DOI: https://doi.org/10.15276/aait.07.2024.9
UDC 004.75

A model and method for enhancing the efficiency of processing operation queues at maximum server equipment load

Sergii S. Surkov[1)]
ORCID: http://orcid.org/0000-0001-9224-7526; k1x0r@ukr.net. Scopus Author ID: 57103247200
Oleksandr M. Martynyuk[1)]
ORCID: http://orcid.org/0000-0003-1461-2000; anmartynyuk@ukr.net. Scopus Author ID: 57103391900
Oleksandr V. Drozd[1)]
ORCID: http://orcid.org/0000-0003-2191-6758; drozd@ukr.net. Scopus Author ID: 55388226700
Myroslav O. Drozd[1)]
ORCID: https://orcid.org/0000-0003-0770-6295; myroslav.drozd@opu.ua. Scopus Author ID: 56667174000
[1)] Odessa Polytechnic National University, 1, Shevchenko Ave. Odessa, 65044, Ukraine

ABSTRACT

Existing solutions aimed at preventing excessive parallelization, reducing processing times, and forecasting load accuracy in operation queues were analyzed. Subsequently, a new model and method designed to enhance the efficiency of processing operation queues, particularly when operating at maximum server equipment load, were evaluated against traditional methods. These methods, including sequential execution, maximal, and constrained parallelism, were assessed. The new method uses two strategies: 'first-in, first-out', useful because parallelism does not guarantee sequential order of results, and maximizing equipment utilization for optimal performance. Utilizing the new adaptive monitoring model based on linear regression, the new method achieves operation execution times comparable to sequential execution and total execution times similar to those achieved with constrained parallelism. Constrained parallelism, although it reduces resource conflicts compared to maximal parallelism, still increases the processing time of each operation, emphasizing the importance of balancing the number of parallel operations with the available system resources. We estimated the complexity of the new model using asymptotic complexity and analyzed it with multi-server queueing models under conditions of both limited and unlimited parallelism. Two series of experiments were carried out for the comparative analysis of a new method for managing loads in operation queues versus traditional approaches. Additionally, the potential for resource flexibility in load management within digital infrastructures is highlighted.

Keywords: Load management; operation queues; digital infrastructure; data processing; parallelism; resource optimization; equipment load forecasting; processing efficiency; operation distribution methods; system performance; minimization of processing time

For citation: **Surkov S. S., Martynyuk O. M., Drozd O. V., Drozd M. O. "A model and method for enhancing the efficiency of processing operation queues at maximum server equipment load".** ***Applied Aspects of Information Technology.*** **2024; Vol. 7 No. 2: 125–134. DOI: https://doi.org/10.15276/aait.07.2024.9**

INTRODUCTION

At the core of modern digital infrastructure, servers and their collective counterparts, server clusters, play a pivotal role. They maintain continuous operations across various online platforms, whether handling online financial transactions, enabling e-commerce, or supporting social interactions on networks, their performance is vital for ensuring a seamless user experience [1].

Cloud providers like Amazon Web Services [2] and Google Cloud [3] actively use mechanisms to ensure server resilience during peak loads. A key method is dynamic resource scaling [4]. If the load on the server environment reaches a threshold value, such as 80 %, the system automatically creates and activates additional virtual machine instances, placing them on different physical servers to balance incoming traffic.

With cloud platforms operational, the likelihood of service interruptions is significantly reduced, they maintain consistent and efficient service performance regardless of the workload.

Modern servers [5] are equipped with specialized tools for executing resource-intensive tasks, such as graphics rendering, video processing [6, 7], [8], large data analysis, and artificial intelligence-based [9] tasks. When faced with high loads or the need to process a vast amount of data, the challenge of handling a large number of accumulated operations can arise. Operation queue management systems are used to optimize this process and ensure effective execution of operations that can run in parallel [10] and are independent of each other.

Optimal operation execution time not only accelerates data processing but also ensures a comfortable user interaction. When equipment is underutilized, server response time decreases, improving system performance. Failing to utilize equipment to its full potential effectively equates to resource wastage. The seamless operation of any digital platform heavily relies on the strategic allocation of resources and maintaining servers at optimal performance levels.

For stable operation queues, systematic work on forecasting and regulating peak loads is necessary. Running applications on an overloaded server can lead to reduced quality of service (QoS). Adapting queues to diverse operational environments demands not only careful handling but also ongoing adjustments as conditions evolve. To ensure that all aspects of computer engineering, particularly queue management, function without disruption, regular research and study are essential.

1. LITERATURE REVIEW AND PROBLEM STATEMENT

Machine learning methods can achieve high accuracy in forecasting load levels for new operations in the queue. However, their significant requirement for computing resources is a notable drawback.

The migration of virtual machines between physical servers, activated when these servers become overloaded [11], necessitates the identification of an optimal server for resource allocation. The main benefit is handling various types of operations using multiple VMs. However, it has limitations such as the uneven distribution of resources among virtual machines and the need to transfer VMs between servers when their resource demands exceed the physical capacities available. Furthermore, transferring VMs from one physical server to another requires significant central processor unit (CPU), graphical processor unit (GPU), disk, and network resources.

A discrete-time queuing model [12] for a production system bottleneck addresses the issue of energy waste in manufacturing systems. An analysis of production system performance was conducted by varying arrival intensity and service rates to reduce energy consumption while maintaining production efficiency. Emphasis is placed on assembly lines where different operation types can be executed on specific equipment.

A model for calculating local ratings [13] and controlling price levels based on the current service system rating was analyzed through a multidimensional Markov chain. In this approach, the global rating system is problematic because it is formed externally to the provider, and the precise mechanism of its formation remains largely unknown. Additionally, the formation of prices for operations and server ratings is problematic as it merges different types of loads into a single metric for both the server and operations, and prices may be calculated during server overload.

The data processing on a server encompasses data transfer, storage, computation, and authentication. The adoption of chunking authentication techniques facilitates a marked increase in the speed of data verification while concurrently preventing system overload. In the context of data transfer, the HyperText Transport Protocol (HTTP/1.0) protocol processes each request as a separate connection. With the introduction of HTTP/1.1[14], it became possible to handle multiple requests within a single connection. However, with the launch of HTTP/2.0 [15, 16], an innovative mechanism was implemented – “multiple data streams within a single connection”, which enhances performance and optimizes data transmission.

In our scenario, all servers are capable of handling operations of various types [17, 18], [19], which are typically not small in scale. Additionally, there is no need for separate environments for different types of operations. Thus, we need to employ a strategy of fully loaded servers, maximize server utilization, and minimize energy consumption.

Despite the availability of existing methods and approaches, there are issues that remain regarding the non-overloading of equipment and the resources required for the implementation of the models and methods themselves. The unconstrained parallelism and heavy methods lead to increased computational resource usage and longer processing times for operations in the queue.

Thus, there is a need to develop new models and methods that are lightweight and do not cause excessive parallelism on the equipment.

2. PURPOSE AND OBJECTIVES OF THE STUDY

Our goal is to reduce the processing time of operations in the queue. To achieve this, we need to mitigate excessive parallelization by developing a new model which improves the accuracy of forecasting equipment load and a method that manages the operations.

To achieve the goal, the following tasks are established:

1. Develop a model and method for the dynamic regulation of operations in the operation queue.

2. Evaluate the complexity of the new model using Big-O notation and assess the new method using the known M/M/c model. M/M/c is a queueing model with a Poisson arrival process, exponential service times, and *c* parallel servers. Big-O notation describes the upper limit of an algorithm's complexity in terms of input size.

3. Conduct an experiment to practically evaluate the new model and method.

The main contribution of this study is the development of these new approaches, which achieve the goal and improve system resilience and adaptability during periods of high demand.

3. METHODOLOGY OF RESEARCH

An analytical review of existing load management solutions utilizing operational queues has been performed. Building on the analytical review, a new model and method were developed using an advanced adaptive monitoring based on linear regression.

The new and model and method utilize system parameters such as CPU and GPU load during the collection of operation statistics, as well as in determining whether to execute a new operation. Persistent connections, such as HTTP/2 and HTTP/3, are used to queue operations. These connections remain active during long waits in the event of queue overload, sending ping frames as heartbeats to maintain the connection when a new operation enters the queue.

Additionally, a stack of operation results with buffered outputs is employed in case of using a FIFO strategy. After the operation is completed, it is removed from the stack. Furthermore, if a disconnect occurs, the result of the operation is saved locally and returned to the client upon reconnection.

To build the model, data arrays are saved after each operation execution. Median values of sorted arrays are used to trim extremely large and small data points. The arithmetic means is calculated, which serves as a linear regression. The root mean square deviation is computed to determine the accuracy of the model for a specific operation.

The new model and method were assessed for complexity using Big-O notation, where each calculation step of the model is analyzed, and for execution using the known M/M/c model to evaluate the new method.

An experiment was conducted to compare various strategies: single-task execution, maximum parallelization, and optimized resource utilization. Key indicators such as resource utilization, execution times and total time were analyzed.

4. MODEL OF DYNAMIC REGULATION OF OPERATIONS IN THE OPERATION QUEUE

In modern digital infrastructure, the number of interactive processes and data streams is constantly increasing. Maintaining system stability and reliability becomes increasingly challenging when managing loads that fluctuate significantly. For optimal resource allocation, it’s crucial to have instruments that allow us to predict system load one step ahead.

A model based on linear regression, aimed at adaptive real-time system monitoring, has been further developed. The model differs from the existing ones by binding each operation with a key. The key is generated using arguments such as the type of operation, its complexity, associated metadata, and estimated data size, through a function defined by the user of the system.

Each operation in the model is associated with a key and consists of an array of values, each corresponding to the load level of a specific type, such as central processing unit (CPU) loads, graphics processing unit (GPU) loads, and network data transmission speeds, among others. To ensure the accuracy and relevance of the information, the array of values is cleared after each computational cycle, thus eliminating the possibility of their reuse in subsequent calculations.

To achieve the most accurate and reliable results, the optimal condition occurs when only a single operation is processed in the queue at any given time. This condition may be established in a controlled environment at the initial stage of the study or manifest during standard operation.

The process initiates by calculating the expected load level (μ) for each individual operation. Subsequently, based on the collected information, the aggregate expected load level for the entire set of operations is calculated.

The model assumes maintaining the load level μ at a relatively stable level, considering that most operations are characterized by a constant load for most of the time. This is achieved through the definition of function *f*, which links the key to a calculated set of pairs.

Each pair consists of an expected load level and its type, as shown in formula (1):

$$f: key \rightarrow \{(t_1, \mu_1), (t_2, \mu_2), \ldots, (t_n, \mu_n)\} \quad (1)$$

where *key* is a string identifying the set of loads; μ_i is expected load level; t_i – type of the load level, which is used in the method; *n* is the total number of pairs in the set associated with a specific key.

The model is formed through the calculation and verification of μ_i for each key. The expected load levels μ_i are calculated and verified to ensure their correspondence with the actual parameters of system operation.

The input data for the model consist of pairs in the following format for each key, as shown in formula (2):

$$\{(t_1, A_1), (t_2, A_2), \ldots, (t_n, A_n)\} \quad (2)$$

where A_i is represents the array of raw load levels collected from the execution of operations for each type t_i.

Each computational cycle involves calculating μ_i for each A_i. For clarity in this text, A_i will be referred to as A, as μ.

To minimize the impact of extremely large or small values and to select the most suitable data set, the median of the sorted data array is used.

The median calculation is performed according to formula (3), which contributes to the stability of the result against deviations:

$$M = \text{QuickSelect}(A, k) \quad (3)$$

where M is the median of the array A; k is the position of the median in the array.

To determine the expected system load, which is represented by a line with the minimum mean square error, data filtering is first performed, excluding significantly high and low values. This is achieved by defining μ as the arithmetic mean of the filtered data set.

Data filtering is carried out according to formula (4):

$$A_{\text{filtered}} = \{a \in \text{A} \mid (1-\rho) \cdot M \leq a \leq (1+\rho) \cdot M\} \quad (4)$$

where ρ is the proportional range around the median, expressed as a percentage.

Then, μ is calculated, which represents the mean value of the array A_{filtered}.

This mean value is used to construct a linear regression line, reflecting the overall behavior of the data set, as shown in formula (5):

$$\mu = \text{mean}(A_{filtered}) \quad (5)$$

To verify the calculated μ value associated with *f*: key and type of load, the mean square error is compared to the product of the maximum permissible relative error and the expected load level. This ensures that the calculated μ is sufficiently precise for predicting future loads.

This comparison is carried out according to formula (6):

$$\text{v}(A) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mu - \hat{a_i})^2} < \mu \cdot \varepsilon \quad (6)$$

where A is the source array; $\hat{a_1}$ is an element within the array A; ε is the user-defined maximum relative standard deviation error applicable across the entire dataset, expressed as a percentage.

If the computed load level closely aligns with the anticipated linear trend, no adjustment of the model key is necessary.

Based on these metrics, high accuracy and responsiveness in identifying high-load states are achieved, leading to reduced delays and accelerated response times, optimizing overall system performance.

5. METHOD OF DYNAMIC REGULATION OF OPERATIONS IN THE OPERATION QUEUE

To improve efficiency, a specialized method of resource distribution has been proposed to dynamically regulate operations within the queue. The new method uses two sub-strategies: first-in first-out (FIFO), which is particularly useful as parallelism does not guarantee the sequential order of results, and maximizing equipment utilization to optimize performance. In the FIFO strategy, operations maintain maximal equipment utilization through parallel processing. To facilitate this, an internal mechanism caches the results of completed operations, storing them until they can be returned in FIFO order.

The new method consisting of the following steps:

1. Extracting an array of values using a generated key (see formula 1) in the context of the model for adaptive monitoring and managing the load of operation queues in real-time.
2. Assessing system load parameters based on monitoring results;
3. Summing current system load indicators with the array extracted from the model and comparing with maximum possible values.
4. Temporarily suspending the execution of operations in the queue upon detecting that established load limits have been exceeded.
5. When choosing a strategy for maximum equipment utilization, upon completing an operation

in the queue, either an operation is selected for which there are enough free equipment resources.

When creating methods for the dynamic management of data streams within operation queues, it is important to consider previous approaches, such as the method developed in references [20, 21], which analyzes the impact of load modes on system memory, focusing on simulating high-load situations and measuring memory usage. The new method, in the comparison to the original one, reduces excessive parallelization, improves forecasting accuracy of system loads and eliminates the necessity to measure metrics in a constrained environment. These enhancements are possible due to the implementation of a dynamic regulation model in the operation queue.

6. ASSESSMENT OF THE COMPLEXITY OF THE NEW MODEL

Each operation in the system is characterized by an array of real numbers, which reflects various types of system load. For each type of load, the first step is to calculate the expected load level μ_{op} for an individual operation. This process includes quick sorting the array, iterating through it for data filtering, calculating the mean μ, and determining the root mean square error. Similar steps are taken to determine the overall expected load level.

The complexity of calculating the expected load level for an operation can be expressed by the formula (7):

$$O_\mu = k\log k + 3k + \sum_{i=1}^{k}(n_i \log n_i + 3n_i) \quad (7)$$

where k is the number of operations; n_i is the number of readings in the i-th operation.

To assess the overall complexity of the model, we use the following formula (8):

$$O = \sum_{j=1}^{n}\sum_{i=1}^{n_j} O_{\mu_i} \quad (8)$$

Here, n represents the total number of different types of operations, and n_j denotes the number of types of loads for each operation.

7. ANALYSIS OF M/M/c MODELS UNDER CONDITIONS OF LIMITED AND UNLIMITED PARALLELISM

To evaluate the proposed method, the M/M/c model was selected, which is classical in queueing theory. This model is described as a system with “Poissonian arrivals, exponential service times, and c servers”. In the context of M/M/c theory, key performance indicators are the average response time and the number of operations in the system. The M/M/c model accounts for the random intensity of arrivals and exponentially distributed processing [22, 23] times, thus assessing the customer's waiting time in the queue, which makes it relevant for analyzing performance in operation queues with parallelism [24].

The M/M/c model comprises:

M is distribution of time between arrivals (between two consecutive clients) follows an exponential law.

M is service time also follows an exponential law.

c is number of service channels (CPU/GPU cores).

Within the context of the M/M/c model in queueing theory, two approaches are considered: the traditional model without parallelism constraints and the adapted model with parallelism constraints.

The waiting time in the queue W_q for the M/M/c model is calculated as shown in formula (9):

$$W_q = \frac{\lambda^c \cdot \mu^{-c}}{C! \cdot (1-\rho)} \quad (9)$$

where λ is arrival rate (requests per unit time); μ is service rate per handler (requests per unit time); $\rho = \frac{\lambda}{c \cdot \mu}$ – the load level of the system.

In the method of dynamic operation queue management, targeted control of the λ parameter through the restriction of parallelism leads to reduced processing time for each operation. Preventing excessive parallelism not only improves processing speed by reducing resource conflicts but also enhances the stability and predictability of system during peak loads. Unlike the traditional M/M/c model, where an increase in the number of parallel operations could lead to overload and increased waiting time, the new approach ensures more efficient and balanced use of equipment resources, leading to improved overall system response time [25, 26].

8. VIDEO CONVERSION OPERATIONS - AN EXAMPLE OF ADAPTIVE SYSTEM MONITORING MODEL

To demonstrate the adaptive system monitoring model, the process of converting video and audio files is used. In this process, video files are converted from 4K to resolutions of 1080p, 720p, and 480p, and audio files from FLAC to MP3 format. Such preprocessing avoids the need for instantaneous conversion when reducing quality

during online streaming and ensures efficient media content management. Before entering the processing queue, each uploaded video [27] and audio file undergoes authentication on the server using our chunking model and method [28], ensuring efficient data authenticity verification.

Conversion operations are implemented using gstreamer [29], which meets the requirements of a real project and includes hardware video encoding and decoding, as well as audio transcoding.

For example, the g streamer commands for such operations are as follows:

For video:

```
gst-launch-1.0 -e \
filesrc location='/some/path/input.mkv' \
! matroskademux ! h264parse ! avdec_h264 \
! videoconvert ! videoscale ! \
'video/x-raw,width=1280,height=720' ! \
vtenc_h265 ! h265parse ! matroskamux ! \
filesink location='/some/path/output.mkv'
```

For audio (FLAC to MP3):

```
gst-launch-1.0 \
filesrc location='/some/path/input.flac' \
! flacparse ! flacdec ! audioconvert \
! lamemp3enc bitrate=320 target=1 \
quality=2 ! id3v2mux ! filesink \
location='/some/path/output.mp3'
```

The experimental results obtained from this model on a MacBook Pro 2021 16" with an M1 Pro processor showed the following load indicators: for the video conversion operation – [GPU: 98.2 %, CPU: 34.21 %], and for the audio encoding operation (FLAC to MP3) – [GPU: 0 %, CPU: 9.5%].

9. PRACTICAL COMPARISON OF LOAD MANAGEMENT METHODS IN OPERATION QUEUES

A demonstration experiment aimed at the comparative analysis of a new method for managing loads in operation queues versus traditional approaches uses the “Big Buck Bunny” [30] video at 1080p resolution and a FLAC audio track, both under Creative Commons license, as materials. There are two series of the experiment: the first with a video length of 10 minutes 35 seconds, and the second – 5 minutes 8 seconds, where the second video is a trimmed version of the first to the specified time.

The analysis methodology covers the following load management strategies:

1. Sequential execution: Processing one operation at a time without parallelism.

2. Maximum parallelism: Implementation without load management, which can lead to system overload[31].

3. Limited parallelism: Considering the overloaded GPU level, affecting execution time.

4. New method: Application of FIFO and maximum load strategies.

The results of the experiments are shown in Table 1 and Table 2, corresponding to durations of 5 minutes 8 seconds and 10 minutes 35 seconds, respectively. These durations reflect different segments of the “Big Buck Bunny” video used to test the load management methods under varying conditions.

Table 1. **Results of load management experiments using the first 5 minutes 8 seconds of the “Big Buck Bunny” Video**

Type	Average video conv. time (s)	Average audio conv. time (s)	Total execution time (s)
Sequential Execution	59.71	7.95	676.12
Maximal Parallelism	623.72	14.97	623.79
Overloaded Level	111.24	8.37	606.39
New Method	60.2	7.9	602.4

Source: **compiled by the authors**

Table 2. **Results of load management experiments using the full 10 minutes 35 seconds of the “Big Buck Bunny” Video**

Type	Average video conv. Time (s)	Average audio conv. time (s)	Total execution time (s)
Sequential Execution	123.02	7.9	1331.44
Maximal Parallelism	1283.01	16.14	1283.21
Overloaded Level	239.26	8.2	1271.78
New Method	122.62	7.9	1226.24

Source: **compiled by the authors**

The analytical evaluation included measuring the average time for video and audio conversion, as well as the total execution time of a queue consisting of ten video and audio operations alternating with each other. Key evaluation criteria were the total execution time of all operations and the time efficiency of each operation.

The sequential execution method demonstrated the minimal average processing time for audio and video. However, the strategy without load

management showed the longest total execution time due to the increased duration of each individual operation, which can be inefficient for the rapid delivery of video content to the end user.

Using maximum parallelism, it was observed that the duration of a single video conversion operation matched the total experiment time, leading to a significant increase in waiting time, which is undesirable from a user service perspective.

Setting a strict limit on the number of parallel operations led to a doubling of the average video conversion time compared to sequential execution, while the audio processing time was comparable to sequential execution, indicating insufficient optimization in the use of equipment resources.

Excessive parallelism led to a proportional increase in operation execution time. This was especially noticeable in the maximum parallelism strategy, where the total time increased due to the time spent switching between tasks.

With the application of the new method, almost identical audio and video execution times were achieved as with sequential execution, with minimized total execution time and nearly maximum equipment load.

The new model and method achieve the shortest operation execution times while simultaneously reaching peak equipment load. Importantly, these efficiencies contributed to a significant reduction in total execution time – 10.9 % in the first series of experiments and 7.9 % in the second series, compared to sequential execution.

The implementation of this model helps to reduce the time of individual operations without increasing the total processing time, with a noted decrease of 0.7 % compared to the baseline of limited parallelism in the series of experiments.

10. DISCUSSION

There are multiple approaches with the aim of preventing excessive parallelism in operation queues on server equipment.

An approach using AI-based models and methods demonstrates the same accuracy in forecasting system load. However, unlike the new model and method, these approaches require significant computational resources.

An approach for migrating virtual machines (VMs) between physical servers was proposed. Multiple VMs running on a single physical server is a common situation for hosting providers. However, it is better to avoid this approach as it necessitates complex methods and does not enable efficient resource management.

Methods employing Markov chains with machine learning use a single parameter to evaluate operations, which is applied to operations with varying loads. The overhead for such methods should be considerable.

Lower overhead and higher forecasting accuracy are achieved by the new model and method. A unique feature is the manual separation of operations by keys on the operation server, which is not considered a drawback since real operations have their endpoints for each operation.

RESUME

In the context of this research, studies and evaluations of various methods for managing loads in operation queues were conducted, with particular attention given to comparing modern approaches with traditional methods in the context of digital infrastructure and data processing. There were assessed the efficacy of various load management strategies, including sequential execution, maximum and limited parallelism, alongside the newly developed model and method.

The scientific novelty lies in the new model and method, based on system monitoring and linear regression, distinguished by the use of operations by keys and the ability to achieve the research goal of processing operations in the queue. The new load management method delivers operation execution times similar to sequential execution and cuts the total execution time down to what you'd see with limited parallelism. Experiments confirmed that with the application of this method, the total execution time of operations is reduced by 10.9 % and 7.9 % in various test scenarios compared to sequential execution. Limited parallelism reduces the risk of resource conflicts unlike maximum parallelism, but it also reduces the processing time of each task and enhances the efficiency of resource use.

The new strategies – FIFO and “maximizing equipment utilization” – not only improve the efficiency of operation processing but also offer enhanced flexibility in resource utilization.

In the future we plan to refine our model and method to dynamically adjust the number of active servers, with the goal of optimizing energy efficiency by maximizing the load on the fewest possible servers. The practical applicability of the new model and method is planned to be tested on real company servers in a production environment.

ACKNOWLEDGEMENT

We express our gratitude to Oleksandr Drozd, whose contributions and unique perspectives on technological solutions have profoundly influenced our work. His dedication to mentorship has not only

shaped our professional paths but also left a lasting legacy of wisdom that will forever resonate in our hearts.

We are equally grateful to our colleagues for their unwavering support and for fostering such an inspiring atmosphere.

REFERENCES

1. Surkov, S. & Martynyuk, O. "Authentication and request processing model in high load modes for IoT components". *Proc. IntelITSIS 2021: Intelligent Information Technologies & Systems of Information Security*. 2021; 2853: 1–12, https://www.scopus.com/authid/detail.uri?authorId=57103247200.

2. "Amazon Web Services. AWS documentation". – Available from: https://aws.amazon.com/documentation. – [Accessed: Dec. 2023].

3. "Google Cloud. Google cloud documentation". – Available from: https://cloud.google.com/docs. [Accessed: Oct. 2023].

4. Chaturvedi, A., Sengar, P. & Sharma, K. "Horizontal dynamic resource scaling by measuring the impacts of scheduling interval in cloud computing". *Proceedings of International Conference on Communication and Artificial Intelligence*. 2021. p. 529–537. DOI: https://doi.org/10.1007/978-981-33-6546-9_50.

5. Romankevich, V. A., Morozov, K. V., Feseniuk, A. P., et al. "On evaluation of reliability increase in fault-tolerant multiprocessor systems". *Applied Aspects of Information Technology*. 2024; 7: 81–95. DOI: https://doi.org/10.15276/aait.07.2024.7.

6. Kavitha, P. "Digital image and video processing: Algorithms and applications". *Journal of Electrical Systems*. 2024; 20: 1390–1396. DOI: https://doi.org/10.52783/jes.1516.

7. Mashtalir, S. V. & Lendel, D. P. "Video fragment processing by Ky Fan norm". *Applied Aspects of Information Technology*. 2024; 7: 59–68. DOI: https://doi.org/10.15276/aait.07.2024.5.

8. Romankevych, V. O., Mozghovyi, I. V., Serhiienko, P. A., et al. "Decompressor for hardware applications". *Applied Aspects of Information Technology*. 2023; 6: 74–83. DOI: https://doi.org/10.15276/aait.06.2023.6.

9. Wirtz, B. W., Weyerer, J. C. & Geyer, C. "Artificial intelligence and the public sector – applications and challenges". *International Journal of Public Administration*. 2019; 42 (7): 596–615, https://www.scopus.com/record/display.uri?eid=2-s2.0-85050557949&origin=resultslist.
DOI: https://doi.org/10.1080/01900692.2018.1498103.

10. "Swift Dev. Team. The swift programming language (Swift 5.9): Concurrency". – Available from: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/concurrency. – [Accessed: Sep. 2023].

11. Kushchazli, A., Safargalieva, A., Kochetkova, I., et al. "Queuing model with customer class movement across server groups for analyzing virtual machine migration in cloud computing". *Mathematics*. 2024; 12: 1–19, https://www.scopus.com/authid/detail.uri?authorId=57279747300.
DOI: https://doi.org/10.3390/math12030468.

12. Wojciech, K. & Iwona, P. "A discrete-time queueing model of a bottleneck with an energy-saving mechanism based on setup and shutdown times". *Symmetry*. 2024; 16: 1–6, https://www.scopus.com/authid/detail.uri?authorId=6507209963.
DOI: https://doi.org/10.3390/sym16010063.

13. D'Apice, C., Dudin, A., Dudina, O., et al. "Analysis of queueing system with dynamic rating-dependent arrival process and price of service". *Mathematics*. 2024; 12: 1–10, https://www.scopus.com/authid/detail.uri?authorId=7006796728.
DOI: https://doi.org/10.3390/math12071101.

14. Fielding, R. & Reschke, J. "Hypertext transfer protocol (HTTP/1.1): Message syntax and routing, IETF RFC 7230". – Available from: https://tools.ietf.org/html/rfc7230. – [Accessed: Sep. 2023].

15. Belshe, M. & Peon, R. "Hypertext transfer protocol version 2 (HTTP/2), IETF RFC 7540". – Available from: https://tools.ietf.org/html/rfc7540. – [Accessed: Oct. 2023].

16. Bishop, M. "Hypertext transfer protocol version 3 (HTTP/3), IETF RFC 9114". – Available from: https://datatracker.ietf.org/doc/html/rfc9114. – [Accessed: Nov. 2023].

17. The, V. T., Tien, N. T. K. & Thanh, T. K. "A survey on deep learning based face detection". *Applied Aspects of Information Technology*. 2023; 6: 201–212. DOI: https://doi.org/10.15276/aait.06.2023.15.

18. Mashtalir, S.V. & Nikolenko, O.V. "Data preprocessing and tokenization techniques for technical Ukrainian texts". *Applied Aspects of Information Technology*. 2023; 6: 318–326. DOI: https://doi.org/10.15276/aait.06.2023.22.

19. Hlukhov, V. S. & Sydorko, D. S. “Algorithms and software for verification of scientific and technical text documents”. *Applied Aspects of Information Technology*. 2023; 6: 304–317. DOI: https://doi.org/10.15276/aait.06.2023.21.
20. Surkov, S. “Reduction of the impact of critical modes for authorization protocols for large requests in operation queue enviroment”. *Applied Aspects of Information Technology*. 2020; 3 (3): 145–153. DOI: https://doi.org/10.15276/aait.03.2020.3.
21. Surkov, S. S. “Comparison of authorization protocols for large requests in the operation queue environment”. *Herald of Advanced Information Technology*. 2020; 3 (3): 163–173. DOI: https://doi.org/10.15276/hait.03.2020.5.
22. Shajil, A. & Srinivasa, K. “SWOT analysis of parallel processing APIs-CUDA, OpenCL, OpenMP and MPI and their Usage in Various Companies”. *International Journal of Applied Engineering and Management Letters*. 2023. p. 300–319. DOI: https://doi.org/10.47992/IJAEML.2581.7000.0206.
23. Nashma, T., Zryan, N., Subhi, Z., et al. “Optimizing time consumption for smartphone-based distributed parallel processing system”. *Passer Journal of Basic and Applied Sciences*. 2024; 6: 254–265. DOI: https://doi.org/10.24271/psr.2024.188570.
24. Xiang, H., Yuan, Y., Sheng, M., et al. “Optimizing the parallelism of communication and computation in distributed training platform”. *Springer, Singapore*. 2024. p. 340–359. DOI: https://doi.org/10.1007/978-981-97-0834-5_20.
25. Raghav, M. “Low latency systems for parallel processing and programmable logic in GPUs and FPGAs”. 2023. p. 1–8. DOI: https://doi.org/10.13140/RG.2.2.16359.01443.
26. Kappes, G. & Anastasiadis, S. “A family of relaxed concurrent queues for low-latency operations and item transfers”. *ACM Transactions on Parallel Computing*. 2022. p. 1–37. DOI: https://doi.org/10.1145/3565514.
27. Sheremet, O. I., Sadovoi, O. V., Harshanov, D. V., et al. “Efficient face detection and replacement in the creation of simple fake videos”. *Applied Aspects of Information Technology*. 2023; 6: 286–303. DOI: https://doi.org/10.15276/aait.06.2023.20.
28. Surkov, S. S. “Model and method of chunk processing of payload for HTTP authorization protocols”. *Proceedings of 2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*. Slavske: Ukraine. 2020. p. 317–321, https://www.scopus.com/authid/detail.uri?authorId=57103247200.
DOI: https://doi.org/10.1109/TCSET49122.2020.235447.
29. GStreamer Dev. Team. “GStreamer: Open source multimedia framework”. – Available from: https://gstreamer.freedesktop.org. – [Accessed: Dec. 2023].
30. “Blender Foundation. About Big Buck Bunny”. – Available from: https://peach.blender.org/about/. – [Accessed: Nov. 2023].
31. “Apple Dev. Team. Grand Central Dispatch”. – Available from: https://github.com/apple/swift-corelibs-libdispatch. – [Accessed: Nov. 2023].

Conflicts of Interest: the authors declare no conflict of interest

Received 12.02.2024
Received after revision 09.04.2024
Accepted 10.05.2024

DOI: https://doi.org/10.15276/aait.07.2024.9
УДК 004.75

Модель та метод для підвищення ефективності обробки черг операцій при максимальному завантаженні серверного обладнання

Сурков Сергій Сергійович[1)]
ORCID: http://orcid.org/0000-0001-9224-7526; k1x0r@ukr.net. Scopus Author ID: 57103247200
Мартинюк Олександр Миколайович[1)]
ORCID: http://orcid.org/0000-0003-1461-2000; anmartynyuk@ukr.net. Scopus Author ID: 57103247200

Дрозд Олександр Валентинович[1)]
ORCID: http://orcid.org/0000-0003-2191-6758; drozd@ukr.net. Scopus Author ID: 55388226700
Дрозд Мірослав Олександрович[1)]
ORCID: http://orcid.org/0000-0003-0770-6295; myroslav.drozd@opu.ua. Scopus Author ID: 56667174000
[1)] Національний університет «Одеська політехніка», проспект Шевченка, 1. Одеса, 65044, Україна

АНОТАЦІЯ

Було проведено аналіз існуючих рішень з метою запобігання надмірній паралелізації, скорочення часу обробки та точного прогнозування навантаження в чергах операцій. Потім було оцінено нову модель та методику, розроблені для підвищення ефективності обробки черг операцій, особливо при максимальному навантаженні серверного обладнання, у порівнянні з традиційними методами. Ці методи, включаючи послідовне виконання, максимальну та обмежену паралелізацію, були оцінені. Новий метод використовує дві підстратегії: ‘перший прийшов-перший пішов’, яка особливо корисна, оскільки паралелізм не гарантує послідовного порядку результатів, і максимізацію використання обладнання для оптимізації продуктивності. Завдяки новій адаптивній моделі моніторингу, що базується на лінійній регресії, новий метод досягає часу виконання операцій, порівнянного з послідовним виконанням, і загального часу виконання, подібного до обмеженої паралелізації. Обмежена паралелізація, хоча й зменшує конфлікти за ресурси порівняно з максимальною паралелізацією, все ж збільшує час обробки кожної операції, що підкреслює важливість балансування кількості паралельних операцій з наявними ресурсами системи. Комплексність нової моделі була оцінена за допомогою асимптотичної складності та проаналізована за допомогою моделі масового обслуговування з паралельними каналами в умовах обмеженої та необмеженої паралелізації. Було проведено дві серії експериментів для порівняльного аналізу нового методу управління навантаженнями в чергах операцій у порівнянні з традиційними підходами. Додатково підкреслюється потенціал гнучкого використання ресурсів у управлінні навантаженнями в цифрових інфраструктурах.

Ключові слова: управління навантаженнями; черги операцій; цифрова інфраструктура; обробка даних; паралелізація; оптимізація ресурсів; прогнозування завантаження

ABOUT THE AUTHORS

Sergii S. Surkov - PhD Student, Department of Computer Intellectual Systems and Networks. Odessa Polytechnic National University, 1, Shevchenko Ave. Odessa, 65044, Ukraine
ORCID: http://orcid.org/0000-0001-9224-7526; k1x0r@ukr.net. Scopus Author ID: 57103247200
***Research field*:** Parallel computing in digital infrastructures; resource allocation algorithms; real-time queue management systems; performance prediction and system efficiency; authorization protocols

Сурков Сергій Сергійович - аспірант кафедри Комп’ютерних інтелектуальних систем та мереж. Національний університет «Одеська політехніка», проспект Шевченка, 1. Одеса, 65044, Україна

Oleksandr M. Martynyuk - PhD, Associate Professor, Department of Computer Intellectual Systems and Networks. Odessa Polytechnic National University, 1, Shevchenko Ave. Odessa, 65044, Ukraine
ORCID: https://orcid.org/0000-0003-2366-1920; anmartynyuk@ukr.net. Scopus Author ID: 57103391900
Research field: Behavioral testing of computer systems; formal verification and recognizing of digital systems; artificial intelligence

Мартинюк Олександр Миколайович - кандидат технічних наук, доцент кафедри Комп’ютерних інтелектуальних систем та мереж. Національний університет «Одеська політехніка», проспект Шевченка, 1. Одеса, 65044, Україна

Oleksandr Va. Drozd - Doctor of Engineering Sciences, Professor, Department of Computer Intellectual Systems and Networks. Odessa Polytechnic National University, 1, Shevchenko Ave. Odessa, 65044, Ukraine
ORCID**:** https://orcid.org/0000-0001-8305-2217; drozd@ukr.net. Scopus Author ID: 55388226700
Research field: Testing and diagnosis of computer systems; arithmetical foundations of computer systems; computer systems and components

Дрозд Олександр Валентинович - доктор технічних наук, професор кафедри Комп’ютерних інтелектуальних систем та мереж. Національний університет «Одеська політехніка», проспект Шевченка, 1. Одеса, 65044, Україна

Myroslav O. Drozd - PhD, Associate professor, Information Systems Department. Odessa Polytechnic National University, 1, Shevchenko Ave. Odessa, 65044, Ukraine
ORCID: https://orcid.org/0000-0003-0770-6295; myroslav.drozd@opu.ua. Scopus Author ID: 56667174000
Research field: On-line testing and circuit checkability in the digital component of safety-related systems

Дрозд Мірослав Олександрович - кандидат технічних наук, доцент кафедри Інформаційних систем. Одеський національний політехнічний університет, проспект Шевченка, 1. Одеса, 65044, Україна