DEPÓSITO LEGAL ppi 201502ZU4666
Esta publicación científica en formato digital
es continuidad de la revista impresa
ISSN 0041-8811
DEPÓSITO LEGAL pp 76-654

Revista de la Universidad del Zulia

Fundada en 1947
por el Dr. Jesús Enrique Lossada

Ciencias Exactas Naturales y de la Salud

Año 10 N° 27
Mayo - Agosto 2019
Tercera Época
Maracaibo-Venezuela

Mathematical models for evaluating efficiency and quality of means for synchronization of interacting processes in reconfigurable computer systems

Alexey I. Martyshkin *

ABSTRACT

In this paper, the modelling method is applied to evaluate the performance characteristics of various methods for synchronization of parallel processes in order to determine which way to use to implement them: software or hardware. Comparative characteristics of process synchronization by the spin lock method and the core lock method are presented. Charts have been constructed that display the obtained characteristics.

KEYWORDS: reconfigurable system, operating system, shared resource, process synchronization, task manager, model, stochastic network

Modelos matemáticos para evaluar la eficiencia y la calidad de los medios para la sincronización de procesos interactivos en sistemas informáticos reconfigurables.

RESUMEN

Se aplica el método de modelado para evaluar las características de rendimiento de varios métodos para la sincronización de procesos paralelos con el fin de determinar qué forma utilizar para implementarlos: software o hardware. Se presentan las características comparativas de la sincronización de procesos mediante el método de bloqueo de giro y el método de bloqueo de núcleo. Se han construido gráficos que muestran las características obtenidas.

PALABRAS CLAVE: sistema reconfigurable, sistema operativo, recurso compartido, sincronización de procesos, administrador de tareas, modelo, red estocástica.

*Candidate of Technical Sciences, Associate Professor, Computational Machines and Systems Department Penza State Technological University (440039, Russia, Penza, 1/11 Baydukovapassage / Gagarina str., 1/11 . e-mail: alexey314@yandex.ru)

Recibido: 14/09/2019 Aceptado: 19/11/2019

Introduction

The performance and reliability of an operating system largely depend on how the functions of computing process control are implemented: in the user space or the kernel of the operating system and how are they performed: in software or hardware. Traditionally, most of the operating system functions, including those for reconfigurable systems, are implemented in software and only part of the functions, such as interrupt control, calculating the executive address in virtual memory systems, providing cache coherence, and some others are implemented in hardware. Currently, reconfigurability is used both in universal computing systems, for example, based on multi-core processors, and in specialized ones implemented, for example, in systems based on a chip (Martyshkin, 2018a). The traditional approach to the implementation of an operating system acts as a brake for obtaining high performance and reliability, therefore, the current trend in the development of the operating system is associated with the expansion of the set of functions implemented in hardware. These include the synchronization functions for interacting parallel processes associated with access to shareable resources.

To implement synchronization, there are used mutual exclusion methods which are implemented through the critical sections mechanism (variable locks, mutexes, semaphores) and the monitor mechanism (Kreutzler, 1986; Tenembaum and Bos, 2015). The critical sections mechanism is that if a process trying to access shareable resources finds it busy, then it must wait for it to be released. This waiting can be organized in two ways: either being in a state of active waiting, i.e. occupying a certain processor, continuously try to access its critical sections, or, having released the processor, switch to the locked state until critical sections and some processor are simultaneously free. The first strategy is implemented in user space and is called spin lock or spinning. The second strategy is implemented by the operating system and is called as locking in the kernel.

Obviously, there are disadvantages in each of these strategies. When spinning is used, the processor performing the corresponding process stays in unproductive idle time waiting for access to shareable resources. Blocking a process in the kernel leads to switching of a current process and, consequently, to a possible reload of the cache memory of the corresponding processor, which, firstly, requires certain time costs and, secondly, increases the likelihood of a cache miss. With respect to the performance of the respective

processor, it is believed that the blocking procedure is equivalent to the loss of t_{cw} clock cycles required to switch the process.

The monitor is an intermediate strategy between spinning and blocking, in which the spin lock time is determined by some random value t_s until the process reaches the monitor entry. Upon reaching the monitor entry, the process either is serviced or leaves the monitor if it is found that the shareable resource is occupied by another process and is set by the kernel in a special queue of blocked processes (Martyshkin, 2018 b; Martyshkin, 2016a; Martyshkin, 2019).

Shareable resources can be characterized as lively or unlively. The shareable resources that are often accessed, that is, the interval between requests is small (Martyshkin, 2016b; Tenenbaum and Bos, 2015) or otherwise with high-intensity request flows, are lively. If the request intensity is high, then a queue will form before the shareable resource, leading to delays in the execution of processes. The shareable resources which are unlively do not create queues and, therefore, delays when accessing the critical section of the corresponding process. We consider the impact of lively and unlively shareable resources on the performance of a reconfigurable computation system by estimating the average process time through using stochastic queuing networks and compare the performance of a reconfigurable computation system for process synchronization strategies based on the critical section mechanism.

1. Mathematical configurable computer system using process synchronization by the spin locking method

The proposed model includes P statistically identical processes (flows), execution of each of them can be considered as an alternation of the computing phase of the processor (all N processors are also the same) with access to one of M statistically identical shareable resources, each of which is selected with a probability of 1 / M, i.e., there is an equally probable choice of shareable resources (Martyshkin, 2016e; Martyshkin and Yasarevskaya, 2015). Each process at the end of the calculation phase (the average time of which is t_p) with probability σ requires access to the i-th shareable resource. At the same time, the process is not blocked, but remains in standby mode (cyclically polls the bit semaphore) and uses the shareable resource during the average residence time (t_s) of the application in

the i-th queuing network (i = 1... M). Thus, the process does not leave the processor, and the duration of one stage of execution increases for the time the application stays in the access and use phase of the shareable resource ($t_p + t_s$). After using the shareable resource (the average time of which is t_{ks}), a new service phase starts, and the process continues the phase of calculations on the same processor.

The process execution ends with the processor release phase (S_0) with probability (1-σ), after which the process leaves the processor node. Then the task manager selects another task from the ready process queue by loading its context into a free processor, after which a new phase of calculations starts for it. Assuming the exponential nature of the time distributions of all the indicated phases of the process, we come to the reconfigurable computation system network model with the set of shareable resources shown in Fig. 1, where the processor node model is represented by a multi-channel queuing network, and the service models in critical sections are represented as a single-channel queueing network. Requests are selected in FIFO order of receipt.

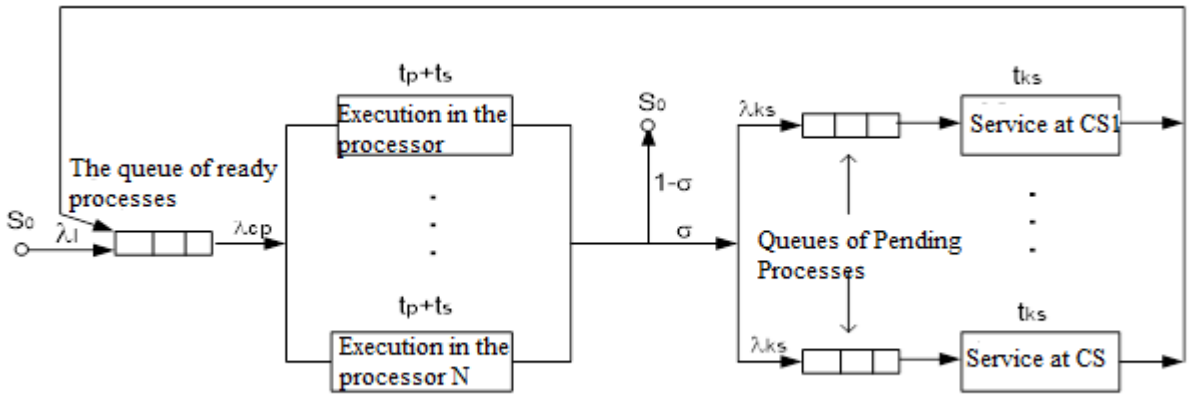

Figure 1 – The network model of a reconfigurable computation system with process synchronization by the spin lock method

We suppose that each process creates a simple flow of requests for the use of shareable resources, and their service times are exponentially distributed. Then the average residence time for one request in the critical section will be [10]

$$t_s = t_{ks} / 1 - \rho \quad , \qquad (1)$$

Where $\rho = \lambda_{ks} t_{ks}$ is loading of a critical section by a process requiring access to a shared resource; λ_{ks} is the intensity of the request flow at the input of the queuing

network simulating the procedure for accessing and using shareable resources $\lambda_{ks} = \lambda_{cp} \sigma / M$; λ_{cp} is the request flow rate at the input of the processor node $\lambda_{cp} = \lambda_I /(1-\sigma)$.

The execution time of one unit process in the processor node presented in the network model as a multi-channel queuing network will be.

$$t_{cp} = \frac{(t_p + t_s)\beta^N}{N!N(1-\beta/N)^2}\pi_0 + (t_p + t_s), \qquad (2)$$

Where $\beta = \lambda_{cp} t_p$ is the average number of busy channels in the queuing network in the calculation phase; π_0 is the probability of the absence of requests in a multi-channel queuing network (Martyshkin, 2016a).

If during the run time, the process would access the shareable resource n times, then the number of calculation steps in the processor node will be (n +1) [8]. The probability of the process moving from the calculation phase to the phase of access to the shareable resource and its use is $\sigma = n / n + 1$. The probability of the process moving into the processor release phase will be equal to 1 / n + 1, respectively. Then the execution time of a single process T in the reconfigurable computation system would be

$$T = t_{cp}(n+1) = t_{cp} /(1-\sigma), \qquad (3)$$

Figure 2 shows the diagram of the reconfigurable computation system analytical model with synchronization of processes by the spin lock method for a single shared resource, and Figure 3 shows the diagram for a set of shareable resources (Andrews, 2003).

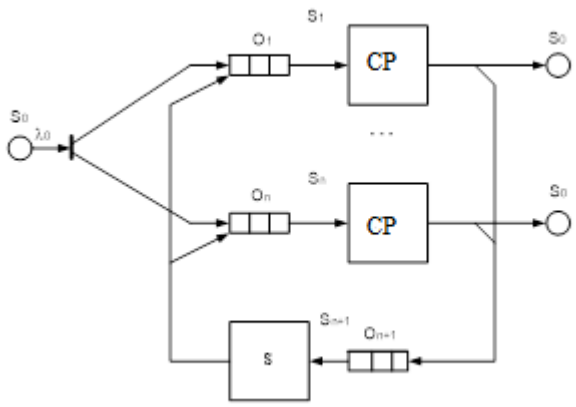

Figure 2 - Diagram of the reconfigurable computation system analytical model with the synchronization of processes by the spin lock method for a single shared resource

Here
- Sl-Sn - processor nodes;
- Sn + 1 - shareable resources (semaphore).

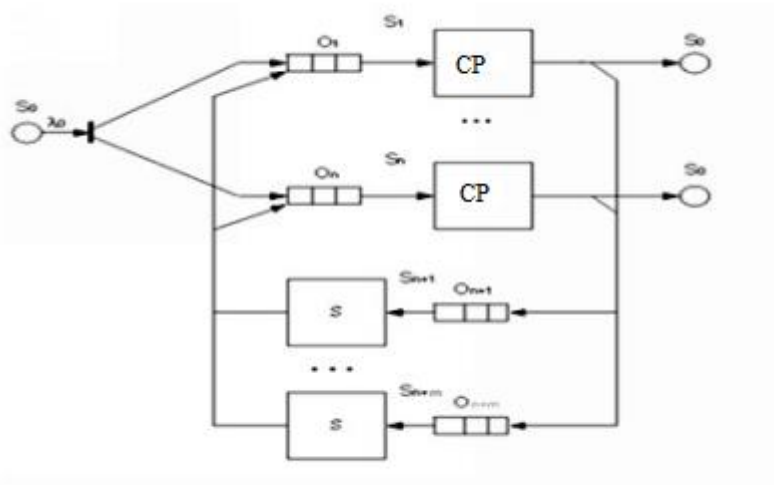

Figure 3 - Diagram of the reconfigurable computation system analytical model with the synchronization of processes by the spin lock method for many shareable resources

Here:
- Sl-Sn - processor nodes;
- Sn + 1, Sn + m - semaphores.

2. Mathematical model of a reconfigurable computation system using synchronization of processes with blocking in the kernel

This process synchronization strategy uses immediate blocking. At the end of the calculation phase, a process with probability σ requiring access to i-th shareable resource and finding the shareable resource busy, cannot enter its critical section, therefore it is blocked by the operating system, freeing the processor where it was running. The duration of the calculation phase is t_p. Then the task manager selects a new task from the ready-made process queue by loading its context into the freed processor. After using the shareable resource (the average time of which is t_{ks}), a new stage of servicing a previously blocked process starts, and the process continues the calculation phase mainly in another processor, which will lead to additional time losses due to a cache reboot. It is believed that the duration of the scheduling phase is determined by the time of switching the context and reloading the cache and is equal to the time value t_{cw} (Martyshkin, 2016c; Martyshkin,

2016d). Process execution with probability (1- σ) ends with the processor release phase (S_0), after which the process leaves the processor node.

The network model is shown in Figure 4. The average time of using a shared resource (staying one request in the critical section) will be

$$t_{ks} = \frac{t_{ks} + t_{cw}}{1 - \lambda_{ks}(t_{ks} + t_{cw})} = \frac{(t_{ks} + t_{cw})M(1-\sigma)}{M(1-\sigma) - \sigma\lambda_1(t_{ks} + t_{cw})} \quad (4)$$

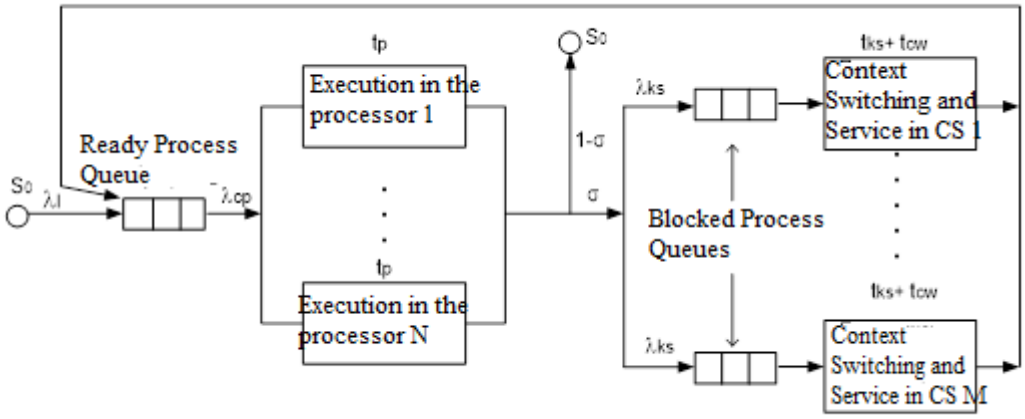

Figure 4 - Network model of a reconfigurable computation system with process synchronization by a method of blocking in the kernel

The execution time of a single process in the computational phase is

$$t_{cp} = \frac{t_p \beta^N}{N!N(1-\beta/N)^2}\pi_0 + t_p \quad (5)$$

The execution time of a single process taking into account the synchronization of parallel processes is determined as

$$T = (n+1)t_{cp} + \frac{nt_s}{M} = \frac{t_{cp}}{1-\sigma} + \frac{\sigma t_s}{M(1-\sigma)}. \quad (6)$$

Figure 5 shows a diagram representing a reconfigurable computation system analytical model with process synchronization using the method of locking in the kernel (monitor method) for a single shared resource. Figure 6 shows a diagram of n-processor reconfigurable computation system based on access to multiple shareable resources using the monitor method.

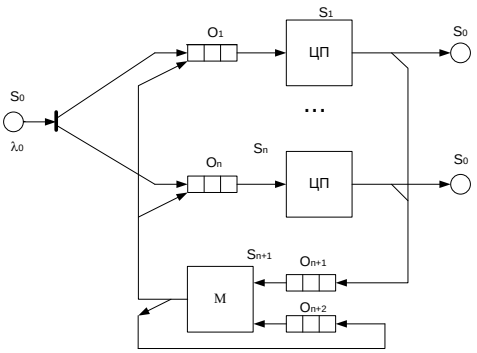

Figure 5 - Diagram of the reconfigurable computation system analytical model with process synchronization by the method of locking in the kernel for a single shared resource

Here:
- Sl-Sn - processor nodes;
- Sn + 1 - monitor;
- On + 2 - the queue of blocked processes

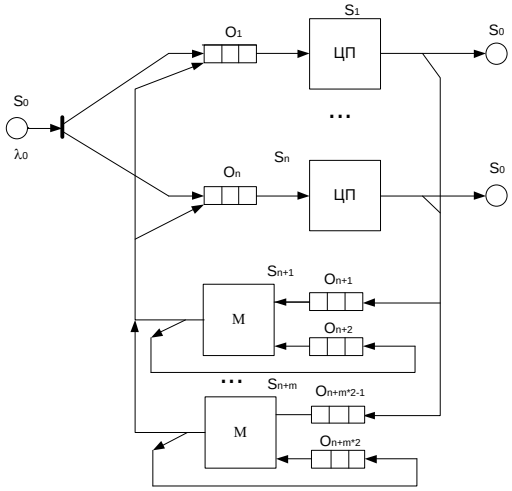

Figure 6 - Diagram of the analytical model of the n-processor reconfigurable computation system based on access to multiple shareable resources through the monitor method

Here:
- CPU $_{n}$ - n processor nodes;
-M - monitor;
- On $_{n+2}$ - the queue of blocked processes.

3. Study of the mathematical models considered

To evaluate performance losses due to conflicts over access to a shared resource (semaphore), an analytical model of n-processor reconfigurable computation system with a single shareable resource is considered (Fig.2). A model of n-processor reconfigurable computation system with a monitor-based access method is shown in Figure 5.

These models are presented in the form of an open stochastic queuing network consisting of n (Sl,..., Sn) single-channel queuing networks simulating processor nodes and a single-channel queuing network (Sn + 1), which simulates one of the access methods to the resource (Aliev, 2009). Moreover, the queuing network S0 acts as an external source of requests (requests for the execution of processes), which can be formed, for example, by user terminals (Kleinrock, 1979a; Kleinrock, 1979b). The queuing network S0 also acts as an absorber of requests served by a stochastic network. We assume that the request execution time vi in each processor node is exponentially distributed (Fundamentals of computational system, 1978).

The graph of the n-processor reconfigurable computation system transitions is shown in Figure 7.

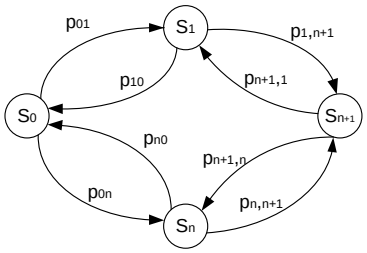

Figure 7 - Transition graph of the n-processor reconfigurable computation system analytic model with multiple shareable resources

Here:

- S0 - source of requests (terminals generating requests);

- Sl-Sn - n processor nodes;

- Sn + 1 - semaphores.

It is believed that applications form the simplest flows of requests, and service times follow exponential law. This distribution will make it possible to obtain results that are obviously worse than real values, which, in turn, will make it possible to upper-bound estimate the obtained results (Martyshkin et al, 2011).

4. Definition of initial data for modelling

The initial data are formed on the basis of architectural features in which the temporal characteristics are different.

The initial data was formed as follows:

- based on theoretical data;

- based on data obtained as a result of using programs that measure the context switching time and semaphore speed.

Thus, we get that the semaphore access time in user space is 50 μs, the access time to the monitor in the kernel space is 250 μs.

The service time in the processor node is taken from the following calculation: from a value commensurate with the minimum semaphore access time to a number much larger than the maximum resource access time. In numerical value, we get: from 50 μs to 600 μs.

In the reconfigurable computation system, when waiting for access to a shared resource, the processing time in the processor node increases by an amount equal to the latency of the critical resource. Upon that, an unproductive idling of the processor node for more than one cycle is likely. To analyse this behaviour of the system, an access control model with a single shareable resource was studied, where the service time of the processor node increases in each cycle (1)

$$V_{cp_n} = V_{cp_{n-1}} + W_{S_{n-1}} \qquad (7)$$

Initial data:

- the number of processor nodes in the queuing network K = 4;
- service time for requests by one processor node ν = 50; 600 μs;
- time for servicing requests with a critical resource ν = 50; 250 μs.

The simulation results are presented in Table 1.

Table 1 - The simulation results

Ps	Pcp	Vs, μs	Vcp, μs	Ws, μs	Wcp, μs
0.25	0.76	50	600	16.6	336.3
0.25	0.77	50	616.6	16.6	366.3
0.25	0.79	50	633.2	16.6	441.3
0.25	0.81	50	649.8	16.6	535.3
0.25	0.83	50	666.4	16.6	655.5
0.25	0.85	50	683	16.6	813
0.375	0.225	250	600	150	2,8
0.375	0.28	250	750	150	7.8
0.375	0.34	250	900	150	18.3
0.375	0.4	250	1050	150	37.5
0.375	0.45	250	1200	150	70.1
0.375	0.5	250	1350	150	123.1
0.375	0.225	250	600	150	2,8
0.375	0.28	250	750	150	7.8

0.375	0.34	250	900	150	18.3
0.375	0.4	250	1050	150	37.5
0.375	0.45	250	1200	150	70.1
0.375	0.5	250	1350	150	123.1
0.5	0.125	50	50	50	0,026
0.5	0.25	50	100	50	0.68
0.5	0.375	50	150	50	4.48
0.5	0.5	50	200	50	17.4
0.5	0.625	50	250	50	53.3
0.5	0.75	50	300	50	152.8
0.6	0,03	200	50	299.2	0,0002
0.6	0.26	200	349.2	299.2	2,8
0.6	0.49	200	648.4	299.2	50.7
0.6	0.71	200	947.6	299.2	364.7
0.6	0.94	200	1246.8	299.2	4130.4
0.6	› 1	200	1546	299.2	

The calculation results of the models showed that the waiting time for the release of the monitor is much higher than the waiting time for the semaphore, which increases the response time of the reconfigurable computation system; spin lock is the least time-consuming over the entire range of load changes.

At the same time, the simulation results confirm that it is most expedient to use spin-locking for processes that often turn to a common resource, the processing time of which on processor nodes is small. The disadvantage of this method is the cost of pre-programming, which greatly complicates its use.

When using a monitor, it takes a longer time to wait for it to be released, however, this method is simpler to use and can be used for processes that have a long processing time in the processor node.

Also, to develop a criterion for choosing the optimal method of access to a single shared resource, a study was made of various service times in the processor node and in the semaphore. In this case, the ratio of the loading coefficient of the semaphore Ps to the loading coefficient of the processor node Pcp was calculated as follows

$$Z = P_s / P_{cp} \quad (8)$$

If we analyse the 4-processor reconfigurable computation system with different service times in the processor node and semaphore, and also take into account the

conclusions made above, we find that in order to determine the optimal method for accessing data, it is necessary that the coefficient Z lie within the limits of

$$0{,}4 * K > Z > 1 * K \quad (9)$$

For the 4-processor reconfigurable computation system, 1.6› Z› 4; for the 8-processor reconfigurable computation system, 3.2› Z› 8.

To evaluate performance losses due to conflicts over access to the semaphore, an analytical model of n-processor reconfigurable computation system with many shareable resources (Fig.3) is considered. The model of n-processor reconfigurable computation system with a monitor-based access method is shown in Figure 2.

The studies were conducted on models with 2 and 4 shareable resources in the reconfigurable computation system.

Initial data:

- the number of processor nodes in the queuing network K = 2... 12
- the time for servicing requests by one processor node ν = 600 μs;
- the time for servicing requests with a critical resource ν = 150 μs.

The intensity of the request flow during modelling changed as follows:

0.0000005% * (number of processor nodes).

The bottleneck in a reconfigurable computation system with 4 shareable resources is a critical resource. With a request flow of 65 * 10^{-7} requests / μs (the number of processor nodes K = 13), the semaphore in the reconfigurable computation system with two critical resources does not cope with the number of calls - Ps› 1, the waiting time of the semaphore increases sharply. In a reconfigurable computation system with 4 critical resources, the load on the semaphore is 2 times lower.

When the access time to the semaphore is increased from 50 μs to 250 μs, the waiting time for the reconfigurable computation system with 2 shareable resources increased many times. The load factor also increases, reaching 1 at 8 processor nodes in a reconfigurable computation system with Vs = 250 μs, while in a reconfigurable computation system with Vs = 50 μs at 8 processor node Ps = 0.2 in a reconfigurable computation system with 2 critical resources.

In general, the simulation results show that the choice of the most optimal method for controlling access to a critical resource, which affects the access time, affects the performance of the entire system.

Formula (9) was obtained above for a model with a single shareable resource. For systems with multiple shareable resources, additional analysis is required. To do this, we studied the access control model with the single shareable resource, where in each cycle the service time of the processor node increases, formula (7) above.

Initial data:

- the number of processor nodes in the queuing network K= 4;
- the service time of requests by one processor node ν| = 50; 600 microseconds;
- the time for servicing requests with a critical resource ν = 50; 250 μs.

The calculation results for the models showed that the waiting time for the release of the monitor is much higher than the waiting time for the semaphore that increases the response time of the reconfigurable computation system; spin lock is the least time-consuming over the entire range of load changes.

At the same time, the simulation results confirm that it is most expedient to use spin-locking for processes that often access to shareable resources, the processing time of which on processor nodes is small. The disadvantage of this method is the cost of pre-programming, which greatly complicates its use.

When using a monitor, it takes a longer time to wait for it to be released, however, this method is simpler to use and can be used for processes that have a long processing time in the processor node.

A study of various service times in the processor node and in the semaphore was also made to develop a criterion for choosing the optimal access method for a single shareable resource. In this case, the ratio of the loading coefficient of the semaphore Ps to the loading coefficient of the processor node Pcp (8) was calculated.

If to analyse the 4-processor reconfigurable computation systems with different service times in the processor node and semaphore, and also take into account the conclusions made above, we find that in order to choose the optimal method of accessing data, it is necessary that the coefficient Z would lie within the limit

$$0{,}4 * K / n \leq Z \leq 1 * K / n, \qquad (10)$$

Where n is the number of critical resources.

For a 4-processor system with 2 shareable resources $0.8 \leq Z \leq 2$; for a 4-processor reconfigurable computation system with 4 resources $0.2 \leq Z \leq 1$.

Previously, data were obtained for the system (Biktashev and Martyshkin, 2013), based on which the graphs depicted in Figures 8 and 9 were constructed.

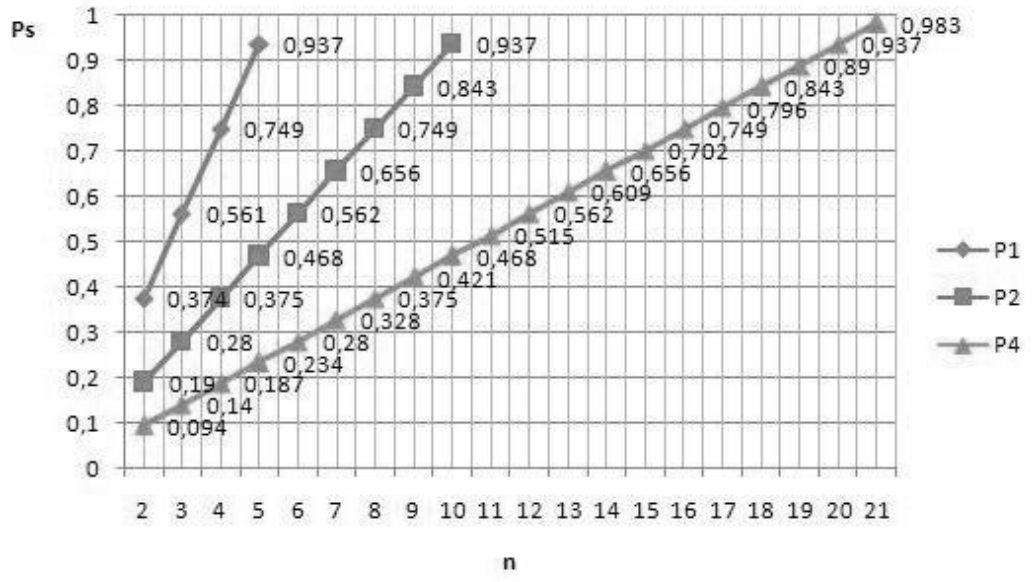

Figure 8 - Dependence of the semaphore load factor in the n-processor reconfigurable computation system on the number of processor node

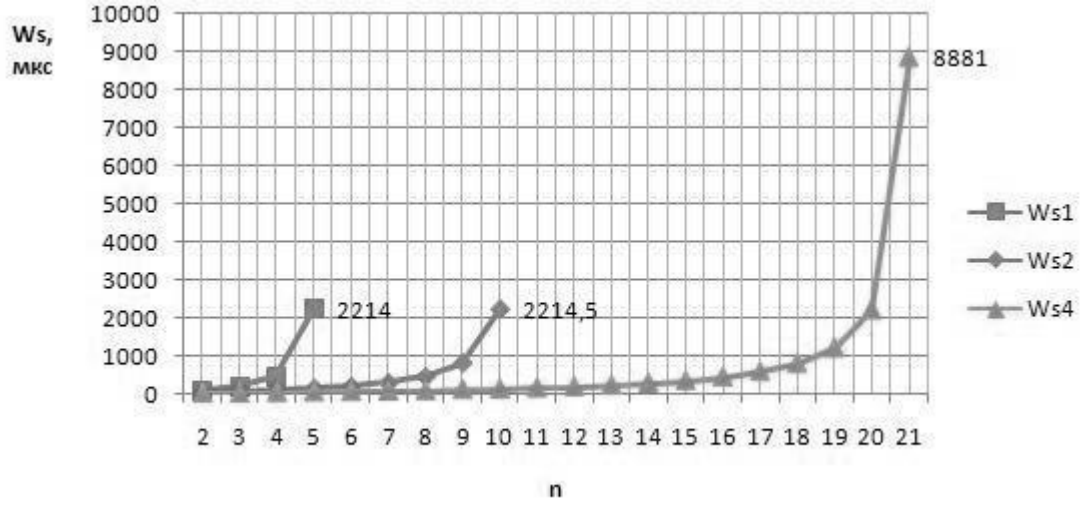

Figure 9 - Dependence of the semaphore latency in the n-processor reconfigurable computation system on the number of processor node

The graphs show that if there is one semaphore, the maximum number of executed processes is no more than 4, since the probability of accessing the semaphore is high, which

affects the increase in the wait time for this resource, while this probability decreases with the increase in the number of semaphores in the reconfigurable computation system.

Conclusions

The calculation for the models showed that spin lock method is the least time-consuming over the entire range of load changes for fixed values of t_{ks}, t_{cw}, σ, and t_p. The method based on locking in the kernel creates a higher complexity and significantly increases the response time (reduces performance) of the reconfigurable computation system.

It is known that the reliability of computing process control functions is higher if they are implemented in the kernel. However, such an implementation reduces the performance of the reconfigurable computation system. To remove the inconsistency and ensure high reliability in combination with high performance, a hardware implementation of the function for controlling the synchronization of parallel processes in the reconfigurable computation system is required.

Acknowledgements

Research was supported by the RFBR Grant for Best Projects of Basic Research, Grant No. 19-07-00516 A.

References

Aliev T.I. (2009). Fundamentals of modelling discrete systems. St. Petersburg: St. Petersburg State University ITMO, 2009.363 p.

Andrews G.R. (2003). Basics of multi-threaded parallel and distributed programming. Transl. from English M. : Williams, 2003.512 p.

Biktashev R.A., Martyshkin A.I. (2013). A set of programs for measuring the performance of operating system functions.// XXI century: results of the past and problems of the present plus. 2013. No. 10 (14). Pp. 190-197.

Fundamentals of computational system theory (1978). Edited by S.A. Mayorov. M.: Higher School, 1978.408p.

Kleinrock L. (1979a). Computing systems with queues: Publishing House Mir - M. : 1979. 600 p.

Kleinrock L. (1979b). Queuing Theory. - M.: Mechanical Engineering, 1979. 432 p.

Kreutzler S. (1986). Designing operating systems for small computers: Translated from English.- M.: Mir, 1986. 680 p

Martyshkin A.I. (2016a). Mathematical modelling and performance evaluation of process synchronization tools in multiprocessor systems. Scientific notes of the International Humanitarian University. Papers of participants of the Second International Multidisciplinary Conference. 2016. P. 151-157.

Martyshkin A.I. (2016b). Semaphore performance issues in parallel systems. The evolution of modern science: Collection of papers of the International scientific-practical conference: in 3 parts. Executive Editor: Asatur Albertovich Sukiasyan.2016. P. 81-83.

Martyshkin A.I. (2016c). Modern methods for measuring the performance of multicore computing systems. New information technologies and systems; collection of scientific papers at the XIII International scientific and technical conference.2016. P. 128-131.

Martyshkin A.I. (2017). Performance evaluation of high-performance systems, taking into account failures and recoveries of computing nodes // Modern innovative technologies for training engineering personnel for mining and transport.2017. Number 4. P. 425-429.

Martyshkin A.I. (2018a). Possible structural organization and performance assessment of reconfigurable computing systems based on a common bus // Models, systems, networks in economics, technology, nature and society.2018. No. 1 (25). Pp. 141-150.

Martyshkin A.I. (2018b). Mathematical models of semaphores for coordinating access to shared resources of multiprocessor systems // Colloquium-journal.2018. No. 8-1 (19). Pp. 36-39.

Martyshkin, A.I. (2016d). 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. Volume 11. Issue 23. PP. 13526-13535

Martyshkin, A.I. (2016e). Mathematical modelling of Tasks Managers with the strategy in space with a homogeneous and heterogeneous input flow and finite queue. ARPN Journal of Engineering and Applied Sciences. 2016. Vol. 11. Issue 19 PP. 11325-11332.

Martyshkin, A.I., Yasarevskaya, O.N. (2015). Mathematical modelling of Task Managers for Multiprocessor systems on the basis of open-loop queuing networks. ARPN Journal of Engineering and Applied Sciences. 2015. Vol. 10. Issue 16 PP. 6744-6749.

Matalytsky M.A., Tikhonenko O.M., Koluzaeva E.V. (2011). Queuing systems and networks: analysis and requests: Monograph. Grodno: GrSU, 2011.816 p.

Tanenbaum E., Bos H. (2015). Modern Operating Systems. 4th ed. SPb. : Peter, 2015.1120 p.: ill. (Series "Classic computer science").