Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2021.020304
ISSN 2790-0916 Vol. 2, Issue 3: 32-40

Implementation Model of Dynamic Load Balancing in Distributed System Based on Consistent Hash Algorithm

Suzanana Ahmad*

Adamson University, Philippines

**corresponding author*

Keywords: Hash Algorithm, Distributed System, Dynamic Load Balancing, Load Prediction

Abstract: In the new stage of network computing, the research and application of distributed systems are becoming more and more extensive. Dynamic load balancing is the core technology of network computing. How to improve the performance of dynamic load balancing has always been a research hotspot of network researchers. The purpose of this paper is to implement a dynamic load balancing model based on a consistent hashing algorithm in a distributed system, and propose a consistent hashing algorithm based on virtual nodes. Firstly, the distributed system and distributed object technology are discussed, and then the load balancing algorithms and models are studied in each chapter. By introducing the concepts of system resource utilization and virtual nodes, the performance and adaptability of the consistent load-constrained hashing algorithm are optimized. To see the prediction accuracy more clearly, randomly select the predicted state of one of the bins. The success rate of the one-step prediction model shows that the success rate of the algorithm's prediction model is generally between 0.8 and 0.9, and the hit rate is relatively stable. The success rate is slightly lower than the single-step success rate, generally between 0.3 and 0.6.

1. Introduction

With the advancement of computing, distributed systems have been widely used and appreciated, but due to the random arrival of tasks, various system load differences occur in distributed systems. To overcome this situation, some load balancing methods are provided. Load balancing technology can distribute server requests to all nodes of the entire network system, so that the load of each server is roughly the same, so it can quickly adapt to the requirements of the server, thereby improving the efficiency of the entire system [1]. At present, almost all large-scale software systems use distributed systems, which have the advantages of convenient use, easy maintenance, and

resource sharing, so they have been widely used and studied [2].

At present, domestic and foreign research work on the load balancing of distributed systems has been carried out and many achievements have been obtained. Mendelson G proposed AnchorHash, a scalable and fully consistent hashing algorithm. AnchorHash achieves high key lookup rate, low memory footprint, and low update time. They formally established strong theoretical guarantees and proposed a high-level implementation with a memory footprint of only a few bytes per resource. Furthermore, evaluations show that AnchorHash scales to 100 million resources on a single core, while still achieving a key lookup rate of over 15 million keys per second [3]. Bajestan EE developed a numerical model to track the load curve and instantaneous changes in ambient temperature and predict the instantaneous hot spot temperature value of the transformer under dynamic load. This thermal model was then used to explore the ability of the studied vegetable oils to cool transformers compared to conventional transformer oils. The actual ambient temperature and load curves, as well as the thermal characteristics of the oil and the characteristics of the transformer, are used as input to the model. The aging rate of transformers in the presence of vegetable oil was also compared with conventional transformer oil. The results show that vegetable oils have better cooling properties, with a 3℃ lower hot spot temperature found compared to petroleum-based oils. Furthermore, the model predicts that the lifetime of the transformer insulation system will be significantly extended when the proposed vegetable oil is used. The results of this study suggest a sustainable way to reuse waste from renewable sources as an alternative insulating liquid for cooling high heat flux electrical/electronic devices [4]. Kundu S proposes a decision-making framework that helps building owners/operators effectively prioritise load shedding on services under uncertainty while minimising any adverse impact on occupants. The proposed framework uses a stochastic (Markov) model to represent the probabilistic behavior of device usage from power consumption data, and a load prioritization algorithm that dynamically ranks building loads using a stochastic multi-criteria decision algorithm. In the context of load shedding as a grid service, the proposed load prioritization framework is illustrated through numerical simulations in a residential building use case, including plug loads, air conditioners, and plug-in electric vehicle chargers. Appropriate metrics are proposed to evaluate the closed-loop performance of the proposed prioritization algorithm under various scenarios and design choices. The scalability of the proposed algorithm is established through computational analysis, while time series graphs are used to visually explain ranking choices [5]. To sum up, in a distributed system, in order to reduce the system simulation time and improve the system automation performance, dynamic load balancing needs to be introduced.

In-depth study of the power load balancing of distributed systems can effectively improve the performance of the distributed computing system simulation platform, thereby promoting the improvement of distributed network system simulation and providing real-time distributed transmission. System simulation, traffic flow distribution prediction information, traffic distribution information early warning, etc. Provides basic support for useful applications in areas such as urban traffic management. Considering the enormous processing power provided by distributed systems, it is not economically feasible for tens or even hundreds of microcomputers or distributed microcomputers to perform the work required by an expensive centralized system. Research on load forecasting and estimation algorithms in distributed systems has special interest and broad market prospects. As microelectronics advances, the price of computers and desktops has dropped dramatically while performance has gotten better.

2. Research on Dynamic Load Balancing in Distributed System Based on Consistent Hash Algorithm

2.1. Features of Distributed Systems

(1) Resource sharing

The resources here can be anything, such as printers, computers, disks, files, data, storage devices, websites, and networks [6-7].

(2) High system reliability

The distributed nature of a power distribution system makes it imply a kind of fault tolerance, if your work session makes the entire system at least partially available and functioning to a certain extent; while on one machine, when a computer fails, The entire system will no longer work [8-9].

(3) Parallelism

Due to the limited processing power of computer systems, due to many factors (such as time difference), the availability of each computer is not equal. Distributed systems can make multiple computers in the same network work together, process in parallel, improve the efficiency of the entire system, and use load balancing algorithms to balance the load of each computer and improve the performance of the entire system [10-11].

2.2. Consistent Hash Algorithm

Consistent hashing is a traditional hashing algorithm, which is widely used in the research of different scenarios due to its advantages in performance and scalability. In the scenario based on load balancing, the consistent hash algorithm extracts characteristic strings for identification from the nodes of the system, such as the node's IP address, name, uuid (Universal Unique Identifier), etc. Mapped to a hash ring of length 232-1 [12-13]. In view of the fact that the consistent hash algorithm needs to avoid producing the same result as much as possible when operating on different strings and reduce the repetition of the final hash map, this paper will use a perfect hash algorithm (Perfect Hashing) for the system as the Basis for calculating the hash value [14-15].

2.3. Dynamic Load Balancing

In practical distributed systems, many applications are complex, and it is not easy to predetermine the computational and communication costs of tasks, and even the tasks of some applications will vary as the process progresses [16-17]. Therefore, the classification methods described above may not be able to efficiently complete the load balancing task, so we propose a dynamic load balancing strategy. A dynamic load balancer evaluates system status as services arrive, collects real-time information on each node, and dynamically distributes requested services. This approach can better allocate resources and improve overall system performance. The implementation of the dynamic load balancing algorithm is more complicated, and the subsequent chapters will introduce the collaborative implementation process in detail [18].

3. Model and Research of Dynamic Load Balancing in Distributed System Based on Consistent Hash Algorithm

3.1. Implementation Model of Dynamic Load Balancing

The chain model in dynamic load balancing is defined as:

$$Linear\_list = (D, R) \tag{1}$$

in: $D = \{a_i | a_i \in D_0, I = 1,2,\Lambda\ ,n, n \geq 0\}$

$$R = \{N\}, N = \{< a_{i-1}, a_i > | a_{i-1}, a_i \in D_0, i = 2,3,\Lambda\ ,n\}$$

D_0 is a data object. A relation N is a set of ordered pairs, which represent the adjacent relation between nodes in a linear model.

3.2. Collection of Load Balancing Information

(1) Load information parameters

The first task of realizing load balancing is to obtain the load information of the server, which is usually realized by the load monitoring module. The previous chapters have introduced the commonly used parameters to identify the server load. Traditional load balancing algorithms usually only select the CPU utilization as the system load parameter. However, in some applications, it is difficult to determine the load of a system with only one parameter.

(2) OID information of parameters

How to obtain parameters is a very important issue. Usually, the local load information can be monitored by some software and commands. There are also some specific programs for monitoring the remote server load information. However, these specific application software have certain limitations and lack applicability to the load balancing framework proposed in this subject. The information is organized in a tree structure, and each node in the tree represents a unique monitoring parameter. Usually, OID is used to identify these parameters. Different operating systems have different OIDs of objects in the SNMP management information base.

3.3 Relevant Calculations of Nodes

When the execution node of the system is put into use for the first time, the system administrator will set an ideal initial load value (Start Load) for it. This value is obtained according to the hardware configuration of each node and is recorded as SLi (i= 1,2....,n). Usually, the higher the hardware configuration of the node, the larger the default value. The dynamic load value calculation formula is expressed as:

$$DL_i = W_1 \times L_{cpu}(i) + W_2 \times L_{io}(i) + W_3 \times L_{memory}(i) \tag{2}$$

Among them, Wi is the scale parameter set by the system for the node, Wi $\in$ [0, 1]. According to the initial load value and the maximum load value of the node, the current load value Li of the node can be calculated, and the formula is as follows:

$$L_i = SL_i + K \times (DL_i - SL_i) \tag{3}$$

Among them, k is the coefficient, and the current load value Li of the node is a measure of the

load status of the node in the system, which is used as the basis for locating the light-loaded node when the heavy-loaded node transfers tasks. Basis for receiving node table.

4. Analysis and Research of Dynamic Load Balancing in Distributed System Based on Consistent Hash Algorithm

4.1. Prediction Accuracy

In order to more clearly see the prediction accuracy, the prediction of one of the buckets is randomly selected, and the concept of hit rate is introduced. The value of hit rate is between 0 and 1. The higher the hit rate, the more accurate the prediction. The single-step prediction model hit rate is shown in Table 1.

Table 1. One-step prediction model hit rate

Iteration steps	Consistent Hash Algorithm	Catch The Wind
2	0.27	0.06
4	0.95	0.79
6	0.86	0.86
8	0.73	0
10	0.82	0

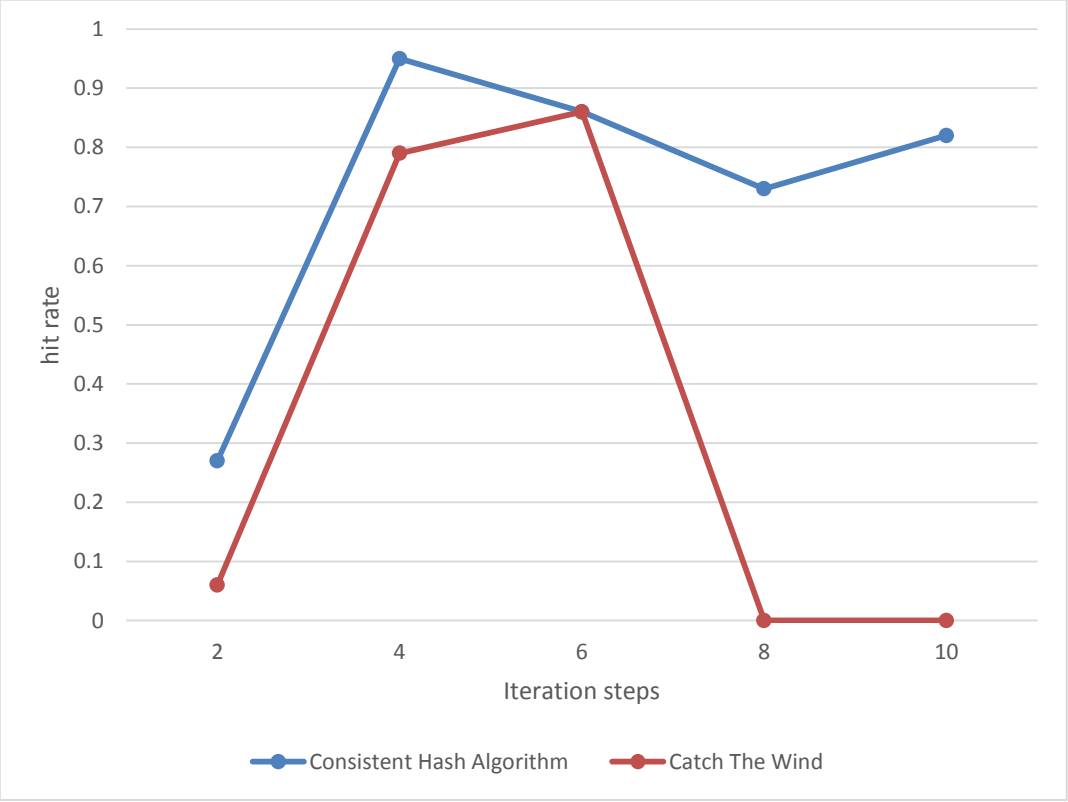

Figure 1. Comparison of hit rates of single-step prediction models

Figure 1 shows the experimental results of the single-step prediction hit rate of a single bucket randomly selected. The horizontal axis represents the number of iteration steps, and the vertical axis represents the hit rate. The hit rate of the prediction model in this system is generally between 0.8 and 0.9, and the hit rate is relatively stable; while the prediction method in the Catch The Wind system only has a high hit rate in a few super steps, and the hit rate fluctuates greatly.

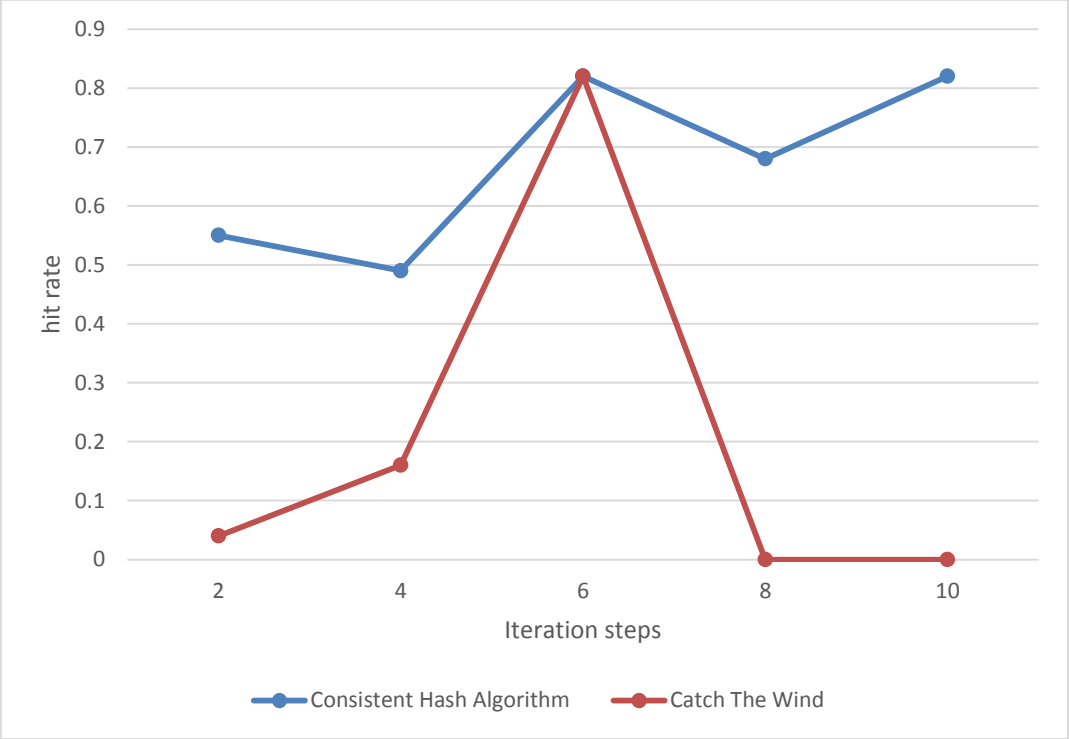

Figure 2. Two-step prediction model hit rate

Figure 2 is the experimental result of the two-step prediction hit rate of a single bucket randomly selected. The horizontal axis represents the number of iteration steps, and the vertical axis represents the hit rate. The multi-step hit rate of the prediction model in this system is slightly inferior to the single-step hit rate, generally between 0.3 and 0.6, but it is still much better than the prediction method of the Catch The Wind system.

4.2. Data Migration Strategy Evaluation Test

(1) Scale of migrated data

For three different data sets, the data migration scale of three migration strategies was tested based on the data migration of a short board task in the same super step when the load is unbalanced in the system, as shown in Table 2.

Table 2. Migration data scale for three strategies

	Bucket Migration	Vertex migration	Edge migration
Dataset 1	3050	2900	1500
Dataset 2	10000	7800	7000
Dataset 3	17800	14900	11900

Table 2 presents the comparative experiments on three real datasets of different sizes, comparing the scale of migration data for bucket migration, vertex migration and edge migration, respectively. Among them, the ordinate represents the amount of load migration data, including the number of vertices and the number of outgoing edges, and the abscissa represents the scale of the processed dataset. It can be seen from Table 3 that the bucket migration strategy needs to migrate a large number of vertices, the data volume of the vertex migration strategy and the edge migration strategy is similar, and the data size of the edge migration strategy is the smallest. This is because the bucket migration strategy is to migrate all the data of the entire bucket to the new task, the vertex migration strategy is to migrate vertices and their outgoing edges as needed, and the edge migration strategy only migrates part of the outgoing edges, so the edge migration strategy will The data size is smaller than the vertex migration strategy.

(2) Migration time cost

The cost of data migration is divided into two aspects, one is the cost of data migration, and the other is the cost of migration roadblock synchronization. For three different data sets, the data migration of a short board task in the system in the same super step is the standard, and the migration time cost of the three migration strategies is tested, as shown in Figure 3.

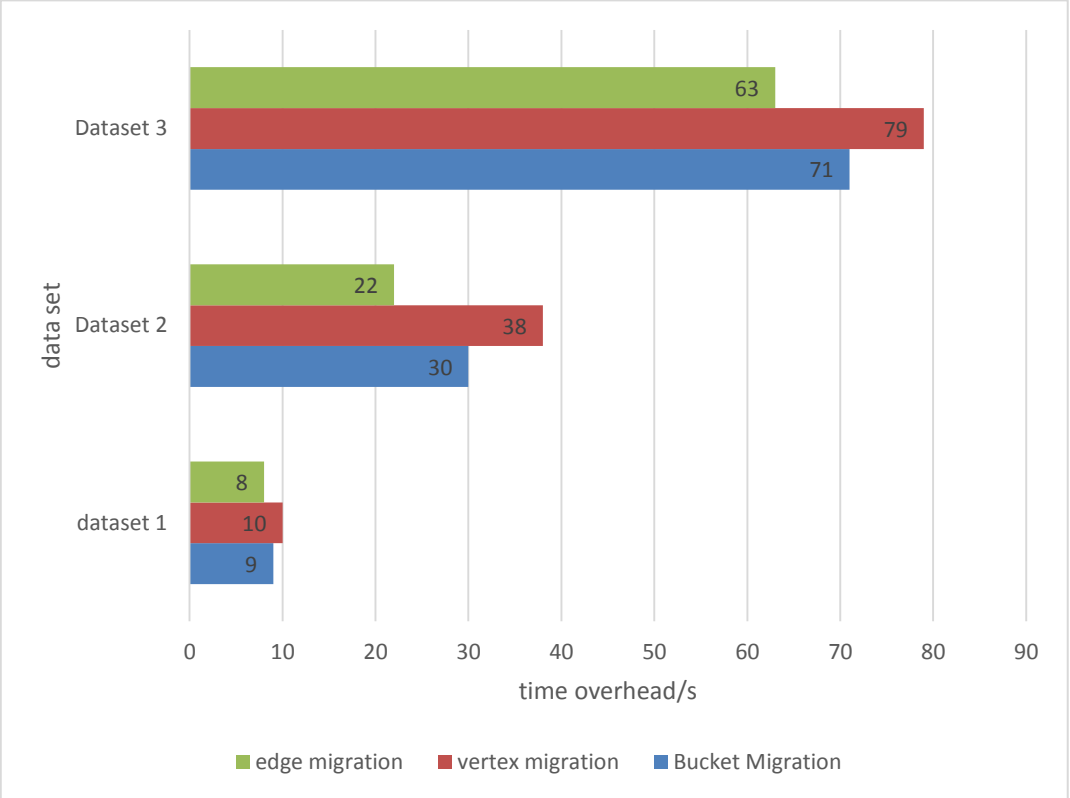

Figure 3. Migration costs for the three strategies

Figure 3 presents the comparative experiments on three real datasets of different sizes, comparing the migration costs of bucket migration, vertex migration and edge migration, respectively. It can be seen from Figure 3 that the time cost of the vertex migration strategy is the largest, the time cost of the bucket migration strategy is second, and the edge migration strategy has the smallest migration cost. This is because, although the scale of data migrated in the process of

vertex migration is small, the routing table that needs to be maintained for item-point migration is expensive; the routing table in the bucket migration strategy is small and easy to maintain; the edge migration strategy does not require a routing table.

5.Conclusion

This paper tests the proposed dynamic load balancing mechanism. Firstly, the software and hardware environment and deployment scheme of the system are introduced. The prediction model and three different transfer strategies were then tested. Experiments show that the mechanism achieves the expected effect. Among them, the average accuracy of the prediction model can reach more than 75%, which is more than 35% higher than that in the Catch The Wind system. The rapid development of grid computing provides a wider application stage for our research. How to extend the workstation cluster environment we study to grid computing environment, and study the model, mechanism and strategy of grid computing resource management and task scheduling, which will also be the focus of our future research.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Taghizadeh S R, Bobarshad H, Elbiaze H. CLRPL: Context-Aware and Load Balancing RPL for IoT Networks under Heavy and Highly Dynamic load. IEEE Access, 2018, PP(99):1-1. https://doi.org/10.1109/ACCESS.2018.2817128

[2] Ginting B M, Mundani R P. Parallel Flood Simulations for Wet-Dry Problems Using Dynamic Load Balancing Concept. Journal of Computing in Civil Engineering, 2019, 33(3):04019013.1-04019013.18.

[3] Mendelson G, Vargaftik S, Barabash K, et al. AnchorHash: A Scalable Consistent Hash. IEEE/ACM Transactions on Networking, 2020, PP(99):1-12.

[4] Bajestan E E, Tiznobaik H, Gheorghe P, et al. Thermal Behavior of Power Transformers Filled With Waste Vegetable Oil-Based Biodiesel Under Dynamic Load. Journal of Energy Resources Technology, 2021, 143(9):1-9. https://doi.org/10.1115/1.4049583

[5] Kundu S, Bhattacharya A, Chandan V, et al. A Stochastic Multi-Criteria Decision Making Algorithm for Dynamic Load Prioritization in Grid-Interactive Efficient Buildings. ASME Letters in Dynamic Systems and Control, 2021, 1(3):1-24. https://doi.org/10.1115/1.4050124

[6] Tola A F, Sedara A M, Olatunde O B, et al. Effect of soil moisture content, dynamic load and wheel slippage in measuring traction in an indoor traction bed. Poljoprivredna Tehnika, 2021,

46(1):22-30. https://doi.org/10.5937/PoljTeh2101022T

[7] Kumar A, Shankar G. Quasi-oppositional harmony search algorithm based optimal dynamic load frequency control of a hybrid tidal–diesel power generation system. IET Generation, Transmission & Distribution, 2018, 12(5):1099-1108. https://doi.org/10.1049/iet-gtd.2017.1115

[8] Thaiyalnayaki S, Sasikala J, Ponraj R. Indexing Near-Duplicate Images In Web Search Using Minhash Algorithm. Materials Today: Proceedings, 2018, 5(1):1943-1949.

[9] Reingold O, Vardi S. expanding the boundaries of local computation algorithms. Journal ofuter & System Sciences, 2019, 82(7):1180-1200. https://doi.org/10.1016/j.jcss.2016.05.007

[10] Katsaros D. Distributed ledger technology: the science of the blockchain (2nd ed.). Computing reviews, 2018, 59(11):596-597.

[11] Korndrfer J, Eleliemy A, Mohammed A, et al. LB4OMP: A Dynamic Load Balancing Library for Multithreaded Applications. IEEE Transactions on Parallel and Distributed Systems, 2021, PP(99):1-1.

[12] Nayeeif A A, Ali H, Hamdan Z K, et al. Effect of diameter on the critical speed of a composite shaft under dynamic load. Journal of Mechanical Engineering Research and Developments, 2021, 44(6):169-176.

[13] Mahdavian A, Ghadimi A A, Bayat M. Microgrid small‐signal stability analysis considering dynamic load model. IET Renewable Power Generation, 2021, 15(13):2799-2813. https://doi.org/10.1049/rpg2.12203

[14] Almoosi Y, Mcconnell J, Oukaili N. Evaluation of the Variation in Dynamic Load Factor Throughout a Highly Skewed Steel I-Girder Bridge. Engineering, Technology and Applied Science Research, 2021, 11(3):7079-7087. https://doi.org/10.48084/etasr.4106

[15] Fomin O, Gerlici J, Lovska A, et al. Creating mathematical model of the bearing structure dynamic load of the flat wagon from round pipes in the main operational modes. Transportation Research Procedia, 2021, 55(46):875-881.

[16] Korndrfer J, Eleliemy A, Mohammed A, et al. LB4OMP: A Dynamic Load Balancing Library for Multithreaded Applications. IEEE Transactions on Parallel and Distributed Systems, 2021, PP(99):1-1.

[17] Giordano A, Rango A D, Rongo R, et al. Dynamic Load Balancing in Parallel Execution of Cellular Automata. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(2):470-484.

[18] Chavarro-Barrera L, Perez-Londono S, Mora-Florez J. An Adaptive Approach for Dynamic Load Modeling in Microgrids. IEEE Transactions on Smart Grid, 2021, PP(99):1-1. https://doi.org/10.1109/TSG.2021.3064046