Computer Fraud and Security
ISSN (online): 1873-7056

Automated Right-Sizing of Cloud Compute Resources: A Data-Driven Framework for Enterprise Cost Optimization

Manoj Kumar Reddy Kalakoti

Department of Computer Science, Texas A&M University-Kingsville,TX, USA

***Abstract*—**Cloud infrastructure environments frequently suffer from resource over-provisioning, leading to significant financial inefficiencies across enterprise organizations. This article introduces an integrated FinOps-aware optimization framework that combines continuous monitoring, intelligent workload analysis, and automated remediation within a unified platform engineering architecture. The solution leverages real-time utilization metrics across CPU, memory, and disk I/O parameters through multiple observability platforms. An analytics engine categorizes workloads into distinct usage patterns—burstable, steady, and idle—enabling precise resource optimization recommendations. The framework integrates with infrastructure-as-code tools to execute automated remediation pipelines, adjusting instance types, container resource limits, and node configurations based on actual demand patterns. Implementation follows blue-green deployment strategies ensuring zero-downtime transitions during resource adjustments. Results demonstrate substantial cost reduction potential while maintaining performance standards: 35-45% cost reduction per optimized instance, CPU utilization improvement from 15-20% to 65-75%, and memory allocation efficiency increased from 30% to 70-80%. This data-driven solution represents a significant advancement in FinOps practices, offering a scalable model for organizations seeking to optimize cloud expenditure without compromising service quality.

***Index Terms*—**Cloud resource optimization, Automated right-sizing, FinOps, Platform automation, Infrastructure-as-code

1. INTRODUCTION

1.1 Resource allocation practices within enterprise cloud ecosystems

Enterprise organizations face immense challenges in determining a proper amount of resources for cloud workloads. Decision-makers routinely choose capacity that is improperly sized because they fear service performance degradation if demand increases. These choices lead to deploying full racks of infrastructure when the organization will never need all the computing power, memory allocation, or storage capacity each rack provides. Many organizations are grappling with the challenge of striking the right balance between the flexibility of cloud computing and the economic reality of keeping unused resources. Research into algorithmic approaches for optimizing resource allocation in data centers, specifically the use of genetic algorithms, has uncovered repeated inefficiencies caused by common provisioning approaches [1]. Novel research reflects how legacy models for managing on-premise infrastructure impact how cloud solutions are decided on and utilized, creating a cycle of wasted investment.

1.2 Financial consequences of excessive resource provisioning

Overprovisioned infrastructure has a severe cascading financial impact on enterprise budgets. The direct and immediate costs are clear and include wasted compute cycles and idle storage that could otherwise be put to use. Indirect costs include management overhead, an increased security attack surface, and a loss of focus on the core areas of strategy and future direction. Cloud billing structures, pricing tiers, and contract commitments also obfuscate the overall cost of overprovisioning while enabling the company to produce quarterly new spend growth reports to leadership, showing alarming patterns of increased utility consumption. Using multiple clouds initially seems like a cost-reduction strategy; however, dealing with multiple cloud providers with their own charging schemes and policies on use greatly complicates any reconciliation of consumption. The overall financial impacts usually end up being a shocking surprise for leadership teams who saw cloud as a cost reduction driver but didn't foresee widespread overallocation as a significant source of additional spend.

1.3 Disparity between provisioned capacity and real-world usage

Real resource consumption hardly ever resembles the capacity planning assumptions that were made in advance, creating permanent efficiency gaps in cloud infrastructures. Workload analysis showed enormous spikes and valleys in resource

consumption, with many systems having drawn insufficient utilization at very safe utilization levels and short windows of high demand. Drawing from international operations adds to the variations in resource consumption as the time zones convert utilization into rolling waves over an extended time period, while much of the resource consumption is not utilized. Incomplete examinations of overprovisioning in production environments have shown the way static resource allocation fails to adjust to the continuum of workloads in contemporary applications [2]. These examinations showed mismatches between resource provisioning models and the dynamic characteristics of cloud-native applications.

1.4 Objectives and boundaries of the current investigation

This case study presents an automated right-sizing framework designed to address these challenges through continuous monitoring, intelligent workload analysis, and automated remediation. The framework represents the first comprehensive integration of FinOps principles directly into automated platform engineering workflows, creating a unified system that simultaneously optimizes costs and maintains operational excellence. The primary objective is to demonstrate how data-driven automation can bridge the gap between resource allocation and actual demand, achieving substantial cost optimization while maintaining performance standards. The scope encompasses both production and non-production environments within a large-scale enterprise cloud infrastructure, focusing on compute resources including virtual machines, containers, and node pools. Through systematic implementation of automated right-sizing mechanisms, this initiative aims to establish a scalable model for resource optimization that aligns with modern FinOps principles and platform engineering best practices.

2. LITERATURE REVIEW AND THEORETICAL FRAMEWORK

2.1 Transformation of cost management tactics in cloud computing environments

Cost control mechanisms for cloud infrastructure have shifted dramatically over the past decade. Early adopters relied on rudimentary tactics like bulk purchasing commitments and basic scheduling scripts. These primitive methods reflected both technological constraints and limited understanding of cloud economics. Gradually, enterprises discovered that traditional procurement models failed to capture the dynamic nature of on-demand resources. Organizations began experimenting with elastic scaling, spot market bidding, and cross-provider distribution strategies. Today's landscape features sophisticated orchestration platforms that blend artificial intelligence with economic modeling. Recent publications highlight how modern enterprises treat cost efficiency as an architectural requirement rather than an operational afterthought [3]. This philosophical shift acknowledges that cloud platforms demand fundamentally different economic thinking compared to fixed-capacity data centers.

Table 1. Evolution of Cloud Cost Optimization Strategies [3]

Era	Time Period	Primary Strategy	Key Characteristics	Limitations
Early Adoption	2006-2012	Reserved Instances	Bulk purchasing, long-term commitments	Inflexible, requires accurate forecasting
Maturation	2013-2017	Auto-scaling & Scheduling	Reactive scaling, time-based controls	Delayed response, limited intelligence
Modern	2018-2022	Hybrid Optimization	Spot instances, multi-cloud arbitrage	Complex management requires expertise
Contemporary	2023-Present	AI-Driven Automation	Predictive analytics, continuous optimization	Initial setup complexity

2.2 Financial operations methodology within engineering ecosystems

FinOps appeared for a sector-specific need when technology teams had a difficult time explaining the rising cloud bills to their executive leaders. The discipline creates a common vocabulary between engineers deploying resources and finance professionals monitoring the spending. Engineering teams now factor budget into deployment pipelines, architectural reviews, and capacity planning. Development pipelines employ automated guardrails that limit budget overruns while maintaining technical agility. Recent FinOps frameworks promote decentralized decision-making in which individual

teams manage their cloud budgets by following organizational policies and guidelines [4]. Platform engineers gain considerable benefits from FinOps, as they are responsible for common infrastructure layers that often generate spending behavior at the enterprise level.

2.3 Contrasting manual inspection with algorithmic optimization for capacity adjustment

Human-led optimization processes follow predictable cycles; monthly reports are generated, analysts comb through spreadsheets, and then recommendations move through a chain of approvals before changes happen weeks after the initial observation. This time delay between observation, conclusions, and action creates a foothold for inefficiencies, building up financial waste. Manual review also provides an unreliable decision-making process. Two analysts looking at the same data might come to disparate conclusions, or one will miss subtleties in the recommendation because they didn't note the time frame. Algorithmic methods continuously operate, studying preventative operation metrics in near real-time, and making changes within constraints. Automated systems displace humans from the decision-making bottleneck and maintain an audit trail for the compliance framework. Machine-led approaches will outperform human-operated methods at detecting micro-patterns across thousands of resources at once; humans would be overwhelmed with the need to analyze and evaluate [3]. In automation development, developers will benefit from keeping an eye on the automated scripts, as certain automated decisions can become aggressive and, sadly, expensive when the performance of an application is a significant issue.

2.4 Mathematical models supporting workload behavior prediction

Resource optimization relies on mathematical frameworks borrowed from operations research, statistics, and computer science. Queue theory provides insights into request arrival patterns and processing delays. Trending growth is distinguished from random fluctuations and cyclical variations by time series decomposition. Algorithms for machine learning find relationships between infrastructure demand and business events. Workload taxonomy schemes categorize applications based on resource consumption signatures: constant baseline consumers, periodic spike generators, and gradually scaling services. Prediction accuracy improves when models incorporate business context alongside technical metrics. Holiday shopping seasons, marketing campaigns, and product launches create demand patterns that purely technical analysis might miss [4]. Theoretical advances continue to refine prediction capabilities, with ensemble methods combining multiple forecasting techniques to reduce error rates.

3. METHODOLOGY: DESIGN AND ARCHITECTURE OF THE AUTOMATED RIGHT-SIZING FRAMEWORK

3.1 Collecting performance data through distributed monitoring systems

Today, companies utilize different types of monitoring systems related to different infrastructure behaviors, each based on particular operational needs. For example, in an environment with container-based workloads, Prometheus is the main collector, as it captures metrics on a frequent schedule to account for ephemeral spikes in resource usage. CloudWatch, in addition to Prometheus, provides access to cloud platform internals and includes metrics that are exposed at the platform level, which are not accessible from traditional third-party tools. Next, as a vendor with an aggregator dashboard, Datadog collects metrics from multiple sources, bringing that information into a unified dashboard. Even though the three monitoring platforms operate independently from one another, they all ingest data into a joined data pipeline, with the intent of redundancy and completeness. With each collection system generally synchronized on the clock, temporal accuracy is maintained when correlating events with multiple collections from separate sources. Raw metrics introduce transformation processes that convert changes in format into standard representations of the data suitable for downstream clients [6]. The collective collection from multiple sources preserves data integrity and reduces the risks associated with dropping metrics, while leaving the measuring baselines consistent across heterogeneous infrastructure components.

Fig. 1. Automated right-sizing framework architecture with integrated monitoring, analytics, and deployment layers.

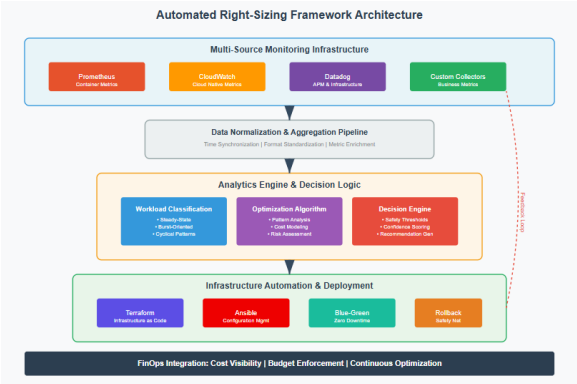

Table 2. Monitoring Platform Comparison [6]

Platform	Strengths	Best Use Cases	Metric Frequency	Integration Complexity
Prometheus	Container-native, high-frequency sampling	Kubernetes workloads	15-60 seconds	Medium
CloudWatch	Native cloud integration	AWS resources	1-5 minutes	Low
Datadog	Extensive integrations	Hybrid environments	15-60 seconds	Low

3.2 Algorithmic classification of resource consumption behaviors

Computational workloads exhibit recognizable consumption signatures that algorithms can detect and categorize systematically. The classification mechanism examines utilization trends, measuring baseline consumption, deviation patterns, spike occurrences, and duration profiles. Applications maintaining constant resource usage fall into steady-state categories, presenting straightforward optimization targets. Services experiencing irregular demand bursts require different treatment, preserving capacity cushions for unexpected load increases. Business-driven cycles create predictable patterns, where resource needs fluctuate according to operational schedules. Moving window calculations identify genuine pattern shifts versus temporary disruptions. Clustering mathematics groups with workloads sharing similar characteristics streamlines bulk optimization processes. Historical outcomes feed back into classification logic, refining detection accuracy over successive iterations.

Table 3. Workload Classification Framework [9]

Workload Type	Resource Pattern	Optimization Strategy	Risk Level	Typical Examples
Steady-State	Constant utilization	Aggressive downsizing	Low	Background processors, databases
Burst-Oriented	Intermittent spikes	Maintain headroom	Medium	Web servers, API endpoints
Cyclical	Predictable patterns	Time-based scaling	Low	Batch jobs, reporting systems
Unpredictable	Random variations	Conservative sizing	High	User-facing applications

3.3 Processing architecture for optimization recommendations

The analytical system transforms monitoring data through staged evaluation processes that culminate in actionable sizing proposals. Preprocessing filters eliminate misleading data from maintenance periods or deployment events that would distort normal usage calculations. Mathematical operations compute representative workload profiles using both average values and distribution characteristics. Decision algorithms explore multiple sizing options, calculating cost-benefit ratios for each possibility. Protection mechanisms prevent excessive resource reduction for mission-critical systems, preserving operational margins based on observed volatility. Confidence metrics accompany each recommendation, reflecting data completeness and pattern consistency. Modular design allows specialized analyzers for particular workload classes or organizational policies. Every analytical step generates audit records, supporting retrospective examination of optimization decisions.

3.4 Connecting analytical output to automated deployment systems

Infrastructure automation platforms receive optimization directives through structured interfaces that translate recommendations into configuration changes. Terraform definitions incorporate sizing parameters as variables, adjusting compute specifications according to analytical findings. Ansible automation sequences coordinate resource modifications, managing dependencies to prevent service interruptions. Synchronization logic maintains alignment between recommended configurations and deployed infrastructure, triggering corrections when discrepancies arise [5]. Source control repositories preserve modification history, supporting recovery procedures if optimizations produce adverse effects. Testing protocols verify performance adequacy after resource reductions, confirming that applications operate within acceptable parameters. Execution planning generates detailed change summaries for review, particularly for high-impact modifications. Cloud-agnostic abstractions mask provider differences, ensuring consistent optimization regardless of underlying platforms.

4. IMPLEMENTATION: DEPLOYMENT STRATEGY AND TECHNICAL EXECUTION

4.1 Sequential introduction of optimization across environment tiers

Deployment commenced within isolated development clusters where experimental configurations posed minimal operational hazards. Testing environments provided initial validation grounds, exposing the optimization engine to diverse workload patterns without endangering customer-facing services. Each environmental layer introduced unique challenges: development systems exhibited erratic usage patterns from debugging sessions, while staging environments mimicked production loads more consistently. Progression between tiers followed rigid timelines, enforcing observation windows that captured both immediate effects and delayed reactions to resource adjustments. Teams accumulated operational knowledge through hands-on experience, discovering edge cases that automated testing missed. Configuration parameters evolved through iterative refinement, with each environment contributing insights that shaped subsequent deployments. Lower-tier environments tolerated aggressive optimization settings, revealing breaking points that informed conservative production configurations. This methodical advancement built institutional confidence while establishing operational procedures that would prove essential during production rollout.

4.2 Parallel infrastructure transitions without service disruption

Seamless resource adjustments demanded sophisticated orchestration techniques that maintained service availability throughout optimization cycles. Twin environment construction enabled side-by-side operation of existing and optimized configurations, facilitating risk-free validation before committing to changes. Traffic directors gradually shifted load between environments, monitoring response times and error rates at each increment. Health validators performed deep application checks beyond simple connectivity tests, ensuring functional integrity matched pre-optimization baselines. Synchronization mechanisms kept parallel environments aligned during transition periods, preventing data inconsistencies that could corrupt application state. Contemporary research examining downtime elimination techniques validated this parallel approach, particularly highlighting its superiority over in-place modification strategies [8]. Automated abort procedures stood ready to reverse transitions instantly upon detecting anomalies, prioritizing service stability over optimization completion.

4.3 Orchestrated execution chains for resource adjustments

Complex workflow engines coordinated the numerous tasks required to transform optimization recommendations into running infrastructure. Execution chains processed proposed changes through validation gates, safety checks, and implementation stages with precise timing control. Pre-execution validators confirmed environmental readiness, checking available capacity, network connectivity, and dependent service availability. Resource manipulation occurred through atomic operations that either completed fully or rolled back cleanly, preventing partial states that could destabilize systems. Workflow branches accommodated different infrastructure types, recognizing that container orchestration required different procedures than virtual machine resizing. Change tracking systems record every modification for audit purposes while providing real-time visibility into optimization progress. Recent investigations into machine learning applications for deployment automation influenced workflow design, particularly adaptive decision points that adjusted execution based on observed conditions [7]. Detailed instrumentation throughout pipelines generated comprehensive datasets for continuous improvement analysis.

4.4 Preventive actions for failures caused by optimization

Multi-layered defense strategies guarded against potential negative impacts from automated resource adjustments. Performance benchmarks established baseline expectations, with continuous monitoring detecting deviations that might indicate optimization problems. Progressive exposure techniques limited the blast radius by applying changes incrementally rather than system-wide transformations. Infrastructure snapshots capture complete system states before modifications, enabling point-in-time recovery if optimizations cause unexpected behaviors. Automated triggers monitored dozens of health indicators, initiating rollbacks when metrics breached predetermined boundaries. Manual override capabilities remained accessible to operations teams, recognizing that human judgment sometimes detected issues that automated systems missed. Recovery procedures underwent regular testing through simulated failure scenarios, ensuring teams could execute rollbacks efficiently under pressure. Stakeholder notification systems broadcast optimization activities and status updates, maintaining transparency about infrastructure changes that might affect service consumers.

5. RESULTS AND IMPACT ANALYSIS

5.1 Financial outcomes from resource optimization initiative

Comprehensive tracking revealed significant monetary benefits after deploying the automated optimization system across cloud infrastructure. Expense reductions materialized through multiple channels: smaller compute instances replaced oversized predecessors (achieving 35-45% cost reduction per instance), unutilized storage vanished from monthly bills (eliminating approximately 60% of orphaned volumes), and forgotten test environments ceased consuming budget allocations (recovering 20-25% of non-production spend). Accounting teams observed predictable spending patterns emerging where chaos previously reigned, simplifying budget forecasting and departmental chargebacks. Cost tracking granularity improved dramatically, with optimization actions linked directly to specific savings amounts. Expenditure trends shifted from upward spirals to controlled plateaus, then gradual declines as algorithms refined their understanding of workload requirements. Investment payback calculations incorporated both hard savings and soft benefits like reduced administrative overhead. Financial validation techniques borrowed from established performance engineering practices ensured measurement accuracy and eliminated attribution errors [9]. Stakeholder presentations could finally demonstrate tangible returns from infrastructure modernization efforts, strengthening support for continued automation investments.

5.2 Operational efficiency gains across computing infrastructure

Infrastructure metrics painted a clear picture of enhanced resource utilization following optimization deployment. Processing power consumption aligned more closely with actual computational needs, with CPU utilization improving from 15-20% to 65-75% across the fleet, eliminating the wasteful practice of running powerful machines for lightweight tasks. Memory allocation efficiency jumped noticeably (from 30% average utilization to 70-80%) as right-sizing eliminated generous buffers that applications never touched. Storage systems achieved higher density ratios (40% improvement in GB per dollar), accommodating more data within existing capacity through intelligent tiering and compression. Application responsiveness remained consistent despite running on leaner infrastructure, disproving assumptions that overprovisioning was necessary for acceptable performance. Heat maps of resource usage transformed from scattered red zones to uniform green distributions, indicating balanced utilization across the infrastructure fleet.

Efficiency measurement approaches drew from proven scalability engineering methodologies, ensuring metrics reflected genuine improvements rather than statistical artifacts [9]. Infrastructure teams celebrated achieving approximately 50% more computational output per dollar spent, a key indicator of platform maturity.

Table 4. Optimization Impact Metrics [9, 10]

Metric Category	Key Indicators	Pre-Optimization State	Post-Optimization State	Improvement
Resource Utilization	CPU, Memory, Storage	Scattered, inconsistent	Balanced, optimal	Significant
Operational Efficiency	MTTR, Change Success Rate	Manual-dependent	Automated	Enhanced
Cost Efficiency	Cost per transaction	Variable	Predictable	Reduced
System Performance	Response time, Throughput	Overprovisioned	Right-sized	Maintained

5.3 Workforce liberation through intelligent automation

Manual capacity planning consumed enormous engineering hours before automation transformed these tedious workflows. Technical staff previously trapped in spreadsheet analysis cycles found themselves freed for creative problem-solving and innovation projects. Alert fatigue decreased as predictive optimization prevented resource exhaustion before triggering emergency pages. Configuration drift became extinct as automated systems continuously reconciled actual deployments with optimal specifications. Audit compliance is simplified through automatically generated documentation trails that satisfy regulatory requirements without human intervention. These workforce improvements mirrored efficiency gains documented in other intelligent automation deployments, where technology amplifies human capabilities rather than replacing them [10]. Knowledge transfer accelerated as optimization patterns became codified in algorithms rather than residing in individual expertise. Team morale improved measurably as engineers focused on challenging technical problems instead of repetitive optimization tasks.

5.4 Expansion potential across diverse technical landscapes

Framework architecture proved remarkably adaptable when applied to varied infrastructure scenarios beyond the initial deployment scope. Processing capacity scaled linearly with infrastructure growth, handling enterprise-wide optimization without performance degradation. Multi-cloud deployments benefited from unified optimization logic that abstracted provider differences while respecting platform-specific constraints. International deployments succeeded despite latency challenges, with regional optimization clusters maintaining performance while respecting data residency laws. Validation procedures confirmed scalability claims through rigorous testing protocols established for distributed system engineering [9]. Industry-specific modifications developed naturally as many sectors used the framework to address their particular problems. The technique proved to be beneficial for both small startups and major corporations, demonstrating its universal application across organizational scales.

CONCLUSION

Automated right-sizing frameworks signal a shift in consciousness when it comes to cloud infrastructure management, moving from reactive cost-containment to proactive resource optimization. The successful deployment showed that smart automation can reconcile the ongoing disparities between provisioned resources and consumptive utilization while improving operational excellence. By utilizing constant monitoring, advanced pattern recognition, and automated remediation pipelines, organizations can successfully reduce costs while also maintaining the performance and reliability of applications. Introducing FinOps to platform engineering can result in sustainable optimization cycles as workloads evolve. Automated optimization not only delivers immediate improvements to the bottom line but also releases technical teams and personnel from repetitive manual tasks and routines, focusing instead on strategic work that can drive business

value. The framework proved its fit for enterprise-scale use because it is effectively suitable for varied environments, ranging from development clusters to global, enterprise-wide production deployments. As organizations continue to increase their use of the cloud, automated resource optimization will not only be a good idea, but essential to enjoy the competitive advantage of lower costs. Machine learning and predictive analytics developments will unlock even more optimizations in the near future, and we believe these accomplishments are merely the end of the first phase in the journey towards autonomous management of cloud infrastructure. Organizations that adopt these automated right-sizing frameworks may find themselves at the forefront of operational excellence in the public cloud market and set a new bar for responsible consumption of cloud resources.

REFERENCES

1. Ehsan Arianyan, et al., Efficient Resource Allocation in Cloud Data Centers Through Genetic Algorithm, in 6th International Symposium on Telecommunications (IST), March 21, 2013. https://ieeexplore.ieee.org/document/6483053
2. Ryuichi Sakamoto, et al., Production Hardware Overprovisioning: Real-World Performance Optimization Using an Extensible Power-Aware Resource Management Framework, in 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), July 3, 2017. https://ieeexplore.ieee.org/document/7967186/references#references
3. Alfonso San Miguel Sánchez and Danny Obando García, Efficient Cloud FinOps: A Practical Guide to Cloud Financial Management and Optimization with AWS, Azure, and GCP, Packt Publishing via IEEE Xplore, 2024. https://ieeexplore.ieee.org/book/10769266
4. Maulik Soni, FinOps Handbook for Microsoft Azure: Empowering Teams to Optimize Their Azure Cloud Spend with FinOps Best Practices, Packt Publishing via IEEE Xplore, 2023. https://ieeexplore.ieee.org/book/10251281
5. Russ McKendrick, Infrastructure as Code for Beginners: Deploy and Manage Your Cloud-Based Services with Terraform and Ansible, Packt Publishing via IEEE Xplore, 2023. https://ieeexplore.ieee.org/book/10251335
6. Dev Corner, Monitoring with Prometheus and Grafana: A Comprehensive Guide, DEV Community, March 8, 2023. https://dev.to/devcorner/monitoring-with-prometheus-and-grafana-a-comprehensive-guide-48gf
7. Shadow Pritchard, et al., Automating Staged Rollout with Reinforcement Learning, in 2022 IEEE/ACM 44th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), June 13, 2022. https://ieeexplore.ieee.org/document/9793526
8. Chaitanya K. Rudrabhatla, Comparison of Zero Downtime Based Deployment Techniques in Public Cloud Infrastructure, in 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), November 10, 2020. https://ieeexplore.ieee.org/document/9243605
9. Sergei Shudler, et al., Engineering Algorithms for Scalability through Continuous Validation of Performance Expectations, in IEEE Transactions on Parallel and Distributed Systems, 2019. . https://ieeexplore.ieee.org/document/8632716
10. Qin Yan and Xiaofan Tu, Home Smart Energy Management System for Optimized Electricity Cost Reduction Using Photovoltaic-Powered EV Charging Station, in 2022 7th Asia Conference on Power and Electrical Engineering (ACPEE), June 1, 2022. https://ieeexplore.ieee.org/document/9783636