Adv Syst Sci Appl 2022; **03**:70–83
Published online at https://ijassa.ipu.ru.

The Analysis of Big Data Centers Performance

Anastasia V. Gorbunova[1*], Vladimir M. Vishnevsky[1]

[1] *V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russia*

Abstract: The article proposes the approach for predicting the performance of big data processing center services. To simulate the process of parallel computing, the fork-join queuing system (QS) with a truncated Pareto distribution of service time and three variants of distributions for the incoming flow is used. The number of devices in each of the fork-join subsystems of the QS can be more than one. The performance scores are the average system response time and its tail delay (99th percentile of the response time distribution). The approach is based on a combination of simulation modeling and neural networks, as one of the methods of machine learning. For at least 93% of the data used to test the proposed method in the course of a numerical experiment the approximation error of both studied characteristics does not exceed 5%, while the maximum approximation error is not more than 10%.

Keywords: parallel computing, data-intensive applications, big data, fork-join queueing system, average response time, artificial neural networks, machine learning methods

1. INTRODUCTION

According to an IDC (International Data Corporation) report [19], digital data continues to grow relentlessly. Thus, according to forecasts, by 2025 the Global Datasphere will increase to 175 zettabytes [19]. This is more than five times the amount of digital data available as of 2018.

In the field of scientific research, this trend has led to the formation of the so-called "fourth paradigm", along with experimental, theoretical and computational ones. Its feature is the use of big data technologies. At the same time, science that uses big data is also characterized by interdisciplinarity, i.e., it combines modeling, theory and experiment.

Working with big data for both scientific and engineering calculations has become available mainly due to the development of cloud technologies, as well as the emergence of many scientific and technical applications that use the services of cloud providers for computing and data processing [17]. At the same time, the issue of improving application performance comes to be in the foreground.

One of the main ways to improve the performance of various kinds of data center services is to parallelize calculations, which can occur both at the hardware and software levels [17]. The distribution of workloads can be based on the parallelization of the data or tasks themselves. By now, many frameworks have been developed with a parallel approach to speed up data-intensive applications. These include, for example, the Google MapReduce [7] framework, Hadoop MapReduce [2], Apache Spark [3] and many others [11].

In addition to addressing development-level performance and support environments for data-intensive applications, there is a natural challenge in practice to optimally allocate

*Corresponding author: avgorbunova@list.ru

data center resources while meeting quality-of-service agreements. Due to the lack of good tools to objectively assess their performance characteristics, the main strategy at present is to overprovision resources, half of which are idle most of the time [1, 5]. This leads to a significant increase in the cost of maintaining equipment (servers).

Most data center services for large-scale computing are based on fork-join structures, they are the main component of the data processing of parallel and/or distributed computing systems. A fork-join structure is understood as a system, upon entering which the task is divided into parts – independent subtasks, each of which is sent for service to the corresponding node. The task is considered completed after processing the last of its components. Thus, the parallelization of data or tasks is modeled. In this case, the processing time of the entire task will be determined by the maximum of the processing times of its subtasks.

From the point of view of performance and efficient use of resources, it is of interest to predict such characteristics of big data analysis systems as its average response time, including in the area of high loads. The equally important characteristic that needs to be evaluated is the tail of latency, by which we mean the 99th percentile of the system response time distribution [5].

This article proposes the new approach for estimation of the average response time and its tail delay for data centers. The approach using neural networks allows one to quickly and with acceptable accuracy obtain estimates of the characteristics of interest. The advantage of the approach compared to previously known ones is its universality, since there are no restrictions on the architecture of fork-join systems. As a result, it becomes possible to construct a more realistic parallel computing model based on practical data. In addition, you can achieve good quality estimates with a basic knowledge of neural networks or some other machine learning methods. At the same time, the speed of the trained neural network is comparable to performing calculations using a simple analytical formula, as opposed to the existing complex computational algorithms.

The article is organized as follows: the section 2 describes the structure of the classical fork-join QS, as well as the main approaches to assessing its characteristics, the section 3 describes in detail the approach proposed by the authors of using artificial neural networks (ANN), the section 4 describes the mathematical model of the parallel structure of the big data processing center, the section 5 presents the results of numerous numerical experiments, the section 6 discusses the features of the proposed approach and the prospects for further research. In Conclusion, some results are summarized.

2. RELATED WORK

Most data centers are based on fork-join structures. By a fork-join structure (system), we mean a queuing system, each node of which is an independent QS. The fork-join type system functions as follows (fig. 2.1):

1) at the moment of its arrival a request is instantly split into K ($K \geqslant 2$) subtasks, each of which, in its turn, is queued for future service;
2) after the end of the service, the subtask enters the synchronization buffer and remains there until all related subtasks are processed, i.e. only after that the request is considered to be served and can leave the system.

A classical fork-join QS assumes that each subsystem has one server. We will expand this system to the most general case, when each QS can contain $n \geq 1$ servers, i.e., each subsystem will be a QS of type $G|G|n$. Thus, it will be possible to analyze the characteristics of the system in the conditions of allocation of additional resources. The results of the study will avoid redundancy in their selection.

One of the first works devoted to the analysis of fork-join QS with subsystems of the form $M|M|1$ is the article [14]. Here, exact expressions were obtained for estimating the

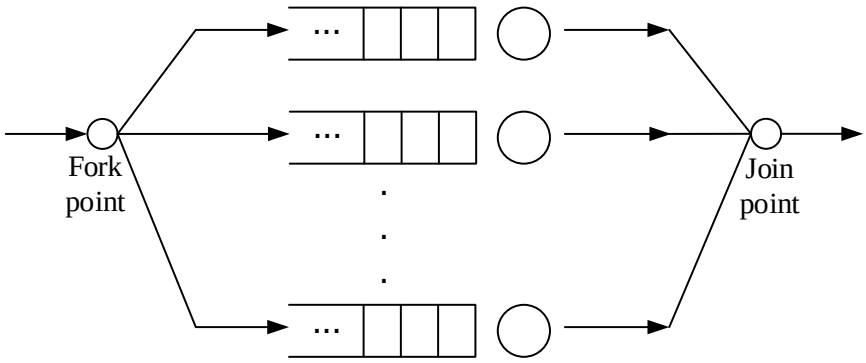


Fig. 2.1. Fork-join system model.

mean response time of the system in the case of two subsystems $M|M|1$, $K = 2$. In most of the following works on the subject of fork-join QS, various methods were used to obtain only approximations of the mean response time for $K > 2$. For a detailed overview of publications up to 2014, see [20].

Among the main approximate methods for studying fork-join QS with K subsystems of the form $M|M|1$ or $M|G|1$ are: matrix-geometric approach, interpolation based on data obtained in extreme cases of high and low input loads, an approach using elements of the theory of order statistics [6], an empirical approach (construction of analytical formulas based on data obtained by simulation), etc.

Note that there are no exact solutions for the mean response time even in the case of exponential incoming and servicing flows for $K > 2$. The complexity of the analysis is explained by the dependence of the residence times of subtasks in subsystems due to their common moments of arrival. Therefore, the analytical approach is based, as a rule, on the assumption of the absence of this dependence. As a result, the application of the estimates obtained is limited either by the number of subsystems K, or by the specific type of distribution of the servicing flow, or by insufficient approximation accuracy. This narrows the scope of the obtained analytical expressions. In particular, there are problems with modern systems due to heavy-tailed distributions. As for the case with subsystems of a more general form, i.e., $G|G|1$, there are very few studies on this topic and they have appeared mainly in recent times.

Fork-join CMO is a natural model for many real-life systems in various areas where parallel processing of tasks is carried out in order to improve performance. In this case, we are talking not only about telecommunication systems, but also about production areas (assembly of orders, logistics, etc.). Therefore, despite a slight decrease in the activity of research in this direction, their relevance is still great.

Among the recent works devoted to this subject, it is worth noting the following. [18] proposes an approach for approximating the distribution of response time in the case of homogeneous K fork-join nodes of a QS of the form $MAP|PH|1$. Numerical experiments have shown that the method is quite accurate mainly for tail delays. It is based on analytical results only for $K = 1$ and $K = 2$. The accuracy of the algorithm developed by the authors of the article depends on the choice of the constant C. The larger the value of this constant, the higher the accuracy of the resulting estimates. However, this significantly increases the complexity of the algorithm itself and, accordingly, the running time of its software implementation, since it contains operations with high-dimensional matrices. The quality of the tail delay approximation also depends on the choice of specific distributions, the load level, and the value of K. As K increases, the error increases, but, for example, for large values of K, the error does not exceed 15% in the case of the Erlang distribution of service time. And for the hyperexponential service time distribution, the approximation error does not exceed 5%.

The article [22] explores two modes of operation of the fork-join QS with subsystems of the form $M|G|1$, in which requests can be split into subtasks, the number of which is less than or equal to the total number of servers in the QS. Propositions about the asymptotic independence of the sojourn times of subtasks (when the number of servers tends to infinity) are proved, and an upper bound on the mean response time for this system is obtained. In [8] for the fork-join QS with two subsystems $M|G|1$ and non-homogeneous servers, the remaining service time is analyzed, which is necessary for the correct shutdown of the system in the event of a disconnection of the incoming flow. The asymptotic independence of the maximum residual service times for a high system load is proved.

The article [15] proposes an approach for estimating the average response time and its tail delay fork-join QS with $G|G|n$ subsystems as a model of some data center. In this case, each QS node is proposed to be considered either as a white box model, i.e., under conditions where the type of service time distribution is known. In the second variant (when there are no assumptions about the structure of the node, namely, about the type of service time distribution on it), it is assumed that we are dealing with a node model as a black box. In this case, it is proposed to estimate the mean values and variance of the response time of an individual node empirically. In other words, it is proposed to represent each node as a QS of the form $G|G|1$ by measuring the first two response times of the node of the system under study, which is technically feasible. In accordance with numerical experiments, the approximation error of the estimated response time characteristics in the region of high loads (above 0.8 for the average response time and above 0.9 for the tail delay) does not exceed 20% and 15%, respectively.

In the articles [10] and [9] for the study of fork-join QS with subsystems of type $M|M|1$ in the first case and $M|G|1$ in the second, the approach based on neural networks is proposed. The results of numerical experiments showed a good quality of approximation, which prompted the authors of the article to conduct a new study in relation to estimating the performance of data processing centers with parallel structures, the nodes of which are modeled by general systems of the $G|G|n$ type.

3. ANALYSIS METHOD OF FORK-JOIN QS USING NEURAL NETWORKS

Neural networks have become widespread due to the possibility of their application to solving poorly formalized problems. In addition, they are one of the main tools for big data analysis. It is currently difficult to name an area in which neural networks or other machine learning methods would not find their application.

The mathematical model that is used to describe the operation of a big data processing center is a queuing system. Machine learning methods and neural networks, in particular, have been applied to solving complex problems of queuing theory relatively recently, although this prospect was predictable. Complex problems in the field of queuing theory are understood as problems that cannot be solved by using well-known analytical methods, or the solution is so complicated that actually obtaining numerical results using the developed algorithms is difficult to implement even taking into account the capabilities of modern computers. An overview of publications on the application of machine learning methods to solving problems in the field of queuing theory can be found in [21].

The idea of using neural networks is based on the possibility of using them to solve the problem of forecasting or, in fact, the problem of approximating a function of several variables. It is clear that the desired estimates of the mean response time and its tail delay depend, for example, on the values of such parameters as the system load, the number of subsystems K and servers in them, the intensities for the incoming flow and for the servicing time on the servers. Therefore, the problem of finding estimates of the response time can be considered as a problem of approximating a function depending on the listed parameters.

To interpolate a function, a set of values of the input parameters and the corresponding true values of the approximated quantities are required. There are several ways to get the

true values. For example, with the help of simulation modeling or with the help of an exact analytical solution, which is not available in this case. As an alternative to the exact analytical solution, one can use, for example, the time-consuming algorithm proposed in [18] to obtain tail delays in the region where a high-precision solution exists.

The need to combine neural networks with simulation or algorithmic solution arises from the significant time costs that are required when using only a simulation or only an algorithm. Therefore, we limit the amount of input data for which it is necessary to use simulation modeling or a computational algorithm. Then, on the resulting set, we train the neural network, which will allow us to predict the characteristics for any intermediate values of input parameters without restrictions on their number in the minimum time comparable to the time required for calculations using a simple analytical formula.

For clarity, we describe the main stages of the proposed method:

1. obtaining by means of simulation modeling or a labor-intensive computational algorithm the values of the characteristics of the analyzed system of interest for a finite set of values from the given numerical intervals for the input parameters on which the performance of the system depends;
2. training an intellectual model on the data obtained using simulation by one of the machine learning methods in order to solve the forecasting problem;
3. almost instantaneous assessment of the desired performance characteristics for any other intermediate values of input parameters on the same numerical intervals using a trained intelligent model.

Among the main advantages of the described technique, universality can be distinguished, since a simulation modeling is sometimes the the only possible way to analyze complex systems. However, it is a resource-intensive tool. Thanks to the use of neural networks, this disadvantage can be eliminated.

4. MATHEMATICAL MODEL OF THE PARALLEL STRUCTURE OF THE BIG DATA PROCESSING CENTER

Let us describe in more detail the architecture of the fork-join QS, which we will use to model the operation of a data center. The system consists of K subsystems of type $G|G|n$ (fig.4.2). A request entering the system is split into K subtasks, each of which is queued in the corresponding subsystem. Servicing in subsystems occurs in the order of receipt of subtasks (First In First Out, FIFO discipline). Each subsystem contains the same number of servers n, $n \geqslant 1$. Selecting a multiline system will allow you to analyze the performance improvement of nodes by provisioning additional resources, as additional server replicas are modeled in this way. Based on the results of the [15] study, we will restrict ourselves to the case with 3 server replicas, i.e. we will check the accuracy of the proposed approach for $1 \leqslant n \leqslant 3$.

As a service time distribution we will consider a truncated Pareto distribution with a distribution function and density of the form [4]:

$$F(x) = \frac{1-(L/x)^{\alpha}}{1-(L/H)^{\alpha}}, \quad f(x) = \frac{\alpha L^{\alpha} x^{-\alpha-1}}{1-(L/H)^{\alpha}}, \quad 0 \leqslant L \leqslant x \leqslant H, \quad \alpha > 0.$$

The truncated Pareto distribution is a three-parameter distribution. The parameter α is a shape parameter, and the parameters L and H are in fact the minimum and maximum values of a random variable with a given distribution.

In [15], based on empirical data for the Google system from [12], the following values for parameters of service time distribution are proposed:

$$\alpha = 2.0119, \quad L = 2.14, \quad H = 276.6, \tag{4.1}$$

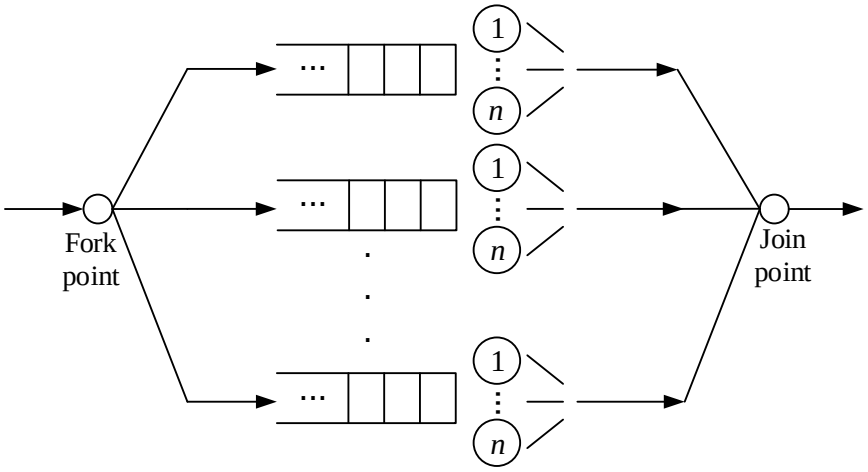


Fig. 4.2. Fork-join model of QS with subsystems of type $G|G|n$.

i.e. the minimum service time is 2.14 ms, and the maximum is 276.6 ms. The k-th moment of the truncated Pareto distribution is given by:

$$m_k = \frac{L^\alpha}{1-(L/H)^\alpha}\frac{\alpha(L^{k-\alpha}-H^{k-\alpha})}{\alpha-k}, \quad \alpha \neq k.$$

Accordingly, we have that the average service time is approximately equal to 4.22 ms, the variance is 22.34, and the coefficient of variation, which is the ratio of the root of the variance to the average value of a random variable, will be approximately $CV \approx 1.22$.

To distribute the incoming flow, consider several options. Since some studies allow the assumption of the Poisson nature of the incoming flow ($CV = 1$) for data processing centers [13, 16], we will not exclude the case of an exponential distribution for the time interval between adjacent incoming requests. We also consider two more types of distribution for the incoming flow with a coefficient of variation different from unity. In particular, for the Erlang distribution it is known that its coefficient of variation is always less than one. Therefore, consider the Erlang distribution with a density of the form

$$f_1(x) = \beta^2 x e^{-\beta x}, \quad x \geqslant 0, \quad \beta > 0, \tag{4.2}$$

and with $CV = 1/\sqrt{2} \approx 0.7$. We also consider a heavy-tailed distribution ($CV > 1$), namely the gamma distribution with density

$$f_2(x) = \frac{\gamma^k}{\Gamma(k)} x^{k-1} e^{-\gamma x}, \quad x \geqslant 0, \quad \gamma > 0, k > 0 \tag{4.3}$$

for which the coefficient of variation $CV = 1/\sqrt{k}$ for $k = 0.25$ will be equal to 2.

5. NUMERICAL EXPERIMENT

Let's check the performance and quality of the approximation of the proposed approach using neural networks. The first version of the data center architecture is a fork-join QS with a Poisson input flow with an intensity λ inversely proportional to the average time between adjacent arrivals of requests $a = 1/\lambda$. The service time has a truncated Pareto distribution with parameters from (4.1) with an average value of $b = 4.22$ ms. In fact, each subsystem is a QS of the form $M|G|n$.

To train the neural network, we will use the input data from the table 5.1, where the load factor

$$\rho = \frac{b}{na} = \frac{\lambda b}{n}$$

takes values on the segment $[0.1, 0.9]$ with a step of 0.1, the number of subsystems K varies from 2 to 24, and the number of servers n from 1 to 3. The output of the neural network will be the mean response time and its 99th percentile. The values of the output parameters were obtained by using simulations carried out in the Python software environment.

Table 5.1. Input data for constructing estimates of the mean response time and its tail delay.

No.	1	2	...	9	10	...	27	28	29	...	621
ρ	0.1	0.2	...	0.9	0.1	...	0.9	0.1	0.2	...	0.9
n	1	1	...	1	2	...	3	1	1	...	3
K	2	2	...	2	2	...	2	3	3	...	24

Let's split the data from the table 5.1 into two sets corresponding to the level of low and high loads, i.e. for $\rho \in [0.1, 0.5]$ and $\rho \in [0.6, 0.9]$. And we will train the neural network on each of these two sets. In this case, the data set is divided in the ratio of 80% and 20% into training and test samples. The latter will be used for validation during training. By the standards of neural networks, it cannot be said that the number of data sets for irradiation is large. Nevertheless, we will check the accuracy of the proposed approach under these conditions.

A two-layer perceptron with two hidden layers of 10 neurons each with a logistic activation function $\phi(x) = 1/(1 + e^{-x})$ was chosen as the neural network structure. To improve the approximation accuracy, we will build two neural networks: the first one is for predicting the mean response time $E[R_K]$, the second one is for the tail delay $R_K(99)$. Training will be carried out by the method of back propagation of errors or the Adam method in the Python software environment, where simulation modeling was carried out.

For an objective check of the forecast quality of trained neural networks, we will use intermediate data absolutely unknown to it from the 5.2 table. For a neural network trained in the area of low loads, consider the values for the prediction $\rho \in [0.15, 0.55]$, and for a neural network trained in the area of higher loads $\rho \in [0.65, 0.85]$.

Table 5.2. Intermediate input data for constructing estimates of the mean response time and its tail delay.

No.	1	2	...	8	9	...	24	25	26	...	552
ρ	0.15	0.25	...	0.85	0.15	...	0.85	0.15	0.25	...	0.85
n	1	1	...	1	2	...	3	1	1	...	3
K	2	2	...	2	2	...	2	3	3	...	24

The figures 5.3 show the deviation of the estimates of the studied characteristics from their true values obtained using simulation modeling. For more detail, let's analyze the relative approximation error

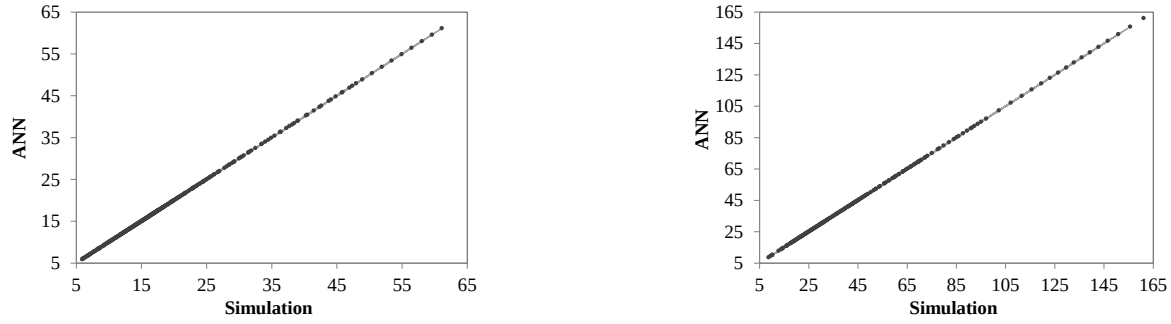
$$PE = \left| \frac{\widehat{x}_j - x_j}{x_j} \right| \cdot 100\%$$

for each set of intermediate input data, as well as its average value

$$MAPE = \frac{1}{n} \sum_{j=1}^{N} \left| \frac{\widehat{x}_j - x_j}{x_j} \right| \cdot 100\%,$$

where $\widehat{x}_j$ is an estimate of the characteristic under study (mean value or tail delay of the response time), obtained by using analytical formulas, and x_j is the real value of one of the

estimated characteristics, obtained as a result of fork-join QS simulation , $j = \overline{1, N}$, N is the number of data sets in the sample for the estimation of the approximation error.



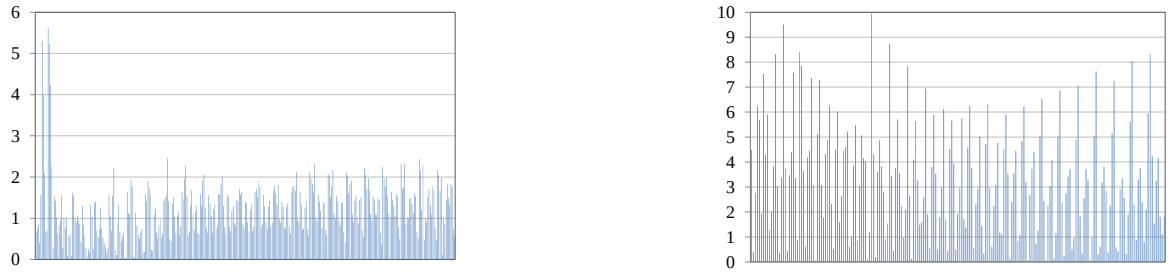
a) b)

Fig. 5.3. The mean response time of a system with Poisson input, a truncated Pareto distribution of service time and load factor a) $\rho \in [0.15, 0.55]$; b) $\rho \in [0.65, 0.85]$

Let us determine the maximum, minimum and mean value of the relative approximation error (Table 5.3). We will also plot relative error histograms in the order in which K, ρ, and n values are presented in the 5.2 table. 98.85% of the mean response time approximation

Table 5.3. Approximation errors of estimates of the mean response time and its tail delay for a system with Poisson input, obtained using a neural network on data sets from the table 5.2

Estimated characteristic	Error types		
	Maximum PE, %	Minimum PE, %	MAPE, %
$E[R_K]$, $\rho \in [0.15, 0.55]$	5.61689	0.02737	1.22500
$E[R_K]$, $\rho \in [0.65, 0.85]$	9.93912	0.05324	3.28332
$R_K(99)$	8.11459	0.00244	1.62527

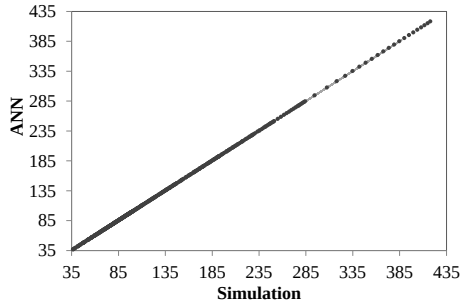


a) b)

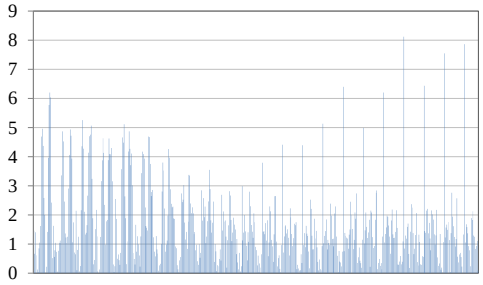
Fig. 5.4. Approximation errors for the mean response time of a system with Poisson input, a truncated Pareto distribution of service time and load factor a) $\rho \in [0.15, 0.55]$; b) $\rho \in [0.65, 0.85]$.

error for $\rho \in [0.15, 0.55]$ does not exceed 3% (fig.5.4a). For $\rho \in [0.65, 0.85]$, the result of predicting the average response time by the neural network turned out to be somewhat worse. In this case, the number of approximation errors, which does not exceed 7%, is approximately 92.27% of their total number (fig.5.4b). In this case, the maximum approximation error does not exceed 10%. For the 99th percentile of the response time distribution (fig.5.5a), the situation is such that 97.54% of the approximation errors do not exceed 5% (fig.5.5b).

Now let's analyze the case with the Erlang distribution of service time from (4.2). To do this, consider a smaller training input set from the 5.4 table. The Erlang distribution parameter is given by $\beta = 2n\rho/b$. After training the neural network, we build the corresponding



a)



b)

Fig. 5.5. For a system with Poisson input and a truncated Pareto distribution of service time a) the 99th percentile of the response time distribution, b) approximation errors of the 99th percentile.

Table 5.4. Input data for constructing estimates of the mean response time and its tail delay in the case of the Erlang or Gamma distribution for the incoming flow of requests.

No.	1	2	...	27	28	29	...	405
ρ	0.1	0.2	...	0.9	0.1	0.2	...	0.9
n	1	1	...	3	1	1	...	3
K	2	2	...	2	3	3	...	16

estimates for the intermediate data from the 5.5 table. So, for the mean response time (fig.5.6a) 98.51% of errors does not exceed 5% (fig.5.7a), and for its tail delay (fig.5.6b), this percentage is already 99.40%, i.e., for only two predictive values, the relative error ranges from 5% to 7% (fig.5.7b). The maxima, minima, and average values of the approximation errors are presented in the 5.6 table. The maximum error in the approximation of $E[R_K]$ and $R_K(99)$ does not exceed 7.8% and 6.3%, respectively, which is acceptable, especially considering that almost 99% of the errors are within 5%, which evidenced by the low values of their average values.

Table 5.5. Intermediate input data for constructing estimates of the mean response time and its tail delay in the case of the Erlang or Gamma distribution for the incoming flow of requests.

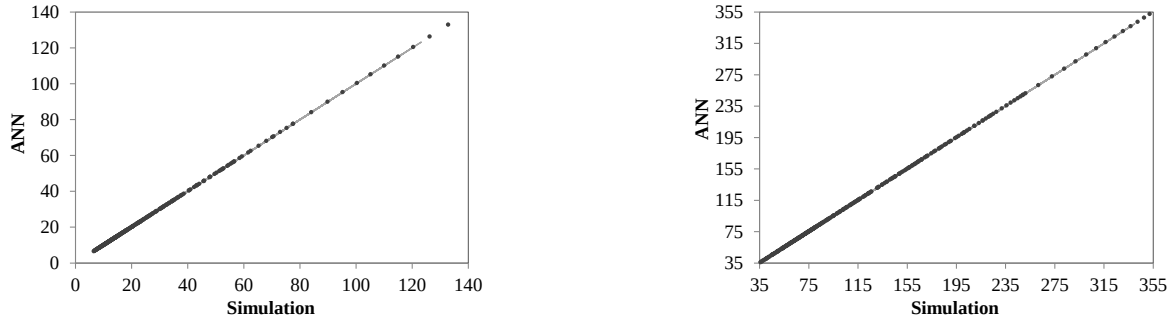
No.	1	2	...	24	25	26	...	336
ρ	0.15	0.25	...	0.85	0.15	0.25	...	0.85
n	1	1	...	3	1	1	...	3
K	2	2	...	2	3	3	...	16

Table 5.6. Approximation errors of estimates of the mean response time and its tail delay for a system with an Erlang distribution of the incoming flow, obtained using a neural network on data sets from the table 5.5.

Estimated characteristic	Error types		
	Maximum PE, %	Minimum PE, %	MAPE, %
$E[R_K]$	7.82757	0.00518	1.75565
$R_K(99)$	6.27911	0.01538	1.55339

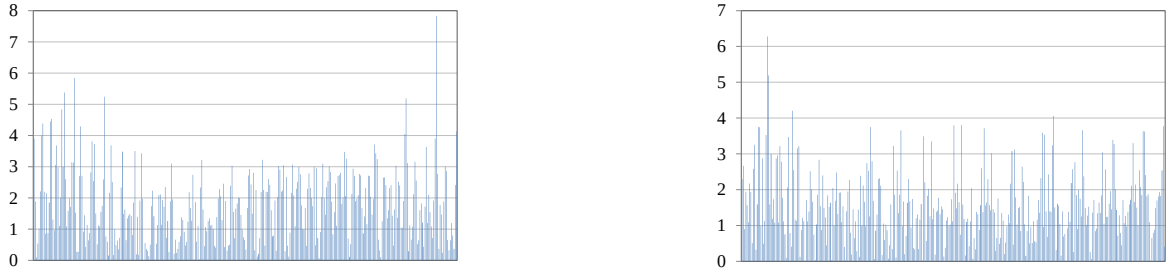
If the time intervals between adjacent incoming requests have a gamma distribution (4.3) with parameters $k = 0.25$ ($CV = 2$) and $\gamma = kn\rho/b$, then after training the neural network on the input data from the table 5.4 we get the following result for intermediate data from the table 5.5.

For the mean response time (fig.5.8a) 96.13% of the relative errors of their total number does not exceed 6% (fig.5.9a). For the tail delay (fig.5.8b), the percentage of errors that do



a) b)

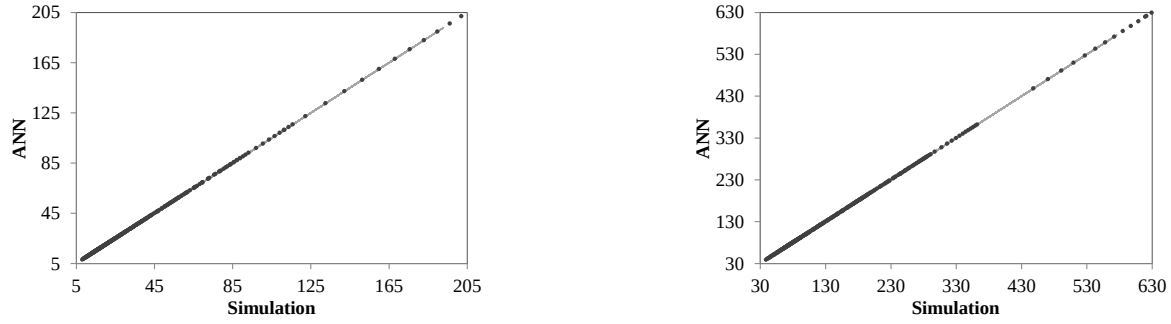
Fig. 5.6. For a system with an Erlang distribution of the incoming flow and a truncated Pareto distribution of service time, a) the mean response time, b) the 99th percentile of the response time distribution.



a) b)

Fig. 5.7. System approximation errors with Erlang distribution of the incoming flow and a truncated Pareto distribution of the service time for a) the mean response time, b) the 99th percentile of the response time distribution.

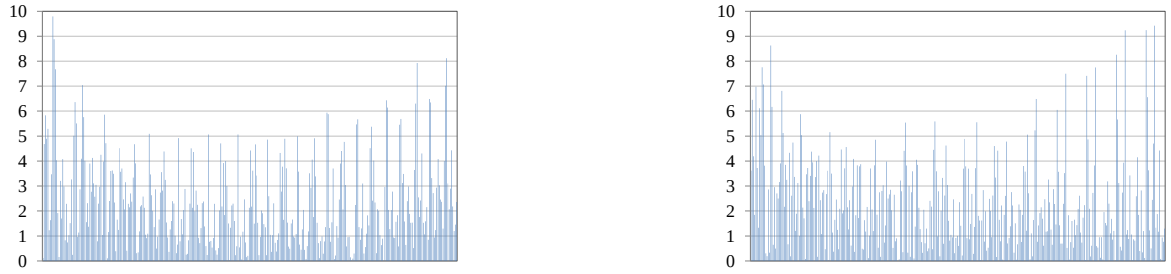
not exceed the threshold of 6% is 94.64% (fig.5.9b). In this case, the maximum relative approximation error does not exceed 9.8% in both cases (table 5.7).



a) b)

Fig. 5.8. For a system with a gamma distribution of the incoming flow and a truncated Pareto distribution of the service time a) the mean response time, b) the 99th percentile of the response time distribution.

If we analyze the level of load reduction in case of allocation of additional servers, then the situation is as follows. Regardless of the type of distribution for the incoming flow out of the three considered, if each node has two servers ($n = 2$) instead of one, the mean response time and tail delay are approximately halved, which was expected. If the number of servers is $n = 3$, then the average response time and its tail delay are reduced by an average of 65% compared to $n = 1$. The calculation took place for the high load level $\rho \in [0.6, 0.9]$, since it



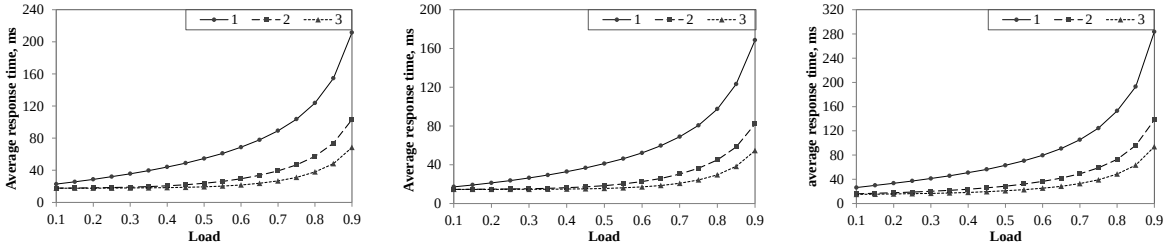
a) b)

Fig. 5.9. System approximation errors with a gamma distribution of the incoming flow and a truncated Pareto distribution of the service time for a) the mean response time, b) the 99th percentile of the response time distribution.

Table 5.7. Approximation errors of estimates of the mean response time and its tail delay for a system with a gamma distribution of the incoming flow, obtained using a neural network on data sets from the table 5.5.

Estimated characteristic	Error types		
	Maximum PE, %	Minimum PE, %	MAPE, %
$E[R_K]$	9.79003	0.00344	2.34142
$R_K(99)$	9.42462	0.01472	2.31488

is in this case that the allocation of additional resources to improve the quality of service is an urgent issue. For clarity, the figure 5.10 shows graphs of the mean response time at a fixed K depending on the load level for various values of $n = 1, 2, 3$.



a) b) c)

Fig. 5.10. Mean response time, if the time intervals between adjacent incoming requests have a) an exponential distribution, $K = 24$, b) an Erlang distribution, $K = 16$, c) a gamma distribution, $K = 16$; 1 — $n = 1$, 2 — $n = 2$, 3 — $n = 3$.

Summarizing, we can draw the following conclusions. For 93% of the input data in the case of an approximation of the mean response time and for 96% of the data for the tail delay, the approximation error does not exceed 5%. Since we considered the absolute values of the relative error, then the compensation of the approximation error in the amount of 5% can, in the case of positive estimates, actually lead to an overallocation of resources in the amount of 10%. However, compared to the 50% overprovisioning of today's data centers, the 40% savings is a significant benefit.

6. DISCUSSION

Now let's analyze the results in more detail. The use of neural networks as one of the methods of machine learning for predicting the mean response time and its tail delay makes it possible to obtain qualitative estimates without any restrictions on the structure of the system

being modeled. This is one of the main advantages of the proposed approach compared, for example, with analytical methods.

Based on the analysis of the publications available to date, a partial review of which was presented in this paper, we can draw the following conclusion. The use of analytical methods, if possible, is usually effective only for certain types of incoming or serving flows or only for some areas of the parameters of these distributions and, accordingly, the level of load (high or low), and also limits the structure of the system (number of servers or buffer capacity in subsystems).

On the other hand, the use of neural networks requires preliminary data preparation for its training. In this case, simulation modeling was used. By itself, simulation modeling makes it possible to obtain practically exact values of the desired characteristics. However, it requires serious computational resources and time costs, so its use in its pure form is not rational. Machine learning significantly narrows the range of parameters for which simulation is required. Due to this, the time spent on its implementation is significantly reduced. At the same time, after training the neural network, it becomes possible to obtain estimates of acceptable quality for any number of intermediate values of the input parameters without any time spent on this.

Training of neural networks aimed at a satisfactory forecast quality does not require too deep immersion in this area due to the fact that many algorithms for their training are qualitatively implemented in various computing environments, there are ready-made libraries, etc. Nevertheless, if it is necessary to obtain more accurate forecasts, for example, in comparison with those presented in this paper, it is possible to involve relevant specialists.

Among the many listed advantages of the proposed approach, it also has some weaknesses in the context of making a decision to allocate additional resources to compensate for forecast errors. Despite the low approximation error, which on average does not exceed 5% for 90% of the data, and in some cases even 3%, there are still a small number of errors that exceed, although not significantly, these thresholds. And it is impossible to predict in advance for which specific values (ranges of values) of the input parameters this will happen. This fact is not affected by the type of distribution, nor by its coefficient of variation, nor by the level of loading, etc., as is the case with approximate analytical approaches. The main reasons for this phenomenon, apparently, are the features of ANN training, the algorithm chosen for this, the size of the training data set, the architecture of the neural network (the number of hidden layers, the number of neurons, activation functions, etc.). Therefore, the continuation of research in this direction, aimed at compensating for this shortcoming, seems to be an urgent task.

7. CONCLUSION

The paper proposed the approach to assess the performance of big data processing center services. Using a combination of simulation modeling with neural networks, estimates were obtained for the mean response time and its tail delay (99th percentile of the response time distribution). The advantage of the approach compared to previously known ones is its universality, since there are no restrictions on the architecture of fork-join systems. The speed of the trained neural network is comparable to performing calculations using a simple analytical formula, as opposed to the existing complex computational algorithms. At the same time, the approximation quality is quite high and does not depend on the architecture of the mathematical model used.

ACKNOWLEDGEMENTS

The reported study was obtained within RFBR grant (project No. 19-29-06043).

REFERENCES

1. Alesawi, S., Nguyen, M., Che, H. & Singhal, A. (2015). Tail latency prediction for datacenter applications in consolidated environments, *in Proc. IEEE International Conference on Computing, Networking and Communications (ICNC)*, Honolulu, HI, USA, 265–269.
2. Apache: Apache hadoop. (2022, April). [Online]. Available: https://hadoop.apache.org
3. Apache spark. (2022, April). [Online]. Available: https://spark.apache.org
4. Harchol-Balter, M. (2013). *Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge, U.K.: Cambridge Univ. Press.
5. Blake, G. & Saidi, A. G. (2015). Where does the time go? Characterizing tail latency in memcached. *in Proc. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 21–31.
6. David, H. A. & Nagaraja H. N. (2003) *Order Statistics*. Hoboken, NJ: John Wiley & Sons.
7. Dean, J. & Ghemawat, S. (2004). Simplified data processing on large clusters, *in Proc. 6th Symposium on Operating Systems Design and Implementation (OSDI)*, San Francisco, CA, 137–150.
8. Gorbunova, A. V. & Lebedev, A. V. (2020). Bivariate Distributions of Maximum Remaining Service Times in Fork-Join Infinite-Server Queues, *Problems of Information Transmission*, 56(1), 73–90.
9. Gorbunova, A. V. & Lebedev, A. V. (2022). Response time estimate for a fork-join system with Pareto distributed service time as a model of a cloud computing system using neural networks, *Communications in Computer and Information Science*, 1552, 318–332.
10. Gorbunova, A. V. & Vishnevsky, V. M. (2020). Estimating the response time of a cloud computing system with the help of neural networks, *Advances in Systems Science and Applications*, 20(3), 105–112.
11. Khalid, M. & Yousaf, M. M. (2021). A comparative analysis of big data frameworks: an adoption perspective, *Applied Sciences*, 11(22), 11033.
12. Meisner, D., Junjie, W. & Wenisch, T. F. (2012). BigHouse: A simulation infrastructure for data center systems, *in Proc. IEEE International Symposium on Performance Analysis of Systems & Software*, New Brunswick, NJ, USA, 35–45.
13. Meisner, D., Sadler, C. M., Barroso, A. L., Weber, W. D. & Wenisch, T. F. (2011). Power management of online data-intensive services, *ACM SIGARCH Computer Architecture News*, 39(2), 319–330.
14. Nelson, R. & Tantawi, A. N. (1988). Approximate analysis of fork/join synchronization in parallel queues, *IEEE Transactions on Computers*, 37, 739–743.
15. Nguyen, M., Alesawi, S., Li, N., Che, H. & Jiang, H. (2020). A black-box fork-join latency prediction model for data-intensive applications, *IEEE Transactions on Parallel and Distributed Systems*, 31(9), 1983–2000.
16. Nguyen, M., Alesawi, S., Li, N., Che, H. & Jiang, H. (2018). ForkTail: A black-box fork-join tail latency prediction model for user-facing datacenter workloads, *in Proc. 27th Int. Symp. High-Perform. Parallel Distrib. Comput.*, 206–217.
17. De Oliveira, D. C., Liu, J. & Pacitti, E. (2019). Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments. *Synthesis Lectures on Data Management*, 14, 1–179.
18. Qiu, Z., Prez, J. F. & Harrison, P. (2015). Beyond the mean in fork-join queues: Efficient approximation for response-time tails, *Performance Evaluation*, 91, 99–116.
19. Reinsel, D., Gantz, J. & Rydning, J. (2018). *IDC Report: The Digitization of the World From Edge to Core*. IDC White Paper No.US44413318. Framingham, MA: International Data Corporation.
20. Thomasian, A. (2014). Analysis of fork/join and related queueing systems, *ACM Computing Surveys (CSUR)*, 47(2), 17:1–17:71.

21. Vishnevsky, V. M. & Gorbunova, A. V. (2022). Application of machine learning methods to solving problems of queuing theory, *Communications in Computer and Information Science*, in print.
22. Wang, W., Harchol-Balter, M., Jiang, H., Scheller-Wolf, A. & Srikant, R. (2019). Delay asymptotics and bounds for multitask parallel jobs, *Queueing Systems*, 91, 207–239.

