

UDC 004.75:519.87(045)

DOI:10.18372/1990-5548.89.21482

¹Oleksandr Rolik,²Valerii Kolesnik

RESOURCE MANAGEMENT METHOD FOR SELF-HEALING CRITICAL IT INFRASTRUCTURE IN A MULTIDOMAIN ENVIRONMENT

Department of Information Systems and Technologies,
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine
E-mails: ¹o.rolik@kpi.ua ORCID 0000-0001-8829-4645
²vkolesnik.kpi@gmail.com ORCID 0000-0001-9179-1335

Abstract—This article solves the scientific and practical problem of ensuring the reliability and quality of information service provisioning in distributed computing environments by developing a resource management method for self-healing critical IT infrastructure. The functioning processes of the IT infrastructure management system), which structurally and logically acts as the customer's internal IT department, are investigated. An approach to categorizing information services and formalizing dynamic Service-Resource Models is proposed, allowing the synthesis of independent duplicate structures for critical category services across multiple domains. The autoscaling and weighted load balancing algorithms are improved by applying the determination of the Pareto-optimal solution set and using the minimum normalized Euclidean distance to the ideal point within the solution set. The Pareto multicriteria optimization method is substantiated, based on mathematically reducing the deviation of actual reliability and response time metrics from their target and normative values to zero. The architectural scheme of the management system is formed, strictly separating the processes of service quality evaluation by monitoring tools and quality level management by the analytical module. Experimental modeling confirmed the ability of the developed system to isolate resource failures and adapt to the nonlinear growth of requests, performing component migration and topology reconfiguration without system downtime.

Keywords—Pareto-optimal solutions, service-resource model, multicriteria optimization, IT service management, information systems.

I. INTRODUCTION

Ensuring the continuity of information service provisioning in distributed computing environments is a fundamental task of IT infrastructure management. Modern Corporate Customers require a guaranteed level of service for a significant number of users, while the physical resources where the functional components of the services are deployed remain objectively unreliable. The situation becomes more complex when the information system utilizes the resources of several domains, some of which are owned by the customer, and the rest are provided by external providers [1].

The concept of a self-healing critical IT infrastructure is applied to solve this problem [2], [3]. Such infrastructure can autonomously identify failures, localize them, and return to normal operation without human intervention. These systems possess four primary characteristics according to [4]: self-configuration, self-healing, self-optimization, and self-protection. The main idea consists in using a feedback loop where the system continuously monitors its state, detects deviations, plans appropriate actions, and triggers actions to correct the critical situation without the need for human intervention.

Resource management in such a multidomain environment is performed by a specialized IT infrastructure management system (ITIMS). A scenario is considered where the ITIMS performs the functions of the customer's internal IT department, possesses complete information about the system state, continuously monitors performance metrics, and makes global deterministic decisions regarding load redistribution, reconfiguration, and self-healing [5]. In the event of a failure, the ITIMS isolates the faulty segment, transfers computational loads to operational domains, and restores the damaged structures without system downtime.

II. LITERATURE REVIEW

Existing studies propose various approaches to increasing reliability and fault tolerance in distributed computing environments. In article[6] authors define dependability as the global ability of a computing system to deliver service that can justifiably be trusted, categorizing dependability threats into a fundamental causal chain of faults, errors, and service failures. To counter these threats, fault tolerance operates through error handling and fault handling. Complementing this framework author of [7] models system reliability, availability,

and queueing behavior, demonstrating that the imperfect fault coverage parameter strongly dictates overall system survivability, as uncovered faults lead directly to immediate failure. However, computational node reservation algorithms and theoretical dependability models treat system components abstractly and do not take into account the heterogeneity of the network topology at the level of individual domains.

For virtualized cloud data center infrastructures research in [8] explores power-aware dynamic virtual machine consolidation via live migration to minimize electrical energy consumption. Through competitive analysis, According to [8] authors establish theoretical upper bounds for online deterministic algorithms and introduce adaptive host-overloading detection heuristics – such as local regression and median absolute deviation – to optimize the economic trade-off between energy costs and SLA violation penalties. Nevertheless, these traditional load balancing methods predominantly neglect economic constraints under domain-specific network topologies as well as proactive self-healing mechanisms.

In parallel, works focus on load balancing methods and resource allocation across distributed system architectures. The research [9] examine structural challenges in distributed systems – including component heterogeneity, failure handling, and concurrency – noting that load sharing and load balancing strategies collect node utilization data to assign processes and avoid performance bottlenecks across physical resources.

Among possible approaches to proactive management, a deep reinforcement learning mechanism for application in Internet of Things (IoT) networks to recover from failures was investigated by [10]. Framing network self-healing as a Markov decision process, authors in [10] deployed deep Q -networks with experience replay, where IoT devices autonomously take recovery actions based on state feedback. Simulation evaluations demonstrated improvements in fault recovery time, network downtime, device energy consumption and throughput retention during disruptions. This allowed improving the recovery speed compared to a conventional reactive approach; however, this study did not provide a comparison with other similar solutions and faced computational overhead on resource-constrained IoT hardware.

III. PROBLEM STATEMENT

Based on the analysis of existing solutions, the problem of comprehensive reliability management, considering the categorization of information services and the necessity of multicriteria

optimization, remains unresolved and highly relevant.

The fundamental problem lies in the complexity of ensuring a guaranteed level of service in a multidomain environment where the underlying physical resources are objectively unreliable. To overcome the limitations of purely reactive approaches and uncoordinated scaling, a comprehensive algorithmic mechanism is required. This mechanism must be capable of proactively managing the quality level, isolating resource failures, and executing Pareto multicriteria optimization of computational loads instantaneously, during operation, without system downtime.

IV. THE AIM AND OBJECTIVES OF THE RESEARCH

This study aims to develop a method for ensuring the reliability and quality of information service provisioning in a self-healing multidomain IT infrastructure based on Service-Resource Models and finding the Pareto set to select optimal configurations during automatic resource migration and recovery.

To achieve this aim, the following objectives must be accomplished:

1) *Formalize a dynamic Service-Resource Model (SRM)* for a multidomain environment to mathematically describe the dependencies between information services and physical resources.

2) *Propose an approach* to categorizing information services and synthesizing independent duplicate structures to ensure the high availability of critical services.

3) *Develop a Pareto multicriteria optimization method* for resource management that balances response time, reliability, and operation cost.

4) *Improve autoscaling and weighted load balancing algorithms* to distribute requests and manage capacities during operation.

5) *Formulate the architectural scheme* of the IT infrastructure management system (ITIMS), separating the monitoring processes from the analytical quality management processes.

6) *Conduct experimental modeling* to validate the ability of the developed method to isolate resource failures and adapt to nonlinear load growth without system downtime.

V. RESOURCE MANAGEMENT METHOD FOR SELF-HEALING CRITICAL IT INFRASTRUCTURE

A. Formalization of the Service-resource Model

To mathematically describe the service provisioning processes, the SRM approach [11] is used, which reflects the structural and functional

dependencies between information services and physical resources. The model is dynamic because the ITIMS continuously monitors the state of resources and alters the topology to ensure self-healing and maintain service quality at the agreed level.

Let a set of information services $\mathcal{S} = \{s_1, s_2, \dots, s_I\}$ provided to users, and a set of available infrastructure domains $\mathcal{D} = \{d_1, d_2, \dots, d_J\}$ be given, among which are domains where the information system components are located. A portion of the domains is owned by the critical infrastructure owner, and a portion belongs to providers or operators. Each domain $d_j \in \mathcal{D}$ is characterized by its own pool of unreliable resources $\mathcal{R}_j = \{r_{j,1}, r_{j,2}, \dots, r_{j,K}\}$.

Each service $s_i \in \mathcal{S}$ is conceptually represented as a directed acyclic graph $\mathcal{G}_i = (\mathcal{N}_i, \mathcal{E}_i)$, where $\mathcal{N}_i$ is a set of logical functional components, which are essentially considered as the nodes of the directed graph, and $\mathcal{E}_i$ is a set of information links between them that determine the request processing sequence; accordingly, these links are defined as the edges of the same graph.

The mapping of the service's logical topology onto the physical infrastructure is defined by the placement function $\mathcal{M}$:

$$\mathcal{M}(t): \mathcal{N}_i \rightarrow \bigcup_{i=1}^J \mathcal{R}_i. \quad (1)$$

The dependence of function $\mathcal{M}$ on time t formalizes the system's self-healing capability. Depending on the architectural requirements, the components $\mathcal{N}_i$ of a specific service can be placed exclusively in one domain, or distributed among several domains $\mathcal{D}^* \subset \mathcal{D}$. If a resource fails, the services associated with it cease to operate; the ITIMS recalculates the placement function and transfers the components to another node to restore the graph $\mathcal{G}_i$ in order to determine the resources and links for continuous service provisioning. For the problems of resource allocation and reallocation according to the problem described above, the Kubernetes environment provides a full range of available tools, as well as the ability to override resource allocation behavior. It is worth noting that Kubernetes can adapt quickly to various requirements. For example, if a web application has a few visitors over a period of time, Kubernetes can reduce the number of active pods, utilizing the

minimum required IT infrastructure's resources to save costs [12].

B. Service Categorization and Synthesis of Duplicate Structures

Information services provided to Corporate Customers with a large number of users have varying levels of impact on business outcomes. Furthermore, regarding critical services, the number of users and the impact of providing such services on aspects of health, life, government functioning, etc., directly affect the reliability requirement for the environment providing these information services, particularly the critical IT infrastructure. To optimize resource utilization, the set $\mathcal{S}$ is divided into two disjoint categories: $\mathcal{S} = \mathcal{S}_{NC} \cup \mathcal{S}_C$.

Category $\mathcal{S}_{NC}$ encompasses services whose functioning does not affect business decision-making. The quality of these services does not have a determining impact on the overall Quality of Experience (QoE). Structural duplication is not applied for $s \in \mathcal{S}_{NC}$ services. In the event of a domain failure, service provisioning stops, and its self-healing occurs by reinitializing the components on new resources after the ITIMS records the failure.

Category $\mathcal{S}_C$ includes critical services for which failure is unacceptable. For this group, ensuring high availability through spatial redundancy is required. If a normative availability A_{SLA} is set for a service $s \in \mathcal{S}_C$, and the basic reliability of a single domain is $A_d < A_{SLA}$, the ITIMS forms N independent structural copies of components or infrastructure fragments based on the graph $\mathcal{G}_i$ in different domains. Self-healing in this case occurs in two stages: first, the ITIMS switches all requests to an operational copy immediately upon detecting that the service is down, without system downtime, and then it restores the lost copy in a new domain in the background to maintain the required value N . Assuming domain failures are statistically independent events, the probability of simultaneous failure of all N copies equals $(1 - A_d)^N$ [13], [14]. The requirement for fulfilling the service level agreement is written as:

$$1 - (1 - A_d)^N \geq A_{SLA}. \quad (2)$$

The necessary number of duplicate structures is determined by the formula:

$$N = \left\lceil \frac{\ln(1 - A_{SLA})}{\ln(1 - A_d)} \right\rceil. \quad (3)$$

Through continuous monitoring, the ITIMS guarantees that the number of active copies never falls below the calculated value N over an extended period [15].

C. Pareto Multicriteria Optimization Method

The process of selecting domains for the initial placement of service components, autoscaling when the number of users increases or other conditions for quality degradation appear, and selecting a target resource during self-healing is formalized as a multicriteria optimization problem.

Let $\mathcal{X}$ be the space of feasible infrastructure configurations.

Abandoning infinite response time minimization and infinite reliability maximization is a necessary condition for the rational use of computing capacities. The optimization performed by the ITIMS is based on minimizing the difference between the actual Service Level Indicator (SLI) and its Service Level Objective (SLO) or normative value according to the Service Level Agreement (SLA) [16]. Since the target and normative indicators are unique for each service $s_i \in \mathcal{S}$.

For any configuration $x \in \mathcal{X}$, strict feasibility constraints are introduced, which must be satisfied for all services:

$$SLI_{T,i}(x) \leq SLO_{T,i}, \forall i \in \{1, \dots, I\}, \quad (4)$$

$$SLI_{R,i}(x) \geq SLA_{R,i}, \forall i \in \{1, \dots, I\}, \quad (5)$$

where $SLI_{T,i}(x)$ is the actual system response time for service s_i , $SLO_{T,i}$ is the target response time for service s_i , $SLI_{R,i}(x)$ is the actual reliability of the configuration for service s_i , and $SLA_{R,i}$ is the normative service provisioning reliability. The provisioning reliability of an individual service must not be worse than specified in $SLA_{R,i}$.

Each configuration variant x for service s_i is evaluated by a criteria vector:

$$\mathcal{V}_i(x) = \langle \Delta T_i(x), \Delta R_i(x), C_i(x) \rangle, \quad (6)$$

where $\Delta T_i(x) = SLO_{T,i} - SLI_{T,i}(x)$ is the response time margin for service s_i ($\Delta T_i(x) \rightarrow \min$), $\Delta R_i(x) = SLI_{R,i}(x) - SLA_{R,i}$ is the reliability margin for service s_i ($\Delta R_i(x) \rightarrow \min$), and $C_i(x)$ is the operation cost of the resources involved in providing service s_i ($C_i(x) \rightarrow \min$). Accordingly, the

difference between the actual value and the target value should approach zero.

Since the criteria remain contradictory (cost minimization contradicts the reduction of deviations due to the need to involve additional domains), the ITIMS searches for the Pareto set $\mathcal{P}_i$ for each service s_i , consisting of nondominated alternatives.

Configuration x_1 dominates x_2 ($x_1 \prec x_2$) if, for all three criteria, the values of x_1 are no greater than x_2 , and for at least one criterion, x_1 is strictly smaller. The set of optimal configurations is defined as:

$$\mathcal{P}_i = \{x \in \mathcal{A} \mid \nexists x' \in \mathcal{X} : x' \prec x\}. \quad (7)$$

When the number of users increases or the need for self-healing after a failure arises, the ITIMS analyzes the set $\mathcal{P}_i$ and performs component migration and load redistribution during operation, without system downtime. This guarantees that the deviation of actual parameters from the set targets approaches zero without excessive resource consumption, and the infrastructure retains the capability for continuous self-healing.

D. Balancing and Autoscaling Method

The functioning of a self-healing infrastructure requires a strict load distribution and resource management algorithm. The developed method is based on the continuous interaction of ITIMS modules and consists of three sequential stages.

Stage 1. Current load distribution. The load balancer receives the total stream of user requests for each service s_i and directs each individual request strictly to one of the available domains. To prevent local response time degradation, the distribution is performed using a weighted algorithm that takes into account the residual computing capacity of the domains.

Let $\mathcal{D}_i^* \subseteq \mathcal{D}$ be the subset of domains where active copies of service s_i provisioning structures are deployed. The monitoring module continuously collects infrastructure state data and, for each domain $d_j \in \mathcal{D}_i^*$, determines the maximum computing capacity $\Pi_j^{\max}$ and the currently utilized capacity $\Pi_j^{\text{util}}(t)$ at time t .

The residual computing capacity of domain d_j is calculated as:

$$\Delta \Pi_j(t) = \Pi_j^{\max} - \Pi_j^{\text{util}}(t). \quad (8)$$

The weight coefficient $w_{i,j}(t)$, which determines the priority of domain d_j when processing requests for service s_i , is calculated as the ratio of this domain's residual capacity to the total residual capacity of all domains serving this service:

$$w_{i,j}(t) = \frac{\Delta\Pi_j(t)}{\sum_{k \in \mathcal{D}_i^*} \Delta\Pi_k(t)}. \quad (9)$$

For a correct probability distribution, the mandatory normalization condition is satisfied:

$$\sum_{j \in \mathcal{D}_i^*} w_{i,j}(t) = 1. \quad (10)$$

Since each user request is a discrete information packet directed to only one domain, the weight coefficient $w_{i,j}(t)$ is used by the load balancer as the routing probability for the next incoming request to domain d_j . Such an algorithm guarantees that domains with a larger volume of free resources receive a proportionally higher load. The updating of weight coefficients based on data from the monitoring module and the corresponding change in routing logic are performed automatically by the load balancer during operation, without system downtime. This allows keeping the actual response time $SLI_{T,i}$ within the set target value $SLO_{T,i}$.

Stage 2. Intelligent resource autoscaling. During the operation of the information system, the number of users and the intensity of their requests are subject to dynamic changes. The monitoring module continuously records these fluctuations, performing current service quality evaluations. To manage the quality level without excessive consumption of computing capacities, the ITIMS abandons static threshold algorithms in favor of reinforcement learning methods. This allows the system to continuously learn and adapt to changing conditions, ensuring precise allocation of infrastructure capacities.

The decision-making process regarding scaling is formalized as a Markov decision process. The state of the environment for an individual service s_i at time t is described by the vector $\mathcal{Z}_i(t)$:

$$\mathcal{Z}_i(t) = \langle SLI_{T,i}(t), \lambda_i(t), C_i(t) \rangle, \quad (11)$$

where $SLI_{T,i}(t)$ is the actual response time, $\lambda_i(t)$ is the intensity of incoming user requests, and $c_i(t)$ is

the current volume of allocated resources across several domains.

The analytical module generates an action vector $u_i(t)$, which involves both the allocation of new resources and the removal of excess resources through spatial and quantitative regulation:

$$\mathcal{U}_i(t) = \langle \Delta\mathcal{H}_i(t), \Delta\mathcal{V}_i(t) \rangle, \quad (12)$$

where $\Delta\mathcal{H}_i(t)$ is the horizontal scaling parameter (change in the number of structural copies in domains), and $\Delta\mathcal{V}_i(t)$ is the vertical scaling parameter (change in the computing capacity of existing nodes). Positive values of the vector parameters correspond to the allocation of additional resources, negative values to the release of unused ones.

The ITIMS continuously optimizes the management policy $\pi: \mathcal{Z}_i \rightarrow \mathcal{U}_i$, maximizing the reward function $\mathcal{W}_i(t)$. This function strictly penalizes the agent for both quality degradation and excessive infrastructure cost:

$$\mathcal{W}_i(t) = -(\alpha \cdot \max(0, SLI_{T,i}(t) - SLO_{T,i}) + \beta \cdot C_i(\mathcal{Z}_i(t))), \quad (13)$$

where α and β are weight coefficients for the significance of the respective criteria. Depending on the degree of service quality degradation or the increase in the number of users, the trained analytical model determines the strictly necessary amount of resources $u_i^*(t) = \pi(\mathcal{Z}_i(t))$. The generated action is compared with the Pareto set $\mathcal{P}_i$ of nondominated solutions to guarantee compliance with the normative reliability $SLA_{R,i}$.

After approving the action vector, the migration management module executes the deployment of additional components or their controlled removal. Information about the updated topology is transmitted to the load balancer. All processes of changing computing capacities are performed during operation without system downtime. In this case, both horizontal and vertical scaling are possible [17], depending on the Pareto-optimal solution selected for further implementation in the IT infrastructure.

Stage 3. Infrastructure self-healing after domain failure. In the event of a hardware or software error in one of the domains $d_f \in \mathcal{D}_i^*$ at time t_f , the monitoring module performs quality evaluation and records a drop in the actual reliability metric $SLI_{R,i}(t_f)$ below the normative value $SLA_{R,i}$. To prevent service degradation, the self-healing

algorithm is implemented through a strict sequence of mathematical transformations and management decisions.

1) *Failure isolation and traffic redistribution.* The load balancer excludes domain d_f from the set of active nodes: $\mathcal{D}_i^*(t_f^+) = \mathcal{D}_i^*(t_f) \setminus \{d_f\}$. The routing weight coefficient for the faulty domain is forcefully zeroed out ($w_{i,f}(t_f^+) = 0$). For operational domains, the request distribution probabilities are recalculated based on the residual computing capacity determined in Stage 1:

$$w_{i,j}(t_f^+) = \frac{\Delta \Pi_j(t_f^+)}{\sum_{k \in \mathcal{D}_i^*(t_f^+)} \Delta \Pi_k(t_f^+)}, \forall j \in \mathcal{D}_i^*(t_f^+). \quad (14)$$

2) *Quality level management and target resource selection.* To restore the required number of independent structural copies N , the analytical module manages the quality level. From the current Pareto set $\mathcal{P}_i$, an optimal backup domain (or several domains, since a set of available domains is used for experiments and full deployment, rather than just two) $d_{\text{new}} \in \mathcal{D} \setminus \mathcal{D}_i^*(t_f^+)$ is selected, which ensures the condition $SLI_{R,i}(x_{\text{rec}}) \geq SLA_{R,i}$ is met. Based on the Markov decision process described in Stage 2, the reinforcement learning agent generates an action vector $\mathcal{U}_i(t_f^+)$. This vector determines the horizontal and vertical scaling parameters in the new domain, $\Delta \mathcal{H}_i$ and $\Delta \mathcal{V}_i$, minimizing the penalty function $\mathcal{W}_i(t)$ for the resource operation cost.

3) *Migration and reintegration.* The migration management module updates the placement function $\mathcal{M}(t)$ by transferring the logical components of graph $\mathcal{N}_i$ to the physical resources of the selected domain d_{new} . After successful initialization, the set of active domains is expanded:

$$\mathcal{D}_i^*(t_{\text{rec}}) = \mathcal{D}_i^*(t_f^+) \cup \{d_{\text{new}}\}. \quad (15)$$

The new domain receives its maximum computing capacity $\Pi_{\text{new}}^{\text{max}}$, after which the load balancer recursively applies the algorithm (14) to include d_{new} in the request processing flow.

All processes of failure isolation, target configuration calculation, component migration, and Service-Resource Model topology restoration are performed automatically by the ITIMS during operation without information system downtime.

E. System Architecture

The ITIMS is considered as a single decision-making center that structurally and functionally performs the tasks of the customer's internal IT department. It possesses complete information about the network configuration, the current state of several available domains, and service provisioning parameters. According to the accepted axiomatic assumptions, the ITIMS and all its internal components are absolutely reliable nodes. The ITIMS architecture is based on a modular principle, which ensures the continuous execution of control, optimization, and reconfiguration functions, as shown in Fig. 1.

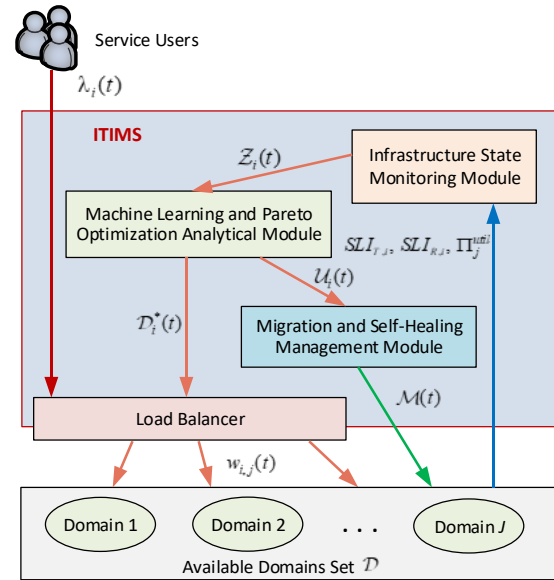


Fig. 1. Structural diagram of the IT infrastructure management system

The structural scheme of the ITIMS includes the following interconnected modules:

1) *Infrastructure State Monitoring Module.* Performs continuous data collection on the state of resources in all domains and conducts current quality evaluation of each individual service provisioning. In addition to measuring the actual metrics $SLI_{T,i}$ and $SLI_{R,i}$, the module determines the currently utilized capacity $\Pi_j^{\text{util}}(t)$ and the intensity of incoming requests $\lambda_i(t)$. Based on these data, the environment state vector $Z_i(t)$ is formed for further analysis.

2) *Machine Learning and Pareto Optimization Analytical Module.* Responsible for service quality level management. The module receives the state vector $Z_i(t)$ from the monitoring module and uses the trained agent (reinforcement learning) to form

the action vector $\mathcal{U}_i(t)$. The generated decisions regarding horizontal and vertical scaling are obligatorily compared with the Pareto set $\mathcal{P}_i$ of nondominated configurations to guarantee compliance with the $SLA_{R,i}$ normative values and minimize resource operation costs.

3) *Migration and Self-Healing Management Module*. Ensures the dynamic alteration of the placement function $\mathcal{M}(t)$. Receiving the coordinates of the target configuration from the analytical module, this module initiates the allocation of new resources or the removal of excess resources, as well as the transfer of the Service-Resource Model $\mathcal{G}_i$ functional components to backup domains instantaneously, during operation, without system downtime.

4) *Load Balancer*. Logically located separately from unreliable domains, but is an integral part of the ITIMS. Ensures the distribution of incoming user requests among available copies of critical category $\mathcal{S}_C$ service provisioning structures. The distribution is performed dynamically based on weight coefficients $w_{i,j}(t)$ relying on the residual computing capacity of the domains.

The ITIMS operates as follows. The infrastructure state monitoring module continuously generates state vectors $\mathcal{Z}_i(t)$ and transmits them to the analytical module. If the actual SLI metrics are recorded approaching the SLO or SLA boundary values, the analytical module initiates quality level management. It calculates the optimal scaling or self-healing policy, taking into account the Markov decision process and the Pareto set $\mathcal{P}_i$. The decision is transmitted to the migration management module, which interacts with the physical infrastructure to alter the configuration. Simultaneously, the load balancer recalculates the weight coefficients $w_{i,j}(t)$ and adapts the user request distribution scheme. All processes of control, optimization, and self-healing are performed automatically by the ITIMS during operation without system downtime.

VI. RESULTS

A. Simulation Conditions

To verify the adequacy of the mathematical models and the effectiveness of the developed method, simulation modeling of the information system and ITIMS behavior was conducted, as well as validation on a real Kubernetes cluster (hybrid testbed k3d and KWOK, where k3d is a tool that runs k3s – a lightweight Kubernetes distribution –

inside Docker containers). The experimental testbed includes five infrastructure domains $\mathcal{D} = \{d_1, d_2, d_3, d_4, d_5\}$, which differ in computing capacity, network latency, and intrinsic reliability (Table I).

TABLE I. DESCRIPTION OF THE EXPERIMENTAL TESTBED DOMAIN CHARACTERISTICS

Domain	Number of processors, CPU	Latency, ms	Reliability (R)
d_1	20	5	0.99
d_2	16	20	0.96
d_3	18	45	0.92
d_4	12	65	0.85
d_5	12	85	0.70

Also, worth mentioning that for the simplicity of experimental setup minimal Euclidian distance to ideal solution was used instead of Markov decision process described before.

The use of an expanded set of domains, rather than strictly two, allows investigating complex processes of multicriteria optimization, component migration, and spatial redundancy without loss of generality. The ITIMS within the experiment functions as the internal IT department, making decisions for the entire set $\mathcal{D}$.

The modeling was conducted for a set of 12 information services, among which 5 are critical ($\mathcal{S}_C = \{\text{core-db, command-ctrl, auth, payments, api-gateway}\}$). According to the service level agreement, the normative reliability indicator $SLA_{R,i}$ requires the simultaneous functioning of several independent structural copies of the Service-Resource Model of each critical service; the necessary number of replicas is determined by the reliability threshold $r_{\min}$ and is ensured by parallel redundancy

$$R_i = 1 - \prod_{j=1}^K (1 - R_k), \text{ where } K \text{ is number of redundant}$$

replicas. Additionally, a cascade dependency is taken into account: the effective service reliability is

$$R_{\text{eff}} = R_{\text{own}} \cdot \prod_{i=1}^D R_i, \text{ where } D \text{ is number of dependant}$$

services, meaning the failure of a shared dependency (e.g., core-db) affects all services using it. The normative quality indicator is limited from below by the threshold $Q_{\min} = 0.55$.

The experiment scenario combines two processes: the nonlinear growth of the load and the subsequent hardware failure of the domains. The

load profile grows approximately 2.2 times relative to the initial level with a subsequent decline (multiplier $1.0 \rightarrow 2.2 \rightarrow 1.3 \rightarrow 1.0$). At time $t = 60$ s, domain d_3 is destroyed (loss of the "frontend" domain), and at time $t = 120$ s, domain d_1 is destroyed (loss of the main data center). The control step is $\Delta t = 20$ s.

B. Adaptation to Load Growth

The initial system state was characterized by a stable stream of requests $\lambda_i(t_0)$, which was processed by a subset of domains with sufficient reliability. The monitoring module performed continuous quality evaluation. During the growth of $\lambda_i(t)$, the quality evaluation recorded the approach of the actual response time $SLI_{T,i}(t)$ to the target $SLO_{T,i}$.

The analytical module initiated the quality level management process. Using Pareto multicriteria optimization (the NSGA-II algorithm [18]) combined with adaptive reconfiguration rules (reactive self-healing upon service failure or unavailability, proactive anticipation based on quality deficit forecast, and overload thresholds initiating a repeated solution of the Pareto problem), the ITIMS formed the action vector $u_i(t)$. In the joint (hybrid) scaling mode, the optimizer simultaneously selected horizontal scaling (number of replicas) and vertical scaling (CPU load factor $f_s \in [1, 3]$): at peak moments, the system maintained up to 19 replicas with an average factor f_s of up to 1.48. The selected configuration x^* belonged to the Pareto set $\mathcal{P}_i$ and ensured the fulfillment of the condition:

$$\Delta T_i(x^*) = SLO_{T,i} - SLI_{T,i}(x^*) \rightarrow 0. \quad (16)$$

All operations for allocating additional resources and recalculating the load balancer weight coefficients $w_{i,j}(t)$ occurred without system downtime.

The dynamics of the autoscaler operation (horizontal and vertical levers relative to the total demand) are shown in Fig. 2.

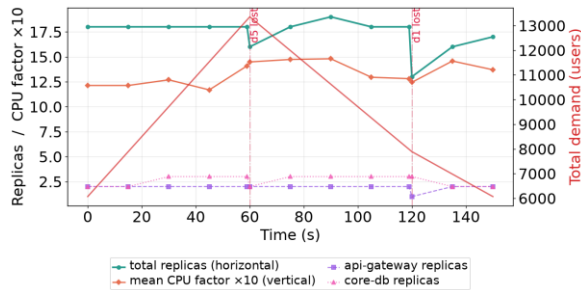


Fig. 2. Dynamics of the autoscaler operation based on Pareto-optimal solutions

A comparison of joint hybrid scaling with the uncoordinated baseline Kubernetes approach (HPA + VPA) is shown in Fig. 3.

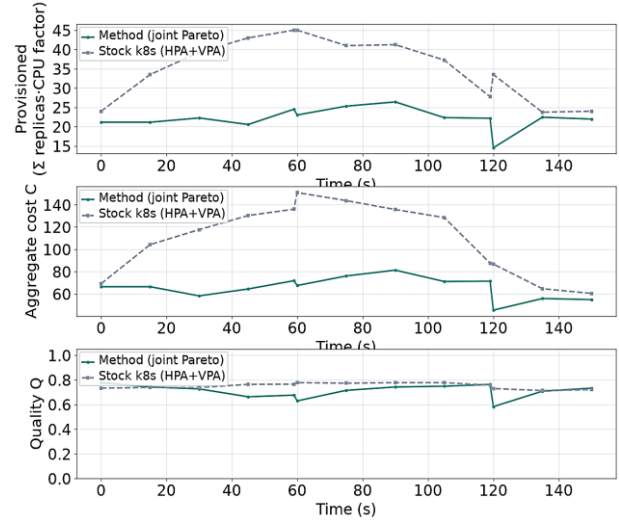


Fig. 3. Comparison of hybrid scaling with the uncoordinated Kubernetes approach

With comparable quality Q , the hybrid solution maintained an average cost $C = 65.4$ versus $C = 108.9$ for HPA + VPA, which is approximately 40% resource savings.

C. Self-Healing After Critical Failure

At time t_f , hardware domain failures were generated (destruction of d_3 at time $t = 60$ s and d_1 at time $t = 120$ s). The monitoring module recorded an instantaneous drop $SLI_{R,i}(t_f) < SLA_{R,i}$. The load balancer immediately isolated the affected domain by zeroing out its routing probability $w_{i,f}(t_f^+) = 0$, and redistributed the requests among the operational domains.

To restore the reliability metrics, the analytical module performed quality level management, selecting backup domains from the operational ones in the set $\mathcal{P}_i$, and the migration management module deployed the functional components $\mathcal{N}_i$ on the new resources. Thanks to dependency-oriented redundancy, cascade failures did not occur. The modeling confirmed that the ITIMS is capable of minimizing deviations:

$$\Delta R_i(x_{\text{rec}}) = SLI_{R,i}(x_{\text{rec}}) - SLA_{R,i} \rightarrow 0. \quad (17)$$

Comparative results regarding survivability and quality are shown in Fig. 4.

The self-healing method maintained the availability of critical services at the level of 1.0

throughout the entire experiment with an average quality $Q=0.708$ (min 0.582, which is above the threshold $Q_{\min}$); conversely, the baseline configurations without redundancy and without considering reliability demonstrated a drop in availability to 0.0 and 0.6, respectively.

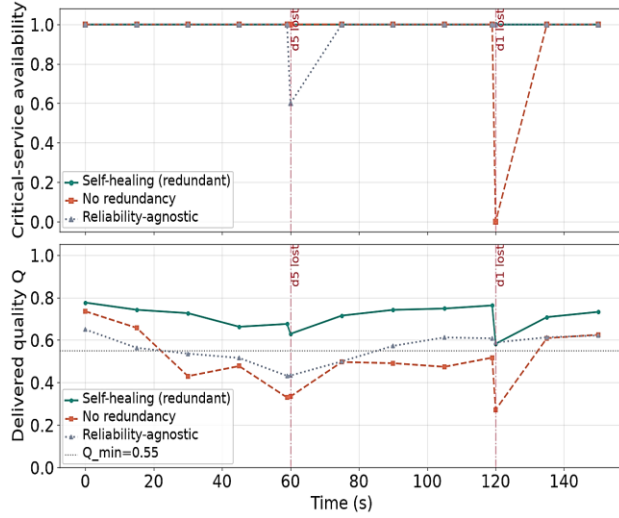


Fig. 4. Infrastructure survivability when using different quality level management methods

The multidimensional structure of the Pareto set tradeoffs (cost / quality / reliability / synchronization period τ / number of replicas) is illustrated in Fig. 5.

Validation on the real cluster (hybrid testbed: d_1 – real k3d, $d_2 - d_5$ – KWOK) is shown in Figs 6 and 7.

The validation confirmed the simulation modeling results: the proposed method ensured the availability of critical services at the level of 1.0 (minimum and average) and quality at the level $Q=0.797$ (min 0.773), recovering within one control step after each failure (zero RTO); the baseline Kubernetes approach showed a minimum availability of 0.6 and quality $Q=0.61$ (min 0.504).

The experimental research results of the proposed method and algorithms prove that the developed method guarantees the maintenance of the specified level of availability and response time. The difference between actual and target values approaches zero without excessive capacity redundancy. Pareto multicriteria optimization combined with adaptive reconfiguration rules allows the ITIMS to successfully perform infrastructure reconfiguration and recovery instantaneously, during operation, without system downtime.

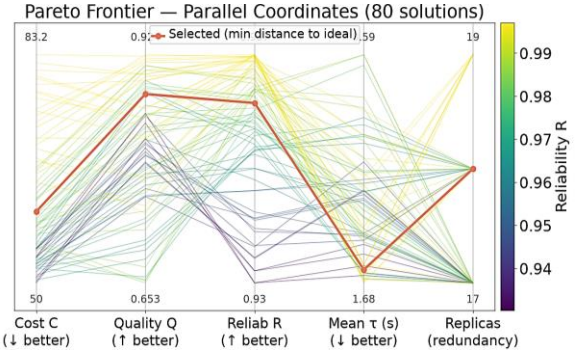


Fig. 5. Multidimensional tradeoff structure of the Pareto set

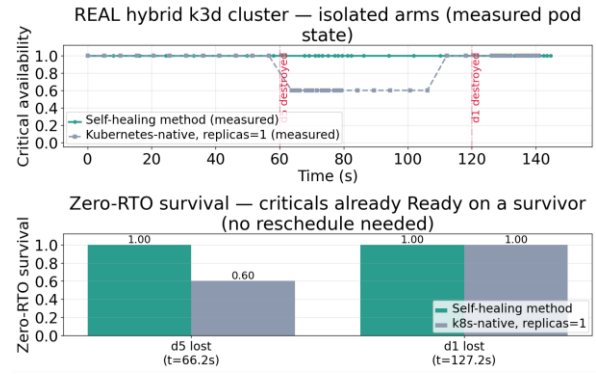


Fig. 6. Comparison of the proposed method based on Pareto-optimal solution search and the baseline algorithm without considering reliability needs

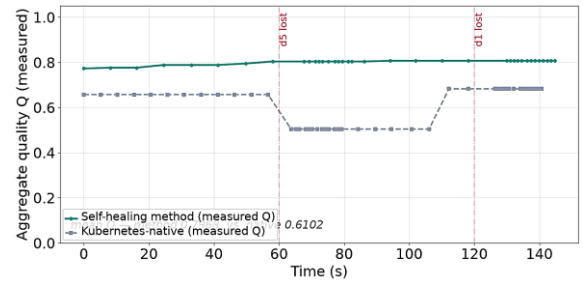


Fig. 7. Comparison of aggregated service quality values for the proposed method and the baseline Kubernetes deployment environment container distribution method

VII. CONCLUSIONS

In this article, the problem of ensuring the reliability and quality of information service provisioning in distributed computing environments has been solved by creating a self-healing critical IT infrastructure.

A resource management method based on the formalization of Service-Resource Models and service categorization is proposed. For services with high reliability requirements, a mathematical framework for synthesizing independent duplicate

structures across multiple domains has been developed. The application of the minimum normalized Euclidean distance to the ideal point within the solution set allowed substantiating the autoscaling algorithms.

Solving the multicriteria optimization problem through the search for the Pareto set of nondominated solutions ensured the rational distribution of computing capacities. The expediency of abandoning infinite response time minimization and infinite reliability maximization in favor of reducing the difference between actual metrics and target or normative values to zero has been proven.

The architecture of the IT infrastructure management system, which functions as the customer's internal IT department, has been developed. The separation of the quality evaluation processes, for which the monitoring module is responsible, and the quality level management, performed by the analytical module, has increased the determinism of decision-making. The weighted request distribution algorithm by the load balancer based on the residual capacity of domains effectively prevents service degradation.

Experimental modeling under conditions of an increasing number of users and infrastructure domain failures confirmed the viability of the proposed mathematical models. The system retains the capability for continuous self-healing, performing component migration and topology reconfiguration instantaneously, during operation, without system downtime.

REFERENCES

- [1] R. Buyya, J. Broberg, and A. Goscinski, *Cloud Computing: Principles and Paradigms*. Hoboken, NJ: John Wiley & Sons, 2011, <https://doi.org/10.1002/9780470940105>.
- [2] K. Rama and M. Reddy, "Self-Healing Systems: Reinforcement Learning for Cloud Resilience," *Int. J. Environ. Sci.*, vol. 11, no. 24S, pp. 5973–5981, 2025, <https://doi.org/10.64252/xqe8dg59>.
- [3] O. Johnphill, A. S. Sadiq, F. Al-Obeidat, H. Al-Khateeb, M. A. Taheir, O. Kaiwartya, and M. Ali, "Self-Healing in Cyber-Physical Systems Using Machine Learning: A Critical Analysis of Theories and Tools," *Future Internet*, vol. 15, no. 7, p. 244, 2023, <https://doi.org/10.3390/fi15070244>.
- [4] S. R. Challa, "Autonomous Cloud Engineering: The Rise of Self-Healing AWS Infrastructure Using AI and Event-Driven Automation," *World J. Adv. Eng. Technol. Sci.*, vol. 15, no. 2, pp. 2576–2587, 2025, <https://doi.org/10.30574/wjaets.2025.15.2.0810>.
- [5] I. Foster and C. Kesselman, Eds., *The Grid 2: Blueprint for a New Computing Infrastructure*. Elsevier, 2003, ISBN: 978-1-55860-933-4.
- [6] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, 2004, pp. 11–33. <https://doi.org/10.1109/TDSC.2004.2>.
- [7] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley & Sons, 2016, ISBN: 978-1-11928-542-7.
- [8] A. Beloglazov and R. Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers," *Concurrency Comput. Pract. Exp.*, vol. 24, no. 13, pp. 1397–1420, 2012, doi: 10.1002/cpe.1867.
- [9] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*. Boston, MA: Addison-Wesley, 2012, ISBN: 978-0-273-76059-7.
- [10] A. Ethan and D. K. Chikwari, "Deep Reinforcement Learning for Self-Healing IoT Networks," *Int. J. Mach. Learn. Res. Cybersec. Artif. Intell.*, vol. 14, no. 1, pp. 1275–1290, 2023.
- [11] O. Rolik and K. Znova, "A Multicriteria Method for Optimizing IT Service Management of a Virtual Provider Based on Deep Reinforcement Learning," *Electron. Control Syst.*, vol. 88, no. 2, p. 93, 2026, <https://doi.org/10.18372/1990-5548.88.20971>.
- [12] K. Senjab, S. Abbas, N. Ahmed, and A. U. R. Khan, "A Survey of Kubernetes Scheduling Algorithms," *J. Cloud Comput.*, vol. 12, no. 1, p. 87, 2023, <https://doi.org/10.1186/s13677-023-00471-1>.
- [13] A. Birolini, *Reliability Engineering*. Berlin, Heidelberg: Springer, 2010, <https://doi.org/10.1007/978-3-642-14952-8>.
- [14] M. Rausand and A. Hoyland, *System Reliability Theory: Models, Statistical Methods, and Applications*. John Wiley & Sons, 2020, ISBN: 978-1-119-37352-0.
- [15] M. Kleppmann, *Designing Data-Intensive Applications*. Sebastopol, CA: O'Reilly Media, 2017, ISBN: 978-1-449-37332-0.
- [16] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, 2016, ISBN: 978-1-491-92912-4.
- [17] V. Omelchenko and O. Rolik, "Automation of Resource Management in Information Systems Based on Reactive Vertical Scaling," *Adapt. Syst. Autom. Control*, vol. 2, no. 41, pp. 65–78, 2022, <https://doi.org/10.20535/1560-8956.41.2022.271344>.

[18] H. Ma, Y. Zhang, S. Sun, T. Liu, and Y. Shan, "A Comprehensive Survey on NSGA-II for Multi-Objective Optimization and Applications," *Artif. Intell. Rev.*, vol. 56, no. 12, pp. 15217–15270, 2023, <https://doi.org/10.1007/s10462-023-10526-z>.

Received: July 19, 2026
Accepted: August 12, 2026
Published: September 18, 2026

Rolik Oleksandr. ORCID 0000-0001-8829-4645. Doctor of Engineering Science. Professor. Head of the Department of Information Systems and Technologies, Faculty of Informatics and Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine. Education: Kyiv Polytechnic Institute, Kyiv, Ukraine, (1981). Research area: Information Technologies, IT Infrastructure Management Systems, IT Service Quality Management, Intelligent Embedded Systems and the Internet of Things. Publications: over 250 publications. E-mail: o.rolik@kpi.ua

Kolesnik Valerii. ORCID 0000-0001-9179-1335. Postgraduate Student. Department of Information Systems and Technologies, Faculty of Informatics and Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine. Education: Kyiv Polytechnic Institute, Kyiv, Ukraine, (2010). Research interests: Information systems and technologies, Docker containers, UAV vehicles. Publications: 17. E-mail: vkolesnik.kpi@gmail.com

О. І. Ролік, В. М. Колеснік. Метод управління ресурсами самовідновлювальної критичної ІТ-інфраструктури в багатодоменному середовищі

У статті розв'язано науково-практичне завдання забезпечення надійності та якості надання інформаційних послуг у розподілених обчислювальних середовищах шляхом розроблення методу управління ресурсами для самовідновлювальної критичної ІТ-інфраструктури. Досліджено процеси функціонування системи управління ІТ-інфраструктурою, яка структурно та логічно діє як внутрішній ІТ-департамент замовника. Запропоновано підхід до категоризації інформаційних сервісів та формалізації динамічних сервісно-ресурсних моделей, що дозволяє синтезувати незалежні дублюючі структури для послуг критичної категорії у декількох доменах. Удосконалено алгоритми автоматичного масштабування та зваженого розподілу навантаження із застосуванням визначення множини Парето-оптимальних рішень та використанні мінімальної нормалізованої Евклідової відстані до ідеальної точки в множині рішень. Обґрунтовано метод багатокритеріальної оптимізації Парето, який базується на математичному зведенні до нуля відхилення фактичних показників надійності та часу відгуку від їхніх цільових і нормативних значень. Сформовано архітектурну схему системи управління, де строго розмежовано процеси оцінювання якості послуг інструментами моніторингу та управління рівнем якості аналітичним модулем. Експериментальне моделювання підтвердило здатність розробленої системи ізолювати відмови ресурсів та адаптуватися до нелінійного зростання кількості запитів, виконуючи міграцію компонентів та реконфігурацію топології без зупинки системи.

Ключові слова: Парето-оптимальні рішення, сервісно-ресурсна модель, багатокритеріальна оптимізація, управління ІТ-послугами, інформаційні системи.

Ролік Олександр Іванович. ORCID 0000-0001-8829-4645. Доктор технічних наук. Професор. Завідувач кафедри інформаційних систем та технологій, Факультет інформатики та обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна. Освіта: Київський політехнічний інститут, Київ, Україна, (1981). Напрямок наукової діяльності: інформаційні технології, системи управління ІТ-інфраструктурою, управління якістю ІТ-послуг, інтелектуальні вбудовані системи та Інтернет речей. Кількість публікацій: понад 250 публікацій. E-mail: o.rolik@kpi.ua

Колеснік Валерій Миколайович. ORCID 0000-0001-9179-1335. Аспірант. Кафедра інформаційних систем та технологій. Факультет інформатики та обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна. Освіта: Київський політехнічний інститут, Київ, Україна, (2010). Напрямок наукової діяльності: інформаційні системи та технології, докер контейнери, БПЛА. Кількість публікацій: 17. E-mail: vkolesnik.kpi@gmail.com