UDC 004.89
DOI https://doi.org/10.32689/maup.it.2025.3.2

Mykhailo BERDNYK
Doctor of Technical Sciences, Associate Professor,
Professor at the Department of Computer Systems Software,
National Technical University "Dnipro Polytechnic"
***ORCID:** 0000-0003-4894-8995*

Igor STARODUBSKYI
Postgraduate Student at the Department of Computer Systems Software,
National Technical University "Dnipro Polytechnic",
igor.starodubsky@gmail.com
***ORCID:** 0009-0004-1864-7889*

USING MACHINE LEARNING METHODS FOR AUTOMATED CLOUD COMPUTING OPTIMIZATION

Abstract. *This study is devoted to the development and experimental validation of a comprehensive approach to the automated optimization of cloud computing through the use of machine learning methods. The proposed solution – an intelligent adaptive orchestrator – integrates three key components: workload forecasting based on time-series models (LSTM, Prophet), dynamic resource management using reinforcement learning methods (PPO), and an anomaly-detection module employing autoencoders and statistical techniques.*

The aim of the article. *To design, implement, and validate an intelligent adaptive orchestration system that eliminates critical limitations of traditional cloud resource management.*

Scientific novelty. *It lies in the design of a system with a modular architecture that ensures scalability, fault tolerance, and flexible adaptation to diverse business objectives through dynamic tuning of the reinforcement learning agent's reward functions, with integration with container orchestration platforms (e.g., Kubernetes) and support for multi-cloud deployments.*

The conclusions. *Within this study, an intelligent orchestrator for cloud resource management was developed, implemented, and experimentally validated, built on the integration of workload forecasting, reinforcement learning, and anomaly detection methods. Experiments conducted both on a controlled laboratory testbed and in the real industrial hybrid infrastructure of SoftRequest LTD confirmed the high effectiveness of the proposed solution. The practical value of the approach lies in the ability to integrate directly with existing orchestration platforms, such as Kubernetes, without the need for substantial infrastructure rework.*

Key words: *Kubernetes, multi-cloud environments, hybrid clouds, intelligent orchestrator.*

Михайло БЕРДНИК, Ігор СТАРОДУБСЬКИЙ. ВИКОРИСТАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ АВТОМАТИЗОВАНОЇ ОПТИМІЗАЦІЇ ХМАРНИХ ОБЧИСЛЕНЬ

Анотація*. Це дослідження присвячене розробці та експериментальній валідації комплексного підходу до автоматизованої оптимізації хмарних обчислень шляхом застосування методів машинного навчання. Запропоноване рішення – інтелектуальний адаптивний оркестратор – інтегрує три ключові компоненти: прогнозування робочих навантажень на основі моделей часових рядів (LSTM, Prophet), динамічне управління ресурсами за допомогою методів навчання з підкріпленням (PPO) та модуль виявлення аномалій із використанням автоенкодерів і статистичних методів.*

Метою статті *є проєктування, впровадження та валідація інтелектуальної адаптивної оркестраційної системи, що усуває критичні обмеження традиційного управління хмарними ресурсами.*

Наукова новизна *полягає в проекованні системи з модульною архітектурою, яка забезпечує масштабованість, відмовостійкість і гнучкість адаптації до різних бізнес-цілей через динамічне налаштування функцій винагороди агента навчання з підкріпленням, з інтеграцією з платформами оркестрації контейнерів (наприклад, Kubernetes) і підтримка мультихмарних розгортань.*

Висновки*. У межах цього дослідження було розроблено, впроваджено та експериментально валідовано інтелектуальний оркестратор для управління хмарними ресурсами, побудований на інтеграції методів прогнозування навантажень, навчання з підкріпленням і виявлення аномалій. Експерименти, проведені як на контрольному лабораторному стенді, так і в умовах реальної промислової гібридної інфраструктури компанії SoftRequest LTD, підтвердили високу ефективність запропонованого рішення. Практична цінність підходу полягає у можливості прямої інтеграції з наявними платформами оркестрації, такими як Kubernetes, без потреби в суттєвій перебудові інфраструктури.*

Ключові слова: *Kubernetes, мультихмарні середовища, гібридні хмари, інтелектуальний оркестратор.*

The problem statement. The widespread adoption of cloud computing has revolutionized access to computational resources, enabling organizations of all sizes to benefit from scalable, flexible, and cost-effective infrastructure. Cloud platforms allow dynamic provisioning of processing power, memory, storage, and network bandwidth, fostering rapid innovation and efficient operations across industries. However, the rapid growth in the scale and heterogeneity of cloud infrastructures has introduced unprecedented challenges in resource management [3].

Modern cloud environments must handle dynamic, unpredictable workloads driven by varying business demands, user behaviors, and external events. Traditional resource management strategies, predominantly based on static thresholds or manual scaling rules, are often unable to respond adequately to such fluctuations. These mechanisms, while simple and computationally inexpensive, operate reactively rather than proactively, leading to either over-provisioning (increasing operational costs) or under-provisioning (resulting in SLA violations and degraded user experience) [5; 11].

Moreover, cloud ecosystems today are no longer homogeneous. They encompass a variety of resource types, including general-purpose CPUs, specialized GPUs, FPGAs, TPUs, and other hardware accelerators, each optimized for different workloads. Hybrid and multicloud deployments, combining public cloud services with private data centers, further complicate the landscape by introducing diverse resource pools, varying pricing models, and heterogeneous performance characteristics [6; 7].

In addition to dynamic workload patterns, cloud systems face sporadic, high-impact events such as flash sales in e-commerce, viral content surges in media platforms, and end-of-month financial report spikes in banking. These events demand immediate scaling responses that static autoscalers cannot efficiently deliver. Any delay in resource provisioning during these critical periods can severely impact service availability, incur financial penalties, and erode customer trust [10].

Therefore, there is a critical need for intelligent, self-adaptive cloud resource management systems that can anticipate workload changes, make optimized scaling decisions in real time, and enhance resilience against unexpected anomalies or failures.

Analysis of recent studies and publications. Recent studies have consistently demonstrated the inadequacy of traditional threshold-based scaling methods in addressing the demands of dynamic cloud environments. Mao and Humphrey [5] highlighted the limitations of static autoscaling strategies, particularly under variable and unpredictable workloads, where fixed rules fail to optimize cost and meet application deadlines effectively. Similarly, Yazdanov and Fetzer [11] investigated vertical scaling mechanisms and found that simple reactive approaches often suffer from delayed response times and inefficient resource utilization.

In response to these challenges, the integration of machine learning (ML) techniques into cloud resource management has become an active area of research. Xu and Li [10] introduced a dynamic cloud resource management framework leveraging ML models to predict workload variations based on historical usage data. Their findings indicated that learning-driven approaches could significantly improve resource allocation efficiency compared to heuristic-based methods.

Forecasting future workloads is a critical component of intelligent orchestration. Qiu et al. [7] proposed an ensemble of predictive models combining statistical techniques and deep learning approaches to anticipate resource demand in cloud data centers. Their study demonstrated that multi-model ensembles outperform single predictors in terms of accuracy and robustness, especially under mixed and volatile traffic conditions. Similarly, Chen et al. [2] developed an RL-driven autoscaler for web applications that incorporates predictive components to enable proactive scaling decisions, thereby improving both SLA compliance and cost-efficiency.

Reinforcement learning (RL) techniques have emerged as a particularly promising solution for dynamic and autonomous resource optimization. Arabnejad and Barbosa [1] proposed a predictive RL-based autoscaler capable of adjusting resource allocations based on both current and forecasted system conditions. Their results showed that RL agents could dynamically learn optimal policies that balance multiple objectives such as cost reduction, SLA adherence, and energy consumption. Tang and Narasimhan [9] further advanced this field by applying continuous policy gradient methods for RL-based resource management, enabling faster adaptation to fluctuating workloads and heterogeneous resource pools.

Another crucial dimension in cloud orchestration is anomaly detection. Traditional resource management frameworks often ignore system anomalies, which can result in undetected failures and degraded performance. Su et al. [8] addressed this gap by developing an adaptive autoscaling mechanism that integrates deep RL with real-time anomaly detection capabilities. Their approach proved effective in identifying service-level anomalies and triggering appropriate scaling or migration actions to maintain service quality and infrastructure stability.

Comprehensive reviews by Hsu and Chung [3] emphasized that future cloud computing architectures must incorporate machine learning–based orchestration as a fundamental capability, integrating workload

prediction, autonomous decision-making, and anomaly detection into a unified control system. Kunal et al. [4] supported this perspective, demonstrating the effectiveness of combining deep reinforcement learning with predictive analytics to achieve intelligent, context-aware scaling of cloud applications.

Despite these advancements, existing research often treats forecasting, scaling, and anomaly detection as isolated modules rather than components of an integrated system. A holistic approach that unifies these capabilities within a modular, self-adaptive orchestrator remains underexplored. This gap motivates the development of a comprehensive, machine learning–driven orchestration framework capable of enhancing the adaptability, resilience, and economic efficiency of modern cloud infrastructures.

The purpose of the article. The primary goal of this research is to design, implement, and validate an intelligent adaptive orchestration framework that addresses the critical limitations of traditional cloud resource management. The proposed orchestrator integrates three interdependent machine learning components: Workload Forecasting Module: To predict short-term workload changes based on time-series analysis, enabling proactive scaling decisions; Reinforcement Learning–Based Orchestrator: To dynamically optimize resource allocation policies, balancing multiple objectives such as SLA compliance, operational cost reduction, and energy efficiency; Anomaly Detection Module: To monitor system behavior in real time and initiate corrective actions in response to detected anomalies. The orchestrator is designed to operate in real time within heterogeneous, hybrid, and multicloud environments, leveraging data collected from diverse telemetry sources. By employing modular architecture principles, the system ensures extensibility, fault tolerance, and adaptability to evolving infrastructure conditions and business priorities.

The research objectives can be summarized as follows: To develop a robust monitoring and data collection infrastructure capable of capturing fine-grained performance metrics and business-level indicators; to design and train machine learning models tailored for workload prediction and anomaly detection; to formulate the resource management problem as a Markov Decision Process (MDP) and develop a reinforcement learning agent using the PPO algorithm; to integrate the components into a coherent orchestration framework interfacing with container orchestration platforms like Kubernetes; to evaluate the system's performance against traditional autoscaling methods in both controlled laboratory settings and real-world production environments; to provide practical insights and architectural recommendations for the integration of machine learning–based orchestration into existing DevOps workflows. By achieving these goals, the study aims to contribute a scalable, modular, and intelligent solution capable of enhancing the adaptability, efficiency, and resilience of modern cloud computing infrastructures.

Summary of the main material. Cloud computing is a model that enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (such as servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort [3]. Resource management in cloud environments involves monitoring, scaling, allocation, and optimization of computational resources to ensure performance requirements are met while minimizing operational costs.

Machine Learning (ML) is a branch of artificial intelligence focused on developing algorithms that can learn from data and make predictions or decisions without being explicitly programmed [10]. In the context of cloud computing, ML is utilized to forecast workload changes, optimize resource allocation processes, and detect anomalies within complex infrastructure systems.

Workload forecasting involves applying time-series models to predict future resource consumption based on historical usage data. Models such as Long Short-Term Memory (LSTM) networks and Prophet are capable of capturing complex temporal dependencies and seasonal patterns, thus enabling proactive scaling of infrastructure resources [7; 2].

Reinforcement Learning (RL) is a type of machine learning where an agent learns to make optimal decisions through interactions with its environment, receiving rewards for successful actions [1; 9]. In cloud resource management tasks, RL enables the formulation of dynamic scaling and service migration strategies based on multi-objective optimization, balancing factors such as service level agreement (SLA) compliance, operational costs, and energy efficiency.

Anomaly detection focuses on identifying deviations from normal system behavior that may indicate hardware failures, cyberattacks, or configuration errors. Methods based on autoencoders, clustering algorithms, and statistical analyses allow the detection of both evident and subtle anomalies in large volumes of telemetry data [8].

Fundamental Principles for Designing an Intelligent Orchestrator:

– modularity: individual components (forecasting, optimization, anomaly detection) function independently and can be scaled or updated without disrupting the system's integrity;

– adaptability: the system continuously learns from new data and adapts its behavior to evolving environmental conditions;

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

– integration: tight integration with container orchestration platforms (e.g., Kubernetes) to automatically enforce orchestration decisions;

– continuous Optimization: regular retraining and updating of machine learning models based on incoming telemetry data and system feedback to maintain and enhance decision-making quality.

Research Methodology. The research was based on an experimental methodology that combined system design, prototype development, deployment, and multi-level experimental validation. The work was carried out in several distinct stages:

– analytical stage: analysis of the problems of dynamic resource management in cloud environments, identification of limitations in traditional autoscaling methods, and exploration of opportunities to apply machine learning approaches in cloud orchestration;

– design stage: development of a modular architecture for an intelligent orchestrator, including subsystems for telemetry collection, workload forecasting (LSTM, GRU, Prophet), reinforcement learning (PPO algorithm), and anomaly detection (autoencoders and clustering methods);

– implementation stage: deployment of a high-frequency monitoring infrastructure based on Prometheus and Elasticsearch, with data collection intervals of 5–15 seconds; development of online learning mechanisms for forecasting models; and construction of an orchestrator supporting dynamic optimization of scaling policies;

– model training stage: initial training of forecasting models using historical workload data with hyperparameter optimization (Grid Search); training of the reinforcement learning agent under simulated workload scenarios using experience buffer mechanisms to improve training stability;

– experimental validation stage: testing system robustness, adaptability, efficiency, and reliability under various operational scenarios, including load surges, node failures, and changing resource pricing in multicloud environments.

To enhance the reliability of the experimental results, the following were used:

– multiple test runs to smooth out random fluctuations;

– control groups based on standard autoscaling mechanisms (HPA, VPA);

– statistical methods for significance testing (e.g., t-tests).

Experimental Base. The functionality and effectiveness of the developed solution were validated in two environments:

– laboratory tested: a 30-node Kubernetes cluster (each node equipped with 8 vCPUs and 32 GB RAM) and two GPU nodes (NVIDIA Tesla T4) of SotRequest LTD. Workloads were generated using specialized traffic emulators simulating typical behaviors of e-commerce and financial services applications;

– industrial deployment: the financial services company SoftRequest LTD, operating a hybrid cloud infrastructure based on OpenStack and AWS. Real-world workloads included transactional processing, real-time analytics, and periodic training of AI models.

Infrastructure and application monitoring were performed using Prometheus and Elasticsearch, while machine learning models were developed and executed using TensorFlow and PyTorch.

The effectiveness of the developed system was assessed using the following indicators:

– average response time and 95th percentile response time under various workload scenarios;

– SLA violation rate – the percentage of requests that exceeded acceptable response time thresholds;

– resource utilization efficiency – CPU and memory utilization rates during orchestrator operation;

– operational costs – total cloud infrastructure expenses (e.g., AWS costs) and energy consumption in the local infrastructure;

– anomaly reaction time – the time interval between the occurrence of an anomaly and the initiation of corrective actions by the system.

Each metric was analyzed under conditions of normal operation, load surges, infrastructure degradation, and was compared against the performance of traditional autoscaling mechanisms.

The architecture begins with a Telemetry Collector (Node-Exporter, cAdvisor) that scrapes fine-grained CPU, memory and I/O metrics from every Kubernetes node and pod. These raw observations are streamed into Prometheus, while logs flow to Elasticsearch, forming the central TSDB layer. A Feature-Engineering / Streaming ETL service continuously aggregates and enriches the metrics, producing compact feature vectors in near-real time. The vectors feed two specialised ML services: a Workload Forecasting module (LSTM / Prophet) that predicts short-term resource demand, and an Anomaly Detection module (autoencoders plus statistical tests) that flags abnormal behaviour. Both forecast outputs and anomaly flags are consumed by a PPO-based Reinforcement-Learning Optimizer, which synthesises them with the current cluster state to choose scaling, placement or throttling actions. Decisions are forwarded to the Adaptive Orchestrator Control Plane, which translates high-level actions into concrete Kubernetes primitives such as HPA/VPA changes, pod rescheduling or spot-instance requests. Through the Kubernetes API /

Controller Runtime, these commands are applied uniformly across a hybrid runtime that spans an on-prem OpenStack cluster and AWS EKS. Executed actions generate new states and rewards that are written, together with the original metrics, into an Experience Buffer for continual learning. A background Model Trainer / Online Tuning service samples from this buffer to update the forecasting, anomaly-detection and RL models, closing the self-adaptation loop. A separate Load Emulator can inject synthetic traffic into the cluster during laboratory testing to validate policies under extreme or corner-case scenarios. Operators observe the entire system through Grafana dashboards fed directly by Prometheus (metrics) and Kibana views backed by Elasticsearch (logs). The dashboards also surface the applied policies coming from the Control Plane, providing full transparency into why and when the orchestrator scaled or migrated workloads. Overall, the design realises a vertically integrated feedback cycle – observe → predict & detect → decide → act → learn – enabling proactive, cost-aware and resilient resource management across multicloud environments.

Technical Implementation. The production-grade prototype was deployed as fourteen containerised micro-services on Kubernetes, each packaged with locked, reproducible Helm charts. At the foundation, a high-frequency observability stack polls every node and pod with Node-Exporter, cAdvisor and kube-state-metrics; raw metrics arrive in Prometheus at five-second granularity while structured logs land in an OpenSearch cluster configured with hot-warm tiering so that recent indices remain on NVMe storage and older shards migrate to cost-optimised disks. Thanos remote-write off-loads up to one-and-a-half years of historical data to an S3-compatible MinIO bucket, ensuring that the modeller can replay the entire operational history without impacting online queries.

A Python FastAPI service, built around the Faust streaming framework, subscribes to the metrics_raw Kafka topic, applies windowed aggregations at five seconds, one minute and fifteen minutes, computes derivative features such as CPU deltas, exponential-weighted moving averages and trend slopes, then serialises the resulting vectors with Apache Arrow before publishing them to the features_v1 topic. Sustained throughput in production is roughly two megabytes per second, and in end-to-end tests the p-99 latency between a Prometheus scrape and a feature message entering the machine-learning layer is below one second.

Forecasting is served by an ensemble endpoint that multiplexes two distinct models: a Prophet instance tuned to capture daily and weekly periodicity with a changepoint prior of 0.2, and a two-layer LSTM built in TensorFlow 2.16 with 128 hidden units per layer and a dropout of 0.2. Hyper-parameters were discovered by a grid search over learning rates and look-back windows using a ninety-day sliding training set; retraining triggers automatically at midnight or whenever the weighted absolute percentage error exceeds fifteen per cent. Inference runs in TensorFlow Serving with a batch size of sixteen and maintains a p-95 latency of forty-five milliseconds per series.

An anomaly-detection micro-service hosts a symmetric dense auto-encoder, forty-eight hours of “healthy” data are replayed nightly to refresh its weights, and the reconstruction error threshold (mean plus three standard deviations) is recalibrated at the end of each run. A lightweight z-score guard operates in the same container so that sudden spikes can be flagged within hundreds of milliseconds, even if the auto-encoder inference queue backs up. Both anomaly flags and point forecasts are appended to the state vector consumed by the reinforcement-learning agent.

Resource allocation is framed as a continuous-action Markov Decision Process and solved with Proximal Policy Optimisation in Ray RLlib 3.0. The fifty-four–dimensional state contains current utilisation, forecast residuals, anomaly indicators, spot-price fluctuations and energy-tariff signals; the three-dimensional action proposes a replica-scaling multiplier between 0.5 and 2.0, a CPU-limit shift between –1 and +1 core, and a node-affinity score that steers the scheduler towards GPU or spot nodes. Eight learner workers execute in parallel on separate vCPUs, while the central parameter server uses a single NVIDIA T4 GPU for policy updates. A Redis-backed experience buffer can hold two million tuples; as soon as four thousand new transitions accumulate – typically about thirty seconds – the agent performs an on-policy update with a clip parameter of 0.2 and a GAE lambda of 0.95.

Decisions emitted by the agent enter a custom Go controller that reconciles an AdaptivePolicy custom resource. The controller translates abstract actions into concrete Kubernetes primitives: it adjusts HPA and VPA targets, annotates pods to trigger rescheduling, calls kubectl drain for live migration and interacts with the AWS EC2 Fleet API to request spot instances. Every change rolls out through a guarded canary stage that initially exposes only one per cent of user traffic; if p-95 latency remains below one hundred and ten per cent of baseline for two reconcile loops, the controller ramps exposure by ten per cent per minute; otherwise, it rolls back and pins the policy revision for post-mortem analysis.

A nightly Kubeflow Pipelines workflow pulls fresh experience data, launches distributed training jobs on two dedicated GPU nodes, performs Optuna-driven hyper-parameter optimisation for the predictive models, version-tags the artefacts with Data Version Control hashes and uploads them to the MinIO model store.

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

Newly trained models are shadow-served for fifteen minutes with ten per cent traffic replay; promotion occurs automatically if the weighted absolute percentage error rises by no more than two percentage points and the reward trend stays positive. Model-drift rules in Prometheus watch the production error metrics and push alerts to Slack when WAPE exceeds eighteen per cent for a sustained quarter hour.

Continuous integration runs in GitLab and executes linting, unit tests, container builds with BuildKit, Trivy image scanning and Helm-chart rendering. Staging deployments occur on an isolated thirty-node cluster where Locust and K6 replay week-long production traces; a pipeline succeeds only if average response time stays within five per cent of baseline while infrastructure cost, computed from mock AWS billing data, falls by at least three per cent. The median pipeline duration is fourteen minutes, and mean time to restore after a failed deployment is under twelve minutes thanks to automated rollbacks and pre-baked golden images.

Security and compliance are enforced with Vault-backed CSI drivers for short-lived secrets, OPA Gatekeeper policies that block privileged or host-PID pods, and cosign signatures that guarantee supply-chain integrity from source commit to running image digest. All inter-service traffic is encrypted with Istio mutual TLS, and cross-cluster links use WireGuard tunnels. Infrastructure spans an eighteen-node private OpenStack cloud for latency-critical workloads and a twelve-node AWS EKS footprint that mixes on-demand and Graviton-powered spot instances for cost-efficient burst capacity; Calico enforces network policies across both sites.

Benchmarking on the production workload showed that the orchestrator lowers average response time by thirty-three per cent, halves SLA violations and cuts cloud spend by roughly twelve per cent over a rolling thirty-day window. Anomaly mean-time-to-detect is about six seconds and mean-time-to-mitigate under fifteen. Collectively, these results confirm that the observe-predict-optimise-act-learn feedback loop operates fast enough to out-perform traditional HPA/VPA scaling while remaining safe, auditable and cost-aware in a heterogeneous multicloud environment.

Performance Evaluation Results. The experimental evaluation was conducted in two environments: on a controlled laboratory testbed at SoftRequest LTD and in its real-world production hybrid cloud infrastructure.

Key performance indicators (KPIs) used for evaluation included:

– average and 95th percentile service response time;

– SLA compliance rate;

– resource utilization efficiency;

– trends in operational cost dynamics;

– anomaly detection and response speed.

The results demonstrated that the proposed architecture ensures high application stability even under sudden and significant workload changes.

System Response Time. Comparative analysis showed a significant reduction in both average response time and the 95th percentile (p95) compared to traditional Kubernetes scaling mechanisms.

Results recorded on the SoftRequest LTD testbed:

– average response time was reduced by 28% compared to HPA;

– p95 response time was reduced by 31% under peak load conditions.

In real-world production at SoftRequest LTD:

– average response time decreased by 30–35% during typical business workloads;

– the proportion of requests processed under 200 ms increased from 92% to 97%, significantly improving the end-user experience.

Thus, proactive workload forecasting enabled minimizing response latency through advance resource scaling.

SLA Violations. Service Level Agreement (SLA) violations are a critical indicator of the quality of cloud infrastructure services. Even short-term SLA breaches can result in penalties and reputational damage.

During the experiments:

– with traditional autoscaling mechanisms (HPA/VPA), the SLA violation rate was approximately 6.2%;

– after deploying the intelligent orchestrator, the violation rate dropped to 2.7%.

These findings are summarized in Table 1.

The reduction of SLA violations by more than half demonstrates the high predictive capabilities of the developed models and the system's ability to react before service degradation occurs.

Table 1

Comparison of SLA Violation Rates

Environment	SLA Violation Rate (Baseline)	SLA Violation Rate (Orchestrator)
Laboratory Testbed	6.5%	2.8%
Production Environment	6.0%	2.6%

Cost Efficiency. Economic efficiency is an essential factor when considering the adoption of new cloud technologies.

During a 30-day observation period:

– infrastructure costs were reduced by 10–15% compared to baseline autoscaling mechanisms.

Main cost-saving factors included:

–prevention of unnecessary resource over-provisioning during short-term peaks;

–automatic utilization of cheaper spot instances during periods of low activity;

rapid deallocation of underutilized resources after load reductions, minimizing charges for idle capacity.

Thus, the implementation of the intelligent orchestrator resulted not only in improved performance metrics but also in significant operational cost optimization.

Anomaly Detection and Reaction. In dynamic multicloud environments, the ability to quickly detect and resolve anomalies is critical to ensuring business continuity.

The anomaly detection module achieved the following:

– average anomaly detection time was 5–7 seconds, more than twice as fast compared to traditional threshold-based monitoring tools;

– response time – the interval between detection and corrective action initiation – remained below 15 seconds.

Table 2 provides examples of typical anomalies detected and the corresponding mitigation actions.

Table 2

Examples of Detected Anomalies and Response Times

Anomaly Type	Detection Time (s)	Response Time (s)	Mitigation Measures
CPU Saturation	5	12	Pod rescheduling
Storage Latency Spike	6	13	Volume migration
DDoS Attack Simulation	7	14	Temporary scaling action

By identifying potential failures at early stages, the system demonstrated high resilience even under heavy and unpredictable workloads.

Key Architectural Advantages. The comprehensive architecture of the developed solution achieved a synergistic effect through:

– integration of forecasting models for advance resource scaling;

– use of reinforcement learning agents to develop optimal resource management strategies under multi-criteria constraints;

– deployment of real-time anomaly detection mechanisms to ensure early failure warning and incident mitigation.

The modular structure of the architecture enabled flexible adaptation of the system to different cloud platforms and existing DevOps workflows without requiring substantial changes to the customer's infrastructure.

Summary of Quantitative Results. The summary of experimental results confirms:

– reduction of average response time by up to 35%;

– decrease in SLA violations by more than 50%;

– reduction of infrastructure operational costs by 10–15%;

– shortening of anomaly reaction time by more than 40% compared to conventional monitoring solutions.

Thus, the developed intelligent orchestrator demonstrated its high effectiveness for deployment in dynamic, heterogeneous, and multicloud environments requiring flexibility, scalability, and high resilience.

Conclusions and prospects for further research. Within the framework of this study, an intelligent orchestrator for cloud resource management was developed, implemented, and experimentally validated, based on the integration of workload forecasting, reinforcement learning, and anomaly detection methods.

The experiments conducted both on the controlled laboratory testbed and in the real-world production hybrid infrastructure of SoftRequest LTD confirmed the high effectiveness of the proposed solution. The key achieved results include:

– a reduction of average service response time by up to 35%;

– a decrease in SLA violation rate by more than 50%;

– a reduction of infrastructure operational costs by 10–15%;

– an improvement in anomaly detection and reaction time by over 40% compared to traditional methods.

The developed architecture demonstrated the ability to efficiently adapt to changing operating conditions, maintained system resilience during load surges, and ensured cost-effective resource utilization without compromising service quality.

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

The practical value of the proposed approach lies in its ability to integrate directly into existing orchestration platforms, such as Kubernetes, without requiring a major overhaul of the infrastructure. This enables enterprises and service providers to minimize implementation costs and significantly enhance the level of operational automation.

Despite the positive results obtained, the study opens several promising directions for future research:

– expanding the forecasting functionality by incorporating external factors (e.g., seasonal demand fluctuations, marketing events, changes in resource pricing by cloud providers);

– developing adaptive self-learning systems that enable the orchestrator to autonomously optimize its models in response to shifts in workload profiles or service architectures without human intervention;

– integrating energy consumption forecasting models to build energy-efficient scaling strategies and reduce the carbon footprint of cloud infrastructures;

– designing security-focused anomaly detection mechanisms aimed at real-time identification of cyberattacks or data leakage attempts;

– applying graph neural networks (GNNs) for more accurate analysis of network relationships between application components and for optimizing service placement based on network topology awareness;

– creating comprehensive multicloud optimization strategies that simultaneously consider cost, performance, energy efficiency, and reliability across different cloud providers.

In the future, the development of such intelligent orchestrators could become a cornerstone in building fully autonomous, self-learning cloud platforms capable of real-time adaptation to global shifts in operational conditions, business priorities, and quality of service requirements.

Thus, the results obtained not only demonstrate the scientific and practical significance of the proposed approach but also lay a strong foundation for continued research in the field of intelligent cloud systems automation.

Bibliography:

1. Arabnejad H., Barbosa J. Predictive Reinforcement Learning-Based Autoscaler for Cloud Resource Provisioning. *Journal of Grid Computing*. 2020. Vol. 18, No 4. P. 761–777. (date of access: 14.09.2025).

2. Chen H., et al. Intelligent Autoscaling for Web Applications in the Cloud via Reinforcement Learning. *IEEE Transactions on Services Computing*. 2021. Vol. 14, No 5. P. 1347–1359. (date of access: 14.09.2025).

3. Hsu C.-H., Chung Y. (Eds.). Cloud Computing and Big Data: Technologies, Applications and Security. Springer. 2021. (date of access: 14.09.2025).

4. Kunal T., Singh P., Rathor S., He H. Resource Scaling for Cloud Applications Using Deep Q-Learning. Proceedings of the 2022 International Conference on Cloud Computing and Big Data Analytics (ICCCBDA). 2022. P. 39–47. (date of access: 14.09.2025).

5. Mao M., Humphrey M. Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC). 2011. P. 1–12. DOI: https://doi.org/10.1145/2063384.2063449

6. Mao Y., Li J., Humphrey M. Cloud Auto-Scaling with Machine Learning. Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). 2018. P. 108–117. (date of access: 14.09.2025).

7. Qiu T., Zhang L., Ghoneim A., Li W., Cai W. Prescience-Based Resource Scaling for Dynamic Workloads in Cloud Datacenters Using Ensemble Forecasting Techniques. *Future Generation Computer Systems*. 2019. Vol. 101. P. 1209–1221. (date of access: 14.09.2025).

8. Su X., Wen S., Su J., Wang J. Adaptive Autoscaling Mechanism Based on Deep Reinforcement Learning for Heterogeneous Cloud Services. Concurrency and Computation: *Practice and Experience*. 2022. Vol. 34, No 11. e6806. (date of access: 14.09.2025).

9. Tang Q., Narasimhan G. A Reinforcement Learning Approach to Efficient Resource Allocation in Cloud Computing. Proceedings of the 2021 IEEE International Conference on Cloud Engineering (IC2E). 2021. P. 45–54. (date of access: 14.09.2025).

10. Xu H., Li B. Dynamic Cloud Resource Management via Machine Learning. *IEEE Transactions on Parallel and Distributed Systems*. 2017. Vol. 28, No 1. P. 147–160. (date of access: 14.09.2025).

11. Yazdanov A., Fetzer C. Vertical Scaling for Cloud Applications. Proceedings of the 2014 IEEE 8th International Symposium on Service Oriented System Engineering (SOSE). 2014. P. 318–325. (date of access: 14.09.2025).

Дата надходження статті: 18.09.2025
Дата прийняття статті: 20.10.2025
Опубліковано: 04.12.2025