


Research article

DOI: <https://doi.org/10.18721/JCSTCS.17307>

UDC 519.872.4:004.451.44



## MODEL OF A SUPERCOMPUTER CLUSTER IN THE FORM OF A QUEUEING SYSTEM WITH A RANDOM LIMIT ON THE EXECUTION TIME OF APPLIED TASKS

*O.I. Zayats, V.E. Baksheev,  
V.S. Zaborovsky, V.A. Muliukha* 

Peter the Great St. Petersburg Polytechnic University,  
St. Petersburg, Russian Federation

✉ [vladimir.muliukha@spbstu.ru](mailto:vladimir.muliukha@spbstu.ru)

**Abstract.** It is well known that the efficiency of task dispatching in any supercomputer system is determined, first of all, by the adequacy of the system model used, as well as the accuracy of the estimation of the parameters of the model itself. The article proposes a new version of the supercomputer cluster model, based on a standard model of the  $M/M/\infty$  class queueing system, which is supplemented with two fundamental clarifications that reflect the features of the supercomputer operation. First, the processing time of each task is limited by the dispatcher using a random variable distributed according to the exponential law. Second, it is considered that each new task requires the allocation of a random number of service channels (processors) for its execution. The parameters of the proposed queueing model are estimated based on statistical processing of data obtained during calculations previously performed on a supercomputer. A number of examples of using the developed model are given. To calculate the parameters of the queueing system, it is proposed to use the method of generating functions.

**Keywords:** queueing system, task dispatching, supercomputer, random execution time, random number of service channels

**Acknowledgements:** The research was financially supported by the Ministry of Science and Higher Education of the Russian Federation within the framework of the state assignment “Development and research of machine learning models for solving fundamental problems of artificial intelligence in the fuel and energy complex” (FSEG-2024-0027). The results of the work were obtained using the computational resources of the shared-use center “Polytechnic Supercomputer Center” of Peter the Great St. Petersburg Polytechnic University (<https://scc.spbstu.ru>).

**Citation:** Zayats O.I., Baksheev V.E., Zaborovsky V.S., Muliukha V.A. Model of a supercomputer cluster in the form of a queueing system with a random limit on the execution time of applied tasks. Computing, Telecommunications and Control, 2024, Vol. 17, No. 3, Pp. 71–83. DOI: [10.18721/JCSTCS.17307](https://doi.org/10.18721/JCSTCS.17307)



Научная статья

DOI: <https://doi.org/10.18721/JCSTCS.17307>

УДК 519.872.4:004.451.44



## МОДЕЛЬ СУПЕРКОМПЬЮТЕРНОГО КЛАСТЕРА В ВИДЕ СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ СО СЛУЧАЙНЫМ ОГРАНИЧЕНИЕМ ВРЕМЕНИ ВЫПОЛНЕНИЯ ПРИКЛАДНЫХ ЗАДАЧ

*О.И. Заяц, В.Е. Бакшеев,  
В.С. Заборовский, В.А. Мулюха*  

Санкт-Петербургский политехнический университет Петра Великого,  
Санкт-Петербург, Российская Федерация

 [vladimir.muliukha@spbstu.ru](mailto:vladimir.muliukha@spbstu.ru)

**Аннотация.** Хорошо известно, что эффективность диспетчеризации задач в любой суперкомпьютерной системе определяется, прежде всего, адекватностью используемой модели системы, а также точностью оценки параметров самой этой модели. В статье предлагается новая версия модели суперкомпьютерного кластера, основанная на типовой модели системы массового обслуживания класса  $M/M/\infty$ , которая дополнена двумя принципиальными уточнениями, отражающими особенности функционирования суперкомпьютера. Во-первых, время обработки каждого задания ограничивается диспетчером с помощью некоторой случайной величины, распределенной по показательному закону. Во-вторых, считается, что каждая новая задача требует для своего выполнения выделения ей случайного числа каналов обслуживания (процессоров). Параметры предложенной модели массового обслуживания оцениваются на основе статистической обработки данных, полученных в ходе расчетов, ранее выполненных на суперкомпьютере. Приводятся ряд примеров использования разработанной модели. Для расчета параметров системы массового обслуживания предлагается использовать метод производящих функций.

**Ключевые слова:** система массового обслуживания, распределение задач, суперкомпьютер, случайное время выполнения, случайное число каналов обслуживания

**Финансирование:** Исследование выполнено при финансовой поддержке Министерства науки и высшего образования Российской Федерации в рамках государственного задания «Разработка и исследование моделей машинного обучения для решения фундаментальных задач искусственного интеллекта в топливно-энергетическом комплексе» (FSEG-2024-0027). Результаты работы получены с использованием вычислительных ресурсов центра коллективного пользования «Суперкомпьютерный центр “Политехнический”» Санкт-Петербургского политехнического университета Петра Великого (<https://scc.spbstu.ru>).

**Для цитирования:** Zayats O.I., Baksheev V.E., Zaborovsky V.S., Muliukha V.A. Model of a supercomputer cluster in the form of a queueing system with a random limit on the execution time of applied tasks // Computing, Telecommunications and Control. 2024. T. 17, № 3. С. 71–83. DOI: 10.18721/JCSTCS.17307

### Introduction

Analytical research of supercomputer systems is of significant theoretical and practical interest. Currently, it is generally accepted that when modeling computer systems, the use of queueing theory is an adequate apparatus [1, 2]. Supercomputer systems (computer clusters) are extremely complex technical devices. If we try to model such a device as accurately as possible with all the smallest nuances of its behavior, the resulting mathematical model turns out to be very complex and, as a rule, makes its detailed analytical study very difficult. Therefore, it seems reasonable to start with a study of a simplified model,

which so far takes into account only the most important, fundamental factors that reflect the very essence of the phenomenon under study.

The queueing model proposed in this article, designed to describe the behavior of a supercomputer, takes into account two such factors that are fundamentally important for its functioning. First, it is necessary to take into account that each newly received service request may require a random number of service channels (processors) for its execution. Second, the execution time of each service request is limited in a certain way by the task manager. We will examine in more detail each of the two above-mentioned assumptions.

Queueing systems with a branching process for executing requests, in which each request is processed by several servers at once, form a special new class [3, 4]. In the English-language scientific literature such systems are called “the fork-join queueing systems”. There is no generally accepted terminological analogue in Russian-language literature yet. Some authors [3, 4] suggest using a term in Russian that is translated in English as “parallel serving systems”.

In this article we will consider a parallel processing system in which each service request is split into a random number of subrequests. Such queueing systems began to be studied quite a long time ago, back in the early 1980s [5, 6]. The general idea of the functioning of a fork-join system is as follows. Its input receives a random stream of calls (requests). At the time of receipt, any request is divided into a random number of smaller related requests (subrequests), each of which can be processed by one of the system servers.

Currently, a large number of works concerning various aspects of the study of fork-join queueing systems have already been published. They can be found, for example, using a detailed review by A. Thomasian [7] or the recently published monograph by S. Sethuraman [8]. Unfortunately, all these works do not take into account such a practically significant factor as the presence of a limitation on the time for executing requests by the service channel. Meanwhile, when using these models to describe the behavior of a supercomputer, the presence of such restrictions is fundamentally important. For example, according to the “Polytechnic Supercomputer Center” of Peter the Great St. Petersburg Polytechnic University, up to 70% of tasks are removed from calculations ahead of schedule by the dispatcher program.

Restrictions on the execution time of applied tasks on a supercomputer cluster are formed in a rather complex way. The approximate task execution time specified by the user, the dispatch algorithms underlying the work of the task scheduler, and the system of priorities and push-out mechanisms used play a role here. In practice, the limitation on the time it takes to complete tasks appears as some random variable.

### Description of the mathematical model of the queueing system

In this section, we will describe in more detail the queueing system under study, the functional diagram of which is presented in Fig. 1. This system works as follows. The system input receives the simplest (that is, stationary Poisson) flow of requests with  $\lambda$  intensity. In such a flow, all intervals  $\tau_k$  between request are independent and distributed according to the same exponential law with probability density

$$a(\tau) = \lambda e^{-\lambda\tau}, \quad (1)$$

so that the average interval between requests  $\bar{\tau}$  is inversely proportional to the intensity of the flow

$$\bar{\tau} = \frac{1}{\lambda}. \quad (2)$$

A fundamentally important distinctive feature of our system is that it is multi-channel, and contains an unlimited number of service channels (servers), and each request entering it requires servicing

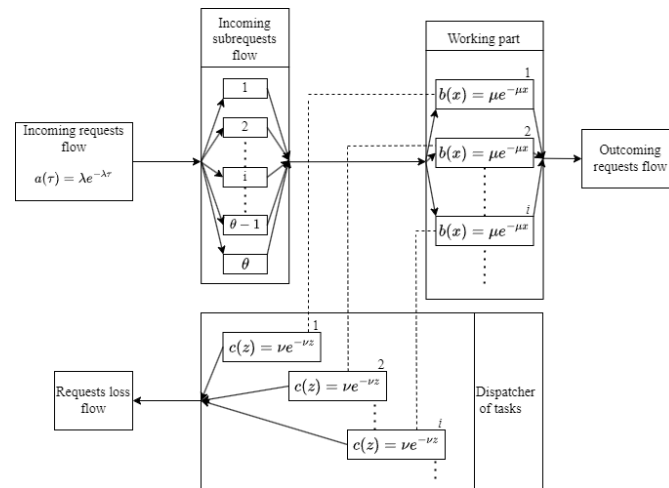


Fig. 1. Functional diagram of the queuing system under study

simultaneously by a random number of servers. The process of executing request can be represented as follows. First, any request can be divided into a number of smaller elementary requests (subrequests), each of which can be executed using only one separate server.

Let us denote the random number of subrequests into which one complete request is divided by the symbol  $\theta$ . The variable  $\theta$  is a discrete integer random variable, which is characterized by the following distribution series:

$$\alpha_i = P\{\theta = i\}, \quad (i = \overline{1, \infty}). \quad (3)$$

Without loss of generality, we assume  $\alpha_0 \equiv 0$ . Here it is assumed that the values  $\theta$  related to different requests are statistically independent and that all of them do not depend on the intervals between the requests appearance  $\tau_k$ .

All service channels (servers) are considered identical, and the execution time of any elementary task (subrequest) on each of them is distributed according to an exponential law with the parameter  $\mu$ . This means that the probability density of service time  $X$  is expressed as

$$b(x) = \mu e^{-\mu x}, \quad (4)$$

and average service time  $\bar{x}$  is given by the formula

$$\bar{x} = \frac{1}{\mu}. \quad (5)$$

The parameter  $\mu$  has the meaning of service intensity.

Our model introduces another important complication that distinguishes it from standard models of queuing systems. The task execution time is limited by the task manager. This dispatcher (special program) issues a constraint in the form of a random variable  $Z$ , distributed according to a given law with a known probability density  $c(z)$ . If the inequality  $Z > X$  is satisfied, then the service process is considered successfully completed. When the opposite inequality is satisfied,  $Z < X$ , the task is removed from execution and sent to the loss flow. In the favorable case, when all the elementary tasks that make up the complete request have been successfully completed, the request as a whole is also considered completed. Otherwise, the entire request is rejected and sent to the loss flow.

This article will examine in more detail the case when the upper bound on the service time is distributed according to the exponential law of the form

$$c(z) = \nu e^{-\nu z}, \quad (6)$$

where  $\nu$  is the intensity of tasks being removed from execution by the dispatcher. In this case, the average time that dispatcher provided to each task for execution is expressed as

$$\bar{z} = \frac{1}{\nu}. \quad (7)$$

The numeric parameter  $\bar{z}$  is one of the most important parameters for the task manager and can be set accordingly.

As a result of the system operating in accordance with the service process discipline described above, the incoming flow of requests is divided into two new flows: the outgoing flow of fully serviced requests, as well as a loss flow, including requests removed from processing by the dispatcher of tasks. It should be mentioned that the calculation of the loss probability, that is, the probability getting into the second stream, represents an important practical problem.

### Obtaining a system of Kolmogorov equations

We will characterize the state of the system described in the previous section using the number of servers (service channels)  $N(t)$  occupied at time  $t$ . The indicated process is Markov process [9]. Let us introduce the probabilities of the states of the considered process

$$P_n(t) = \mathcal{P}\{N(t) = n\}, \quad (n = \overline{0, \infty}). \quad (8)$$

Process  $N(t)$  is ergodic [9], therefore there are final probabilities

$$P_n = \lim_{t \rightarrow \infty} P_n(t), \quad (n = \overline{0, \infty}) \quad (9)$$

that do not depend on the initial state.

The labeled state graph for the process considered has the form shown in Fig. 2. In order not to clutter the figure, it shows in detail the picture of transitions only for three states:  $n = 0$ ,  $n = 1$  and an arbitrary  $n > 0$ . From the transition diagram it is clear that from the state  $n = 0$  you can go to the state  $n > 0$ , located *in the right part of the figure*, with intensity  $\lambda \alpha_n$ , and you can return to the state  $n = 0$  only from the state  $n = 1$ , and with intensity  $\mu$ .

A state with an arbitrary  $n > 0$  can be reached from any state  $i$  located to the left of  $n$ , with intensity  $\lambda \alpha_{n-i}$ , since a new service request is allowed, designed to use an arbitrary number of servers. Similarly, from the state  $n > 0$  you can reach any state  $i > n$ , located to the right of  $n$ , with intensity  $\lambda \alpha_{i-n}$ . In this case, the total intensity of the transition to all states lying to the right of  $n$  will obviously be equal to  $\lambda$ , since the distribution (3) the normalization conditions is always satisfied, which has the form

$$\sum_{i=1}^{\infty} \alpha_i = 1. \quad (10)$$

The only way to return to the state  $n$  on the right is to move back from state  $(n+1)$  with intensity  $(\mu + \nu)(n+1)$ .

Using the labeled state graph in Fig. 2, we can write the Kolmogorov system of equilibrium equations according to the usual rules [9]:

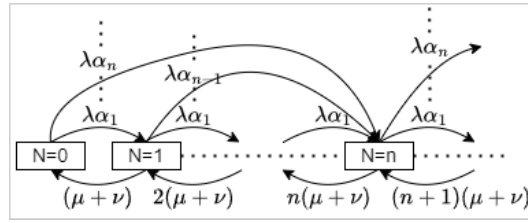


Fig. 2. Labeled state graph for the queueing system under consideration

$$\begin{cases} -\lambda p_0 + (\mu + \nu) p_1 = 0, & n = 0, \\ -[\lambda + (\mu + \nu)n] p_n + (\mu + \nu)(n + 1) p_{n+1} + \sum_{i=0}^{n-1} \lambda \alpha_{n-i} p_i = 0, & n > 0. \end{cases} \quad (11)$$

Equations (11) express the balance of random flows that occurs when the system occupies state number  $n$ . In this case, terms with a minus sign are equal to the intensity of flows leaving state  $n$ , and terms with a plus sign correspond to all possible flows entering this state. Then physical meaning of the equilibrium equations (11) is that in a steady state, for all states of the system, the intensity of incoming flows must be equal to the intensity of outgoing flows.

Equations (11) can be rewritten in a universal form, if we introduce the concept of “empty sum”. A sum is called empty, if its lower summation limit is greater than its upper summation limit. This amount is considered to be zero by definition. Then equations (11) can be rewritten as one equation valid for all  $n \geq 0$ :

$$-[\lambda + (\mu + \nu)n] p_n + (\mu + \nu)(n + 1) p_{n+1} + \sum_{i=0}^{n-1} \lambda \alpha_{n-i} p_i = 0, \quad (n = \overline{0, \infty}). \quad (12)$$

Note, that when  $n = 0$ , the third term in (12) is an empty sum.

Writing equations in the form (12) is more convenient, when using the method of generating functions, which will be discussed below.

At the end of this section, an important note should be made. In our system, the total intensity at which request processing on the server stops is equal to  $\mu + \nu$ . From the point of view of the final probabilities of the state, it does not matter at all with what intensity this request will be fully serviced, and with what intensity it will remain unserved, since it will be removed from service by the dispatcher. If the total intensity of termination of processing requests  $\mu + \nu$  is given, then the form of the Kolmogorov equations and the values of the final state probabilities are also specified uniquely.

Therefore, for example, in a system without a dispatcher but with the same  $\mu + \nu$  service intensity as before, the final probabilities will be the same as in our system. Of course, the loss probabilities in the two above-mentioned systems will be completely different. A special section of this article will be devoted to calculating the probability of losses.

### The generating functions method

The concept of generating functions is a powerful tool for solving problems involving the analysis of numerical sequences, such as the sequence  $\{p_n\}_{n=0}^{\infty}$  of state probabilities in our case. The idea of the method is to move from considering an infinite set of variables  $p_n$  depending on an integer index  $n$ , to a single function depending on a continuously changing argument.

Let us introduce the generating function  $G(z)$  for probabilities  $p_n$  in the form of the following power series

$$G(z) = \sum_{n=0}^{\infty} p_n z^n, \quad (|z| \leq 1). \quad (13)$$

The function  $G$  is guaranteed to exist at least in region  $|z| \leq 1$ , since the series

$$\sum_{n=0}^{\infty} p_n = 1 \quad (14)$$

is undoubtedly convergent.

To calculate the function  $G(z)$ , multiply both sides of equality (12) by  $z^n$  and sum over all  $n$  from zero to infinity. The resulting sums are converted as follows:

$$\sum_{n=0}^{\infty} n p_n z^n = z \sum_{n=0}^{\infty} p_n n z^{n-1} = z \frac{dG(z)}{dz}, \quad (15)$$

$$\sum_{n=0}^{\infty} (n+1) p_{n+1} z^n = \sum_{n=1}^{\infty} n p_n z^{n-1} = \frac{dG(z)}{dz}. \quad (16)$$

Double sum is calculated by changing the order of summation

$$\begin{aligned} \sum_{n=0}^{\infty} \sum_{i=0}^{n-1} \alpha_{n-i} p_i z^n &= \sum_{n=1}^{\infty} \sum_{i=0}^{n-1} \alpha_{n-i} p_i z^n = \sum_{i=0}^{\infty} \sum_{n=i+1}^{\infty} \alpha_{n-i} p_i z^n = \\ &= \sum_{i=0}^{\infty} p_i z^i \sum_{n=i+1}^{\infty} \alpha_{n-i} z^{n-i} = \sum_{i=0}^{\infty} p_i z^i \sum_{k=1}^{\infty} \alpha_k z^k = G(z) Q(z). \end{aligned} \quad (17)$$

Here by  $Q(z)$  we mean the generating function of the  $\alpha_i$  probabilities

$$Q(z) = \sum_{k=1}^{\infty} \alpha_k z^k, \quad (|z| \leq 1). \quad (18)$$

After these transformations, equation (12) takes the form

$$-\lambda G - (\mu + \nu) z \frac{dG}{dz} + (\mu + \nu) \frac{dG}{dz} + \lambda G Q = 0. \quad (19)$$

Expressing the derivative  $\frac{dG}{dz}$  from here, we obtain the following differential equation for the function  $G(z)$ :

$$\frac{dG}{dz} = \frac{\lambda}{\mu + \nu} \frac{1 - Q(z)}{1 - z} G. \quad (20)$$

It should be noted that the function  $Q(z)$  here is known, since all probabilities  $\alpha_i$  are specified according to the conditions of the problem.

Equation (20) must be solved under the initial condition

$$G(1) = 1, \quad (21)$$

which is a universal general property of all generating functions, resulting from the normalization condition (14) for the probabilities  $p_n$ .

Solving equation (20) under initial condition (21) we finally obtain:

$$G(z) = e^{\frac{\lambda}{\mu+\nu} \int_1^z \frac{1-Q(s)}{1-s} ds}. \quad (22)$$

Now let us look at some examples of applying the solution we just obtain.

**Example 1.** Each request is processed by only one server.

In this case, the probabilities  $\alpha_i$  are given in the form

$$\alpha_i = \delta_{i,1}, \quad (i = \overline{1, \infty}), \quad (23)$$

where  $\delta_{i,k}$  denotes the Kronecker delta symbol, and the generating function (18) is reduced to the simplest linear function

$$Q(z) = z. \quad (24)$$

Then solution (22) is nothing more than an exponential of the form

$$G(z) = e^{\frac{\lambda}{\mu+\nu}(z-1)}, \quad (25)$$

expanding which in powers of  $z$  we get

$$p_n = e^{-\frac{\lambda}{\mu+\nu}} \left( \frac{\lambda}{\mu+\nu} \right)^n, \quad (n = \overline{0, \infty}). \quad (26)$$

Thus, if each request is processed by only one server, then the total number of busy servers will be distributed according to Poisson's law with the parameter  $\frac{\lambda}{\mu+\nu}$ . This result is well known from the classical queueing theory as applied to the system of  $M/M/\infty$  class [9].

**Example 2.** Geometric law of distribution of the number of involving servers.

Let us assume that the number  $\theta$  of servers used to process a single request is distributed according to a geometric law

$$\alpha_i = \mathcal{P}\{\theta = i\} = (1-\alpha)\alpha^{i-1}, \quad (i = \overline{1, \infty}), \quad (27)$$

where  $0 < \alpha < 1$  denotes the parameter of the geometric law. The geometric law is interesting because it is the only discrete law that has the property of no aftereffect [9].

It is easy to show that the generating function (18) of the following form corresponds to law (27):

$$Q(z) = \frac{(1-\alpha)z}{1-\alpha z}, \quad (28)$$

wherein



$$1 - Q(z) = \frac{(1-z)}{1 - \alpha z}. \quad (29)$$

Substituting expression (29) into formula (22), we obtain

$$G(z) = e^{\frac{\lambda}{\mu + \nu} \int_1^z \frac{ds}{1 - \alpha s}}. \quad (30)$$

The integral in the exponent (30) is tabular. Omitting a number of simple intermediate calculations, we get

$$G(z) = \left( \frac{1 - \alpha}{1 - \alpha z} \right)^{\frac{\lambda}{(\mu + \nu)\alpha}}. \quad (31)$$

**Example 3.** Average number of busy servers.

Using the general formula (20) for the generating function, one can easily find the average number of busy servers in the entire system in steady state. It is well known, that

$$\bar{n} = M[N] = \left. \frac{dG(z)}{dz} \right|_{z=1}, \quad (32)$$

also it's obvious that

$$\lim_{z \rightarrow 1} \frac{1 - Q(z)}{1 - z} = Q'(1) = \bar{\theta}, \quad (33)$$

where  $\bar{\theta}$  denotes the average number of servers per processed request.

Passing to the limit as  $z \rightarrow 1$  in formula (20) leads us to the final expression

$$\bar{n} = \frac{\lambda}{\mu + \nu} \bar{\theta}. \quad (34)$$

Thus, the average number of busy servers in the system is directly proportional to the arrival rate and the average number of servers per request and inversely proportional to the sum of the execution and reset by the dispatcher intensities.

### Calculating the probability of losing a service request

In the classical  $M/M/\infty$  system, loss of requests is impossible, because such a system is an immediate queueing system with an infinite number of servers. For any newly received request for service, there is always a free server that begins to process it, and the service is always brought to its logical end [9].

In our system, processing also starts immediately after the arrival of request, but it may not be completed due to the intervention of the dispatcher (task scheduler), which dumps partially completed tasks into the loss flow. As a result, each request leaving the system can end up in one of two flows: either in the outcoming flow of fully serviced requests, or the loss flow formed by requests removed from service. In this section we will calculate the probability of being in the first of these flows.

Let us denote by  $A$  the random event that the request is eventually fully serviced. We denote the probability of event  $A$  by  $P_{serv}$ . Using the total probability formula, we can write

$$P_{serv} = \sum_{k=1}^{\infty} \mathcal{P}\{A|\theta = k\} \mathcal{P}\{\theta = k\}. \quad (35)$$

Here  $\theta$  is the random number of servers that will be required to process the request in question. The first factor in each term of the sum (35) has the meaning of the conditional probability of event  $A$ , provided that exactly  $k$  servers are required, and the second factor, according to (3), is equal to  $\alpha_k$ .

Let us calculate the conditional probabilities appearing in (35). To do this, we first find the partial probability  $p$  that one server will successfully complete the processing of the subrequest assigned to it. It's not hard to understand that

$$p = \mathcal{P}\{X < Z\}, \quad (36)$$

where  $X$  is the execution time of the subrequest, and  $Z$  is the limitation on this time on the part of the dispatcher.

As a result, we obtain a simple expression for the probability (35)

$$P_{serv} = \sum_{k=1}^{\infty} \alpha_k p^k, \quad (37)$$

in which  $p$  remains unknown for now.

To calculate the probability  $p$ , we introduce a joint law of distribution of random variables  $X$  and  $Z$ , which we denote by  $f_{xz}(x, z)$ . According to the assumptions made at the beginning of this article, we get

$$f_{xz}(x, z) = f_x(x) f_z(z) = b(x) c(z) = \mu \nu e^{-(\mu x + \nu z)}. \quad (38)$$

Probability (36) is represented as an integral

$$p = \int_0^{\infty} \int_x^{\infty} f_{xz}(x, z) dz dx, \quad (39)$$

which, due to the fulfilment of (38), can be easily calculated

$$p = \frac{\mu}{\mu + \nu}. \quad (40)$$

Substituting this expression into (37), we obtain

$$P_{serv} = \sum_{k=1}^{\infty} \alpha_k \left( \frac{\mu}{\mu + \nu} \right)^k = Q\left( \frac{\mu}{\mu + \nu} \right), \quad (41)$$

where  $Q(z)$  is the generating function (18) for the probabilities  $\alpha_k$ , describing the distribution of the number  $\theta$  of servers involved in one complete request processing.

If we use the original formulas (5) and (7), then the expression (40) can be rewritten in the equivalent form

$$p = \frac{\bar{z}}{\bar{z} + \bar{x}}, \quad (42)$$

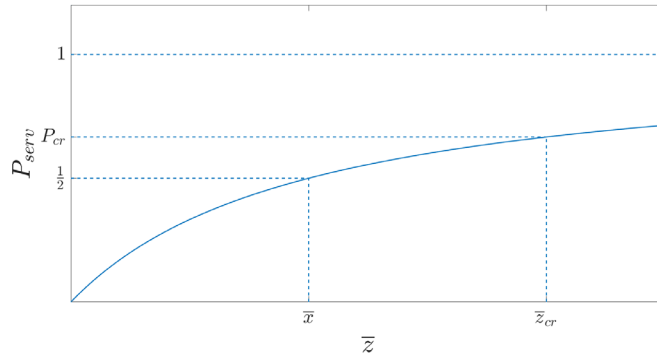


Fig. 3. Dependence of probability  $P_{serv}$  on the average time of limitation  $\bar{z}$  for Example 4 for  $\bar{x} = 1$

which gives an alternative expression for the desired probability

$$P_{serv} = Q\left(\frac{\bar{z}}{\bar{z} + \bar{x}}\right). \quad (43)$$

Now let us consider some examples that explain the application of the resulting formulas.

**Example 4.** Requests without subrequests.

In Example 1 the probabilities are given in the form (23), which leads to the generating function given (24), and then for the probability of successfully completing tasks we get the simplest expression

$$P_{serv} = \frac{\bar{z}}{\bar{z} + \bar{x}}. \quad (44)$$

The dependence of probability (44) on the average time of limitation is shown in Fig. 3. The graph shows that when inequality  $\bar{z} > \bar{x}$  is satisfied, the probability of successful completion of tasks will be greater than the probability of their early reset. With the opposite sign of inequality  $\bar{z} < \bar{x}$ , on the contrary, the reset will, on average, occur more often than the normal standard completion of calculations.

If some critical value for probability (44) is specified in the form  $P_{cr}$  and it is required that probability  $P_{serv}$  exceed it, then  $\bar{z}$  must be higher than the critical value

$$z_{cr} = \bar{x} \frac{P_{cr}}{1 - P_{cr}}. \quad (45)$$

**Example 5.** Requests that are divided into a fixed nonrandom number of subrequests.

Let us assume that the probabilities  $\alpha_i$  are expressed in the form

$$\alpha_i = \delta_{i,k}, \quad (46)$$

where  $k$  is some integer positive, so that the generating function (18) turns out to be equal to the specified integer power of  $z$

$$Q(z) = z^k. \quad (47)$$

Then using formula (43) we get

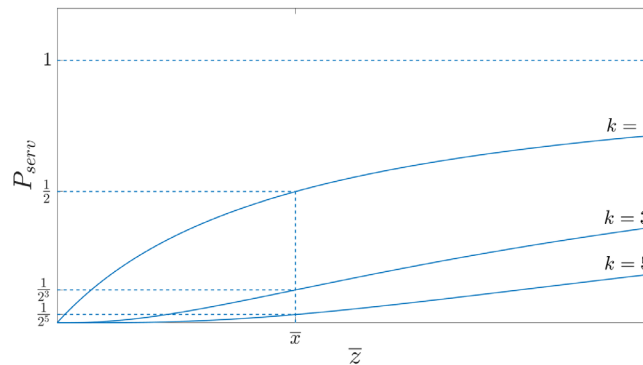


Fig. 4. Dependence of probability  $P_{serv}$  on the average time of limitation  $\bar{z}$  for Example 5 for  $k = 1, 3, 5$  and  $\bar{x} = 1$

$$P_{serv} = \left( \frac{\bar{z}}{\bar{z} + \bar{x}} \right)^k. \quad (48)$$

A graph of dependence of  $P_{serv}$  on  $\bar{z}$  for different  $k$ , relating to this case, is shown in Fig. 4.

The behavior of the curves in Fig. 4 shows that an increase in the number of servers involved leads to a decrease of the probability of successful completion of tasks.

### Conclusion

In this article, we used an appropriately modified classical model of the  $M/M/\infty$  queueing system to mathematically describe the behavior of a supercomputer cluster. In this model, each service request needs to use a random number of servers (processors) simultaneously. Service occurs according to an exponential law, identical for all running servers. The incoming flow is assumed to be the simplest (stationary Poisson). The execution time of each service request is limited by the dispatcher (task scheduler) by a certain random variable distributed according to an exponential law. If during this time the task has not yet been completed, then the task is discarded from service and falls into the loss flow. The paper provides an analytical solution to the described problem using the method of generating functions. The main contribution of this article is that it explicitly obtained the law of distribution of the number of busy servers, as well as the probability of successful completion of tasks in the form of a certain function of the average time limitation. The probability that any task will be completed to its logical end is one of the main indicators of the quality of a computing cluster.

### REFERENCES

1. **Vishnevskii V.M.** Teoreticheskie osnovy proektirovaniia komp'iuternykh setei [Theoretical foundations of computer network design]. Moscow: Tekhnosfera, 2003, 506 p. (in Russ.)
2. **Harchol-Balter M.** Performance modeling and design of computer systems: Queueing theory in action. Cambridge: University Press, 2013, 576 p.
3. **Gorbunova A.V., Zaryadov I.S., Samouylov K.E., Sopin E.S.** A Survey on Queueing Systems with Parallel Serving of Customers. Discrete and Continuous Models and Applied Computational Science. 2017, Vol. 25, no. 4, pp. 350–362. DOI: 10.22363/2312-9735-2017-25-4-350-362
4. **Gorbunova A.V., Zaryadov I.S., Samouylov K.E.** A Survey on Queueing Systems with Parallel Serving of Customers. Part II. Discrete and Continuous Models and Applied Computational Science. 2018, Vol. 26, no. 1, pp. 13–27. DOI: 10.22363/2312-9735-2018-26-1-13-27

5. **Green L.** A queueing system in which customers require a random number of servers. *Operations Research*. 1980, Vol. 28, no. 6, pp. 1335–1346.
6. **Green L.** Comparing operating characteristics of queues in which customers require a random number of servers. *Management science*. 1981, Vol. 27, no. 1, pp. 65–74.
7. **Thomasian A.** Analysis of fork/join and related queueing systems. *ACM Computing Surveys*. 2014, Vol. 47, no. 2, pp. 1–71. DOI: 10.1145/2628913
8. **Sethuraman S.** *Analysis of Fork-Join Systems: Network of Queues with Precedence Constraints*. Boca Raton: CRC Press, 2022, 104 p. DOI: 10.1201/9781003150077
9. **Kleinrock L.** *Queueing systems*. NY: Wiley-Interscience, 1975, 417 p.

### INFORMATION ABOUT AUTHORS / СВЕДЕНИЯ ОБ АВТОРАХ

**Zayats Oleg I.**

**Заяц Олег Иванович**

E-mail: zay.oleg@gmail.com

**Vaksheev Vitaliy E.**

**Бакшеев Виталий Ефимович**

E-mail: yaming2110@gmail.com

**Zaborovsky Vladimir S.**

**Заборовский Владимир Сергеевич**

E-mail: vlad2tu@yandex.ru

**Muliukha Vladimir A.**

**Мулюха Владимир Александрович**

E-mail: vladimir.muliukha@spbstu.ru

ORCID: <https://orcid.org/0000-0002-3583-7324>

*Submitted: 30.07.2024; Approved: 26.09.2024; Accepted: 04.10.2024.*

*Поступила: 30.07.2024; Одобрена: 26.09.2024; Принята: 04.10.2024.*