VOL. 10, NO. 16, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

MATHEMATICAL MODELING OF TASK MANAGERS FOR MULTIPROCESSOR SYSTEMS ON THE BASIS OF OPEN-LOOP QUEUING NETWORKS

Martyshkin A. I. and Yasarevskaya O. N.
Penza State Technological University, Baydukov Proyezd / Gagarin Street, 1a/11, Penza, Penza region, Russia
E-Mail: alexey314@yandex.ru

ABSTRACT

The paper aims at carrying out the mathematical modeling and the performance analysis of a parallel computer system. Research methods are based on using the theory of analytical, numerical and simulation modeling, the theory of systems and queuing networks, the probability theory and the stochastic process theory. The authors deal with the analytical models of Task Managers for parallel processing systems based on the open-loop queuing systems. They investigate the methods of studying first-in-first-out Task Managers. The analytical models based on stochastic queuing networks for obtaining the managers' probabilistic and temporal characteristics are presented in the article. The results of the work done are equations for calculating the mean residence time of the problem in each system under study. The analytical calculations have been verified for their adequacy by simulation modeling. The experimental results have been displayed on the graph. During the problem investigation the appropriate conclusions have been made for each system type presented in the article. The considered models of Managers can be used in general purpose systems (for example in operating systems).

Keywords: mathematical modelling, processes planning, task manager, stochastic network, queuing system, operating system, service discipline.

1. INTRODUCTION

The design of new multiprocessor operating systems is a rather acute problem of decreasing overheads arising during planning processes. The function of task dispatching being chosen for the processing by CPUs is a part of a scheduler of an operating system [5]. The least expensive way to solve the problem is to carry out mathematical modeling of various characteristics of Task Manager. There are several different ways of constructing Task Managers in parallel processing systems (particular in multiprocessor systems). There are two main ones: with time-sharing (general queue of processes ready for processing) and with separation in space (individual queue of processes to service facilities (CPUs)) [2, 5, 8, 9]. The article discusses in details Managers with general tasks queue shared by all the processors. We assume that a kernel function of the operating system, including Task Manager, is implemented on a dedicated (control) processor.

We know quite a number of dispatching disciplines, that is, rules of forming a queue of runnable tasks, in accordance with which this queue is formed. Sometimes they are referred to as service disciplines, omitting the fact that the distribution of quanta of CPU time is talked about. Some dispatching disciplines show the best results for a particular service strategy, while for the strategy of another type they can be completely unacceptable. The discipline FIFO will be considered in the paper, which still finds its application because of its simplicity and ability to balance dynamically the processors usage [5].

2. GOAL SETTING

Consider the Managers with a shared queue, described in [2, 8, 9] details based on the models in the form of open-loop stochastic queuing systems (Figure-1). In the model the task coming from outside (from the source S_0 with the intensity λ_0) or after a context switch can be set in any CPU. In this article tasks setting according to a CPU is selected with equal probability to assess approximately the situation of mathematical modeling (analytical and simulation) of a real process, to avoid a system usage, when all tasks try to get a room in one or more of the CPUs, and some CPUs are idle. A queue is formed with a limited number of places (queue failures), so when it becomes full, a part of requests will be denied in service that corresponds to the actual mode of operation of a computer system, when a downloaded application remains on the hard disk until the tasks queue ready to service is busy in the processor node [1, 7]. The intensity of the tasks flow in this case will be equal to $\lambda_1 = \lambda_0 + \lambda_2$. The processor node of the network is represented as a multichannel queuing network of a *M*/*M*/*n*/*m* (S1) type, with the number of service facilities (*n*) and the limitation of the queue length (*m*). The CPUs form the flow of requests to the Manager, and if it is busy, a requesting processor enters a standby mode. In this case, a queue of pending processors is formed, which does not exceed the number of these facilities [1, 15, 19]. The intensity of the tasks flow coming from the queue O2 will be equal to $\lambda_2 = \lambda_1 - \lambda_0$. One task can be on Manager Service, and (*n*-1) tasks will wait in the queue in front of it.

The model of the access mechanism to this resource is represented as a single-channel queuing system (S2).

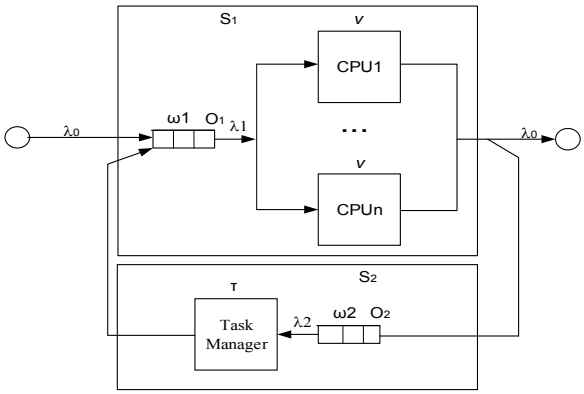

Figure-1. The diagram of the analytical model of *n*-processor system with a shared Task Manager.

There are two broad classes of service disciplines: priority and non-priority disciplines. The former means that a tasks selection from the queue happens to be in a predefined order without taking into account their relative importance and the time of operation (for example, FIFO and LIFO). The latter means that some tasks are provided with the prior right to enter the state of execution. The article deals with FIFO systems.

Let us consider the queues in queuing networks for this system.

3. THE CALCULATION OF CHARACTERISTICS OF QUEUING NETWORK

Consider Task Manager with the discipline FIFO (first in first out - «first come, first served») according to which tasks are served "first in first out" that is in the order of their appearance. Task Manager queues tasks which are not executed during the time given by the processor (quantum) again in front of the CPU for further processing. This approach allows you to implement the strategy of the service «if it is possible to finish calculations in order of their appearance». This discipline requires no external intervention during calculations; there is no redistribution of CPU time. We can say that it refers to nonpreemptive disciplines. The advantages are as follows. It is easy to implement and low system resources are used to form a tasks queue. However, this discipline leads to the fact that an increase of the computing system usage results in the growth of the average response time for the service, and short tasks (requiring small amount of computer time) have to wait as long as time-consuming ones [5, 16, 18]. The queue of tasks ready for processing in queuing systems S_1 is a queue with response and length limitations. The task coming at time when the channel is busy is queued and it waits for its service. Assume that no matter how many problems come to the input of the serving queuing system, it cannot accommodate more than m-tasks, one of which is being serviced and (m-1) are in standby mode. Tasks that are not included are served anywhere else, it means they are lost.

$\omega_1 = \lambda_1 / \mu_1$ – is the intensity of a stream, which represents the average number of tasks coming to queuing system S_1 during the service time of one task. Equations for the final probabilities can be found using the formulae obtained in [2]:

$$P_0 = \left(1 + \frac{\omega_1}{1!} + \frac{\omega_1^2}{2!} + ... + \frac{\omega_1^n}{n!} + \frac{\omega_1^{n+1} \cdot (1-(\omega_1/n)^m)}{n \cdot n! \cdot (1-\omega_1/n)}\right)^{-1}$$

where n - the number of service facilities (processors), m – the number of places in the queue.

The formation of a queue takes place when at the moment of the next task coming to the queuing system S1 all n processors are busy, that is if there will be either n, or n +1, ..., or (n + m-1) requests in the system. Since these events are inconsistent, then the probability of the queue formation $p_{оч}$ is the sum of the corresponding probabilities p_n, p_n +1, ..., p_n + m-1 [1, 3, 6, 7]:

$$p_{оч} = \sum_{i=0}^{m-1} p_{n+i} = \frac{\omega_1}{n!} \cdot \frac{1-(\omega_1/n)^m}{1-\omega_1/n} \cdot p_0.$$

A denial of the service occurs when all m places are occupied:

$$p_{otk} = p_{n+m} = \frac{\omega_1^{n+m}}{n^m n!} p_0.$$

Let's put down the average number of tasks in the service queue in the following equation [10]:

$$l_1 = \sum_{i=1}^{m} i \cdot p_{n+i} = \frac{\omega_1^{n+1} \cdot (1-(\omega_1/n)^m \cdot (1+m \cdot (1-\omega_1/n))}{n \cdot n! \cdot (1-\omega_1/n)^2} \cdot p_0.$$

The model of Task Manager seems to be a single-channel, closed queuing system (S_2) with a limited number of sources of service requests. The considered system is similar to the «task about machine-tools» [4]. A CPU generated the request is suspended if at the time of the request Manager is busy. In this model this state is modeled by the queue of processors O_2 for Manager. If Task Manager is free, it starts serving the request, which may take the average time required for the context switch τ.

Queuing system S2 has a number of states, which we enumerate according to the number of requests occupied by CPU generation:

S0 - all CPUs are capable of generating Task Manager requests, Task Manager is free;

S1 - one CPU is suspended, Task Manager is busy servicing a request);

S2 - two CPUs are suspended (Task Manager is busy servicing a request, one request is in the queue);

Sn - *n* CPUs are suspended, Task Manager is busy servicing a request, (n-1) requests are in the queue.

Next, we present an equation to determine the probability of requests absence to queuing network S_2 [4, 11]:

$$\pi_0 = \frac{1}{1+n(\lambda_2/\mu_2)+n(n-1)(\lambda_2/\mu_2)^2+...+n(n-1)...1\cdot(\lambda_2/\mu_2)^n}$$

For example β – the average number of CPUs being busy with stream service coming from the ready tasks queue.

These CPUs are a source of the requests to Task Manager, which intensity is λ_2. At the entrance of queuing system S_2 requests are generated with the average intensity $\beta \cdot \lambda_2$; all these requests are served in Task Manager, therefore, $\beta \cdot \lambda_2 = (1-\pi_0)\cdot\mu_2$, whence $\beta = \frac{\mu_2}{\lambda_2}\cdot(1-\pi_0)$, or $\beta = \frac{1-p_0}{\omega_2}$.

Now we define the average number of CPUs waiting for Task Manager service. The number of serving CPUs (β) is the sum of the number of tasks (l_2) queuing to Task Manager service, and the number of tasks (r_2) directly being serviced is $\beta = l_2 + r_2$. The number of tasks being included in service is equal to one if Task Manager is busy, and zero if it is free, that is the average value of r_2 is the probability that Task Manager is busy ($r = 1 - \pi_0$). If we subtract this value from the value of the average number of CPUs engaged in service, we'll get the average number of tasks waiting for service [4, 13]:

$$l_2 = \frac{1-\pi_0}{\omega_2} - (1-\pi_0) = (1-\pi_0)(1+\frac{1}{\omega_2})$$

Now let's consider the intensity of tasks streams in queuing networks including the queuing systems S_1 and S_2. They can be represented by the following system of equations:

$$\begin{cases} \lambda_0 = P_{00}\lambda_0 + P_{10}\lambda_1 + P_{20}\lambda_2; \\ \lambda_1 = P_{01}\lambda_0 + P_{11}\lambda_1 + P_{21}\lambda_2; \\ \lambda_2 = P_{02}\lambda_0 + P_{12}\lambda_1 + P_{22}\lambda_2. \end{cases}$$

$$\begin{cases} \lambda_0 = P_{10}\lambda_1; \\ \lambda_1 = P_{01}\lambda_0 + P_{21}\lambda_2; \\ \lambda_2 = P_{12}\lambda_1. \end{cases}$$

$$\begin{cases} \lambda_1 = \lambda_0 / P_{10}; \\ \lambda_2 = \lambda_0 \cdot P_{12} / P_{10}. \end{cases}$$

The intensities λ_1 and λ_2 will depend on the incoming intensity of tasks of the source S_0 and the transition probabilities from S_1 to S_2 (P_{12}) and from S_1 to S_0 (P_{10}). Since there are denials of service in the queuing system S_2, the intensities of flows λ_1 and λ_2 will decline:

$$\begin{cases} \lambda_1 = \lambda_0 / P_{10} \cdot (1 - P_{otk}); \\ \lambda_2 = \lambda_0 \cdot P_{12} / P_{10} \cdot (1 - P_{otk}). \end{cases}$$

The average delay time (latency) of a task in queues is determined by the formulae

$$w_1 = \frac{l_1}{\lambda_1}, \quad w_2 = \frac{L_2}{\lambda_2}$$

Determine the transmission ratio [2] showing how many times the task will go through this queuing system, $\alpha_i = \frac{\lambda_i}{\lambda_0}$, where λ_i – the output intensity on the *i*-th queuing system, λ_0 – intensity of the tasks source.

$$\alpha_1 = \frac{\lambda_0}{P_{10}}\cdot\frac{1}{\lambda_0}; \qquad \alpha_1 = \frac{1}{P_{10}};$$

$$\alpha_2 = \frac{P_{12}\cdot\lambda_0}{P_{10}}\cdot\frac{1}{\lambda_0}. \quad \alpha_2 = \frac{P_{12}}{P_{10}}.$$

Since each task can get the service in the *i*-th queuing system in the mean α_i times, the queuing time for service and time of its residence in the system will increase in α_i times [14]. Here is the average queuing time of tasks in network queues $W = \alpha_1 w_1 + \alpha_2 w_2$

$$W = \frac{w_1}{P_{10}} + \frac{w_2 \cdot P_{12}}{P_{10}} = \frac{w_1 + w_2 \cdot P_{12}}{P_{10}}$$

The residence time of tasks in the system with Task Manager and tasks general queue

$$U = \alpha_1 \cdot u_1 + \alpha_2 \cdot u_2, u_1 = w_1 + \nu; \; u_2 = w_2 + \tau.$$

$$U = \frac{w_1 + \nu}{P_{10}} + \frac{P_{12}\cdot(w_2+\tau)}{P_{10}} = \frac{w_1 + \nu + P_{12}\cdot(w_2+\tau)}{P_{10}}.$$

General queue with processor relative priorities and manager relative priorities

Priorities are referred to as relative ones in case if they are taken into account at the moment of task selection for serving and if they don't influence the system operation at the moment of task serving of any priority. After the completion of serving of any task from the queue, a task with a high priority is taken for serving. If during the process of its serving some other tasks with a higher priority come, the serving process of the current task won't be interrupted. This task remains with the highest priority up to the end of the serving process. With the introduction of relative priorities to the manager queue the response time of top-priority tasks is noticeably

shortened. They don't stand in queues as in case of above mentioned disciplines without serving priority.

The equation for average response time in the n-channel system is given below:

$$w_1^{RP} = (T/n)^{1/k}\,\Gamma\cdot(1+1/k) = \\ = \frac{1}{k}\cdot\left(T/n\right)^{1/k}\Gamma\cdot(1/k),$$

where n - the number of channels in the system; k - the number of tasks priorities types; Γ - the gamma function; T - the distribution moment.

The equation for average response time in one-channel system (M/G/1 type) is given below [1]:

$$w_2^{RP} = \frac{\sum_{i=1}^{H}\lambda_i b_i^2(1+\nu_{b_i}^2)}{2(1-R_{k-1})(1-R_k)},$$

where λ_i – the intensity of a coming stream of tasks of priority; b_i – the average value at the distribution according to the arbitrary law B (τ) of serving period τ_b; ν_{b_i} – coefficient of variation; R_{k-1} and R_k – total usage made by tasks with the priority not less than (k-1) and k correspondingly:

$$R_{k-1} = \sum_{i=1}^{k-1}\rho_i;\ R_k = \sum_{i=1}^{k}\rho_i$$

On the basis of all mentioned above and formulae for response time in queues with relative priorities, let's determine the average response time of tasks in the system:

$$w^{RP} = \left(\frac{1}{k}\cdot\left(T/n\right)^{1/k}\Gamma\cdot(1/k)\right) + \left(\frac{\sum_{i=1}^{H}\lambda_i b_i^2(1+\nu_{b_i}^2)}{2(1-R_{k-1})(1-R_k)}\right)$$

General queue with processor relative priorities and manager absolute priorities

Task Manager operating on the absolute priorities principal (processor is at the disposal of a top-priority task; if priorities are equal it operates on the order principal) is used in some OS of real time.

In other words, the appearance of a higher priority task may be a reason for a task removal. So if it is necessary to arrange tasks serving in such a way that all of them accept processor time on a regular and peer basis then a system operator can arrange this discipline itself. It is enough for all users' tasks to have similar priorities and formulate one top-priority task which shouldn't be active but however it will be planned for serving by a timer (in specified periods of time).

Thanks to high priority of this task the current application will be removed from serving and will be placed to the end of the queue. Since this task is not active then the processor becomes free and the next task from the queue will be taken. The priority is absolute if it interrupts low priority task serving. The interrupted task either can be lost or placed back to the queue for further serving which can be of two variants: serving from the very start, further serving (task serving proceeds from the interrupted moment). In this article we mean further serving of the interrupted task.

On the basis of [1] the task average response time for k-class:

$$w_k^{AП} = \frac{\sum_{i=1}^{H}\lambda_i b_i(1+\nu_{b_i}^2)}{2(1-R_{k-1})(1-R_k)} + \frac{R_{k-1}b_k}{1-R_{k-1}},\ (k=1,\ldots,H),$$

where R_{k-1} и R_k – total tasks usage having the priority not less than (k-1) and k correspondingly and determined according to the formulae (2).

Total average response time of k-class tasks for the described system will be determined by the following equation:

$$w_k^{ОП+АП} = \left(\frac{1}{k}\cdot\left(T/n\right)^{1/k}\Gamma\cdot(1/k)\right) + \left(\frac{\sum_{i=1}^{H}\lambda_i b_i(1+\nu_{b_i}^2)}{2(1-R_{k-1})(1-R_k)} + \frac{R_{k-1}b_k}{1-R_{k-1}}\right)$$

AN EXAMPLE AND SIMULATION RESULTS

The analytical modeling results obtained are verified and proven by simulation modeling which provides the evidence of the adequacy of the developed methods of carrying out Task Managers' research with a general queue. During the control checks of relevance and adequacy of the model we use the following data: the intensity of input flow of tasks λ_0 = 0, 03; 0, 06; 0, 09 tasks per microsecond (respectively low, medium and high usage of CPU). Time context switch by Task Manager is r = 9 microseconds, which corresponds to the average values of the context switch time of operating systems with slow feedback (for example in LinuxRT) [12]. The probability of treated tasks is P_{10}= 0, 05. The probability of sending task for further processing is P_{12} = 0, 95. Processing time for one task by a processor is ν =10 microseconds. The size of a general queue in front of CPUs is N=128 tasks. The Number of CPUs varies from 4 to 100. Here are the analytical

and simulation graphs obtained at medium, low and high CPUs usage

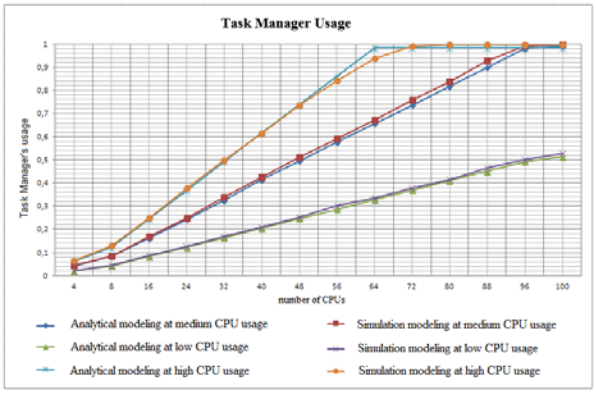

Figure-2. The dependence of Task Manager usage on the number of processors.

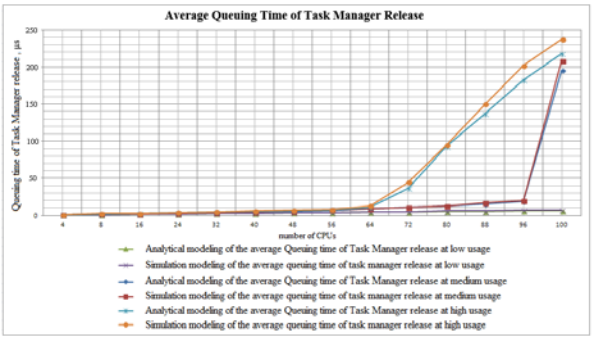

Figure-3. The dependence of the average latency of Task Manager release on the number of processors.

The mathematical modeling results presented in the graphs showed that Task Manager with requested parameters under the worst usage circumstances can be used up to soft real-time systems, because latency does not exceed 13 microseconds. This latency corresponds to many existing real-time systems, for example, LinuxRT [12].

If the number of CPUs in the system doesn't exceed 16 you can use Task Manager in hard real-time systems, as its latency is less than 3 microseconds. If the number of processors is more than 64, you should use more efficient Task Manager implemented, for example, in hardware.

DISCUSSIONS

In this paper, we have obtained formulae for the calculation of queuing time in queues (latency) of a processor node and manager, response time in the system as a whole.

The adequacy of the analytical model of Task Managers with a general queue is confirmed by the data obtained in the course of simulation. Error does not exceed 20%, which is quite acceptable for evaluating variants of the implementation of Task Managers at the initial development stage.

This work has been funded by the scholarship of the President of the Russian Federation (SP-828.2015.5).

REFERENCES

[1] Aliyev T.I. 2009. The Basics of Discrete System Modeling. SPb.: SPbGU ITMO. p. 363.

[2] Biktashev R.A., Martyshkin A.I. 2012. Modeling Task Managers of Multi-processor Systems // Advances of Modern Natural Science: Scientific Theoretical Journal. (6): 83-85.

[3] Bocharov P.P., Pechinkin A.V. 1995. The Queuing Theory: Textbook. RUDN. p. 529.

[4] Ventcel E. E. 2012. Introduction to operations research. M.: EE Media. p. 390.

[5] Gorgeyev A.V. 2004. Operating Systems. The 2nd Edition. SPb.: Piter. p. 417.

[6] Kleinrock L. 1979. Computing with Queues. M.: Mir. p. 600.

[7] Lozhkovskiy A.G. 2012. The Queuing Theory in Telecommunications: Textbook. Odessa: ONAS named after A.S. Popov. p. 112.

[8] Martyshkin A.I. 2012. Investigating Multi-processor Task Managers on Queuing Models // 21st Century: The Resumes of the Past and the Challenges of the Present Plus: Scientific and Methodological Journal. Penza: PGTA. (5): 139-146.

[9] Martyshkin A.I. 2013. The Mathematical Model of a Common Queue Task Manager for Parallel Processing Systems // Modern Methods and Means of Processing Spatio-temporal Signals: Proceedings of Articles of the 11th All-Russia Scientific and Technical Conference. Penza: PDZ. pp. 87-91.

[10] Matalytskiy M.A., Tichonenko O.M., Koluzayeva Ye.V. 2011. The Queuing Systems and Networks: the Analysis and Applications: The Monograph. Grodno: GrGU. p. 816.

[11] Matveyev V.F., Ushakov V.G. Queuing Systems. M.: MGU. p. 240.

[12] Mixalev V. 2012. Performance Test Results QNX Neutrino. // Modern Automation Technology: Scientific and Technical Journal. (2): 82-88.

[13] Abramov Vyacheslav M. 2006. Stochastic Analysis and Applications. 24(6): 1205-1221.

[14] Kobersy Iskandar, S., Ignatev Vladimir V., Finaev Valery I and Denisova Galina V. 2014. Automatic optimization of the route on the screen of the car driver. ARPN Journal of Engineering and Applied Sciences. 9(7): 1164-1169.

[15] Kempa Wojciech M. 2010. Stochastic Models. 26(3): 335-356.

[16] MacGregor Smith J., Kerbache Laoucine. 2012. International Journal of Production Research. 50(2): 461-484.

[17] Masuyama Hiroyuki, Takine Tetsuya. 2003. Stochastic Models. 19(3): 349-381.

[18] Finaev Valery, I., Beloglazov Denis A., Shapovalov Igor O., Kosenko Evgeny Y.a and Kobersy Iskandar S. 2015. Evolutionary algorithm for intelligent hybrid system training. ARPN Journal of Engineering and Applied Sciences. 10(6): 2386-2391.

[19] Nadarajah Saralees. 2008. Stochastic Analysis and Applications. 26(3): 526-536.