IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/45AI/2025/4526025/
Vol. 2025 (20), Article ID 4526025

Mathematical Model of MLOps System*

Mateusz MILCZAREK and Adrian P. WOZNIAK

Military University of Technology, Warsaw, Poland

Correspondence should be addressed to: Mateusz MILCZAREK, mateusz.milczarek.wat@gmail.com

* Presented at the 45th IBIMA International Conference, 25-26 June 2025, Cordoba, Spain

Abstract

MLOps is a rapidly growing field in which optimizing resource allocation has become a key challenge. Despite the growing popularity of MLOps in production environments, there is a lack of articles in the literature that propose formal mathematical models for MLOps processes. This gap limits the development of analytical tools that support effective resource management in such systems. This study presents a mathematical model of an MLOps system and formulates a corresponding optimization problem. The model consists of three layers — processes, components, and servers — which serve as input parameters for the system, allowing for a flexible yet comprehensive representation of real-world MLOps architectures. Using this structure as a foundation, constraints and optimization criteria were established to enhance resource management. Finally, an objective function was proposed, whose minimization leads to improved system performance, reduced operational costs, and increased process reliability.

Keywords: MLOps, machine learning, mathematical modeling, mathematical model

This work was supported by the Military University of Technology under Project UGB 531-000023-W500-22.

Introduction

Technologies related to machine learning and artificial intelligence have become increasingly important in recent years. Report [1] states that the MLOps market was worth $196.63B in 2023 and projects a 36.6% annual growth rate, in value through 2030.

Although engineering practice widely uses MLOps, few formal mathematical models accurately capture the processes in these systems. Such models are necessary to understand the dynamics of interactions between system components, optimize the flow of data and resources, and predict system behavior under changing operational conditions. There are numerous publications that attempt to describe MLOps. According to Testi et al. [2], MLOps focuses on deploying machine learning models into production as an iterative process. This approach continuously improves models using newly acquired data. MLOps originates from DevOps practices and integrates them with machine learning methodologies [2] [3] to automate, monitor, and manage the lifecycle of ML models. The key steps in designing the MLOps process include:

- Data collection and preparation - extracting, processing, and cleaning data to train models.
- Model preparation - creating and evaluating machine learning models based on collected data.
- Continuous Integration and Delivery (CI/CD) - automated deployment of models into production.
- Model performance monitoring - automated validation of model performance to prevent degradation.

Cite this Article as: Mateusz MILCZAREK and Adrian P. WOZNIAK, Vol. 2025 (20) "Mathematical Model of MLOps System" Communications of International Proceedings, Vol. 2025 (20), Article ID 4526025, https://doi.org/10.5171/2025.4526025

- Model retraining - updating models when prediction accuracy declines.

The MLOps infrastructure uses tools commonly found in the DevOps cycle, such as CI/CD and model monitoring, but extends them with several unique elements. These include:

- Processes for training and monitoring machine learning models.
- Inference (scoring) process using trained models.
- Feature Store as a repository for managing features used in training.
- Model repository for managing and versioning trained models.
- Model Orchestrator to oversee model production, monitoring, and scoring. An inference (score) process using trained models.

Different strategies for allocating processes to available resources in MLOps can lead to significant variations in system performance, both in terms of execution time and infrastructure efficiency. To enhance this process, optimization is required, necessitating the formulation of an appropriate optimization problem. A crucial step in this direction is the development of a mathematical model that formally describes the relationship between resource allocation and system performance.

A representative example of suboptimal resource allocation occurs in an MLOps system deployed within the infrastructure of a large retail company, where task-to-resource allocation follows a naive “first-available” server strategy without prioritization or optimization. In this setup, a layer of processes e.g. model training, monitoring, scoring, initiate actions that are then executed by dedicated components e.g., for data retrieval, weight updates, or prediction generation. These components are deployed to servers without regard to computational load, priority, or resource suitability. As a result, lightweight monitoring components may run on GPU-equipped servers, training jobs may be assigned to servers lacking GPU acceleration, and latency-sensitive scoring tasks may be delayed due to placement on overburdened CPUs. This leads to inefficient resource utilization, increased inference latency, and higher operational costs. In contrast, the mathematical model proposed in this paper, which divides the system into three layers while explicitly accounting for task characteristics and hardware requirements, enables optimized allocation. This approach reduces congestion, maximizes the use of specialized hardware where beneficial, and minimizes execution time across the MLOps pipeline.

The rest of this article is organized as follows. Chapter two presents the current state of knowledge on the mathematical modeling of systems with MLOps architecture. Chapter three formally describes the system and formulates functions to optimize resource allocation. It begins with an introduction to the input data, describing the various layers of the MLOps system. These layers are modeled in a generalized manner, ensuring flexibility so that users are not constrained by predefined processes, but can adapt them to the specific requirements of the system being optimized. Next, the decision variables relevant to the overall approach are defined, followed by the formulation of the necessary constraints. Subsequently, minimization criteria are introduced, considering factors such as execution time, cost, the number of threshold violations, and variance. These criteria enable the optimal allocation of resources within a MLOps-based system, ensuring a balance between performance, cost, and operational efficiency. This approach facilitates efficient resource management and adaptation to the dynamic requirements of machine learning processes. Finally, the objective function is formulated to be minimized based on the specified criteria.

Related Works

MLOps is a relatively new field that emerged to meet the growing demand for developing, deploying, and managing machine learning models in production environments. As a result, the number of publications on MLOps remains limited. Additionally, no publications have been found that directly attempt to mathematically model a system with MLOps architecture. Such models need to consider their characteristics, including the processes responsible for training, monitoring, and using machine learning models for scoring, among other tasks. To broaden the scope of our search, we also examined articles on the most popular solutions for orchestrating and

distributing model training, namely Kubeflow and Kubernetes. The paper by Zhang et al. [4] presents a mathematical model for scheduling software tasks for robot fleets on Kubernetes clusters. A genetic algorithm widely used in optimization tasks for appropriate allocation of computing resources was used to optimize task allocation. However, this article does not focus on the MLOps process. Ding et al. [5] also addressed the problem of optimizing task allocation for microservices in Kubernetes clusters. A mathematical model was also developed for this purpose, but it does not consider the performance of GPUs and is limited to the Kubernetes platform. It also does not consider the versioning of data and models. In the work of Viody et al. [6] the authors propose an extended version of a scheduling program for Kubeflow in an application for distributed training of deep neural networks. However, mathematical modeling is present here only in a residual form - for counting the load of individual nodes. The article by Menouer et al. [7] is another proposal of a scheduling algorithm for Kubernetes. As in the previous article, mathematical modeling is used here to describe resource allocation and considers only CPU performance, RAM capacity, and disk space. Shridhar et al. [8] focus on distributed model training in the MLOps process and appropriate allocation of computing resources in the cloud. The authors propose a heuristic approach to the resource allocation problem. However, in the posted mathematical model, they focus only on GPU performance and network latency. Another paper by Mondal et al. [9] also covers an attempt to improve the performance of the Kubernetes platform. The mathematical model posted there, however, only covers the problem of latency and network bandwidth between consecutive nodes. The work of Fard et al. [10] also focused on optimizing the placement of microservices on Kubernetes clusters. For optimization, the scheduling problem is modeled as a complex variant of the knapsack problem. Based on the review of related work, several gaps in the mathematical modeling of the MLOps process have been identified. While the reviewed studies primarily focused on mathematical modeling for the Kubernetes platform, they did so in a limited capacity and did not explicitly address the modeling of the MLOps process.

Mathematical model of the MLOps system

The system with MLOps architecture is divided into three distinct layers: processes, components, and servers. A process in MLOps is an organized and repeatable set of steps necessary to perform tasks, such as training or monitoring models, among others. The individual steps (services) required to execute a given process are made available through components, which are run on specific servers that make their computing power available to them. To optimize the system, in addition to defining the above layers, constraints, optimization criteria, and the objective function to be minimized are also specified. A detailed discussion of these aspects is presented in the following subsections.

Input data

The first part of the proposed model, based on the MLOps architecture, is to define the inputs that a user can provide to the system. MLOps is an architecture largely based on DevOps practices. The main characteristics of the described architecture are the processes of model training, inference (score) and monitoring of the model with its characteristics, such as data drift. The following subsections break down the various layers of input data that make up the MLOps system. Starting with the process layer, then moving through the component layer (which implements the steps of each process), and ending with the server layer on which these components are executed. The various layers were modeled generically. The user is not faced with ready-made schemes of subsequent processes but may define each process in terms of, for example, its required computing power or the process steps necessary for its execution. This approach allows the user to precisely define the steps involved in each process within the optimized MLOps system, providing flexibility in the model.

Process Layer Model

The initiator of processes in a system with MLOps architecture is the organization in which this system is implemented. The layer of processes that make up the overall MLOps approach is defined as a set of P:

$$P = \{1,2,3, \dots, p, \dots P\}$$

which stands for the set of process type numbers in the MLOps approach. A process in MLOps comprises certain steps necessary for the execution of a given process. Therefore, for each process, a process definition is defined, which will determine exactly what steps the process will comprise. Let us assume that PD (process definition) is the definition of a process. So, for each successive instance of a p-th process, we have its definition, which can be written in the form of a vector:

$$PD = \{PD_p\}_P$$

In turn, the definition of the p-th process can be written as:

$$PD_p = \langle SV_p, NS_p, \{\Psi\} \rangle$$

where:

- SV_p - (step vector) vector of process step numbers. The process definition does not have to contain all the steps, but only the selected ones that are necessary to complete the process. In addition, the steps in each process p should be executed sequentially.
- $NS_p \in N^+$- (number of steps) number of steps in a process. This is the number that determines how many steps a process consists of.
- $\{\Psi\} \in R^+$ [s] - A random variable indicating the time between process runs.

Thanks to this approach, the user at the input can define exactly what steps each process implemented in the optimized MLOps system consists of, which allows for flexibility in the proposed model. In addition, thanks to the distinction of process steps, certain steps in the process can be performed in parallel in several instances, e.g. the model training process, while others that can occur in a single instance, e.g. feature store. Next, a set of process steps was defined:

$$PS = \{1,2,3, \dots, ps, \dots, PS\}$$

So, the process step vector can be defined as:

$$SV_p = [ps]_{NS_p}$$

where ps is a reference to a process step.

For each step, its definition can be written in the form of:

$$SD_s = \langle L_{ps}^{CPU}, L_{ps}^{GPU}, L_{ps}^{RAM}, NL_{ps}^{IN}, NL_{ps}^{OUT}, MPT_{ps}, C \rangle$$

where:
$L_{ps}^{CPU} \in R^+$ [GHz*H] – (load) processor performance required by the process,
$L_{ps}^{GPU} \in R^+$ [TOPS*H] – graphics card performance required by the process,
$L_{ps}^{RAM} \in R^+$ [MB] – amount of ram required by the process,
$NL_{ps}^{IN} \in R^+$ [Mb/s] – (network load) input network load,
$NL_{ps}^{OUT} \in R^+$ [Mb/s] – output network load,
$MPT_{ps} \in N^+$– (maximum number of parallel threads) maximum number of parallel threads on which a process step is executed,
C – the component on which the step is run.

At this stage, all the parameters necessary for a given step in a given system with MLOps architecture were highlighted. Here we have considered both CPU, RAM, network bandwidth and GPU, which is an important computing platform especially when training neural networks. The maximum number of parallel threads is

important to determine how many instances it pays to disperse computation. Above a certain level, the cost of, for example, maintaining components or communication between successive instances may outweigh the benefits of such an approach.

Component layer model

Another layer of input data consists of components, on which the tasks of each process step are executed. Currently, Docker is the most popular solution for creating isolated environments for each component. With this approach, you can easily manage dependencies for each component and increase the number of component instances with the required process steps. This approach is useful for tasks such as parallelizing computations (e.g., training a specific model), creating new instances for subsequent inferences (scores), or training different machine learning models within complex MLOps systems.

The set of individual components is defined as follows:

$$C = \{1,2,3, \dots, c, \dots, C\}$$

Then each component c can be specified by its definition (component definition) written in the form of:

$$CD_c = \langle C_s, L_c^{CPU}, L_c^{RAM}, PL_c \rangle$$

where:

C_s - component assignment server,
$L_c^{CPU} \in R^+$ [GHz] - CPU load to maintain component activity at rest,
$L_c^{RAM} \in R^+$ [MB] - the amount of ram required to keep the component active at rest,
$PL_c \in B$ – (permanent launch) determines whether the component is running all the time. Some components, such as those that provide a model repository service, should be always kept active.

Server layer model

The final input layer in the MLOps mathematical model consists of the servers on which its components run. Servers for components can be both local servers - physical machines that perform specific tasks, and they can also be resources made available through cloud computing. A collection of servers can be defined as:

$$S = \{1,2,3, \dots, s, \dots, S\}$$

The definition of each component is as follows:

$$DS_s = \langle P_s^{CPU}, P_s^{GPU}, P_s^{RAM}, NB_s^{OUT}, NB_s^{IN}, PRICE, NUMBER \rangle$$

where:

$P_s^{CPU} \in R^+$ [GHz] - (power) CPU performance available on the server,
$P_s^{GPU} \in R^+$ [TOPS] - GPU performance available on the server,
$P_s^{RAM} \in R^+$ [MB] - the amount of available operating memory on the server,
$NB_s^{OUT} \in R^+$[Mb/s]- (network bandwidth) server network bandwidth at the output,
$NB_s^{IN} \in R^+$ [Mb/s] - server's network bandwidth at the input,
$PRICE \in R^+$ [PLN/s] - the cost incurred per second of server operation. The amount is calculated from the time the component is started on the server until it is shut down,
$NUMBER \in N^+$– number of servers with the above parameters available for allocation.

With this approach, the user can specify the available computing resources, including the number and cost of instances of each type. For physical machines, the cost can be determined based on power consumption, estimated hardware maintenance, and software license costs. In cloud environments, server usage costs are typically calculated based on the active time of a specific server, such as for virtual machines.

Decision variables and constraints

In the optimization process in a system with MLOps architecture, a key role is played by decision variables and constraints, which together define the space of acceptable solutions and guide the search for optimal system parameters. The decision variables in the proposed mathematical model are as follows:

$$x = \{C^S, CKL, MSU\}$$

where:

$C^S = \left[C^S_{c,s}\right]_{CxS}$ - means a matrix of allocation of components to servers, since each component can be run in parallel on multiple instances of the same server. $C^S_{c,s} \in N$.
$CKL = \left[CKL_{c,ps}\right]_{CxPS}$ - denotes a matrix for equalization of calculations. The maximum number of components on which a step can be run. The rationale for this is when a component is deployed on multiple servers, but a step requiring that component would not be executed on all of them in parallel. This is dictated by the preservation of resources in case of, for example: a score, which can be called at any time and must be executed in a strictly specified time. $CKL_{c,ps} \in N$.
$MSU = \left[MSU_{c,s}\right]_{CxS}$ - permanent startup matrix. A matrix indicating which components should be permanently running on servers. For example, these components may include those responsible for maintaining a model repository. $MSU_{c,s} \in B$.

Decision variable C^S must meet the following constraints:

- The sum of the load generated by the operation of the components must not exceed the processor processing power that this server has:

$$s \in (1,S) \sum_{c=1}^{C} C^S_{c,s} L^{CPU}_c < m^S_s$$

- The sum of the load generated by the operation of the components must not exceed the available memory that this server has:

$$s \in (1,S) \sum_{c=1}^{C} C^S_{c,s} L^{RAM}_c < p^S_s$$

- At least one instance of each component must be allocated to some server:

$$c \in (1,C) s \in (1,S) C_{c,s} = 1$$

Objective function

In order to effectively optimize a system with MLOps architecture, it is necessary to define appropriate criteria for evaluating system performance. These criteria allow formalizing the goals that the process of resource allocation and task execution management should meet. This chapter presents four key optimization criteria that reflect both system performance and cost and quality aspects. The selected optimization criteria are:

- $\overline{k}_1(t,X,D) = E\big(k_1(t,X,D)\big)$– the average expected process execution time in MLOps weighted by the expected number of instances at a given time t of system operation, therefore:

$$k_1(t,X,D) = \sum_{p=1}^{P}\left(W_p^t * E\left(T_p(t,X,D)\right)\right)$$

where:

W_p^t - denotes the weight of the process p,
T_p - denotes a random variable denoting the execution time of the p-th process of any instance.

The average expected execution time of processes in an MLOps system is a key optimization criterion, as it affects the operational efficiency and responsiveness of the entire system. High values of this indicator can indicate problems with resource allocation, infrastructure overload or inefficient task scheduling. Optimization of this criterion allows you to increase the throughput of the system and reduce the time of obtaining results, which is important, for example, scenarios that require fast inference (score) in real time. Weighting execution times by the number of instances allows you to further take into account the scale of the system load - processes run more frequently have a greater impact on the final value of this indicator. Thanks to this, optimization does not favor single, infrequently executed operations, but adapts the system performance to its real load.

- $\overline{k}_2(t,X,D) = E\big(k_2(t,X,D)\big)$– - expected process execution time variance weighted by the expected number of instances at a given time t of system operation, therefore:

$$k_2(t,X,D) = \sum_{p=1}^{P}\left(W_p^v * E\left(V_p(t,X,D)\right)\right)$$

where:

W_p^v– process variance weight p,
V_p - process variance p.

Variance plays a key role in optimizing MLOps processes, as it affects the predictability and stability of the system. Low variance means that processes operate in a repeatable manner, which is key to ensuring high availability and reliability of the MLOps infrastructure. High variance, on the other hand, can lead to unstable system performance, inefficient use of resources and difficulty in scaling models.

- $\overline{k}_3(t,X,D) = E\big(k_3(t,X,D)\big)$ the average expected number of time limit violations at a given moment t, so:

$$k_3(t,X,D) = \sum_{p=1}^{P}\left(E\left(L_p(t,X,D)\right)\right)$$

where:

L_p - number of process time limit exceedances p.

Another proposed criterion is exceeding time limits in processes. A high value of this indicator means that the system often fails to meet the target execution times, which can lead to degradation of service quality, delays in the delivery of results, and increased operating costs. By minimizing this criterion, the predictability of the system can be improved, and calculation times can be better managed.

- $\overline{k}_4(t,X,D) = E\big(k_4(t,X,D)\big)$– the total cost of implementing all processes in the system at a given moment t, so:

$$k_4(t,X,D) = \sum_{p=1}^{P}\big(K(t,X,D)\big)$$

where:
K - denotes the cost of executing the p-th process at a given time t.

The cost of process execution is also an important aspect of optimization in systems with MLOps architecture, as it directly affects the economic efficiency of model deployment and maintenance. It includes both computational costs, e.g. CPU and GPU power consumption and RAM, as well as costs related to cloud infrastructure. Minimizing this criterion allows for more economic management of resources and optimal use of available infrastructure without sacrificing the quality of system performance.

Based on the above criteria, a vector objective function was adopted:

$$\overline{k}(t,X,D) = \overline{k}_1(t,X,D), \overline{k}_2(t,X,D), \overline{k}_3(t,X,D), \overline{k}_4(t,X,D)$$

Summary

This article analyzes key aspects of the MLOps structure, considering the three fundamental layers: processes, components and servers. Constraints related to available resources and optimization criteria that determine the efficiency of the entire system were formulated. On this basis, an optimization task was set to minimize operating costs, reduce the execution time of processes and increase their reliability.

The mathematical model presented here is the first step toward more informed and efficient management of MLOps infrastructure. Proper modeling and optimization can contribute to better resource utilization, cost reduction, and increased reliability of deployed systems.

References

- Grand View Research, Artificial Intelligence Market Size, Share & Trends Analysis Report By Solution, By Technology (Deep Learning, Machine Learning, NLP, Machine Vision, Generative AI), By Function, By End-use, By Region, And Segment Forecasts, 2024 – 2030
- M. Testi *et al*., "MLOps: A Taxonomy and a Methodology," in *IEEE Access*, vol. 10, pp. 63606-63618, 2022
- Dominik Kreuzberger, et al., Machine Learning Operations (MLOps): Overview, Definition, and Architecture, IEEE Access, 2023
- Y. Zhang, F. Mirus, F. Pasch, K. -U. Scholl, C. Wurll and B. Hein, "A Comprehensive Modeling and Scheduling Approach for Allocating Distributed Multi-Robot Software to the Edge/Cloud," *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates, 2024, pp. 5799-5806
- Z. Ding, S. Wang and C. Jiang, "Kubernetes-Oriented Microservice Placement With Dynamic Resource Allocation," in *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1777-1793, 1 April-June 2023
- Y. Viody and A. I. Kistijantoro, "Container Migration for Distributed Deep Learning Training Scheduling in Kubernetes," *2022 9th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, Tokoname, Japan, 2022, pp. 1-6

- T. Menouer, C. Cérin and P. Darmon, "KOptim: Kubernetes Optimization Framework," *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, San Francisco, CA, USA, 2024, pp. 900-908
- A. Shridhar and D. Nadig, "Heuristic-based Resource Allocation for Cloud-native Machine Learning Workloads," *2022 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Gandhinagar, Gujarat, India, 2022, pp. 415-418
- Mondal, S.K.; Zheng, Z.; Cheng, Y. On the Optimization of Kubernetes toward the Enhancement of Cloud Computing. *Mathematics* 2024, *12*, 2476.
- H. M. Fard, R. Prodan and F. Wolf, "Dynamic Multi-objective Scheduling of Microservices in the Cloud," *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, Leicester, UK, 2020, pp. 386-393