QUERIES SERVICE TIME RESEARCH AND ESTIMATION DURING INFORMATION EXCHANGE IN MULTIPROCESSOR SYSTEMS WITH "UNI BUS" INTERFACE AND SHARED MEMORY

Martyshkin A.I.[1]
Yasarevskaya O.N.[2]

1. Associate professor of the "Computing Machines and Systems" Chair of Penza State Technological University, PhD (Engineering Sciences), associate professor. E-Mail: alexey314@yandex.ru
2. Associate professor of the "Foreign Languages" Chair of Penza State Technological University, PhD (Pedagogy), associate professor. E-Mail: olga-appointment@yandex.ru

Abstract: The issues connected with estimating service time of queries (transactions) during the information exchange in multiprocessor systems with a unibus interface and shared memory are analyzed and studied in the article. The article aims at developing and making research of models based on systems and queueing networks, the "processor-memory" subsystem, as well as estimating the queries service time during the information exchange in multiprocessor systems with shared memory. The subject matter of the study is the analysis of time delays associated with conflict situations occured during the realization of interprocessor exchange when many processors turn to the exchange unibus and memory. The object of the article research is the "processor-memory" subsystem of existing multiprocessor systems and well-known versions of the architecture of this subsystem . The main task defined by the authors of the scientific article is to develop and make research of mathematical models of the "processor-memory" subsystem of the mentioned systems and to estimate the processing time of inputting queries during the information exchange in systems with shared memory. Mathematical models for carrying out queries service time research have been proposed. Equations for determining the main probabilistic-temporal characteristics of the "processor-memory" subsystem have been presented. The mentioned probabilistic-temporal models have been developed using the theory of queueing networks and probability theory. In conclusion the authors make the main judgements about the work done. The mathematical models studied in the article make it possible to estimate the main probabilistic-temporal characteristics of multiprocessor systems without developing real models or prototypes. As a result some effect is achieved, because it is possible to estimate the characteristics of new multiprocessor computer systems and choose the most optimal ones without creating a real expensive system.

Keywords: Simulation, Analytical Model, Imitation Model, Queueing Network System, Transaction, Read-Operation, Record-Operation, Multiprocessor System, "Processor-Memory" Subsystem, Memory Architecture, Memory Bandwidth, Memory Controller, Memory Latency, Buffer Element.

1. INTRODUCTION

The improvement of high-performance computing systems capacities (CSs) is directly connected with the increase of the speed and capacity of the memory being part of the system. Despite the rapid increase of the RAM bandwidth the indexes gap existing now in the "processor-memory" subsystem is not decreasing, but is increasing enormously. To obtain the necessary characteristics at the system design stage one should carry out an analysis of various variants of the structural organization. There are two main stages of the analysis: macroanalysis and microanalysis [17] which allow us to obtain quantitative and qualitative indexes of CSs. The usage of CSs analytical modeling methods for these purposes can be explained by the existence of delays of the computing process due to limited system resources. Multiprocessor systems (MPSs) are affected by this phenomenon most of all, where resources are shared not only between devices, as in the case of uni-processor CSs, but also between several central processors (CPUs) claiming an access to them.

There are many sources of delays in any CS: CPU, RAM, external storage devices, I / O channels, exchange pathways, etc. The research subject of this article is to analyze the delays associated with conflicts arising during the interprocessor exchange when a set of CPUs have an access to the unibus (UB) and memory. The main goal of the article is to develop mathematical models of the "processor-memory" subsystem of MPSs and, in addition, to estimate the processing time of inputting transactions while exchanging data in systems with shared memory. This issue is relevant because of the CSs developers' constant desire to deminish the problem of, the so-called, "bottlenecks", such as communication devices (an obviuos example is UB). The paper deals with the analytical modeling of MPSs with the bus architecture and uni-(shared) memory. There are two types of such a memory actively used in modern MPSs: lumped and distributed [2]. The lumped memory provides the same access time for all the CPUs, so the MPSs with this memory organization is called symmetric (SMP) or UMA systems [17]. Such a mathematical model [13] has some variants of the organization. The real memory modules bandwidth and, as a consequence, the performance of the whole MPS depend on the realization of these variants. In cases where the UB is used as an interaction medium [2], the capacity of the MPS is also greatly influenced by its bandwidth. MPSs, in which memory with a unified address space is distributed over processor modules, is called DSM or NUMA-systems [17].

Memory rivalry shared by several devices including CPUs, direct memory access channels, etc., is the most common reason of a slowdown. Even the organization of memory in the form of several independent blocks does not exclude conflicts because of the random nature of blocks queries. UB sharing between CSs devices also leads to a slowdown. It is so because if the bus is loaded, for example, if you try to hold it with several CPUs simultaneously, after detection and conflict resolution, only one device will continue working, while others will stop working. Thus, rivalry for UB requires a detailed analysis, since the conflicts arising influence both its bandwidth and the overall performance of MPSs. To carry out a CSs research and analysis one should use as a rule techniques of probability theory and queueing theory, namely, systems (QSs) and queueing networks (QNs). Any part of a CS can be represented in the form of QS, and a set of all CS devices is represented by QN. It is rather easy to make a research with minimal financial costs. It is not necessary to build a real system. A mathematical model is quite enough [3].

2. THEORETICAL ANALYSIS

This article is of a research nature. To solve the problem a number of literature sources have been analyzed [4; 5; 14]. describing researches in the field under study searching for unimpaired and open issues. A number of issues related to the research of the "processor-memory" subsystem were not covered in scientific publications. Partially some problematic issues were considered in [6; 7]. MPSs parallel memory can be realized by two ways: 1) by layered blocks using address striping and unified addressing and data buffering (Figure 1a); 2) by independent blocks having their own addressing and data buffering (Figure 1b) [17].

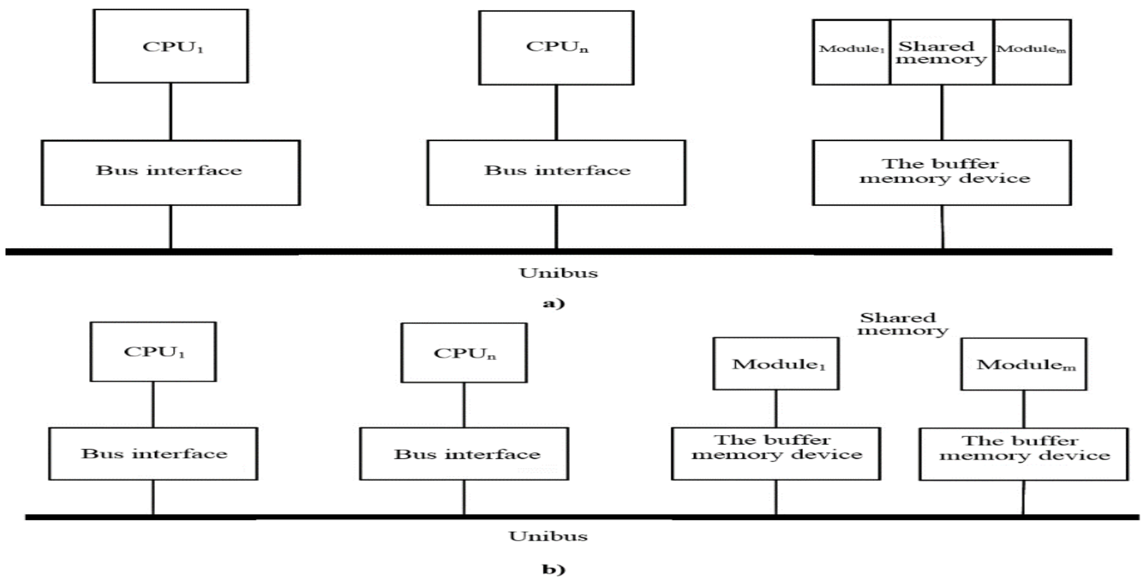


Figure 1: MPS units with shared memory: a – address striping memory; b – independent blocks memory

As you can see from Figure 1, several CPUs use one and the same bus to access memory and interact with each other simultaneously. The main functions of the bus interface are as follows: UB holding / disposal handling; transmission of addresses, data and control signals into and out of the memory; recognition of its own address set on UB devices of the system; external interrupts providing; transmitting data buffering.

Each CPU is assigned a unique number, which is called its own address. The exchange protocol provides the recognition of its own address, when the message is transmitted externally to this CPU module. The bus interface constantly keeps track the identifier of the address assigned. Once it detects its own address the CPU module is switched to the UB and data from the bus is received. In this paper [8] principles and algorithms of similar systems functioning are discussed. While working there can arise conflicts reducing CPU performance, as the execution time of a command increases during a memory query. The more CPUs are included in an MPS, the higher the memory queries number is and, consequently, the higher the performance losses of an individual CPU are due to conflicts in accessing UB and memory. In this article, we will examine in more detail probabilistic and time-dependent characteristics of a MPS with shared memory.

Assume that the "processor-memory" exchange is made word by word or by groups of words. If you use exchange without transactions splitting, a simple bus cycle is necessary to perform a write/read transaction. Then it is disposed and can be used for processing another transaction [9]. Time required for a trasaction execution is

$$\tau_T = \tau_W + \tau_B, \qquad (1)$$

where τ_W – time spent by the CPU for UB holding; τ_B – bus cycle. Time holding τ_W depends on the way of UB handling, and the minimal value of a bus cycle is

$$\tau_B = \tau_A + \tau_M, \qquad (2)$$

where τ_B – time spent for an address being trasmitted from the CPU into memory; τ_M – memory cycle. Taking into accout that the second summand is more than the first one it is evident that the bus cycle time depends greatly on the memory cycle time. Memory cycle declination is achieved via the parallel operation of its modules. To provide a sufficiently high memory speed one should improve the bandwidth of the UB. This can be achieved by applying the method of transactions splitting [10; 13] when transferring data between the CPU and the shared memory. It results in bus cycle declination due to the usage of high-speed buffer memory in the circuits of CPU interaction via the UB and in memory controllers (often specialized [11]). In this case, it is possible to transfer several transactions via the UB during the memory cycle, and as a result, a number of independent memory modules can operate in multiple.

The transactions splitting mode has its own peculiarities. They are as follows: only one transaction is created in the memory during recording; and there are two transactions during reading. The first is directly connected with the address inputting into memory that is stored in the buffer of the memory controller, after that the UB is disposed for other interactions. The second transaction is directly connected with the data recovery from the memory to the CPU. The memory buffer device (controller) [12] in this case is a more intelligent device than a similar controller used in the transactions non-splitting mode. The functions of the controller are as follows: the data posting to the specified CPU module of the shared memory, and after having been read, it is to hold the UB and transmit data to the destination point. In other words, the memory controller device should possess a high-speed memory for storing transactions, and moreover, it is necessary to have a control circuit providing an access to the UB. This method is called the exchange of transmitting data with buffering or packet switching [13].

Recording execution time is

$$\tau_T = \tau_W + \tau_A + \tau_{BUF}, \qquad (3)$$

Reading execution time is

$$\tau_T = 2 \cdot \tau_W + \tau_A + \tau_{BUF}, \qquad (4)$$

where τ_{BUF} –buffer memory addressing time, and bus cycle is

$$\tau_B = \tau_A + \tau_{BUF}. \qquad (5)$$

If packet exchange is used for the exchange between the CPU and memory, the transaction execution time in the transactions non-splitting mode is equal to

$$\tau_T = {}^{\tau_W}\!/_{k} + {}^{\tau_A}\!/_{k} + \tau_M, \qquad (6)$$

where k – the number of words in one packet.

The bus cycle is reduced in this case and is

$$\tau_B = {}^{\tau_A}\!/_{k} + \tau_M. \qquad (7)$$

Correspondently in exchange execution with transactions splitting it is equal to

$$\tau_T = {}^{\tau_W}\!/_{k} + {}^{\tau_A}\!/_{k} + \tau_{BUF}, \qquad (8)$$

bus cycle is

$$\tau_B = {}^{\tau_A}\!/_{k} + \tau_{BUF}. \qquad (9)$$

In real systems using a synchronous bus [16; 18] CPU one clock cycle (t) is allocated to access the UB, address outputting, and to the buffer memory, and m cycles are allocated to access the memory. Then with word-by-word exchange in the transactions non-splitting mode $\tau_T = 2 \cdot t + m$, where m can take values from 10 to 60 clock cycles of the CPU for the cached memory. In the transactions splitting exchange mode $\tau_T = 3 \cdot t$.When we use the exchange transactions non-splitting packet mode then $\tau_T = {}^{2 \cdot t}\!/_{k} + m$. If we use transactions splitting then $\tau_T = {}^{2 \cdot t}\!/_{k} + t$. Since a MPS consists of a number of CPUs, it is obvious that there will be conflicts between them for accessing the UB and the shared memory.

This, of course, leads to an increase of the transactions execution time due to idle states queueing for their servicing. Let's consider a technique according to which it is possible to determine the influence of conflict situations on the transaction execution time. Models for delays studying are presented as open two-phase network models (Figure 2), where the source of queries is CPUs providing task flows (transactions). Servicing devices are the UB and shared memory. In the models transaction flows are represented as task flows for servicing φi, $\phi_i = N_i / T_i$, where Ni is an average number of transactions of the i-th CPU to the memory during the *Ti* problem solution time. Queries servicing from the flow φ_i passes through two phases: the first one simulates delays, which are directly connected with the transactions processing of the UB, at the second – with the shared memory. In order to simplify and estimate the results obtained "from above", we assume that all flows are the simplest ones, and the service time is either constant or exponential. Figure 2 schematically shows the mathematical models of the systems represented in Figure 1: the system shown in Figure 1a corresponds to the model shown in Figure 2a, and the model shown in Figure 1b corresponds to the model shown in Figure 2b.

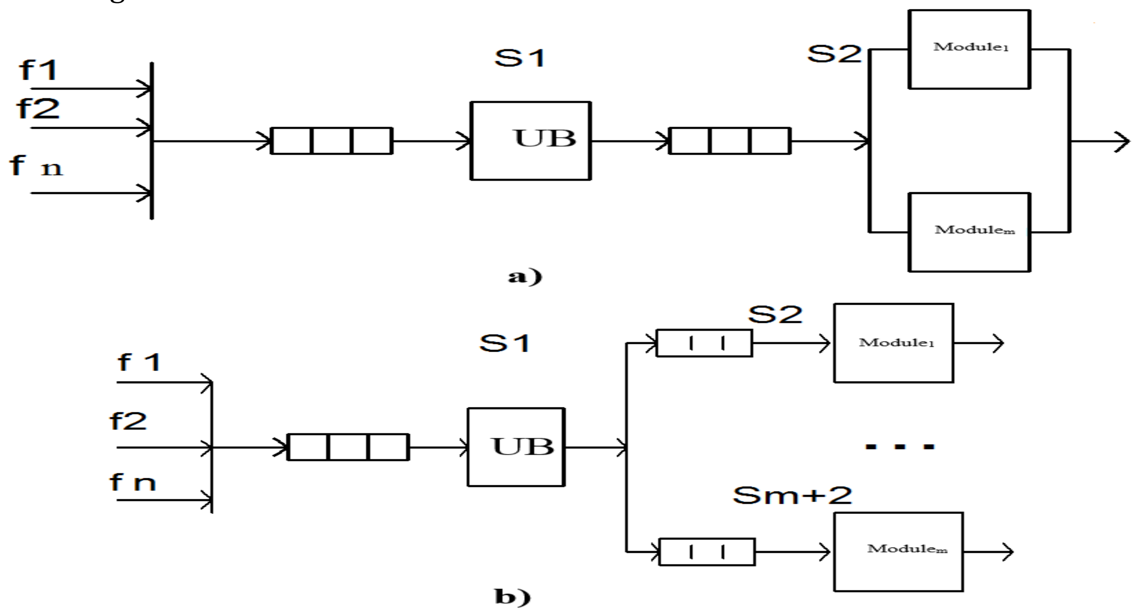


Figure 2: MPS models circuits: a) address striping memory; b) independent blocks memory

Delays estimation in systems using non-splitting transactions exchange is done according to the following procedure. Since the UB service model is represented by a single-channel QS, the service time at the first phase is τ_T. The service model in the memory can be represented by either a multi-channel QS, or by a set of single-channel QSs [16]. The service time at the second phase is τ_B. The input task flow rate of the two-phase model is $\lambda = \sum_{i=1}^{n} \phi_i$, where φ_i – transactions flow rate generated by *i*-th CPU. The service latency period in queues at the first phase is [1].

$$w_1 = \frac{\lambda \xi_{UB}^2}{2(1 - \lambda \xi_{UB})}, \qquad (10)$$

where ξ_{UB} – UB service time determined by equations (1, 3, 4, 6, 8). At the second phase it is done correspodently

$$w_2 = \frac{t_B(\lambda\xi_{Module})^2}{mm!(1-\lambda\,\xi_{Module})^2[\sum_{i=1}^{m-1}\frac{(\lambda\xi_{Module})^i}{i!}+\frac{(\lambda\xi_{Module})^m}{m!(1-\lambda\xi_{Module}/m)}]}, \qquad (11)$$

where ξ_{Module} – service time by shared memory modules detrmined by equations (2, 5, 7, 9). The service process of inputing tasks into the model shown in Figure 2, b, is as follows. The task from the input flow with the rate λ_0 servived in QS S1, with the probability p_{1j}, is inputted in one of QSs Sj (j = 2, ..., m + 1). It is accepted in the article that the service in QS Sj is carried out in accordance with the FIFO discipline. The input flow rate of a two-phase QS is $\lambda_j = P_{1j}\cdot\lambda$, where tasks query probability in a QS Sj (j=2,…,m+1) can be deterrmined as $p_{1j} = N_{1j}/N$, where N_{1j} is an average number of transactions being served by the *j*-th memory module during the tasks solution time; *N* is a total number of transactions inputted into the memory for servicing, i.e. $N=\sum_{j=2}^{m+1}N_j$. The task latency period at the first service phase is determined by the equation (6). The task latency period at the second service phase is :

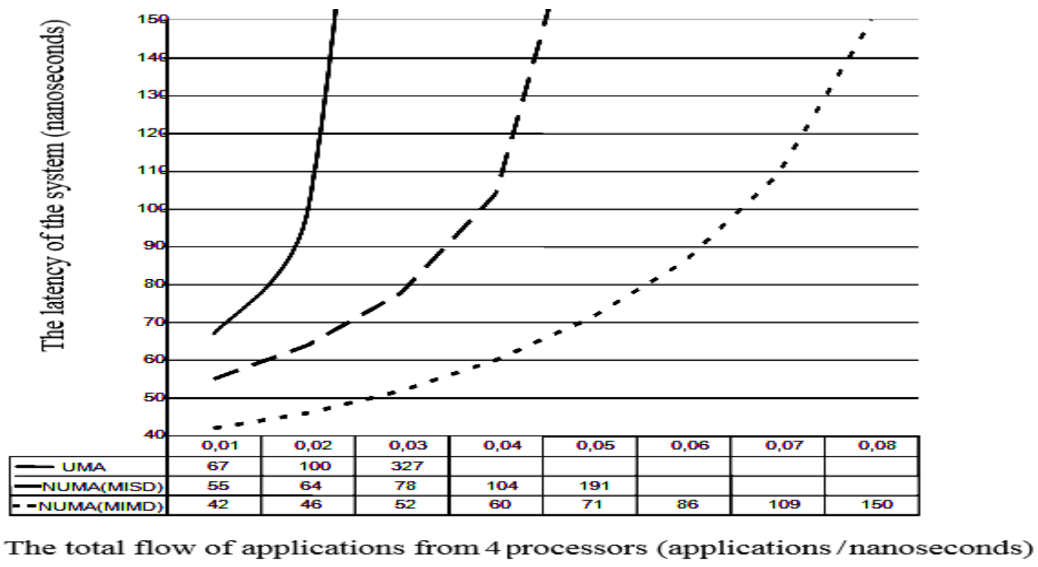
$$w_2 = \frac{p_{1j}\lambda\xi^2_{Module}}{2(1-\lambda\xi_{Module})}. \qquad (12)$$

The total service latency time in queues is $w = w_1 + w_2$, the exchange operations execution time between the CPU and shared memory (latency) is

$$\tau^*_T = \tau_T + w. \qquad (13)$$

In practical analysis the Netburst CPU of Pentium architecture with the clock speed of 2000 GHz, DDR PC-3200 memory and QDR bus were used. The memory access time was calculated on the basis of corresponding timings and statistical data. The time obtained is 37.5 ns. The analysis of the influence of queries frequency to the shared memory on the actual memory bandwidth was carried out taking into account that some of the queries will be satisfied in the cache of the CPU. It is obvious the queries frequency to shared memory modules will directly depend on the architecture of the CPU, its speed, as well as the number of CPUs in the system, the amount cache memory capacity, and the efficiency of the algorithm for cache coherency maintaining. Four-CPUs system was adopted for the analysis where all levels contain 512 KB of cache. In accordance with the statistical data [2] about 99% of the queries get in the cache with such a capacity. The data flow from the 2 GHz CPU to the shared memory is assumed to be 0.02 queries / ns. It is assumed that the memory query happens to be in each processor clock cycle.

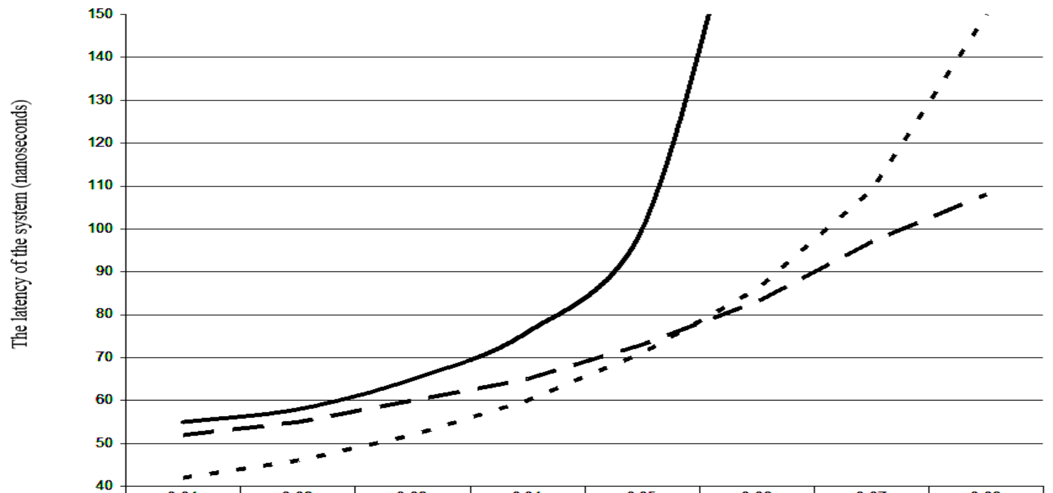
Two variants of MPS functioning with NUMA-architecture were analyzed during the process of simulation: 1) each CPU executes its own part of the program, mainly located in the local memory; here, the probability of accessing remote memory (the structure of inputting MISD task) is quite low; 2) all the CPUs execute the parts of the program, the data for which is distributed equally throughout the whole memory; the probability of the access to memory modules is high [the structure of inputtingg MIMD task]. Both variants discussed for MPSs with *UMA*-architecture will be equivalent since there is no concept of local and remote memory in such MPSs. For the architecture estimation the latency of the memory subsystem, measured in nanoseconds, was used. It should be noted immediately that the latency values can not be considered to be accurate when working with an open-type QNs. The result is not the exact amount of the CPU latency time of the data from the memory, but the compatibility ratio for the CPU speed and the memory subsystem. The higher the ratio, the greater the difference in performance in favor of the CPU and the more tangible delays in real CSs associated with obtaining data from the memory are. To visualize the value of such ratios in the program [15] a series of calculations was done with a different input flow of queries (with different CPU clock speeds). The diagrams (Figure 3 and Figure 4) show us the results obtained in the process of simulating.



	0,01	0,02	0,03	0,04	0,05	0,06	0,07	0,08
— UMA	67	100	327					
—NUMA(MISD)	55	64	78	104	191			
- -NUMA(MIMD)	42	46	52	60	71	86	109	150

Figure 3: Memory subsystems latency dependence on the queries flow frequency using address striping memory

The graph (Figure 3) shows the memory subsystem latency dependence on the queries flow frequency to the memory from 4 CPUs using address striping memory. We can see the power balance among the architectures of the memory subsystem. It is clerly seen that the best characteristics are demonstrated by the MPS with the *NUMA* memory architecture (with the structure of the inputting *MIMD* task) in comparison with the rest two functions options considered above. With a total flow of 0.08 queries / ns from 4 CPUs, the latency of the memory subsystem is 150 ns, while the MPS with the *NUMA* memory architecture (with the structure of the inputting *MISD* task) (with a flow of 0.05 queries / ns, the latency is 191 ns) and MPS with the memory architecture *UMA* (with a flow of 0.03 queries / ns latency is 327 ns) with such a stream of tasks can not manage. Analyzing the load factors of individual QSs, it is possible to determine the "bottlenecks" of the presented architectures. By changing the processing time in a QS or the number of channels, it is possible to determine the maximum parameters of the nodes taking the CS from bottlenecks off. In all three variants the "bottleneck" was the UB.



	0,01	0,02	0,03	0,04	0,05	0,06	0,07	0,08
UMA	55	58	65	76	100	192		
NUMA(MISD)	52	55	60	65	73	83	97	108
NUMA(MIMD)	42	46	52	60	71	86	109	150

The total flow of applications from 4 processors (applications/nanoseconds)

Figure 4: Memory subsystems latency dependence on the queries flow frequency to the memory using the memory of an independent blocks set

Figure 4 shows the memory subsystem latency dependence on the queries flow frequency to the memory from 4 CPUs using the memory of an independent blocks set. You can see that the latency has decreased greatly (especially for MPS with *UMA* memory architecture). With a total flow of 0.08 queries / ns from 4 CPUs, the latency of the memory subsystem is 108 ns [in the MPS with the *NUMA* memory architecture (with the inputting *MISD* task structure), with a similar MPS flow with the NUMA memory architecture (with the inputting *MIMD* task structure) the latency of the memory subsystem is 150 ns, while in the MPS with the memory architecture *UMA* with the flow of 0.06 queries / ns the latency is 192 ns]. The results of the analytical modeling were tested on imitation models built using the program [6] and differ by no more than 15%. It is quite acceptable at the stage of the preliminary design of a MPS. Based on these results a number of conclusions can be made and specific solutions can be proposed allowing to increase the speed and performance of the memory subsystem, as well as the whole MPS.

3. CONCLUSIONS

In this paper, the time for transactions servicing during the data exchange in multiprocessor systems on the basis of the UB with the shared memory has been estimated. Analytical models based on open queueing networks have been developed. Equations for calculating the transactions execution time during the exchange of the "processor-memory" subsystem have been proposed. A general analysis of MPSs performance has been carried out. It should be noted if there are overflowing flows at the input of the network model or there is a dependence of the generation of new queries on the condition of existing queries execution, then closed models should be used for the analysis. If the queries flows at the network input and the service time of individual devices do not possess the exponential law, there are overflowing flows in the simulating system, and the formation of a new query depends on the service conditions of the previously generated query, then simulations should be used to calculate models. Later in the models it is supposed to take into account the traffic appearing in the switching environment, as a result of providing cache coherency and memory interaction with external storage devices. The mathematical models discussed in the article make it possible to estimate the characteristics of multiprocessor systems without developing real models of such high tech and resource-intensive systems. Due to this, actually, the economic effect is achieved, since it is possible to estimate the characteristics of the systems being developed and choose the most optimal options without developing a real system.

REFERENCES

[1] Aliev T.I. Modeling fundamentals of discete systems//St.Petersburg: St.Petersburg State University ITMO, 2009, pp. 363.
[2] Biktashev R.A., Knyaz'kov V.S. Multiprocessor systems//Architecture, topology, performance analysis. Penza: Penza State University, 2004, 107.
[3] Boguslavskiy L.B. Probabilistic methods and models for managing data flows and resources in networks and multiprocessor systems: DSc (Computer sciences) thesis//M.: Institute of Control Sciences, 1995, pp. 38.
[4] Martens-Atyushev D.S., Martyshkin A.I. Development and research of reconfigurable computing cluster for digital signal rocessing//Modern information technology, 2015, Vol.21, pp.190-195.
[5] Martens-Atyushev, D.S., Martyshkin, A.I. Reconfigurable computing cluster for digital signal processing. Modern methods and means of processing spatial-temporal signals: XIII All-Russia Science and Technology Conference under the editorship of Sal'nikov I.I. Penza, 2015, pp.112-117.
[6] Martyshkin A.I. The study of memory subsystems with transaction buffering using queuing models. XXI: Resumes of the Past and Challenges of the Present plus, 2011, Vol.3, pp.124-131.
[7] Martyshkin A.I. Certificate of State Registration of a Computer Program. 610325. 2015.
[8] Martyshkin A.I. Development and research of open-loop models of the subsystem "processor-memory" of multiprocessor systems of UMA, NUMA architectures//Vestnik of Ryasan State Radioengineering University, 2015, Vol.54, No.1, pp.121-126.
[9] Martyshkin A.I. Development of hardware buffer memory device of a multiprocessor system//Fundamental Studies, 2015, Vol.12, No.3, pp.485-489.
[10] Martyshkin A.I. Mathematical modeling of the memory hardware buffer of multiprocessor systems//In Optoelectronic devices in image recognition systems, image processing and character information systems. Recognition–2015. XII International Science and Technology Conference, 2015, pp. 247-249).
[11] Martyshkin A.I. Realization of memory hardware buffer of multiprocessor systems. New Information Technologies and Systems: XII International Science and Technology Conference, 2015, pp.96-99.
[12] Martyshkin A.I. The study of algorithms for the process planning in real-time systems. Modern methods and means of processing spatial-temporal signals: XIII All-Russia Science and Technology Conference under the editorship of Sal'nikov I.I. Penza, 2015, pp.118-124.
[13] Martyshkin A.I. Development and research of open-loop models the subsystem" Processor-memory" of Multiprocessor systems architectures UMA, NUMA and SUMA//ARPN Journal of Engineering and Applied Sciences, 2016, Vol.11, No.2), pp.13526-13535.
[14] Martyshkin A.I. Mathematical models for estimating transactions service time of exchange operations execution in multiprocessor systems with shared memory//Izvestiya of Tula State University. Engineering Sciences, 2018, Vol.2, pp.166-174.
[15] Martyshkin A.I., Biktashev R. A. Certificate of State Registration of a Computer Program. 611117. 2013.
[16] Martyshkin A.I., Yasarevskaya O.N. Mathematical modeling of Task Managers for Multiprocessor systems on the basis of open-loop queuing networks. ARPN Journal of Engineering and Applied Sciences, 2015, Vol.10, No.16, pp.6744-6749
[17] Tanenbaum A.S., Bos H. Modern operating systems. Pearson. 2015.

[18] Tsil'ker B.Ya., Orlov S.A. The structure of computers and systems (2-d edition), - St.Petersburg: Peter, 2011, pp. 688.