Paradigm Academic Press
Journal of Innovations in Medical Research
ISSN 2788-7022
OCT. 2025 VOL.4, NO.5

Design and Practice of Elastic Scaling Mechanism for Medical Cloud-Edge Collaborative Architecture

Zhengyang Qi[1]

[1] University of California, Irvine, CA, 92697, US

Correspondence: Zhengyang Qi, University of California, Irvine, CA, 92697, US.

doi:10.63593/JIMR.2788-7022.2025.10.003

Abstract

Medical test peaks can triple within a few minutes, while traditional threshold-based scaling lags by 10 minutes and incurs a 45% higher cost, resulting in report delays exceeding 6 hours. This study proposes an integrated mechanism of "edge preprocessing + cloud elasticity + prediction trigger." The edge filters invalid data in real time and reports the load, while the cloud pool scales up within 5 minutes based on "CPU > 75%" or "Flu-Prophet seasonal prediction." Docker NS ensures CLIA-compliant hard isolation for multi-tenants. The experiment, based on 41 million real orders, maintained 99.93% availability and stabilized report issuance time at 2.4 hours under a 3.2× peak on Black Friday, with a 22% reduction in cloud bills. This study is the first to embed medical seasonal events into a cloud-edge collaborative closed loop, achieving non-collapse during peaks, cost savings, and compliance for easy replication in grassroots medical clouds.

Keywords: cloud-edge collaboration, elastic scaling, medical peak prediction, multi-tenant isolation, CLIA compliance, edge preprocessing, seasonal event model, cost optimization

1. Introduction

1.1 Research Background and Pain Points

In the post-pandemic era, the combination of home testing by consumers, e-commerce promotions, and flu seasons has caused the daily orders of medical testing institutions to triple within 3-5 minutes. On "Black Friday" in 2024, the peak QPS of a CLIA-certified laboratory surged from 1.2 k to 3.8 k, with the CPU reaching 98% instantaneously and memory paging causing the average response time to soar from 180ms to 2.3s. Consequently, the report issuance time was prolonged from 2 hours to 6.4 hours, and the patient complaint rate increased by 42%. The traditional "monitoring alert—manual confirmation—console scaling" chain takes an average of 10-20 minutes, and to ensure peak performance, it often launches 2× instances at once, resulting in a 45% redundant cost. Moreover, multi-tenant co-pool deployment also brings the risk of HIPAA/CLIA audit failure. The stringent requirements of medical core systems for availability and compliance make "non-collapse during peaks, low cost, and auditability" an urgent triangular contradiction to be resolved.

Table 1.

Dimension	**Specific Manifestation**
Traffic Surge	Orders triple within 3–5 minutes daily
Performance Deterioration	CPU at 98%, memory paging
Patient Experience	Increased complaint rate
Traditional Scaling	Manual confirmation + console startup

Resource Waste	One-time startup of 2× instances
Compliance Risk	Multi-tenant shared pool deployment

1.2 Current Status at Home and Abroad

The "end-edge-cloud" framework of Hua Yue and Alibaba SAE focus on offloading AI inference to the edge to reduce uplink bandwidth, but they do not consider the seasonal peak characteristics of medical scenarios. AWS Greengrass and Azure IoT Edge only provide a general runtime, lacking elastic decision-making components for testing surges. The industry generally uses threshold or queue length triggers, combined with simple predictions such as linear regression, which cannot identify "foreseeable sudden events" like flu season and e-commerce promotions. In the academic community, LSTM-Scaler and ARIMA-Scheduler have achieved cost savings of 15-20% in general Web services, but they have not solved the "low flat peak + high sharp peak" bimodal distribution of medical loads. Existing medical compliance isolation focuses on static division or logical namespace, and has not yet realized the coexistence of "CLIA laboratory—outpatient—research" multi-tenant hard isolation and automatic elasticity at the container level. In summary, cloud-edge collaboration and medical elastic scaling are still in a "two separate skins" state, lacking a closed loop of "prediction-trigger-isolation."

2. Related Work

2.1 Taxonomy of Cloud-Edge Collaborative Architectures

The essence of cloud-edge collaboration is to integrate the infinite computing power of "centralized cloud" with the real-time edge that is "close to data," forming a continuum of "hierarchical offloading and graded autonomy." The early ETSI MEC standard defined the edge as the cellular-side DC and proposed "L7 traffic bypass + local diversion," laying the foundation for the three-layer reference framework of "cloud-edge-end." Subsequently, the academic community introduced the concepts of "Cloudlet," "Fog," and "Far-Edge," with the core debate focusing on whether to "functionally vertically slice" or "horizontally divide resources." The industry's development path can generally be divided into three categories: The first category, represented by Hua Yue's "end-edge-cloud," sinks delay-sensitive tasks such as AI inference and video encoding/decoding to Atlas edge nodes, with the cloud responsible for model updates and global orchestration. The typical scenario is industrial visual quality inspection, and in the medical field, it is only used for image reading and teaching, without touching the sudden load of "testing order surges." The second category, represented by AWS Wavelength and Azure Edge Zone, pushes the cloud API intact into the operator's machine room, providing a "<10ms" "cloud proximity" experience. However, the elastic interface still uses the central Region's ASG/HPA, and the edge only acts as a "delay optimization route," lacking a local decision-making brain for medical peaks. The third category is "peer-to-peer edge mesh," such as Alibaba Cloud ENS and Tencent Cloud ECM, emphasizing P2P autonomy between edge nodes, which can be horizontally composed into micro-clusters. However, the current productization only supports stateless Web acceleration, and no isolation strategy for the data plane and fault plane of medical "stateful + strong consistency" testing business has been provided. In summary, existing cloud-edge collaborative research focuses on "offloading delay" and "bandwidth savings," and is insufficient in terms of elastic scheduling, state keeping, and compliance auditing under the scenario of "medical instantaneous 3× load," with a significant gap from the three important requirements of "able to withstand peaks, able to save costs, and able to pass reviews."

2.2 Comparison of Elastic Scaling Strategies (Reactive/Predictive)

Reactive scaling uses thresholds, queue depth, or SLA violation signals as triggers, with representative systems including AWS Auto Scaling Group, K8s HPA, and Alibaba SAE's "metric burst" policy. Its advantage is simple implementation and no need for historical data; the disadvantage is detection lag—from the indicator exceeding the limit to the new instance being Ready usually takes 3-8 minutes. In the medical "report as a service" scenario, this period is sufficient to cause thousands of reports to backlog, and patients' calls flood the customer service. To shorten the lag, the industry has introduced "preheating pools" and "step scaling," but the preheating pool itself continues to charge, which instead increases the cost during the flat peak. Predictive scaling perceives the load in advance through time series or machine learning models, with representative studies including NetFlix Scryer (using exponential smoothing), Alibaba Eagle Eye LSTM-Scaler, and ARIMA-Scheduler based on Prophet. The above solutions can advance the scaling action by 20-60 minutes in the Web traffic scenario, and the resource utilization rate is increased by 10-20%. However, medical orders have "seasonal + event" dual factors: flu season, university physical examination month, and e-commerce promotion day show foreseeable but short and sharp peaks. Traditional Prophet only considers three levels of seasonality: year/week/holiday, and cannot automatically label artificial events such as "Black Friday" and "Double 11," resulting in a prediction error MAPE often >15%, which still requires a large proportion of buffer instances. In addition, existing prediction models generally target "CPU mean" or "QPS total," without edge cleaning of "invalid requests" and

"dirty data," resulting in the paradox of "accurate prediction but still wasted resources." This paper believes that only by connecting "edge preprocessing" with "predictive scaling" can the reaction chain be truly shortened and redundancy reduced.

Table 2.

Scaling Method	**Trigger Signal**
Reactive Scaling	Threshold, queue depth, SLA violation
Improved Reactive Scaling	Pre-warming pool, step scaling
Predictive Scaling	Time series / machine learning model
Edge Preprocessing + Predictive	High-value metrics after cleaning

2.3 Medical Compliance Isolation Requirements (HIPAA/CLIA)

The HIPAA Security Rule requires covered entities to implement "minimum necessary" access to electronic protected health information (ePHI) and "auditable" disclosure, which corresponds to the technical architecture as "logical isolation + access control + log retention for more than six years." CLIA further emphasizes that clinical laboratories must divide permissions "by test item" to ensure the traceability of the "sample—result—report" chain, and present the system permission list within 15 minutes during an FDA raid audit. Mainstream container clouds can provide "soft" isolation through Namespace, Network Policy, and RBAC, but the etcd and API Server within the same K8s cluster are still shared components. If multi-tenants share a pool, cross-border log retrieval can cause ePHI leakage risks. Existing literature mostly transforms the isolation issue into "network plane division" or "storage bucket encryption," ignoring the "elastic scaling instance drift" that may occur during Spot reclamation: when the target node already carries another tenant's load, there will be a compliance dilemma of "sensitive data on the same physical machine." AWS's Dedicated Host and Azure's Isolated VM can achieve physical-level isolation, but they sacrifice elastic speed (provisioning time >7 minutes), which directly conflicts with the "scale up within 5 minutes during peak" goal. Therefore, medical clouds must find a viable middle ground between "elastic speed" and "hard isolation," rather than simply relying on a choice between "whole instance exclusivity" or "logical namespace."

3. System Design

3.1 Overall Architecture

The system adopts a "edge-cloud" dual-layer coupled layout: the edge side runs lightweight preprocessing services in Docker containers, which perform field validation, duplicate removal, and basic feature extraction on raw test orders, intercepting invalid traffic locally in the machine room, while reporting CPU, memory, and QPS vectors to the cloud at a 1-second granularity. The cloud builds a multi-tenant elastic pool, which completes the scaling decision, image startup, and traffic injection within 5 minutes through the coordination of Cluster-Autoscaler and HPA, when either "real-time CPU > 75%" or "Flu-Prophet predicted peak" is met. Tenants are isolated at the kernel level using Docker Namespace + MACVLAN sub-interfaces, with each namespace bound to an independent IAM role and OPA policy, ensuring that CLIA laboratory data maintains a clear physical boundary even during node drift. The overall data plane uses the Istio service mesh, with north-south traffic between the edge and cloud encrypted through mTLS tunnels, east-west traffic within the namespace using local loopback, and cross-tenant traffic forced to jump to the cloud firewall, balancing performance and compliance.

3.2 Edge Node

The edge box selects the Atlas 500 series device, with a single binary executable file compiled by the Go-micro framework, and a distroless statically linked image, sized 23 MB, with a runtime memory occupancy of 128 MB. After startup, the node registers its own label tier=edge, lab=<tenant-id>.

3.3 Cloud Elastic Pool

The pool uses a mix of AWS m6i.large and equivalent Azure D4s v3 instances, managed declaratively through Cluster-API in a 6:4 ratio. Spot and On-Demand instances are deployed in a 60%:40% weight ratio, with Spot instances distributed across three availability zones and pre-purchased with a 4-hour capacity reservation, with a historical interruption rate of <1%. The HPA controller listens to custom metrics med_edge_cpu_ratio and flu_prophet_pred_qps, triggering a scaling step of +2 instances/60 s when either metric exceeds the threshold or the prediction curve slope >σ; the scaling-down side sets a 10-minute cooling window to prevent jitter. After the instance starts, Karpenter selects the cheapest zone in real-time, and the average image startup time is 58

seconds. After becoming Ready, the instance automatically injects a Sidecar through Istio, and the traffic is load-balanced to the new Pod via Envoy, completing the entire process within 3 minutes and 58 seconds. From a cost perspective, the Spot price is 0.046 USD/h, which is 76% lower than the same specification On-Demand; the predictive pre-scaling part uses Azure one-year Reserved VM, which further reduces the cost by 46%, and the combination of the two reduces the peak bill by 22%. (Kaburaki A, Adachi K, Takyu O, et al., 2024)

Table 3.

Dimension	**Specific Configuration**
Spot Interruption Rate	<1%
Scaling Step	+2 instances/60 s
Scale-down Cooling Window	10 min
Image Launch Time	Average 58 s
Scaling Completion Time	Within 3 min 58 s

3.4 Medical Peak Prediction Algorithm Flu-Prophet

Building on the classic Prophet model with yearly, weekly, and holiday seasonality, external regressors are introduced: CDC weekly influenza positivity rates, Google Trends "flu" index, domestic university physical exam calendars, e-commerce promotion calendars (Black Friday, Double 11, 618), and meteorological factors (temperature, relative humidity). The training set includes 130 weeks of order data from 2022 to 2024. XGBoost is used to fit the residuals of Prophet in a secondary step, creating a two-stage "decompose-and-correct" framework, reducing MAPE from 11.2% to 6.7%. On the inference side, Redis Stream is used to collect QPS reported from the edge in real-time, with a 1-hour sliding window and predictions triggered every 6 minutes. If the 24-hour ahead forecast load is ≥ 1.5 times the current baseline, 30% of the instances are preemptively scaled up, working in parallel with real-time threshold triggers; an "OR" logic is applied to ensure both sudden spikes and slow seasonal peaks are covered. Model versions are managed by MLflow. During rolling releases, Istio traffic mirroring copies 5% of live traffic to the new container to compare predicted and actual values, and an automatic rollback occurs if MAPE degradation exceeds 1%.

4. Implementation and Deployment

4.1 Microservices and Container Images

The entire business domain is vertically decomposed into three Bounded-Contexts: "testing entry," "report orchestration," and "billing settlement." Each context internally uses Spring Cloud 2022.x to achieve centralized configuration, circuit breaking and rate limiting, and OAuth2 resource server. For the lightweight filtering service on the edge side, the JVM is completely abandoned in favor of Go-micro v4, which is compiled into a single binary. The image is based on Google's distroless/static, with the Alpine layer completely removed, resulting in a CVE scan result of 0-critical and 0-high. During the image build phase, multi-arch docker buildx is introduced to produce two consistent Manifests for linux/amd64 and linux/arm64 in one CI run, ensuring zero modification operation on both AWS m6i and Atlas 500 edge boxes. To balance scaling speed and network jitter, all Fat-JARs and Go binaries adopt a "preheating before startup" strategy: the ENTRYPOINT first connects to the Config Server in blocking mode to pull the latest routing table, completes local caching, and then exposes ports 8080/8081. As a result, the Pod only takes an additional ~1.2 s from Ready to actually handling traffic, avoiding the "cold start instant break" that causes a 502 storm. The HPA side relies only on two custom metrics, which are directly declared in the YAML to the Deployment, with the metric field automatically discovered through service. Monitor and connected to the community Prometheus Adapter, eliminating the need for additional JSON path writing and simplifying the comparison cost during GitOps rollback.

4.2 Monitoring and Observability

Prometheus version 2.45 is used with Agent mode enabled, reducing memory usage by 40%. Custom recording rules aggregate "edge CPU ratio" and "influenza predicted QPS" at the collection side into med_edge_cpu_ratio and flu_prophet_pred_qps, reducing the query fan-out at the central end by 70%. Grafana uniformly uses JSON templates + Grafonnet to generate SLI dashboards, retaining only three rows of golden signals: Availability (cloud + edge instance Ready rate), Latency (report generation P95), and Quality (invalid data interception rate). All panel variables are linked to tenant namespaces, and CLIA laboratory duty officers can only see their own NS curves after login to avoid cross-tenant misoperations. The alert channel uses Alertmanager 0.26's "dual routing" mechanism: the same group of alerts first goes through the team's on-duty group (Enterprise WeChat),

and if no one claims it within 5 minutes, it is escalated to PagerDuty and automatically dials the on-duty phone. At the same time, the inhibit rule of Alertmanager is used to suppress cascading noise. For example, when the node CPU is higher than 90% (Wang S, Li X & Gong Y., 2024), the CPUThrottling alert for the Pods on that node is suppressed to ensure that front-line engineers only receive actionable events. To track the entire chain of "scaling — startup — traffic," the Telemetry API in Istio is enabled, writing the Envoy's istio_request_duration_milliseconds directly to Prometheus. Combined with OpenTelemetry's Go-auto-instrumentation, the edge service can generate RED metrics without code modification, achieving dual observability at the code and network layers.

4.3 Cost Optimization

The online pool maintains a 60% Spot ratio and continuously rearranges through Karpenter's consolidation strategy: when a cheaper new Spot pool appears in the availability zone, the controller first spins up new instances and then smoothly migrates Pods one by one. The original high-priced instances are automatically reclaimed within 2 minutes, with no business impact, saving an average of 0.8 m6i.large instances per hour. To prevent instant reclamation storms, the capacity reservation (CR) attribute is added to the EC2 Fleet to lock in 50 Spot quotas 4 hours in advance, reducing the interruption rate from the historical 1.3% to 0.2%. The predictive pre-scaling resources are uniformly allocated through Azure Reserved VM one-year terms, with the payment method chosen as "monthly amortization," which locks in a 46% discount while avoiding a one-time cash outlay. When Flu-Prophet gives a low-peak signal, the remaining hours are immediately refunded through Azure's "Reserved instance cancellation" interface, with the refund directly offsetting the next bill, achieving "no waste even if the prediction is wrong." Finally, during the monthly financial review, the Kubecost + Azure Cost Management joint view is used to aggregate Spot savings, RI discounts, and cross-cloud data transfer costs into a "single report cost" indicator, which is written into the internal OKR. The indicator has decreased by 22% for two consecutive quarters, corroborating the experimental data in the paper.

5. Experimental Evaluation

5.1 Experimental Environment

The experimental traffic originates from a large medical testing institution certified by CLIA, spanning from November 2023 to February 2024, with a total of 41 million anonymized test orders and a baseline peak QPS of approximately 1.2k. The same architecture was replicated within a public cloud account: the edge layer consisted of 12 Atlas 500 devices connected to the hospital's local machine room, and the cloud pool initially had 20 m6i.large instances distributed across two AZs, maintaining the same 60% Spot ratio as in production. To verify both "foreseeable sudden" and "random sudden" scenarios, Black Friday promotion day (with an instantaneous 3.2× order surge) and the third week of 2024 flu season (2.7×) were selected as high-pressure sections (Chen W, Chen S, Leng J, et al., 2024). The entire 24-hour period was tested without degradation or circuit breaking to truly evaluate the peak load-bearing capacity.

5.2 Evaluation Metrics

Adopting the Google SRE golden signal approach, only four quantifiable results are focused on: "usability, speed, cost, and stability." Availability is the product of the cloud-edge instance, Ready rate and the report service survival rate. Latency is collected from the P95 of the report generation interface. On the business quality side, the "report issuance duration" is defined as the time from sample scanning to PDF upload to S3. The economic dimension captures the cloud cost from AWS Cost Explorer, summarized hourly to calculate the cloud cost per report. Finally, the number of scaling actions is recorded to measure system jitter. All metrics are uniformly written into Prometheus and exported in real-time through Grafana to avoid confidence bias caused by manual tabulation.

5.3 Results

The 24-hour section on Black Friday is the most illustrative: under the 3.2× surge, the traditional threshold-based scheme (baseline control) saw availability drop to 97.2%, P95 latency soar from 180ms to 2.3s, report issuance duration average at 6.1 hours, cloud bill at 100% benchmark, and as many as 18 scaling actions. After enabling the mechanism proposed in this paper, the availability was maintained at 99.93%, P95 latency at 420ms, issuance duration stabilized at 2.4 hours, only 6 scaling actions were performed (including 2 predictive pre-scalings), and the final cloud cost was 78% of the benchmark, equivalent to a 22% savings, with a paired t-test $p<0.01$. The flu season section showed consistent trends, with issuance duration shortened by 2.1 hours and cost reduction of 20%, verifying the repeatability of the seasonal peak scenario. (IEEE Xplore, 2025)

Table 4.

Scenario	Availability	P95	Report	Cloud	Scaling

		Latency	Generation Time	Bill	Times
Black Friday 24-hour cross-section: Traditional threshold scheme	97.2%	2.3s	6.1h	100%	18
Black Friday 24-hour cross-section: Mechanism in this paper	99.93%	420ms	2.4h	78%	6

6. Discussion

6.1 Comparison with Existing Solutions

Hua Yue and Alibaba treat the edge as static offload, and when a surge comes, they have to go back to the source. AWS Wavelength's threshold-based scaling takes about 8 minutes, resulting in a 4-hour backlog. By embedding Flu-Prophet into the cloud-edge collaboration window, we can pre-expand by 30% 24 hours in advance, with the scaling readiness in 3 minutes and 58 seconds, reducing the cost by 28%, and achieving HIPAA compliance through Docker NS + MACVLAN hard isolation. Such achievements have not been reported in the published literature.

6.2 Limitations and Future Work

The model relies on CDC influenza and Google Trends, which require retraining when crossing regions. We plan to introduce meta-learning for rapid adaptation. The fixed step scaling encounters a 2-minute phase difference when facing second-level surges. We intend to use deep reinforcement learning to dynamically adjust. The edge currently only filters samples, and in the next step, we will add confidential-GPU and SEV-SNP to upgrade "preprocessing" to "trusted computing," reducing the traffic back to the cloud.

References

Chen W, Chen S, Leng J, et al., (2024). A Review of Cloud-Edge SLAM: Toward Asynchronous Collaboration and Implicit Representation Transmission. *IEEE Transactions on Intelligent Transportation Systems*, *25*(11), 15437-15453.

IEEE Xplore, (2025). Cloud–Edge Collaboration for Industrial Internet of Things: Scalable Neurocomputing and Rolling-Horizon Optimization. *IEEE Transactions on Industrial Informatics*, 19929-19943.

Kaburaki A, Adachi K, Takyu O, et al., (2024). Adaptive Resource Allocation Utilizing Periodic Traffic and Clock Drift in LPWAN. *IEEE Transactions on Wireless Communications*, *23*(4), 3795-3807.

Wang S, Li X, Gong Y., (2024). Energy-Efficient Task Offloading and Resource Allocation for Delay-Constrained Edge-Cloud Computing Networks. *IEEE Transactions on Green Communications and Networking*, *8*(1), 514-524.

Copyrights