Journal of Theoretical and Applied Information Technology
30th June 2020. Vol.98. No 12
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

APPLICATION OF LOAD BALANCING ALGORITHMS TO IMPROVE THE QUALITY OF SERVICE DELIVERY USING MODIFICATIONS OF THE LEAST CONNECTIONS ALGORITHM

SERIK JOLDASBAYEV, GULNAR. BALAKAYEVA, ORYNBASAR JOLDASBAYEV

AL-FARABI KAZAKH NATIONAL UNIVERSITY

E-mail: serykjoldasbaev@mail.ru

ABSTRACT

To develop solutions to the problems of providing high-speed Internet, that is, a high-quality service, up to a certain point, there is the possibility of improving quality by increasing the hardware resources of the system, but, as practice shows, quantity does not always mean quality, and the effectiveness of the service delivery system takes into account the advantageous positioning of resources with algorithmic load balancing on servers with maximum benefit, both for the user and for the party providing services .

This article provides the results of research and analysis of balancing algorithms, implementation methods for load balancing on servers and improving the quality of service delivery. Research in this direction is very relevant and in demand, the article provides an analysis and description of static and dynamic solutions, the advantages and disadvantages of algorithms. A modernization of the Least Connections algorithm is proposed.

Keywords: *Improving The Quality Of Service Delivery, Qos, Load Balancing, Balancing Algorithms.*

1. INTRODUCTION

Currently, Internet use is growing rapidly around the world. So, in Kazakhstan, for the first time, the real Internet appeared back in 1997 and cost $ 10 per hour, with an average of 14,400 bps of services. The Internet was a luxury, its accessibility and speed increased over time, but nevertheless it still leaves much to be desired in many regions of the country, even in the radius of large megacities [7]. It will be correct if we note that the project “Kazakhtelecom” OJSC - “Internet Zone” played a large role [8]. At the end of 2000, 1,945 sites were registered in Kazakhstan, while now 8-12 new sites appear in one week, and the number of resources and users around the world doubles in about one year [4].

The constant increase in volumes and resources when using the Internet leads to the fact that for many services it is important to be able to work stably under heavy loads, since for many competitive companies that use this or that service, this plays a significant role in all production issues, up to customer migration to competitors. And it is not surprising that many prefer to use the services of large companies advanced in this area, for example, Amazon, Google, etc., which use server clusters as a means of application deployment and as load balancers [5]. Such server clusters (hereinafter referred to simply as servers) make it possible to not worry about system failures - well-designed load balancers provide optimal control of incoming requests to servers, which contributes to the implementation of uniform load on nodes, reduces performance losses and ensures the maximum possible response time to a request.

To date, many load balancing algorithms have been developed on servers, but not all algorithms are applied in practice. Server load balancing algorithms are the division of the computational load between the processors of the computing system, which determines the optimal load of each processor as much as possible for most of the time [26]. Basically, many algorithms work taking into account the load on a particular server (server clusters), taking into account only its computing power. In many cases, development testing is carried out in homogeneous systems.

Typically, large services are deployed on clusters consisting of many, in particular, heterogeneous nodes [6]. Under such conditions, the problem arises of distributing balancing dynamically, possibly even by different operating systems. Load balancing in such services is an urgent task, since there is no universal solution for all servers. Nevertheless, in order to expand the capabilities of such systems, certain balancing algorithms are applicable.

With static balancing, the algorithm is predefined, distributes requests to nodes according to certain rules, does not take into account a load of servers in real-time [27].

Semi-dynamic load balancing on servers is due to the separation of computing tasks at the initialization stage and as requests arrive on the server.

Dynamic load balancing - the separation and distribution of objects is periodically updated throughout the entire response time to user requests and objects are moved across computing nodes in accordance with a more optimal plan. Separation of requests can be determined by various specified criteria - workload, the performance of nodes, etc. heuristic indicators [28].

2 MODELLING OF SERVICE DELIVERY SYSTEMS USING QUEUING SYSTEMS

Queuing systems (QS) describes the totality of interactions between servers and applications representing requests to these servers. Applications processed by one system can freely enter the input to another. The system is closed if the number of applications in it is set by the final, constant, number of customers, and each customer expects a processed application before sending the next application. An example of such systems is usually local area networks. If the applications come from outside in the form of an incoming stream and the processed applications are displayed from the network, then the system is open. Networks of systems where both types of applications can be mixed systems [29].

An open network consisting of one server works by the following principle: requests are received in the system, they are waiting in a queue, they are processed by the server one by one and submitted as processed requests at the output.

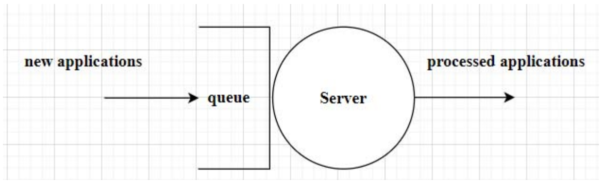

Figure - 1. Scheme For Submitting And Processing Applications On The Server

Queuing systems can be divided into classes according to many parameters, such as the type of distribution function, distribution function of the processing time of an application, etc. For the network, parameters can be set, such as the intensity of the incoming stream and the average time for servicing the application on the server. Having solved the equations of the theory of queuing, one can determine the following characteristics of system performance:

- system load - a fraction of the time required to process applications;
- time spent - the average time during which the application has been in the system from the time it was sent to the response;
- queue length - the number of applications on the server;
- throughput - the average number of applications processed by the system per unit time.

A queuing network can be described as a graph $G =< N, E >$, where $N = \{C, Q, S, T\}$: C – plenty of sources corresponding to application classes,
Q – plenty of queues, S – plenty of drains for withdrawing requests from the system. The rules for setting the system operation are determined by the probability of sending a request along the edges of the graph when sending it from a vertex to another vertex. For queues, the priority policy for servicing requests is set, for example, FIFO policy: first-come-first-served [30].

The Kendall method is mainly used to classify Queuing systems [31]. We introduce the following relations are valid for all arbitrary Queuing systems:

T – length of time monitoring the system;

A – the number of applications received in the queue;

C – the number of applications processed;

Using these notations, we describe the following system characteristics:

Incoming flow rate:

$$\lambda = \frac{A}{T},$$

System throughput:

$$X = \frac{C}{T}.$$

If the system consists of one server and the processing time of applications is known T_p, system load can be determined U and average application processing time T_a:

$$U = \frac{T_p}{T},$$

$$T_a = \frac{T_p}{C}$$

In this case, the workload can be expressed as follows:

$$U = XT_a.$$

According to Little's definition [32], the average number of applications L in the system is equal to the average input stream intensity λ_a times the average time the application spent in the system R (time equal to the sum of the waiting time and processing the application):

$$L = \lambda_a R,$$

where

$$R = T_p + T_a$$

These descriptions fully satisfy the conditions of Jackson networks [33], consisting of m servers, which is characterized by the following:

- the incoming flow of applications is Poisson,
- the time for processing applications on each server has an exponential distribution,
- the processed application on one server can go to another server or leave the system with a fixed probability P_{ij}, provided that the transition events are independent of each other and have the same distribution,
- the load on all servers is less than one $U < 1$.

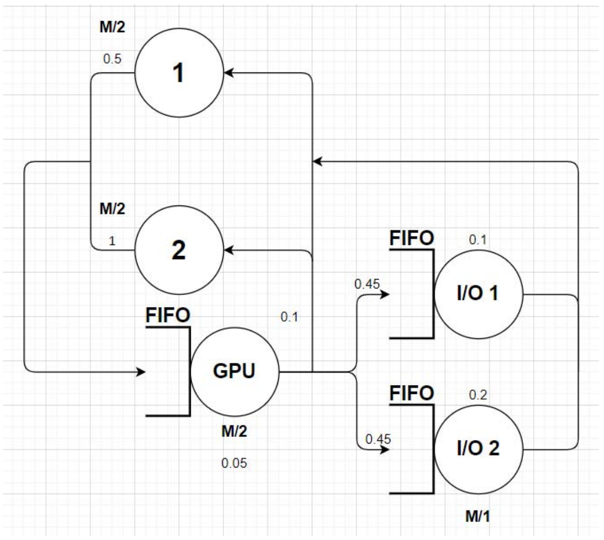

Figure - 2. Example QS for computer

For the Jackson network, you can specify that the servers should be considered independently of each other and each server creates a separate Poisson stream of processed applications. An example of such a network is a network of two servers connected in series with an incoming Poisson stream of applications. The transfer of applications from one network to another is caused by the absence of loops in the network graph. If this condition is not met, then it will violate the Markov property of the system and the flows of processed applications will not be Poisson. It should be noted that for networks with loops, Jackson's theorem is valid - in such a network, servers can be considered independently and the distribution of the number of applications in server queues can be represented as a product:

$$p(k_1, \dots, k_n) = p_1(k_1) \cdot \ldots \cdot p_n(k_n),$$

$p_i(k_i)$ – the probability of finding k_i applications on the i-th server. This representation is called the multiplicative form (product forms).

A generalized Jackson network is called a BCMP network [34], where multiple request classes can be present and servers must belong to one of the following types:

- FCFS - processing requests in order of receipt, indicative distribution of processing time with the same intensity for all classes;
- PS - processor split mode, each requested class is served independently with different processing time distributions;
- IS - infinite server-requests stay for some time, regardless of the number of requests;

- LCFS-PR - the processing of the last received application with the displacement of the processing one.

The BCMR theorem states that in BCMR networks, servers can be considered independently of each other and the probability that a certain number of requests are located on the system servers can be obtained as the product of the corresponding probabilities for individual servers:

$$p(\bar{k}) = p_1(\bar{k}_i) *..* p_n(\bar{k}_n)$$

where $\bar{k} = (\bar{k}_1, \dots, \bar{k}_n)$, $\bar{k}_i = (k_{i,1}, .. k_{i,r}, \dots, k_{i,R})$ and $k_{i,r}$ – number of requests r on server i.

When considering flows in transport networks, a Braess-type paradox was formulated. The paradox is that when using a suboptimal algorithm for distributing traffic flows, adding additional paths in the system can lead to an increase in the time of passing the request from the source to the recipient. Similar paradoxes occur in load distribution in distributed and parallel systems. In [35], we consider a system of two different servers that can transmit requests to each other for processing. It is shown that if each server tries to minimize the average time spent on requests (a criterion for a class of requests), then for some positive values of the communication channel bandwidth between servers, the average time spent on each of them is longer than in the absence of an exchange of requests. At the same time, it seems obvious that the stay time, at least, should not increase. This is the paradox. Note that if the average stay time for two servers is minimized (the criterion for all requests) or for each request (an individual criterion), then there is no paradox. The phenomenon of a Braess-type paradox is related to the structure of the equilibrium achieved in the system. In the case of a class criterion, a Nash equilibrium occurs, which may not correspond to the global optimum achieved using the General criterion. A detailed analysis of the types of equilibrium and paradoxes that arise during load balancing in distributed systems can be found in work [36].

2. REVIEW OF PROCEEDINGS

A large number of well-known scientists, for example, Kleinrock, S. Blake, D. Grossman, Z. Wang, Steklov VK, Berkman LN, as well as research centres such as Mobile Ad-hoc Networks, Internet Engineering Task Force, Center for Embedded Networked Sensing deals with the management and distribution of traffic. However, despite the huge number of publications and the efforts of manufacturers, the task of constructing traffic patterns that best reflects its functioning in real conditions is still not solved [21-25].

In [37], the problem of constructing optimal tree structures for distributed service networks (Grid) is investigated. A system of N nodes is considered. the average request processing time in the open model for various configurations of interactions between nodes is calculated using the QS network theory. In this case, the system structure is a tree, and the master-slave paradigm is used to organize interaction between nodes. In our opinion, the main value of the work consists of using several QS models to describe a single computational system. To describe the process of testing the status of slave nodes and the process of processing requests by slave nodes, various models are used, the parameters of which are related to each other.

In [38] the task of minimizing the response time of a distributed system is set. The system is modelled by a Queuing network consisting of servers connected to a network with an arbitrary topology. The performance characteristics of the nodes can be different. It is assumed only that each request can be processed on any server. Requests from outside come in the form of a Poisson intensity stream.

The average value analysis (MVA) method is used to calculate the characteristics of closed Queuing networks. There is an exact MVA method and an approximate one. In [39], the exact MVA method for closed networks with a single request class was considered.

The Satin system described in the work of Nieuwpoort [40] is intended for executing programs, solving the problems on the principle of "divide and conquer" on systems with distributed memory. The Satin system uses an algorithm for Random Stealing (RS) subtasks to distribute the load between locally distributed nodes. For the case of load distribution between nodes of different clusters connected by a slow WAN, another paper by Newport, [41] proposed an algorithm for random borrowing taking into account Cluster-Aware Random Stealing (CRS). The proof of the stability of the RS algorithm for systems with a complete graph of the structure is given in [42]. Taking into account the place of the algorithm in the classification based on the research presented in [41], it follows that the algorithm is stable, will be ineffective for a medium and lightly loaded system, and also for solving a large number of small tasks, since only one task is transferred during balancing. for a large load, the algorithm will be highly efficient, as the author's research shows.

In [43] the PICO system is described, which is a Framework developed in C ++ using MPI to implement general branch and bound method algorithms. The environment of this system is a mechanism for implementing various algorithms based on the application of the Inversion of Control (IoC) principle [44] for the interaction of branches and borders. According to the description of the authors, PICO was developed as a result of generalizing the experience of developing other systems that solve optimization problems using the algorithms of the branch and bound method in distributed environments. In balancing this method, not all nodes are involved, only those whose loading deviates from the average by more than a given value. The algorithm implies the availability of global information about the workload of nodes in the system and is distributed by a complex mechanism based on the construction of a tree of connections between nodes.

In [45], the author proposed a load balancing algorithm for systems built using the master-slave paradigm. The balancing algorithm can be described as centralized, initiated by the coordinator, using the free sub-tasks in the pool for balancing. The algorithm collects loading information at fixed intervals. Note that the algorithm will be poorly scalable in terms of the number of masters in the system, as well as when solving a large number of simple subtasks since the number of nodes on the master will change faster than the period of the state polling. As a result, the coordinator will have incorrect download data. In addition, the absence of a local subtask pool on child nodes leads to a large overhead for the transfer of subtasks.

Specifically, because reinforcement learning offers the potential to develop optimal allocation policies without explicit model knowledge by learning from the consequences of each action, existing works on ML algorithms mainly focus on reinforcement learning [15,16]. They require neither an explicit system model nor an explicit traffic model to learn.

RL refers to a learning process, where a learning agent can learn to make appropriate decisions through interactions with an external environment [3]. Specifically, beyond the learning agent and the environment, a reinforcement learning system consists of a policy, a reward function and a value function. Let S be the set of environment states and A be the set of actions, respectively.

Another popular machine learning algorithm is the support vector machine (SVM). It has been widely applied for different areas such as pattern recognition, classification and data mining. However, SVMs are not preferred in on-line applications since the training and testing complexity of standard SVM are O(nm + m3) and (m) respectively, where n is the data size and m denotes the number of support vectors. On the other hand, some approximated methods have been proposed to reduce the complexity [19]. For example, [20] reduces the complexity to O(nd 2 max), where d max is the number of basis functions elected.

A few works on machine learning algorithms have been proposed for the resource management problem [14-19]. For admission control, [27] derived a complex rule set that can be used to identify the optimal configuration for unobserved workload based on machine learning algorithms. [19] applied RL to configure parameters automatically in multi-tier Web systems, where eight parameters at the web tier and application tier are selected to consist of the state space. For each parameter, there are three possible actions: increase, decrease and keep. The policy is based on the e-greedy method. In order to suppress the poor performance due to bad initialization, they proposed an algorithm to construct different initialization policies for different scenarios. For VM scaling, [18] proposed an iterative model training technique based on artificial neural network (ANN) to predict computing resource demand in virtual environments. [17] applied RL to train nonlinear approximators (e.g., multi-layer perceptrons) instead of the lookup table for VM horizontal scaling, where the state is defined as the request arrival rate and the action is to determine the number of servers allocated. Since the state space grows exponentially with the number of parameters in practice, the authors applied a nonlinear function approximator as an external policy to avoid poor performance that would be expected during online learning.

Recently, a few works on fuzzy control for resource management have been proposed in [15,16,28]. In [28], the admission control is conducted by fuzzy control in order to manage the QoS, where the turning parameter Maxclients in each interval is controlled by the fuzzy controller. For VM scaling, [15] attempted to capture the non-linear behaviors in VM resource usages by designing a fuzzy model estimator. The approach is divided into two steps. First, fuzzy logic-based modelling method is used to learn the system behaviors without requiring any prior knowledge. Then a predictive controller predicts the resource demand of all VMs and takes actions based on this model. [16] proposed a neural fuzzy controller for percentile-based end-to-end delay guarantee through a virtualized multi-tier server cluster, where Gaussian membership functions are first used to fuzzify the average service time, s i , and the variance of service time, sigma 2i , distribution of requests at tier i, respectively. Then a fuzzy neural network is applied for online learning at the Inference stage. In addition, an output scaling factor is introduced to further enhance performance. It is model-independent and capable of adapting control parameters through fast online learning. Compared with other supervised machine learning techniques, it does not require off-line training.

In [46] authors propose HovercRaft, a new approach by which adding nodes increases both the resilience and the performance of general-purpose state-machine replication. They achieve this through an extension of the Raft protocol that carefully eliminates CPU and I/O bottlenecks and load balances requests. Their implementation uses state-of-the-art kernel-bypass techniques, datacenter transport protocols, and in-network programmability to deliver up to 1 million operations / second for clusters of up to 9 nodes, linear speedup over unreplicated configuration for selected workloads, and a 4× speedup for the YCSBE-E benchmark running on Redis over an unreplicated deployment.

A raft is a consensus algorithm that depends on a strong leader and exposes the abstraction of a replicated log. The leader receives client requests, puts them in its log, thus guaranteeing a total order, and replicates those to the follower through an append entries request [47].

3. ANALYSIS OF METHODS FOR DEVELOPING BALANCING ALGORITHMS

From the point of view of efficiency, the algorithm is considered good if it satisfies certain requirements that are acceptable within the real-time operation. For example, if the algorithm allows the system to provide horizontal scaling, continue to work when some nodes fail, that is, be fault tolerant.

The methods for developing balancing algorithms [8,9], although they have different approaches, meet the following requirements:

1. Predictability.
2. Uniform or fair loading of system resources.
3. Scalability.

In many works, today they focus on the main balancing algorithms that have the most practical application, such algorithms as Round Robin, Weighted Round Robin, Least Queue, Load Least, Sticky session, Least Connections (Least Connections, Locality-Based Least) algorithms Connection Scheduling, Locality-Based Least Connection Scheduling with Replication Scheduling) [5-10].

In many works, today they emphasize as the main balancing algorithms that have the most practical application, such algorithms as Round Robin, Weighted Round Robin, Least Queue, Load Least, Sticky session, algorithms of the Least Connections group (Least Connections, Locality-Based Least Connection Scheduling, Locality-Based Least Connection Scheduling with Replication Scheduling) [5-10].

Using the following notation of the properties of the algorithms, we will try to give a detailed description of them:

ω_i – service intensity,

$p_i = \lambda_i/\lambda$ – the probability of sending a request to the i-th server,

$\lambda_i = \phi_i + \sum_{j=1}^{n} x_{ji}$ – the intensity of the flow of applications arriving at i-th server,

$\rho_i = \lambda_i/\omega_i$ – server i load.

1) Round Robin (RR) - the distribution of applications takes place in turn, from the first to the final cyclic, all servers receive an average of the same number of applications:

$$p_i = \frac{1}{n} = const,$$

$$\lambda_i = \frac{\lambda}{n},$$

$$T = \sum_{i=1}^{n} \frac{1}{n\omega_i - \lambda}.$$

2) Weighted Round Robin (WRR) - the distribution of applications in order, provided that

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

each server is assigned a weight coefficient depending on the performance and power of the node, and applications for them are received accordingly with the accepted rules:

$$\frac{p_i}{w_i} = const,$$

$$T = \sum_{i=1}^{n} \frac{1}{\frac{w}{w_i}\omega_i - \lambda},$$

$$w = \sum_{i=1}^{n} w_i.$$

3) Least Queue - a dynamic feedback algorithm, the application will be sent to the server with the least number of applications at the time, in this order the queue length on all servers will be the same:

$$\sum Q_i = \frac{\rho_i^2}{1-\rho_i} = const, i = \min_{j=1,n}\{j: \lambda < \lambda^j\},$$

$$p_j = \frac{i\omega_j - \omega_\Sigma^i}{i\lambda} + \frac{1}{i}, \Leftarrow \lambda > \lambda^j = \omega_\Sigma^{j-1} - (j-1)\omega_j,$$

$$p_j = 0, \Leftarrow \lambda < \lambda^j,$$

$$T = \frac{i}{\omega_\Sigma^i - \lambda}$$

4) Least Load - dynamic feedback algorithm. The application is sent to the server that is least loaded. The amount of server load can be determined, for example, by the time of connection with the server. The load on all servers is the same:

$$1 - U_i = \rho_i = const,$$

$$p_i = \frac{\omega_i}{\omega_\Sigma},$$

$$T = \frac{n}{\omega_\Sigma - \lambda}$$

5) Least Connections - a dynamic feedback algorithm, taking into account the number of connections supported by the servers at the current time. The application is sent to the server that is least loaded.

In [48] research, the authors proposed a variant where the static round-Robin algorithm is more efficient than the least-connected algorithms since it can provide more bandwidth with less CPU load and less overall latency.

Table-1 - Comparative characteristics of balancing algorithms

Name	Description	Benefits	Disadvantages
Round Robin	iterating through a circular cycle	protocol independence, implementation cost, lack of communication between servers	uniformity of resources, lack of information about congestion
Weighted Round Robin	iterating through a circular cycle, taking into account server weights	flexible load distribution, efficiency with known composition of servers in the cluster	preliminary determination of server performance and power
Least connections	requests are sent to the server with the least number of active connections.	reliability and increased fault tolerance by submitting a request to a less loaded node, cost, lack of need for data on the composition of servers	does not take into account the load of individual requests
Weighted Least Connections	when load balancing, it takes	determination of the node load and takes into	does not take into account the load of

	into account the number of active connections and the weight coefficient of servers	account the weight coefficient of servers	individual requests
Least Connections, Locality-Based Least Connection Scheduling	"LC +" principle each client server is assigned a group of cat IP client requests . are directed to the main server if it is loaded redirects the request to another server	Effective for caching proxies	does not take into account the load of individual requests, requires additional resources
Locality-Based Least Connection Scheduling with Replication Scheduling	each IP address or group of IP addresses is assigned to a group of servers the request is sent to the least loaded server	Avoid Over Replication	requires additional instructions and energy costs during peak load
	from the group if all servers from the main group are overloaded, a new server will be reserved		
Sticky session	requests are sent to the server of the cluster to which the request was sent when creating the session	protocol independence, lack of communication between servers, support in NGINX web server	The load on a specific server is not taken into account when distributing

The considered algorithms can be effectively used to balance the load on servers under certain conditions.

3.1 Comparative Analysis Of Algorithms By Hierarchy Analysis

To systematize expert knowledge and analyze the effectiveness of the considered algorithms, the hierarchy analysis method (HAM) is well suited. HAM does not provide an unambiguous correct solution, however, it allows you to interactively find the option that is best suited to solve a specific problem with given restrictions. To compare the algorithms, the following criteria were selected:

K1 - justice;
K2 - uniformity;
K3 - processing time;
K4 - response time;
K5 - scalability - the algorithm should remain operational with increasing load, changing the number and characteristics of computing nodes;
K6 - implementation complexity.

According to the method of analyzing hierarchies, it is first of all necessary to determine the relative importance of criteria based on the principle of discrimination and comparative judgments. A matrix of pairwise comparisons of criteria is given in table-2.

Table-2 - A matrix of pairwise comparisons of criteria

	K1	K2	K3	K4	K5	K6
K1	1	7	1	1	3	3
K2	1/7	1	1/5	1/7	1/3	1
K3	1	5	1	1/3	3	5
K4	1	7	3	1	3	7
K5	1/3	3	1/3	1/3	1	3
K6	1/3	1	1/5	1/7	1/3	1

To determine the relative importance of the criteria, it is necessary to calculate the estimates of the components of the eigenvector. First you need to calculate the geometric mean in each row of the table of pairwise comparisons:

$$b_i = \sqrt[n]{\prod_{k=1}^{n} a_{ik}}\,, i = 1, n$$

The calculation results are shown in summary table-3:

Table-3 – The geometric mean in each row

b_1	b_2	b_3	b_4	b_5	b_6
1,994 7574	0,332 8783	1,70997 59	2,758 9242	0,83268 32	0,383 3672

Next, we calculate the amount

$$B = \sum_{i=1}^{n} b_i\,, B = 8{,}01258616$$

We normalize the vector *b*

$$x_i = \frac{b_i}{B}$$

Results are shown in summary table-4:

Table-4 – Normalize the vector b

x_1	x_2	x_3	x_4	x_5	x_6
0,248 953	0,041 5444	0,213 4112	0,344 3238	0,10392 19	0,04784 56

We carry out a normalization check

$$\sum_{i=1}^{n} x_i = 1$$

i.e

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 0{,}248953009 + 0{,}041544423 + 0{,}21341124 + 0{,}344323808 + 0{,}1039219 + 0{,}047845621 = 1.$$

After calculating the estimates of the significance of the criteria for the distribution of second-level queries, conclusions can be drawn about their significance, these results are shown in table-5. We can conclude that the most significant contribution to the analyzed system is made by criterion K4.

Table-5 – Significance of the criteria

criteria	pleace	weight
K1	2	0,248953009
K2	6	0,041544423
K3	3	0,21341124
K4	1	0,344323808
K5	4	0,1039219
K6	5	0,047845621

Next, you need to calculate the consistency ratio to confirm the correctness of the judgments of experts. To do this, calculate the sum of the matrix elements for each column:

$$y_i = \sum_{i=1}^{n} a_{ik}$$

The calculation results are shown in table-6:

Table-6 – Sum of matrix elements for each column

y_1	y_2	y_3	y_4	y_5	y_6
3,809 5238	24	5,733 3333	2,952 381	10,666 667	20

We calculate the largest eigenvalue:

$$\lambda_{max} = \sum_{i=1}^{n} x_i y_i\,, \lambda_{max} = 6{,}251004076$$

We determine the consistency index:

$$I = \frac{\lambda_{max} - n}{n - 1}, I = 0{,}050200815$$

We calculate the consistency ratio:

$$R = \frac{I}{C} 100\% = 4\%$$

The value of the consistency ratio does not exceed 10%, which is an acceptable value and indicates the correctness of the judgments.

The algorithm described above compares the algorithms for each of the criteria. As a result of pairwise comparisons of the algorithms, the total numbers of 1-5 places occupied by each of the algorithms were obtained:

Table-6 – Significance of criteria for algorithms

Algorithm	1	2	3	4	5
A1 - RR	1	2	1	2	
A2 - LC	2	3	1		
A3 - WLC	4	2			
A4 - WRR	4	2			

Table-7 – Contribution of criteria to the value of a utility function

Algorithm	K1	K2	K3	K4	K5	K6
A1	11%	2%	17%	31%	15%	25%
A2	19%	17%	21%	19%	9%	15%
A3	31%	3%	20%	35%	10%	2%
A4	31%	3%	20%	35%	10%	2%

When developing a load balancing system, it is also necessary to take into account that none of the considered algorithms individually provides fault tolerance of the computing system as a whole. Thus, it is necessary to further develop a strategy for distributing queries in the event of a failure of computing nodes.

As part of further research, it is planned to develop a simulation model of a load balancer on a distributed computing system that includes these algorithms. The simulation model will allow you to analyze:

• the effectiveness of the application of the considered algorithms in conditions close to real ones;

• compare the effect of load growth on the performance of a computer system;

• the ability to scale the computing system by adding new nodes and increasing the resources of existing nodes.

Based on the above research, we want to modify the Least Connection algorithm and supplement it with the following modifications.

4. IMPLEMENTATION OF LOAD BALANCING ALGORITHMS ON SERVERS DEPLOYED IN A SINGLE-INSTANCE APPLICATION CLUSTER. MODIFICATION OF THE LEAST CONNECTIONS ALGORITHM.

Based on the studies, the quality of load balancing on servers is implemented by the following algorithms: Round Robin, Weighted Round Robin, Load Least, Least Connections. Theoretical and computational studies have shown the advantages and disadvantages of these load balancing algorithms on servers deployed in a single-instance application cluster.

Currently, the Least Connections balancing algorithm is relevant, which in particular is used for services deployed in a single-instance application cluster - each node has its own application instance, a distributed cache is used as a hash table, the data in which is available on all servers [11]. To improve the original algorithm, a modification is proposed where not only the number of active connections is used, but also a certain priority to the server depending on its resources (power) compared to others in the system.

The advantage of this algorithm is the ability to initialize new cluster nodes, not only from the file with the cluster settings, but also as new requests are received. If the server to which the request is being addressed is not located in the server settings, the parameters of the requested node are saved and the possibility of dynamically expanding the composition of servers (clusters) is provided, and the lowest load is assigned to this node, since the computing power of this server is unknown.

Suppose a cluster consists of N number of servers. Depending on the incoming requests, the server cluster provides different numbers of nodes: $S = \overline{1, N}$. First of all, the algorithm makes the determination of the presence of the parameters of the target server from a user request by comparing it with a hash table where the server data is stored in cluster S_i. If the identical server

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

is not located, a new node is recorded S_{i+1}. Following is the definition of the servers used:

a) if only one server $S = 1$ is used, then the current request will be sent to this server,

b) if the number is more than two $S > 2$, then the server list is sorted, depending on the number of active connections and server weights provided even during initialization.

Next is the definition and selection of the server with the least number of active connections S_i^{min} by the server power factor k_i and the request is redirected to this server.

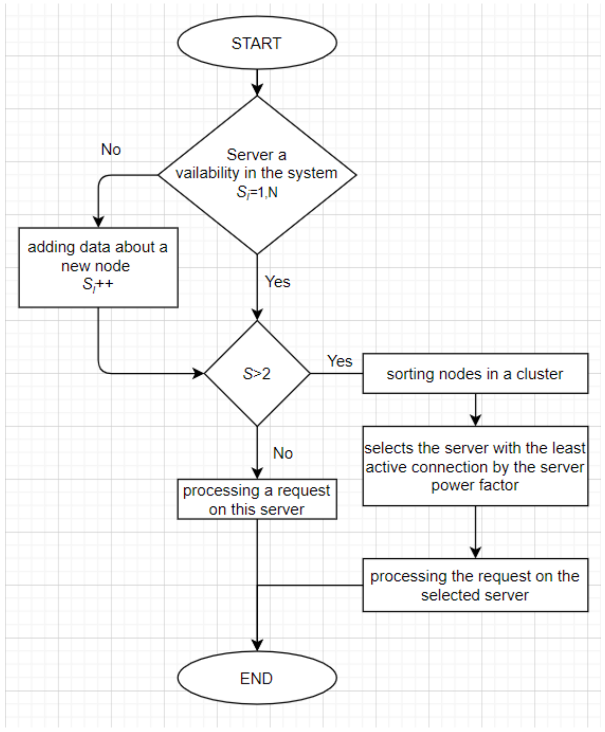
START
Server a vailability in the system S_i=1,N
No
Yes
adding data about a new node S_i++
S>2
Yes
No
sorting nodes in a cluster
selects the server with the least active connection by the server power factor
processing a request on this server
processing the request on the selected server
END

Figure-3 – Flowchart of the modified Least Connection algorithm

Thus, the modified Least Connections algorithm for single-instance application clusters contains the predominant difference with the original sample due to the labeling of weights on the servers.

The program code developed by the authors contains the sorting of cluster nodes according to the number of active connections and the load factor for each individual node, and the issuance of the corresponding address.

In addition, the modified Least Connections algorithm, due to its dynamic characteristics, can evenly distribute the load across all server nodes deployed in a single-instance application cluster.

We have a server with three nodes. Consider the following example:

Server-1 processes 3 active transactions.

Server-2 processes 20 active transactions.

Server-3 does not process any active transactions.

The load balancer selects the service using the value (K) of the following expression:

K is the number of active transactions. Requests are delivered as follows:

Server-3 receives the first request because the service does not process any active transactions. A service without an active transaction is selected first.

Server-3 receives the second and third requests because the service has the next fewest active transactions.

Server-1 receives the fourth request. When server-1 and server-3 have the same number of active transactions, Least Connections performs load balancing in cyclic mode. So server-3 gets the fifth request, server-1 gets the sixth request, server-3 gets the seventh request, and server-1 gets the eighth request, and so on.

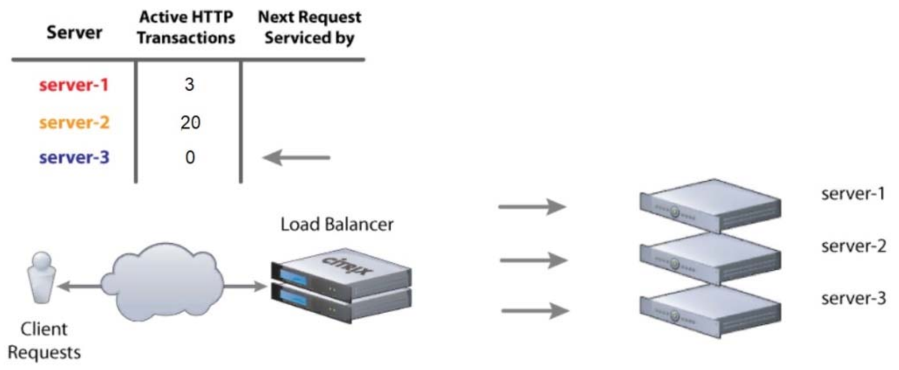

Figure-4 – Visual Representation Of The Balancer Operation

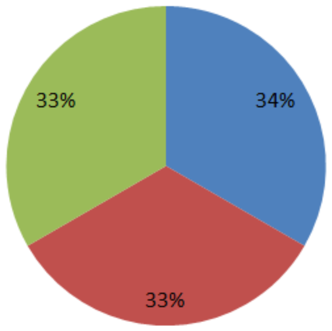

Figure-5 – The Distribution Of Requests To Across Three Server Nodes Using The Modified Least Connections Algorithm

5. CONCLUSION

Studies and implementation of balancing algorithms Round Robin, Weighted Round Robin, Load Least, Least Connections, testing of the results lead to the following conclusions:

- Least Connections algorithm is efficient enough to solve the problem of load balancing on servers deployed in a single-instance application cluster;
- received uniform load balancing in server nodes;
- the application of the proposed modifications of the algorithm makes it possible to scale applications and increases fault tolerance due to the uniform distribution of load among the nodes, which also increases the fault tolerance of the system;

In general, as a result of the research, we can conclude that when using the Least Connections algorithm, the risk of failure of the "weak" server nodes is reduced by determining the computational characteristics and introducing the server power factor. In addition, this algorithm helps to reduce unnecessary delays and, due to its dynamic characteristics, can evenly distribute the load across all server nodes.

Thus, the studies conducted by the authors of this article made it possible to analyze and solve the load balancing problem using the Least Connections algorithm in accordance with the requirements of increasing efficiency and increasing the load distribution performance.

When considering and comparing existing balancing strategies, you should: carefully analyze all the advantages and disadvantages of the selected load balancing option; take into account that simpler approaches give better results; choose the model and method of load balancing that is most suitable for a specific application; try to implement new algorithms in the form of separate software modules or products with a convenient interface for the SOFTWARE that is intended to use the user application.

REFERENCES:

[1] Hyunyoung Kil, Reeseo Cha, Wonhong Nam. Transaction history-based web service composition for uncertain QoS. *International journal of web and grid services*, vol. 12 (2016):42.

[2] Balakayeva G.T., Aidarov K.A. Research of algorithms and methods of load balancing and construction of models for queuing networks. Proceedings of the International

Conference on Computational and Applied Mathematics "VPM'17"in the framework of the Marchuk Scientific Readings, Novosibirsk, June 25 – July 14 [Electron. resource], (2017): 17–21.
[3] Goldstein B.S., Marshak MA, Mishin E.D., Sokolov N.A., Tum A.V. "Indicators of the functioning of the multiservice communication network of public use". Journal of Communication Engineering, no. 3–4 (2009): 17.
[4] Balakayeva G., Aidarov K., Simulation of load balancing algorithms based on queuing networks. Abstracts of VI Congress of the Turkic World Mathematical Society (TWMS 2017), Astana, (2017): 313.
[5] EL-Sanosi I. and Ezhilchelvan, P. Improving zookeeper atomic broadcast performance by coin tossing. In European Workshop on Performance Engineering, Springer, (2017): 249–265.
[6] Flannagen E. Michael. Syngress (2001) "Administering CISCO QoS in IP-Networks", Syngress Media, ISBN 1928994210, 9781928994213, (2001): 519.
[7] Goldstein B.S., Marshak M.A., Mishin E.D., Sokolov N.A., Tum A.V. Kontrol pokazatelei kachestva obsluzhivanya s uchotom perekhoda k seti svyazi sleduyushego pokolenya [Control of quality of service indicators, taking into account the transition to a next-generation communication network]. Tekhnika Svyazi, no 1 (2009).
[8] Andrzejak A., Arlitt M., Roila J. Bounding the Resource Savings of Utility Computing Models. Technical Report HPL-2002, Internet Systems and Storage Laboratory, HP Laboratories, (December 2002): 339.
[9] Kharchenko V, Illiashenko O, Boyarchuk A, Sklyar V, Phillips C Emerging curriculum for industry and human applications in Internet of Things. In: 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) Bucharest, Romania: Institute of Electrical and Electronics Engineers Inc., (2017): 918-922.
[10] Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., Doyle, R. P. Managing energy and server resources in hosting centers. ACM SIGOPS Operating Systems Review., no 35(5), (2001): 103.
[11] Lee, Y. C., Zomaya, A. Y. . Energy efficient utilization of resources in cloud computing systems. The Journal of Supercomputing, 60(2), (2010): 268–280.
[12] Enokido, T., Aikebaier, A., Takizawa, M. A Model for Reducing Power Consumption in Peer-to-Peer Systems. IEEE Systems Journal, 4(2),(2010): 221–229.
[13] Liu, S., Ren, S., Quan, G., Zhao, M., Ren, S. Profit Aware Load Balancing for Distributed Cloud Data Centers. 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, (2013): 611–622.
[14] Vakilinia, S., Heidarpour, B., Cheriet, M. Energy Efficient Resource Allocation in Cloud Computing Environments. IEEE Access, 4, (2016), 8544–8557.
[15] Zhang, W., Zhang, Z., Chao, H.-C. . Cooperative Fog Computing for Dealing with Big Data in the Internet of Vehicles: Architecture and Hierarchical Resource Management. IEEE Communications Magazine, 55(12), (2017): 60–67.
[16] Nagpure, M. B., Dahiwale, P., Marbate, P. An efficient dynamic resource allocation strategy for VM environment in cloud. 2015 International Conference on Pervasive Computing (ICPC) (2015).
[17] Mukherjee, M., Shu, L., Wang, D. Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges. IEEE Communications Surveys and Tutorials, (2018): 1–1.
[18] Bodenstein, C., Schryen, G., Neumann, D. Energy-aware workload management models for operation cost reduction in data centers. European Journal of Operational Research, 222(1), (2012): 157–167.
[19] Mohammad Ali, H. M., El-Gorashi, T. E. H., Lawey, A. Q., Elmirghani, J. M. H. Future Energy Efficient Data Centers With Disaggregated Servers. Journal of Lightwave Technology, 35(24), (2017): 5361–5380.
[20] Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P. P., Kolodziej, J., Balaji, P., . . . Zomaya, A. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. Computing, 98(7), (2014): 751–774.
[21] Ge, Y., Zhang, Y., Qiu, Q., Lu, Y.-H. A game theoretic resource allocation for overall energy minimization in mobile cloud computing system. Proceedings of the 2012

ACM/IEEE International Symposium on Low Power Electronics and Design - ISLPED '12., (2012): 279-284.

[22] Kliazovich, D., Arzo, S. T., Granelli, F., Bouvry, P., Khan, S. U. e-STAB: Energy-Efficient Scheduling for Cloud Computing Applications with Traffic Load Balancing. 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, (2013):7-13.

[23] Aikebaier, A., Yang, Y., Enokido, T., Takizawa, M. Energy-Efficient Computation Models for Distributed Systems. 2009 International Conference on Network-Based Information Systems (2009): 424-431.

[24] Sharma, B., Wood, T., Das, C. R. HybridMR: A Hierarchical MapReduce Scheduler for Hybrid Data Centers. 2013 IEEE 33rd International Conference on Distributed Computing Systems (2013): 102-111.

[25] Gao, Y., Guan, H., Qi, Z., Song, T., Huan, F., Liu, L. Service level agreement-based energy-efficient resource management in cloud data centers. Computers and Electrical Engineering, 40(5), (2014): 1621–1633.

[26] Gao, Y., Guan, H., Qi, Z., Song, T., Huan, F., Liu, L. Service level agreement-based energy-efficient resource management in cloud data centers. Computers and Electrical Engineering, 40(5), (2014): 1621–1633.

[27] Andrzejak A., Arlitt M., Roila J. Bounding the Resource Savings of Utility Computing Models. Technical Report HPL-2002, Internet Systems and Storage Laboratory, HP Laboratories, (December 2002): 339.

[28] Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., Doyle, R. P. Managing energy and server resources in hosting centers. ACM SIGOPS Operating Systems Review., no 35(5), (2001): 103.

[29] Mitrani I. Probabilistic Modelling. Cambridge Univ. Press. 1998

[30] Clifford E. Cummings, "Simulation and Synthesis Techniques for Asynchronous FIFO Design," SNUG 2002 (Synopsys Users Group Conference, San Jose, CA, 2002) User Papers, March 2002, Section TB2,. Also available at www.sunburst-design.com/papers.

[31] Sen, Rathindra P. Operations Research: Algorithms And Applications. Prentice-Hall of India, 2010, ISBN 81-203-3930-4.

[32] Alberto Leon-Garcia «Probability, statistics, and random processes for electrical engineering». — 3rd. — Prentice Hall, 2008. — ISBN 0-13-147122-8.

[33] Kelly, F. P. . "Networks of Queues". Advances in Applied Probability. 8 (2): 416–432, 1976, doi: 10.2307/1425912. JSTOR 1425912.

[34] F. Baskett, K.M. Chandy, R.R. Muntz, F.G. "Palacios Open, closed and mixed networks of queues with different classes of customers" Journal of the ACM (JACM) 2002.

[35] H. Kameda, E. Altman, T. Kozawa and Y. Hosokawa „Braess-like paradoxes in distributed computer systems. IEEE Trans" Automatic Contr. 2000.

[36] H. Kameda, O. Pourtallier "Paradoxes in distributed decisions on optimal load balancing for networks of homogeneous computers" Journal of the ACM (JACM), 2002.

[37] J. Palmer, I. Mitrani "Optimal tree structures for Large service networks" in 1st EuroNGI Conference on Next Generation Internet Networks (NGI 2005), IEEE Computer, 2005.

[38] A.A. Tantawi, D. Towsley "Optimal static load balancing in distributed Computer Systems" Journal of the ACM, 1985.

[39] E.D. Lazowska, J. Zahorjan, G.S. Graham, K.C. Sevcik "Quantative Computer Performance: Computer System analysis using Queuing Network Models" Prentice Hall, 1984.

[40] R.V. van Newport, J. Maassen, G. Wrzesinska, T. Kielmann and H.E. Bal "Adaptive load balancing for Divide-and-Conquer Grid Applications" in Journal of Supercomputing, 2006.

[41] R.V. van Newport, T. Kielmann and H.E. Bal "Efficient load balancing for wide-area divide-and-conquer applications" in Proceedings of the Eighth ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming, New York, USA, 2006.

[42] R.D Blumofe, C. E. Leiserson "Scheduling Multithreaded Computations by Work Stealing" In: 35th Annual Symposium on Foundations of Computer Science (FOCS

'94), Santa Fe, New Mexico, 1994: pp. 356-368.

[43] J. Eckstein, C.A. Phillips and W.E. Hart, "PICO: An Object-Oriented Framework for Parallel Branch and Bound", RUTCOR Research Report 40-2000, Rutgers University, Piscataway, 2000.

[44] Robert C. Martin "Agile Software Development: Principles, Patterns and Practices" Pearson Education, 2002.

[45] Kento Aida, Wataru Natsume, Yoshiaki Futakata "Distributed Computing with Hierarchical Master-worker Paradigm for Parallel Branch and Bound Algorithm," Proc. 3rd IEEE / ACM International Symposium on Cluster Computing and the Grid, 2003.

[46] M Kogias, E. Bugnion “HovercRaft: Achieving Scalability and Fault-tolerance for microsecond-scale Datacenter Services” EuroSys 2020, Heraklion, Crete, Greece, Avril 27-30, 2020, DOI 10.1145/3342195.3387545.

[47] D. Ongaro, J.K. Ousterhout In Search of an Understandable Consensus Algorithm. In Proceedings of the 2014 USENIX Annual Technical Conference (ATC), 2014: pp. 305–319.

[48] H. Triangga, I. Faisal, I. Lubis “Analisis Perbandingan Algoritma Static Round-Robin dengan Least-Connection Terhadap Efisiensi Load Balancing pada Load Balancer Haproxy” Nfotekjar: Jurnal Nasional Informatika Dan Teknologi Jaringan- Vol. 4 No. 1 (2019).