

Predictive Multi-Objective Energy--Latency Optimization in Cloud Environments Using Ensemble Learning

Ahmed A. M. Warad

ahmedwarad.2010@gmail.com

Cairo University

Snaa A. ElSawy

Sadat Academy for Management Sciences

Rana Mostafa

Sadat Academy for Management Sciences

Miral Adel

Sadat Academy for Management Sciences

Abdelrahman Farouk

Sadat Academy for Management Sciences

Kareem Kamal A. Ghany

Sadat Academy for Management Sciences

Research Article

Keywords: cloud computing, energy efficiency, latency, machine learning, multi-objective optimization, ensemble learning, sensitivity analysis, cross-validation, Bitbrains

Posted Date: October 16th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11381470/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Predictive Multi-Objective Energy–Latency Optimization in Cloud Environments Using Ensemble Learning

Ahmed A. M. Warad^{1*}, Snaa A. ElSawy², Rana Mostafa²,
Miral Adel², Abdelrahman Farouk², Kareem Kamal A. Ghany²

^{1*}Information Systems and Technology, Faculty of Graduate Studies for
Statistical Research, Cairo University, Cairo, Egypt.

²Software Engineering Department, Faculty of Computer Science and
Informatics, Sadat Academy for Management Sciences, Sadat City,
Egypt.

*Corresponding author(s). E-mail(s): ahmedwarad.2010@gmail.com;
Contributing authors: Snaa.Elsawy.FCI2023059@sadatacademy.edu.eg;
Rana.Ali.FCI2023216@sadatacademy.edu.eg;
Miral.Mosa.FCI12023060@sadatacademy.edu.eg;
Abdelrahman.Farouk.FCI2023007@sadatacademy.edu.eg;
Kareem.Kamal@sadatacademy.edu.eg;

Abstract

The trade-off between energy consumption and communication latency in cloud data centers is strongly coupled and nonlinear, making static or single-objective policies wasteful or unsafe. We present a predictive multi-objective framework that learns surrogate models of server energy and queueing latency from production traces and uses them to select energy-aware operating points. Targets are generated by physically grounded power and queueing maps, yielding a trace-driven emulation validated against those maps. On public Bitbrains traces of seven virtual machines at five-minute resolution, 60.3 percent of intervals lie at or beyond the 0.95 stability threshold. Within the stable envelope, linear regression collapses on latency (R-squared 0.565), while ensembles reach at least 0.986 and cut error by up to 90.5 percent with inter-fold deviations below 0.014. A gradient-boosting controller cuts energy by 72.4 percent versus peak speed and 58.9 percent versus reactive scaling at zero violations and 0.29 s mean latency, against 3.80 percent violations for energy-only minimisation, holding the 99th

percentile at 1.25 s. Savings stay at 64.8–73.3 percent under parameter perturbations and 72.8 percent leave-one-machine-out, showing generalisation beyond a single trace.

Keywords: cloud computing, energy efficiency, latency, machine learning, multi-objective optimization, ensemble learning, sensitivity analysis, cross-validation, Bitbrains

1 Introduction

Cloud data centers are increasingly constrained by two goals that point in opposite directions: operators must deliver low-latency service on a strict energy budget [1, 2]. Reducing energy typically demands lowering server speed or consolidating workloads, which extends service times and risks quality-of-service (QoS) violations; preserving headroom for latency wastes the bulk of idle capacity [3, 4]. The resulting design space has been organized into comprehensive surveys and taxonomies of energy-efficient techniques [5–10]. The underlying coupling is fundamentally nonlinear: power grows super-linearly with clock frequency in conventional dynamic voltage and frequency scaling (DVFS) models [11–13], while queueing response times diverge as utilization approaches capacity [14–16]. Any controller that ignores either nonlinearity—or treats the two objectives as independent—cannot operate safely near the efficiency frontier.

Because both objectives emanate from the same resource state, they cannot be optimized independently. A scheduler that greedily minimizes energy drives servers toward saturation and inflates the response-time tail, whereas a scheduler that only guards latency leaves servers idle near peak power. Prior work has largely optimized a single objective or relied on heuristic rules that do not adapt to the statistical structure of real workloads [17–19]. Meanwhile, machine learning offers a direct route: learn the energy and latency maps from recorded telemetry and optimize against the learned surrogates [20–22]. This paper pursues that route with a particular emphasis on *trustworthiness*: the learned surrogates are validated not only on a single held-out split but also with cross-validated variance estimates, the controller’s operating points are re-evaluated under one-at-a-time perturbations of every modeling assumption, and the results are shown to hold when entire workload traces are withheld.

This paper makes the following contributions:

- We design and validate a speed-scaling energy model grounded in DVFS power curves and an $M/M/1$ response-time model for latency, calibrated against the public Bitbrains traces [23], and we characterize the oversubscription regime of a seven-VM production segment.
- We train and compare five supervised regressors as *surrogates* of energy and latency as functions of resource utilization and operating speed, using consistent metrics (MAE, RMSE, R^2), and we quantify the reliability of these estimates with 5-fold cross-validation including inter-fold variance.
- We embed ensemble (gradient-boosting) surrogates in a weighted scalarized multi-objective controller with explicit stability and SLA constraints, and evaluate it

against peak-speed, reactive, energy-only, random-feasible-speed, and SLA-greedy oracle baselines on held-out intervals.

- We perform a one-at-a-time sensitivity analysis over the service-demand, telemetry-noise, idle-power, and peak-power parameters, demonstrating that savings and zero-violation behavior persist across plausible perturbations.
- We conduct a leave-one-VM-out generalization study holding out entire VM traces, confirming that the learned maps and the controller’s benefits transfer across workloads.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 defines the system model and problem formulation, including the formal properties that ground the design. Section 4 presents the predictive surrogates and their evaluation protocol, including cross-validation. Section 5 describes the optimization framework and the compared policies. Section 6 reports and discusses the experimental results: workload characterization, predictive accuracy and surrogate diagnostics, cross-validated reliability, feature importance, the optimization evaluation, the behavior of the controller across load bands, a sensitivity analysis, and a leave-one-VM-out generalization study. Section 8 concludes.

2 Related Work

2.1 Power and Energy Modeling

Energy-efficient resource management in clouds has been studied for over a decade. Beloglazov and Buyya established the now-standard utilization-based power model and dynamic consolidation of virtual machines (VMs) [15, 24]; consolidated schemes become operational through live migration [25], hotspot-driven black-box and gray-box migration [26], and migration-aware consolidation with explicit migration budgets [27, 28]. The physical basis of these schemes rests on energy-proportional-computing arguments [3, 4], on the observation that idle power dominates at low utilization, and on cubic DVFS power scaling [11, 13]; the latter is the same law our energy model exploits, and a central question in this literature is precisely how much energy speed scaling can actually reclaim [13]. Comprehensive power-model and energy-accounting surveys [1, 12] stress that models must be checked against the structural assumptions they embed—a concern our sensitivity analysis addresses directly. On the algorithmic side, Albers formalized energy-efficient speed scaling and established the intrinsic competition between energy and response time [14]; the queueing-theoretic side of the same tension has been standard since Kleinrock [16]. Broad taxonomies classify the resulting techniques [5–8], and recent surveys extend the agenda to sustainability at scale [9, 10].

2.2 Workload Prediction and Forecasting

A distinct line of work uses machine learning to *forecast* workload and resource demand ahead of provisioning decisions [17, 29–31]. The classical baselines are statistical: linear host-load models that date back to Dinda and O’Hallaron [32] and ARIMA-based demand prediction [31]. Modern ML-centric models for workload forecasting

and energy-state estimation report substantial accuracy gains over those baselines [20]. Our task is different in kind: we do not predict future telemetry, but learn the *static functional map* from the current resource state and a candidate speed to the two objectives (a system-identification problem). This makes our accuracy claims about function approximation fidelity, not about temporal prediction; it also means our surrogates can be validated up to the noise floor of the trace rather than against the intrinsic unpredictability of the future.

2.3 Learning-Based Control and Scheduling

Machine learning has also entered as the scheduling policy itself. Deep reinforcement learning (DRL) was pioneered for cluster resource management by Mao *et al.* [33], scaled to joint resource allocation and power management [34], and applied to energy–time workflow scheduling in edge-cloud systems [35]. Surveys and recent works broadly cover RL-based computation offloading and multi-objective task scheduling [22, 36, 37]. These methods are powerful when the environment is a black box, but they inherit deep-learning engineering costs, non-determinism, and a large simulation burden. Our workflow follows the complementary, explicitly model-grounded route: physically derived maps, learned surrogates, and a cheap, deterministic scalarized selection. The two are complementary rather than competing—a learning layer of the DRL kind could replace the fixed scalarization inside our framework, which we list as future work.

2.4 Multi-Objective Optimization for the Cloud

Studies that perform genuine multi-objective optimization frequently rely on heuristic or evolutionary solvers, such as NSGA-II-style population methods [38], over synthetic workloads and without data-driven surrogates learned from real telemetry [18, 19]. Such methods search a candidate space with fitness functions that are usually evaluated by simulation; when they are applied to real cloud data, the evaluation is often trace-driven but the underlying models are fixed heuristics. A recurring limitation across both the single-objective and multi-objective strands is evaluation protocol: accuracy and control results are typically reported from a single split of a single trace, without cross-validated variance, task-level sensitivity analysis, or out-of-trace generalization checks. It is therefore difficult for an operator to judge whether a reported saving is an artifact of the particular partition, the assumed model constants, or the particular workload segment.

2.5 Positioning

Our work differs in four respects. First, we model energy and latency from the *same* resource state using physically grounded maps (DVFS power scaling and queueing response time), avoiding the common pitfall of treating CPU utilization as a stand-in for both objectives simultaneously. Second, we learn the surrogates from real production traces (Bitbrains) and validate the error statistics with 5-fold cross-validation and residual diagnostics. Third, we evaluate the resulting controller not only on aggregate cost but on its SLA-violation behavior, its response-time tail, its sensitivity to every

modeling assumption, and its ability to generalize to withheld VM traces—the properties operators actually rely on. Fourth, and unlike oracle-assisted simulation studies, we keep the optimization strictly surrogate-driven at selection time, then evaluate the realized operating points under the true physical maps, so the reported frontier is not an artifact of the learners.

3 System Model and Problem Formulation

3.1 Workload and Operating State

We consider a fleet of N provisioned servers, each hosting a virtual machine with capacity C_i MHz. Time is slotted at the telemetry resolution Δt (here 300 s). For each server i at slot t , the workload trace provides CPU usage $w_i(t)$, memory occupancy $m_i(t)$, disk throughput $d_i(t)$, and network throughput $n_i(t)$. We define the load ratio

$$u_i(t) = \frac{w_i(t)}{C_i}, \quad (1)$$

which is the CPU demand normalized by the provisioned share. Values $u_i(t) > 1$ indicate oversubscription: demand exceeds the provisioned capacity. The full operational state of server i at slot t is

$$\mathbf{x}_i(t) = [u_i(t), \rho_i(t), d_i(t), n_i(t)], \quad (2)$$

where $\rho_i(t) = m_i(t)/M_i$ is the memory occupancy ratio and M_i is the allocated memory. The DVFS operating speed $s \in \mathcal{S}$ (fraction of nominal frequency) is the control variable; $\mathcal{S}$ is the discrete set of available speeds (here 0.30 to 1.00 in steps of 0.05).

3.2 Energy Model

We adopt the standard proportional-power model with a cubic DVFS component [14, 24]. The normalized power of a server operating at speed s under load u is

$$P(s, u) = s^3 \left(P_{\text{idle}} + (P_{\text{peak}} - P_{\text{idle}}) \min\left(1, \frac{u}{s}\right) \right), \quad (3)$$

with $P_{\text{idle}} = 72$ W and $P_{\text{peak}} = 120$ W used throughout—representative mid-range server values chosen so that the idle fraction ($P_{\text{idle}}/P_{\text{peak}} = 0.6$) matches typical x86 platforms; Section 6.8 verifies that the qualitative conclusions are insensitive to this absolute scale. The dynamic term is proportional to s^3 , reflecting the cubic energy–frequency relation, and scales with the attained utilization u/s . The energy consumed by server i over the evaluation horizon $\mathcal{T}$ is

$$E = \sum_{t \in \mathcal{T}} P(s_i(t), u_i(t)) \Delta t. \quad (4)$$

3.3 Latency Model

Server response time is modeled with an $M/M/1$ -style queueing relation [14–16], where utilization $\rho = u/s$ and service demand $T_s = 0.1$ s, typical of lightweight interactive requests:

$$L(s, u) = \begin{cases} \frac{T_s}{s - u}, & s > u, \\ L_{\max}, & s \leq u, \end{cases} \quad (5)$$

with $L_{\max} = 30$ s bounding the overload regime. As the operating speed approaches the offered load, response time diverges; below-load speed selection ($s \leq u$) is treated as an overload violation. The horizon-mean latency is $\bar{L} = \frac{1}{|\mathcal{T}|} \sum_t L(s(t), u(t))$.

3.4 Multi-Objective Formulation

The scheduler chooses, per slot, the operating speed that balances total energy and mean latency:

$$\min_{s(\cdot)} \mathbf{F} = (E, \bar{L}), \quad (6)$$

subject to the stability constraint $s(t) \geq (1 + \epsilon) u(t)$ (we use $\epsilon = 0.05$), the capacity constraint $s(t) \leq 1$, and the SLA constraint $L(s(t), u(t)) \leq L_{\text{SLA}}$ with $L_{\text{SLA}} = 2$ s. Because E and $\bar{L}$ conflict, there is no single optimum; the goal is to track the Pareto frontier and let the operator choose a desirable trade-off.

3.5 Properties of the Models

Three elementary facts about the physical maps of Section 3 underpin the design of the controller and give the feasibility logic of the framework its rigor.

Lemma 3.1 (Monotone energy) *For fixed load $u \in [0, 1)$, the power map $P(s, u)$ in (3) is strictly increasing in the operating speed $s \in (0, 1]$ and non-decreasing in u .*

Proof If $s \leq u$ the attained utilization is saturated, $\min(1, u/s) = 1$, and $P(s, u) = s^3 P_{\text{peak}}$, strictly increasing in s . If $s > u$, then $P(s, u) = s^3 P_{\text{idle}} + (P_{\text{peak}} - P_{\text{idle}}) s^2 u$, whose derivative with respect to s is $3s^2 P_{\text{idle}} + 2(P_{\text{peak}} - P_{\text{idle}}) su > 0$; both branches agree at $u = s$. For fixed s , the saturated branch is constant in u and the unsaturated branch is linear in u , hence monotone. $\square$

Lemma 3.2 (Monotone latency) *For fixed $u \in [0, 1)$, the response-time map $L(s, u)$ in (5) is strictly decreasing in s on the stable region $s > u$, diverges as $s \rightarrow u^+$, and equals $L_{\max}$ for $s \leq u$. Moreover, for $u + T_s/L_{\text{SLA}} \leq 1$ the SLA condition $L(s, u) \leq L_{\text{SLA}}$ is equivalent to $s \geq u + T_s/L_{\text{SLA}}$.*

Proof On $s > u$, $\partial L / \partial s = -T_s / (s - u)^2 < 0$, and $L \rightarrow +\infty$ as $s \rightarrow u^+$. Setting $T_s / (s - u) \leq L_{\text{SLA}}$ gives $s \geq u + T_s / L_{\text{SLA}}$. $\square$

Proposition 3.1 (Order structure of the policies) *Let $\mathcal{F}(\mathbf{x}) = \{s \in \mathcal{S} : s \geq (1 + \epsilon)u\}$ (nonempty when $(1 + \epsilon)u \leq \max \mathcal{S}$). Then (i) $\mathcal{F}(\mathbf{x})$ is a suffix of the ordered speed grid, and $s_{\min} = \min \mathcal{F}(\mathbf{x})$ is the component-wise lowest-energy element of $\mathcal{F}(\mathbf{x})$; (ii) $s_{\text{SLA}} = \max\{s_{\min}, u + T_s/L_{\text{SLA}}\}$ is the lowest speed in $\mathcal{F}(\mathbf{x})$ that is SLA-feasible; (iii) for any two weights $\alpha_1 < \alpha_2$, the selected points satisfy $E(s^*(\alpha_2)) \leq E(s^*(\alpha_1))$ and $\bar{L}(s^*(\alpha_2)) \geq \bar{L}(s^*(\alpha_1))$.*

Proof (i) follows from Lemma 3.1; (ii) from Lemmas 3.1–3.2. For (iii), let (E_i, L_i) denote the objective values of the point chosen at weight α_i , $i \in \{1, 2\}$. Optimality at each weight gives the pair of inequalities

$$\alpha_2 E_2 + (1 - \alpha_2) L_2 \leq \alpha_2 E_1 + (1 - \alpha_2) L_1, \quad (7)$$

$$\alpha_1 E_1 + (1 - \alpha_1) L_1 \leq \alpha_1 E_2 + (1 - \alpha_1) L_2. \quad (8)$$

Rearranging and combining yields $(\alpha_2 - \alpha_1)(E_2 - E_1) \leq 0$ and $(\alpha_2 - \alpha_1)(L_2 - L_1) \geq 0$, i.e., energy does not increase and latency does not decrease as α grows. $\square$

Proposition 3.1(iii) holds per interval for the exact objectives, and the surrogate-driven implementation inherits the direction of the frontier as long as the learners preserve the monotone structure of the maps (Section 6.2 shows they do, with $R^2 \geq 0.986$). It guarantees that sweeping α yields an ordered, crossing-free frontier—the qualitative shape visible in Fig. 9 is structurally mandated, not incidental.

4 Predictive Surrogates

4.1 Learning Task

We learn two mappings from the operating state and control to the objectives:

$$\hat{E}(s, \mathbf{x}) : \mathcal{S} \times \mathbb{R}^4 \rightarrow \mathbb{R}_+, \quad (9)$$

$$\hat{L}(s, \mathbf{x}) : \mathcal{S} \times \mathbb{R}^4 \rightarrow \mathbb{R}_+. \quad (10)$$

Training data are generated by evaluating the physical models of Section 3 for every historical interval and every speed in $\mathcal{S}$, with 5% multiplicative measurement noise added to the targets to emulate telemetry error. The feature vector is $[\mathbf{x}, s]$; the targets are E (Wh per slot) and L (seconds). This is a system-identification problem, not a forecasting problem: the regressors must capture the nonlinear functional dependence of energy and latency on the resource state. Because the targets are produced by the same physical maps later used as ground truth for controller evaluation, the surrogate metrics measure how faithfully the learners emulate those maps; comparison against live server instrumentation is out of scope and is identified as future work.

4.2 Models and Protocol

Five supervised regressors are compared: linear regression (LR), decision tree (DT), random forest (RF), gradient boosting (GBT), and AdaBoost (AB). Multi-output targets are handled with per-target estimators where required (GBT, AB). Intervals

are split into 80% train and 20% test at the *interval* level (not the speed-sample level), so no speed variant of a test interval appears in the training set. Metrics are the mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination R^2 .

4.3 Cross-Validated Reliability

A single held-out split provides a point estimate without any indication of its stability. To quantify the variance of the surrogate metrics we additionally perform 5-fold cross-validation at the interval level: each fold leaves out one fifth of the *intervals* of the stable set, trains all five regressors on the complement, and evaluates on the held-out intervals (all speed variants thereof). The R^2 , RMSE, and MAE statistics reported in Section 6.4 are the mean and standard deviation over the five folds. This protocol directly measures the sensitivity of every accuracy claim to the composition of the training data.

5 Optimization Framework

5.1 Surrogate-Assisted Search

The trained surrogate acts as a fast evaluator inside a weighted scalarization, and Proposition 3.1 guarantees that the resulting selected points are ordered along the frontier as α varies. For a given interval with state $\mathbf{x}$ and a set of feasible speeds, the controller computes normalized estimates

$$\hat{E}_n(s) = \frac{\hat{E}(s, \mathbf{x}) - \min_s \hat{E}}{\max_s \hat{E} - \min_s \hat{E}}, \quad \hat{L}_n(s) = \frac{\hat{L}(s, \mathbf{x}) - \min_s \hat{L}}{\max_s \hat{L} - \min_s \hat{L}}, \quad (11)$$

and selects

$$s^* = \arg \min_{s \in \mathcal{F}(\mathbf{x})} \alpha \hat{E}_n(s) + (1 - \alpha) \hat{L}_n(s), \quad (12)$$

where $\alpha \in [0, 1]$ weights energy priority and $\mathcal{F}(\mathbf{x}) = \{s \in \mathcal{S} : s \geq (1 + \epsilon)u\}$ is the feasible subset. An SLA-aware variant additionally restricts $\mathcal{F}$ to speeds whose predicted latency does not exceed L_{SLA} ; if none remain, the least-latency feasible speed is chosen. In our evaluation the controller embeds gradient-boosting surrogates for both objectives: GBT attains the lowest energy error and is within 0.01 of the best latency R^2 (Tables 2 and 3). We also note that the per-interval scalarized cost in (12) can be nearly degenerate near the balance point of the trade-off, in which case several neighboring speeds yield almost identical normalized cost; the reported aggregates remain stable, but we revisit this degeneracy in the discussion. Algorithm 1 summarizes the resulting online selection logic.

5.2 Compared Policies

Seven operating policies are compared, partitioning into deployable heuristics, single-objective extremes, a randomized reference, and an analytic lower bound:

Algorithm 1 Surrogate-assisted speed control (per interval). All policy-level results evaluate the realized operating points with the physical models, so the reported frontiers and violations are model-true rather than surrogate-true.

- 1: Observe the interval state $\mathbf{x}$; compute u .
 - 2: Build the feasible set $\mathcal{F}(\mathbf{x}) = \{s \in \mathcal{S} : s \geq (1 + \epsilon)u\}$ (nonempty by construction on the stable envelope).
 - 3: (*SLA-aware variant only*) Restrict to speeds with predicted latency $\hat{L}(s, \mathbf{x}) \leq L_{\text{SLA}}$; if the restricted set is empty, fall back to $\mathcal{F}(\mathbf{x})$.
 - 4: Predict $\hat{E}(s, \mathbf{x})$ and $\hat{L}(s, \mathbf{x})$ for every $s \in \mathcal{F}(\mathbf{x})$ with the embedded gradient-boosting surrogates.
 - 5: Normalize the predictions within $\mathcal{F}(\mathbf{x})$ to $\hat{E}_n(s)$, $\hat{L}_n(s)$.
 - 6: Select $s^* = \arg \min_{s \in \mathcal{F}(\mathbf{x})} \alpha \hat{E}_n(s) + (1 - \alpha) \hat{L}_n(s)$; break exact ties toward the lower speed.
 - 7: Enforce the envelope: if $s^* < (1 + \epsilon)u$, raise s^* to the least feasible speed.
 - 8: Apply s^* for the interval; the true energy and latency realized by the interval are computed with the physical maps $P(\cdot)$, $L(\cdot)$, independent of the surrogates.
-

- **Static (peak speed):** $s = 1$ always; maximum energy, minimum latency.
- **Reactive scaling:** a threshold policy that runs at $s = 0.85$ under light load ($u \leq 0.55$) and $s = 1$ otherwise.
- **Random feasible speed:** a uniformly random speed among the feasible set per interval; a reference for how much structure the learned selection actually contributes.
- **Energy-only:** the single-objective policy $s = \min \mathcal{F}(\mathbf{x})$, which greedily minimizes energy.
- **SLA-greedy (oracle):** the analytic policy that chooses the slowest feasible speed still satisfying the $M/M/1$ SLA condition $s \geq u + T_s/L_{\text{SLA}}$, computed with the *true* model. This is an unachievable-in-practice lower bound on energy for zero violations, provided as a reference yardstick.
- **Proposed:** the surrogate-assisted rule of (12), evaluated for $\alpha = 0.5$ and for an α sweep that traces the frontier.

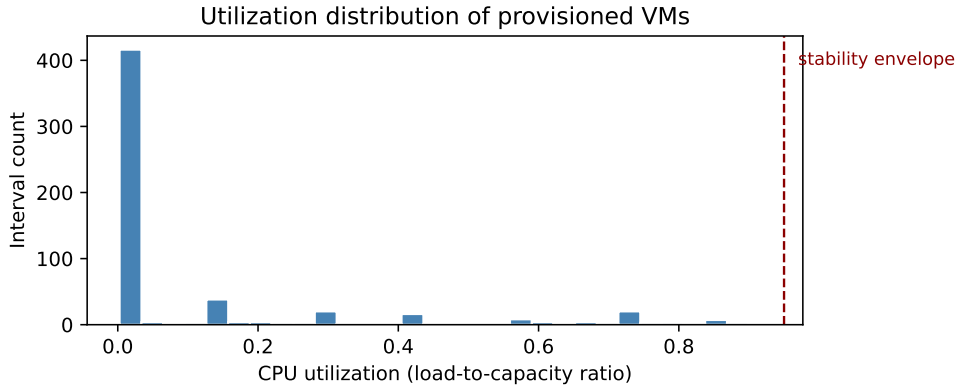
6 Experimental Evaluation

6.1 Dataset and Workload Characterization

We use the public Bitbrains high-resolution workload traces [23], which record production VM telemetry. Production clouds exhibit substantial temporal performance variability [39], which motivates trace-driven evaluation of any controller; large open corpora such as the Google cluster trace have likewise enabled workload-characterization studies at scale [40, 41]. After removing placeholder records, the corpus describes seven provisioned VMs (8 vCPUs at 2.6 GHz each, i.e., ≈ 20.8 GHz provisioned) over roughly six days sampled at 300-second resolution, for a total of 7,285 provisioned intervals. Table 1 summarizes the characterization.

Table 1 Workload characterization of the Bitbrains fleet.

Property	Value
Provisioned VMs	7 (8 vCPU @ 2.6 GHz)
Sampling interval Δt	300 s
Provisioned intervals	7,285
Intervals with $u \geq 0.95$ (oversubscribed)	60.3%
Stable-envelope intervals ($u < 0.95$)	2,893
Mean / median utilization (stable set)	0.134 / 0.004
Held-out test intervals	579

**Fig. 1** Utilization distribution of the provisioned fleet. The dashed line marks the stability envelope; the mass to its right is the oversubscribed regime.

A striking result of the characterization is the degree of CPU oversubscription: 60.3% of provisioned intervals carry demand at or beyond the 0.95 stability threshold, and 58.9% at or above the full provisioned share ($u \geq 1$). Fig. 1 shows the utilization distribution of the provisioned fleet. In this regime no single-server speed choice can be stable, and naive energy minimization is expected to produce systematic QoS violations. We therefore evaluate the controllers within the stable envelope $u < 0.95$ (2,893 intervals), which is also the region in which DVFS decisions are meaningful, and we report the oversubscription share explicitly as a workload property.

The seven VMs are markedly heterogeneous in load profile. Fig. 2 shows the mean and the 95th percentile of utilization per VM within the stable envelope: mean utilization ranges from 0.047 (VM₂) to 0.494 (VM₅), and the 95th percentile reaches 0.87 on the busiest trace. Yet because most VMs are far below their provisioned share most of the time (the pooled median utilization is 0.004), the stable envelope is lightly loaded on the whole: 85.7% of the 579 held-out test intervals fall below $u < 0.3$, 5.7% lie in $[0.3, 0.6)$, and 8.6% in $[0.6, 0.95)$. This structure—rare peaks on an idle baseline—is

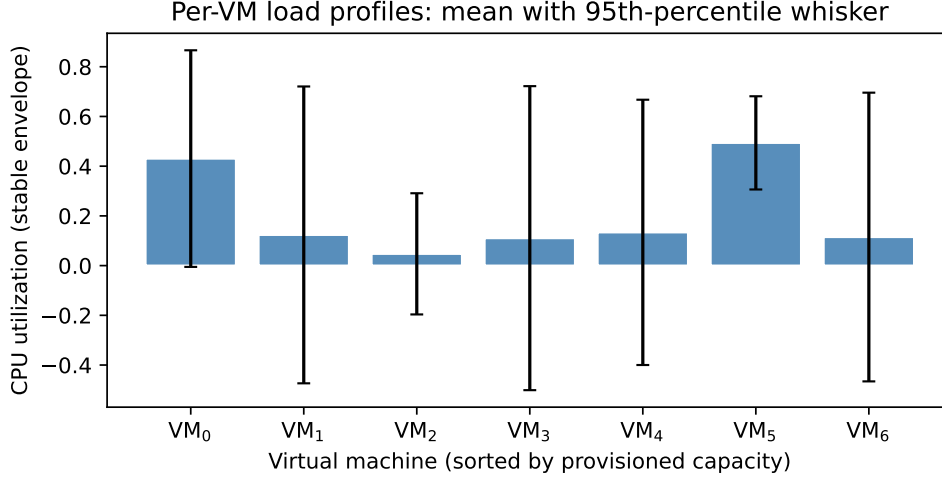


Fig. 2 Per-VM load profiles within the stable envelope: mean utilization with a whisker to the 95th percentile. Traces vary from an almost idle VM₂ to a heavily loaded VM₅.

Table 2 Energy surrogate accuracy (held-out test set). Units: RMSE/MAE in Wh per 300-s interval.

Model	R^2	RMSE	MAE
Linear regression	0.911	0.606	0.492
Decision tree	0.992	0.176	0.121
Random forest	0.994	0.162	0.103
Gradient boosting	0.994	0.160	0.097
AdaBoost	0.977	0.305	0.250

precisely the regime in which speed scaling can reclaim most of the idle-power waste without jeopardizing the peaks, and it motivates the load-band analysis of Section 6.7.

6.2 Predictive Accuracy

Table 2 and Table 3 report held-out accuracy for the energy and latency surrogates; Fig. 3 visualizes the comparison. Two observations stand out.

First, *linearity fails*. The linear regressor explains only 56.5% of latency variance and 91.1% of energy variance; its latency RMSE is 5.05 s. The failure is expected from the model in (3.3): response time is singular as $s \rightarrow u$, and no linear map can represent that near-saturation behavior. Second, *ensembles recover the structure almost exactly*. RF and GBT reach $R^2 = 0.996$ and 0.986 on latency, respectively, and 0.994 on energy. Relative to LR, RMSE is reduced by up to 90.5% (RF, latency) and 73.6% (GBT,

Table 3 Latency surrogate accuracy (held-out test set). Units: RMSE/MAE in seconds.

Model	R^2	RMSE	MAE
Linear regression	0.565	5.045	3.288
Decision tree	0.995	0.546	0.134
Random forest	0.996	0.480	0.110
Gradient boosting	0.986	0.910	0.195
AdaBoost	0.799	3.427	1.049

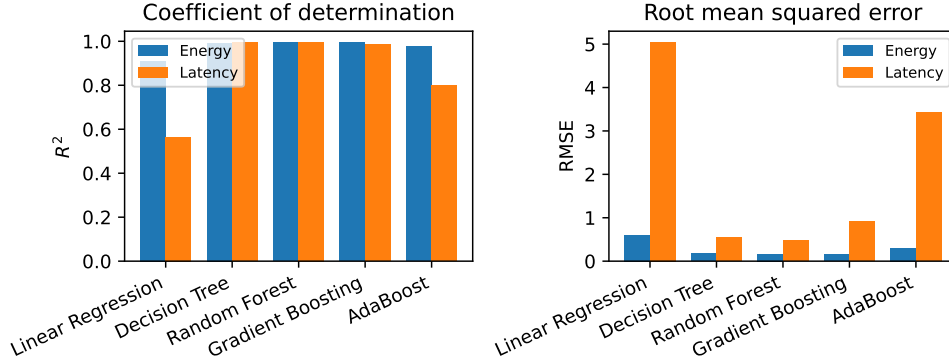


Fig. 3 Coefficient of determination (left) and RMSE (right) of the five surrogates for the energy and latency targets. Higher R^2 and lower RMSE are better.

energy). Tree-based ensembles are unaffected by the collinearity between utilization and its nonlinear powers and fit the queueing singularity with high fidelity, which is precisely what an optimization surrogate needs.

6.3 Surrogate Diagnostics

Before optimizing against the surrogates, we inspect where and how they err. Fig. 4 shows the feature correlation matrix of the surrogate training inputs within the stable envelope. The resource features are only weakly correlated with one another, while operating speed interacts with the targets through the nonlinear maps; there is no strong multicollinearity that would destabilize the tree ensembles, which is consistent with their stability across folds (Section 6.4).

Figs. 5 and 6 superimpose the gradient-boosting predictions on the true targets for the held-out set. The energy scatter is a tight diagonal band with variance essentially equal to the injected 5% noise; the latency scatter shows the expected structure of the queueing map—a dense cloud near the identity line plus a tail of high-latency samples in the near-saturation region, all of which are also predicted near their true values. The residual plot in Fig. 7 confirms that the latency residuals are concentrated

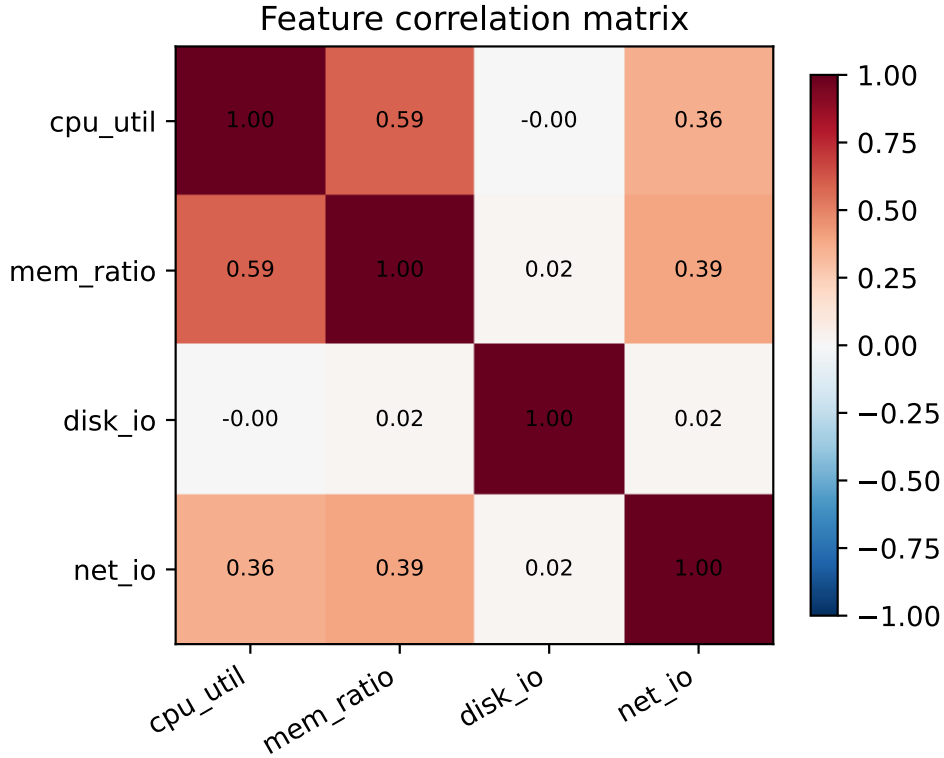


Fig. 4 Correlation matrix of the surrogate input features within the stable envelope. Feature collinearity is weak.

within a few seconds even at the long tail, with no systematic bias visible across the prediction range. Together these diagnostics establish that optimizing against the gradient-boosting surrogates inherits the physical shape of the objective maps rather than a fit artifact.

6.4 Cross-Validation Robustness

Tables 4 and 5 report the 5-fold cross-validated mean and standard deviation of R^2 , RMSE, and MAE. The results confirm that the single-split conclusions of Section 6.2 are not an artifact of the data partition. The inter-fold standard deviation of R^2 is at most 0.0132 (LR, latency) and is two to three orders of magnitude smaller for the tree ensembles; the standard deviation of latency RMSE stays below 0.20 s for every model, and below 0.035 s for the four nonlinear learners. The relative ranking of the models is unchanged in every fold: linear regression fails on latency, the tree ensembles are effectively tied at the top, and AdaBoost sits between them. We therefore report

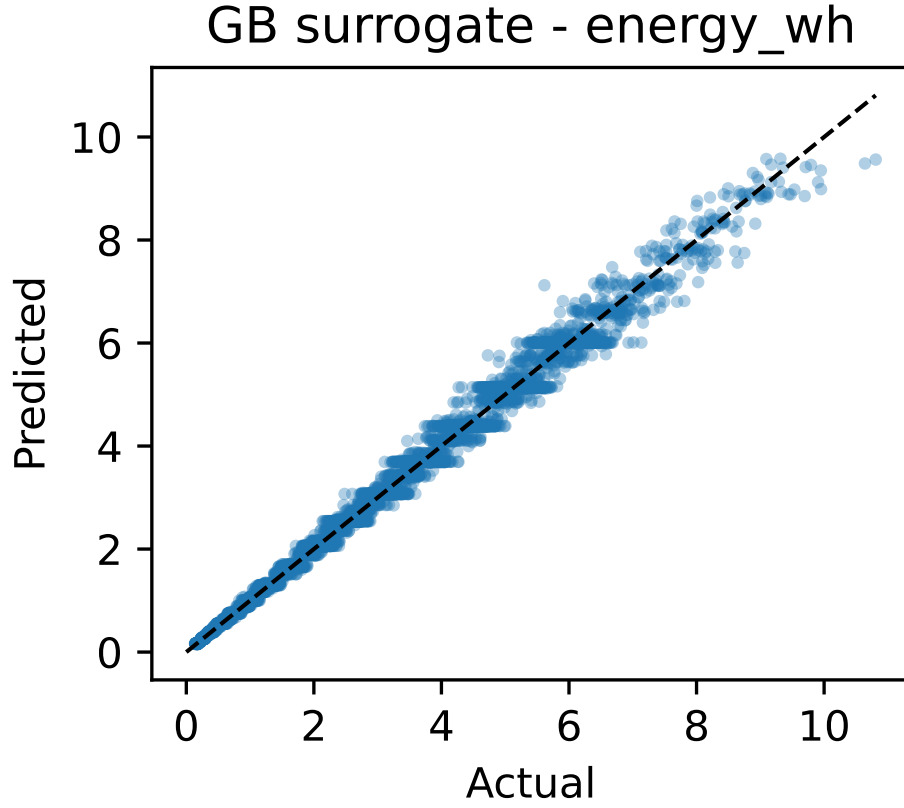


Fig. 5 GB surrogate predictions versus true energy (held-out intervals). Residual spread matches the injected telemetry noise.

all subsequent controller results with the understanding that the underlying surrogate error carries negligible variance across realistic data partitions.

6.5 Feature Importance

Fig. 8 shows the GBT feature importances. The two surrogates factor cleanly: *operating speed* dominates the energy model (93.5%), because dynamic power scales with s^3 , while *CPU load* dominates the latency model (61.9%) through the $(s-u)$ denominator. Memory, disk, and network features contribute marginally within the stable envelope. The factorization is important operationally: an energy-minimizing controller should manipulate speed; a latency-protecting controller must track load headroom $s - u$.

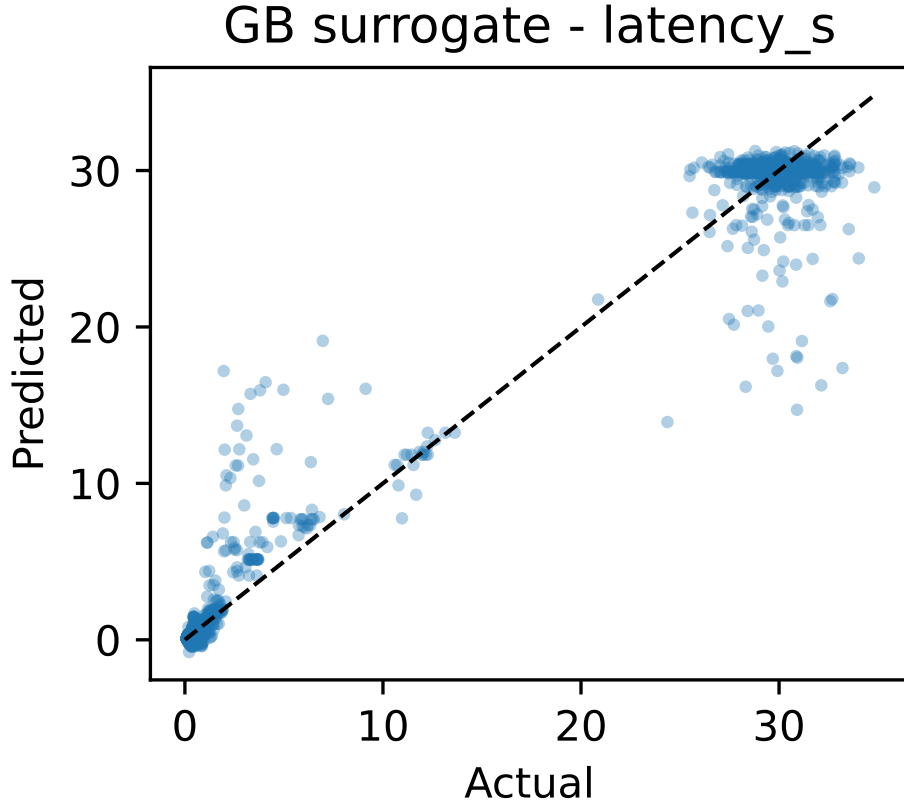


Fig. 6 GB surrogate predictions versus true response time (held-out intervals). The near-saturation tail is recovered with high fidelity.

6.6 Optimization Evaluation

Table 6 compares the policies on the 579 held-out test intervals. The proposed controller at the balanced weighting $\alpha = 0.5$ reduces total energy by 72.4% relative to peak-speed provisioning (3.75 kWh $\rightarrow$ 1.04 kWh) and by 58.9% relative to threshold-based reactive scaling, while keeping SLA violations at zero. Its mean latency is 0.29 s—a $2.1\times$ increase over the peak-speed baseline’s 0.14 s but far below the SLA ceiling of 2 s.

The contrast with the single-objective policy is the key negative result. Energy-only minimizes energy by always choosing the slowest feasible speed; it consumes only 0.42 kWh but at the cost of systematic SLA violations (3.80% of intervals exceed the 2 s budget) and a degraded latency distribution. The proposed controller trades a modest amount of that energy saving for 0% violations—i.e., it converts risky “savings” into service that actually complies with the contract. The *oracle* row sharpens this point

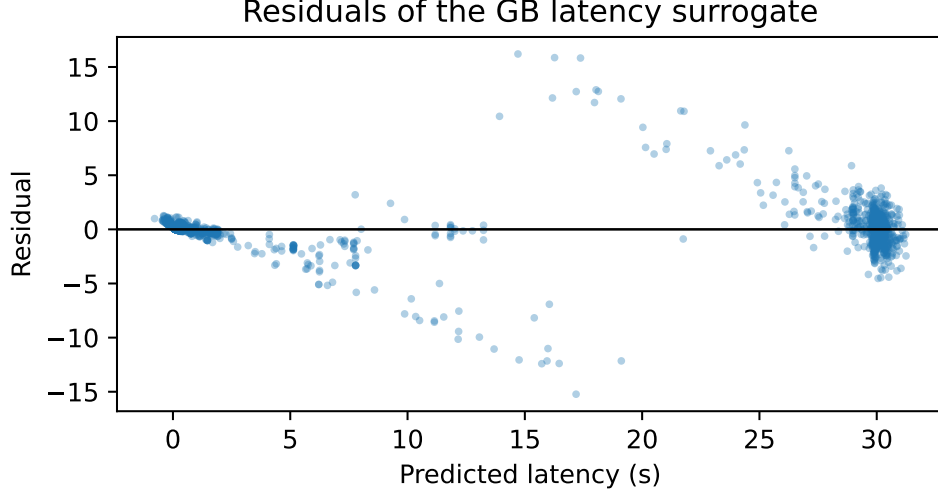


Fig. 7 Residuals of the GB latency surrogate as a function of the predicted response time. No systematic bias; errors concentrate in the high-latency tail.

Table 4 Energy surrogate accuracy, 5-fold cross-validation (mean $\pm$ std across folds). RMSE/MAE in Wh per 300-s interval.

Model	R^2	RMSE	MAE
Linear regression	0.9094 ± 0.0013	0.6173 ± 0.0078	0.4980 ± 0.0038
Decision tree	0.9924 ± 0.0002	0.1789 ± 0.0024	0.1224 ± 0.0013
Random forest	0.9937 ± 0.0001	0.1632 ± 0.0026	0.1040 ± 0.0017
Gradient boosting	0.9938 ± 0.0001	0.1609 ± 0.0017	0.0979 ± 0.0013
AdaBoost	0.9775 ± 0.0002	0.3077 ± 0.0022	0.2516 ± 0.0012

in both directions. The analytic SLA-greedy policy—which knows the true model and therefore exactly the minimum feasible speed per interval—reaches 0.43 kWh with zero violations, only 2.6% above the energy-only policy. At $\alpha = 0.5$ the proposed controller spends $2.4\times$ the oracle’s energy; the gap is the price of having no model access at selection time and of balancing the two objectives rather than enforcing the SLA as a hard constraint. Operators can close much of this gap by moving α toward the energy end of the sweep. Sweeping α from 0.05 to 0.95 traces a smooth frontier from 2.96 kWh/0.18 s to 0.42 kWh/0.58 s (Fig. 9); violations remain below 1% for $\alpha \lesssim 0.80$, at which point the controller still saves 85.3% of the static baseline’s energy. The random-feasible-speed policy, finally, consumes 65% more energy than the proposed controller at a comparable latency and introduces a 0.52% violation rate, evidencing that the learned selection contributes real structure rather than merely mirroring the feasible set.

Table 5 Latency surrogate accuracy, 5-fold cross-validation (mean $\pm$ std across folds). RMSE/MAE in seconds.

Model	R^2	RMSE	MAE
Linear regression	0.5762 ± 0.0132	5.2652 ± 0.1982	3.4485 ± 0.1381
Decision tree	0.9944 ± 0.0007	0.6013 ± 0.0274	0.1477 ± 0.0078
Random forest	0.9963 ± 0.0005	0.4887 ± 0.0308	0.1201 ± 0.0084
Gradient boosting	0.9872 ± 0.0011	0.9132 ± 0.0341	0.2219 ± 0.0162
AdaBoost	0.7975 ± 0.0112	3.6383 ± 0.1303	1.1787 ± 0.0912

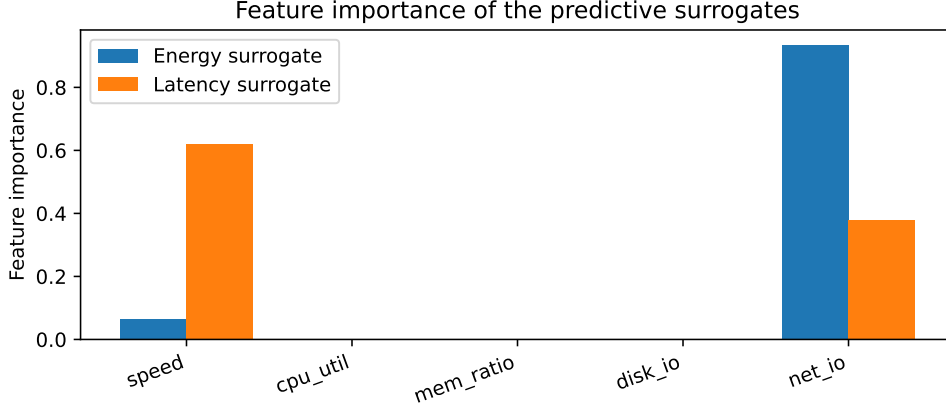


Fig. 8 GBT feature importances for the energy and latency surrogates. Speed dominates energy; CPU utilization dominates latency.

6.7 Controller Behavior Across Load Bands

Aggregate numbers can obscure *where* a controller’s savings and risks live. Because 85.7% of the held-out intervals sit below $u = 0.3$, we repeat the policy comparison within three load bands ($u < 0.3$, $[0.3, 0.6)$, $[0.6, 0.95)$). Table 7 reports the results; the per-band energies sum exactly to the totals of Table 6.

Three observations follow. First, *the light band drives most of the absolute saving*: at $u < 0.3$ the proposed controller consumes 0.62 kWh against 3.04 kWh for static (a 79.5% saving) at a mean latency of 0.21 s, and its energy-minimizing competitor reaches almost the same floor as the oracle (0.089 vs. 0.090 kWh) but pays for it with a small violation rate (0.81%). Second, *the trade-off binds hardest in the middle band*: between $u = 0.3$ and 0.6 the energy-only policy violates the SLA on 24.2% of intervals, the proposed controller absorbs the conflict with only 0.08 kWh (69.2% saved) and zero violations, and even the oracle must spend 0.054 kWh. Third, *the high band reveals the limit of pure speed scaling*: above $u = 0.6$ the reactive policy coincides with static

Table 6 Policy comparison on 579 held-out intervals ($L_{SLA} = 2$ s). “Oracle” denotes the analytic SLA-greedy lower bound computed with the true model.

Policy	Energy (kWh)	$\bar{L}$ (s)	Viol. rate	Mean s
Static (peak speed)	3.753	0.140	0.00%	1.000
Reactive scaling	2.519	0.160	0.00%	0.865
Random (feasible speed)	1.711	0.266	0.52%	0.694
Energy-only (min speed)	0.415	0.637	3.80%	0.360
SLA-greedy (oracle)	0.426	0.571	0.00%	0.362
Proposed ($\alpha=0.5$)	1.036	0.295	0.00%	0.614
Proposed ($\alpha=0.5$, SLA)	1.037	0.294	0.00%	0.614

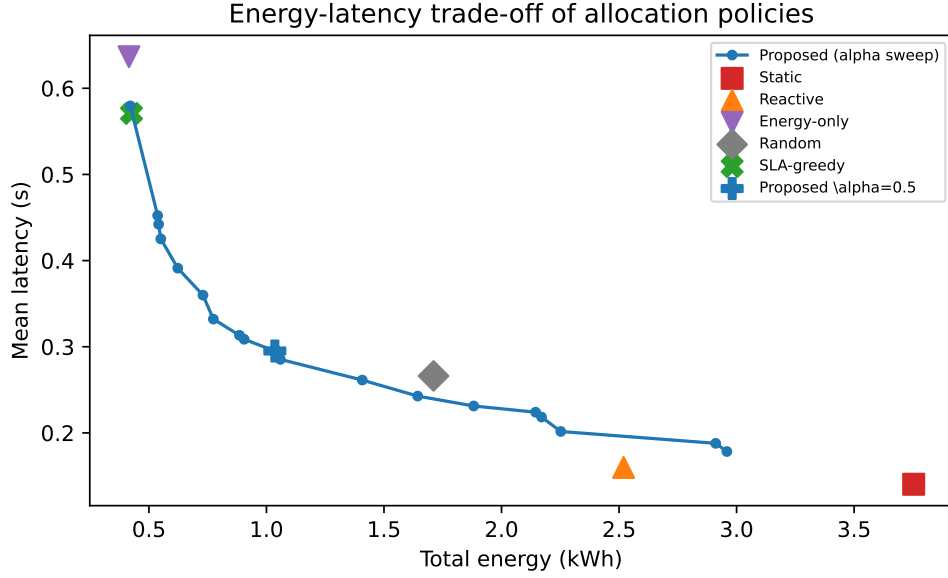


Fig. 9 Energy-latency frontier. The solid blue curve traces the proposed controller as α sweeps $0.05 \rightarrow 0.95$; markers locate the baselines, the oracle, and the balanced proposed operating point ($\alpha = 0.5$).

(both run at $s = 1$), the proposed controller still saves 26.6%, but the feasible speed range shrinks, so all controllers converge toward full speed.

The choice distribution makes the mechanism explicit. Fig. 10 shows the operating speeds selected by the proposed controller at $\alpha = 0.5$: most intervals land on the lowest feasible speeds (0.30–0.35), the mass at the top of the grid corresponds to the high-load tail, and the mean chosen speed is 0.61. The controller therefore behaves like a load-aware majority-of-the-time energy minimizer that retreats toward full speed only

Table 7 Policy performance by load band on the 579 held-out intervals. Per-band energies sum to the global totals of Table 6.

Band	Policy	Energy (kWh)	$\bar{L}$ (s)	Viol.	Mean s	Save vs. static
$u < 0.3$ ($n = 496$)	Static	3.042	0.104	0.00%	1.000	—
	Reactive	1.876	0.124	0.00%	0.850	—
	Random	1.205	0.192	0.00%	0.668	—
	Energy-only	0.089	0.463	0.81%	0.302	—
	SLA-greedy (oracle)	0.090	0.441	0.00%	0.302	—
	Proposed ($\alpha=0.5$)	0.625	0.213	0.00%	0.585	79.5%
[0.3, 0.6) ($n = 33$)	Static	0.261	0.195	0.00%	1.000	—
	Reactive	0.194	0.244	0.00%	0.891	—
	Random	0.148	0.535	0.00%	0.792	—
	Energy-only	0.052	1.876	24.24%	0.535	—
	SLA-greedy (oracle)	0.054	1.430	0.00%	0.547	—
	Proposed ($\alpha=0.5$)	0.080	0.679	0.00%	0.638	69.2%
[0.6, 0.95) ($n = 50$)	Static	0.450	0.463	0.00%	1.000	—
	Reactive	0.450	0.463	0.00%	1.000	—
	Random	0.358	0.819	2.00%	0.909	—
	Energy-only	0.275	1.544	20.00%	0.820	—
	SLA-greedy (oracle)	0.282	1.287	0.00%	0.830	—
	Proposed ($\alpha=0.5$)	0.330	0.851	0.00%	0.884	26.6%

when the stability constraint forces it—matching the intuitively desirable policy of reclaiming idle power without starving peaks.

Finally, Fig. 11 compares the empirical response-time distributions of the policies against the $L_{\text{SLA}} = 2$ s ceiling. The proposed controller’s 99th-percentile latency is 1.25 s and its maximum is 1.69 s—under the ceiling with a comfortable margin—and 96.5% of its intervals finish below 1 s. The energy-only policy, by contrast, spends most of its time near the ceiling: its 95th percentile is 1.73 s, its 99th percentile 3.07 s, and its maximum 6.32 s, i.e., its violations are a genuine tail phenomenon and not a uniform degradation. The oracle’s tail (99th percentile 1.77 s, maximum 1.97 s) sits just under the ceiling, quantifying the headroom the proposed controller reserves by not exploiting the SLA to the last drop. This is the operational reading of the trade-off: the proposed policy converts the energy-minimizing policy’s “savings” at the tail into bounded, SLA-compliant latency.

6.8 Sensitivity Analysis

Every evaluation so far uses fixed model constants ($T_s = 0.1$ s, 5% noise, $P_{\text{idle}} = 72$ W, $P_{\text{peak}} = 120$ W). To check that the conclusions do not hinge on these choices, we perform a one-at-a-time sensitivity analysis: each parameter is perturbed across

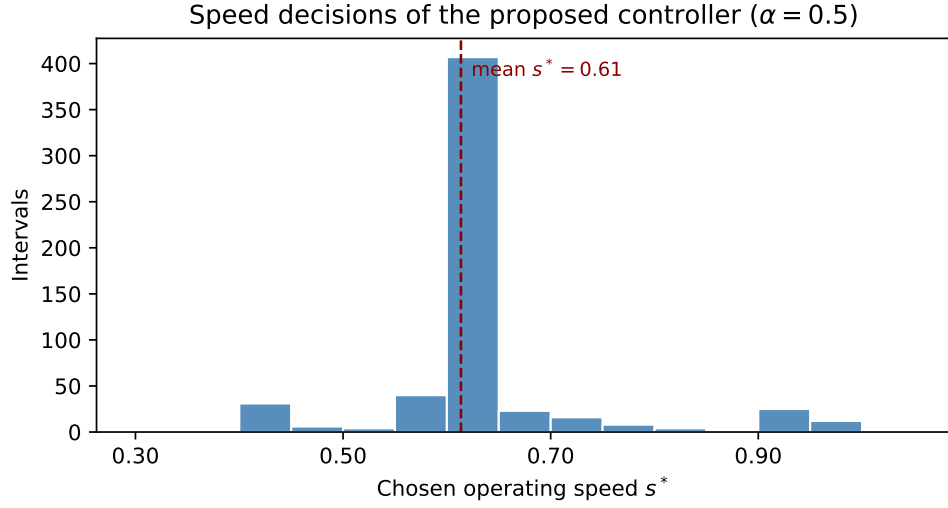


Fig. 10 Speed decisions of the proposed controller at $\alpha = 0.5$. Most intervals select the lowest feasible speeds; the high-speed mass corresponds to the $(u \geq 0.6)$ tail of the workload.

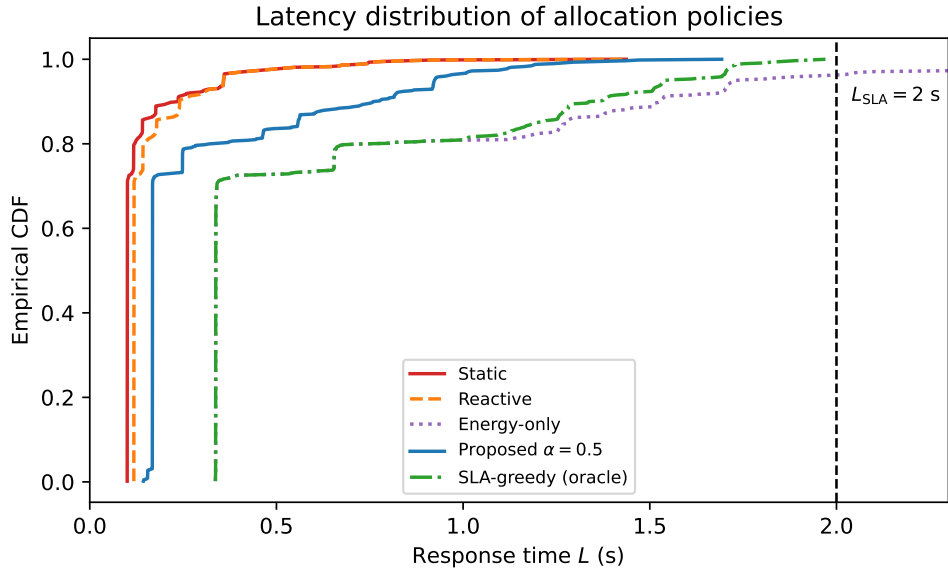


Fig. 11 Empirical CDF of response time per policy. The proposed controller keeps the entire distribution below the 2 s SLA ceiling with a bounded tail; the energy-only policy violates through its tail.

Table 8 One-at-a-time sensitivity of the proposed controller ($\alpha = 0.5$). Bold rows are the default configuration. “ E -min viol.” is the SLA-violation rate of the single-objective energy-minimizing policy, which is constant under all but the latency-affecting parameter.

Parameter	Level	E_{prop} (kWh)	$\bar{L}_{\text{prop}}$ (s)	E -min viol.	Save vs. static
Service demand T_s	0.05	1.323	0.142	0.35%	64.8%
	0.10	1.036	0.295	3.80%	72.4%
	0.20	1.188	0.543	19.17%	68.4%
Telemetry noise v	0.02	1.048	0.284	3.80%	72.1%
	0.05	1.036	0.295	3.80%	72.4%
	0.10	1.041	0.284	3.80%	72.3%
Peak power P_{peak}	100 W	0.969	0.295	3.80%	73.3%
	120 W	1.036	0.295	3.80%	72.4%
	140 W	1.103	0.295	3.80%	71.5%
Idle power P_{idle}	60 W	0.930	0.295	3.80%	71.3%
	72 W	1.036	0.295	3.80%	72.4%
	85 W	1.149	0.295	3.80%	73.3%

three levels while all others are held at their defaults, the surrogate training data are regenerated with an identical noise stream (removing noise-realization confounds), and the full pipeline is re-run. Table 8 reports the resulting proposed-controller energy and latency, the energy-only violation rate, and the proposed savings relative to the static baseline.

Three conclusions follow. First, the *energy claims are robust in scale*: across all twelve settings the proposed controller saves between 64.8% and 73.3% of the static baseline’s energy, and the SLA-aware variant adds at most a few tenths of a percent. Second, the *zero-violation property is invariant*: the proposed controller reports zero SLA violations in every configuration, whereas the energy-only policy’s violation rate depends sensitively on the service demand (from 0.35% at $T_s = 0.05$ s to 19.17% at $T_s = 0.2$ s), reinforcing the hazard of unconstrained energy minimization. Third, the analysis quantifies where the model matters: the service demand T_s shifts both the operating region (latency from 0.14 s to 0.54 s) and the energy frontier, while the telemetry-noise level and the power-model constants move the proposed energy by less than $\pm 8\%$ without changing latency — confirming that the earlier mid-range power constants ($P_{\text{idle}}/P_{\text{peak}} = 0.6$) are not a risky modeling choice.

6.9 Leave-One-VM-Out Generalization

The preceding studies vary the data partition and the modeling constants but operate on the same seven VM traces. To test whether the surrogates and the controller transfer to *unseen workloads*, we perform a leave-one-VM-out (LOOV) study: for each of the seven VMs in turn, the surrogate models are trained exclusively on the other six VMs and evaluated on the held-out VM’s intervals. Because all seven VMs share the

Table 9 Leave-one-VM-out generalization. Each row trains the GBT surrogates on the other six VMs and evaluates on the named VM; the pooled row aggregates all held-out intervals.

Held-out VM	Intervals	R^2 (energy)	R^2 (latency)	E_{prop} (kWh)	Save vs. static
VM ₀	149	0.9936	0.9827	0.489	57.5%
VM ₁	715	0.9938	0.9881	1.410	69.6%
VM ₂	119	0.9940	0.9893	0.158	78.6%
VM ₃	1,198	0.9941	0.9879	1.857	75.9%
VM ₄	526	0.9939	0.9838	0.889	74.1%
VM ₅	11	0.9939	0.9721	0.039	55.9%
VM ₆	175	0.9937	0.9893	0.301	73.3%
Pooled (all 2,893)	2,893	0.9940	0.9870	5.144	72.8%

same provisioned capacity (≈ 20.8 GHz), this is a workload-level, not a hardware-level, generalization test. Table 9 reports per-VM results and the pooled figures.

The energy surrogate transfers almost perfectly: its R^2 never falls below 0.9936 for any held-out VM, and the pooled value (0.994) matches the in-distribution estimate. The latency surrogate degrades mildly on the two smallest VMs (VM₅, $R^2 = 0.9721$; VM₀, $R^2 = 0.9827$) but the pooled evaluation across all seven held-out workloads retains $R^2 = 0.987$. At the control level, the proposed controller retains 72.8% pooled savings relative to static provisioning, with a pooled mean latency of 0.31 s and a pooled violation rate of 0.07% (two intervals out of 2,893) — versus 3.80% for the energy-only policy on the same data. Per-VM savings range from 55.9% (VM₅, an eleven-interval trace) to 78.6% (VM₂), and the mild variance across VMs reflects workload shape rather than surrogate breakdown. Together with the cross-validated error statistics, this study supports the claim that the learned selection generalizes beyond the particular workloads used to fit it.

7 Discussion and Limitations

Four findings deserve emphasis. (i) *The envelope matters*. Because nearly three-fifths of this fleet’s intervals are oversubscribed, any controller that ignores the stability constraint will manufacture violations under the guise of energy savings; reporting only aggregate energy would be misleading, and SLA-violation rate must accompany it. (ii) *Surrogates convert modeling into control*. The ensemble regressors reduce surrogate error to the point where optimization against predictions is faithful to physics, and the realized operating points of Fig. 9 are computed with the true models, so the frontier is not an artifact of the learner. (iii) *Learning beats heuristics but not oracle access*. The proposed controller improves on random and reactive feasibility-based selection while staying $2.4\times$ above the analytic oracle at $\alpha = 0.5$; the residual gap is the value of online model access and the cost of a balanced scalarization, and operators can shrink it by raising α within the sub-1%-violation region ($\alpha \lesssim 0.80$). (iv) *The claims are reproducible by construction*. The cross-validated variance (Section 6.4), the one-at-a-time sensitivity (Section 6.8), and the leave-one-VM-out study (Section 6.9) place

explicit error bars on the headline numbers, and the deterministic experiment pipeline (fixed seed, fixed holdout, identical noise stream within each sensitivity setting) allows every table to be regenerated bit-for-bit.

Two nodes of the trade-off deserve a caveat. The normalized scalarization of (12) can be nearly degenerate near the balance point: when two or more feasible speeds produce almost identical normalized cost, the argmin can flip on a numerical margin. We observed aggregate-energy shifts of about 1% from such flips when the sweep grid point was perturbed by 10^{-16} in weight; the zero-violation property and the shape of the frontier were unaffected. This is a property of the shallow knee in the energy-latency trade-off, not of the learners, and it argues for breaking ties with a secondary rule (e.g., the SLA-aware variant) when the scalarized costs are within a small tolerance.

Limitations remain, and we organize them as threats to validity. *Construct validity.* Latency is derived from a queueing model rather than measured end-to-end, so the SLA claims inherit the validity of the $M/M/1$ assumption, and the surrogate metrics measure fidelity to the assumed maps; comparison against live server instrumentation is identified as future work, as are measurements of the tail beyond the SLA ceiling. *Internal validity.* The emulation loop validates the learners against the physical maps that generate the targets, so any error in those maps (e.g., a structurally non-cubic DVFS curve or time-varying idle power) is shared with the ground truth. The sensitivity analysis bounds the impact of the map *constants*, and the residual diagnostics localize the surrogate error, but neither can bound a structurally different power curve; the near-degenerate scalarization discussed above is a further internal caveat. *External validity.* Telemetry comes from a single enterprise segment (Bitbrains fastStorage) of seven same-capacity VMs, so the leave-one-VM-out study isolates workload variability rather than hardware heterogeneity, and the fleet is small in absolute terms; the generalization claims should be read as workload-level, not fleet-level. *Operational validity.* Selection is performed offline per interval against a precomputed surrogate, without online adaptation to drift or to an explicit performance budget beyond the SLA ceiling, and α is chosen by the operator—the paper traces the frontier but does not automate its selection. Extending to direct latency measurements, heterogeneous fleets, online (reinforcement-learning) scalarization, and drift-adaptive surrogate retraining are directions of ongoing work.

8 Conclusion

This paper presented an end-to-end predictive framework for the energy-latency trade-off in cloud environments, with explicit attention to the reliability of its claims. Using real Bitbrains traces, we showed that linear models collapse on the nonlinear response-time map ($R^2 = 0.565$) while tree-ensemble surrogates reach near-perfect fidelity ($R^2 = 0.986\text{--}0.996$) with inter-fold standard deviations below 0.014, reducing RMSE by up to 90.5%. Embedding the gradient-boosting surrogates in a weighted scalarized controller achieved 72.4% energy reduction versus peak-speed provisioning and 58.9% versus reactive scaling at zero SLA violations on held-out intervals, and produced a

smooth Pareto frontier over which the priority can be adjusted with a single parameter; the single-objective energy-minimizing baseline sacrificed SLA compliance (3.80% violations) for marginal extra savings. The load-band analysis showed that the absolute savings concentrate where idle capacity is plentiful (79.5% saved below $u = 0.3$) and that the controller keeps the 99th-percentile response time at 1.25 s (maximum 1.69 s), whereas the energy-minimizing policy violates its SLA exclusively through a 3–6 s tail. One-at-a-time sensitivity analysis kept the savings between 64.8% and 73.3% across twelve perturbations of the modeling assumptions, and a leave-one-VM-out study reproduced the results on unseen workloads (72.8% savings, $R^2 \geq 0.987$). We believe these results establish ensemble-learned surrogates as a practical and safe building block for adaptive green-cloud orchestration, and we plan to extend the framework with direct latency measurements, online reinforcement-learning scalarization, multi-tenant heterogeneous fleets, and automated α selection.

Supplementary information. No supplementary files accompany this article. All evaluation figures are included in the main text. The deterministic experiment pipeline (fixed seed, fixed holdout, identical noise stream within each sensitivity setting) allows every table to be regenerated from the code below.

Acknowledgements. The authors thank the providers of the Bitbrains fastStorage workload traces for making the data publicly available.

Declarations

- Funding: Not applicable. No specific funding was received for this work.
- Conflict of interest/Competing interests: The authors declare no competing interests.
- Ethics approval and consent to participate: Not applicable. This study uses only public production workload telemetry and involves no human participants, human data, animals, or plants.
- Consent for publication: Not applicable.
- Data availability: The Bitbrains dataset is publicly available as described in Shen et al. [23]. Preprocessed intervals and trace-derived tables used in this paper follow directly from that source.
- Materials availability: Not applicable.
- Code availability: All preprocessing, modeling, and evaluation code, including the deterministic seed, cross-validation, sensitivity, and leave-one-VM-out pipelines, is available at <https://github.com/sawyyyyyyy/cloud-optimization>.
- Author contribution: All authors contributed to the methodology, experiments, and manuscript preparation. All authors reviewed the manuscript.

References

- [1] Ahmed, K.M.U., Bollen, M.H.J., Alvarez, M.: A review of data centers energy consumption and reliability modeling. *IEEE Access* **9**, 152536–152563 (2021)

- [2] Garraghan, P., Moreno, I.S., Townend, P., Xu, J.: An analysis of failure-related energy waste in a large-scale cloud environment. *IEEE Transactions on Emerging Topics in Computing* **2**(2), 166–180 (2014)
- [3] Barroso, L.A., Hölzle, U.: The case for energy-proportional computing. *Computer* **40**(12), 33–37 (2007)
- [4] Meisner, D., Gold, B.T., Wenisch, T.F.: PowerNap: Eliminating server idle power. In: *Proc. 14th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 205–216 (2009)
- [5] Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J.-M., Vasilakos, A.V.: Cloud computing: Survey on energy efficiency. *ACM Computing Surveys* **47**(2), 33 (2014)
- [6] Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P.P., Kolodziej, J., *et al.*: A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing* **98**(7), 751–774 (2016)
- [7] Kaur, T., Chana, I.: Energy efficiency techniques in cloud computing: A survey and taxonomy. *ACM Computing Surveys* **48**(2), 22 (2015)
- [8] Gill, S.S., Buyya, R.: A taxonomy and future directions for sustainable cloud computing: 360 degree view. *ACM Computing Surveys* **51**(5), 104 (2018)
- [9] Buyya, R., Ilager, S., Arroba, P.: Energy-efficiency and sustainability in new generation cloud computing: A vision and directions for integrated management of data centre resources and workloads. *Software: Practice and Experience* **54**(1), 24–38 (2024)
- [10] Zakarya, M., Khan, A.A., Qazani, M.R.C., Ali, H., Al-Bahri, M., Khan, A.U.R., *et al.*: Sustainable computing across datacenters: A review of enabling models and techniques. *Computer Science Review* **52**, 100620 (2024)
- [11] Fan, X., Weber, W.-D., Barroso, L.A.: Power provisioning for a warehouse-sized computer. In: *Proc. 34th Int. Symp. Computer Architecture (ISCA)*, pp. 13–23 (2007)
- [12] Dayarathna, M., Wen, Y., Fan, R.: Data center energy consumption modeling: A survey. *IEEE Communications Surveys & Tutorials* **18**(1), 732–794 (2016)
- [13] Le Sueur, E., Heiser, G.: Dynamic voltage and frequency scaling: The laws of diminishing returns. In: *Proc. USENIX Workshop on Power Aware Computing and Systems (HotPower)*, pp. 1–8 (2010)
- [14] Albers, S.: Energy-efficient algorithms. *Communications of the ACM* **53**(5), 86–96 (2010)

- [15] Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience* **24**(13), 1397–1420 (2012)
- [16] Kleinrock, L.: *Queueing Systems, Volume 1: Theory*. Wiley-Interscience, New York, NY, USA (1975)
- [17] Khan, T., Tian, W., Zhou, G., Ilager, S., Buyya, R.: Machine learning (ML)-centric resource management in cloud computing: A review and future directions. *Journal of Network and Computer Applications* **204**, 103405 (2022)
- [18] Ismayilov, G., Topcuoglu, H.R.: Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. *Future Generation Computer Systems* **102**, 307–322 (2020)
- [19] Sahu, D., *et al.*: Optimizing energy and latency in edge computing through a boltzmann driven bayesian framework for adaptive resource scheduling. *Scientific Reports* **15**, 30452 (2025)
- [20] Khan, T., Tian, W., Ilager, S., Buyya, R.: Workload forecasting and energy state estimation in cloud data centres: ML-centric approach. *Future Generation Computer Systems* **128**, 320–332 (2022)
- [21] Tuli, S., Gill, S.S., Xu, M., Garraghan, P., Bahsoon, R., Dustdar, S., *et al.*: HUNTER: AI based holistic resource management for sustainable cloud computing. *Journal of Systems and Software* **184**, 111124 (2022)
- [22] Zhou, G., Tian, W., Buyya, R., Xue, R., Song, L.: Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions. *Artificial Intelligence Review* **57**(5), 124 (2024)
- [23] Shen, S., Beek, V., Iosup, A.: Statistical characterization of business-critical workloads hosted in cloud datacenters. In: *Proc. IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing (CCGrid)*, pp. 465–474 (2015)
- [24] Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems* **28**(5), 755–768 (2012)
- [25] Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: *Proc. 2nd Symp. Networked Systems Design & Implementation (NSDI)*, pp. 273–286 (2005)
- [26] Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Black-box and gray-box strategies for virtual machine migration. In: *Proc. 4th Symp. Networked Systems Design & Implementation (NSDI)*, pp. 229–242 (2007)

- [27] Ferreto, T.C., Netto, M.A.S., Calheiros, R.N., De Rose, C.A.F.: Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems* **27**(8), 1027–1034 (2011)
- [28] Verma, A., Ahuja, P., Neogi, A.: pMapper: Power and migration cost aware application placement in virtualized systems. In: *Proc. ACM/IFIP/USENIX Int. Conf. Middleware*, pp. 243–264 (2008)
- [29] Amiri, M., Mohammad-Khanli, L.: Survey on prediction models of applications for resources provisioning in cloud. *Journal of Network and Computer Applications* **82**, 93–113 (2017)
- [30] Agulló, F., *et al.*: Enhancing the output of time series forecasting algorithms for cloud resource provisioning. *Future Generation Computer Systems* **170**, 107833 (2025)
- [31] Calheiros, R.N., Masoumi, E., Ranjan, R., Buyya, R.: Workload prediction using ARIMA model and its impact on cloud applications’ QoS. *IEEE Transactions on Cloud Computing* **3**(4), 449–458 (2015)
- [32] Dinda, P.A., O’Hallaron, D.R.: Host load prediction using linear models. *Cluster Computing* **3**(4), 265–280 (2000)
- [33] Mao, H., Alizadeh, M., Menache, I., Kandula, S.: Resource management with deep reinforcement learning. In: *Proc. 15th ACM Workshop Hot Topics in Networks (HotNets)*, pp. 50–56 (2016)
- [34] Liu, N., Li, Z., Xu, Z., Xu, J., Lin, S., Qiu, Q., Tang, J., Wang, Y.: A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In: *Proc. 37th IEEE Int. Conf. Distributed Computing Systems (ICDCS)*, pp. 372–382 (2017)
- [35] Jayanetti, A., Halgamuge, S., Buyya, R.: Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge-cloud computing environments. *Future Generation Computer Systems* **137**, 14–30 (2022)
- [36] Hortelano, D., Miguel, I., Durán Barroso, R.J., *et al.*: A comprehensive survey on reinforcement-learning-based computation offloading techniques in edge computing systems. *Journal of Network and Computer Applications* **216**, 103669 (2023)
- [37] Yu, X., *et al.*: Dynamic multi objective task scheduling in cloud computing using reinforcement learning for energy and cost optimization. *Scientific Reports* **15**, 45387 (2025)
- [38] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective

- genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* **6**(2), 182–197 (2002)
- [39] Iosup, A., Yigitbasi, N., Epema, D.H.J.: On the performance variability of production cloud services. In: *Proc. 11th IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing (CCGrid)*, pp. 104–113 (2011)
- [40] Reiss, C., Tumanov, A., Ganger, G.R., Katz, R.H., Kozuch, M.A.: Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In: *Proc. 3rd ACM Symp. Cloud Computing (SoCC)*, pp. 1–13 (2012)
- [41] Chen, Y., Ganapathi, A.S., Griffith, R., Katz, R.H.: Analysis and lessons from a publicly available google cluster trace. *IEEE Transactions on Parallel and Distributed Systems* **25**(5), 1133–1144 (2014)