

Adaptive machine learning for intelligent resource allocation in cloud computing: a comparative study

Agariadne Dwinggo Samala¹, Soha Rawas²

¹Department of Electronics Engineering, Faculty of Engineering, Universitas Negeri Padang, Padang, Indonesia

²Department of Mathematics and Computer Science, Faculty of Science, Beirut Arab University, Beirut, Lebanon

Article Info

Article history:

Received Oct 13, 2025

Revised Aug 20, 2026

Accepted Aug 27, 2026

Keywords:

Adaptive machine learning

Cloud computing

Comparative analysis

Optimization

Resource utilization

ABSTRACT

Efficient cloud resource allocation requires controllers that anticipate workload variation rather than respond only after demand changes. This study evaluates an adaptive machine learning (AML) controller that integrates one-step-ahead random forest forecasting, capacity-constrained resource adjustment, and 24-hour model retraining. A 30-day CloudSim experiment modeled 50 hosts and 200 virtual machines (VM) at five-minute intervals and compared AML with fixed and reactive dynamic allocation using resource utilization, simulated cost, and response time. Across 20 independent AML runs, mean resource utilization was $85.2\% \pm 1.8\%$ (95% confidence interval: 84.4–86.0%), and mean high-load response time was 54.8 ± 3.2 ms. In the descriptive baseline summaries, AML achieved approximately 85% utilization, compared with 75% for dynamic allocation and 60% for fixed allocation, while the corresponding high-load response times were approximately 55, 70, and 100 ms. AML also reduced the rounded simulated cost by 10% relative to the dynamic baseline and 40% relative to fixed allocation. Together, the results indicate that converting near-term forecasts into capacity-constrained actions can improve the utilization–latency–cost balance within the evaluated simulation. The study supports prediction-guided allocation as a practical basis for intelligent cloud resource decisions under controlled workload conditions.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Agariadne Dwinggo Samala

Department of Electronics Engineering, Faculty of Engineering, Universitas Negeri Padang

Padang, Indonesia

Email: agariadne@ft.unp.ac.id

1. INTRODUCTION

Cloud computing provides elastic processing, memory, storage, and network capacity, but burst workloads and heterogeneous service demands make provisioning difficult. Resource-allocation research consequently spans scheduling, load balancing, quality-of-service control, and workload-aware placement [1]–[5]. When allocation lags demand, utilization can fall, response time and energy use can rise, and service-level objectives can be violated [4], [5]. Fixed allocation is simple but cannot adapt to changing demand, whereas reactive dynamic allocation changes capacity only after monitored conditions cross a rule or threshold [6]–[8]. Prediction-guided allocation can act earlier by estimating near-term demand; however, its value must be assessed at the controller level, where forecast timing, capacity constraints, utilization, response time, and cost interact [9], [10].

Recent studies report scalable machine-learning allocation, adaptive forecasting, proactive provisioning, hybrid prediction, and reinforcement-learning controllers [9], [11]–[18]. Nevertheless, many studies emphasize predictor accuracy or compare against a single policy, while others require more complex

decision mechanisms. The specific gap addressed here is therefore a transparent prediction-to-action loop evaluated against both static and reactive baselines with the same system-level metrics. The proposed adaptive machine learning (AML) controller uses recent CPU, memory, network, and disk-I/O observations to forecast one five-minute step ahead, converts the forecast into a capacity-bounded adjustment, and refreshes the predictor every 24 hours. Accordingly, the study asks whether forecast-informed action improves utilization, response time, and simulated cost relative to fixed and reactive allocation, and whether AML outcomes remain stable across repeated seeds and workload-availability scenarios.

The contribution is a systems-level integration rather than a new learning algorithm or a proof of global optimization. Specifically, the study: i) defines an interpretable random-forest prediction-to-action controller, ii) compares fixed, reactive dynamic, and predictive policies using utilization, response time, and simulated cost, and iii) reports sensitivity results and uncertainty across 20 AML runs. The comparative study is therefore explicit from the outset and its claims are bounded to the available CloudSim evidence. The remainder of the paper reviews related work, defines the controller, reports the simulation and analysis procedures, presents the comparative results, discusses validity and deployment implications, and concludes with specific validation priorities.

2. LITERATURE REVIEW

Cloud resource management must reconcile variable workloads, heterogeneous resources, and competing performance and cost objectives. Reviews of cloud, fog, and edge allocation identify scheduling, placement, elasticity, and quality-of-service management as related decision problems [1], [2], [4], [19]. Static rules are easy to audit but can overprovision during quiet periods and underprovision during bursts. Reactive policies improve flexibility by responding to observed utilization, yet their threshold, cooldown, and scaling-delay choices can still produce oscillation or late action [5]–[8], [20]. Optimization and workload-aware placement methods address related trade-offs in geo-distributed and large-scale data centers [21]–[24].

Learning-based studies use neural networks, deep learning, task scheduling, feature optimization, and hybrid methods to infer workload structure or predict demand [25]–[32]. Related quality of service (QoS), optimization, and workload-allocation studies supply nonpredictive comparison points [33]–[36]. Hyperparameter selection can materially change model behavior, so comparison protocols should document search ranges and validation criteria [37], [38]. Their benefit depends not only on model fit but also on how predictions are translated into feasible resource actions and evaluated under consistent workloads. Kirchoff *et al.* [15] compare forecasting techniques and provisioning impact using public traces, CloudProphet [16] predicts public-cloud performance, and Duc *et al.* [18] study proactive allocation in cloud-edge applications. These studies motivate temporal validation, explicit forecasting errors, and end-to-end controller metrics; the present archive supports the last of these but not a standalone predictor benchmark.

Reinforcement learning and multi-model approaches can learn richer decision policies [9], [14], [39], but they also introduce additional training, stability, and interpretability considerations. For a frequent control loop, random forest offers nonlinear modeling with bounded tree traversal and an auditable separation between prediction and capacity enforcement [15], [40]. The remaining research gap is thus not the absence of machine learning (ML) for cloud allocation. It is the need for a concise controller-level comparison that links forecasting to action and reports utilization, response time, cost, sensitivity, and repeated-run uncertainty together, while clearly separating descriptive simulation evidence from production claims. Accordingly, this study evaluates an incremental, interpretable integration of known components. Its novelty lies in the closed prediction–allocation–monitoring–retraining workflow and the three-policy comparison; it does not claim a novel random forest variant, a globally optimal allocation solver, or state-of-the-art forecasting accuracy.

3. PROPOSED ADAPTIVE RESOURCE-ALLOCATION FRAMEWORK

3.1. Framework architecture

The framework comprises three phases: workload monitoring and preprocessing, predictor fitting and periodic updating, and capacity-constrained allocation [19], [27], [41]. At each five-minute decision interval, current CPU, memory, network, and disk-I/O observations form the feature vector. The predictor estimates demand for the next interval, and the controller converts this forecast into a feasible resource adjustment. The resulting utilization, simulated cost, and response-time measurements are recorded and returned to the subsequent decision cycle.

Monitoring uses the same feature definitions for model training and online prediction, thereby maintaining consistency between the offline and operational stages. The controller bounds every allocation or release according to the current allocation, resource compatibility, and available host capacity, while newly observed measurements support subsequent decisions and periodic retraining. The retained experiment

description documents the removal of incomplete observations and the normalization of four predefined features, but it does not preserve a statistical outlier threshold, noise-removal algorithm, or automated feature-selection procedure; these operations are therefore not claimed. This reporting boundary follows general preprocessing guidance that transformations affecting the retained sample should be explicitly specified and reproducible [42], [43]. The complete prediction-to-action sequence is summarized in Figure 1.

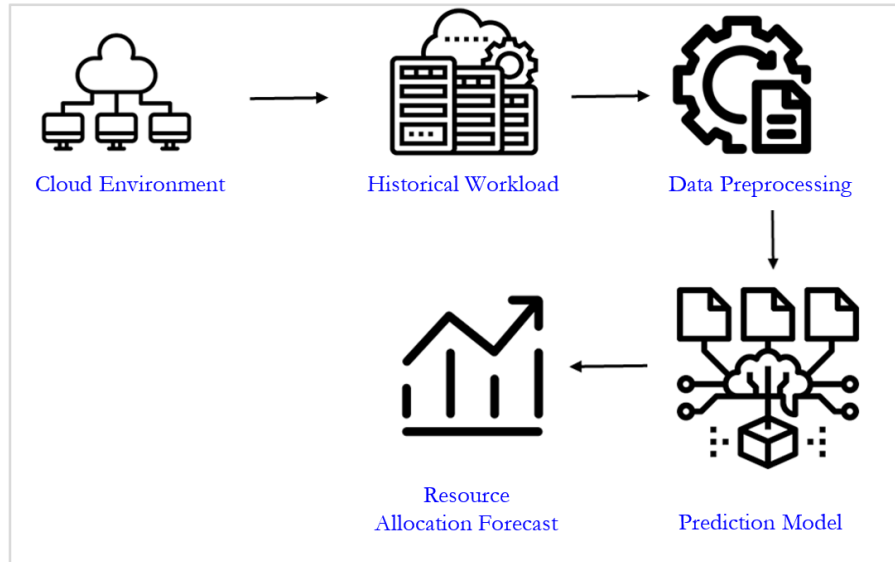


Figure 1. Prediction-to-action architecture of the adaptive resource-allocation framework

The predictor is a random forest regressor. Its relevant hyperparameters are the number and depth of trees, the minimum samples for a split and leaf, and the number of candidate features per split; learning rate, regularization coefficients, and hidden layers do not apply to this model. The retained final setting is reported in subsection 4.3. During operation, the predictor estimates demand for the next five-minute interval. The controller allocates only available capacity, releases predicted surplus, and records utilization, simulated cost, and application response time. Periodic retraining makes recent observations available to later decisions, but does not by itself demonstrate robustness to all forms of production drift [29], [30].

3.2. Model formulation

The implemented method is a constrained predictor-controller, not a weighted-sum optimization solver. Let $\mathbf{x}_i(t)$ be the monitored feature vector for virtual machine (VM) i and $\mathbf{a}_i(t)$ the current allocation vector. The symbol M_t denotes the random forest available at interval t , while Cap_j denotes the total capacity of resource dimension j . The native Word equation below defines the one-step-ahead forecast.

$$\hat{d}_i(t+1) = M_t(\mathbf{x}_i(t))$$

The forecast is converted into an allocation delta and a bounded update:

$$\Delta_i(t) = \hat{d}_i(t+1) - a_i(t)$$

$$a_i(t+1) = a_i(t) + \text{clip}(\Delta_i(t), -a_i(t), C_{i,\text{free}}(t))$$

The allocation remains feasible only when:

$$\sum_i a_{ij}(t+1) \leq \text{Cap}_j, \quad \forall j$$

$$a_{ij}(t+1) = 0 \quad \text{if virtual machine } i \text{ and resource } j \text{ are incompatible.}$$

The clipping operation prevents a release larger than the current allocation and an increase larger than compatible free capacity. Resource utilization, response time, and simulated cost are evaluation outcomes rather than terms combined with undocumented weights. This formulation matches Algorithm 1 by separating demand estimation from explicit feasibility checks. It supports interpretable, capacity-constrained decisions but does not establish global optimality or solve a Pareto-front problem.

Algorithm 1. Adaptive ML resource allocation

Input: workload observations D ; resource pool R ; horizon $H = 1$ interval (5 minutes)

Output: feasible allocation $A(t)$ at each interval t

```

1: Fit Random Forest  $M$  on the initial training data
2: Set adaptation window  $W = 288$  intervals (24 hours)
3: for each five-minute interval  $t$  do
4:  $x(t) =$  monitor CPU, memory, network, and disk I/O
5:  $\text{demand\_hat} = M.\text{predict}(x(t), \text{horizon} = H)$ 
6:  $\text{delta} = \text{demand\_hat} - \text{current\_allocation}(R)$ 
7: if  $\text{delta} > 0$  then
8: allocate  $\min(\text{delta}, \text{available\_capacity}(R))$ 
9: else if  $\text{delta} < 0$  then release  $\text{abs}(\text{delta})$ 
10: record  $A(t)$ , utilization, cost, and response time
11: if  $t \bmod W = 0$  then retrain  $M$  on latest  $W$  records
12: end for
13: return  $A(1), \dots, A(T)$ 

```

4. MATERIALS AND METHODS

The evaluation uses a controlled CloudSim experiment. Its workflow comprises data preparation, random forest prediction, adaptive allocation, and comparison with fixed and reactive baselines. The following subsections distinguish retained settings from information that was not preserved. This separation makes the reproducibility boundary explicit and prevents undocumented details from being presented as experimental facts.

4.1. Simulation environment

CloudSim modeled a 30-day experiment with 50 hosts and 200 VM [44], [45]. Observations were reported at five-minute intervals, and all three allocation policies were evaluated within the same simulator. This common environment supports an internally controlled comparison, although it does not reproduce multi-tenant interference, startup delay, or provider-specific control planes. The archived manuscript does not retain the CloudSim or Java version, per-host processing, memory, storage, and bandwidth capacities, VM specifications, provisioner and scheduler classes, workload-generation distributions, pricing rate card, response-time instrumentation, or exact seed list. Consequently, the study can be assessed as a comparative simulation but cannot yet be reproduced bit-for-bit; no vendor-specific or production-cloud equivalence is claimed.

4.2. Data preprocessing

Preprocessing removed incomplete observations and normalized the four recorded resource signals. The archive does not document a separate noise-removal algorithm or an outlier rule, so neither operation is asserted. The simulation-generated data contain CPU utilization (%), memory use (MB), network bandwidth (Mbps), and disk-I/O (KB/s) from the 50-host, 200-VM environment. Approximately 17,000 feature records are reported over 30 days. Because the retained description does not explain how simultaneous host or VM measurements were aggregated into those records, the observation unit remains a reproducibility limitation.

The four inputs were predefined rather than selected by a documented automated procedure. This manuscript therefore treats the step as feature definition, not experimentally validated feature selection [30]. Normalization reduces scale differences across the four measurement units [43]. The retained record does not specify the scaler, its parameters, or whether it was fitted only on the initial training portion; this information should be archived in a reproducibility package before a confirmatory study.

4.3. Machine learning model training

The numerical prediction target is next-interval resource demand derived from recent CPU, memory, network, and disk-I/O observations. Random forest regression was fitted to this task and then embedded in the allocation loop. The forecast horizon is one five-minute decision interval. Because the target is continuous,

suitable standalone predictor measures are mean absolute error (MAE), root mean squared error (RMSE), and R-squared rather than classification accuracy, precision, recall, or F1-score [15], [46].

Grid search produced the retained configuration of 100 trees, maximum depth 10, minimum split size 5, minimum leaf size 2, and max_features = 'sqrt'. These choices correspond to standard Random Forest hyperparameters [38]. The candidate ranges, search criterion, validation folds, initial training-window size, elapsed search time, and software versions were not retained. Accordingly, the final configuration is reproducible, but the model-selection process is not. Random forest was chosen for nonlinear interactions, robustness to heterogeneous signals, bounded tree traversal, and interpretability in a frequent control loop [15], [40]. CloudProphet and recent provisioning studies illustrate the value of explicit accuracy and latency comparisons [15], [16]; because no such standalone benchmark was archived here, this study does not claim that random forest is the most accurate or fastest available predictor.

4.4. Adaptive resource allocation

At each five-minute interval, the controller forms $x_i(t)$, forecasts demand one interval ahead ($H=1$), and computes the allocation delta in subsection 3.2. This temporal resolution is the evaluated decision cadence, not a claim of instantaneous production response. Positive deltas are bounded by compatible available capacity, negative deltas are bounded by the current allocation, and zero deltas leave the allocation unchanged. This implements proactive capacity adjustment without implying that the rule globally optimizes all VM [47].

Every 288 intervals, or 24 hours, the predictor is retrained using the most recent observations, as specified in Algorithm 1. This interval corresponds to one full daily cycle at the five-minute sampling cadence. The daily window is a design choice rather than the result of a reported ablation study. Alternative retraining frequencies therefore remain unevaluated.

4.5. Experimental evaluation

Twenty AML runs were compared descriptively with fixed and reactive dynamic allocation in the same CloudSim environment. Each AML run used a different random seed, and all runs were retained. The fixed and dynamic policies are represented by their reported aggregate summaries. Their run-level outputs were not retained, which precludes paired hypothesis tests and standardized effect sizes.

The primary controller outcomes are aggregate resource utilization, total simulated cost, and application response time. The sensitivity table additionally reports the study's demand-satisfaction score and a relative cost index normalized to 1000 for the low-workload/high-availability reference. Numeric boundaries for the low, medium, and high scenario labels, the exact aggregation formula for utilization and demand satisfaction, the pricing rate card, and the response-time calculation were not archived. Table 1 is therefore interpreted directionally rather than converted into service-level attainment or provider currency.

Table 1. Sensitivity analysis using demand-satisfaction score and relative cost index (low workload/high availability=1000)

Workload	Resource availability	Demand-satisfaction score	Relative cost index
Low	High	0.90	1000
Medium	High	0.85	1050
High	High	0.80	1100
Low	Medium	0.88	950
Medium	Medium	0.82	1000
High	Medium	0.75	1050
Low	Low	0.85	900
Medium	Low	0.80	950
High	Low	0.70	1000

4.6. Baseline resource allocation approaches

The proposed controller was compared with fixed and reactive dynamic allocation. Fixed allocation represents a static policy that does not respond to observed workload changes. Dynamic allocation represents a reactive policy that responds to the current state without forecasting. The following subsections define these baselines and identify the parameters that were not retained.

4.6.1. Fixed allocation

Fixed allocation holds a predetermined capacity vector throughout a run [6]–[8]. It therefore cannot react to workload changes after the run begins. This behavior can produce either unused capacity or a shortfall when demand diverges from the predetermined level. The exact capacity vector used for the reported baseline was not retained, so the policy is reproducible conceptually but not parametrically.

4.6.2. Dynamic allocation

Reactive dynamic allocation changes capacity from the currently observed workload without using a forecast [5]–[8]. It can therefore respond after the monitored state changes, but it cannot act on an unobserved future interval. Its behavior depends materially on thresholds, step sizes, cooldown periods, and release delays. These parameters were not retained and should be reported in a confirmatory comparison to establish baseline strength.

4.7. Predictor selection and evaluation metrics

Predictor selection emphasized nonlinear modeling, bounded inference cost, noise tolerance, and interpretability [15], [16], [40]. These criteria justify random forest as a practical candidate, but not as a state-of-the-art winner without a common temporal benchmark. A standalone forecasting evaluation with MAE, RMSE, R^2 , training time, and inference latency was not preserved. Linear regression, decision trees, support-vector regression, boosting, and neural models are consequently discussed as future comparators rather than presented as experimentally defeated baselines.

The available empirical evidence concerns the end-to-end controller and is evaluated using utilization, simulated cost, response time, sensitivity, and repeated-run variability. This keeps the reported evidence aligned with the resource-allocation decision while explicitly separating it from missing forecasting evidence. The controller itself operates sequentially: later decisions can use only observations already generated by the simulation. However, the archive does not preserve the initial train/test boundary or the fitting scope of preprocessing parameters. A strict leakage-free chronological holdout should therefore be required in the next evaluation [48].

The configured model is first fitted on the available initial history and is then updated on the latest 288 observations every 24 hours. Because the size of the initial history was not retained, cold-start performance and the first valid forecast cannot be reconstructed exactly. Across 20 AML runs, the mean, sample standard deviation, and two-sided 95% t confidence interval are reported in Table 2. All runs were retained. These intervals quantify AML run-to-run variation only; the absence of baseline run distributions prevents inferential superiority claims.

Table 2. Run-to-run stability of AML across 20 independent simulation runs

Metric	Mean	Sample SD	95% t confidence interval
Resource utilization	85.2%	1.8%	84.4-86.0%
Total cost	US\$9,045	US\$120	US\$8,989-US\$9,101
High-load response time	54.8 ms	3.2 ms	53.3-56.3 ms

4.8. Algorithm pseudocode

Algorithm 1 states the monitoring, prediction, constrained-adjustment, and daily-retraining cycle used by the adaptive controller. Its inputs are the workload observations, resource pool, and one-interval forecast horizon. At each interval, it returns a feasible allocation after prediction and available-capacity checks. It also records the controller outcomes and triggers model retraining after 288 five-minute intervals.

With B trees and maximum depth D, one prediction requires $O(BD)$ tree-node decisions; the retained setting gives an upper bound of approximately 1000 decisions before allocation checks. Retraining on n records is approximately $O(Bn \log n)$, and feasibility checks scale with the number of candidate resources. These bounds motivate the use of a shallow forest in the controller. Measured training time, inference latency, controller overhead, and memory use were not archived, so the theoretical bounds are not presented as runtime benchmarks.

5. RESULTS AND ANALYSIS

Figures 2 to 4 present the descriptive three-policy comparison, Table 1 follows the figures with the reported sensitivity scenarios, and Table 2 follows Table 1 with uncertainty across 20 AML runs. Baseline distributions are unavailable, so between-policy results are reported as differences rather than significance tests. Figure 2 shows the reported aggregate utilization: approximately 85% for predictive ML (AML), 75% for reactive dynamic allocation, and 60% for fixed allocation. The AML value is 10 percentage points above the dynamic baseline and 25 percentage points above the fixed baseline. Because the aggregation rule and baseline run distributions were not retained, this is descriptive evidence within the configured simulation rather than a population-level estimate.

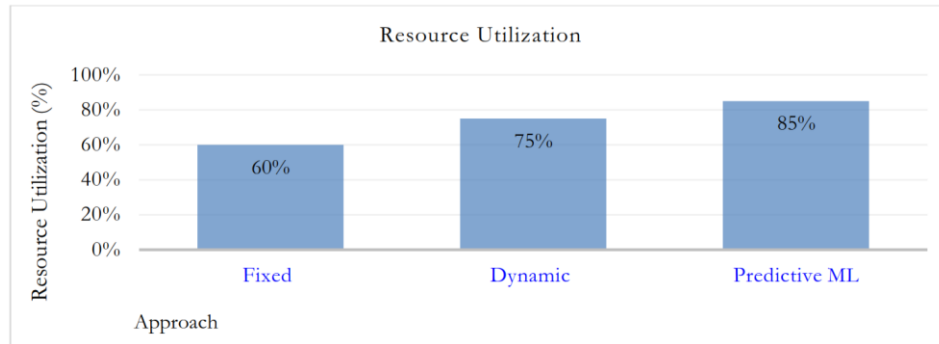


Figure 2. Resource utilization is compared across the three allocation strategies

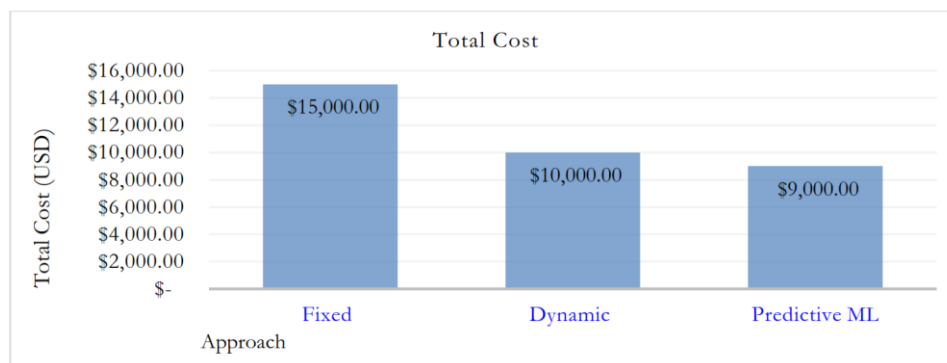


Figure 3. Total simulated cost is compared across the three allocation strategies

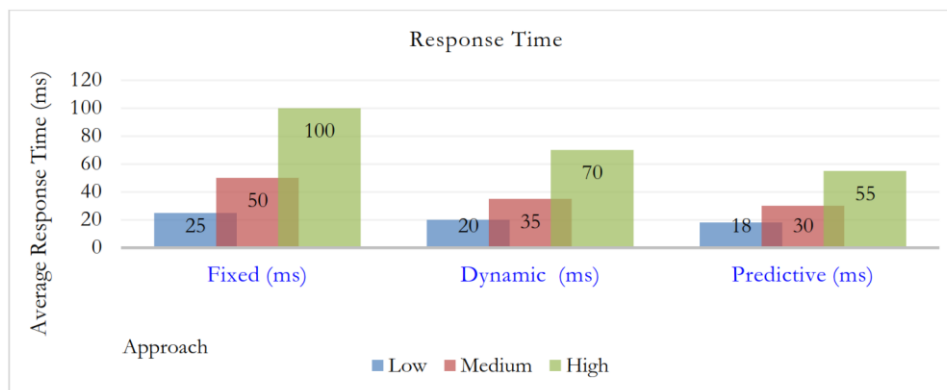


Figure 4. Application response time is compared across allocation strategies under low, medium, and high workloads

Figure 3 reports rounded total simulated cost of approximately US\$9,000 for predictive ML (AML), US\$10,000 for dynamic allocation, and US\$15,000 for fixed allocation. The AML summary is 10% below the dynamic value and 40% below the fixed value. Across the 20 AML runs summarized in Table 2, mean total cost was US\$9,045 (sample SD US\$120). This pattern shows that the reported utilization advantage was not accompanied by a higher summarized cost. However, the underlying pricing rate card was not archived, so the currency-labelled simulator outputs should not be interpreted as transferable provider bills.

Figure 4 reports lower AML application response time in each workload level. For low, medium, and high workloads, respectively, the summarized response times are 25, 50, and 100 ms for fixed allocation; 20, 35, and 70 ms for dynamic allocation; and 18, 30, and 55 ms for predictive ML (AML). Under high workload, the AML value is descriptively 21.4% lower than dynamic allocation and 45.0% lower than fixed allocation.

The response-time instrumentation was not retained, so the values cannot be equated with controller inference latency.

Table 1 reports sensitivity across combinations of workload and available capacity. The demand-satisfaction score decreases as workload increases and availability becomes more constrained; under high workload it falls from 0.80 at high availability to 0.70 at low availability. The relative cost index spans 900–1100 around the stated low-workload/high-availability reference of 1000. Because the scenario thresholds and score formula were not archived, the table demonstrates a directional performance–cost trade-off but does not establish exact service-level attainment.

Table 2 summarizes AML run-to-run stability across 20 simulations. Mean utilization is 85.2% (sample SD 1.8%), mean total cost is US\$9,045 (sample SD US\$120), and mean high-load response time is 54.8 ms (sample SD 3.2 ms). The reported 95% t intervals are 84.4–86.0%, US\$8,989–US\$9,101, and 53.3–56.3 ms, respectively. They are internally consistent with $n=20$ and describe AML variability, not statistical superiority over baselines.

Taken together, the summaries indicate that prediction-guided action used a larger share of provisioned resources. The same policy also produced lower reported response times and did not increase reported cost relative to the two baselines. The repeated AML runs support stability inside this simulator. External validity, exact replication, and causal attribution to forecasting nevertheless require stronger configuration records and common run-level baselines.

6. DISCUSSION

The main finding is a consistent descriptive advantage for AML in the configured CloudSim scenarios. Across 20 AML runs, utilization averaged 85.2% $\pm$ 1.8%, cost US\$9,045 $\pm$ US\$120, and high-load response time 54.8 $\pm$ 3.2 ms. The confidence intervals characterize AML stability only and do not replace a paired comparison with fixed and dynamic policies. A plausible mechanism is timing: AML uses predicted next-interval demand before adjustment, whereas the dynamic baseline waits for the observed state. This timing can reduce a one-interval lag during rising demand and release capacity when a decline is forecast. The mechanism is consistent with proactive provisioning literature [15], [18], [47], but it is not isolated experimentally because predictor-only and retraining-frequency ablations were not retained.

The five-minute monitoring cadence and 24-hour retraining window close the loop between prediction, allocation, observation, and updating. This is the study's decision-making contribution: ML produces an auditable capacity action rather than only a forecast. The selected cadence remains a design choice, and shorter bursts or faster drift may not be captured. Relative to recent studies, the framework is intentionally modest. Kirchoff *et al.* [15] and CloudProphet [16] provide stronger predictor-level evaluations, while multi-model and cloud-edge work considers richer forecasting contexts [9], [18]. The present contribution is the transparent three-policy, system-level integration; it should not be positioned as a new random forest method or as outperforming those state-of-the-art approaches.

Internal validity is limited by missing baseline parameters and run distributions. Construct validity is limited by undocumented aggregation, pricing, and response-time formulas. Reproducibility is limited by missing software versions, infrastructure and workload specifications, preprocessing boundaries, grid ranges, initial training size, and seeds. External validity is limited to CloudSim, without production interference, startup delay, telemetry overhead, failures, or provider pricing.

6.1. Validity and deployment priorities

Before submission of a confirmatory study, the authors should archive the CloudSim project, versioned software environment, host and VM configuration, scheduler and provisioner classes, workload generator, price model, metric formulas, all seeds, and per-run outputs for every policy. A chronological holdout should report MAE, RMSE, R^2 , training time, and inference latency for random forest and at least one recent learning-based comparator. A real-cloud or cluster testbed should then measure controller overhead, VM startup and scaling delay, service-level violations, telemetry cost, and behavior under abrupt bursts and failures. A fallback policy is needed for cold start and prediction failure; hierarchical control may improve scale, but its communication and consistency costs require measurement. These additions would permit paired tests and effect sizes under identical seeds, separate the value of forecasting from periodic retraining, and reveal failure cases in which forecast error or capacity scarcity makes reactive control preferable. Production superiority is not claimed because full infrastructure settings, run-level baseline distributions, standalone forecasting errors, and real-cloud validation were unavailable. Until these gaps are addressed, the current evidence supports feasibility and stable AML output within the configured simulation, not production scalability, fault tolerance, or universal cost savings.

7. CONCLUSION

This study compared fixed, reactive dynamic, and prediction-guided allocation in CloudSim using a random forest controller, five-minute decisions, 24-hour retraining, and 20 AML runs. AML produced the highest reported aggregate utilization (approximately 85% versus 75% and 60%), the lowest summarized high-load response time (55 ms versus 70 ms and 100 ms), and lower simulated cost than fixed allocation with a result close to the dynamic baseline. Its intelligent decision-making contribution is the explicit conversion of near-term workload forecasts into auditable, capacity-constrained allocation actions. The evidence is descriptive and simulation-specific. Missing infrastructure and workload settings, preprocessing and search boundaries, pricing and metric formulas, initial training size, baseline run distributions, and standalone forecasting errors limit reproducibility and prevent inferential superiority claims. Future work should publish a complete experiment package, use leakage-free chronological evaluation, compare recent predictors and controllers under identical seeds, report paired statistics and computational overhead, test alternative retraining windows, and validate the controller on a real cloud or cluster under bursts, failures, and scaling delays.

FUNDING INFORMATION

This work was supported by the EQUITY Kemdiktisaintek Program, funded by the Indonesia Endowment Fund for Education (LPDP). The research grant or contract numbers are 4310/B3/DT.03.08/2025 and 2692/UN35/KS/2025. Universitas Negeri Padang also provided financial support for the article processing charge.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Agariadne Dwinggo Samala	✓	✓	✓		✓	✓		✓	✓		✓		✓	
Soha Rawas	✓	✓		✓					✓	✓		✓		

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal analysis

I : **I**nterpretation

R : **R**esources

D : **D**ata Curation

O : **O**riginal Draft

E : **E**diting

Vi : **V**isualization

Su : **S**upervision

P : **P**roject administration

Fu : **F**unding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that there is no conflict of interest.

DATA AVAILABILITY

The workload observations and run-level outputs that support the findings are available from the corresponding author upon reasonable request.

REFERENCES

- [1] V. Patsias, P. Amanatidis, D. Karampatzakis, T. Lagkas, K. Michalakopoulou, and A. Nikitas, "Task allocation methods and optimization techniques in edge computing: A systematic review of the literature," *Future Internet*, vol. 15, no. 8, pp. 1-30, 2023, doi: 10.3390/fi15080254.
- [2] D. Alsadie, "A comprehensive review of AI techniques for resource management in fog computing: Trends, challenges, and future directions," *IEEE Access*, vol. 12, pp. 118007-118059, 2024, doi: 10.1109/ACCESS.2024.3447097.
- [3] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Netw. Syst. Manag.*, vol. 23, no. 3, pp. 567-619, 2015, doi: 10.1007/s10922-014-9307-7.
- [4] P. Nawrocki and M. Smendowski, "A survey of cloud resource consumption optimization methods," *J. Grid Comput.*, vol. 23, no. 1, 2025, doi: 10.1007/s10723-024-09792-0.
- [5] S. S. Gill, I. Chana, M. Singh, and R. Buyya, "CHOPPER: An intelligent QoS-aware autonomic resource management approach for cloud computing," *Cluster Comput.*, vol. 21, pp. 1203-1241, 2018, doi: 10.1007/s10586-017-1040-z.
- [6] T. Lorigo-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *J. Grid Comput.*, vol. 12, no. 4, pp. 559-592, 2014, doi: 10.1007/s10723-014-9314-7.

- [7] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107-1117, 2013, doi: 10.1109/TPDS.2012.283.
- [8] C. Qu, R. N. Calheiros, and R. Buyya, "Auto-scaling web applications in clouds: A taxonomy and survey," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1-33, 2018, doi: 10.1145/3148149.
- [9] D. Wladdimiro *et al.*, "A multi-model predictive framework for adaptive resource management in stream processing systems," *Future Gener. Comput. Syst.*, vol. 182, pp. 1-15, 2026, doi: 10.1016/j.future.2026.108461.
- [10] M. Smendowski and P. Nawrocki, "Optimizing multi-time series forecasting for enhanced cloud resource utilization based on machine learning," *Knowl.-Based Syst.*, vol. 304, pp. 1-21, 2024, doi: 10.1016/j.knosys.2024.112489.
- [11] F. N. Manhary, M. H. Mohamed, and M. Farouk, "A scalable machine learning strategy for resource allocation in database," *Sci. Rep.*, vol. 15, pp. 1-17, 2025, doi: 10.1038/s41598-025-14962-5.
- [12] S. H. Anbarkhan, "Optimizing cloud resource allocation with machine learning: Strategies for efficient computing," *Ing. Syst. Inf.*, vol. 30, no. 1, pp. 1-9, 2025, doi: 10.18280/isi.300101.
- [13] L. Li and X. Gao, "Profit-efficient elastic allocation of cloud resources using two-stage adaptive workload prediction," *Appl. Sci.*, vol. 15, no. 5, pp. 1-28, 2025, doi: 10.3390/app15052347.
- [14] M. N. Alatawi, "Optimizing multitenancy: Adaptive resource allocation in serverless cloud environments using reinforcement learning," *Electronics*, vol. 14, no. 15, pp. 1-25, 2025, doi: 10.3390/electronics14153004.
- [15] D. F. Kirchoff, V. Meyer, R. N. Calheiros, and C. A. F. De Rose, "Evaluating machine learning prediction techniques and their impact on proactive resource provisioning for cloud environments," *J. Supercomput.*, vol. 80, no. 15, pp. 21920-21951, 2024, doi: 10.1007/s11227-024-06303-6.
- [16] D. Huang, L. Costero, A. Pahlevan, M. Zapater, and D. Atienza, "CloudProphet: A machine learning-based performance prediction for public clouds," *IEEE Trans. Sustain. Comput.*, vol. 9, no. 4, pp. 661-676, 2024, doi: 10.1109/TSUSC.2024.3359325.
- [17] B. Barua and M. S. Kaiser, "AI-driven resource allocation framework for microservices in hybrid cloud platforms," *arXiv*, 2024, doi: 10.48550/arXiv.2412.02610.
- [18] T. Le Duc, C. Nguyen, and P. O. Östberg, "Workload prediction for proactive resource allocation in large-scale cloud-edge applications," *Electronics*, vol. 14, no. 16, pp. 1-36, 2025, doi: 10.3390/electronics14163333.
- [19] M. V. Fard, A. Sahafi, A. M. Rahmani, and P. S. Mashhadi, "Resource allocation mechanisms in cloud computing: A systematic literature review," *IET Softw.*, vol. 14, no. 6, pp. 638-653, 2020, doi: 10.1049/iet-sen.2019.0338.
- [20] A. Mampage, S. Karunasekera, and R. Buyya, "A holistic view on resource management in serverless computing environments: Taxonomy and future directions," *ACM Comput. Surv.*, vol. 54, no. 11s, pp. 1-36, 2022, doi: 10.1145/3510412.
- [21] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew, "Greening geographical load balancing," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 657-671, 2015, doi: 10.1109/TNET.2014.2308295.
- [22] S. R. Swain, A. Parashar, A. K. Singh, and C. N. Lee, "An intelligent virtual machine allocation optimization model for energy-efficient and reliable cloud environment," *J. Supercomput.*, vol. 81, no. 1, 2025, doi: 10.1007/s11227-024-06734-1.
- [23] S. Rawas, "Energy, network, and application-aware virtual machine placement model in SDN-enabled large-scale cloud data centers," *Multimedia Tools Appl.*, vol. 80, no. 10, pp. 15541-15562, 2021, doi: 10.1007/s11042-021-10616-6.
- [24] S. Abedi, M. Ghozaei-Arani, E. Khorami, and M. Mojarad, "Dynamic resource allocation using improved Firefly Optimization Algorithm in cloud environment," *Appl. Artif. Intell.*, vol. 36, no. 1, 2022, doi: 10.1080/08839514.2022.2055394.
- [25] D. Minarolli and B. Freisleben, "Distributed resource allocation to virtual machines via artificial neural networks," in *Proc. 22nd Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process. (PDP)*, 2014, pp. 490-499, doi: 10.1109/PDP.2014.102.
- [26] P. K. Bal, S. K. Mohapatra, T. K. Das, K. Srinivasan, and Y.-C. Hu, "A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques," *Sensors*, vol. 22, no. 3, pp. 1-16, 2022, doi: 10.3390/s22031242.
- [27] D. Saxena and A. K. Singh, "Workload forecasting and resource management models based on machine learning for cloud computing environments," *arXiv*, 2021, doi: 10.48550/arXiv.2106.15112.
- [28] S. R. Katragadda, A. Tanikonda, S. R. Peddinti, and B. K. Pandey, "Predictive machine learning models for effective resource utilization forecasting in hybrid IT systems," *J. Sci. Technol.*, vol. 3, no. 6, pp. 92-112, 2022.
- [29] T. N. Gongada, G. B. Desale, S. P. Ghodake, K. Sridharan, V. S. Rao, and Y. A. B. El-Ebiary, "Optimizing resource allocation in cloud environments using Fruit Fly Optimization and convolutional neural networks," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 5, pp. 1173-1182, 2024, doi: 10.14569/IJACSA.2024.01505119.
- [30] G. Zhang, X. He, and X. Wang, "PCL-RC: A parallel cloud resource load prediction model based on feature optimization," *J. Cloud Comput.*, vol. 14, pp. 1-17, 2025, doi: 10.1186/s13677-025-00770-9.
- [31] Y. Sanjalawe, S. Fraihat, S. Al-E'mari, M. Abualhaj, S. Makhadmeh, and E. Alzubi, "Smart load balancing in cloud computing: Integrating feature selection with advanced deep learning models," *PLOS ONE*, vol. 20, no. 9, pp. 1-50, 2025, doi: 10.1371/journal.pone.0329765.
- [32] F. S. Alsubaei, A. Y. Hamed, M. R. Hassan, M. Mohery, and M. K. Elnahary, "Machine learning approach to optimal task scheduling in cloud communication," *Alexandria Eng. J.*, vol. 89, pp. 1-30, 2024, doi: 10.1016/j.aej.2024.01.040.
- [33] H. M. Khan, F. F. Chua, and T. T. V. Yap, "ReSQoV: A scalable resource allocation model for QoS-satisfied cloud services," *Future Internet*, vol. 14, no. 5, pp. 1-20, 2022, doi: 10.3390/fi14050131.
- [34] H. Yu, "Evaluation of cloud computing resource scheduling based on improved optimization algorithm," *Complex Intell. Syst.*, vol. 7, no. 4, pp. 1817-1822, 2021, doi: 10.1007/s40747-020-00163-2.
- [35] M. Alicherry and T. V. Lakshman, "Network aware resource allocation in distributed clouds," in *Proc. IEEE INFOCOM*, 2012, pp. 963-971, doi: 10.1109/INFOCOM.2012.6195847.
- [36] M. S. Qureshi, M. B. Qureshi, M. Fayaz, M. Zakarya, S. Aslam, and A. Shah, "Time- and cost-efficient cloud resource allocation for real-time data-intensive smart systems," *Energies*, vol. 13, no. 21, pp. 1-25, 2020, doi: 10.3390/en13215706.
- [37] M. A. K. Raiaan *et al.*, "A systematic review of hyperparameter optimization techniques in convolutional neural networks," *Decis. Anal. J.*, vol. 11, pp. 1-32, 2024, doi: 10.1016/j.dajour.2024.100470.
- [38] P. Probst, M. N. Wright, and A.-L. Boulesteix, "Hyperparameters and tuning strategies for random forest," *WIREs Data Min. Knowl. Discov.*, vol. 9, no. 3, 2019, doi: 10.1002/widm.1301.
- [39] H. Li, L. Lu, W. Shi, G. Tan, and H. Luo, "Energy-aware scheduling for Spark job based on deep reinforcement learning in cloud," *Computing*, vol. 105, no. 8, pp. 1717-1743, 2023, doi: 10.1007/s00607-023-01171-z.
- [40] D. Bodra and S. Khairnar, "Machine learning-based cloud resource allocation algorithms: A comprehensive comparative review," *Front. Comput. Sci.*, vol. 7, pp. 1-17, 2025, doi: 10.3389/fcomp.2025.1678976.

- [41] G. Peng, H. Wang, J. Dong, and H. Zhang, "Knowledge-based resource allocation for collaborative simulation development in a multi-tenant cloud computing environment," *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 306-317, 2018, doi: 10.1109/TSC.2016.2518161.
- [42] M. Hussain, S. Mateen, and A. Haque, "Data collection and pre-processing," in *Artificial Intelligence for Power Electronics*, Wiley, 2025, pp. 105-135, doi: 10.1002/9781394270804.ch4.
- [43] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera, "Big data preprocessing: Methods and prospects," *Big Data Anal.*, vol. 1, 2016, doi: 10.1186/s41044-016-0014-0.
- [44] R. Andreoli, J. Zhao, T. Cucinotta, and R. Buyya, "CloudSim 7G: An integrated toolkit for modeling and simulation of future-generation cloud computing environments," *Softw. Pract. Exp.*, vol. 55, no. 6, pp. 1041-1058, 2025, doi: 10.1002/spe.3413.
- [45] 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," *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23-50, 2011, doi: 10.1002/spe.995.
- [46] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)? Arguments against avoiding RMSE in the literature," *Geosci. Model Dev.*, vol. 7, no. 3, pp. 1247-1250, 2014, doi: 10.5194/gmd-7-1247-2014.
- [47] S. Choularas and S. Sotiriadis, "An adaptive auto-scaling framework for cloud resource provisioning," *Future Gener. Comput. Syst.*, vol. 148, pp. 173-183, 2023, doi: 10.1016/j.future.2023.05.017.
- [48] C. Bergmeir, R. J. Hyndman, and B. Koo, "A note on the validity of cross-validation for evaluating autoregressive time series prediction," *Comput. Stat. Data Anal.*, vol. 120, pp. 70-83, 2018, doi: 10.1016/j.csda.2017.11.003.

BIOGRAPHIES OF AUTHORS



Agariadne Dwinggo Samala    is a professional educator, futurist, and dedicated researcher, currently serving as an Associate Professor at the Faculty of Engineering, Universitas Negeri Padang (UNP), Indonesia, and an affiliated Tutor at Universitas Terbuka, Indonesia. With more than five years of experience, he actively contributes to the academic community as a member of several professional associations, including APTIKOM (Association of Higher Education in Informatics and Computers), the Digital Society Lab at the Institute for Philosophy and Social Theory, the International Association of Engineers (IAENG), and Ikatan Ilmuwan Indonesia Internasional (I4). His research interests include technology-enhanced learning (TEL), educational technology, immersive learning applications and experiences, digital learning, microlearning, and emerging technologies in educational contexts. He can be contacted at email: agariadne@ft.unp.ac.id.



Dr. Soha Rawas    holds a Ph.D. in Mathematics and Computer Science from Beirut Arab University (2019). Her research spans artificial intelligence, deep learning, IoMT, cloud computing, medical image processing, secure intelligent systems, and AI in education, with applications across healthcare, learning technologies, and smart systems. She is a faculty member in the Department of Mathematics and Computer Science, Faculty of Science at BAU, and serves as the Director of the Continuing Education Center (CEC), leading strategic development, digital transformation, and innovative training programs across diverse professional fields. She can be contacted at email: soha.rawas2@bau.edu.lb.