{T_p} \tag{4}$$

where T_1 is a execution time of the algorithm executed on one individual processor ($p=1$), T_p is a execution time of the parallel algorithm carried out on p processors. Effectiveness E of parallel system determines the degree to which were used computational units:

$$E_p = \frac{S_p}{p} \tag{5}$$

The granularity of the algorithm is useful when selecting a parallel algorithm for a particular architecture. The granularity may be defined as the ratio of computation time T_{ref} to the time of communication T_{kom}, necessary for the implementation of the problem:

$$G = \frac{T_{obl}}{T_{kom}} \tag{6}$$

Memory bandwidth and I/O bus is an important parameter to judge performance of parallel algorithms. It may play a significant role in the case of applications that require large amounts of memory reference.

$$B_M = f \cdot M_i \cdot 2 \tag{7}$$

where f – frequency clock memory [Hz], M_i - the width of the memory interface [B], 2 – double way transfer.

4 2DHMM

There are two fundamental problems of interest that must be solved for 2DHMM, in order that be useful in face recognition applications. These problems are efficiently compute P(O|λ) and estimate model parameters λ = (A, B, π). Solutions to these problems are algorithms forward, backward and Baum-Welch [10].

Forward algorithm

$$\alpha_{t+1}(i,j,k) = \left[\sum_{l=1}^{N^2} \alpha_t(i,j,k) a_{ijl} \right] b_{ij}(o_{t+1}) \tag{8}$$

$$P(O|\lambda) = \sum_{i,j,k} \alpha_T(i,j,k) \tag{9}$$

Backward algorithm

$$\beta_t(i,j,k) = \sum_{l=1}^{N^2} a_{ijl} b_{ij}(o_{t+1}) \beta_{t+1}(i,j,k) \tag{10}$$

Baum-Welch algorithm

$$\xi_t(i,j,l) = \frac{\alpha_t(i,j,k)a_{ijl}b_{ij}(o_{t+1})\beta_{t+1}(i,j,k)}{P(O \mid \lambda)} = \\ = \frac{\alpha_t(i,j,k)a_{ijl}b_{ij}(o_{t+1})\beta_{t+1}(i,j,k)}{\sum_{k=1}^{K}\sum_{l=1}^{N^2}\alpha_t(i,j,k)a_{ijl}b_{ij}(o_{t+1})\beta_{t+1}(i,j,k)} \quad (11)$$

5 Parallel realization of 2DHMM face recognition system

For parallel realization of face recognition system based on 2DHMM we used the most popular methods of code parallelization are parallel data processing and task parallel processing. *Parfor* loop is used to implement data parallel processing, and execution of task parallel processing is shown in Figure 1 and 2.

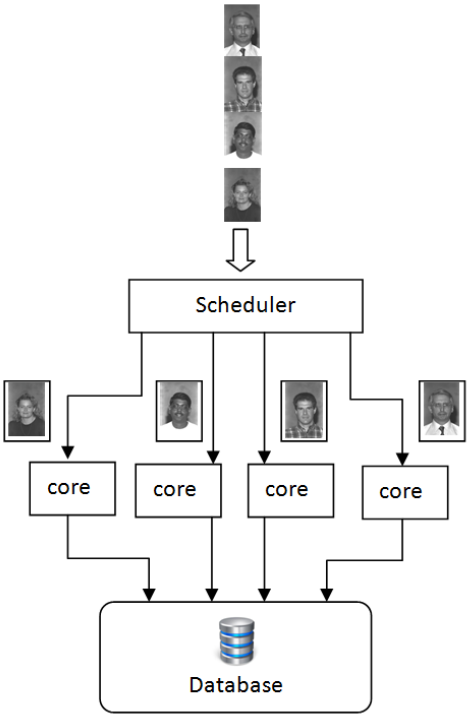

Fig. 1. Scheme of task parallel computing for learning

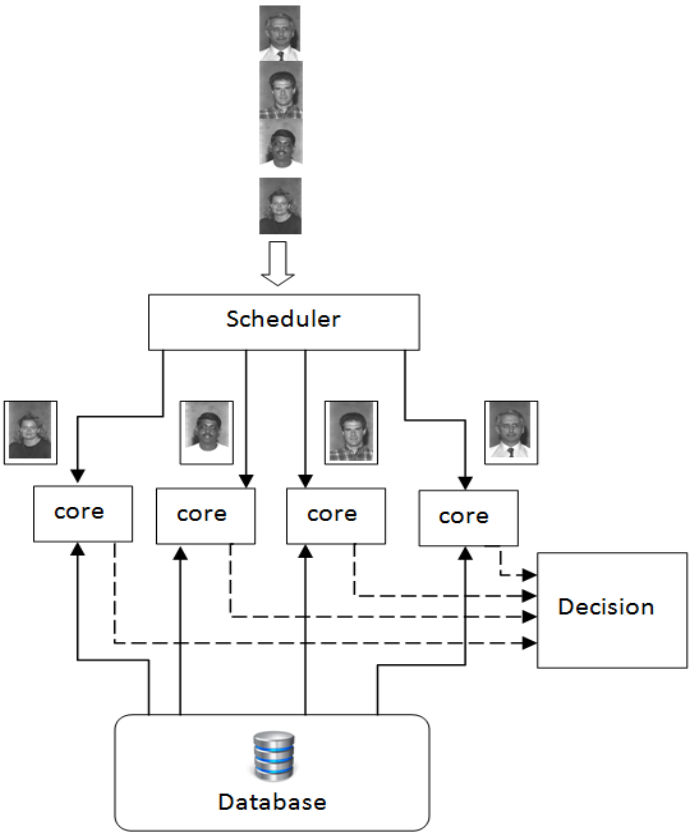

Fig. 2. Scheme of task parallel computing for testing

6 Experiment

The aim of the research was to develop a parallel system of face recognition based on two-dimensional hidden Markov models (2DHMM). The procedure of the person' identification used for feature extraction wavelet transform in this system. The feature vector obtained from this transformation is utilized for training and testing the system. In this method, for identification purposes, both 2D and 3D images of the face were exploited. The detailed description of this sequential method of the face recognition one may find in the article [10]. Because of long computation times in this method we decided to apply the parallel processing. The analysis of algorithms of the systems using 2DHMM allowed uses the parallel processing. The study of learning mode of the system allowed for the application the parallelization of data processing and the parallelization of tasks. While in test mode it was possible to uses only the parallelization of tasks. One task was the processing (training or testing) of one face image. For research we used the face base UMB-DB [11]. The experiment carried on the processor Intel i5

3.3 GHz with 4 cores and 4 threads, and the results of the experiment are presented in table 1.

The results show that compared to sequential calculations, the best results were obtained for parallelization of tasks. In the learning mode and the testing mode with the use of a 2D image, we got a low speedup of calculations, because the source data was much less. However for 3D images, 3.2 speedup was get for the learning mode, and 2.7 for the testing mode. In case of using for the identification both of image, 2D as well as 3D, speedup for training mode and test mode was respectively 3.3 and 2.8.

Table 1. Comparison of processing time

Type of processing	Type of face image	Time of learning [s]	Time of testing [s]
Sequential	2D	126	117
Sequential	3D	1145	1090
Sequential	2D+3D	1262	1282
DPP	2D	102	1632
DPP	3D	762	2117
DPP	2D+3D	1082	3776
TPP	2D	122	98
TPP	3D	359	401
TPP	2D+3D	386	451

DPP – data parallel processing
TPP – task parallel processing

To evaluate the efficiency of the implementation of parallel algorithms for learning and testing HMM used in computation time, acceleration and efficiency. The results are shown in Table 2 and 3.

Table 2. Speedup

Type of processing	Type of face image	Learning	Testing
DPP	2D	1.24	0.07
DPP	3D	1.50	0.51
DPP	2D+3D	1.17	0.34
TPP	2D	1.03	1.19
TPP	3D	3.19	2.72
TPP	2D+3D	3.27	2.84

Table 3. Efficiency

Type of processing	Type of face image	Learning	Testing
DPP	2D	0.31	0.02
DPP	3D	0.38	0.13
DPP	2D+3D	0.29	0.09
TPP	2D	0.26	0.30
TPP	3D	0.80	0.68
TPP	2D+3D	0.82	0.71

7 Conclusion

The use of data parallel processing with *parfor* loop did not give significant speedup calculations. This is due to the structure of training and testing algorithms of HMM, that they have in their structure the operations that do not allow parallelization of the calculation process.

In conclusion, when we are creating a biometric system, using a face image of individuals and it is based on 2DHMM, it is worth to use parallel processing with application parallelization of tasks. In such a system, each processor independently processes one object, i.e. one person. The result of applying a parallel structure system is a three-fold reduction in computation time.

References

1. Culler, D.E., Singh, J.P., Gupta, A.: Parallel Computer Architecture – A Hardware/Software Approach, Morgan Kaufmann Publishers (1999)
2. Kurzak, J., Bader, D., Dongara, J.: Scientific Computing with Multicore and Accelerators, Chapman & Hall/CRC Computer and Information Science Series (2010)
3. Vetter, J.: Keeneland: Bringing Heterogeneous GPU Computing to the Computational Science Community, Computing in Science & Engineering 13, 90-95 (2011)
4. Kurowski, K., Back W., Dubitzky, W., Gulyas L., Kampis, G., Manowski, M., Szemes, G., Swain, M.:Complex system symulation with QosCosGrid, 9th Int. Conf. on Computational Science, 387-396 (2009)
5. Blaziewicz, M.,Brandt, S., Kierzynka, M., Kurowski, K., Ludwiczak, B., Tao, J., Weglarz, J.: CaKernel – A paralel application programming Framework for heterogenous computing architectures, Scientific Programming, 19 (4), 185-197 (2011)
6. Czech, Z.: Wprowadzenie do obliczeń równoległych, Wydawnictwa Naukowe PWN (2010)
7. Wyrzykowski, R., Rojek, K.,Szustak, L.:Model-driven adaptation of double-precision matrix multiplication to the Cell processor architecture, Parallel Computing 38 (4), 260-276 (2012)
8. Hockney, R.W.: The Science of Computer Benchmarking, SIAM, Philadelphia (1995)
9. Wyrzykowski R.: Klastry komputerów PC I architektury wielordzeniowe: budowa i wykorzystanie, Exit, Warszawa (2009)
10. Bobulski J., 2DHMM-Based Face Recognition Method, Image Processing And Communications Challenges 7, Advances in Intelligent Systems and Computing 389, 11-18 (2016)
11. Colombo A., Cusano C., Schettini R., Umb-db: A database of partially occluded 3d faces, Proc. ICCV 2011 Workshops 1 (2011) 2113-2119.