mathematics

Article

Investigation of the Fork–Join System with Markovian Arrival Process Arrivals and Phase-Type Service Time Distribution Using Machine Learning Methods

Vladimir Mironovich Vishnevsky [1,*,†], Valentina Ivanovna Klimenok [2,†], Aleksandr Mikhailovich Sokolov [1,†] and Andrey Alekseevich Larionov [1,†]

[1] Institute of Control Sciences of Russian Academy of Sciences, 117997 Moscow, Russia; aleksandr.sokolov@phystech.edu (A.M.S.); larioandr@gmail.com (A.A.L.)
[2] Department of Applied Mathematics and Computer Science, Belarusian State University, 220030 Minsk, Belarus; klimenok@bsu.by
[*] Correspondence: vishn@inbox.ru
[†] These authors contributed equally to this work.

Abstract: This paper presents a study of fork–join systems. The fork–join system breaks down each customer into numerous tasks and processes them on separate servers. Once all tasks are finished, the customer is considered completed. This design enables the efficient handling of customers. The customers enter the system in a MAP flow. This helps create a more realistic and flexible representation of how customers arrive. It is important for modeling various real-life scenarios. Customers are divided into $K \geq 2$ tasks and assigned to different subsystems. The number of tasks matches the number of subsystems. Each subsystem has a server that processes tasks, and a buffer that temporarily stores tasks waiting to be processed. The service time of a task by the k-th server follows a PH (phase-type) distribution with an irreducible representation (β_k, S_k), $1 \leq k \leq K$. An analytical solution was derived for the case of $K = 2$ when the input MAP flow and service time follow a PH distribution. We have efficient algorithms to calculate the stationary distribution and performance characteristics of the fork–join system for this case. In general cases, this paper suggests using a combination of Monte Carlo and machine learning methods to study the performance of fork–join systems. In this paper, we present the results of our numerical experiments.

Keywords: fork–join system; Markovian arrival process; phase-type distribution; stationary performance characteristics; machine learning

MSC: 60K25

check for updates

Citation: Vishnevsky, V.M.; Klimenok, V.I.; Sokolov, A.M.; Larionov, A.A. Investigation of the Fork–Join System with Markovian Arrival Process Arrivals and Phase-Type Service Time Distribution Using Machine Learning Methods. *Mathematics* **2024**, *12*, 659. https://doi.org/10.3390/math12050659

Academic Editors: Tadashi Dohi and Antonio Di Crescenzo

Received: 9 November 2023
Revised: 13 February 2024
Accepted: 21 February 2024
Published: 23 February 2024

1. Introduction

Queueing systems with data partitioning and resynchronization (generally accepted as fork–join systems) are natural models of various computer and communication systems, particularly systems that perform parallel computing, customer processing in distributed databases, and parallel disk access [1–4]. The key feature of these systems is that after a customer enters the system, it is divided into a particular set of K tasks that are serviced by K parallel servers. A customer leaves the system when all subsystems complete all customer tasks. Thus, tasks corresponding to the same customer are joined before departing the system, and, accordingly, the service time of a customer corresponds to the maximum time spent in the system by one of the tasks.

In the case of infinite buffer, difficulties arise in calculating the joint stationary distribution of queue lengths, even under the most straightforward assumptions about the nature of the input flow and service times. Finding such a distribution is the subject of [5], where the case $K = 2$, infinite buffer to servers, stationary Poisson flow, and exponential distributions of service times are considered. For such a system, a functional equation for the generating function of the joint distribution of queue lengths is obtained, and questions of the asymptotic behavior of joint probabilities are investigated. The results of [5] are used in [6], where a formula is derived for the mean sojourn time in the mentioned system under the simplifying assumption that the service rates on both servers are equal. In [7], the authors consider a fork–join system with two subsystems and more general time distributions between hyperexponential arrivals and the Erlang distribution of service times.

Due to the complexity of the problem of finding the sojourn time distribution in fork–join systems, in most works, the authors limit themselves to analyzing only mean sojourn time [5–8], in particular, finding upper or lower bounds for this time [9–11]. The overview of research in this area until 2014 can be found in [12].

Extensive applications of fork–join systems not only in computing systems and networks with parallel services but also in other industries, including the banking sector, medicine [13,14], and some others [15–18], stimulated further research in this area. The authors in [19] considered a method for calculating the stationary distribution of a fork–join system with a Markovian arrival, $K \geq 2$ homogeneous servers characterized by an exponential service time distribution, and infinite buffer, which involves solving a nonlinear matrix integral equation and the truncation of the state space of the process describing the operation of the system. The main results are obtained for the case $K = 2$. In [20], a fork–join system with parallel servicing of customers ($K = 2$), Poisson input flow, and PH-distribution of service time is considered. The conditions for a stationary mode, the main characteristics of system performance, and analytical expressions for the lower and upper bounds of the mean sojourn time in the system are found. An analysis of more general cases of studying fork–join systems, particularly with an arbitrary distribution of service time, was carried out in the literature only using approximate methods [21–24]. In the articles [25,26], systems with multiple resources are explored. A distinctive feature of such systems is that tasks from the customer simultaneously occupy a random number of servers. It depends on the number of tasks in the customer. In the article [27], you can familiarize yourself with the use of neural networks for the analysis of fork–join systems and other complex problems of queuing theory (for example, [28,29]).

The previous review showed that, despite using $K = 2$ and making basic assumptions about the input flow and service time, a detailed analytical solution for determining the stationary characteristics of a fork–join system was not found. This article is important because it analyzes a fork–join system, considering the different assumptions about input flows and service time distribution. This research holds significant practical and theoretical value, making it truly novel and original. This work holds both practical and theoretical significance, marking a crucial advancement in the field. For the case where $K = 2$, we obtained analytical expressions for the probability of losses, average queue length, and the distribution function of the response time for the first time. If $K \geq 2$, a new method is created and used. This method combines machine learning and the Monte Carlo methods. It efficiently calculates the characteristics of a complex fork–join system on a large scale. It is particularly useful for building real systems that require effective parallel task management.

This paper considers the various architectures of fork–join systems in more general assumptions about input flow and the distribution of service times. Customers enter the system according to the Markovian arrival process (MAP). Service times have a phase distribution (PH distribution). We organized the article as follows. Section 2 presents the problem statement. Section 3 is devoted to implementing a mathematical algorithm for calculating the stationary distribution and characteristics for the particular case of a fork–join system with $K = 2$ with an input MAP flow and PH—service time distribution. Section 4 describes a method for solving the general problem of studying a fork–join system using simulation and machine learning methods. Section 5 presents the numerical results of the study.

2. Problem Statement

We consider a fork–join queueing system with $K \geq 2$ servers in parallel (see Figure 1). Customers enter the system in a Markovian arrival process (MAP), which is specified by the underlying process $\nu_t, t \geq 0$, which is a Markov chain with the state space $\{0, 1, \ldots, W\}$, and the matrices D_0 and D_1. The non-diagonal entries of the matrix D_0 are the rates of transitions of the chain $\nu_t, t \geq 0$, which are not accompanied by the generation of a customer, while the entries of the matrix D_1 are the rates of transitions of the chain which are accompanied by the generation of a customer. The matrix $D(1) = D_0 + D_1$ represents an infinitesimal generator of the chain $\nu_t, t \geq 0$. The arrival rate λ is defined as $\lambda = \theta D_1 e$, where θ is a vector of the stationary distribution of the process ν_t, which is defined as the unique solution of the system of linear algebraic equations $\theta D(1) = 0$, $\theta e = 1$. Here and further, e is a column vector consisting of ones, and 0 is a row vector consisting of zeros. A more detailed description of the MAP can be found, for example, in [30,31].

An arriving customer entering the system forks into K tasks to be served in K independent subsystems $G_1, G_2, \ldots, G_k$. The service time of a task by the k-th server has a PH (phase-type) distribution with an irreducible representation $(\boldsymbol{\beta}_k, S_k)$. This means that the service process on the k-th server is managed by the Markov chain $m_t^{(k)}$, $t \geq 0$, with the state space $\{1, \ldots, M^{(k)}, M^{(k)} + 1\}$, where the state $M^{(k)} + 1$ is an absorbing one. The rates of transitions in the set of unessential states are given by the matrices S_k and the transition rates to absorbing states are given by the vectors $\mathbf{S}_0^{(k)} = -S_k\mathbf{e}$. More details about the PH can be found, for example, in [31,32]. A customer that arrives when the buffer of one of the systems $G_i, i = 1, 2, \ldots, K$ is full leaves the system without a service (it is considered to be lost). A customer accepted into the system leaves the system as soon as all servers complete their task services. Thus, tasks corresponding to the same customer are joined before departing the system.

The main stationary characteristics of the performance of the described fork–join system are found, including the sojourn time in the system, average queue lengths, and the loss probability of customers.

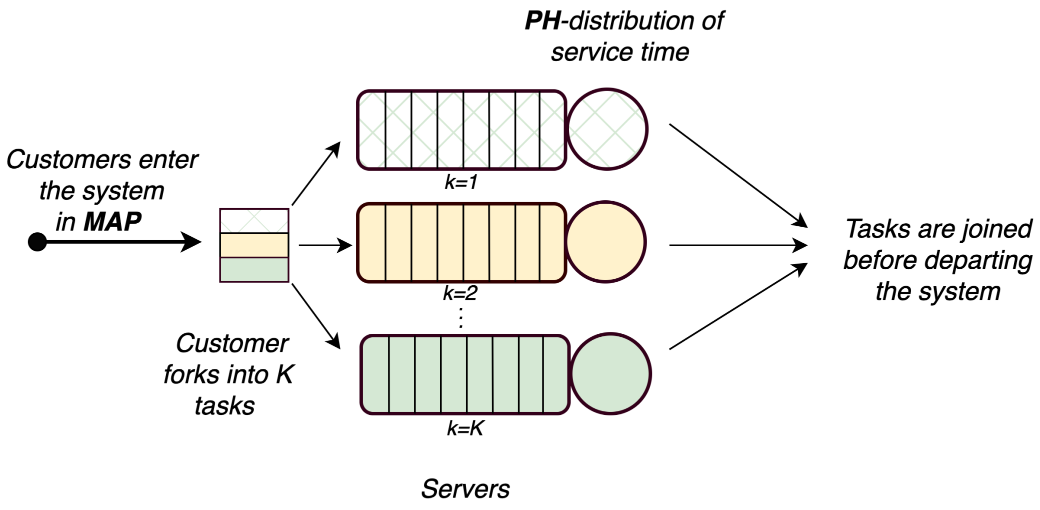
***PH*-distribution of service time**
*Customers enter the system in **MAP***
Customer forks into K tasks
k=1
k=2
k=K
Tasks are joined before departing the system
Servers

Figure 1. A fork–join system when a customer consists of $K > 2$ tasks.

3. Study of the Characteristics of a Fork–Join System with Parallel Servicing of Tasks ($K = 2$)

The analysis of this particular case is of theoretical interest, because the study of such a system under the assumption of an incoming MAP flow and a PH distribution of service time has not been previously considered in the literature. In addition, the numerical results of this section will be used to validate a simulation model that describes the functioning of the general fork–join system ($K \geq 2$) considered in this article.

3.1. Markov Chain Describing the Process of System Functioning

Assume that the first service device ($k = 1$) has infinite buffer, and the capacity of the second ($k = 2$) is limited.

At time t, let the following:

- i_t denotes the number of tasks in the subsystem G_1, $i \geq 0$;
- j_t denotes the number of tasks in the subsystem G_2, $j = \overline{0, J}$;
- ν_t denotes the states of the underlying process of the MAP, $\nu_t = \overline{0, W}$;
- $m_t^{(k)}$ denotes the states of underlying process of the PH service process on the kth server, $m_t^{(k)} = \overline{1, M^{(k)}}$, $k = 1, 2$.

The operation of the system is described by a regular irreducible Markov chain $\xi_t, t \geq 0$, with the state space

$$\Omega = \{(0,0,\nu), \nu = \overline{0,W}\} \bigcup \{(i,0,\nu,m^{(1)}), i > 0, \nu = \overline{0,W}, m^{(1)} = \overline{1,M^{(1)}}\} \bigcup$$

$$\{(0,j,\nu,m^{(2)}), j = \overline{0,J}, \nu = \overline{0,W}, m^{(2)} = \overline{1,M^{(2)}}\} \bigcup$$

$$\{(i,j,\nu,m^{(1)},m^{(2)}), i > 0, j = \overline{1,J}, \nu = \overline{0,W}, m^{(1)} = \overline{1,M^{(1)}}, m^{(2)} = \overline{1,M^{(2)}}\}.$$

Let us arrange the states of the chain ξ_t in the lexicographic order and denote by $Q_{i,i'}$ the matrix of transition rates of the chain from the states with the value i of the first component to the states with the value i' of this component. We also denote by Q_r the matrix that is defined as $Q_r = Q_{i,i+r-1}, i \geq 1, r \geq 0$.

To proceed to writing the infinitesimal generator Q of the Markov chain ξ_t, $t \geq 0$, we introduce the following notations:

- I (O) is an identity (zero) matrix;
- $\bar{W} = W + 1$;
- $\otimes(\oplus)$ is a symbol of the Kronecker product (sum) of matrices;
- $diag\,\{Z_l,\ l = \overline{1, L}\,\}$ is a diagonal matrix with diagonal blocks Z_l;
- $diag^-\{Z_l, l = \overline{0, L}\}$ is a sub-diagonal matrix with sub-diagonal blocks Z_l;
- $diag^+\{Z_l, l = \overline{0, L}\}$ is an over-diagonal matrix with over-diagonal blocks Z_l;

Lemma 1. *The infinitesimal generator Q of the Markov chain ξ_t, $t \geq 0$, has a block tridiagonal structure*

$$Q = \begin{pmatrix} Q_{0,0} & Q_{0,1} & O & O & O & \cdots \\ Q_{1,0} & Q_1 & Q_2 & O & O & \cdots \\ O & Q_0 & Q_1 & Q_2 & O & \cdots \\ O & O & Q_0 & Q_1 & Q_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix};$$

where non-zero blocks $Q_{i,i'}$ have the following form:

- *$Q_{0,0}$ is a square matrix of size $\bar{W}(1 + JM_2)$,*

$$Q_{0,0} = diag\{D_0,\ D_0 \oplus S_2,\ D_0 \oplus S_2,\ \ldots,\ D_0 \oplus S_2,\ D(1) \oplus S_2\} +$$

$$diag^-\{I_{\bar{W}} \otimes \mathbf{S}_0^{(2)},\ I_{\bar{W}} \otimes \mathbf{S}_0^{(2)}\boldsymbol{\beta}_2,\ I_{\bar{W}} \otimes \mathbf{S}_0^{(2)}\boldsymbol{\beta}_2,\ \ldots, I_{\bar{W}} \otimes \mathbf{S}_0^{(2)}\boldsymbol{\beta}_2\};$$

- *$Q_{0,1}$ is a matrix of size $\bar{W}(1 + JM_2) \times \bar{W}M_1(1 + JM_2)$,*

$$Q_{0,1} = diag^+\{D_1 \otimes \boldsymbol{\beta}_1 \otimes \boldsymbol{\beta}_2\, D_1 \otimes \boldsymbol{\beta}_1 \otimes I_{M_2},\ D_1 \otimes \boldsymbol{\beta}_1 \otimes I_{M_2}, \ldots,\ D_1 \otimes \boldsymbol{\beta}_1 \otimes I_{M_2}\};$$

- *$Q_{1,0}$ is a matrix of size $\bar{W}M_1(1 + JM_2) \times \bar{W}(1 + JM_2)$,*

$$Q_{1,0} = diag\{I_{\bar{W}} \otimes \mathbf{S}_0^{(1)},\ I_{\bar{W}} \otimes \mathbf{S}_0^{(1)} \otimes I_{M_2},\ I_{\bar{W}} \otimes \mathbf{S}_0^{(1)} \otimes I_{M_2},\ \ldots,\ I_{\bar{W}} \otimes \mathbf{S}_0^{(1)} \otimes I_{M_2}\};$$

- *Q_0 is a square matrix of size $\bar{W}M_1(1+JM_2)$,*

$$Q_0 = diag\{I_{\bar{W}} \otimes \mathbf{S}_0^{(1)},\, I_{\bar{W}} \otimes \mathbf{S}_0^{(1)}\boldsymbol{\beta}_1 \otimes I_{M_2},\, I_{\bar{W}} \otimes \mathbf{S}_0^{(1)}\boldsymbol{\beta}_1 \otimes I_{M_2},\, \dots,\, I_{\bar{W}} \otimes \mathbf{S}_0^{(1)}\boldsymbol{\beta}_1 \otimes I_{M_2}\};$$

- *Q_1 is a square matrix of size $\bar{W}M_1(1+JM_2)$,*

$$Q_1 = diag\{D_0 \oplus S^{(1)},\, D_0 \oplus S_1 \oplus S_2,\, D_0 \oplus S_1 \oplus S_2,\, \dots,\, D_0 \oplus S_1 \oplus S_2,\, D(1) \oplus S_1 \oplus S_2\} +$$

$$diag^{-}\{I_{\bar{W}M_1} \otimes \mathbf{S}_0^{(2)},\, I_{\bar{W}M_1} \otimes \mathbf{S}_0^{(2)}\boldsymbol{\beta}_2,\, I_{\bar{W}M_1} \otimes \mathbf{S}_0^{(2)}\boldsymbol{\beta}_2,\, \dots,\, I_{\bar{W}M_1} \otimes \mathbf{S}_0^{(2)}\boldsymbol{\beta}_2\};$$

- *Q_2 is a square matrix of size $\bar{W}M_1(1+JM_2)$,*

$$Q_2 = diag^{+}\{D_1 \otimes I_{M_1} \otimes \boldsymbol{\beta}_2,\, D_1 \otimes I_{M_1M_2},\, D_1 \otimes I_{M_1M_2},\, \dots,\, D_1 \otimes I_{M_1M_2}\}.$$

Proof. We represent each of the matrices $Q_{i,i'}$ in the block form $Q_{i,i'} = (Q_{i,i'})_{j,j'=\overline{0,J}}$ and explain the probabilistic meaning of their blocks.

The matrix $Q_{0,0}$ describes the rates of transitions of the Markov chain ξ_t, $t \geq 0$, that do not lead to an increase in the number of tasks in the subsystems G_1 and G_2. In this matrix $(Q_{0,0})_{j,j'} = O, j' > j, j' < j-1$, and the others blocks are described as follows:

$(Q_{0,0})_{0,0}$ contains the rates of transitions of the chain ξ_t caused by the idle transitions of the MAP (the rates are described by the matrix D_0);

$(Q_{0,0})_{j,j-1}, j = \overline{1,J}$, contains the rates of transitions of the chain ξ_t accompanied by the end of servicing on server 2 (the rates describe the matrix $I_{\bar{W}} \otimes \mathbf{S}_0^{(2)}$, if $j=1$, and the matrix $I_{\bar{W}} \otimes \mathbf{S}_0^{(2)}\boldsymbol{\beta}_2$, if $j>1$);

$(Q_{0,0})_{j,j}, j = \overline{1,J-1}$, contains the rates of transitions of the chain ξ_t, that do not lead to the arrival of a customer in the MAP or the completion of the service on server 2. Such transitions are made during idle transitions of the underlying process MAP or the underlying process of the service on server 2 (the rates are described by the matrix $D_0 \oplus S_2$);

$(Q_{0,0})_{J,J}$ contains the rates of transitions of the chain ξ_t that are caused by a change in the state of the underlying process of the MAP (accompanied or not accompanied by the arrival of a customer) or idle transitions of the underlying process of the PH service on server 2 (the rates are described by the matrix $D(1) \oplus S_2$);

The matrix $Q_{0,1} = (Q_{0,1})_{j,j'=\overline{0,J}}$ describes the rates of transitions of the Markov chain ξ_t which entail the arrival of a customer in the MAP. It has a block over-diagonal structure, where, for $j=0$, the over-diagonal blocks describe the transitions of the underlying process of the MAP accompanied by the establishment of the initial phase of the service processes on the first and on the second servers (the matrix $D_1 \otimes \boldsymbol{\beta}_1 \otimes \boldsymbol{\beta}_2$)) or, for $0 < j < J$, transitions of the underlying process of the MAP accompanied by the establishment of the initial phase of the service process on the first server and an increase in the queue in the system G_2 by one (the matrix $D_1 \otimes \boldsymbol{\beta}_1 \otimes I_{M_2}$).

The matrix $Q_{1,0}$ describes the rates of transitions of the Markov chain ξ_t that lead to the completion of the service on the second server ((the rates are described by the matrix $I_{\bar{W}} \otimes \mathbf{S}_0^{(1)}$, if $j=1$, and by the matrix $I_{\bar{W}} \otimes \mathbf{S}_0^{(1)}\boldsymbol{\beta}_1 \otimes I_{M_2}$, if $j>1$);

The blocks Q_0, Q_1, Q_2 of the generator are described similarly to the blocks $Q_{1,0}, Q_{0,0}, Q_{0,1}$ but taking into account the fact that, at the time of the chain transition, the system G_1 is not empty. □

Corollary 1. *The process $\xi_t, t \geq 0$, belongs to the class of quasi- birth-and-death (QBD) processes. Proof of the corollary follows from the structure of the infinitesimal generator and the definition of a quasi- birth-and-death process given in [32].*

3.2. Ergodicity Condition

The criterion for the existence of a stationary regime in the system under consideration coincides with the necessary and sufficient condition for the ergodicity of the Markov chain ξ_t, $t \geq 0$. This condition is defined in the following theorem.

Theorem 1. *The necessary and sufficient condition for the ergodicity of the Markov chain $\xi_t, t \geq 0$, is the fulfillment of the inequality*

$$\lambda(1 - y_J\mathbf{e}) < \mu_1, \tag{1}$$

where

$$y_J = \frac{(\frac{\lambda}{\mu_2})^J/J!}{\sum\limits_{j=0}^{J}(\frac{\lambda}{\mu_2})^j/j!}. \tag{2}$$

Proof. According to [33], a necessary and sufficient condition for ergodicity is the fulfillment of the inequality

$$\mathbf{x}Q_2\mathbf{e} < \mathbf{x}Q_0\mathbf{e}, \tag{3}$$

where the stochastic vector $\mathbf{x}$ is a solution to a system of linear algebraic equations

$$\mathbf{x}(Q_0 + Q_1 + Q_2) = \mathbf{0}, \tag{4}$$

$$\mathbf{x}\mathbf{e} = 1.$$

Let us represent the vector $\mathbf{x}$ in the form $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_J)$, where the vector $\mathbf{x}_0$ has order an of $\bar{W}M_1$ and vectors $\mathbf{x}_j, j = \overline{1, J}$ have orders $\bar{W}M_1M_2$. Then, taking into account the expressions for the blocks Q_0, Q_1, Q_2, specified in Lemma 1, the system (4) will be written as

$$\mathbf{x}_0A_0 + \mathbf{x}_1(I_{\bar{W}M_1} \otimes \mathbf{S}_0^{(2)}) = \mathbf{0}$$

$$\mathbf{x}_{j-1}(D_1 \otimes I_{M_1} \otimes \boldsymbol{\beta}_2) + \mathbf{x}_jA + \mathbf{x}_{j+1}(I_{\bar{W}M_1} \otimes \mathbf{S}_0^{(2)}\boldsymbol{\beta}_2) = \mathbf{0}, j = \overline{1, J-1}, \tag{5}$$

$$\mathbf{x}_{J-1}(D_1 \otimes I_{M_1} \otimes \boldsymbol{\beta}_2) + \mathbf{x}_J(A + D_1 \otimes I_{M_1M_2}) = \mathbf{0},$$

where

$$A_0 = I_{\bar{W}} \otimes (S_1 + \mathbf{S}_0^{(1)}\boldsymbol{\beta}_1) + D_0 \otimes I_{M_1},$$

$$A = I_{\bar{W}} \otimes (S_1 + \mathbf{S}_0^{(1)}\boldsymbol{\beta}_1) \otimes I_{M_2} + D_0 \otimes I_{M_1M_2} + I_{\bar{W}M_1} \otimes S_2.$$

Let us now represent the vectors $\mathbf{x}_j$ in the form $\mathbf{x}_0 = \boldsymbol{\theta} \otimes \mathbf{y}_0, \mathbf{x}_j = \boldsymbol{\theta} \otimes \mathbf{y}_j \otimes \boldsymbol{\delta}_2, j = \overline{1, J}$, where $\boldsymbol{\theta}$ is the stationary vector of MAP, $\boldsymbol{\delta}_2$ is a stationary vector PH_2, i.e., is the only solution to the system $\boldsymbol{\delta}_2(S_2 + \mathbf{S}_0^{(2)}\boldsymbol{\beta}_2) = \mathbf{0}, \boldsymbol{\delta}_2\mathbf{e} = 1$, and vectors $\mathbf{y}_j$ of order M_1 satisfy the normalization equation $\sum\limits_{j=0}^{J} \mathbf{y}_j\mathbf{e} = 1$. We use these expressions in the equations of system (5), having previously multiplied the first equation by $\mathbf{e}_{\bar{W}} \otimes I_{M_1}$, and each of the subsequent equations on $\mathbf{e}_{\bar{W}} \otimes I_{M_1} \otimes \mathbf{e}_{M_2}$.

Taking into account the relations $\boldsymbol{\theta}\mathbf{e} = 1$, $-\boldsymbol{\theta}D_0\mathbf{e} = \boldsymbol{\theta}D_1\mathbf{e} = \lambda$, $\mu_2 = \boldsymbol{\delta}_2\mathbf{S}_0^{(2)}$, we reduce system (5) to the following system for vectors $\mathbf{y}_j$:

$$\mathbf{y}_0[-\lambda I_{M_1} + S_1 + \mathbf{S}_0^{(1)}\boldsymbol{\beta}_1] + \mathbf{y}_1\mu_2 = \mathbf{0}$$

$$\mathbf{y}_{j-1}\lambda I_{M_1} + \mathbf{y}_j[(S_1 + \mathbf{S}_0^{(1)}\boldsymbol{\beta}_1) - \lambda I_{M_1} - \mu_2 I_{M_1}] + \mathbf{y}_{j+1}\mu_2 = \mathbf{0}, j = \overline{1, J-1}, \tag{6}$$

$$\mathbf{y}_{J-1}\lambda + \mathbf{y}_J[(S_1 + \mathbf{S}_0^{(1)}\boldsymbol{\beta}_1) - \mu_2 I_{M_1}] = \mathbf{0}.$$

Next, we multiply the equations of system (6) on the right by $\mathbf{e}_{M_1}$. As a result, we are convinced that the probabilities $\mathbf{y}_j\mathbf{e}, j = \overline{0, J}$, satisfy the system of equilibrium equations for the process of death and reproduction with death rates μ_2 and reproduction rates λ. Having added the normalization equation, we come to the conclusion that the probability $\mathbf{y}_J\mathbf{e}$ present on the left side of (8) has the form (2).

Now, consider inequality (3), which specifies the ergodicity condition. Our goal in this inequality is to move from vectors $\mathbf{x}_j$ to vectors $\mathbf{y}_j$, whose order is $\bar{W}M_2$ times smaller, and to simplify this inequality as much as possible.

We use the representation $\mathbf{x}_j = \boldsymbol{\theta} \otimes \mathbf{y}_j \otimes \boldsymbol{\delta}_2$ and substitute the vectors $\mathbf{x}_j$, in this form, into the inequality (3). Then, taking into account the form of the blocks Q_0, Q_2 of the generator specified in Lemma 1, we reduce inequality (3) to the form

$$\sum_{j=0}^{J-1} \boldsymbol{\theta} D_1 \mathbf{e} \mathbf{y}_j \mathbf{e} < \sum_{j=0}^{J} \mathbf{y}_j \mathbf{S}_0^{(1)}. \tag{7}$$

Taking into account the relations $\boldsymbol{\theta} D_1 \mathbf{e} = \lambda$, $\sum\limits_{j=0}^{J-1} \mathbf{y}_j\mathbf{e} = 1 - \mathbf{y}_J\mathbf{e}$, $\sum\limits_{j=0}^{J} \mathbf{y}_j = \boldsymbol{\delta}_1$, $\mu_1 = \boldsymbol{\delta}_1 \mathbf{S}_0^{(1)}$, let us transform inequality (7) into the form

$$\lambda(1 - \mathbf{y}_J\mathbf{e}) < \mu_1. \tag{8}$$

Then, taking into account the notation $y_J = \mathbf{y}_J\mathbf{e}$, inequality (8) is reduced to form (1). □

Remark 1. *The left-hand side of inequality (1) is a rate of the flow of customers accepted into the system, and the right-hand side is a rate of the output flow from the first server (serving an infinite buffer) under overload conditions. It is clear that, for the existence of a stationary regime in the system, it is necessary and sufficient that the first of the mentioned rates is less than the second one. Inequality (1) can be rewritten in terms of the system load factor, ρ,*

$$\rho = \frac{\lambda(1 - y_J)}{\mu_1} < 1. \tag{9}$$

3.3. Stationary Distribution and Performance Measures

Denote by $\mathbf{p}_i, i \geq 0$ the vectors of the stationary probabilities of the Markov chain $\xi_t, t \geq 0$, corresponding to the state i of the first component. To find these vectors, we use a special algorithm for calculating the stationary distribution of a quasi-birth-and-death process [32,33].

Formulas for calculating stationary probability vectors are as follows (Algortihm 1):

Algorithm 1 Formulas for calculating stationary probability vectors.

$$\mathbf{p}_i = \mathbf{p}_1 \mathcal{R}^{i-1}, \ i \geq 1,$$

where the matrix $\mathcal{R}$ is the minimal non-negative solution of the matrix equation

$$\mathcal{R}^2 Q_0 + \mathcal{R} Q_1 + Q_2 = O,$$

and vectors $\mathbf{p}_0$ and $\mathbf{p}_1$ are the only solutions to the following system of linear algebraic equations

$$\mathbf{p}_0 Q_{0,0} + \mathbf{p}_1 Q_{1,0} = \mathbf{0},$$

$$\mathbf{p}_0 Q_{0,1} + \mathbf{p}_1 (Q_1 + \mathcal{R} Q_0) = \mathbf{0},$$

$$\mathbf{p}_0 \mathbf{e} + \mathbf{p}_1 (I - \mathcal{R})^{-1} \mathbf{e} = 1.$$

Having calculated the stationary distribution $\mathbf{p}_i$, $i \geq 0$, we can calculate a number of performance measures of the system under consideration.

- Joint distribution of the number of tasks in the subsystems G_1 and G_2

$$p_{0,0} = \mathbf{p}_0 \begin{pmatrix} \mathbf{e}_{\bar{W}} \\ \mathbf{0}^T_{J\bar{W}M_2} \end{pmatrix}. \quad p_{0,j} = \mathbf{p}_0 \begin{pmatrix} \mathbf{0}^T_{\bar{W}[1+(j-1)M_2]} \\ \mathbf{e}_{\bar{W}M_2} \\ \mathbf{0}^T_{\bar{W}(J-j)M_2} \end{pmatrix}, i = 0, j = \overline{1, J}.$$

$$p_{i,0} = \mathbf{p}_i \begin{pmatrix} \mathbf{e}_{\bar{W}M_1} \\ \mathbf{0}^T_{J\bar{W}M_1M_2} \end{pmatrix}, i > 0. \quad p_{i,j} = \mathbf{p}_i \begin{pmatrix} \mathbf{0}^T_{\bar{W}M_1[1+(j-1)M_2]} \\ \mathbf{e}_{\bar{W}M_1M_2} \\ \mathbf{0}^T_{\bar{W}(J-j)M_1M_2} \end{pmatrix}, i > 0, j = \overline{1, J}.$$

- Stationary distribution of the number of tasks in the subsystem G_1

$$p_i = \mathbf{p}_i \mathbf{e}, i \geq 0.$$

- Average number of tasks in the subsystem G_1

$$L_1 = \sum_{i=1}^{\infty} i p_i.$$

- Variance of the number of tasks in the subsystem G_1

$$D_1 = \sum_{i=1}^{\infty} i^2 p_i - L^2.$$

- Stationary distribution of the number of tasks in the subsystem G_2

$$q_j = \sum_{i=0}^{\infty} p_{i,j}.$$

- Average number of customers in the subsystem G_2.

$$L_2 = \sum_{j=1}^{J} j q_j.$$

- Row vector of joint probabilities that the subsystem G_1 is empty and the MAP is in the state $\nu, \nu = \overline{0, W}$,

$$\boldsymbol{\kappa}_0 = \mathbf{p}_0 \begin{pmatrix} I_{\bar{W}} \\ I_{\bar{W}} \otimes \mathbf{e}_{JM_2} \end{pmatrix}.$$

- Row vector of joint probabilities that there are $j > 0$ tasks in the subsystem G_1, the MAP is in the state ν, and the service process on server 1 is in the state m_1, $\nu = \overline{0, W}, m_1 = \overline{1, M_1}$,

$$\boldsymbol{\kappa}_i = \mathbf{p}_i \begin{pmatrix} I_{\bar{W}M_1} \\ I_{\bar{W}M_1} \otimes \mathbf{e}_{JM_2} \end{pmatrix}, i > 0.$$

- Row vector of joint probabilities that the usbsystem G_2 is empty and the MAP is in the state $\nu, \nu = \overline{0, W}$

$$\gamma_0 = \mathbf{p}_0 \begin{pmatrix} I_{\bar{W}} \\ O_{\bar{W}JM_2 \times \bar{W}} \end{pmatrix} + \sum_{i=1}^{\infty} \mathbf{p}_i \begin{pmatrix} I_{\bar{W}} \otimes \mathbf{e}_{M_1} \\ O_{\bar{W}JM_1M_2 \times \bar{W}} \end{pmatrix}.$$

- Row vector of joint probabilities that there are $j > 0$ tasks in the subsystem G_2, the MAP is in the state ν, and the service process on server 2 is in the state m_2, $\nu = \overline{0, W}, m_2 = \overline{1, M_2}$,

$$\gamma_j = \mathbf{p}_0 \begin{pmatrix} O_{\bar{W} \times \bar{W} M_2} \\ O_{\bar{W}(j-1)M_2 \times \bar{W} M_2} \\ I_{\bar{W} M_2} \\ O_{\bar{W}(J-j)M_2 \times \bar{W} M_2} \end{pmatrix} + \sum_{i=1}^{\infty} \mathbf{p}_i \begin{pmatrix} O_{\bar{W} M_1 \times \bar{W} M_2} \\ O_{\bar{W}(j-1)M_1 M_2 \times \bar{W} M_2} \\ I_{\bar{W}} \otimes \mathbf{e}_{M_1} \otimes I_{M_2} \\ O_{\bar{W}(J-j)M_1 M_2 \times \bar{W} M_2} \end{pmatrix}, j = \overline{1, J}.$$

- Loss probability

$$P_{loss} = \frac{1}{\lambda} \gamma_J (D_1 \otimes I_{M_2}) \mathbf{e}.$$

3.4. Sojourn Time of Tasks in the Subsystems G_1 and G_2

Denote by $V_k(t)$ the stationary distribution function of the sojourn time of a task accepted into the usbsystem G_k and by $v_k(u) = \int\limits_{t=0}^{\infty} e^{-ut} dV_k(t), Re\, u \geq 0$, the Laplace–Stieltjes transform of this distribution function, $k = 1, 2$. Using the probabilistic interpretation of the Laplace–Stieltjes transform and the total probability formula, we can write the following expressions for the functions $v_1(u)$ and $v_2(u)$.

$$v_1(u) = \frac{1}{\lambda} \left[\boldsymbol{\kappa}_0 D_1 \mathbf{e} \chi_1(u) + \sum_{i=1}^{\infty} \boldsymbol{\kappa}_i (D_1 \mathbf{e} \otimes I_{M_1})(uI - S_1)^{-1} \mathbf{S}_0^{(1)} [\chi_1(u)]^i \right], Re\, u \geq 0.$$

$$v_2(u) = \frac{1}{\lambda} \left[\gamma_0 D_1 \mathbf{e} \chi_2(u) + \sum_{j=1}^{J-1} \gamma_j (D_1 \mathbf{e} \otimes I_{M_2})(uI - S_2)^{-1} \mathbf{S}_0^{(2)} [\chi_2(u)]^j \right], Re\, u \geq 0.$$

The average value $\bar{v}_k$ of the sojourn time of a customer accepted into the subsystem G_k is calculated by the well-known formula $\bar{v}_k = -\frac{v_1(u)}{du}|_{u=0}$. After differentiation, we obtain the following expressions for $\bar{v}_k, k = 1, 2:$

$$\bar{v}_1 = \frac{1}{\lambda} \left\{ [\boldsymbol{\kappa}_0 + \sum_{i=1}^{\infty} i \boldsymbol{\kappa}_i (I_{\bar{W}} \otimes \mathbf{e}_{M_1})] D_1 \mathbf{e} b_1 - \sum_{i=1}^{\infty} \boldsymbol{\kappa}_i (D_1 \otimes S_1^{-1}) \mathbf{e} \right\},$$

$$\bar{v}_2 = \frac{1}{\lambda} \left\{ [\gamma_0 + \sum_{j=1}^{J-1} j \gamma_j (I_{\bar{W}} \otimes \mathbf{e}_{M_2})] D_1 \mathbf{e} b_2 - \sum_{j=1}^{J-1} \gamma_j (D_1 \otimes S_2^{-1}) \mathbf{e} \right\}.$$

Also note that any initial moment of the sojourn time of order $l > 0$ for the subsystem G_k can be calculated by the formula

$$\bar{v}_k^{(l)} = (-1)^l \frac{dv_k^l(u)}{du^l}|_{u=0}, k = 1, 2, l > 0.$$

3.5. Sojourn Time of a Customer in the System

In this section, we consider the average sojourn time of a customer in the system from the moment it arrives to the moment when tasks corresponding to the same customer are joined before departing the system. Denote this average as $\bar{V}$. Below, we derive the lower and upper bounds for $\bar{V}$. In doing so, we take into account the following considerations.

(1). The average $\bar{V}$ is not less than the average value of the maximum service times, $\bar{V}_{max,service}(G_1, G_2)$, of tasks belonging to the same customer.

(2). The average $\bar{V}$ is not greater than the average value of the maximum sojourn times of tasks belonging to the same customer, $\bar{V}_{max,sojourn}(G_1, G_2)$, if we assume that the subsystems G_1 and G_2 are independent.

(3). The average sojourn time in the subsystem G_2, $\bar{V}(G_2)$, is not greater than the average sojourn time in the similar subsystem $\hat{G}_2$ having an infinite buffer, $\bar{V}(\hat{G}_2)$.

(4). The average sojourn time in the subsystem G_1 where the original MAP is thinned out due to the finite buffer in the subsystem G_2, $\bar{V}(G_1)$ is no more than the average sojourn time in the similar subsystem $\hat{G}_1$ with an infinite buffer and the original MAP.

From point 1, it follows that

$$\bar{V} \geq \bar{V}_{max,service}(G_1, G_2). \tag{10}$$

Let us calculate the right-hand side of inequality (10).

Let ζ_k be a random variable equal to the service time on the first server, $k = 1, 2$. Then, the average $\bar{V}_{max,service}(G_1, G_2)$ is equal to the average value of the random variable $\zeta = max\{\zeta_1, \zeta_2\}$. Taking into account that the service times on the servers of subsystems G_1 and G_2 are independent random variables, it is easy to see that the distribution function $\Phi(t)$ of the random variable ζ is found as $\Phi(t) = Pr(\zeta < t) = Pr(\zeta_1 < t, \zeta_2 < t) = (1 - \boldsymbol{\beta}_1 e^{S_1 t}\mathbf{e})(1 - \boldsymbol{\beta}_2 e^{S_2 t}\mathbf{e})$. Let us now calculate the mean value of the random variable ζ.

$$\bar{V}_{max,service}(G_1, G_2) = \int_0^\infty t d\Phi(t) =$$

$$\int_0^\infty t d[(1 - \boldsymbol{\beta}_1 e^{S_1 t}\mathbf{e})(1 - \boldsymbol{\beta}_2 e^{S_2 t}\mathbf{e})] =$$

$$= -\int_t^\infty t[\boldsymbol{\beta}_1 S_1 e^{S_1 t}\mathbf{e}(1 - \boldsymbol{\beta}_2 e^{S_2 t}\mathbf{e}) + (1 - \boldsymbol{\beta}_1 e^{S_1 t}\mathbf{e})\boldsymbol{\beta}_2 S_2 e^{S_2 t}\mathbf{e}] =$$

$$= -\int_0^\infty t\boldsymbol{\beta}_1 S_1 e^{S_1 t}\mathbf{e} + \int_0^\infty t\boldsymbol{\beta}_1 S_1 e^{S_1 t}\mathbf{e}\boldsymbol{\beta}_2 e^{S_2 t}\mathbf{e}) - \int_0^\infty t\boldsymbol{\beta}_2 S_2 e^{S_2 t}\mathbf{e} + \int_0^\infty t\boldsymbol{\beta}_1 e^{S_1 t}\mathbf{e}\boldsymbol{\beta}_2 S_2 e^{S_2 t}\mathbf{e} =$$

$$= -\int_0^\infty t\boldsymbol{\beta}_1 S_1 e^{S_1 t}\mathbf{e} + \int_0^\infty t\boldsymbol{\beta}_1 S_1 e^{S_1 t}\mathbf{e}\boldsymbol{\beta}_2 e^{S_2 t}\mathbf{e}) - \int_0^\infty t\boldsymbol{\beta}_2 S_2 e^{S_2 t}\mathbf{e} + \int_0^\infty t\boldsymbol{\beta}_1 e^{S_1 t}\mathbf{e}\boldsymbol{\beta}_2 S_2 e^{S_2 t}\mathbf{e}) =$$

$$\Big\{\int_0^\infty t\boldsymbol{\beta}_1 S_1 e^{S_1 t}\mathbf{e}\boldsymbol{\beta}_2 e^{S_2 t}\mathbf{e}) = \int_0^\infty t\boldsymbol{\beta}_1 S_1 e^{S_1 t}\mathbf{e} \otimes \boldsymbol{\beta}_2 e^{S_2 t}\mathbf{e}) = (\boldsymbol{\beta}_1 \otimes \boldsymbol{\beta}_2)(S_1 \otimes I_{M_2})\int_0^\infty t e^{(S_1 \oplus S_2)t} dt\mathbf{e} =$$

$$= (\boldsymbol{\beta}_1 \otimes \boldsymbol{\beta}_1)(S_1 \otimes I_{M_2})(S_1 \oplus S_2)^{-2}\mathbf{e} = (\boldsymbol{\beta}_1 S_1 \otimes \boldsymbol{\beta}_2)(S_1 \oplus S_2)^{-2}\mathbf{e}\Big\} =$$

$$= [\boldsymbol{\beta}_1(-S_1)^{-1} + \boldsymbol{\beta}_2(-S_2)^{-1} + (\boldsymbol{\beta}_1 S_1 \otimes \boldsymbol{\beta}_2)(S_1 \oplus S_2)^{-2} + (\boldsymbol{\beta}_1 \otimes \boldsymbol{\beta}_2 S_2)(S_1 \oplus S_2)^{-2}]\mathbf{e}. \tag{11}$$

Furthermore, from points 2–4, the following inequalities follow

$$\bar{V} \leq \bar{V}_{max,sojourn}(G_1, G_2) \leq \bar{V}_{max,sojourn}(\hat{G}_1, \hat{G}_2). \tag{12}$$

The systems $\hat{G}_1, \hat{G}_2$ introduced above are $MAP/PH/1$ systems that differ only in the parameters of the service time distribution. The distribution of the service time in the system $\hat{G}_k$ is described by an irreducible representation $(\boldsymbol{\beta}_k, S_k), k = 1, 2$. Let us find an upper bound, $\bar{V}_{max,sojourn}(\hat{G}_1, \hat{G}_2)$, of the desired average, $\bar{V}$, as the average of the maximum of sojourn times in the systems $\hat{G}_1$ and $\hat{G}_2$. It is known, as can be seen in [33], that the sojourn time in the system $MAP/PH/1$ has a PH distribution. Denote the irreducible representation of this distribution for the system $\hat{G}_k$ as $(\tau_k, T_k), k = 1, 2$. To calculate the upper bound $\bar{V}_{max,sojourn}(\hat{G}_1, \hat{G}_2)$, we will use the proof similar to that for formula (12). As a result, we obtain the following relations:

$$\bar{V}_{max,sojourn}(\hat{G}_1, \hat{G}_2) = \int_0^\infty t d(1 - \boldsymbol{\tau}_1 e^{T_1 t}\mathbf{e})(1 - \boldsymbol{\tau}_2 e^{T_2 t}\mathbf{e}) =$$

$$= [\boldsymbol{\tau}_1(-T_1)^{-1} + \boldsymbol{\tau}_2(-T_2)^{-1} + (\boldsymbol{\tau}_1 T_1 \otimes \boldsymbol{\tau}_2)(T_1 \oplus T_2)^{-2} + (\boldsymbol{\tau}_1 \otimes \boldsymbol{\tau}_2 T_2)(T_1 \oplus T_2)^{-2}]\mathbf{e}. \quad (13)$$

Next, it is necessary to obtain expressions for the parameters $\tau_k, T_k, k = 1,2$, of the sojourn time distributions in the systems $\hat{G}_1$ and $\hat{G}_2$. These expressions were obtained in [33] and will be given below for the convenience of the reader. Since all further results on finding the sojourn time are valid for any system $MAP/PH/1$, including the systems $\hat{G}_1$ and $\hat{G}_2$, in order to avoid cumbersome expressions, we will omit the index k in the notation $\boldsymbol{\beta}_k, S_k, \tau_k, T_k$. That is, we will consider the distribution of the sojourn time in the $MAP/PH/1$ system with MAP given by the matrices D_0, D_1 and PH, and the service time distribution given by the Mth-order representation $(\boldsymbol{\beta}, S)$.

According to [33], when deriving expressions for the parameters τ, T, the blocks of the Markov chain generator describing the operation of the system $MAP/PH/1$ are used. This generator looks like

$$A = \begin{pmatrix} A_{0,0} & A_2 & O & O & \dots \\ A_0 & A_1 & A_2 & O & \dots \\ O & A_0 & A_1 & A_2 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

where

$$A_{0,0} = D_0 \otimes I_M,\ A_0 = D_1 \otimes I_M,\ A_1 = D_0 \oplus S,\ A_2 = I_{\bar{W}} \otimes (\mathbf{S}_0\boldsymbol{\beta}).$$

Denote by $\boldsymbol{\pi}_i, i \geq 0$, the vectors of the stationary distribution of the system states at an arbitrary time. The vectors $\boldsymbol{\pi}_i, i \geq 0$, are calculated using an algorithm similar to Algorithm 1 with minimal changes. For the convenience of the reader, we present this algorithm below (Algorithm 2).

Algorithm 2 The vectors of the stationary state probabilities of the $MAP/PH/1$ system.

$$\boldsymbol{\pi}_i = \boldsymbol{\pi}_0 R^i,\ i \geq 0,$$

where the matrix R is the minimal non-negative solution of the matrix equation

$$R^2 A_0 + RA_1 + A_2 = O,$$

and the vector $\boldsymbol{\pi}_0$ is the unique solution of the following system of linear algebraic equations

$$\boldsymbol{\pi}_0(A_{0,0} + RA_0) = \mathbf{0},\ \boldsymbol{\pi}_0(I - R)^{-1}\mathbf{e} = 1.$$

Next, we use the results of [33] for the sojourn time in the queueing system, the operation of which is described by the general QBD process. According to these results, the indicated sojourn time has a PH distribution, the parameters of which are defined in [33]. In our case, we modify these results to the case of the $MAP/PH/1$ system, which is a special case of the system considered in [33]. Taking into account this difference, we derive the following statement.

Statement 1. *The sojourn time in the system $MAP/PH/1$ under consideration has a PH distribution with an irreducible representation $(\boldsymbol{\tau}, T)$, where the row vector $\boldsymbol{\tau}$ and the square matrix T have orders $(\bar{W}M)^2$ and are calculated by the following formulas:*

$$\boldsymbol{\tau} = \boldsymbol{\zeta}^T(I_{\bar{W}M} \otimes \hat{\Delta}), \quad T = (A_1 + A_2) \otimes I_{\bar{W}M} + A_0 \otimes \hat{\Delta}^{-1}\hat{R}^T\hat{\Delta}, \quad (14)$$

where

$$\boldsymbol{\zeta} = \begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \\ \vdots \\ \zeta_{\bar{W}M} \end{pmatrix}, \tag{15}$$

$\boldsymbol{\zeta}_l$ is a column vector of order $\bar{W}M$ whose lth entry is equal to one and the remaining entries are equal to zero, $\hat{\Delta}$ -diagonal matrix of order $\bar{W}M$, whose diagonal entries are equal to the corresponding entries of the vector

$$\hat{\boldsymbol{\eta}} = \{\boldsymbol{\pi}_0 A_2[I + (A_1 + RA_0)^{-1}A_2]^{-1}\mathbf{e}\}^{-1}\boldsymbol{\pi}_0 A_2(I - \hat{R})^{-1}, \tag{16}$$

where the matrix $\hat{R}$ is calculated as

$$\hat{R} = -(A_1 + RA_0)^{-1}A_2. \tag{17}$$

Corollary 2. *The upper bound $\bar{V}_{max,sojourn}(\hat{G}_1, \hat{G}_2)$ is calculated by formula (13), where the vectors $\boldsymbol{\tau}_k$ and the matrices T_k, $k = 1,2$, are defined by formulas (14)–(17) in which all occurring notations (except $\bar{W}$) are provided with the index k.*

Thus, it follows from formulas (10)–(13) that the lower and upper bounds for mean sojourn time, $\bar{V}$, in the fork–join queue under consideration are defined from the following inequalities:

$$\begin{gathered}[\boldsymbol{\beta}_1(-S_1)^{-1} + \boldsymbol{\beta}_2(-S_2)^{-1} + (\boldsymbol{\beta}_1 S_1 \otimes \boldsymbol{\beta}_2)(S_1 \oplus S_2)^{-2} + (\boldsymbol{\beta}_1 \otimes \boldsymbol{\beta}_2 S_2)(S_1 \oplus S_2)^{-2}]\mathbf{e} \leq \\ \bar{V} \leq \\ [\boldsymbol{\tau}_1(-T_1)^{-1} + \boldsymbol{\tau}_2(-T_2)^{-1} + (\boldsymbol{\tau}_1 T_1 \otimes \boldsymbol{\tau}_2)(T_1 \oplus T_2)^{-2} + (\boldsymbol{\tau}_1 \otimes \boldsymbol{\tau}_2 T_2)(T_1 \oplus T_2)^{-2}]\mathbf{e},\end{gathered} \tag{18}$$

where the vectors $\boldsymbol{\tau}_k$ and the matrices T_k, $k = 1,2$, are defined by Statement 1 and Corollary 2.

4. Study of General Fork–Join Systems

In the previous section, we obtained the analytical solution only when the customer consisted of two tasks. It is a complex problem to obtain a similar solution for the general case $K > 2$ even with the most straightforward assumptions about the incoming flow of customers and the distribution of the task's service time (the Poisson incoming flow and exponential service time). Such a solution and the corresponding algorithms and programs for finding numerical results are necessary for using the fork–join model of systems for practical applications.

The Monte Carlo method takes several seconds to several minutes to calculate the performance characteristics of the fork–join system depending on the input parameters and the required accuracy. This method has proven itself well when solving a direct problem, i.e., when all the system's characteristics are known, and it is necessary to find the performance characteristics of the system. In system design problems in which the inverse problem is solved, to select a system based on initial performance characteristics and find optimal characteristics for it, it is necessary to go through many different options, using, for example, the branch and bound method. Using the Monte Carlo method in this case can take significant time and computational resources. In order to accelerate the production of results and find quick performance estimates, in this article, we suggest a combined approach based on the Monte Carlo method and machine learning (ML) to find the performance characteristics of a general fork–join system.

Figure 2 shows a diagram using the combined method. Within the framework of this scheme, at the first stage, an algorithm and software package is implemented in Python to obtain numerical results for assessing the performance characteristics of the fork–join system ($K = 2$), the description of which is given in Section 3. We used these numerical results to validate the simulation model for a fork–join system of general form ($K \geq 2$) developed at the second stage. We used the open source library Pyqumo when we developed the simulation model (Monte Carlo method). The library is available at the link https://github.com/ipu69/pyqumo (accessed on 8 November 2023). After that, we generated synthetic data for training ML algorithms.

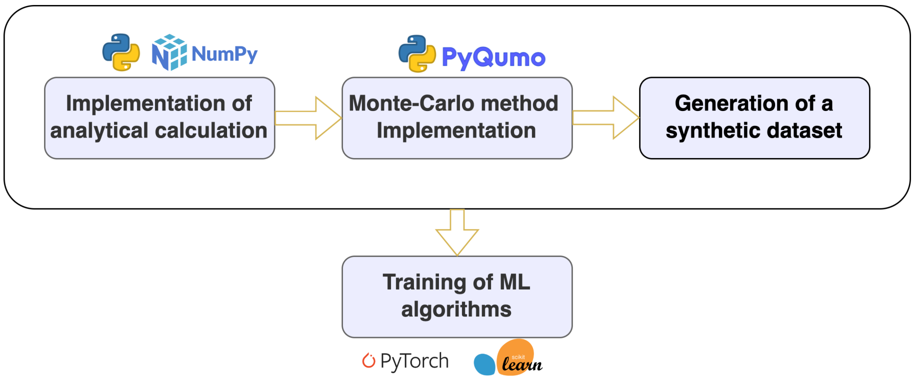
NumPy
PyQumo
Implementation of analytical calculation
Monte-Carlo method Implementation
Generation of a synthetic dataset
Training of ML algorithms
PyTorch
learn

Figure 2. Scheme of a combined method for calculating the performance characteristics of a fork–join system using Monte Carlo and ML methods.

The following section will provide a detailed description of the implementation of the Monte Carlo method and the use of ML algorithms to predict the performance of a general fork–join system.

4.1. Description of Simulation Model of a General Type Fork–Join System

The discrete-event model allows us to obtain accurate results by modeling many events. There are events such as the generation of new customer, the processing of task, and the servicing and joining of tasks. All events are in a priority queue and sorted by the event's time. Thus, time in the model changes in jumps. After the occurrence of an event, the time inside the model will change to the time of the next event.

The selection of the next event and the generation of the subsequent one is carried out until the required accuracy is achieved or the required number of customers is generated. Sometimes, a time limit is set for executing the simulation model. The algorithm for executing the simulation model (the Monte Carlo method) is shown in Figure 3.

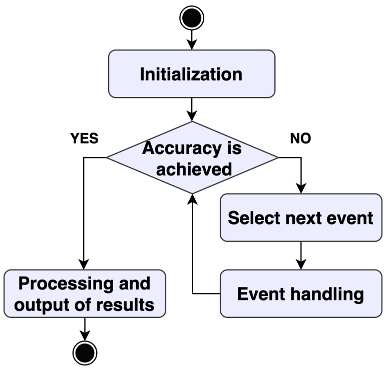

Figure 3. Operation scheme of the simulation model (Monte Carlo method).

The input parameters for the simulation model are as follows:

- MAP, which is specified by the matrices D_0 and D_1;
- An array of PH_k distributions, each of which consists of both the S_k and β_k matrices, the size of array equals to subsystems number;
- An array of buffer sizes R_k for each server, the size of array equals to subsystems number.

To validate the simulation model, we used the implementation of the analytical model of Section 3 and compared the results of their execution.

4.2. Description of the Machine Learning Model

In this section, we describe machine learning methods to predict various characteristics of a fork–join system. We built models to estimate the system response time and the probability of losing a customer.

The main problem when using ML models to predict the performance of a fork–join system is the parameterization of the distributions. As noted in the article, MAP and PH distributions have a matrix representation. In addition, to specify systems, it is necessary to specify the PH distribution for each service. To parameterize MAP, we used the following parameters: intensity (λ_a), coefficient of distribution asymmetry (γ_a), coefficient of variation (cv_a), and distribution lag (l). Next, we used the MAP recovery algorithm described in [34]. A similar algorithm was used to restore PH on each server. To restore the PH distributions, It is necessary to know the following parameters: intensity (λ_s), coefficient of variation (cv_s), and asymmetry coefficient (γ_s). A similar method is used for recovery [34,35]. When parameterizing the K distributions in the model, we added several restrictions:

- All PH distributions have the same cv_s and γ_s;
- The buffer capacity of all subsystems is the same;
- To set the intensity of distributions, two schemes are used: in the first, λ_{smax} and λ_{smin} are specified, and the dependence $\lambda(i)$ is restored using 2 points, where i is the index of server, $\lambda(1) = \lambda_{smin}$, and $\lambda(K) = \lambda_{smax}$, the remaining values $i = 2, \ldots, K-1$ are calculated from the resulting equation. In the second case, λ_{smax} and λ_{smin} are also generated. The share of slow servers is also generated, whose intensity is equal to λ_{smin}, and the remaining servers have an intensity of λ_{smax}, respectively.

Thus, the features when training ML algorithms were as follows:

- λ_a—intensity of the MAP;
- cv_a—coefficient of variation of MAP;
- γ_a—coefficient of asymmetry of MAP;
- l_a—MAP distributions lag ;
- cv_s—coefficient of variation of PH distribution;
- γ_s—coefficient of asymmetry of PH distribution;
- λ_{smin}—minimum intensity of PH distribution;
- λ_{smax}—maximum intensity of PH distribution;
- K—the number of tasks that the customer contains, and the number of subsystems;
- R—buffer capacity for servers;
- N—number of slow servers ($0 \leq N \leq K$).

Figure 4 shows the complete data preparation cycle for training ML models. In the first stage, a dataset is generated; the intervals for each characteristic are given in Table 1. To transfer the parameters to the simulation model, MAP and PH distributions are restored. After restoring the distributions, we calculated characteristics using the Monte Carlo method.

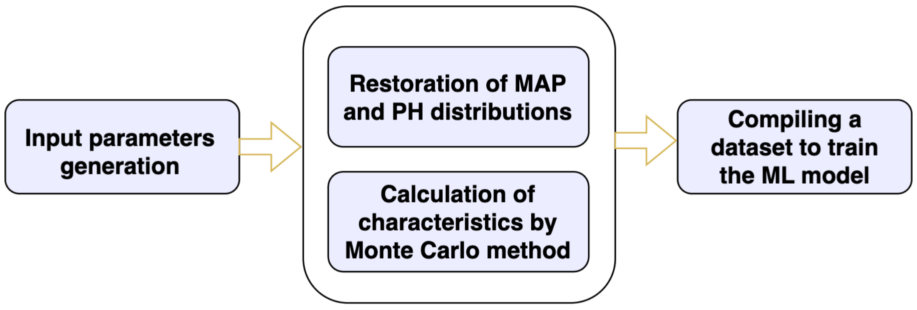
Input parameters generation
Restoration of MAP and PH distributions
Calculation of characteristics by Monte Carlo method
Compiling a dataset to train the ML model

Figure 4. Scheme for generating a dataset for training ML models.

Table 1. Table of limits within which the value of each model parameter was generated.

Parameter	Minimum Value	Maximum Value
λ_a	0.1	50
cv_a	0.1	5
γ_a	0	100
l_a	0.5	1
cv_s	0.1	5
γ_s	1	100
λ_{smin}	0.1	50
λ_{smax}	λ_{smin}	50
K	1	32
R	0	20
N	0	K

5. Results of Numerical Research

This section presents numerical results obtained during the study of fork–join systems. At the beginning, we present the results of the validation of the simulation model using an analytical solution for $K = 2$. We also present the time it takes to find stationary characteristics of the system depending on the parameters: buffer size, order of the input MAP flow, orders of distributions PH_1, and PH_2. The remainder describes the results obtained using the combined method to predict the probability of loss and system sojourn time.

The analytical model was implemented in Python language using the Numpy library. The ML algorithms were trained in Jupyter-Lab. We used scikit-learn and PyTorch libraries. The research results are available at https://github.com/ipu69/queues-fork-join (accessed on 8 November 2023).

5.1. Validation of Monte Carlo Method Implementation

As noted in the previous section, the implementation of the analytical calculation of a particular case of the fork–join system for $K = 2$ was used to validate the implementation of the Monte Carlo method for calculating the characteristics of an arbitrary system. A feature of the calculation of the analytical model is the significant time required to calculate the characteristics.

Figure 5 shows the dependence of the time of calculating characteristics analytically depending on various system parameters: MAP size, PH size, and buffer size. The dependence on the size of MAP and PH is power-law, and on the buffer size, it is linear. This is due to the dependence of the block sizes $Q_{i,j}$ on the matrices $D_{0,1}$, $S_{1,2}$ and the dependence of the number of blocks on the parameter J in the analytical model.

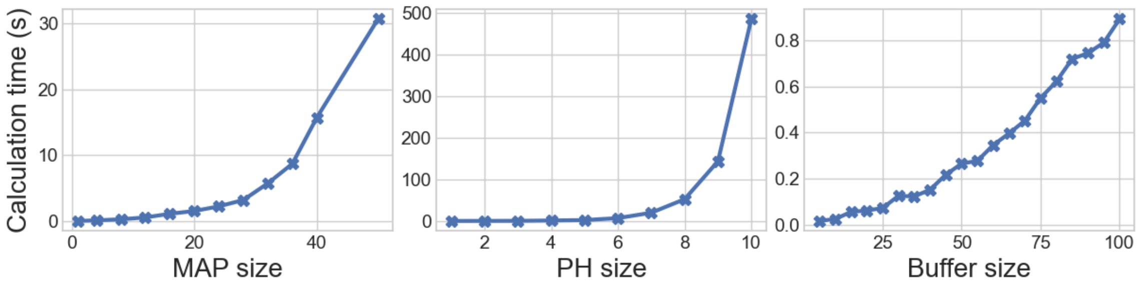

Figure 5. Dependence of the time for calculating the characteristics of a fork–join system in an analytical way depending on various system parameters: MAP size, PH size, and buffer size.

Figure 6 shows the results of comparing the calculations of the characteristics of the fork–join system for the case $K = 2$ using the analytical method and the Monte Carlo method. According to the validation results, the error in calculating characteristics using the Monte Carlo method was about 1%. One of the advantages of calculating characteristics using the Monte Carlo method is that the time for calculation almost does not depend on input parameters. The calculation time does not depend on the size of MAP and PH, but depends only on the number of simulated customers and the accuracy of the calculation. Figure 7 shows the dependence of the execution time of the simulation model on the number of generated customers.

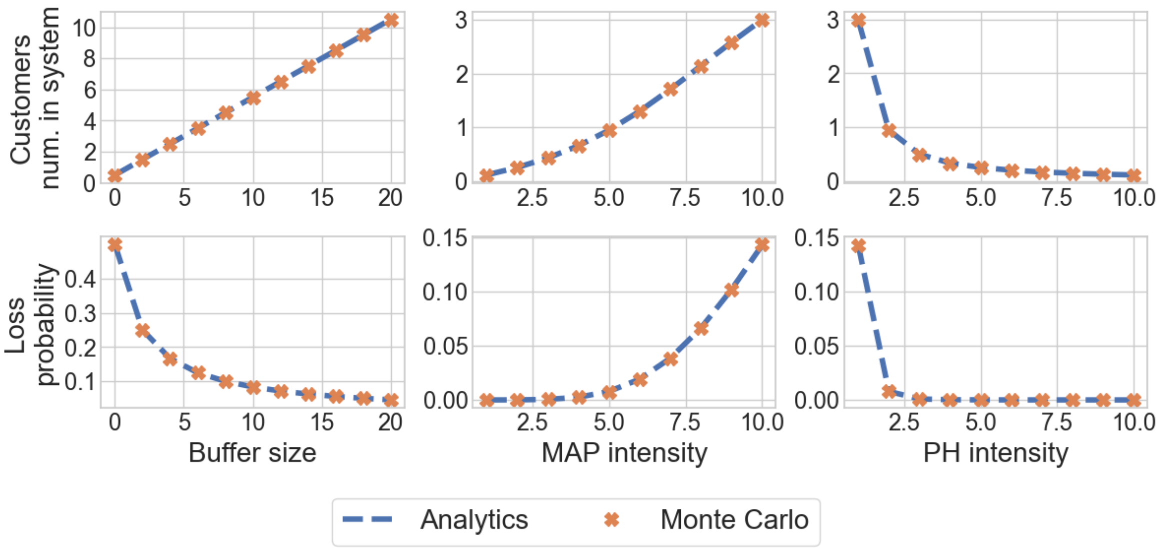

Figure 6. Validation of Monte Carlo calculations using the implementation of analytical calculations in Python.

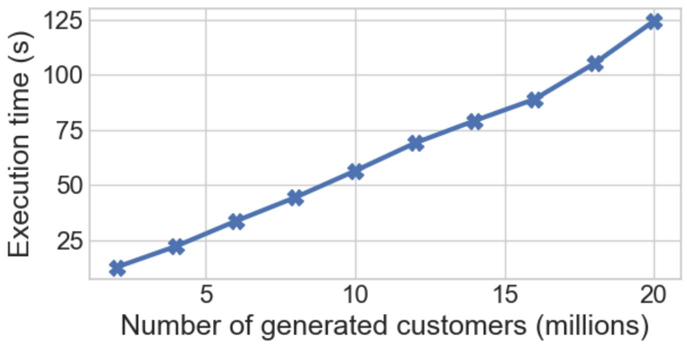

Figure 7. Dependence of the time for calculating the characteristics of the fork–join system on the number of generated customers.

5.2. Predicting Customer Sojourn Time Using ML Algorithms

In this problem, using the fork–join system parameters, we trained classical ML [36,37] algorithms and a neural network to predict the sojourn time of the fork–join system for the case when a linear function approximates the tasks PH intensities at the servers. We generated a synthetic dataset consisting of approximately 150 thousand records. In the experiment, we used classical algorithms on trees—decision trees [36] and gradient boosting [37]—as well as a neural network. During training, the dataset was divided into 80% for training, 15% for testing, and 5% for validation.

Figure 8 shows scatterplots for various methods. The best result in terms of metrics was shown by gradient boosting. The maximum subtree depth in the gradient boosting algorithm was equal to 10, and the number of trees was equal to 1200.

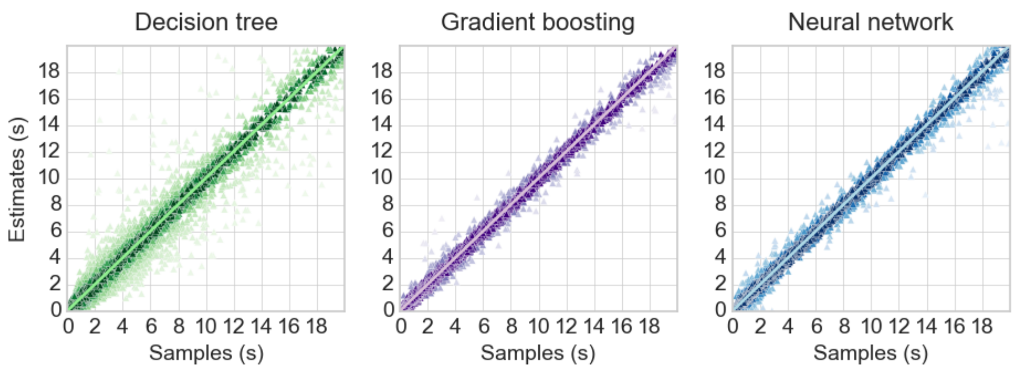

Figure 8. Scatterplots of the system sojourn time prediction for regression algorithms.

The neural network for the regression problem contained one input layer and one hidden layer. The hidden layer consisted of 128 neurons. The input layer used $LeakyReLu$ as the activation function, and the hidden layer used $ReLu$. We used mse as the loss function and Adam's algorithm [38] as the optimization algorithm. Absolute errors for ML algorithms are shown in Figure 9.

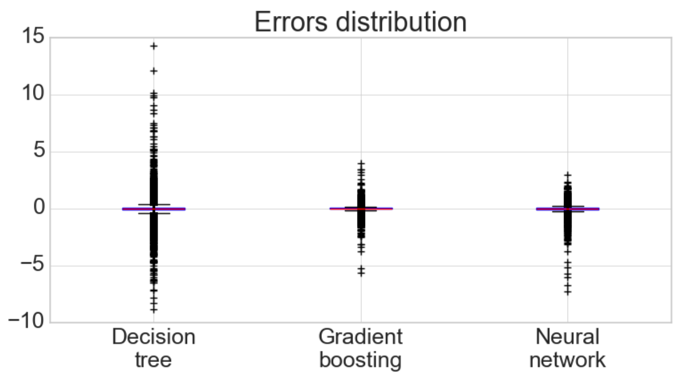

Figure 9. Error chart of ML algorithms when predicting system sojourn time.

The validation used a dataset in which all parameters were fixed and others varied. The values by which the dependence of the system sojourn time was plotted with other fixed parameters: the intensity of the MAP flow, the buffer size, as well as the number of serving devices or the number of requests in the application. The following fixed system parameters were used during validation: $\lambda_a = 5$ $\lambda_{smin} = 20$, $\lambda_{smax} = 20$, $K = 4$, $R = 14$, $cv_a = 4$, $cv_s = 0.9$, $\gamma_a = 20$, $l_a = 0.5$, $\gamma_a = 60$. When validating the intensity MAP, parameter λ_a varied from 5 to 50. When constructing the sojourn time dependence, the buffer size varied from 2 to 18 with a step of 2. Accordingly, when constructing the dependence on the number of servers, which is the same as the number of requests in the customer, parameter N varied from 4 to 14.

Figure 10 shows the results of a comparison of the sojourn time calculations using ML algorithms and using the Monte Carlo method. It is worth noting that the trained algorithms demonstrated relatively high accuracy rates. Table 2 shows a comparison of metrics for algorithms. Gradient boosting and neural network show approximately the same results.

Table 2. Metric values for regression models predicting system sojourn times.

Algorithm	Correlation	R2	Mean Square Error
Decision tree	0.97	0.95	0.27
Gradient boosting	0.99	0.99	0.029
Neural network	0.99	0.99	0.037

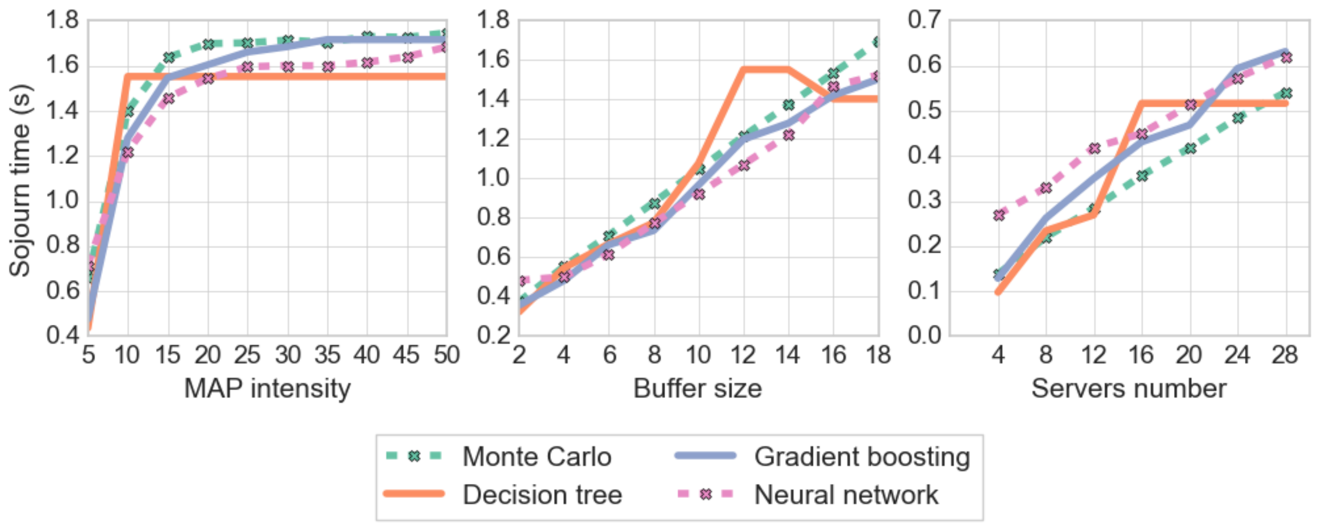

Figure 10. Comparison of prediction results of ML algorithms with results obtained using Monte Carlo method.

Table 3 shows the sojourn time calculation times using the Monte Carlo method and using various ML algorithms. ML algorithms provide a speed of calculation time of approximately 10^4 times, which gives a significant gain when designing natural technical systems. For example, to design a fork–join system, it is necessary to solve the problem using the branch-and-bound method to find optimal parameters. When using the Monte Carlo method to solve problems of this kind, finding the optimal solution can take much time. The use of ML algorithms reduces the time required to find a solution.

Table 3. Comparison of the system sojourn time calculations using different algorithms.

Algorithm	Time, s
Monte Carlo method	10
Decision tree	0.0003
Gradient boost	0.0012
Neural network	0.0017

5.3. Predicting the Loss Probability of a Customer Using ML Methods

During this experiment, the accuracy of predicting the loss probability in a fork–join system was studied in function of the speed of service, and servers were divided into slow and fast. For training, we used a synthetic dataset generated using the Monte Carlo method, consisting of approximately 100 thousand rows. This experiment also used classical tree algorithms and a neural network. As noted in the previous section, one of the features was the number of slow servers.

The scatterplot is shown in Figure 11. As well as for predicting the system sojourn time, the gradient boosting algorithm showed the best results. The optimal hyperparameters for classical algorithms were found using the sklearn library. Thus, for a decision tree, the optimal maximum depth is 12; and the mse function is selected as a criterion. For gradient boosting, the optimal depth equals 8, the number of weak classifiers is 1000, and we used mse as the error function. Metric values for regression models are shown in Table 4.

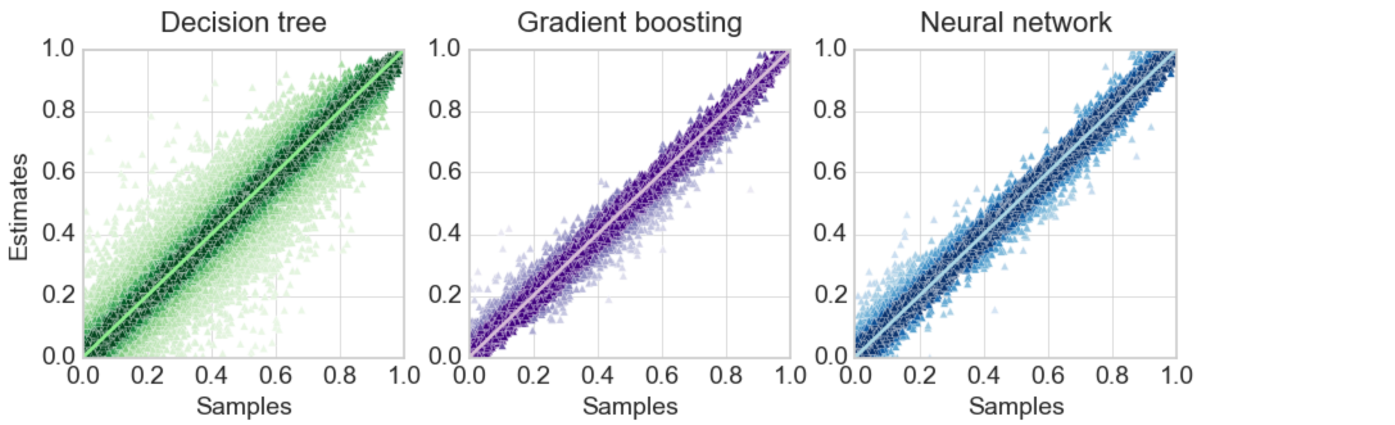

Figure 11. Scatterplots of prediction of customer loss probability for regression algorithms.

Table 4. Metric values for regression models predicting customer loss probability.

Algorithm	Correlation	R2	Mean Square Error
Decision tree	0.96	0.92	0.007
Gradient boost	0.99	0.99	0.0007
Neural network	0.99	0.99	0.0009

Several charts are presented comparing the calculation of characteristics using the Monte Carlo method and the results of predicting the values of ML algorithms. Figure 12 shows the dependence of loss probability on the intensity of the input customers flow, the number of slow servers in the system, and the intensity of slow server servicing. The rest remained constant when examining the dependence of the probability of loss on a specific metric. Absolute errors for ML algorithms are shown in Figure 13.

Thus, when studying the dependence of the loss probability on the MAP intensity, the remaining parameters were equal to the following values: $\lambda_{smin} = 10$, $\lambda_{smax} = 100$, $K = 10$, $N = 5$, $R = 10$, $cv_a = 4$, $cv_s = 0.9$, $\gamma_a = 20$, $l_a = 0.4$, $\gamma_a = 60$. Using similar parameters, we calculated the dependence of the customer loss probability on the number of servers with a low intensity of servicing tasks. When constructing the dependence, the intensity of the MAP was fixed and was equal to $\lambda_a = 50$, and the number of slow servers varied from 0 to 10. In the third experiment, when constructing the dependence of the customer loss probability on the intensity of slow server servicing, the intensity of MAP flow was also fixed at the level of $\lambda_a = 50$, and the number of slow servers also remained constant throughout the experiment and amounted to $N = 5$, that is, half of all servers $K = 10$. The service intensity on the slow server varied from 10 to 100 (the service intensity of a conventional server). As shown in Figure 12, they adequately describe the dependence of the customer loss probability on the parameters. In this experiment, the neural network showed the best accuracy; tree algorithms the demonstrated lower prediction accuracy.

Table 5 compares the time for calculating the loss probability for different methods. Just as when calculating the sojourn time of a system, ML algorithms allow us to obtain results much faster than the Monte Carlo method.

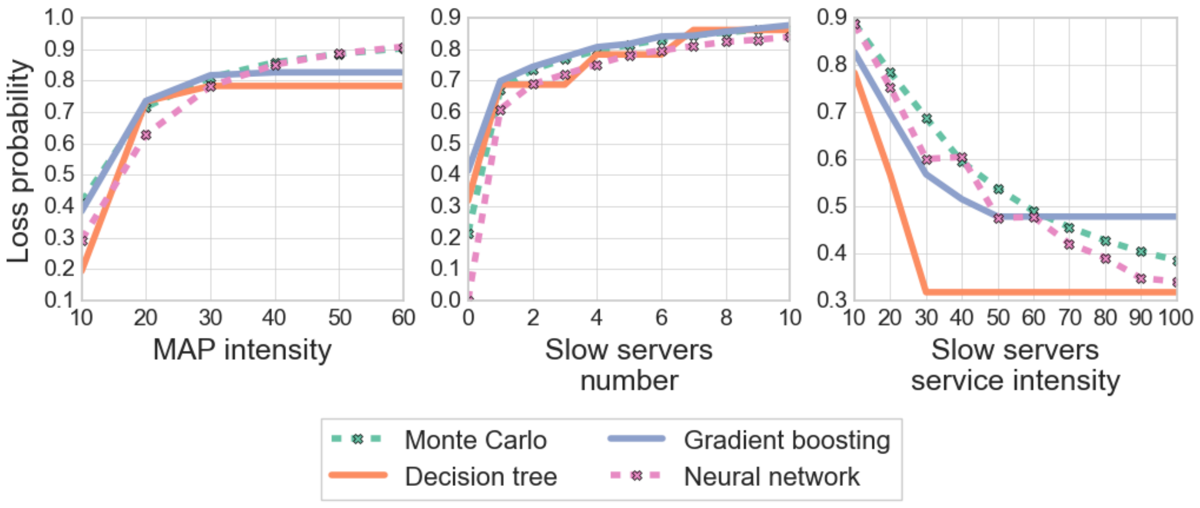

Figure 12. Comparison of prediction results of ML algorithms with results obtained using the Monte Carlo method.

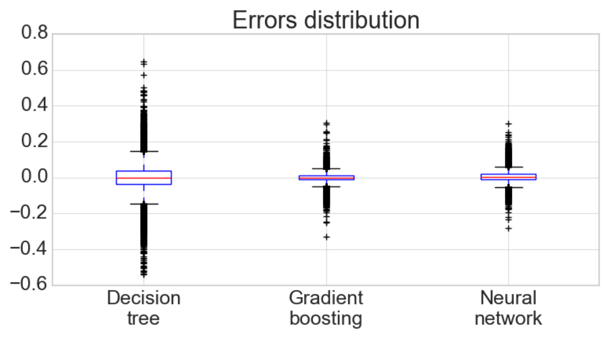

Figure 13. Comparison of the results of predicting the customer loss probability using ML algorithms with the results obtained using the Monte Carlo method.

Table 5. Comparison of the time for calculating the customer loss probability in the system using various algorithms.

Algorithm	Time, s
Monte Carlo method	10
Decision tree	0.00027
Gradient boost	0.0011
Neural network	0.0017

6. Conclusions

In this paper, we investigated a fork–join queuing system with an input MAP flow of customers consisting of an arbitrary number of tasks K into which the customer forks when it enters the system, as well as an arbitrary size of the buffer of the subsystem processing a specific task. Such systems have not been studied in the literature. An analytical solution is presented for the fork–join system when a customer forks into two tasks for processing. We obtained a stationary distribution for this case, formulas for calculating the system's main characteristics, including the average size of the queue, the customer loss probability, and the sojourn customer time in the system.

We suggested a new methodology for quickly estimating the system performance using a combination of Monte Carlo simulation and machine learning to study the general $K \geq 2$ case. Numerical experiments using the suggested method confirmed the high accuracy of estimating the characteristics of a fork–join system (96–98%). Additionally, estimation with a trained machine learning model requires orders of magnitude less time than the Monte Carlo method. Gradient boosting showed the best results among ML algorithms used in this paper. This methodology makes it possible to use the fork–join model in complex optimization problems that arise in the design and implementation of modern systems. Moreover, using this methodology, it is possible to solve the inverse problem. For example, using a known sojourn time, select the optimal number of servers or server performance. This method can be used to solve optimization problems and enhance the performance of the fork–join system.

Author Contributions: Conceptualization, V.M.V. and V.I.K.; methodology, V.M.V., V.I.K. and A.A.L.; software, A.M.S. and A.A.L.; validation, A.M.S., A.A.L. and V.I.K.; formal analysis, V.I.K. and V.M.V.; investigation, V.I.K., V.M.V., A.A.L. and A.M.S.; resources, V.M.V.; data curation, V.I.K.; writing—original draft preparation, V.I.K., A.M.S., A.A.L. and V.M.V.; writing—review and editing, A.M.S., A.A.L. and V.M.V.; visualization, A.M.S. and A.A.L.; supervision, V.M.V. and A.A.L.; project administration, V.M.V. and A.A.L.; funding acquisition, V.M.V. All authors have read and agreed to the published version of the manuscript.

Funding: The reported study was funded by Russian Science Foundation, project number 22-49-02023.

Data Availability Statement: The source code and all research data are available in Github repository: https://github.com/ipu69/queues-fork-join (accessed on 8 November 2023).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MAP	Markovian arrival process
PH	Phase type distribution
ML	Machine learning
ANN	Artificial neural network
MSE	Mean squared error
CV	Coefficient of variation

References

1. Vianna, E.; Comarela, G.; Pontes, T.; Almeida, J.; Almeida, V.; Wilkinson, K.; Kuno, H.; Dayal, U. Analytical performance models for mapreduce workloads. *Int. J. Parallel Program.* **2013**, *41*, 495–525. [CrossRef]
2. Rizk, A.; Poloczek, F.; Ciucu, F. Stochastic bounds in Fork–Join queueing systems under full and partial mapping. *Queueing Syst.* **2016**, *83*, 261–291. [CrossRef]
3. Nguyen, M.; Alesawi, S.; Li, N.; Che, H.; Jiang, H. ForkTail: A black-box fork-join tail latency prediction model for user-facing datacenter workloads. In Proceedings of the HPDC 2018—2018 International Symposium on High-Performance Parallel and Distributed Computing, Tempe, AZ, USA, 11–15 June 2018; pp. 206–217. [CrossRef]
4. Enganti, P.; Rosenkrantz, T.; Sun, L.; Wang, Z.; Che, H.; Jiang, H. ForkMV: Mean-and-Variance Estimation of Fork-Join Queuing Networks for Datacenter Applications. In Proceedings of the 2022 IEEE International Conference on Networking, Architecture and Storage (NAS), Philadelphia, PA, USA, 3–4 October 2022; pp. 1–8. [CrossRef]
5. Flatto, L.; Hahn, S. Two Parallel Queues Created By Arrivals With Two Demands I. *SIAM J. Appl. Math.* **1984**, *44*, 1041–1053. [CrossRef]
6. Nelson, R.D.; Tantawi, A.N. Approximate Analysis of Fork/Join Synchronization in Parallel Queues. *IEEE Trans. Comput.* **1988**, *37*, 739–743. [CrossRef]
7. Kim, C.; Agrawala, A.K. Analysis of the Fork-Join Queue. *IEEE Trans. Comput.* **1989**, *38*, 250–255. [CrossRef]
8. Varma, S.; Makowski, A.M. Interpolation approximations for symmetric Fork-Join queues. *Perform. Eval.* **1994**, *20*, 245–265. [CrossRef]
9. Lui, J.C.; Muntz, R.; Towsley, D. *Computing Performance Bounds for Fork-Join Queueing Models*; University of California: Los Angeles, CA, USA, 2001.
10. Balsamo, S.; Donatiello, L.; Van Dijk, N.M. Bound performance models of heterogeneous parallel processing systems. *IEEE Trans. Parallel Distrib. Syst.* **1998**, *9*, 1041–1056. [CrossRef]
11. Lebrecht, A.S.; Knottenbelt, W.J. Response Time Approximations in Fork-Join Queues. In Proceedings of the 23rd Annual UK Performance Engineering Workshop, Ormskirk, UK, 9–10 July 2007.
12. Thomasian, A. Analysis of fork/join and related queueing systems. *ACM Comput. Surv.* **2014**, *47*. [CrossRef]

13. Jiang, L.; Giachetti, R.E. A queueing network model to analyze the impact of parallelization of care on patient cycle time. *Health Care Manag. Sci.* **2008**, *11*, 248–261. [CrossRef]
14. Armony, M.; Israelit, S.; Mandelbaum, A.; Marmor, Y.N.; Tseytlin, Y.; Yom-Tov, G.B. On patient flow in hospitals: A data-based queueing-science perspective. *Stoch. Syst.* **2015**, *5*, 146–194. [CrossRef]
15. Narahari, Y.; Sundarrajan, P. Performability analysis of fork–join queueing systems. *J. Oper. Res. Soc.* **1995**, *46*, 1237–1249. [CrossRef]
16. Gallien, J.; Wein, L.M. A simple and effective component procurement policy for stochastic assembly systems. *Queueing Syst.* **2001**, *38*, 221–248. [CrossRef]
17. Kemper, B.; Mandjes, M. Mean sojourn times in two-queue fork-join systems: Bounds and approximations. *Spectrum* **2012**, *34*, 723–742. [CrossRef]
18. Schol, D.; Vlasiou, M.; Zwart, B. Large Fork-Join Queues with Nearly Deterministic Arrival and Service Times. *Math. Oper. Res.* **2022**, *47*, 1335–1364. [CrossRef]
19. Qiu, Z.; Pérez, J.F.; Harrison, P.G. Beyond the mean in fork-join queues: Efficient approximation for response-time tails. *Perform. Eval.* **2015**, *91*, 99–116. [CrossRef]
20. Klimenok, V. Performance characteristics of the fork-join queuing system. *Informatics* **2023**, *20*, 50–60. [CrossRef]
21. Marin, A.; Rossi, S. Power control in saturated fork-join queueing systems. *Perform. Eval.* **2017**, *116*, 101–118. [CrossRef]
22. Lee, K.; Shah, N.B.; Huang, L.; Ramchandran, K. The MDS Queue: Analysing the Latency Performance of Erasure Codes. *IEEE Trans. Inf. Theory* **2017**, *63*, 2822–2842. [CrossRef]
23. Wang, W.; Harchol-Balter, M.; Jiang, H.; Scheller-Wolf, A.; Srikant, R. Delay Asymptotics and Bounds for Multi-Task Parallel Jobs. *SIGMETRICS Perform. Eval. Rev.* **2019**, *46*, 2–7. [CrossRef]
24. Nguyen, M.; Alesawi, S.; Li, N.; Che, H.; Jiang, H. A black-box fork-join latency prediction model for data-intensive applications. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 1983–2000. [CrossRef]
25. Morozov, E.; Rumyantsev, A. Stability analysis of a MAP/M/s cluster model by matrix-analytic method. In Proceedings of the Computer Performance Engineering: 13th European Workshop, EPEW 2016, Chios, Greece, 5–7 October 2016; Volume 9951 LNCS, pp. 63–76. [CrossRef]
26. Rumyantsev, A.; Morozova, T.; Basmadjian, R. Discrete-event modeling of a high-performance computing cluster with service rate control. In Proceedings of the Conference of Open Innovation Association, FRUCT, Bologna, Italy, 13–16 November 2018; pp. 224–231. [CrossRef]
27. Vishnevsky, V.; Gorbunova, A.V. Application of Machine Learning Methods to Solving Problems of Queuing Theory. *Commun. Comput. Inf. Sci.* **2022**, *1605 CCIS*, 304–316. [CrossRef]
28. Efrosinin, D.; Vishnevsky, V.; Stepanova, N. Optimal Scheduling in General Multi-Queue System by Combining Simulation and Neural Network Techniques. *Sensors* **2023**, *23*, 5479. [CrossRef]
29. Dieleman, N.; Berkhout, J.; Heidergott, B. A neural network approach to performance analysis of tandem lines: The value of analytical knowledge. *Comput. Oper. Res.* **2023**, *152*, 106124. [CrossRef]
30. Lucantoni, D.M. New results on the single server queue with a batch markovian arrival process. *Commun. Statistics. Stoch. Model.* **1991**, *7*, 1–46. [CrossRef]
31. Dudin, A.N.; Klimenok, V.I.; Vishnevsky, V.M. *The Theory of Queuing Systems with Correlated Flows;* Springer: Berlin, Germany, 2019; pp. 1–410. [CrossRef]
32. Neuts, M.F. *Matrix-Geometric Solutions in Stochastic Models*; The Johns Hopkins University Press: Baltimore, MD, USA, 1981; p. 348.
33. Ozawa, T. Sojourn time distributions in the queue defined by a general QBD process. *Queueing Syst.* **2006**, *53*, 203–211. [CrossRef]
34. Horváth, G. Efficient analysis of the queue length moments of the MMAP/MAP/1 preemptive priority queue. *Perform. Eval.* **2012**, *69*, 684–700. [CrossRef]
35. Vishnevsky, V.; Larionov, A.; Ivanov, R.; Semenova, O. Estimation of IEEE 802.11 DCF access performance in wireless networks with linear topology using PH service time approximations and MAP input. In Proceedings of the 2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT), Moscow, Russia, 20–22 September 2017; pp. 1–5. [CrossRef]
36. Gordon, A.D.; Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. Classification and Regression Trees. *Biometrics* **1984**, *40*, 874. [CrossRef]
37. Friedman, J.H. Stochastic gradient boosting. *Comput. Stat. Data Anal.* **2002**, *38*, 367–378. [CrossRef]
38. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.