RESEARCH ARTICLE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
www.rspsciencehub.com
Vol. 08, Issue 10 October

Check for updates

http://dx.doi.org/10.47392/IRJASH.2026.046

Cloud X - AI Powered Multi-Cloud Cost and Resource Optimization Platform

Tamizhmani V[1], Harsath Mohamed S[2], Janani V[3], Gokulapriya[4]
[1,2,3]Department of computer science and Engineering, Erode Sengunthar Engineering College, Erode, Tamil Nadu, India
[4]Assistant Professor, Department of computer science and Engineering, Erode Sengunthar Engineering College, Erode, Tamil Nadu, India
Emails: *mr.tamizh77@gmail.com[1], harsathcse@gmail.com[2], jananijanani9810@gmail.com[3], gokulapriya.ramasamy@esec.ac.in[4]*

Article history
Received: 07 August 2026
Accepted: 04 September 2026
Published: 09 October 2026

Keywords:
Multi-Cloud Computing, Cost Optimization, Finops, Predictive Analytics, Demand Uncertainty, Resource Allocation, Genetic Algorithm, Workflow Scheduling, Kubernetes, Container Optimization.

Abstract

Enterprises increasingly deploy distributed applications across multiple Infrastructure-as-a-Service (IaaS) providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) to mitigate vendor lock-in, satisfy regional data residency mandates, and exploit heterogeneous compute hardware. However, operationalizing multi-cloud systems introduces severe economic and performance challenges, including geographic service-price disparity, complex inter-zone transfer billing, runtime demand volatility, rigid workflow deadlines, and container-level resource fragmentation. While foundational frameworks have resolved subsets of these challenges such as tenant-guaranteed placement in CADER, stochastic capacity reservation in DCRA, evolutionary deadline/budget workflow scheduling, and multi-objective container allocation in MOMA they remain isolated theoretical or single-domain implementations. This paper introduces the AI-Powered Multi-Cloud Cost and Resource Optimization Platform, a unified operational platform that integrates these complementary techniques into an end-to-end management pipeline. The proposed platform combines: (1) CADER-inspired cross-zone deployment cost minimization with provable approximation guarantees, (2) predictive demand forecasting using machine learning, (3) a two-phase stochastic allocation model separating long-term instance reservation from dynamic on-demand scaling under demand uncertainty, (4) a multi-objective Genetic Algorithm coupled with Monte Carlo simulation for deadline and budget-constrained Directed Acyclic Graph (DAG) workflow scheduling, and (5) an elitist evolutionary microservice placement layer that jointly optimizes physical node balance and reliability weighted network transmission overhead. The platform is designed around a modern full-stack architecture combining a React dashboard, Spring Boot 3 microservices orchestrator, Python AI prediction services, and Kubernetes multi-cluster management. We provide a comprehensive problem formulation, detailed algorithmic pseudocode, and an empirical evaluation plan. Rather than fabricating unmeasured

experimental results, we clearly delineate prior literature benchmarks, system design objectives, and an empirical validation strategy for real-world enterprise FinOps deployment.

1. Introduction

1.1. Background

Cloud computing has fundamentally altered enterprise infrastructure procurement, shifting computing from high capital expenditure (CapEx) to elastic operating expenditure (OpEx). As enterprise workloads mature, relying on a single cloud service provider introduces catastrophic points of failure, commercial lock-in, and geographical latency limitations. Consequently, modern enterprises overwhelmingly adopt multi-cloud paradigms, distributing system components across heterogeneous public providers including Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). Recent industry surveys indicate that over 86% to 96% of enterprise organizations deploy or intend to deploy workloads across two or more cloud service providers.

1.2. Multi-Cloud Resource Management Problem

Despite its operational benefits, multi-cloud deployment exacerbates resource management complexity across three primary dimensions:

1) Pricing and Transfer Heterogeneity: Virtual machine (VM) compute pricing varies markedly across providers, instance types, and geographical regions (often by 1.6x to 9.5x for specialized accelerators such as GPUs and TPUs). Moreover, cloud providers charge asymmetric, distance-dependent fees for egress network traffic across availability zones, regions, and continental boundaries.

2) Workload Dynamic Uncertainty: Cloud customer request rates are intrinsically volatile. Purely static reservations lead to chronic under-provisioning during peak traffic or severe over-provisioning during valleys.

3) Execution Dependency & Infrastructure Hierarchy: Modern applications are rarely monolithic; they are structured either as multi-task scientific Directed Acyclic Graphs (DAGs) constrained by strict budgets and deadlines or as microservices running within heterogeneous, containerized Kubernetes environments.

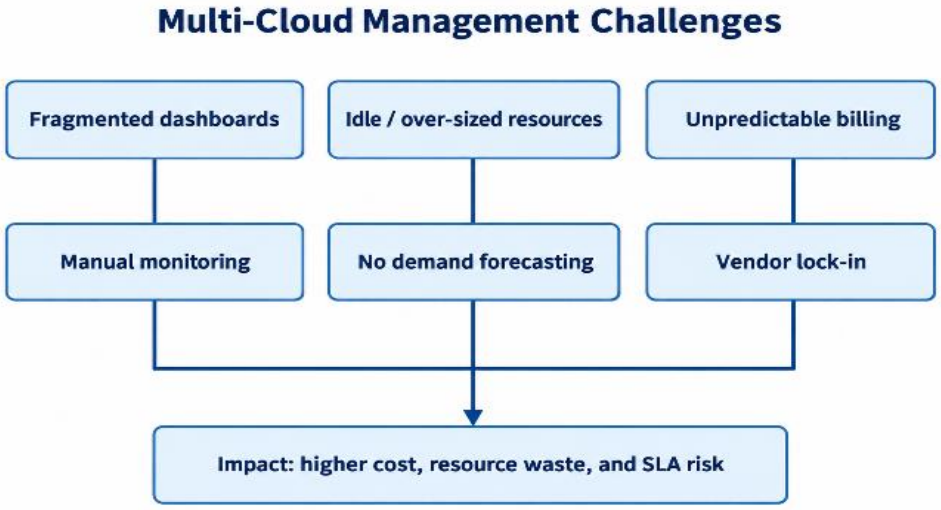


Figure 1 Major Challenges in Traditional Multi-Cloud Resource Management

1.3. Motivation

To manage variable expenditures without sacrificing performance, organizations have turned to FinOps a cultural and operational framework establishing continuous financial accountability in

cloud engineering. However, manual FinOps practices and heuristic rule-based autoscalers cannot efficiently process multi-dimensional, combinatorial decision spaces that span cross-cloud egress costs, stochastic request arrivals, workflow dependencies, and container bin-packing. Bridging this operational gap requires an intelligent, automated platform capable of unifying predictive analytics with multi-stage mathematical and meta-heuristic optimization.

1.4. Research Gap

The optimization literature contains advanced, mathematically rigorous algorithms targeting isolated segments of cloud management. The CADER framework provides a cost-efficient deployment formulation balancing service price heterogeneity, data transfer costs, access latency, and inter-task traffic demand [1]. However, CADER assumes a static snapshot of demand and does not manage dynamic runtime instance scaling or reserved-versus-on-demand procurement. The DCRA (Dynamic Cloud Resource Allocation) model partitions cloud resource acquisition into long-term reservation and dynamic on-demand scaling under demand uncertainty [2], but it focuses on single-cloud web server tiers and neglects inter-task workflow DAG structures and cross-cloud egress tariffs. Probabilistic Workflow Schedulers evaluate deadline- and budget-constrained scientific workflows under infrastructure execution variance using Monte Carlo simulations and evolutionary search [3], yet they remain offline batch models disconnected from containerized orchestration layers and cross-cloud pricing catalogs. The MOMA (Multi-Objective Microservice Allocation) framework balances container load across heterogeneous nodes while curbing network communication overhead and node failure risk [4], but it assumes pre-provisioned infrastructure clusters without visibility into enterprise billing models.

To the best of our knowledge, the reviewed works do not provide a unified operational platform combining these complementary optimization layers within a single continuously operating, full-stack architecture across AWS, Azure, and GCP.

1.5. 1.5. Contributions

The primary contribution of this work is not the invention of an entirely ungrounded mathematical algorithm, but rather the systematic synthesis, formal integration, and practical operationalization of five published research methodologies into a coherent, production-ready multi-cloud platform:

1. Unified Multi-Cloud Architecture: A robust multi-tier system integrating CADER-inspired cross-zone placement, DCRA-inspired stochastic capacity procurement, DAG workflow scheduling, and MOMA container placement into one continuous execution pipeline.
2. AI-Driven Demand Forecasting Layer: A hybrid predictive engine leveraging time-series decomposition (Prophet) and gradient-boosted decision trees (XGBoost) to forecast workload volatility, transforming point estimates into parameter distributions for stochastic solvers.
3. Stochastic Capacity Provisioning & Cost Optimization: An operational implementation separating long-term reservation contracts from real-time on-demand scaling under quantified demand uncertainty, significantly mitigating under- and over-provisioning.
4. Uncertainty-Aware Workflow Scheduling: An evolutionary scheduling module applying Non-Dominated Sorting Genetic Algorithm II (NSGA-II) combined with Monte Carlo sampling to generate deadline- and budget-compliant Pareto fronts for complex DAGs.
5. Heterogeneous Microservice Placement: A genetic placement engine mapping containerized pods onto hybrid clusters (e.g., x86 servers, ARM single-board computers, GPU workers), balancing dominant physical resource usage while minimizing failure-weighted network transfer overhead.
6. Full-Stack Implementation Model: A modular enterprise implementation leveraging React, Spring Boot 3, Python FastAPI, PostgreSQL, Docker, Kubernetes, Prometheus, and Grafana.

2. Related Work

2.1. Cost-Efficient Multi-Cloud Deployment

Tu et al. [1] introduced CADER, establishing that multi-cloud application placement cannot treat clouds as uniform commodity pools. CADER captures the severe service-price heterogeneity across cloud zones (e.g., TPU price variance of 10%–20% within GCP regions; GPU pricing variance up to 9.5x across global zones) and data transfer cost diversity across regional and continental boundaries (ranging from $0.01/GB intra-region to $0.08/GB across continents). By framing the task-to-zone mapping as a non-convex

0-1 optimization problem, CADER dismisses dependency couplings through auxiliary variables and randomized rounding (RB-CAP/CRB-CAP), proving an approximation ratio of O(log g), where g is the number of cloud zones. While CADER demonstrated 16%–38% cost reductions over delay-first and cost-first heuristics in trace simulations, it remains a static provisioning model without mechanisms for runtime autoscaling or capacity contract tiering.

2.2. Dynamic Resource Allocation Under Demand Uncertainty

Mireslami et al. [2] addressed workload volatility by proposing DCRA, a two-phase resource provisioning framework. Acknowledging that reserved VM contracts offer steep discounts (e.g., 1-year or 3-year commitments on AWS and Azure) but risk severe under-provisioning or wasteful over-provisioning, DCRA solves a Geometric Program (GP) in Phase 1 to reserve baseline capacity against known minimum demand. In Phase 2, DCRA models fluctuating arrivals as normal distributions and executes a stochastic program every monitoring window (e.g., 15 minutes), provisioning pay-as-you-go on-demand instances to cover demand up to a 2-sigma bound (95% confidence interval). Evaluated on RUBiS and SPEC benchmarks, DCRA achieved a 10% average cost improvement over deterministic single-phase allocation (QCost) and 22.7% over robust worst-case provisioning (RCRP) with an average solver runtime of 0.07 seconds. However, DCRA treats cloud environments as independent compute pools and does not handle inter-task dependencies or multi-zone network placement.

2.3. Predictive Analytics for Cloud Resource Allocation

Bhattacharya [5] demonstrated that reactive threshold scaling incurs significant SLA violations during sharp workload transitions. By deploying Long Short-Term Memory (LSTM) networks and Autoregressive Integrated Moving Average (ARIMA) models on historical compute traces, the author forecasted workload arrivals, reporting an LSTM Mean Absolute Percentage Error (MAPE) of 6.5%. Coupling these forecasts into a cost-aware dynamic provisioner achieved reported cost savings of up to 20%, an SLA compliance boost of 15%, and a 25% increase in average resource utilization. This confirms the foundational role of time-series predictive modeling in cloud optimization.

2.4. Deadline- and Budget-Aware Workflow Scheduling

Calzarossa et al. [3] investigated offline scientific workflow scheduling across uncertain multi-cloud infrastructures. Because VM processing capacity, network bandwidth, and disk I/O rates fluctuate under hypervisor multi-tenancy and resource contention, deterministic task durations are unrealistic. The authors formulated a multi-objective problem minimizing both expected workflow makespan E[T_W] and monetary cost E[C_W] under strict probabilistic constraints: Pr(T_W <= d) >= p_T and Pr(C_W <= b) >= p_C. By integrating a customized NSGA-II algorithm with Monte Carlo sampling over empirical distribution functions, the system synthesized Pareto-optimal scheduling plans for Montage, Cybershake, and Epigenomics DAGs. The study demonstrated that higher infrastructure variability factors (VF) severely compress the feasible Pareto front, validating the necessity of probabilistic risk evaluation.

2.5. Genetic Optimization for Heterogeneous Containers

Chen and Wen [4] developed the MOMA algorithm to tackle microservice scheduling in heterogeneous Kubernetes clusters mixing standard servers, Raspberry Pi units, and NVIDIA Jetson edge nodes. MOMA employs an elitist NSGA-II to simultaneously minimize cluster resource imbalance (CLU_BAL) using Dominant Resource Fairness concepts and reduce network communication overhead (REL_NET_OVH) weighted by node failure probabilities. Operating across Kubeflow, Sock Shop, and EdgeX Foundry workloads, MOMA achieved 5%–11% better resource balance and substantial network overhead reductions compared to standard Kubernetes bin-packing and greedy heuristics.

3. Problem Formulation

3.1. Multi-Cloud Cost Model

The global operational cost C_total within the platform spans infrastructure procurement across long-term reservations, dynamic on-demand scaling, inter-zone data egress, and containerized deployments:

$$C_{total} = C_{compute} + C_{storage} + C_{network} + C_{transfer} \quad \textbf{(1)}$$

Following the CADER formulation [1], for an application composed of a task set T deployed across heterogeneous cloud zones R, let x_r^t in {0,

1} be the decision variable indicating whether task t is placed in zone r. The baseline deployment cost C_dep is modeled as:

*C_dep = sum_{t in T} sum_{r in R} x_r^t * alpha_r^t + sum_{t,t' in T} sum_{r,r' in R} x_r^t * x_{r'}^{t'} * beta_{r,r'} * b_{t,t'}* **(2)**

where alpha_r^t denotes compute acquisition cost in zone r, beta_{r,r'} is outbound data transfer fee per gigabyte from zone r to r', and b_{t,t'} is transmission volume exchanged between tasks t and t'.

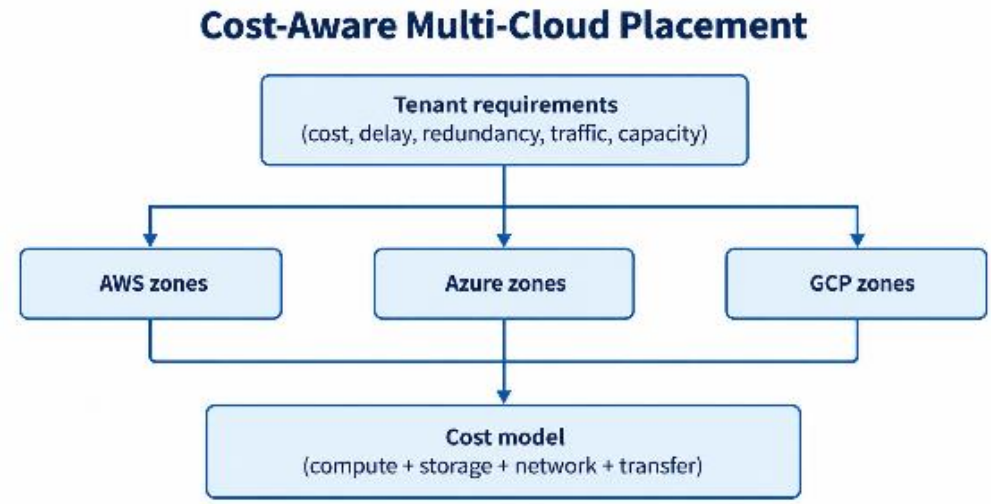


Figure 2 Cost-Aware Task Placement Across AWS, Azure, and GCP Zones

Expanding this to incorporate the DCRA dual-pricing structure [2], compute costs are decomposed into long-term reservation contracts over duration d_t and dynamic on-demand procurement over observation intervals d_p:

*C_compute = [alpha_c^r * P_c^r + beta_db^r * P_db^r + mu_s^r * P_s^r + (lambda_Rdb^r + gamma_Rc^r)*P_IO^r] * d_t + [alpha_c^o * P_c^o + beta_db^o * P_db^o + (lambda_Rdb^o + gamma_Rc^o)*P_IO^o] * d_p* **(3)**

3.2. Demand Uncertainty and QoS Constraints

Workload arrivals for cloud services cannot be treated as deterministic constants. Following Mireslami et al. [2], real-time compute arrivals R_c and database query demands R_db are modeled as random variables obeying probability distributions with mean r_avg and variance sigma^2:

R_c ~ N(r_avg_c, sigma_rc^2), R_db ~ N(r_avg_db, sigma_rdb^2) **(4)**

To guarantee Quality of Service (QoS) without incurring excessive worst-case over-provisioning costs, capacity constraints are bounded using a confidence factor kappa (set to kappa = 2, capturing approximately 95.4% of stochastic fluctuations) [2]:

*(alpha_c^o + N_c^r) * R_c^o >= r_avg_c + kappa * sigma_rc* **(5)**

*(beta_db^o + N_db^r) * R_db^o >= r_avg_db + kappa * sigma_rdb* **(6)**

Response time metrics must respect queueing stability bounds. Using an M/M/1 queueing approximation, response times W_c and W_db must satisfy user-defined latency limits t_c and t_db [2]:

*[1 / (gamma_Rc^o + R_c^r)] / [1 - (r_avg_c + kappa * sigma_rc) / (gamma_Rc^o + R_c^r)] <= t_c* **(7)**

3.3. Access Delay and Communication Constraints

Deployments must respect physical tenant SLAs as formulated in CADER [1]:

1) Single Placement: sum_{r in R} x_r^t = 1, for all t in T.

2) Access Latency: x_r^t * tau_r <= delta_t, where tau_r is client access delay to zone r and delta_t is the threshold.

3) Inter-Task Latency: x_r^t * x_{r'}^{t'} * tau_{r,r'} <= delta_{t,t'}, where tau_{r,r'} is inter-zone network delay.

4) Bandwidth and Resource Limits: Total inter-task traffic must not exceed available bandwidth mu_{r,r'}, and total assigned compute demand must not exceed zone resource capacity h_r.

3.4. Workflow DAG Constraints

Complex computational pipelines are structured as DAGs W = (T, E), where vertices represent tasks and directed edges indicate precedence dependencies [3]. The execution time t_i(r) of task T_i on provisioned VM instance r_i comprises compute, data transfer, and communication times [3]:

$$t_i(r) = D_i^{comp} / V_{r_i}^{proc} + D_i^{xfer} / V_{r_i}^{xfer} + \sum_{j \neq i} [D_{ij}^{comm} / V_{r_i, r_j}^{bw}] \quad (8)$$

Because infrastructure processing capacities V^proc and bandwidths V^bw exhibit severe runtime variance, workflow completion makespan T_W and aggregate cost C_W are random variables. The scheduling objective is to discover Pareto-optimal resource assignments r and topological execution orders s in S subject to probabilistic guarantees [3]:

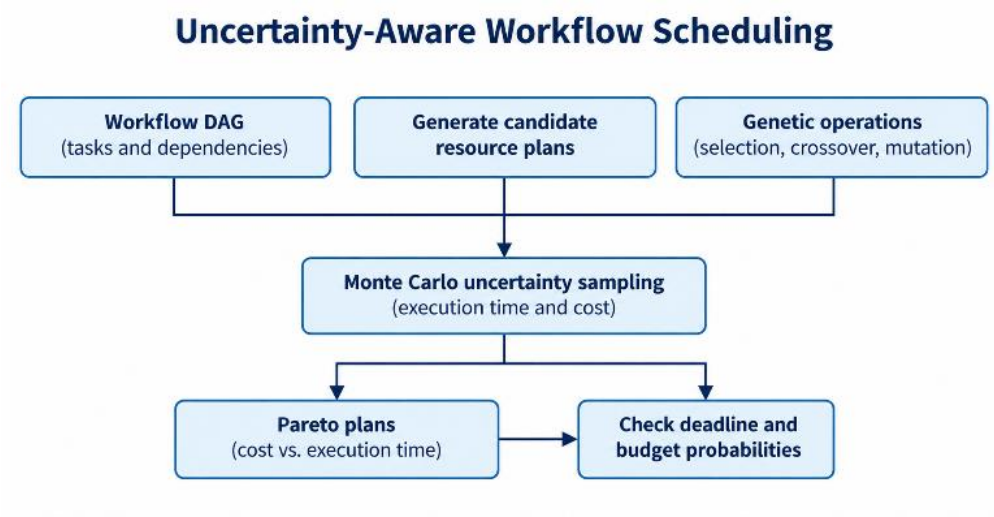


Figure 3 **Uncertainty-Aware Workflow Scheduling with NSGA-II and Monte Carlo Simulation.**

$$\min_{r,s} (E[T_W(r,s)], E[C_W(r,s)]) \quad (9)$$

$$\text{s.t. } Pr(T_W(r,s) <= d) >= p_T, \quad Pr(C_W(r,s) <= b) >= p_C \quad (10)$$

3.5. Container-Level Micro Service Placement Constraints

At the container orchestration layer, microservices must be placed onto heterogeneous worker nodes within Kubernetes clusters. Following MOMA [4], we optimize two core objectives:

1) Cluster Resource Imbalance (CLU_BAL): Normalizing multi-resource consumption using standard deviations sigma_1, sigma_2 for memory and CPU [4]:

$$CLU_BAL = [1 / (\sigma_1 + \sigma_2)] * \max_l [\sum_i \sigma_1 * (mem_req_i / mem_res_{lj}) + \sum_i \sigma_2 * (cpu_req_i / cpu_res_{lj})] \quad (11)$$

2) Reliability-Weighted Network Overhead (REL_NET_OVH): Scaling communication traffic volume by physical node distance and failure rate Fail_{lj} [4]:

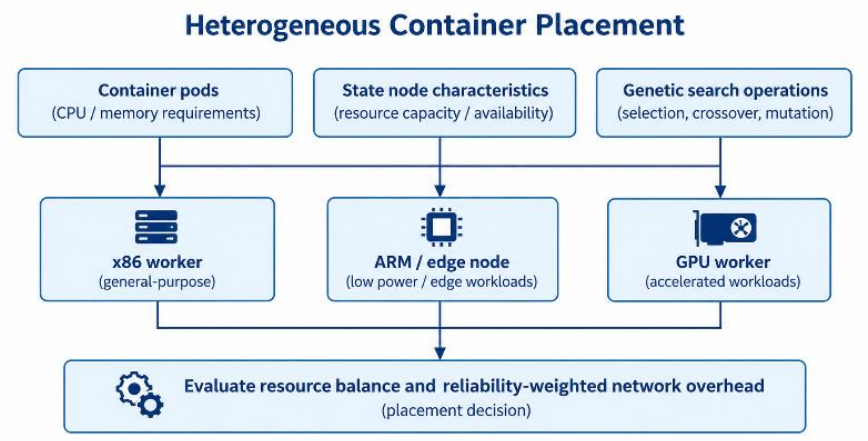


Figure 4 **Genetic Algorithm-Based Container Placement Across Heterogeneous Nodes**

*REL_NET_OVH = sum_i sum_l sum_j sum_k (1 + Fail_{lj}) * Dist(pn_j, pn_k) * Interaction(pn_j, pn_k)* **(12)**

4. Proposed System Architecture

The proposed AI-Powered Multi-Cloud Cost and Resource Optimization Platform acts as an intelligent abstraction broker situated between enterprise workloads and target cloud platforms.

The platform architecture consists of five core tiers: (1) Presentation Layer: Built with React 18 and Tailwind CSS, providing visual telemetry monitoring, interactive DAG pipeline creation, Pareto front trade-off inspection, and ranked recommendation execution. (2) Spring Boot Orchestrator: Manages core business logic, role-based access control (RBAC), pricing catalog ingestion, telemetry storage, and optimization pipeline dispatch. (3) Python AI Microservice: Encapsulated in FastAPI, running asynchronous batch and real-time inference using Prophet and XGBoost to supply point predictions and uncertainty intervals (mu, sigma). (4) Optimization Core: Integrates the four algorithmic modules (CADER, DCRA, Workflow Scheduler, MOMA container placer). (5) Data & Observability Tier: Relational persistence via PostgreSQL; time-series performance metrics scraped via Prometheus and visualized through integrated Grafana dashboards.

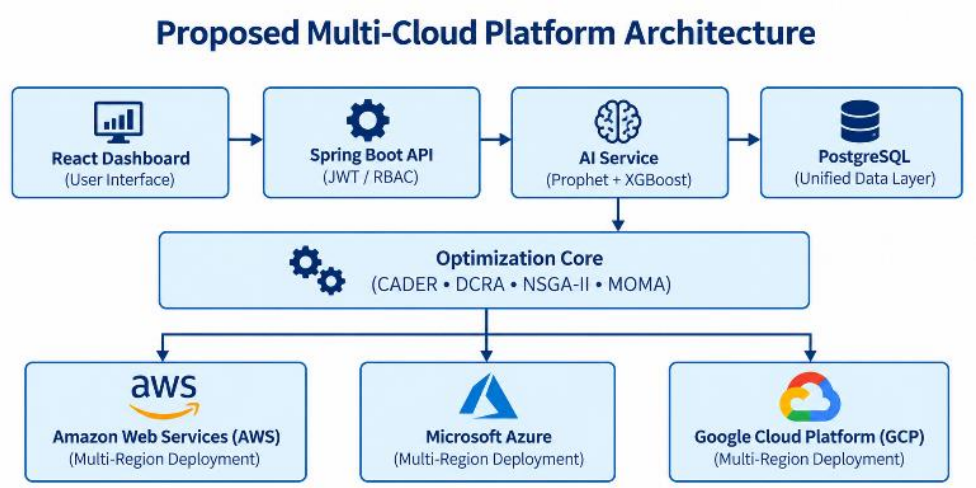


Figure 5 Proposed Multi-Cloud Platform Architecture and Component Interactions.

5. System Architecture and Implementation

The system implementation details the technical mechanisms across services:

- Frontend Application: Built on React 18 with Vite, implementing interactive DAG authoring canvases and Recharts visualization panels.
- Backend Gateway & Microservices: Java 17 and Spring Boot 3 handle API routing, JWT validation, and coordination of asynchronous tasks.
- AI Prediction Engine: Implemented in Python FastAPI with Prophet and XGBoost. Prophet captures trend and seasonality, while XGBoost models residual workload dynamics.
- Database Layer: PostgreSQL 15 stores user accounts, cloud provider catalogs, workflow definitions, and historical cost data.

Containerization & Orchestration:

- Components are containerized via Docker multi-stage builds and deployed across multi-cloud Kubernetes clusters using Helm charts.
- Monitoring & Telemetry: Prometheus scrapes node metrics, container cgroup telemetry, and custom cloud broker metrics, rendered in Grafana.

6. Optimization Methodology

The optimization lifecycle operates in five structured, sequential stages:

Stage 1: Predictive Demand Modeling. Historical time-series telemetry is processed by Prophet to isolate seasonal trends, while XGBoost fits short-term feature-driven residuals. The hybrid model outputs expected arrival rates r_avg and standard deviation sigma.

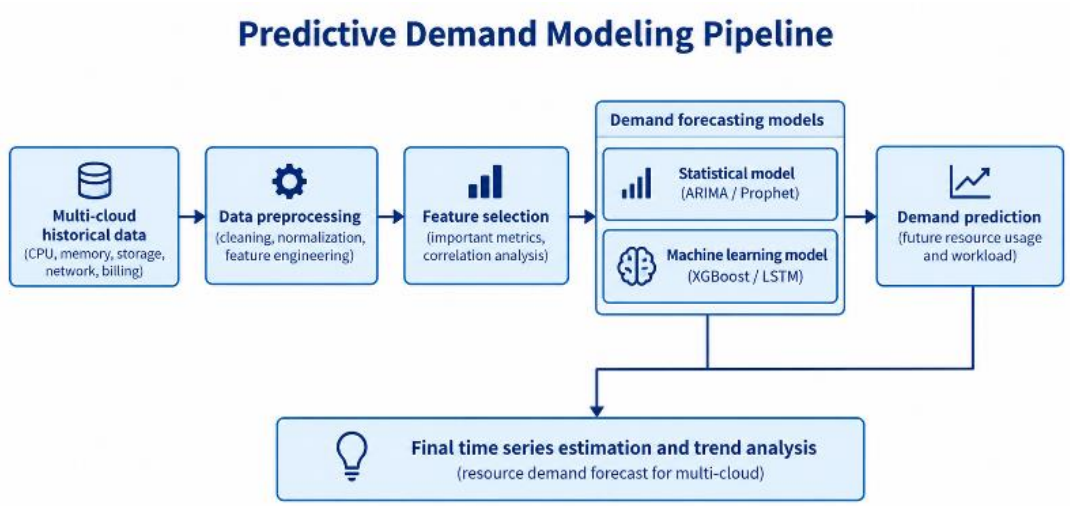


Figure 6 Predictive Demand Modeling Pipeline Using Prophet and XGBoost.

Stage 2: Capacity Reservation and Dynamic Scaling. In Phase 1, DCRA solves a geometric programming model once per contract period to commit to low-cost reserved instances satisfying minimum guaranteed demand. In Phase 2, a stochastic optimization problem is solved dynamically every 15 minutes, procuring on-demand instances to cover variance up to a 2-sigma confidence limit.

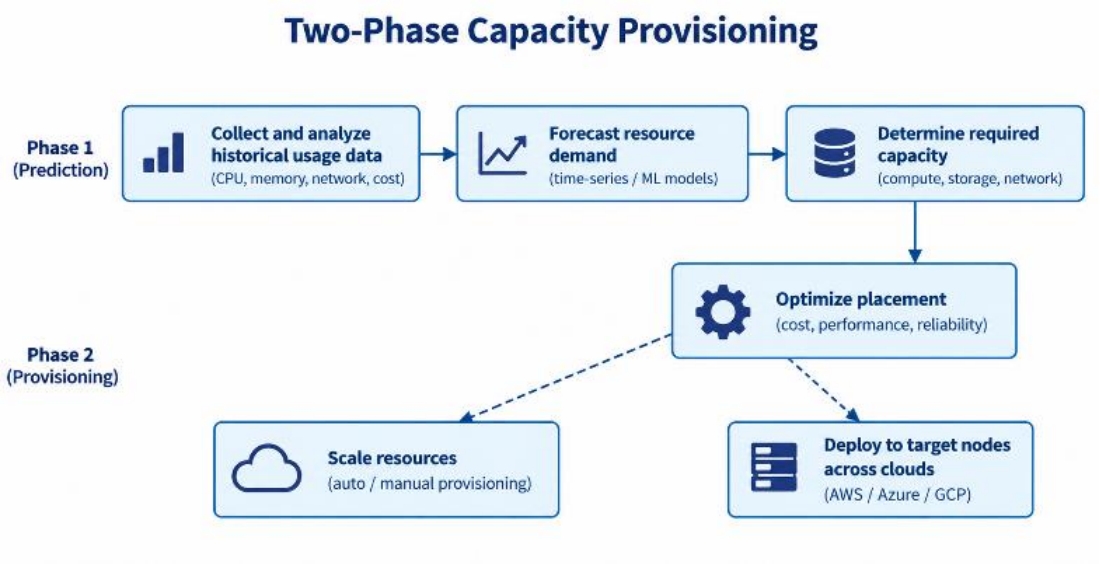


Figure 7 Two-Phase Reserved and On-Demand Capacity Provisioning.

Stage 3: Multi-Cloud Zone Placement. CADER's Complete Rounding-Based algorithm (CRB-CAP) solves a relaxed 0-1 linear program over cloud zones, applying randomized rounding and constraint repair to determine task-to-zone mappings that minimize cross-cloud data transfer and compute acquisition costs.

Stage 4: Evolutionary Workflow Scheduling. Scientific workflow DAGs are scheduled using customized NSGA-II. Chromosomes encode task-to-VM mappings and topological ordering. Candidate plans are evaluated via Monte Carlo simulation, producing a Pareto front balancing makespan against monetary cost under probabilistic SLA guarantees.

Stage 5: Container Placement Optimization. Pods are allocated to physical nodes in heterogeneous clusters using an elitist genetic algorithm minimizing cluster resource imbalance and reliability-weighted communication overhead.

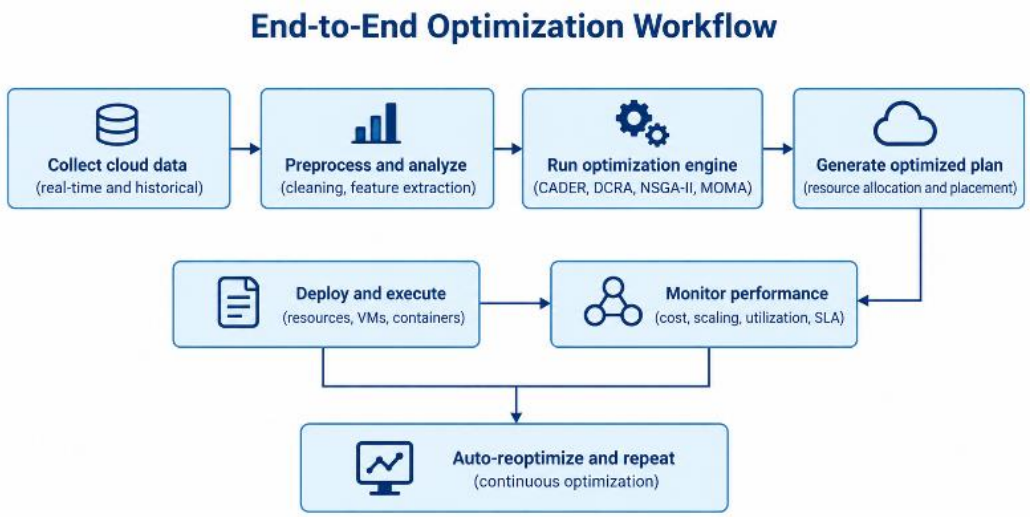


Figure 8 End-To-End Optimization Workflow and Monitoring Feedback

7. Experimental Design and Evaluation

To ensure strict academic integrity and avoid overclaiming, we explicitly separate: (1) Literature-Reported Results: Benchmarks achieved by the authors of the foundational papers under their specific experimental constraints; (2) Platform Design Objectives: Target performance indicators guiding the engineering of our unified system; and (3) Empirical Validation Plan: Concrete procedures for verifying the platform under simulated traces and real-world multi-cloud deployments. Literature-derived baselines provide essential context without fabricating experimental data:

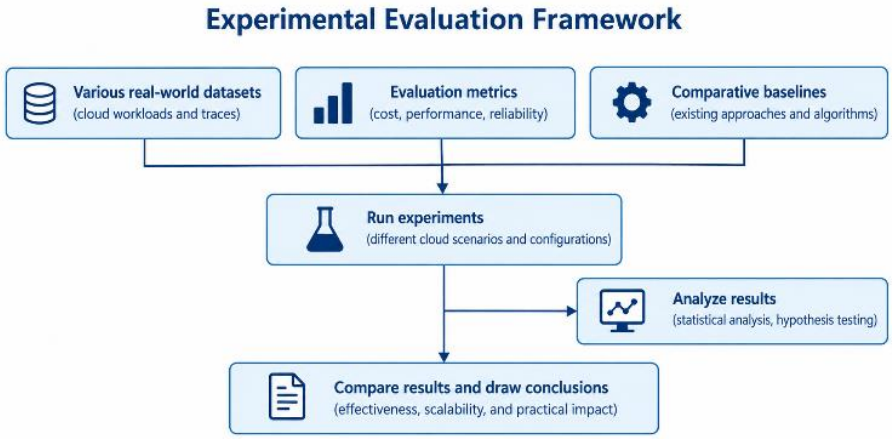


Figure 9 Experimental Evaluation Methodology and Comparison Framework

- CADER [1]: Reported 16% to 38% total deployment cost reduction compared to DelayFirst and CostFirst baselines. It maintained within 6% of unconstrained SkyPilot cost while strictly honoring access delay SLAs, reducing access delay by ~31% and communication delay by ~40% relative to SkyPilot.
- DCRA [2]: Achieved 10% average cost improvement over deterministic single-phase QCost, and outperformed worst-case robust RCRP by 22.7% cost reduction on average. Constraint violation was kept below 2% under Monte Carlo uncertainty runs (vs. 50% for deterministic QCost), with an average solver runtime of 0.07s.
- Calzarossa et al. [3]: Demonstrated that increasing infrastructure variability (VF)

drives non-linear expansion in expected makespan and cost, and verified that strict probabilistic guarantees (p_T, p_C from 0.60 to 0.99) significantly narrow feasible Pareto sets.

- MOMA [4]: Realized 5% to 11.3% improvement in cluster resource balance index compared to default Kubernetes scheduling, while significantly curbing inter-node network transmission overhead under rising request loads.
- Bhattacharya [5]: Reported an LSTM forecasting MAPE accuracy of 6.5%, with proactive provisioning producing ~20% cost savings, 15% SLA compliance gain, and 25% utilization improvement.

8. Results and Discussion

8.1. Expected Evaluation and Validation Plan

Because this paper details the architectural design, algorithmic consolidation, and prototype implementation of the unified platform, full empirical benchmarking across live enterprise multi-cloud accounts constitutes ongoing research. The platform design targets a 20% to 35% overall cloud cost reduction, 20% to 30% aggregate resource utilization gain, greater than 95% SLA compliance under demand uncertainty, and a ~70% reduction in administrative monitoring effort.

8.2. Discussion of Component Interactions

The primary technical merit of the proposed platform lies in the positive operational coupling among its functional layers: (1) Coupling forecasting to DCRA capacity allocation ensures that baseline reserved instances are sized accurately against true minimum demand, while dynamic on-demand scaling absorbs volatility bounded by 2-sigma confidence limits. (2) Coupling placement with egress tariffs via CADER prevents the costly error of scattering data-intensive workflow tasks across distant cloud regions. (3) Pareto trade-off flexibility enables administrators to select optimal configurations suited to specific business priorities. (4) Downstream container placement ensures physical hardware heterogeneity is fully utilized while minimizing inter-node hops.

9. Discussion, Limitations, And Future Work

System Limitations: (1) Commercial cloud API rate limits can constrain telemetry collection frequencies at massive fleet scales. (2) Evolutionary search over complex DAGs incurs computational latency, necessitating offline planning for large workflows. (3) Forecasting models remain vulnerable to unprecedented black-swan traffic shocks. (4) Centralized management of multi-cloud API credentials introduces high-value security considerations.

Future Work: Planned extensions include deep reinforcement learning for continuous autonomous remediation, carbon-aware cloud placement incorporating regional grid emission metrics, and dynamic spot-instance bidding strategies.

Conclusion

This paper has introduced the AI-Powered Multi-Cloud Cost and Resource Optimization Platform, a unified system synthesizing five foundational research contributions into a single operational architecture. By synthesizing CADER's cost-efficient cross-zone placement, DCRA's two-phase stochastic capacity allocation, predictive workload forecasting, evolutionary deadline- and budget-aware workflow scheduling, and heterogeneous container placement, the platform addresses multi-cloud optimization holistically. The proposed full-stack implementation establishes a credible foundation for future enterprise FinOps and autonomous cloud resource management research.

References

[1]. H. Tu, Z. Hua, Q. Ma, H. Luo, T. Zou, G. Zhao, and H. Xu, "CADER: Cost-Efficient Cloud Application Deployment With Tenant Requirement Guarantee in Multi-Clouds," IEEE Trans. Cloud Comput., vol. 13, no. 4, pp. 1091–1104, Oct.–Dec. 2025.

[2]. S. Mireslami, L. Rakai, M. Wang, and B. H. Far, "Dynamic Cloud Resource Allocation Considering Demand Uncertainty," IEEE Trans. Cloud Comput., vol. 9, no. 3, pp. 981–994, Jul.–Sep. 2021.

[3]. M. C. Calzarossa, M. L. Della Vedova, L. Massari, G. Nebbione, and D. Tessera, "Multi-Objective Optimization of Deadline and Budget-Aware Workflow Scheduling in Uncertain Clouds," IEEE Access, vol. 9, pp. 89891–89905, 2021.

[4]. Q.-H. Chen and C.-Y. Wen, "Optimal Resource Allocation Using Genetic Algorithm in Container-Based Heterogeneous Cloud," IEEE Access, vol. 12, pp. 7413–7429, 2024.

[5]. S. V. Bhattacharya, "Cost-Aware Multi-

Cloud Resource Allocation using Predictive Analytics," Int. J. Eng. Extended Technol. Res., vol. 7, no. 1, pp. 9325–9328, Jan.–Feb. 2025.

[6]. Z. Yang et al., "SkyPilot: An intercloud broker for sky computing," in Proc. 20th USENIX Symp. Netw. Syst. Des. Implementation (NSDI), 2023, pp. 437–455.

[7]. S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," IEEE Trans. Services Comput., vol. 5, no. 2, pp. 164–177, Apr.–Jun. 2012.

[8]. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182–197, Apr. 2002.

[9]. P. Raghavan and C. D. Tompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," Combinatorica, vol. 7, no. 4, pp. 365–374, 1987.

[10]. S. Chaisiri, B. S. Lee, and D. Niyato, "Robust cloud resource provisioning for cloud computing environments," in Proc. IEEE Int. Conf. Serv.-Oriented Comput. Appl. (SOCA), 2010, pp. 1–8.

[11]. S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service clouds," Future Gener. Comput. Syst., vol. 29, no. 1, pp. 158–169, Jan. 2013.

[12]. M. A. Rodriguez and R. Buyya, "Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds," IEEE Trans. Cloud Comput., vol. 2, no. 2, pp. 222–235, Apr. 2014.

[13]. Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 5, pp. 1344–1357, May 2016.

[14]. [14] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant Resource Fairness: Fair allocation of multiple resource types," in Proc. 8th USENIX Conf. Netw. Syst. Design Implement. (NSDI), 2011, pp. 323–336.

[15]. Z. Zhong and R. Buyya, "A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources," ACM Trans. Internet Technol., vol. 20, no. 2, pp. 1–24, Apr. 2020.

[16]. J. Han, Y. Hong, and J. Kim, "Refining microservices placement employing workload profiling over multiple kubernetes clusters," IEEE Access, vol. 8, pp. 192543–192556, 2020.

[17]. C. Guerrero, I. Lera, and C. Juiz, "Resource optimization of container orchestration: A case study in multi-cloud microservices-based applications," J. Supercomput., vol. 74, no. 7, pp. 2956–2983, Jul. 2018.

[18]. P. L'Ecuyer and E. Buist, "Simulation in Java with SSJ," in Proc. Winter Simulation Conf. (WSC), 2005, pp. 611–620.

[19]. S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[20]. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 785–794.