STUDY OF DISTRIBUTED TASK MANAGER MATHEMATICAL MODELS FOR MULTIPROCESSOR SYSTEMS BASED ON OPEN NETWORKS OF MASS SERVICING

Alexey I. MARTYSHKIN

Penza State Technological University, Russia, Baidukova passage / Gagarina street, 1a / 11, Penza, Penza Region, 440039
E-Mail: alexey314@yandex.ru

Abstract: The article performs mathematical modeling and the study of the multiprocessor system characteristics involving distributed task managers (with a spatial separation, with a homogeneous and heterogeneous incoming order flow and a limited queue length). The methods of the study are based on the use of analytical modeling theory provisions, the theory of systems, mass service networks and probability theory. The technique of multiprocessor system study is described with the specified types of task managers. We offer refined models to estimate the main temporal characteristics of dispatchers, taking into account the redistribution of task flows in order to equalize the processor load. The results of the study are analytical expressions to estimate the response time in the multiprocessor system under consideration. The adequacy of analytical calculations is verified by simulation modeling. The conclusions are presented in the end.

Keywords: analytical modeling, planning, dispatching, distributed task manager, division in space, service method, multiprocessor system.

1 Introduction

The basic ways of task manager development in multiprocessor systems (MPS) are widely known: with time-sharing (general task manager) (Tanenbaum E., Bos H. *2015*, Martyshkin A.I.,Yasarevskaya O.N. 2015) and the division in space (distributed task manager) (Tanenbaum E., Bos H. 2015, Martyshkin A.I. 2016). There is a lack of its organization in MPS with a single task manager, thus, the performance index of the entire MPS falls. The main "stumbling block" is manifested in the conflicts that arise when a task manager requests that only a certain processor communicates with the global queue of tasks ready for maintenance, which takes time. Moreover, in order to get a new task, it is necessary to enter into interaction with the task manager, which takes time again. At a certain point, the waiting tasks are not processed with free processors in MPS, because the task manager does not have time to serve them all. The way out of this situation is its another organization - with individual task queues for processors, as will be discussed in this work.

There are many planning disciplines in real-time systems (Tanenbaum E., Bos H. 2015), according to which the queue of tasks pending processing is developed. Considering modern real-time systems, they noticed that the tasks coming into service, represent a heterogeneous flow of different priority applications. High-priority applications are serviced faster, because a rapid reaction to them and a result delivery are necessary. The problems with a lower priority behave differently. An interrupted task is put into a "sleep" mode and awaits processing. When a received task has a relative priority in comparison with a performed task, it waits the end of the current work. The paper deals with MPS with the task manager, which is based on the spatial separation strategy, as a system with a homogeneous and heterogeneous flow of incoming service requests.

2 Problem Formulation

The mathematical model of the distributed task manager (Tanenbaum E., Bos H. 2015, Martyshkin A.I. 2016, A.I. Martyshkin. 2016, Martyshkin A.I., Vorontsov A.A., Valova O.O. 2015) consists of n single-channel mass service systems (MSS) ($S_1,...,S_n$) (Figure 1), where each MSS simulates the maintenance in the "task manager-processor" subsystem. A more detailed description of a similar model was given in (4, 5). A specific MSS simulates the maintenance by the task manager and the processor (S_1, S_2, …, S_m). From the S_0 source the flows of service requirements λ_0 are received, and it also absorbs the serviced tasks. In the article, the distribution of tasks is chosen equiprobable for an approximate estimation of the real MSS behavior, in order to avoid its overloading, when all tasks will tend to be serviced on one or a number of processors, and some of them will be idle. Task managers form the final queues. At that, task managers not only organize queues for MSS processors, but also balance the load according to a certain system algorithm set by designers. In accordance with this, the tasks waiting to be processed can be selected from any i-th queue of a more loaded processor and to be placed in the queue of the less-loaded j-processor at the moment with a certain specific probability. Let us dwell in more detail on the expressions and the calculations, using which it is possible to estimate the probabilistic and the temporal characteristics of these types of task managers, and in the second half of the article, based on the results of the computational experiments that confirm the adequacy of the proposed models.

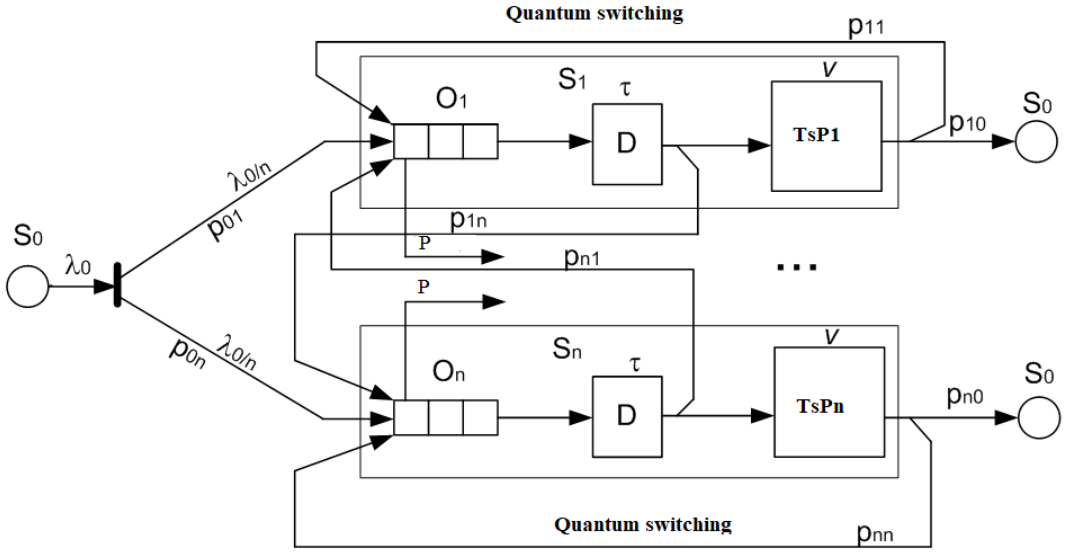

Fig 1. MPS model scheme with distributed task manager

The considered MPS is represented by an open network of mass service (ONMS), as a set of single-channel MSS ($S_1, S_3,..., S_{n-3}, S_{n-1}$). The intensity of maintenance

by the "task manager-processor" subsystem of the requirement flow is equal to $(\tau_{DZ} + \nu_{TsP})^{-1}$, where τDZ is the time of the task manager operation, νTsp – the processor operation time. The task, arrived at the moment when the subsystem is busy, is set in the queue and awaits maintenance. Let's assume that no matter how many tasks are queued, it can not accommodate more than k tasks, of which one is serviced, and k-1 is expected. The applications that do not fall into the O_i queue are serviced elsewhere, routed to a different queue of the "task manager-processor" subsystem with a redistribution probability

$$p = \rho_i^{k+1}(1-\rho_i)/1-\rho_i^{k+2}, \quad (1)$$

where $\rho_i = \lambda_i \cdot (\tau_{DZ} + \nu_{TsP})$.

The transmission coefficient for a particular problem is determined by the expression (A.I. Martyshkin. 2016)

$$\alpha_i = \left(\lambda_i + \sum_{\forall n(n \neq i)} p_{ij} \cdot \lambda_i \right) / \lambda_0, \quad i,\ j = \overline{1,n} \quad . \quad (2)$$

In (Martyshkin A.I. 2016, A.I. Martyshkin. 2016, Martyshkin A.I., Vorontsov A.A., Valova O.O. 2015), the mean waiting times of the task in queues and the response time of the system were determined.

Let's obtain mathematical calculations to determine the average number of tasks (S), both waiting in line, and serviced. We can solve this problem in this way: let's take the total number of problems in the system S as the sum of the tasks in the queue B and the tasks directly serviced by Ω. Then S = B + Ω. In accordance with (A.I. Martyshkin. 2016), we obtain the following:

$$S = M[S] = M[Z] + M[\Omega] = Loch_{cp} + \varpi \quad , \quad (3)$$

where $Loch_{cp}$ is the average number of waiting tasks in a queue;

ϖ – the average number of tasks being served.

The value $Loch_{cp}$ is defined in (4), let's find the value ϖ.

Due to the fact that the task manager in the considered part of the network model is one, the value Ω can be either 0 or 1. The value is equal to 0 if the task manager is free. The probability of this will be $P_0 = \frac{1-\psi}{1-\psi^{m+2}}$. It takes the value 1 in the case when the task manager is busy with maintenance. This probability makes $P_0 = \frac{\psi - \psi^{m+2}}{1-\psi^{m+2}}$. Taking into account the expressions obtained in (4) and given in the article, we have the mathematical expectation of served task number

$$\varpi = 0 \cdot P_0 + 1 \cdot (1 - P_0) = \frac{\omega - \omega^{m+2}}{1-\omega^{m+2}}$$

So, the average value of the number of tasks waiting and serviced by the task manager will be the following one:

$$Z = Loch_{cp} + \frac{\omega - \omega^{m+2}}{1-\omega^{m+2}} \quad , \quad (4)$$

Let's calculate the average time value of a task waiting arriving at MPS at any time in the queue for the task manager t_{oz}. With a certain probability P_0 the task manager is not busy and the task immediately goes to processing. With a certain probability P_1 the incoming task will fall into the MSS and waits for service during the period of time $1/\mu_D$. With the probability P_2 there will be one more task will be in a queue prior to our task and the average waiting time will be $2/\mu_D$, and so on. At κ=r+1 a new task will find the task manager already busy with processing and another r tasks in the queue. In this case, the waiting time will also be zero, because a task does not fit into the given queue, but goes to the other. According to (A.I. Martyshkin. 2016), the average latency of the task is

$$\bar{t}_{oz} = \frac{1}{\psi \cdot \mu_D} \cdot Loch_{cp} = \frac{Loch_{cp}}{\lambda_{00}} \quad (5)$$

Now let's estimate the time of finding the task in the subsystem "queue - task manager - processor". Let's define W_{CMO} as the time of finding the task in the MSS. This time is found from the sum of a number of

$$W_{CMO} = T_{oz} + T_{oDP} + T_{oTsP}$$

parameters ,

where T_{oz} is the waiting time in the queue before the task manager; T_{oDP} – the time for a task processing by a task manager, T_{oTsP} – task processing time by the processor.

According to the theorem of mathematical expectation addition

$$t_o = M[W_{CMO}] = M[T_{oz}] + M[T_{oDP}] + M[T_{oTsP}]$$

$$M[T_{oz}] = \bar{t}_{oz}$$

(4). For this work ,

$$M[T_{oDP}] = Q_o \cdot \bar{t}_{oDP} = \frac{Q_o}{\mu_D}$$

$$M[T_{oTsP}] = \mu_{TsP}$$

,

$$W_{CMO} = T_{oz} + T_{oDP} + T_{oTsp} = \frac{Loch_p}{\lambda_{00}} + \frac{Q_o}{\mu_D} + \mu_{TsP} \quad (6)$$

3 Computational Experiment

According to the received expressions and using the developed programs (The certificate of state registration of the computer program №2015610322, The certificate of state registration of the computer program №2015610325), the MPS study was carried out, which includes distributed task managers. The results showed that the task managers under consideration can be used up to "soft" real time systems at the worst load conditions, since latency does not exceed 15 mcs, which correlates with many existing real-time systems, for example, LinuxRT (Mikhalev V. 2012).

The obtained results of analytical modeling are verified by simulation modeling, which confirms the adequacy of the developed methodology to conduct the study of distributed task managers. However, there is a serious lack of distributed task manager organization as compared to the task managers with a common queue. A task manager with a single queue ensures that a task is invariably serviced on one of the system processors since it completely provides this process itself. In the case when a number of processors can not provide the processing of tasks, "fail", the task manager will not take them into account during task assignment; these processors will stop to support the interaction interface. In the MPS, which includes distributed task managers, a manager does not control the process of task retrieval from the queues for processor servicing. Therefore, if a number of processors "fail", their queue will receive tasks for a certain time until their number becomes large enough. In such a situation, it becomes necessary to implement an additional mechanism to monitor the processor functioning by the manager in order to detect faulty processors in time, as well as the ability to re-write the tasks from the queue of a non-working processor to other queues or to switch queues between processors. Such mechanisms increase the system resources used during a task manager organization. But this is justified by a significant increase of MPS speed in general.

The MPS with 4 processors and, accordingly, with 4 task managers was taken for computational experience. It has been obtained experimentally that the probability that a task will be redistributed for other queues after a dispatch is the following one: 0.05 at long tasks. The probability of a task result provision to a user makes 0.05 for long tasks; 0.3 for medium tasks; 0.7 for short tasks. The probability of a long task additional serving makes 0.9; average - 0.65; short - 0.25.

In the software package (The certificate of state registration of the computer program №2015610322), in accordance with the type of a system task being solved, we enter the necessary data so that it can calculate the values of the task flow transition intensities between network MSS. The values of the communication ratios between the devices are also calculated. The expression is determined from (Martyshkin A.I. 2016) to find the response time of such a system

From this we find, taking account the calculations in (A.I. Martyshkin. 2016)

$$U = \left(\sum_{i=1}^{n} \lambda_0^{-1} \cdot \lambda_i + \sum_{\forall n (n \neq i)} p_{ji} \cdot \lambda_i \right) \cdot \left(w_i + k \cdot \left(t_k + \delta + \tau + \varsigma \right) \right). \quad (7)$$

The processing time of each task is selected during the generation exponentially by input flow intensity. An average processing time of a task is chosen exponentially according to the product of one quantum time by the number of quanta necessary to complete a task. The model takes into account the phenomenon of restarting the processor cache when a new task enters it. The cache reset is selected as a static value and occurs with a specified probability. When the queue is full before the "task manager - processor", the task is redistributed to other queues with the probability $p_{\text{перерасп}i}$. The task is processed in the processor for a whole quantum. At the end of processing, the processing time is deducted from the internal variable of the task - the time of one quantum in the processor. Then the completeness check is performed by comparing the internal task variable that is responsible for the remaining time required to complete the processing with zero. If the task is completely processed, then it leaves the system.

In the course of the computational experiment, the complexity of the tasks changed (low for the tasks requiring a high reactivity, medium and high - for the tasks requiring a low reactivity). The load of processors was at the level of 65%, which corresponds to an average load of the system. The number of processors varied from 2 to 20. The complexity of the tasks was taken as follows: for highly reactive tasks - 0.1 ms, for the tasks with an average laboriousness - 0.5 ms, and finally for the most laborious tasks - 1.0 ms. The time of the quantum for the conducted experiments is assumed to be constant and equal to 0.1 ms. The task manager operation time during task context switching is 5 mcs (obtained by measuring on the prototype system using the program (The certificate of state registration of the computer program №2015610325)); the cache reset time is assumed equal to 5 mcs (the score is obtained in the RightMark Memory Analyzer program).

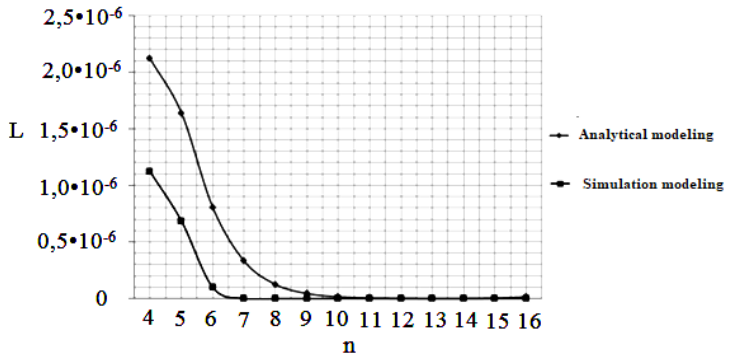

Fig2. Dependence of an average queue length before the subsystem "task manager-processor" on the number of processors in a system

Figure 2 shows the dependence of an average queue length before the subsystem "task manager-processor" (L) on the number of processors (n). The calculations were carried out in the program (The certificate of state registration of the computer program №2015610322). The graph shows that the queue tends to zero as the number of processors increases. Hence it follows that when you solve the problems with a high labor input, which corresponds to a low reactivity of the system and the mode of hard real time, it is minimal for any number of processors and decreases with labor input increase.

Suppose that the system in question is not exponential, then it can be represented as MSS M/G/1, then the characteristics of the system will be obtained, which are close to the real ones. In this paper, we consider the system with n processors and a heterogeneous flow of tasks arriving at the i-th MSS, and with relative priorities of task selection from a queue. According to (Zakharikova E.B. 2012.), we can replace a system of type M/G/1 with a finite queue by a similar system with an unbounded queue. It is easier to analyze and determine the main characteristics of SeMO, consisting of MSS with unlimited task queues, than consisting of MSS with a queue length limitation. If a queue contains a large number of places (at least 16 for real-time tasks), then the calculation error will be less than 2.5% (Zakharikova E.B. 2012.), which is quite acceptable. In work (Martyshkin A.I. 2016), the average waiting time of all priority tasks was calculated in all queues of the considered system, as well as the response time of the system with distributed task managers at a heterogeneous incoming task flow with relative and absolute priorities.

Let's consider the technique to estimate the MPS bandwidth by the example of two-stream processing, with one stream of the highest priority over another stream. The calculations of the characteristics for similar MPS were carried out in (Martyshkin A.I., Biktashev R.A., Vostokov N.G. 2013.). First, let's estimate the calculation error obtained by the numerical method. Figure 3 demonstrates the graphs showing the dependence of service suspension probability ($P_{\text{приост}}$) on the value of the reduced density of task streams (ρ), arriving in the processors and calculated by the expression (Martyshkin A.I. 2016- A.I. Martyshkin. 2016) for a different number of processors in the system. According to the assumption made in (Martyshkin A.I. 2016- A.I. Martyshkin. 2016), the reduced densities of the first and the second priority task flows are equal to each other. All calculations were carried out in the program (The certificate of state registration of the computer program №2015610322).

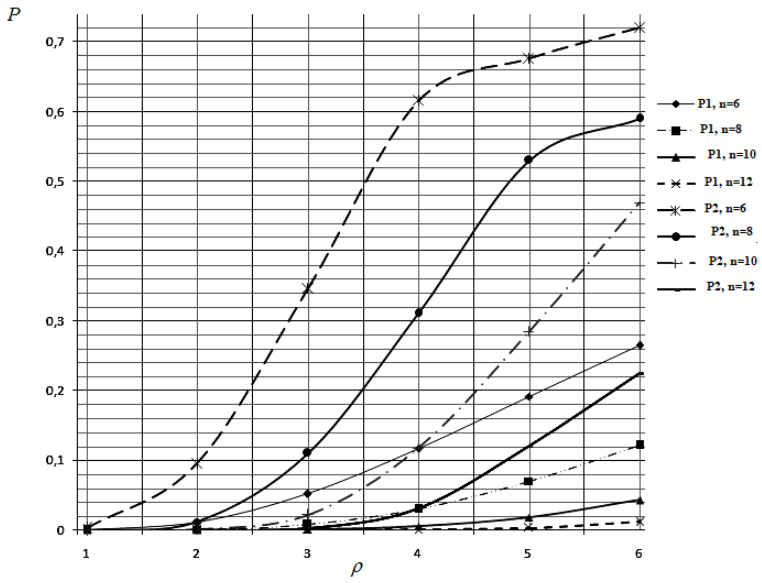

Fig 3. Probability P_{n1}, P_{n2} dependence on task stream density, calculated for different number of CP

According to the accepted assumption that the tasks of the first priority have an absolute priority in relation to the tasks of the second priority, it is possible to determine the bandwidth for each type of priority. An absolute bandwidth of the system with the tasks of two priority streams will be determined according to the following expressions:

$$A_1 = \lambda_1 \cdot (1 - P_{n1}),$$
$$A_2 = \lambda_2 \cdot (1 - P_{n2}), \quad (8)$$

where λ_1 – the intensity of the first priority task flow, λ_2 – the intensity of the second priority task flow, P_{n2} – the probability of second priority task service suspension due to the loading of the processors by the first priority task servicing, $P_{npuocm1}$ - the probability of the first priority task suspension due to the processor loading by the same priority task servicing. For example, let the intensity of the input stream of tasks of the most priority flow be equal to $\lambda_1 = 10$ z/ms, the intensity of the second stream makes $\lambda_2 = 12$ z/ms. Let's calculate the bandwidth with the number of processors equal to 6, 8, 10, 12. The reduced density of the first and the second flow is equal to each other. The probability of first priority task suspension is for the indicated number of processors, 0,1172, 0,03, 0,005, 0,0006, respectively. The probability of the second priority task service suspension for similar characteristics makes 0.6154, 0.31, 0.111 and 0.03, respectively. Hence, we find an absolute bandwidth with the number of processors equal to 6:

$$A_1 = 10 \cdot (1 - 0{,}1172) = 8{,}828 \text{ z/ms}; \quad A_2 = 12 \cdot (1 - 0{,}6154) = 4{,}6152 \text{ z/ms}$$

When the number of processors is 8:

$$A_1 = 10 \cdot (1 - 0{,}03) = 9{,}7 \text{ z/ms}; \quad A_2 = 12 \cdot (1 - 0{,}31) = 8{,}28 \text{ z/ms}$$

When the number of processors is 10:

$$A_1 = 10 \cdot (1 - 0{,}005) = 9{,}95 \text{ z/ms}; \quad A_2 = 12 \cdot (1 - 0{,}117) = 10{,}596 \text{ z/ms}$$

When the number of processors is 12:

$$A_1 = 10 \cdot (1 - 0{,}0006) = 9{,}994 \text{ z/ms}; \quad A_2 = 12 \cdot (1 - 0{,}03) = 11{,}64 \text{ z/ms}$$

4 Conclusions

The expressions were obtained to evaluate the latency of distributed task managers and the processor.

The adequacy of the proposed mathematical model of the task manager is confirmed by the available reference data (Zakharikova E.B. 2012.) and the results obtained on the proposed simulation model. The error in the results does not exceed 20%, which is quite satisfactory to evaluate possible ways of task manager implementation in a multiprocessor system during the draft design phase.

Based on the abovementioned, we note that it is advisable to use task managers with the division in space for hard real-time MPS with a large number of processors, with a small of processors it is advisable to use the task manager with time separation.

The simulation results confirm the fact that the developed model of the spatially separated task manager shows better characteristics closer to real systems than classical exponential models based on MSS M/M/1. The considered models of task managers can be used in the development of new operating systems, including real-time operating systems.

Literature:

1. Tanenbaum E., Bos H. *Modern operating systems*. SPb.: Peter, - 2015. - 1120 p.
2. 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. – N. 16. – P. 6744-6749.
3. Martyshkin A.I. *Mathematical modeling 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, No. 19, PP. 11325-11332.
4. A.I. Martyshkin. *The study of distributed task managers of multiprocessor systems based on mass service networks* // XXI century: the results of the past and the problems of the present. - 2016. - No. 3 (31). - pp. 190-194.
5. Martyshkin A.I., Vorontsov A.A., Valova O.O. *Mathematical modeling of task managers with spatial separation and a heterogeneous task flow for servicing and a limited queue length* // XXIst Century: the results of the past and the problems of the present. - 2015. - No. 3 (25). - pp. 142-149.
6. The certificate of state registration of the computer program №2015610322
7. The certificate of state registration of the computer program №2015610325
8. Mikhalev V. *Results of QNX Neutrino performance tests*. // Modern automation technologies: Scientific and technical journal. 2012. № 2. pp. 82-88.
9. Zakharikova E.B. *Simulation of system dynamics and mass service networks* / E.B. Zakharikova, P.P. Makarychev // In the world of scientific discoveries. - Krasnoyarsk: Publishing house "Scientific and Innovation Center". - 2012. - №8 (32) (Mathematics, Mechanics, Informatics). - pp. 222-235.
10. Martyshkin A.I., Biktashev R.A., Vostokov N.G. *Mathematical modeling of task managers for the systems of parallel processing based on open mass service systems* // In the world of scientific discoveries. - 2013. - No. 6.1 (42) (Mathematics, Mechanics, Informatics). - pp. 81-101.