BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 62, No. 4, 2014
DOI: 10.2478/bpasts-2014-0083

Probabilistic elements in analysis of performance of multiprocessor systems

K. TABOREK* and E. HRYNKIEWICZ

Silesian University of Technology, Department of Electronics, 16 Akademicka St., 44-100 Gliwice, Poland

Abstract. The paper presents important probabilistic elements that should be taken into consideration in the analysis of performance of classical multiprocessor systems. These elements represent the following quantities: modified arrival rate for processor requests and a few probabilities, which determine the frequency of certain events when a multiprocessor system is working. There are four peculiar events: service of another job, existence of the queue, a processor request while the given task is waiting into the queue and the return of another task into the queue while the given task is waiting in the queue. The first three events happen more often when a system consists of less number of processors, whereas the fourth event happens more often when more processors work in a system. Including (or not) the probabilities of these events to the analysis of performance of multiprocessor systems exerts its much influence on the precision of computations. All the mentioned quantities were described in detail. Formulas for these quantities were derived. Examples of applications of the formulas to the prediction of performance of various multiprocessor systems were presented.

Key words: arbitration circuit, multiprocessor system, performance analysis, queueing model.

1. Introduction

With the passing of time multiprocessor systems have been developed in many directions. Today, we have various kinds of multiprocessor system architectures. There are multiprocessor systems with common memory and with distributed memory. Both the mentioned types of multiprocessor systems we may further divide into other types of systems. For instance, we can divide the multiprocessor systems with common memory into systems with a single bus and with multiple buses. Additionally, the common memory may be divided into modules etc.

There are many publications in which different methods of analysis of multiprocessor systems are presented. The subject of such analysis is often cache memory. The size of this memory exerts its influence on performance of the whole system [1–4].

In many papers there are methods of performance analysis for multiprocessor systems with multiple buses or crossbar switches [5, 6].

Various architectures of multiprocessor systems are assessed whether they suit peculiar software applications [7–9]. There are even publications in which performance prediction methods of very program-loaded multiprocessor systems are presented. Here, the systems are examined paying special attention to executing as largest number of applications as possible [10].

It would be very useful to have an accurate method of analysis of multiprocessor system performance and thanks to this method we could examine also an influence of other valid components of multiprocessor system on its performance. In particular we could predict performance of a multiprocessor system depending on the type of an arbitration circuit. Such a method for classical multiprocessor systems is presented in this paper. This method includes special probabilistic elements – probabilities of particular cases. These elements increase its accuracy.

2. A classical multiprocessor system and its performance

Let us consider a classical microprocessor system as it is shown in Fig. 1.

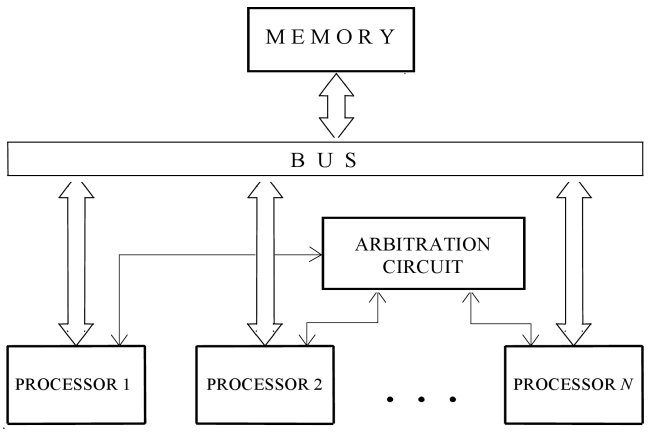

Fig. 1. A classical multiprocessor system

There are N processors and only one memory. The memory is common for all the processors in the system. The processors are connected to the memory through a single time-sharing bus. In one moment even a few processors can send their requests to access the memory. Of course, at one

*e-mail: ktaborek@polsl.pl

time only one processor can transfer data through the bus. Thus, the bus is a bottleneck of the whole system. A special circuit assigns the bus to one of the requested processors. This circuit is called arbitration circuit or simply arbiter. It selects one processor from among the requested ones using an arbitration algorithm (protocol or discipline). Usually, priority protocols are used in multiprocessor systems.

The most important feature of each multiprocessor system is its performance. A special index is used to determine the performance of multiprocessor systems [11]. This index is called speed-up and is defined as a ratio of times

$$S_{upN} = \frac{T_1}{T_N}, \tag{1}$$

where T_1 – execution time of a program in the system with single processor, T_N – execution time of the same program in the system with N processors.

It is assumed that the program can be divided into parallel components and each of which can be executed independently of the others.

If a multiprocessor system is the real system i.e. it is already built, we can simply measure the execution times of a program and to put them in Eq. (1).

As it turns out we can also calculate (predict) these execution times using the queueing theory.

3. The assumed queueing model and mean waiting times

The first step in our analysis of a multiprocessor system was acceptance of a suitable queueing model for the system. The assumed model is shown in Fig. 2.

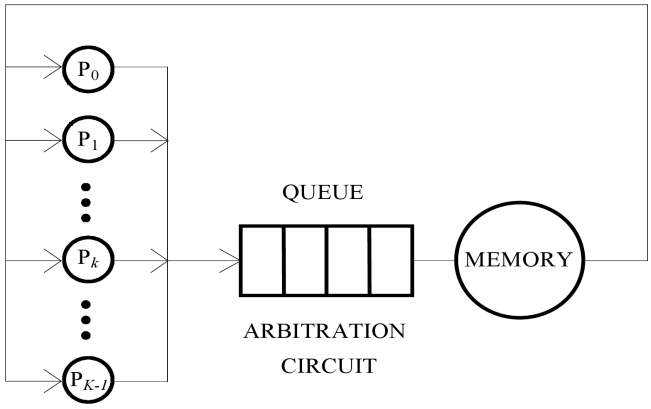

Fig. 2. The assumed queueing model

The presented network consists of multiple servers, one queue and one additional server. The multiple servers correspond to processors working parallel in the system. Further, these multiple servers will be called sources. The queue models the arbitration circuit. This circuit controls an access to the bus when processors want to communicate with the memory. The additional server corresponds to the common memory. Here a processor reads or writes data. The processor executes its machine cycles. If we make the assumption that all the processors in the system are the same type and they operate with the same clock frequency (homogeneous system), then these cycles are always equal and we have constant customer service time in the server.

In the presented queueing model the number of customers is constant. It is equal to the number of all the possible processor requests. Because each processor can send only one its request at the same time, the number of customers in the network is equal to the number of processors in the system. Here we must say that requests of processors may have various priorities, therefore we must divide all the customers into different types (classes). Let us denote the number of priorities by K. If each processor has got its priority that is different from the others, then the number of priorities is equal to the number of processors in the system, that is $K = N$.

For easier analysis we may also assume that all the times of staying of customers in their sources have got exponential distributions.

According to the Kendall notation [12] we can describe this queueing network as M/D/1//1.

We also assume that the queueing protocol includes non-pre-emptive disciplines only, i.e. a higher priority task cannot interrupt a lower priority one when its service has already begun.

Considering the total waiting time of a type k customer in the queue (the request of a priority k processor) we may notice that this time may be composed of three parts: $W_1^{(k)}$ – time of service completion of the customer which is just served, $W_2^{(k)}$ – time of services of the customers which have already been in the queue and they have got higher priorities then the type k customer, $W_3^{(k)}$ – time of services of the customers which have arrived into the queue during the type k customer is waiting and they have got higher priorities then the type k customer.

Hence, we can write

$$W^{(k)} = W_1^{(k)} + W_2^{(k)} + W_3^{(k)}. \tag{2}$$

Using mean quantities and taking into consideration that there are K customers in the system

$$E[W^{(k)}]_K = E[W_1^{(k)}]_K + E[W_2^{(k)}]_K + E[W_3^{(k)}]_K. \tag{3}$$

Applying proper procedures [13, 14] it is possible to derive the recursive formula given by Eq. (4). Thanks to it we can compute mean waiting times of a type k customer in the queue.

$$\begin{gathered} E[W^{(0)}]_K = p_{1K}\frac{B}{2}, \\ E[W^{(k)}]_K = E[W^{(k-1)}]_K \\ \cdot \frac{1 - p_{3K}B\sum\limits_{j=0}^{k-2} p_{zK}^{(jk)}\lambda^{*(j)} + p_{2K}B\lambda^{*(k-1)}}{1 - p_{3K}B\sum\limits_{j=0}^{k-1} p_{zK}^{(jk)}\lambda^{*(j)}}, \end{gathered} \tag{4}$$

where K – the number of all customers (we have assumed that the number of priorities is equal to the number of proces-

sors, that is $K = N$), B – constant customer service time in the server (it is equal to the processor machine cycle), $\lambda^{*(k)}$ – modified arrival rate of a type k customer (modified intensity of requests of a priority k processor), p_{1K}, p_{2K}, p_{3K}, $p_{zK}^{(jk)}$ – probabilities of certain cases.

Equation (4) has the essential meaning for further computations. However, in this formula there are a few quantities which are either inconvenient to using or require an additional description.

4. Modified intensity of processor requests

The arrival rate $\lambda^{*(k)}$ given in Eq. (4) is not convenient for use. It is related to the input of the queue. It would be significantly better if we used the arrival rate related to the source of a type k customer. These kinds of arrival rates are just closely depended on program jobs of processors and it is possible to determine them numerically. Further, for a type k customer this arrival rate will be called the arrival rate for a type k customer or for a priority k processor or in other words the request intensity of processor k. It will be denoted by $\lambda^{(k)}$ (without any asterisk).

Let us consider the interarrival time for a type k customer. This time is equal to the sum of three partial times:

- staying time of the type k customer in its source (the inverse of the arrival rate for the priority k processor),
- constant service time of this customer in the server,
- mean waiting time of this customer in the queue.

Arrival rate at the input of the queue is the inverse of this interarrival time. Thus, we may define our modified intensity as follows:

$$\begin{aligned}\lambda^{*(k)} &= \frac{1}{\dfrac{1}{\lambda^{(k)}} + B + E[W^{(k)}]_K} \\ &= \frac{\lambda^{(k)}}{1 + B\lambda^{(k)} + \lambda^{(k)} E[W^{(k)}]_K}.\end{aligned} \tag{5}$$

If $k = 0$, then mean waiting time of the highest priority customer is known and then we can compute $\lambda^{*(0)}$. If we put this quantity in Eq. (4), then we will obtain $E[W^{(1)}]_K$. Next we can compute $\lambda^{*(1)}$, etc.

How we see, quantities $\lambda^{*(k)}$ are dependent on K. Therefore these quantities must be computed separately for systems with various numbers of processors.

5. Probability of another job service

Equation (4) includes a probability which is denoted by p_{1K}. It is the probability of another job service. In other words, it is the probability of the case when a customer arrives to the queue during another customer is still serving. It corresponds to that situation in which there is at least one another customer in the service center. Our service center consists of a queue and a single server. In [13] we can find a formula for the probability that the $M/M/1//K$ service center is empty, i.e. there are no customers in it. The formula is as follows:

$$p_K(0) = \frac{1}{\sum\limits_{i=0}^{K} \left(\dfrac{\lambda}{\mu}\right)^i \dfrac{K!}{(K-i)!}}. \tag{6}$$

In Eq. (6) there is service rate μ. In our case we may replace this service rate with the constant quantity of service time according to relation $B = 1/\mu$.

In Eq. (6) there is also arrival rate λ of a single source. In our queueing model we have multiple sources (multiple servers). Here we may say that Poisson processes have got a very important property. If statistically independent Poisson processes are merged, the merged arrival process is also a Poisson process whose arrival rate is the sum of the arrival rates of the individual processes [15]. Therefore we may treat the arrival rate in Eq. (6) like the mean arrival rates of all sources in the system.

Of course, our sought after probability is the complement to one of the probability given by Eq. (6), so we may write

$$p_{1K} = 1 - p_K(0). \tag{7}$$

Taking Eqs. (6) and (7) and above-mentioned supplements into consideration we can finally write the following formula

$$p_{1K} = 1 - \frac{1}{\sum\limits_{i=0}^{K-1} [(K-1)B\lambda)]^i \dfrac{(K-1)!}{(K-1-i)!}}, \tag{8}$$

where

$$\lambda = \frac{1}{K} \sum_{k=0}^{K-1} \lambda^{(k)}. \tag{9}$$

Obtained from Eq. (8) numerical results are shown in Table 1.

Table 1
Values of probability p_{1K} versus the number of processors

K	P_{1K} for $B\lambda =$				
	0.01	0.1	0.2	0.4	0.8
1	0.00	0.00	0.00	0.00	0.00
2	0.01	0.09	0.17	0.29	0.44
3	0.04	0.32	0.53	0.74	0.89
4	0.09	0.62	0.84	0.96	0.99
5	0.15	0.85	0.97	1.00	1.00
6	0.24	0.96	1.00	1.00	1.00
7	0.33	0.99	1.00	1.00	1.00
8	0.44	1.00	1.00	1.00	1.00

The values of probability p_{1K} were computed for the systems with various numbers of processors and additionally for different values of relative intensity of processor requests$B\lambda$.

If $K = 1$, i.e. when only one processor works in the system, then we obtain $p_{11} = 0$. Indeed, this processor never waits to access memory.

If $K > 1$, then probability p_{1K} fast increases with K. The greater values $B\lambda$, the sooner p_{1K} increases.

6. Probability of existence of the queue

In Eq. (4) there is a probability which is denoted by p_{2K}. This probability is referred to that situation in which there are at least two another customers in the service center. One customer is just serving and potential others are waiting in the queue. This case is identical to real existence of the queue. In [13] we can find a formula for the probability that in the $M/M/1//K$ service center there are n customers. The formula is as follows:

$$p_K(n) = p_K(0)\left(\frac{\lambda}{\mu}\right)^n \frac{K!}{(K-n)!}. \tag{10}$$

Similarly, like in the case of Eq. (6) also in the case of Eq. (10) we may replace service rate μ with the constant quantity of service time according to relation $B = 1/\mu$. Also similarly, we may treat arrival rate λ in Eq. (10) like the mean of arrival rates of all sources in the system. This mean arrival rate is given by Eq. (9).

This time, our sought after probability is equal to the total probability reduced by probabilities of those situations when the service center is empty and there is only one customer in it. So we may write

$$\begin{aligned} p_{2K} &= 1 - p_K(0) - p_K(1) \\ &= 1 - \frac{1 + (K-1)^2 B\lambda}{\sum\limits_{i=0}^{K-1} [(K-1)B\lambda]^i \dfrac{(K-1)!}{(K-1-i)!}} \end{aligned} \tag{11}$$

Equation (11) also includes the obviousness that if a customer only just arrives to the service center, then the same customer cannot already be present in this service center.

Obtained from Eq. (11) numerical results are shown in Table 2.

Similarly as for the previous probability also the values of probability p_{2K} were computed for the systems with various numbers of processors and additionally for different values of relative intensity of processor requests $B\lambda$.

If $K = 1$ or $K = 2$, then we obtain $p_{21} = p_{22} = 0$. Indeed, for so small number of processors any requested processor will never come across another requests that could wait in the queue.

If $K > 2$, then probability p_{2K} increases fast with K although slower then for probability p_{1K}.

Also, the greater values $B\lambda$, the sooner p_{1K} increases.

Table 2
Values of probability p_{2K} versus the number of processors

K	p_{2K} for $B\lambda =$				
	0.01	0.1	0.2	0.4	0.8
1	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00
3	0.00	0.05	0.15	0.33	0.55
4	0.01	0.27	0.55	0.81	0.93
5	0.02	0.61	0.88	0.97	1.00
6	0.04	0.87	0.98	1.00	1.00
7	0.09	0.97	1.00	1.00	1.00
8	0.17	1.00	1.00	1.00	1.00

7. Probability of a processor request while another one is just waiting in the queue

Equation (4) includes a probability denoted by p_{3K}. It is the probability of the situation in which while a type k customer is just waiting, another customer arrives. First we will look at the situation from the point of view of the type k customer. This situation can be happened if this customer is waiting in the queue. This condition was already defined by probability p_{1K}. But the point of view of the second customer is different. For this customer the necessary condition of this situation is existence of the queue. This condition is defined by probability p_{2K}. Of course, probability p_{2K} includes probability p_{1K} so, we may finally write

$$p_{3K} = p_{2K}. \tag{12}$$

8. Probability of a return of the served job while another request is still waiting in the queue

In Eq. (4) there is one more probability which is denoted by $p_{zK}^{(jk)}$. It is the probability that during waiting a type k customer in the queue another customer of type j, which was earlier in the queue, would return to the queue again still before the type k customer service. As it turns out this probability is the sum of two partial probabilities.

$$p_{zK}^{(jk)} = p_{zK}^{(j)} + p_{zK}^{(jk)'}. \tag{13}$$

The first element in Eq. (13) is the probability that a type j customer do not arrive to the service center (a priority j processor do not send its request). In other words the type j customer there is in its source. This customer stays in the source for the period of time whose value is determined by its arrival rate. Thus, the first element of our sought after probability is the relation of this time to the interarrival time of this customer.

$$\begin{aligned} p_{zK}^{(j)} &= \frac{\dfrac{1}{\lambda^{(j)}}}{\dfrac{1}{\lambda^{(j)}} + B + E[W^{(j)}]_K} \\ &= \frac{1}{1 + B\lambda^{(j)} + \lambda^{(j)} E[W^{(j)}]_K}. \end{aligned} \tag{14}$$

The second element in Eq. (13) is the probability that the type j customer, which was earlier in the queue, return to its source still before the service of a type k customer. This event can be happened in the inverse of situation than that determined by the first element. So, in Eq. (16) there is additionally a complement to 1.

Here we have the following situation. The priority j processor had sent its request to access the memory and then the priority k processor sent its request. We assume that priority j is greater than priority k. Thus, in the moment when the type k customer arrived the type j customer had already been waiting in the queue. We may assume that this period of time was equal to half the mean service time of the type j customer. On account of different priorities the type j customer was served first and it returned to its source. From this

moment on a probability exists that the type j customer will arrive to the queue again. Furthermore, if this arrival happens before the service of the type k customer, the type j customer will serve first again, etc. Of course, this situation will be the more feasible, if the more customers there are simultaneously in the queue.

We may write the following expression

$$\frac{E[W^{(k)}]_K - \frac{1}{2}(E[W^{(j)}]_K + B)}{E[W^{(k)}]_K}. \tag{15}$$

It is relation of the mean period of time in which existing of the probability of a next arrival of the type j customer while the type k customer is still waiting to the mean time waiting of the type k customer in the queue.

Taking the above consideration and Eq. (15) we can write equation for the second element in Eq. (13) as follows

$$p_{zK}^{(jk)'} = (1 - p_{zK}^{(j)}) \left(\frac{E[W^{(k)}]_K - \frac{1}{2}(E[W^{(j)}]_K + B)}{E[W^{(k)}]_K} \right). \tag{16}$$

In Eq. (16) there is a mean waiting time of the type k customer. That time is still unknown to us. However, we may say that for a given system the mean waiting times for lower priority customers are more and more longer. So, let us write

$$E[W^{(k)}]_K = E[W^{(k-1)}]_K + \Delta. \tag{17}$$

In Eq. (17) there is an error that is denoted by Δ. To reduce this error to a minimum we may assume with an approximation that it is the difference between the mean waiting times of a type $k-1$ customer and a type $k-2$ customer. Taking this assumption and Eqs. (16) and (17) into consideration we may write as follows

$$p_{zK}^{(jk)'} \cong (1 - p_{zK}^{(j)}) \left(1 - \frac{E[W^{(j)}]_K + B}{2(2E[W^{(k-1)}]_K - E[W^{(k-2)}]_K)} \right). \tag{18}$$

Equation (18) is valid if $K > 2$, $1 < k < K$ and $j < k$.

Because of the above approximation we must make yet another condition as follows

$$E[W^{(j)}]_K + B < 2(2E[W^{(k-1)}]_K - E[W^{(k-2)}]_K). \tag{19}$$

The condition from Eq. (19) must be met because obtained from Eq. (18) numerical values of probabilities must not be negative. If the condition from Eq. (19) is not met, we must make that $p_{zK}^{(jk)'} = 0$.

For $k = 0$, then the mean waiting time in the queue of the highest priority customer $E[W^{(0)}]_K$ is known. Thanks to this we can compute $p_{zK}^{(0)}$. When we place this value to Eq. (4), we will compute $E[W^{(1)}]_K$, thanks to this we can compute $p_{zK}^{(1)}$, etc.

Probabilities $p_{zK}^{(k)}$ are dependent on K. Therefore these probabilities must be computed separately for systems with various numbers of processors.

9. Results of experiments

Two examples of applications of Eq. (4) are presented bellow.

Example 1. Let us consider a classical multiprocessor system that is equipped with the arbitration circuit with fixed priorities. It means the algorithm which grants all processors fixed priorities was implemented in this arbitration circuit. All the processors have got different priorities.

In [16] was presented a procedure for computation of performance of a multiprocessor system with the fixed priority arbitration circuit. Of course, Eq. (4) is the essential component of this procedure. The obtained numerical results are shown in Table 1 and in a graphical form in Fig. 3.

Table 3
Computed and measured speed-ups versus number of processors (variable load and fixed priority arbiter)

K	$S_{upK\#-}$	$S_{upK\#-(m)}$
1	1.00	1.00
2	1.87	1.87
3	2.63	2.69
4	3.35	3.46
5	4.06	4.19
6	4.78	4.90
7	5.47	5.43
8	5.35	5.23

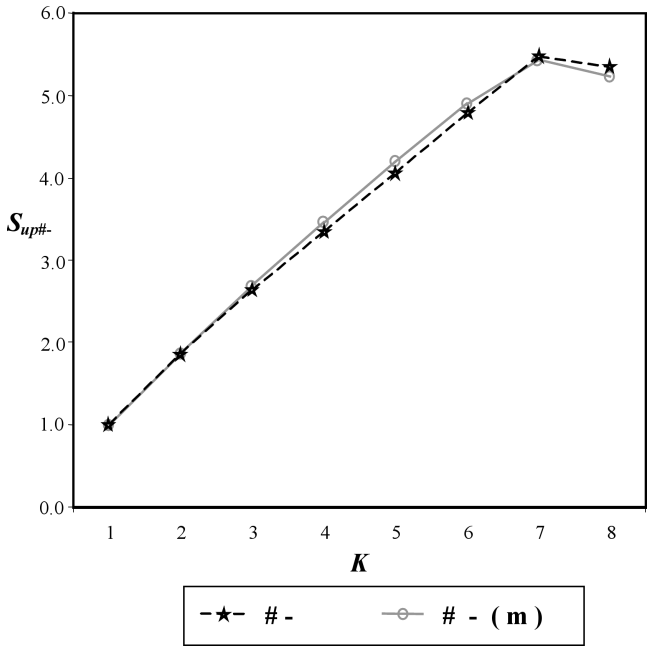

Fig. 3. Computed and measured speed-ups versus number of processors (variable load and fixed priority arbiter)

The obtained analytic results were verified in the real multiprocessor system [17]. Measurements are additionally denoted by (m).

The computations and measurements were made for variable load of processors in the system. The kind of this load is denoted by # and described in [18].

Example 2. Let us consider a classical multiprocessor system that is equipped with a special arbitration circuit. This arbiter we may called arbitration circuit with cyclically shifted priorities (further denoted by CP). It means the algorithm in which all priorities are changed during every service (when any processor is accessing the memory) was implemented in this arbitration circuit. At the beginning different priorities must be determined to all processors. When the selected processor is serving all the other processors in the system decrease their priorities, except the one which had got the lowest priority. The least important processor becomes the most important one in the system. The exchange of priorities of the processors during a service is shown graphically in Fig. 4.

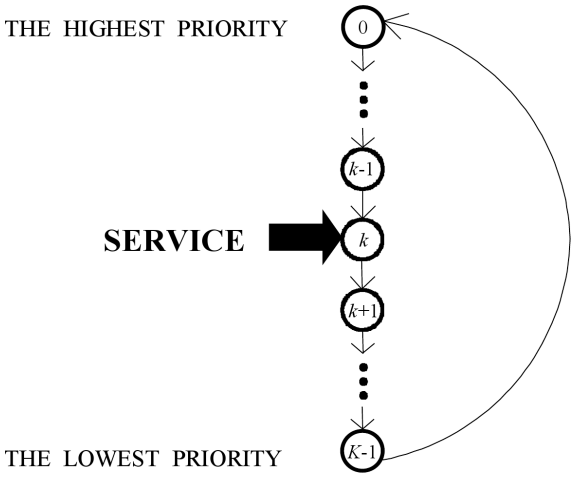

Fig. 4. Exchange of priorities for the CP discipline

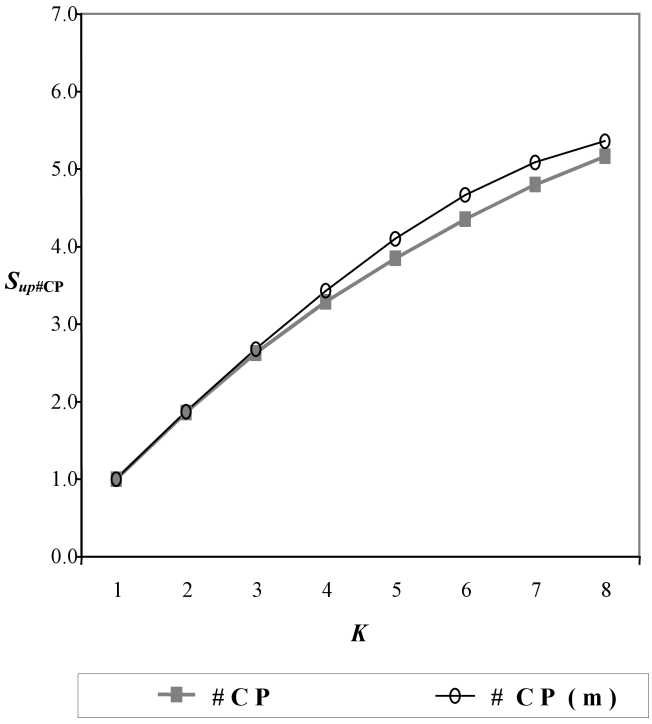

Fig. 5. Computed and measured speed-ups versus number of processors (variable load and CP arbiter)

All the processors have got different priorities.

In [19] was presented a procedure for computation of performance of a multiprocessor system with the CP arbitration circuit. Also here, Eq. (4) is the essential component of this procedure. The obtained numerical results are shown in Table 2 and in a graphical form in Fig. 5.

Table 4
Computed and measured speed-ups versus number of processors (variable load and CP arbiter)

K	$S_{upK\#CP}$	$S_{upK\#CP(m)}$
1	1.00	1.00
2	1.87	1.87
3	2.63	2.68
4	3.29	3.43
5	3.85	4.10
6	4.35	4.67
7	4.80	5.09
8	5.17	5.36

Similarly, the obtained analytic results were verified in the real multiprocessor system [17] and measurements are denoted by (m).

Also, the computations and measurements were made for variable load of processors working in the system. The kind of this load is denoted by #.

10. Conclusions

Equation (4) has the essential meaning for performance computations of classical multiprocessor systems. In order to increase the accuracy of the analytical results Eq. (4) was equipped with several additional quantities. These quantities are exactly described in this paper. Thanks to Eq. (4) we can analyse multiprocessor systems in various hardware configurations. For instance, we may analyse multiprocessor systems that are equipped with various arbitration circuits. As in the included examples the arbitration circuits may differ from their algorithms of request service of processors.

It is very important that it is not necessary to build real multiprocessor systems with their equipment in all configurations. On the base of presented computations we can say which of the considered systems achieve higher performance.

In the case of multi-core processors in which two or more central processing units (cores) work together on the same chip, we often deal with the phenomenon of fast increasing the temperature on some of the cores. We ought to take into consideration this phenomenon in physical structures of such multiprocessor systems. We should supply proper solutions to limit this disadvantageous phenomenon. For example, in one of the solutions we can use proper sensors to monitor temperature of the cores of a multi-core processor [20].

Here, we must say that this mentioned above aspect of increasing of temperature is not included in the presented analysis in this paper.

All the analytically obtained results of performance of multiprocessor systems were verified in the real multiprocessor system. This real system was equipped with a special

measuring circuit. Thanks to this circuit we could measure execution times of programs in this system. Of course, in the purpose of verification of the analytical results proper test programs were written so that, we ensured the same conditions as for the analytical computations.

On the base of the two included examples of application of Eq. (4) and the other ones [14, 21] we may say that Eq. (4) gives high precision of analytical results. For the presented results the maximum error is not greater than 3.5%. Programs destined to analysis or simulation of queueing networks usually achieve accuracy of between ten or twenty per cent. For instance, in [22] was presented a model of a similar multiprocessor system validated by system measurements. For this model the maximum error of analytical results is equal to 9%.

REFERENCES

[1] E. Berg, H. Zeffer, and E. Hagersten, "A statistical multiprocessor cache model", *IEEE Int. Symp. on Performance Analysis of Systems & Software* 1, 89–99 (2006).

[2] Chi Xu, Xi Chen, R.P. Dick, and Z.M. Mao, "Cache contention and application performance prediction for multi-core systems", *IEEE Int. Symp. on Performance Analysis of Systems & Software* 1, 76–86 (2010).

[3] J. Leverich, H. Arakida, A. Solomatnikov, A. Firoozshahian, M. Horowitz, and C. Kozyrakis, "Comparative evaluation of memory models for chip multiprocessors", *ACM Trans. on Architecture and Code Optimization* 5 (3), CD-ROM (2008).

[4] P. Prieto, V. Puente, and J.-A. Gregorio, "Multilevel cache modeling for chip-multiprocessor systems", *Computer Architecture Letters* 10 (2), 49–52 (2011).

[5] Kim Jongioon and A. El-Amawy, "Performance and architectural features of segmented multiple bus system", *Int. Conf. on Parallel Processing* 1, 154–161 (1999).

[6] C.-H. Tung and C.W. McCarron, "Analysis of a multiple bus multiprocessor", *26th Asilomar Conf. on Signals, Systems and Computers* 2, 925–929 (1992).

[7] I. Assayad and S. Yovine, "Performance analysis of embedded multiprocessor industrial applications: methodology and tools", *14th IEEE Int. Conf. Electronics, Circuits and Systems* 1, 907–910 (2007).

[8] S. Manolache, P. Eles, and Zebo Peng, "Schedulability analysis of multiprocessor real-time applications with stochastic task execution times", *ACM Int. Conf. on Computer Aided Design* 1, 699–706 (2002).

[9] J. Rosen, A. Andrei, P. Eles, and Zebo Peng, "Bus access optimization for predictable implementation of real-time applications on multiprocessor systems-on-chip", *28th IEEE Int. Real-Time Systems Symp.* 1, 49–60 (2007).

[10] A. Kumar, B. Mesman, H. Corporaal, and Yajun Ha, "Iterative probabilistic performance prediction for multi-application multiprocessor systems", *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems* 29 (4), 538–551 (2010).

[11] E. Gelenbe, *Multiprocessor Performance*, John Wiley & Sons Ltd., Chichester, 1989.

[12] D.G. Kendall, "Some Problems in the Theory of Queues", *J. Royal Statistical Society* B 13, 151–185 (1951).

[13] T. Czachórski, *Queueing Models in Performance Evaluation of Computer Networks and Systems*, Jacek Skalmierski's Computer Workshop, Gliwice, 1999, (in Polish).

[14] K. Taborek, *Arbitration Circuits for Multi-processor* Systems, *Doctor's Thesis*, Silesian Technical University, Gliwice, 2003, (in Polish).

[15] S. Lavenberg, *Computer Performance Modeling Handbook*, Academic Press, New York, 1983.

[16] K. Taborek, "An Analytical method of performance prediction of multiprocessor systems", *Electrical Review* 10, 72–75 (2011), (in Polish).

[17] K. Taborek and E. Hrynkiewicz "A multiprocessor system for arbitration circuit examination – hardware implementation", *Electronics* 9 (LI), 48–51 (2010), (in Polish).

[18] K. Taborek and Z. Pogoda, "Irregular load of processors in multiprocessor system", *Electronics* 10 (L), 60–63 (2009), (in Polish).

[19] K. Taborek and E. Hrynkiewicz, "Arbitration circuit with cyclically shifted priorities for multi-processor system", *3rd Int. IFAC Workshop on Discrete-Event System Design* 1, CD-ROM (2006).

[20] M. Frankiewicz and A. Kos, "Overheat protection circuit for high frequency processors", *Bull. Pol. Ac.: Tech.* 60 (1), 55–59 (2012).

[21] K. Taborek, "An analytical method for activity description of arbitration circuits with rotation of priorities", *Electronics* 10 (LIII), 76–78 (2012), (in Polish).

[22] Thing-Fong Tsuei and M.K. Vernon, "A Multiprocessor Bus Design Model Validated by System Measurement", *IEEE Trans. on Parallel and Distributed Systems* 6 (3), 712–727 (1992).