Zbigniew HANDZEL

Uniwersytet Jagielloński w Krakowie

Implementing the rate monotonic scheduling algorithm for heterogeneous processors

Abstract. *The paper discusses task scheduling issues in real-time systems with hard time constraints. In the paper, we propose to use rate monotonic scheduling for heterogeneous multiprocessor systems. Moreover, for the sake of load balancing we propose to use a genetic algorithm.*

Streszczenie. *W artykule rozważono zagadnienia szeregowania zadań w systemach czasu rzeczywistego z ostrymi ograniczeniami czasowymi. Rozważono możliwości wykorzystania metody rate monotonic scheduling w przypadku heterogenicznych systemów wieloprocesorowych. Ponadto w celu równoważenia obciążenia poszczególnych jednostek obliczeniowych zaproponowano wykorzystanie algorytmu genetycznego. (**Implementacja algorytmu szeregowania według monotonicznego tempa dla heterogenicznych procesorów**).*

Keywords: task scheduling, multiprocessor systems, heterogeneous processors, genetic algorithms.
Słowa kluczowe: szeregowanie zadań, systemy wieloprocesorowe, procesory heterogeniczne, algorytmy genetyczne.

Introduction

At present, real-time computer systems constitute a separate and well-defined class of industrial computer systems which differ in many important aspects from the general purpose computer systems. In the case of real-time computer systems with hard time constraints, the time of task performance is the most crucial parameter which requires a special consideration. In the case of hard real-time systems, it is not enough that a given task delivers results which are logically correct, but additionally these results must be delivered on time. If the logically correct results obtained during the computational process are delivered after the predefined time constraints, the obtained results are no longer valid. Moreover, if the time constraints are not met in the case of the hard real-time system, this can result in severe damages, catastrophes and serious economic losses or even a loss of human lives [1].

This is the main reason why real-time systems with hard time constraints must be designed and developed with great attention. In many cases some formal methods must be used in order to ensure that all the tasks which are performed in the system will always meet their time constraints in any possible conditions and scenarios of the system operation [2].

These issues have contributed to a considerable development of task scheduling and allocation methods and theory, which provide answers to questions concerning where and when a given task must be performed in order to meet its time constraint. In the case of industrial hard real-time systems, the most commonly encountered tasks are independent, pre-emptive and periodic. At the beginning of the execution of each task the system reads the actual values of the input signals from the sensors, then some computations are performed according to the control algorithms, and in the end the computed values of control signals are sent to the actuators. Such tasks are performed periodically, i.e. for each task there is given the value of its period T and its performance time C, which is calculated for the worst case scenario. The C/T ratio determines the fraction of time which the processor spends on the performance of the given periodic task, so it is also a value of the processor load coefficient [3].

Until now, various real-time task scheduling algorithms have been proposed. In the case of a set of independent, pre-emptive and periodic tasks the most popular task scheduling algorithm, which is implemented in most of real-time operating systems, is the Rate Monotonic Scheduling (RMS).

Basic properties of rate monotonic scheduling

The rate monotonic scheduling algorithm belongs to the broader class of static task scheduling algorithms and is used for the purpose of scheduling a set of independent, pre-emptive and periodic tasks. For each task that is to be scheduled we must know the value of its period T and the worst case performance time C, so that the value of the processor load coefficient could be calculated as C/T [4].

The scheduling procedure is based on systems of priorities. The priorities are assigned to the task according to the rule that the shorter the period of the given task, the higher priority value it obtains. The reason for it is that the tasks with shorter values of their periods have less time to wait for the beginning of their performance, because they are very close to their time constraints, so their execution must begin as soon as possible in order to meet their time constraints. The only way to achieve this is to assign appropriately high values of priorities to the tasks with shorter values of their periods [5].

If more than one task is in the ready state, the task with the highest priority value is currently performed. If the task with the higher priority value enters into the ready state, the currently performed task is pre-empted and a new coming task is performed. The pre-empted task can be resumed only in the situation when there is no other task of a higher priority value in the ready state.

If the set of the tasks being scheduled is given and the characteristics of the tasks are known, an important question is whether the time constraints of all the tasks will always be met. This question is answered by the Liu and Layland theorem which is given by the following formula

$$\sum_{i=1}^{N} \frac{C_i}{T_i} \le N\left(2^{\frac{1}{N}} - 1\right) \tag{1}$$

In the inequality (1), N is the number of the tasks scheduled. The inequality (1) delivers only a sufficient condition for the set of schedulable tasks. However, the condition (1) is not a necessary condition for the set of schedulable tasks. Moreover, if the condition (1) is not fulfilled, it does not automatically follow that the set of tasks is not schedulable. In such a case one must first of all check whether the necessary condition is fulfilled. The necessary condition is given by the following formula

$$\sum_{i=1}^{N} \frac{C_i}{T_i} \le 1 \tag{2}$$

Moreover, for each task of the scheduled set of tasks it must be checked, whether their time constraints are met in the worst case scenario, i.e. under the conditions when all the tasks enter into the ready state at the same moment. If under the worst-case conditions the performance of all the scheduled tasks is ended before the elapse of their time constraints, it means that the given set of tasks is schedulable under any circumstances. In order to prove this, one has to calculate for each task the time of its execution end. If the time of the execution end of each task is shorter than its time deadline, it means that the set of tasks is schedulable [5].

To calculate the time of the execution end of a periodic task the recurrent formula can be used. If we consider the lowest priority task, then the first estimation of its time of the execution end is assumed as the sum of its execution time and the times of execution of all the other tasks. This results from the fact that before the execution of the lowest priority task can be started, all the other tasks must be performed at least once. Thus, the first estimation of the execution end time of a task is given by the following formula

$$t_0 = \sum_{i=1}^{N} C_i \tag{3}$$

Then, we must systematically repeat the recurrent procedure, which is given by the following formula

$$t_{m+1} = \sum_{i=1}^{N} C_i \cdot \left\lceil \frac{t_m}{T_i} \right\rceil \tag{4}$$

In the formula (4), the symbol $\lceil X \rceil$ denotes the smallest natural number which is greater than or equal to X. The recurrent procedure is repeated until the following condition is fulfilled

$$t_{m+1} = t_m \tag{5}$$

In such a case we consider time t_m as the time of the execution end of the lowest priority task. If this time is shorter than the deadline of the lowest priority task, we can consider this task schedulable under any circumstances, because it proved to be schedulable in the worst-case scenario.

The recurrent procedure, which is discussed above, must be repeated for all the tasks and all the tasks in the worst-case scenario must be proved to be able to end their executions before the elapse of their deadlines. Only if this condition is met, the given set of periodic tasks may be considered schedulable.

In its original form the RMS algorithm was proposed for the purpose of scheduling a set of periodic, pre-emptive and independent tasks only for one single processor. The paper puts forward a proposition of application of the RMS algorithm also for multiprocessor systems.

Task scheduling for multiprocessor systems

Currently, the multiprocessor solutions are becoming more and more popular due to their capacity to supply greater amounts of computational power than single processor systems, which is absolutely necessary in many hard real-time applications. The multiprocessor systems can be divided into homogeneous and heterogeneous systems. In the homogeneous multiprocessor systems all processors available in the system have the same parameters and, what is especially relevant, they have the same computational power. On the other hand, in the heterogeneous multiprocessor systems the processors of the system differ in their properties and their levels of available computational power are significantly different.

In the paper, a three-processor heterogeneous system is considered. There is also given a set of N periodic, independent and pre-emptive tasks that are to be scheduled onto the three-processor heterogeneous system. The tasks are denoted as: z_1, z_2, z_3, ... , z_N. The periods T_i (i = 1, 2, 3,..., N) are known for each task. Likewise, known are the worst-case execution times for each task. The execution times are denoted for the first processor as C^{I}_i (i = 1, 2, 3,..., N), for the second processor as C^{II}_i (i = 1, 2, 3,... ,N), and for the third processor as C^{III}_i (i = 1, 2, 3,..., N). Thus, we obtain three different values of processor load coefficients: C^{I}_i/T_i for the first processor, C^{II}_i/T_i for the second processor, and C^{III}_i/T_i for the third processor.

The objective is to find such a task allocation scheme that all the tasks could be scheduled with the RMS algorithm within the processors to which they are allocated. It is very significant to which processor each of the tasks is allocated. There are some allocation schemes that are not admissible, because there exists at least one task that is not schedulable, as it can not meet its deadline. On the other hand, there are such allocations that can guarantee that all the tasks are schedulable and their execution times elapse before their deadlines.

In order to find the task allocation scheme that is both schedulable and can guarantee a sufficient balancing of the values of processor load coefficients, we propose to use a computational technique of evolutionary algorithms [6].
In order to implement a computational technique based on the evolutionary algorithm two main factors should be determined [7]. The first of them involves the mode of coding the solutions on the genetic material of the individuals [8, 9]. For this purpose we have chosen the mode of coding which is directly based on the natural number system, because it is simple to both implement and interpret [10 – 12].

Each of the tasks was associated with one gene. The value of the genes could only be equal to one, two, or three. If the value of the gene is equal to one, it means that the task which is associated with that gene must be allocated to the processor P1. Similarly, if the value of the gene is equal to two, it means that the task which is associated with that gene must be allocated to the processor P2. And finally, if the value of the gene is equal to three, it means that the task which is associated with that gene must be allocated to the processor P3.

Another relevant matter concerns an adequate selection of the fitness function formula [13]. We define the fitness function as a sum of the values of time by which the deadlines of the tasks are exceeded. If all the tasks meet their deadlines, then the value of the fitness function is equal to zero [14]. In any other cases if at least one task exceeds its deadline, the value of the fitness function is greater than zero. The aim of the evolutionary algorithm is to find such an allocation scheme of the tasks for which the fitness function is equal to zero [15, 16].

The initial population of the evolutionary algorithm was generated randomly and it was composed of 100 individuals. The individuals underwent genetic operations of mutation and selection. During the mutation operation the randomly chosen gene obtained a random value of one, two, or three, so the allocation scheme of the task associated with this gene was changed randomly. The genetic operation of selection was realized as a tournament selection, during which the individuals were coupled into pairs and from each pair only the individual of the lower value of the fitness function passed to the next generation.

After the elapse of a few hundred of generations, the admissible tasks allocation schemes were obtained. The following subsets of tasks were allocated to three heterogeneous processors P1, P2 and P3.

In Fig. 1 we present a plot of load coefficient values for the processor P1, which were obtained with the use of the evolutionary algorithm. It can be concluded that the convergence of the evolutionary algorithm is sufficiently good, which allows to find the appropriate solution after the elapse of a relatively short period of time.

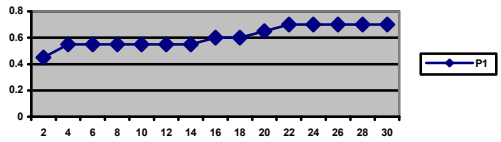

Fig. 1. Plot of load coefficient values obtained for the processor P1

Similarly, in Fig. 2 we present load coefficient values obtained for the processor P2. Also, in Fig. 3 we present load coefficient values obtained for the processor P3.

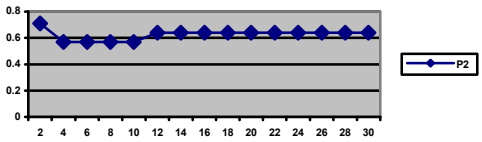

Fig. 2. Plot of load coefficient values obtained for the processor P2

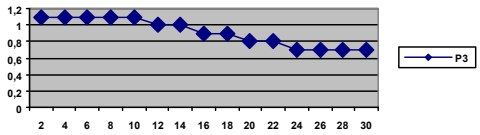

Fig. 3. Plot of load coefficient values obtained for the processor P3

Sumary

In the paper we have demonstrated that a computational technique based on the use of evolutionary algorithms can be effectively implemented for the purpose of task allocation in a multiprocessor system. The allocation schemes that are found by the evolutionary algorithm are admissible, because the hard time constraints are met for each task.

The obtained results can be easily extended to cover the cases of any number of heterogeneous processors simply by changing the mode of coding the solutions on the genetic material of the individuals. Moreover, the case of homogeneous processors can be treated as a special case of heterogeneous processors such that the computational power of all processors is equal. Thus the results obtained for the case of heterogeneous processors are more general and can be easily extended also to the case of homogeneous processors.

Further work in this domain will concentrate on the extension of the obtained results to include multiprocessor tasks [17]. Multiprocessor tasks are tasks that need access to more than one processor at the same time in order to be executed [18]. The issues related to multiprocessor task scheduling are far more complicated than scheduling single-processor tasks and require the development of appropriate methods for multiprocessor task allocations in multiprocessor systems [19]. It is of relevance to notice that the rate monotonic scheduling algorithm can also be extended to multiprocessor tasks, which is relatively straightforward in the case of homogeneous multiprocessor systems but not in the case of heterogeneous processors.

REFERENCES

[1] Hsueh W., Lin K. J., Scheduling real-time systems with end-to-end timing constraints using the distributed pinwheel model, *IEEE Transactions on Computers*, vol. 50, n. 1 (2001), 51-67

[2] Lala J. H., Harper R. E., Architectural principles for safety-critical real-time applications, *Proceedings of the IEEE*, vol. 82, n. 1 (1994), 25-41

[3] Ramamritham K., Stankovic J. A., Scheduling algorithms and operating systems support for real-time systems, *Proceedings of the IEEE*, vol. 82, n. 1 (1994), 55-67

[4] Shin K. G., Ramanathan P., Real-time computing: A new discipline of computer science and engineering, *Proceedings of the IEEE*, vol. 82, no. 1 (1994), 6-24

[5] Sha L., Rajkumar R., Sathaye S. S., Generalized rate-monotonic scheduling theory: A framework for developing real-time systems, *Proceedings of the IEEE*, vol. 82, n. 1 (1994),68-82

[6] Gajer M., Accelerating the rate of evolutionary processes with the use of constant learning, *Przeglad Elektrotechniczny*, 87 (2011), n. 1, 204-209

[7] Gajer M., Implementation of evolutionary algorithms in the discipline of Artificial Chemistry, *Electrical Review*, 87 (2011), n. 4, 198-202

[8] Gajer M., The implementation of the evolutionary computations in the domain of electrical circuits theory, *Przeglad Elektrotechniczny*, 87 (2011), n. 6, 150-153

[9] Gajer M., Visualization of particle swarm dynamics with the use of Virtual Reality Modeling Language, *Przeglad Elektrotechniczny*, 87 (2011), n. 11, 20-24

[10] Gajer M., The analysis of impact of learning on the rate of evolution in the case of a multimodal fitness function, *Przeglad Elektrotechniczny*, 86 (2010), n. 2, 24-29

[11] Gajer M., The implementation of the evolutionary algorithm for the analysis of nonlinear electrical circuits, *Przeglad Elektrotechniczny*, 86 (2010), n. 7, 342-345

[12] Gajer M., The optimization of power flow in high-voltage transmission lines with the use of the evolutionary algorithm, *Electrical Review*, 86 (2010), n. 8, 239-244

[13] Gajer M., The optimization of load distribution with the use of the evolutionary algorithm, *Electrical Review*, 86 (2010), n. 11a, 265-270

[14] Gajer M., Task scheduling in real-time computer systems with the use of an evolutionary computations technique, *Przeglad Elektrotechniczny*, 86 (2010), n. 10, 293-298

[15] Gajer M., Determining the working points of bipolar transistors with the use of the evolutionary strategy, *Przeglad Elektrotechniczny*, 87 (2011), n. 12a, 124-128

[16] Gajer M., Reduction of thermal transmission losses with the implementation of a genetic algorithm, *Przeglad Elektrotechniczny*, 88 (2012), n. 3a, 129-130

[17] Stanley P. Y., Chung K. P., Duncan K. W., On-line scheduling of equal-length intervals on parallel machines, *Information Processing Letters*, 112 (2012), 376-379

[18] Aspnes J., Yitong Y., Randomized load balancing by joining and splitting bins, *Information Processing Letters*, 112 (2012), 309-313

[19] Wenhua L., Zhenkun Z., Sufang Y. W., Online algorithms for scheduling unit length jobs on parallel-batch machines with lookahead, *Information Processing Letters*, 112 (2012), 292-297

[20] Handzel Z., Latawiec K. J., A simulation approach to statistical evaluation of periodic task scheduling in a real-time computer system *Methods and Models in Automation and Robotics (MMAR), 2010 15th International Conference on Digital Object Identifier*: (2010) , 271 – 274

[21] Manabe Y., Aoyagi S. - A Feasibility Decision Algorithm for Rate Monotonie Scheduling of Periodic Real-Time Tasks, *Proceedings of the IEEE*, vol. 95, n. 1 (1995), 212-218

***Author**: dr inż. Zbigniew Handzel, Uniwersytet Jagielloński w Krakowie, Wydział Zarządzania i Komunikacji Społecznej, ul. Prof. S. Łojasiewicza 4, 30-348 Kraków, E-mail: zbigniew.handzel@uj.edu.pl*