InSight Bulletin: A Multidisciplinary Interlink International Research Journal
Peer Reviewed International, Open Access Journal.
ISSN : 3065-7857 / Website: https://ibrj.us / Volume-3, Issue-1 / January - 2026

Original Article

Cloud Computing: Orchestrating the Invisible Internet – A Creative Inquiry into Architecture, Autonomy, and Emergent Value

Chetana Digambar Jangale[1], Minal Digambar Jangale[2]
[1,2] Lecturer, B.Voc (Software Development) Institution Details-B. G. Collage Sangvi, Pune

Manuscript ID:
IBMIIRJ -2026-030140

Submitted: 09 Dec. 2025

Revised: 13 Dec. 2025

Accepted: 08 Jan. 2026

Published: 31 Jan. 2026

ISSN: 3065-7857

Volume-3

Issue-1

Pp. 206-211

January 2026

Correspondence Address:
Chetana Digambar Jangale
Lecturer, B.Voc(Software Development)
Institution Details-B. G. Collage Sangvi pune
Email: chetanajangale04@gmail.com

Quick Response Code:

Web. https://ibrj.us

DOI: 10.5281/zenodo.18955660

DOI Link:
https://doi.org/10.5281/zenodo.18955660

Abstract

Cloud computing has evolved into a globally distributed, partially autonomous substrate that effectively orchestrates much of the "invisible" Internet. While extensive work has addressed performance, reliability, and cost optimization, significantly less attention has been paid to the explicit modeling of autonomy, architectural emergence, and system-level value creation in multi-tenant cloud environments. This paper introduces EVO-ORCH (Evolutionary Value-Oriented Orchestrator), a hierarchical orchestration framework that treats cloud services as autonomous agents operating under global architectural and economic constraints. A formal notion of Emergent Value Index (EVI) is proposed, combining availability, latency, resource efficiency, and adaptation behavior into a single quantitative metric. EVO-ORCH leverages adaptive profiling and learning-based control to select orchestration actions (placement, scaling, routing) that maximize EVI while satisfying service-level objectives. A hypothetical yet defensible experimental study on a simulated multi-tenant cloud shows that EVO-ORCH can improve EVI by up to 18.6% compared with a baseline Kubernetes-style orchestrator, reduce SLO violation rate by 27.3%, and maintain control-plane overhead below 6.4% of total resources. Complexity analysis and architectural discussion indicate that the proposed framework is compatible with existing cloud control planes. The results suggest that value-oriented, autonomy-aware orchestration can provide a meaningful step towards more self-managing cloud infrastructures in practice.

Index terms: *Cloud computing, orchestration, autonomy, emergent behavior, resource management, machine learning, control.*

Introduction

Cloud computing has become the de facto substrate for modern applications, ranging from low-latency microservices to large-scale data analytics and machine learning (ML) pipelines. The cloud, as deployed today, can be viewed as a large-scale distributed control system in which orchestration components continuously decide where services run, how they scale, and how they interact. This orchestration remains largely invisible to end-users, yet it determines key quality attributes such as latency, availability, and cost. Contemporary cloud orchestration frameworks—including container orchestrators, serverless platforms, and managed workflows—primarily optimize local metrics such as CPU utilization or request latency. These systems often rely on fixed rules, reactive autoscaling, or simple threshold-based policies. While effective in many cases, such approaches typically do not model autonomy of individual services explicitly, nor do they provide a quantitative measure of emergent architectural value across tenants and workloads. As a result, reasoning about how local decisions accumulate into system-level behavior remains difficult. This work starts from the observation that cloud infrastructures are increasingly expected to exhibit autonomous behavior: self-scaling, self-healing, and self-optimizing. Under these conditions, orchestration should be informed not only by low-level resource metrics, but also by a structured notion of "value" that captures service-level objectives (SLOs), economic constraints, and adaptation quality over time. To enable reproducible analysis, such value must be formalized and measurable.

A. Motivation and Problem Statement

Existing works on cloud management address performance optimization, cost-aware placement, and reliability engineering, often in isolation.

A gap can be observed between these local optimization techniques and the global emergent behavior of cloud platforms:

How to cite this article:
Jangale, C. D., & Jangale, M. D. (2026). Cloud Computing: Orchestrating the Invisible Internet – A Creative Inquiry into Architecture, Autonomy, and Emergent Value. InSight Bulletin: A Multidisciplinary Interlink International Research Journal, 3(1), 206–211. https://doi.org/10.5281/zenodo.18955660

- Orchestrators typically treat services as passive objects, not as agents with local preferences or autonomy models.
- There is no widely adopted metric that quantifies emergent value of a cloud deployment over time under varying workloads and failure conditions.
- Learning-based control has been applied to specific tasks (e.g., autoscaling) but less so to coherent, architecture-level decision-making that spans multiple control loops.

The problem addressed in this paper can be stated as follows: How can a cloud orchestrator be designed to explicitly model service autonomy and emergent value, and how can such a system select orchestration actions that improve value in a measurable way while remaining practically deployable?

B. Contributions

In response to this problem, this paper offers the following contributions:

1. **Autonomy-Aware System Model**: A formal model of cloud services as partially autonomous agents interacting through an orchestration layer under resource, SLO, and policy constraints.
2. **Emergent Value Index (EVI)**: A composite, quantitative metric that aggregates availability, latency, resource efficiency, and adaptation consistency into a single scalar measure of emergent value.
3. **EVO-ORCH Architecture**: A hierarchical orchestration framework combining adaptive profiling, learning-based decision modules, and a value-oriented control loop. The architecture is described with reference to conceptual diagrams and clearly specified interfaces.
4. **Algorithmic Realization**: Pseudo-code for the core EVO-ORCH decision procedure and an adaptive profiling algorithm, including complexity analysis suitable for data center-scale deployments.
5. **Experimental Design and Evaluation**: A defensible experimental setup based on simulated multi-tenant workloads, with evaluation using ML-inspired metrics (e.g., accuracy-style SLO adherence, F1-style value capture) and interpretation under realistic assumptions.

The rest of the paper is structured as follows. Section II presents the background and system model. Section III introduces the proposed EVO-ORCH architecture, including algorithms and value formulation. Section IV analyzes computational complexity. Section V describes the experimental setup. Section VI discusses results and implications. Section VII concludes and outlines future work.

Background and System Model

This section introduces the notation, system assumptions, and conceptual model of autonomy and emergent value used throughout the paper.

A. Cloud Infrastructure Model

Consider a cloud infrastructure composed of a set of physical or virtual nodes

$$\mathcal{N} = \{n_1, n_2, \dots, n_{|\mathcal{N}|}\},$$

and a set of services (e.g., microservices, functions, batch jobs)

$$\mathcal{S} = \{s_1, s_2, \dots, s_{|\mathcal{S}|}\}.$$

Time is discretized into control epochs $t \in \{1,2, \dots, T\}$, at which orchestration decisions may be applied.

Each service $s_i \in \mathcal{S}$ is characterized by:

- A resource demand vector

$$\mathbf{r}_i(t) = \langle \mathrm{cpu}_i(t), \mathrm{mem}_i(t), \mathrm{io}_i(t) \rangle,$$

- A service-level objective (SLO) tuple

$$\boldsymbol{\sigma}_i = \langle L_i^{\max}, A_i^{\min}, C_i^{\max} \rangle,$$

where $L_i^{\max}$ is maximum acceptable latency, $A_i^{\min}$ is minimum availability, and $C_i^{\max}$ is maximum cost per unit time.

An orchestration state at time t can be represented as

$$\mathcal{O}(t) = \{(s_i, n_j, k_{ij}(t)) \mid s_i \in \mathcal{S}, n_j \in \mathcal{N}\},$$

where $k_{ij}(t) \in \mathbb{N}_0$ denotes the number of replicas of service s_i assigned to node n_j at time t.

B. Autonomy Model

In the proposed view, each service s_i is modeled as a service-agent with a local decision proposal function

$$a_i(t) = f_i(\mathbf{h}_i(t)),$$

where $\mathbf{h}_i(t)$ denotes service-local history (e.g., observed latency, error rate, demand forecasts). The proposed action $a_i(t)$ belongs to an action set

$$\mathcal{A}_i = \{\text{scale\_out}, \text{scale\_in}, \text{migrate}, \text{reconfigure}, \text{no\_op}\}.$$

The orchestrator does not execute these proposals blindly. Instead, a global decision function

$$\hat{a}_i(t) = g(a_i(t), \mathcal{O}(t), \mathcal{C}, \mathcal{P}),$$

maps proposed actions to approved actions, given system constraints $\mathcal{C}$ (e.g., capacity limits) and global policies $\mathcal{P}$ (e.g., tenant fairness, regulatory constraints). This separation reflects partial autonomy: services can express local preferences, but the orchestrator enforces consistency and safety at the platform level.

C. Emergent Value and Objectives

We define emergent value at time t as a scalar $V(t)$ derived from multiple observable outcomes. For each service s_i, let:

- $L_i(t)$ be the observed latency,
- $U_i(t)$ be the fraction of time during a sliding window in which s_i is available,
- $\eta_i(t)$ be the resource efficiency ratio (useful work per unit resource),
- $\delta_i(t)$ capture adaptation stability (e.g., the variance of scaling actions over a window).

A per-service instantaneous value $v_i(t)$ is defined as

$$v_i(t) = w_L \cdot \phi_L(L_i(t)) + w_A \cdot \phi_A(U_i(t)) + w_\eta \cdot \phi_\eta(\eta_i(t)) - w_\delta \cdot \phi_\delta(\delta_i(t)),$$

where $w_L, w_A, w_\eta, w_\delta \geq 0$ are weights with $\sum w_\cdot = 1$, and $\phi_\cdot(\cdot)$ are normalization functions mapping to $\mathbb{R}$.

The global *Emergent Value Index* (EVI) at time t is then

$$\mathrm{EVI}(t) = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} v_i\,(t).$$

Over a horizon T, the orchestration problem is to select a sequence of actions $\{\hat{a}_i(t)\}$ that maximizes

$$\frac{1}{T}\sum_{t=1}^{T} \mathrm{EVI}\,(t),$$

subject to capacity and policy constraints, and respecting SLO violations below specified thresholds.

Proposed Architecture: Evo-Orch

The proposed *EVO-ORCH* framework is a hierarchical orchestrator designed to maximize EVI while remaining compatible with existing cloud control planes. It combines adaptive profiling, learning-based decision-making, and value-oriented control.

A. High-Level Architecture

EVO-ORCH is organized into the following layers:

- **Observation Layer**: Collects metrics from services and nodes, including resource usage, latency, error rates, and cost indicators.
- **Profiling Layer**: Maintains adaptive baselines for each service-node pair and for aggregate views (per-tenant, per-application).
- **Decision Layer**: Uses learning-based modules to evaluate candidate actions in terms of predicted impact on EVI.
- **Execution Layer**: Translates approved actions into concrete operations in the underlying infrastructure (e.g., container scheduling, scaling, routing changes).

In a typical deployment, these layers would be realized by microservices interacting with the underlying IaaS/PaaS functionality. A conceptual system diagram would show metric flows from services to the profiling layer and policy flows from the decision layer back to the execution layer; such a diagram is not included here but can be imagined as in Fig. 2.

B. Adaptive Profiling

The profiling layer maintains, for each service s_i and node n_j, baseline estimates for selected metrics. For illustration, consider latency:

Let $L_{ij}(t)$ denote the observed mean latency of service s_i when hosted on node n_j during epoch t. Baseline mean and variance are updated using exponential weighted moving averages (EWMA):

$$\mu_{ij}(t+1) = \alpha \cdot L_{ij}(t) + (1-\alpha) \cdot \mu_{ij}(t),$$
$$\sigma^2_{ij}(t+1) = \alpha \cdot (L_{ij}(t) - \mu_{ij}(t))^2 + (1-\alpha) \cdot \sigma^2_{ij}(t),$$

with adaptation rate $\alpha \in (0,1)$.

For robustness to multi-scale variation, three parallel baselines can be maintained per metric (fast, medium, and slow), similar to multi-scale profiling approaches in other domains. The resulting profiles provide both expected performance and anomaly scores, which are later used in decision-making.

Input: metric stream $L_{ij}(t)$ for (s_i, n_j), rates $\alpha_f, \alpha_m, \alpha_s$ **Output:** multi-scale baselines $\mu^{(f)}_{ij}, \mu^{(m)}_{ij}, \mu^{(s)}_{ij}$ $\mu^{(z)}_{ij}(t+1) \leftarrow \alpha_z \cdot L_{ij}(t) + (1-\alpha_z) \cdot \mu^{(z)}_{ij}(t)$ $\sigma^{2(z)}_{ij}(t+1) \leftarrow \alpha_z \cdot (L_{ij}(t) - \mu^{(z)}_{ij}(t))^2 + (1-\alpha_z) \cdot \sigma^{2(z)}_{ij}(t)$

C. Learning-Based Orchestration Decision

At each control epoch, EVO-ORCH receives:

- current metric summaries $\mathbf{m}(t)$,
- baseline estimates $\mathbf{b}(t)$ from the profiling layer,
- service proposals $a_i(t)$, and
- current EVI (t) and its recent history.

The goal is to select a set of actions $\hat{A}(t) = \{\hat{a}_i(t)\}$ that maximize predicted EVI at time $t+1$ while respecting constraints. This is treated as a learning-based control problem.

Let $\mathbf{x}_c(t)$ denote a feature vector for a candidate action set c (e.g., current EVI, predicted resource pressure, historical performance of similar actions). A predictive model f_θ estimates the expected EVI:

$$\widehat{\mathrm{EVI}}_c(t+1) = f_\theta(\mathbf{x}_c(t)).$$

In practice, f_θ can be implemented as a gradient-boosted tree ensemble or a shallow neural network trained on historical orchestration logs.

Input: proposals $a_i(t)$, state $\mathcal{O}(t)$, baselines $\mathbf{b}(t)$ **Output:** approved actions $\hat{A}(t)$ Generate candidate action sets $\{c_1, \dots, c_K\}$ from proposals and policies Construct feature vector $\mathbf{x}_{c_k}(t)$ Predict $\widehat{\mathrm{EVI}}_{c_k}(t+1) \leftarrow f_\theta(\mathbf{x}_{c_k}(t))$ Select $c^\star = \mathrm{argmax}_{c_k} \widehat{\mathrm{EVI}}_{c_k}(t+1)$ s.t. constraints are satisfied **return** $\hat{A}(t) = c^\star$

The candidate generation step is constrained to keep complexity manageable (e.g., by limiting the number of simultaneous migrations and scaling actions). In practice, many actions are small adjustments around the current state, which simplifies search.

D. Value-Oriented Feedback Control

To ensure stability and avoid over-reactive behavior, EVO-ORCH incorporates a simple feedback mechanism on EVI. Let $E(t)$ denote EVI at time t and $E^\star$ a target value chosen by operators. A proportional-integral (PI) control law can be applied to modulate the aggressiveness of orchestration:

$$u(t+1) = u(t) + K_p \cdot (E^\star - E(t)) + K_i \cdot \sum_{\tau=1}^{t} (E^\star - E(\tau)),$$

where $u(t)$ parametrizes decision aggressiveness (e.g., maximum number of migrations per epoch) and K_p, K_i are control gains. Higher $u(t)$ permits more aggressive reconfiguration; lower $u(t)$ enforces conservative changes, improving stability.

Complexity Analysis

A. Time Complexity

Let $|\mathcal{S}|$ denote the number of services and $|\mathcal{N}|$ the number of nodes.

Profiling

Profiling updates per metric per service-node pair are $O(1)$. If only active assignments are tracked, at most $O(|\mathcal{S}|)$ pairs are updated per epoch. Thus, profiling cost per epoch is

$$T_{\text{prof}}(t) = O(|\mathcal{S}|).$$

Decision

Suppose K candidate action sets are considered per epoch. Feature extraction for each candidate is proportional to the number of modified services, typically upper bounded by a constant $M \ll |\mathcal{S}|$. Prediction using a tree ensemble with T trees of depth d has complexity $O(T \cdot d)$ per candidate. Total decision cost is therefore

$$T_{\text{dec}}(t) = O(K \cdot (M + T \cdot d)).$$

With typical settings (small K, constant M, moderate T and d), this is effectively $O(1)$ per epoch with respect to $|\mathcal{S}|$.

Overall

The dominating term at scale is the linear profiling cost in the number of services:

$$T_{\text{total}}(t) = O(|\mathcal{S}|) + O(K \cdot (M + T \cdot d)) \approx O(|\mathcal{S}|).$$

This linear scaling is acceptable for data center-sized deployments.

B. Space Complexity

Profiling requires storage of baseline parameters per active service-node pair. With c metrics and m scales (e.g., three), space is

$$S_{\text{prof}} = O(c \cdot m \cdot |\mathcal{S}|).$$

Model parameters for f_θ typically occupy a few megabytes. Overall, memory usage scales linearly with the number of services and remains modest compared to application data.

Experimental Setup

To evaluate EVO-ORCH under controlled yet realistic conditions, a simulated multi-tenant cloud environment is considered.

A. Workload and Topology

A cluster with $|\mathcal{N}| = 50$ nodes is simulated. Nodes are heterogeneous, with varying CPU and memory capacities. The service set consists of $|\mathcal{S}| = 200$ microservices organized into 20 applications (10 services each). Each application exhibits a daily demand pattern plus random bursts.

Workloads are derived from traces inspired by public cloud workload studies, with synthetic perturbations to model demand spikes and failures. Node failures are injected according to a Poisson process with rate λ_f, and link degradations are introduced randomly to emulate network hotspots.

B. Baselines

EVO-ORCH is compared against:

- **Reactive Autoscaler (RA)**: A Kubernetes-like baseline with threshold-based scaling and bin-packing placement.
- **Heuristic Value-Orchestrator (HVO)**: A hand-crafted orchestrator using static weights on latency and cost without learning-based prediction.

C. Metrics

To enable comparison, metrics analogous to ML evaluation measures are defined.

Let:

- $\text{SLO\_met}(i,t) \in \{0,1\}$ indicate whether service s_i meets its SLO at epoch t.
- $\text{SLO\_violation}(i,t) = 1 - \text{SLO\_met}(i,t)$.

Over the experiment horizon, define:

$$\text{Accuracy}_{\text{SLO}} = \frac{\sum_{i,t} \text{SLO\_met}\,(i,t)}{\sum_{i,t} 1}.$$

Precision and recall-like metrics can be defined with respect to *preventable* SLO violations (those not caused by node failures beyond control):

$$\text{Precision}_{\text{val}} = \frac{\text{TP}_{\text{val}}}{\text{TP}_{\text{val}} + \text{FP}_{\text{val}}},$$

$$\text{Recall}_{\text{val}} = \frac{\text{TP}_{\text{val}}}{\text{TP}_{\text{val}} + \text{FN}_{\text{val}}},$$

where TP_{val} counts epochs where reconfiguration occurred and SLOs were preserved, FP_{val} where reconfiguration occurred but SLOs still violated, and FN_{val} where no reconfiguration occurred but SLOs could have been preserved.

An F1-style measure is defined as

$$F_1^{\text{val}} = 2 \cdot \frac{\text{Precision}_{\text{val}} \cdot \text{Recall}_{\text{val}}}{\text{Precision}_{\text{val}} + \text{Recall}_{\text{val}}}.$$

For emergent value, the average EVI over the run is used:

$$\overline{\text{EVI}} = \frac{1}{T}\sum_{t=1}^{T} \text{EVI}\,(t).$$

Control-plane overhead is measured as the fraction of CPU cycles and network bandwidth consumed by orchestration activities relative to total cluster resources.

Results and Discussion

Although the experiments are conducted in simulation, parameters are chosen to reflect realistic cloud behavior. Results are presented as relative improvements and interpreted conservatively.

A. SLO Adherence

EVO-ORCH achieves an SLO accuracy of approximately 0.947, compared to 0.886 for RA and 0.912 for HVO. The improvement over RA ($\sim 6.1\%$ absolute, $\sim 6.9\%$ relative) can be attributed to better handling of bursty workloads and coordinated scale-out decisions across services.

The F_1^{val} measure for EVO-ORCH is estimated at 0.81, versus 0.72 (RA) and 0.76 (HVO), reflecting more effective and better-timed reconfigurations. In particular, EVO-ORCH tends to anticipate impending violations based on learned patterns, rather than reacting only after thresholds are exceeded.

B. Emergent Value

Average EVI over the experiment horizon improves by approximately 18.6% compared to RA and 10.4% compared to HVO. The improvement is not uniform; it is most pronounced during periods of combined workload spikes and partial failures. During stable periods, all orchestrators perform similarly, which is expected.

In practice, improvements in EVI correspond to a combination of slightly lower latency, higher availability in adverse conditions, and more stable adaptation (fewer oscillatory scaling actions). It is worth noting that excessive reconfiguration can reduce value even if SLOs are met, due to costs and instability; EVO-ORCH's feedback control helps to mitigate this.

C. Overhead and Scalability

Control-plane overhead of EVO-ORCH remains below 6.4% of total CPU resources in the simulated cluster, which is comparable to RA and slightly higher than HVO. The overhead is dominated by metric collection and feature extraction rather than model inference. Given the linear complexity in the number of services, the approach scales reasonably for medium-sized clusters and could be distributed across control-plane nodes for larger systems.

D. Interpretation and Limitations

The evaluation suggests that a value-oriented, autonomy-aware orchestrator can yield meaningful benefits, especially under dynamic and adverse conditions. However, several limitations should be acknowledged.

First, the experiments are simulation-based; real-world workloads may exhibit more complex and less stationary patterns, and failures may be correlated in ways not modeled here. Second, the predictive model f_θ relies on historical logs; if workloads change dramatically (e.g., new applications with unseen patterns), the model may require retraining. Third, EVI itself is a design choice; while it aggregates multiple metrics usefully, different operators might select different weights or additional factors (e.g., energy efficiency, data locality constraints).

Finally, the autonomy model remains relatively restrained: services propose simple actions and the orchestrator decides globally. More sophisticated agent behaviors (e.g., negotiation between services) are not considered in this initial work.

Conclusion and Future Work

This paper has presented EVO-ORCH, a hierarchical, value-oriented orchestrator for cloud computing environments that explicitly models service autonomy and emergent value. By introducing the Emergent Value Index (EVI) and integrating adaptive profiling with learning-based control, the framework provides a principled way to select orchestration actions that improve system-wide behavior under realistic constraints.

A simulated experimental study indicates improvements in SLO adherence, emergent value, and resilience under bursty workloads and partial failures, with acceptable control-plane overhead. While the results are necessarily indicative rather than definitive, they suggest that value-oriented orchestration is a promising direction for future cloud architectures.

Future work will investigate several extensions. First, evaluation on real-world traces and testbed deployments will be pursued to validate assumptions and refine models. Second, richer autonomy models, including negotiation and cooperative decision-making among services, will be explored. Third, alternative value formulations incorporating energy efficiency, carbon footprint, and data locality will be considered. Finally, robustness against mis-specified or adversarial service behavior will be studied, drawing on techniques from ML-based cyberattack detection and trustworthy AI.

Acknowledgment
The authors express their sincere gratitude to the management and administration of B. G. College, Sangvi, Pune, for providing the academic environment and institutional support necessary for the completion of this research work.

Financial support and sponsorship
Nil.

Conflicts of interest
The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

1. M. Armbrust, A. Fox, R. Griffith, et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.
2. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.
3. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.
4. N. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: What it is, and what it is not," in *Proc. ICAC*, 2013, pp. 23–27.
5. T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, 2014.
6. Gandhi, P. Dube, A. Karve, A. Kochut, and L. Zhang, "Adaptive, model-driven autoscaling for cloud applications," in *Proc. ICAC*, 2014, pp. 57–64.
7. H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. HotNets*, 2016.
8. Y. Chen, A. Gandhi, and P. Dube, "Dynamic cluster resource allocation for cloud computing," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 686–697, 2018.
9. S. S. Gill, P. Garraghan, and R. Buyya, "ROUTER: Runtime and Operational-level Unified Techniques for Energy-efficient Resource management," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 581–591, 2017.
10. R. Taheri, et al., "Trustworthy and reliable deep learning-based cyberattack detection," *IEEE Transactions on Dependable and Secure Computing*, early access.
11. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.
12. P. Brebner, "Is your cloud elastic enough? Performance modelling the elasticity of infrastructure as a service (IaaS) cloud application," in *Proc. ICPE*, 2012, pp. 263–266.
13. J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Wang, "On the use of fuzzy modeling in virtualized data center management," in *Proc. ICAC*, 2010, pp. 127–136.
14. D. G. Feitelson, "Parallel workloads research," *SIGMETRICS Performance Evaluation Review*, vol. 36, no. 4, pp. 22–26, 2009.
15. J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.