DOI: https://doi.org/10.15276/ict.02.2025.24
UDC 004.76

Optimization of Dispatcher-Deployer-Executor for short-term resource leasing in multiserver systems

Sergii S. Surkov[1)]
Postgraduate Student of the Department of Computer Intellectual Systems And Networks
ORCID: https://orcid.org/0000-0001-9224-7526; k1x0r@ukr.net. Scopus Author ID: 57103247200
Oleksandr M. Martynyuk[1)]
PhD, Associate Professor of the Department of Computer Intellectual Systems and Networks
ORCID: https://orcid.org/0000-0003-1461-2000; anmartynyuk@ukr.net. Scopus Author ID: 57103391900
[1)] Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine

ABSTRACT

In modern distributed computing systems, architectures incorporating multiple servers and operation queues facilitate the management of heterogeneous resource-intensive workloads, such as artificial intelligence training, media file transcoding, and computational modeling for industrial optimization. Traditional single-processor systems are characterized by sequential constraints and inefficient resource allocation, whereas parallelism in multiserver configurations results in elevated energy consumption. The present study enhances the Dispatcher-Deployer-Executor (DDE) architecture for real-time resource consumption optimization in multiprocessor environments. The objective is to improve operation execution speed and minimize execution costs in scenarios requiring limited computational volumes and maximum parallelism. Through theoretical modeling and empirical verification, the extended DDE introduces a container mode for Deployer/Executor components, which operate within containers, enabling orchestration for short-term leasing via standard tools (e.g., Docker and Kubernetes). This implements dynamic resource scaling and efficient application of the model and method in cloud environments, with the primary advantage of rapid access to high-performance computations accompanied by substantial savings, as it eliminates the need for server acquisition or long-term leasing for restricted calculation volumes. The main application scenario is tasks oriented towards the central processing unit (CPU), as containers are optimized for CPU-bound computations, such as calculating the trajectories of dust particles through numerical solution of differential equations of motion. The results indicate a significant reduction in resource consumption and financial costs compared to the baseline mode while maintaining performance; quick rental of containers provides an advantage in cost savings and flexibility. The extended DDE contributes to sustainable computations, balancing efficiency, reliability, and speed for various tasks. Future extensions may integrate the use of standby mode and hibernation to reduce energy consumption.

Keywords: Multi-server architecture; energy optimization; heterogeneous workloads; DDE framework; task queuing; server state management; Docker; Kubernetes

Relevance. In modern computational systems, operation queues and multi-server configurations play a key role in resource management and handling large task volumes [1]. Multi-server structures distribute tasks across nodes for parallel execution, which increases throughput and reduces waiting time compared to single-computer processing. The growing resource demands of tasks, such as audio/video conversion, artificial intelligence training, and industrial equipment optimization, require method adaptation in multi-task environments [2, 3], with attention to short-term resource leasing and high server costs. Prior work proposed an optimization method for computations in dust particle flow modeling to reduce time through parallelization, but for final calculations, high precision is required, and server clusters process slowly, necessitating solutions. The shift to containerization of the model and method is driven by the need for greater flexibility in distributed environments: containers enable dynamic resource scaling without server purchase, providing quick leasing for limited computations and lowering costs while maintaining performance. This approach integrates a dispatcher for load monitoring, considering task heterogeneity, which improves distribution and real-time resource use [4, 5]. The basis lies in analyzing the balance between reliability, speed, and flexibility, with focus on containerization of components (Deployer, Executor) and dispatcher integration for load control accounting for heterogeneity.

Research Objective. The objective of the present study is to refine the model and method for enhancing the efficiency of processing operation queues at maximum server equipment load [6] for the optimization of containerization in multi-core systems during the execution of heterogeneous

tasks, including artificial intelligence model training, audio/video conversion, and industrial equipment optimization, without compromising performance. Associated tasks include: refinement of the component containerization mechanism; integration of a dispatcher for load monitoring with dynamic scaling; optimization and analysis of container activation based on task execution time and executor states.

The research employs a combined theoretical and empirical methodology for updating the Dispatcher-Deployer-Executor (DDE) architecture[7], with an emphasis on real-time containerization improvement. The theoretical part involves creating a mathematical model that details container states, resource use, and task completion times under changing load levels. The empirical part consists of software development in the Rust language[8] with the asynchronous Tokio framework, along with tests on varied computational tasks. Container transitions are set using idle time thresholds, taking into account task length, queue size, and switch duration. Containerization improvement includes task grouping by resource needs.

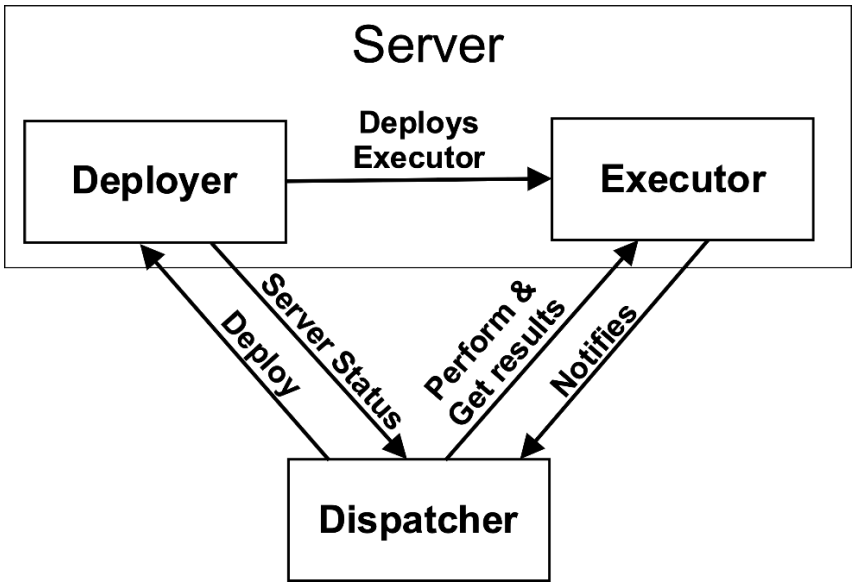

Fig. 1. **Diagram of the interaction of system components: Executor, Dispatcher, and Deployer**

Within the Dispatcher-Deployer-Executor (DDE) framework, procedures are set for queue management under high load in parallel systems. The dispatcher controls request assignment by complexity and load, and serves as a reverse proxy for HTTP/2 [9, 10] multiplexing. The deployer launches executors, and requests to it are directed through the dispatcher. The executor carries out tasks directly. Use of modified HTTP/2 with server-started requests in a reverse REST-API setup allows asynchronous work without polling.

Chunked authentication employing HMAC signatures is utilized to verify data segments within trusted networks. The TLS 1.3 [11] protocol is implemented for unsecured channels, incorporating 1-RTT handshake, 0-RTT resumption, and forward secrecy. The dispatcher incorporates SSL certificate pinning and dynamic certificates for mutual TLS (mTLS), supported by two-factor authentication comprising a code and public key, with a validity period of one month and automatic renewal. The system is implemented in the Rust programming language utilizing the Tokio asynchronous runtime to achieve high performance and security. A hybrid approach is adopted: TLS for external communications and chunked authentication for local networks. The dispatcher is capable of invoking webhooks to initiate and terminate containers following resource provisioning in cloud environments.

Container preparation within the DDE architecture involves configuring the deployer and executor as standard components with the containerization mode enabled. This configuration mode, activated upon system startup, initiates systematic measurement of operation execution metrics on the executor nodes. Specifically, it enables real-time profiling of task durations, resource utilization patterns, and throughput rates during execution phases. These measurements are aggregated and analyzed to inform adaptive load distribution strategies, ensuring equitable allocation of computational tasks across containerized instances [12]. By quantifying execution profiles, such as average processing time per task type, peak memory consumption, and CPU cycle efficiency, the system refines scheduling algorithms to minimize bottlenecks and optimize resource provisioning. This approach facilitates predictive scaling in dynamic environments, where heterogeneous workloads vary in complexity, thereby enhancing overall system responsiveness and efficiency without requiring manual intervention.

Short-term container leasing provides economic efficiency by charging only for actual resource usage, excluding costs for acquiring and maintaining proprietary servers. For tasks with uneven load, such as one-time dust particle trajectory computations, this reduces total expenses, as the user pays for hours or minutes rather than permanent infrastructure [13]. For instance, in cloud platforms like AWS [14] or Google Cloud [15], the cost of leasing a container for 1 hour ranges from approximately 0.1-0.5 USD depending on configuration, compared to capital expenditures on a server. This promotes flexibility, enabling resource scaling for peak loads without overpayments, and lowers entry barriers for small businesses or research projects. It supports features like application isolation, IAM (Identity and Access Management) roles, automated patching, encrypted storage, and AWS security integrations, with pay-as-you-go pricing for efficient resource utilization. ECS (Elastic Container Service) operates by deploying, managing, and scaling containerized applications, integrating with EC2 (Elastic Compute Cloud), Fargate, and Spot Instances for batch workloads, data processing.

For system capability demonstration, particularly in operation queue processing context, numerical computation of dust particle trajectories in a rectangular cross-section channel accounting for secondary flows is implemented. Mathematical models of secondary flows are applied for particle trajectory calculation in gas flows, which is significant for gas turbine systems, boilers, and industrial gas pipelines. Numerical modeling enables analysis of particle distribution, their deposition on surfaces, and the impact of secondary vortex flows induced by channel geometry. This facilitates channel design optimization to minimize contamination, enhance reliability, safety, and environmental sustainability of systems. Particle trajectory calculation is based on solving motion differential equations considering inertial forces and aerodynamic drag in primary and secondary gas flows. Particles are modeled as spherical objects, with their motion described by a system of first-order ordinary differential equations, solved using the Runge-Kutta-Fehlberg method of 4-5th order with adaptive integration step selection. Obtained results serve as a basis for selecting optimal channel cross-section dimensions, reducing deposit accumulation, and decreasing maintenance frequency, contributing to resource savings and equipment service life extension.

Experimental results. Empirical evaluations were performed on a leased cluster comprising 10 AWS EC2 m7a.4xlarge instances, each equipped with fourth-generation AMD EPYC processors operating at up to 3.7 GHz all-core turbo frequency, 16 virtual CPUs, and 64 GiB of memory. The tasks involved a single series of computations for particle trajectories within square channels. The dispatcher partitioned the workload into subtasks and initiated activations as required.

For identical computations, the execution time on the leased cluster comprising 10 instances was 374s, in contrast to 941s on the local cluster of 4 instances with equivalent configurations; central processing units exhibited 100% utilization in both scenarios, indicating enhanced computational efficiency through increased parallelism under constrained expenditures; upon completion of the experiments, containers were deprovisioned. In comparison with the local cluster of 4 instances (with equivalent configurations), the local arrangement is suitable for preliminary experiments and validation procedures, whereas cloud leasing is advantageous for extensive

computations, as leasing 10 instances for 2h incurs costs of approximately 1-2 USD (configuration-dependent), in contrast to the substantial expenses associated with procuring or configuring additional servers locally.

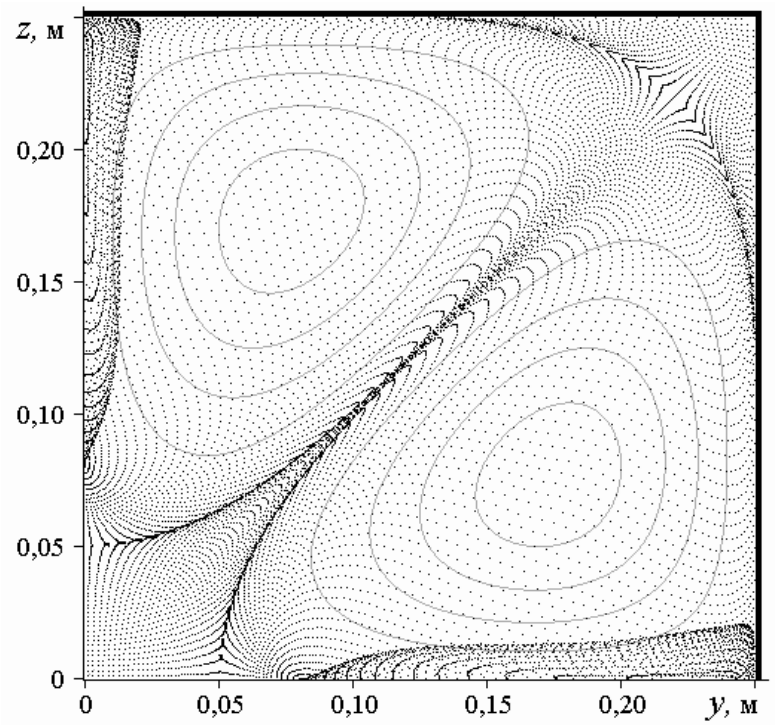

Fig. 2. **Distribution of dust particles across the pipe cross-section at a distance of 10 m from the pipe entrance**

Conclusion. Experimental results confirm the efficiency of the proposed approach, highlighting the utilization of short-term container leasing at low cost. Within the particle dynamics framework, the approach supports parallel processing while mitigating operational expenditures. Limitations encompass dependence on solid-state drives, fixed threshold parameters and 0.5 % overhead from TLS/HMAC protocols. The system demonstrates compatibility with dynamic scaling in AWS and Google Cloud environments, applicable to artificial intelligence, multimedia processing, and modeling tasks, thereby facilitating data center cost reductions. The principal finding underscores the utilization of short-term container leasing at low cost, enabling dynamic resource allocation under constrained financial and infrastructural conditions without performance loss in multi-server systems. Modifications to container management address challenges of heterogeneous workloads. Experiments validate the approach, enhancing throughput. Future developments may include the implementation of hibernation and sleep modes for energy saving and integration with edge computing to improve efficiency. The framework advances sustainable practices in distributed computational environments.

REFERENCES

1. D'Apice C., Dudin A., Dudina O., Manzo R. "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.

2. Kushchazli A., Safargalieva A., Kochetkova I., Gorshenin A. "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.

3. 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.

4. Abu M., Mallick M., Nath R. “Securing the server-less frontier: Challenges and innovative solutions in network security for server- less computing”. 2024; 193: 1–45, https://www.scopus.com/authid/detail.uri?authorId=57103247200.

5. 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.

6. 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 technologies*. 2024; 7: 125–134. DOI: https://doi.org/10.15276/aait.07.2024.9.

7. Surkov S. S., Surkov S. V., Martynyuk O. M., Drozd M. O. “Optimizing Computational Time for Numerical Simulation of Dust-Laden Flows through Queue Management in Multi-Server Environments”. *2024 14th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. 2024. p. 1–6. DOI: https://doi.org/10.1109/DESSERT65323.2024.11122138.

8. Klabnik S., Nichols C. “The Rust Programming Language”. – URL: https://doc.rust-lang.org/book/. – [Accessed: 25.03.2025].

9. Bishop M. “Hypertext transfer protocol version 3 (HTTP/3), IETF RFC 9114”. – URL: https://datatracker.ietf.org/doc/html/rfc9114. – [Accessed: Nov. 2023].

10. Belshe M., Peon R. “Hypertext transfer protocol version 2 (HTTP/2), IETF RFC 7540”. – URL: https://tools.ietf.org/html/rfc7540. – [Accessed: Oct. 2023].

11. Rescorla E. “HTTP over TLS, IETF RFC 2818 (Informational)”. – URL: http://tools.ietf.org/html/rfc2818. – [Accessed: Apr. 2024].

12. Xiang H., Yuan Y., Sheng M., Rui X., Bo W., Tiejun L., Wei J., Lizhou W., Jianmin Z. “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.

13. 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.

14. “Amazon Web Services”. *AWS documentation*. – URL: https://aws.amazon.com/documentation/. – [Accessed: Dec. 2023].

15. “Google Cloud”. G*oogle cloud documentation*. – URL: https://cloud.google.com/docs. – [Accessed: Oct. 2023].

DOI: https://doi.org/10.15276/ict.02.2025.24
УДК 004.76

Контейнеризована оптимізація Dispatcher-Deployer-Executor для короткострокової оренди ресурсів у багатосерверних системах

Сурков Сергій Сергійович[1)]
Аспірант каф. Комп’ютерних інтелектуальних систем та мереж
ORCID: https://orcid.org/0000-0001-9224-7526; k1x0r@ukr.net. Scopus Author ID: 57103247200
Мартинюк Олександр Миколайович[1)]
Канд. техніч. наук, доцент каф. Комп’ютерних інтелектуальних систем та мереж
ORCID: https://orcid.org/0000-0003-1461-2000; anmartynyuk@ukr.net. Scopus Author ID: 57103247200
[1)] Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна

АНОТАЦІЯ

У сучасних розподілених обчислювальних системах архітектури з кількома серверами та чергами операцій забезпечують управління гетерогенними ресурсозатратними навантаженнями, такими як навчання штучного інтелекту, транскодування медіафайлів та обчислювальне моделювання для промислової оптимізації. Традиційні однопроцесорні системи характеризуються послідовними обмеженнями та неефективним розподілом ресурсів, тоді як паралелізм у мультисерверних конфігураціях призводить до підвищеного енергоспоживання. Дане дослідження удосконалює архітектуру Dispatcher-Deployer-Executor (DDE) для оптимізації ресурсоспоживання в реальному часі в мультипроцесорних середовищах. Мета полягає в підвищенні швидкодії виконання операцій та мінімізації вартості виконання для випадків, коли потрібна невелика кількість обчислень та максимальний паралелізм. За допомогою теоретичного моделювання та емпіричної верифікації розширена DDE вводить режим container для Deployer/Executor, які функціонують у контейнерах, забезпечуючи оркестрацію на короткострокову оренду стандартними засобами (наприклад, Docker та Kubernetes). Це реалізує динамічне масштабування ресурсів та ефективне застосування моделі й методу в хмарних середовищах, з основною перевагою швидкого доступу до потужних обчислень при суттєвій економії, оскільки виключається необхідність придбання серверів чи довгострокової оренди для обмежених обсягів розрахунків. Основний сценарій застосування – задачі, орієнтовані на центральний процесор (CPU), оскільки контейнери оптимізовані для CPU-bound обчислень, таких як розрахунок траєкторій частинок пилу шляхом числового рішення диференціальних рівнянь руху. Результати свідчать про суттєве зменшення споживання ресурсів та фінансових витрат порівняно з базовим режимом при збереженні продуктивності; швидка оренда контейнерів забезпечує перевагу в економії витрат та гнучкості. Розширена DDE сприяє сталим обчисленням, балансуючи ефективність, надійність та швидкість для різноманітних задач. Майбутні розширення можуть інтегрувати використання режиму очікування та гібернації для зменшення енергоспоживання.

Ключові слова: багатосерверна архітектура; оптимізація енергоспоживання; гетерогенні навантаження; архітектура DDE; черги завдань; керування станами серверів; Docker, Kubernetes