

Evaluating an energy aware cloud native microservices architecture to reduce cost barriers in resource limited healthcare systems

Tia Haddad¹

Kingston University

Souheil Khaddaj¹

Kingston University

Sarah Barman¹

Kingston University

Simon de Lusignan²

University of Oxford

Pushpa Kumarapeli¹

p.kumarapeli@kingston.ac.uk

Kingston University

Research Article

Keywords: health information systems, energy efficiency, universal health coverage, resource-limited settings, cloud-native microservices, Kubernetes

Posted Date: October 15th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10826860/v1>

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

[Read Full License](#)

Additional Declarations: No competing interests reported.

Evaluating an energy aware cloud native microservices architecture to reduce cost barriers in resource limited healthcare systems

Tia Haddad¹; Souheil Khaddaj¹; Sarah Barman¹; Simon de Lusignan²; Pushpa Kumarapeli^{1*}

¹School of Computer Science and Mathematics, Kingston University London, Penrhyn Road, Kingston upon Thames, KT1 2EE, United Kingdom ²Nuffield Department of Primary Care Health Sciences, University of Oxford, Woodstock Road, Oxford, OX2 6GG, United Kingdom

*Corresponding Author: Pushpa Kumarapeli School of Computer Science and Mathematics Kingston University London Penrhyn Road Kingston upon Thames, KT1 2EE United Kingdom Email: p.kumarapeli@kingston.ac.uk

ABSTRACT

Purpose: Cloud-native microservices architectures offer the scalability and resilience that health information systems require to support Universal Health Coverage, yet their operational compute costs remain a significant adoption barrier in resource-limited settings, where conventional autoscaling mechanisms scale resources unchecked during processing surges, leading to over-provisioning and prohibitive running costs. This study presents a formative proof-of-concept evaluation of an energy-aware autoscaling framework that introduces cost-control into Kubernetes-orchestrated healthcare microservices.

Methods: A containerised cardiovascular disease risk processing system, comprising data cleansing and diagnostic risk calculation microservices, was deployed on a single-node Minikube environment to approximate hardware constraints typical of resource-limited facilities. The framework integrates real-time energy telemetry from KEPLER, aggregated via Prometheus, into KEDA-governed scaling decisions through a threshold-based cost-control mechanism parameterised by processing function. The system was stress-tested across four progressive workload scenarios (10,000 to 50,000 clinical records; 15 to 30 concurrent requests) mapped to facility-to-district-level healthcare operations and compared against conventional horizontal pod autoscaling.

Results: The energy-aware autoscaler achieved post-scaling energy reductions of 12.1% to 33.4% across all scenarios while maintaining error-free processing and stable memory utilisation. CPU utilisation increased only as necessary and returned to baseline after processing. Conventional autoscaling produced uncontrolled resource growth and latency degradation under equivalent conditions. Cost projections using published cloud pricing for Sub-Saharan Africa and South Asia estimate annual savings of approximately US\$1,340 to US\$1,521 for a 10-node district deployment under surge conditions, a meaningful proportion of LMIC digital health operating budgets.

Conclusion: Embedding energy-aware cost control into cloud-native health information architectures can meaningfully reduce the operational expense barriers currently limiting

adoption in resource-constrained settings, supporting the equitable deployment of digital health infrastructure essential for achieving UHC goals.

Keywords: health information systems; energy efficiency; universal health coverage; resource-limited settings; cloud-native microservices; Kubernetes

INTRODUCTION

Background

Universal Health Coverage (UHC), a central target of Sustainable Development Goal 3 (SDG-3), depends on the availability of scalable, resilient, and interoperable health information systems (HIS) to support equitable healthcare delivery, data-driven decision-making, and population-level health management [1-3]. Digital health technologies play a critical role in health system strengthening by improving access to care, supporting service continuity, and enabling the efficient processing of clinical and administrative data [4,5]. As the World Bank has emphasized, while there is tremendous promise in leveraging technology for UHC, effective deployment requires context-specific strategies with ample flexibility to adapt as needs and opportunities evolve [6].

Despite this promise, the adoption and sustainability of digital health systems in resource-limited settings remain constrained by well-documented barriers. Systematic reviews of UHC implementation challenges have consistently identified financial limitations, inadequate infrastructure, workforce shortages, and governance deficiencies as the dominant obstacles, particularly in low- and middle-income countries (LMICs) [7-9]. At the information systems level, studies have identified unreliable infrastructure, complex system design, high startup and maintenance costs, and limited technical capacity as recurring barriers to HIS advancement [9]. Critically, many of these barriers are not confined to initial deployment; the ongoing operational costs of maintaining digital health infrastructure frequently exceed the capacity of health facilities operating under severe budget constraints [10]. When external funding cycles end, systems that cannot sustain themselves financially are abandoned, undermining the long-term health system goals they were designed to support [9,11].

These financial pressures extend directly into the domain of health information architecture. Digital health solutions must fit within coherent digital health information architectures and the UHC agenda, requiring partnerships between developers, implementers, policymakers, and funders to ensure that the systems built are not only functional but financially sustainable to operate [10]. The cost of cloud infrastructure, in particular, is not merely a technical concern but a health financing concern: if the digital systems that support UHC are themselves financially unsustainable to operate, they undermine the very goals they are intended to serve.

Cloud-Native Microservices for Healthcare

In response to the scalability and modularity demands of modern health information systems, cloud-native microservices architectures (MSAs) have emerged as an increasingly adopted paradigm. MSA decomposes complex applications into independent, loosely coupled services that can be deployed, scaled, and managed individually, enhancing fault isolation, service resilience, and continuous deployment capabilities [12-14]. In healthcare, this architectural approach enables individual system components, such as data cleansing, risk calculation, or patient record management, to be updated, replaced, or scaled independently without affecting the broader application, reducing both cost and risk associated with system modifications [15,16].

The adoption of cloud-native MSA for national-scale health information systems is already under way. Zao et al [17] describe the design of a cloud-native National Digital Health Information Infrastructure for Malawi, developed to address data quality, security, and service fragmentation challenges within the country's national health strategy. Similarly, federated microservices frameworks integrating Kubernetes orchestration with privacy-preserving analytics have been proposed for processing healthcare data at scale across institutional boundaries [18]. These developments confirm that cloud-native MSA is increasingly recognized as an appropriate architectural foundation for health information systems in diverse settings, including LMICs.

However, deploying cloud-native microservices on cloud infrastructure introduces substantial operational overhead. Containerized environments managed through orchestration platforms such as Kubernetes generate cumulative resource consumption through container lifecycle management, interservice communication, and autoscaling mechanisms [19,20]. As healthcare data volumes grow, driven by electronic health records, clinical measurements, Internet of Medical Things devices, and population-level datasets [21,22], the computational demands placed on cloud-native architectures increase correspondingly. In resource-limited settings, where cloud compute costs are borne directly by health facility budgets or time-limited donor funding, this operational overhead can be prohibitive [9,10]. The risk is that MSA, while architecturally appropriate, recreates the very cost barrier that resource-limited health systems already face: systems that are technically capable but financially unsustainable to operate.

Resource Efficiency in Cloud-Native Systems

Cloud computing platforms provide elastic resource provisioning and on-demand scalability, making them well suited for processing large and variable healthcare datasets [23,24]. However, existing cloud resource management strategies have primarily focused on infrastructure-level optimisation, such as virtual machine consolidation or hardware utilisation, providing limited visibility into software-level resource consumption within containerised environments [23,25,26]. As cloud-native deployments scale, the energy consumption associated with container orchestration becomes an increasingly significant operational concern, with direct implications for compute costs [19,27].

Current autoscaling mechanisms for Kubernetes, including the widely used horizontal pod autoscaler (HPA), typically rely on performance-oriented metrics such as CPU utilisation, memory usage, or request throughput to trigger scaling decisions [24,28]. While effective at maintaining service performance, these mechanisms do not account for the energy or cost implications of scaling individual microservices under variable workloads. During workload surges characteristic of healthcare data processing, such as batch data cleansing, epidemiological reporting, or outbreak surveillance, HPA can scale resources unchecked, leading to over-provisioning and disproportionate compute expenditure without commensurate performance gains [24,29].

Recent work has begun to explore energy-aware and carbon-aware approaches to cloud autoscaling. Gebreweld [30] evaluated the energy consumption impact of carbon-aware autoscaling using Kubernetes Event-Driven Autoscaling (KEDA) but focused on low-priority, nonreal-time workloads rather than healthcare scenarios requiring dynamic responsiveness. Hanafy et al [31] proposed CarbonScaler, a carbon-aware autoscaler for batch applications leveraging workload elasticity, but similarly targeted batch processing rather than the variable, on-demand workloads characteristic of healthcare data systems. Cortellessa et al [32] explored refactoring strategies to balance performance, cost, and power consumption in microservices deployments, while De Nardin et al [33] proposed Elergy, a proactive elasticity model for optimizing energy use in CPU-bound applications. A recent systematic literature review of energy consumption in microservices architectures confirmed that while interest in the topic is growing, practical frameworks that integrate energy metrics into real-time autoscaling decisions for domain-specific workloads remain scarce [34].

Critically, none of these approaches address the specific challenge of managing energy and cost in cloud-native microservices processing large-scale, heterogeneous data under the operational constraints of resource-limited settings. The existing literature on cloud-native healthcare related architectures has focused primarily on functionality [15,16], security and trustworthiness [17], privacy preservation [18], and scalability [35,36], but not on whether resource-limited health facilities can afford to run these systems. This gap is significant: if cloud-native HISs are to support UHC goals, they must be designed with operational cost awareness built in from the outset, not optimised retrospectively.

Study Objective

This study addresses this gap by proposing and evaluating an energy-aware autoscaling approach for cloud-native healthcare microservices deployed on Kubernetes. By integrating real-time energy consumption metrics from the Kubernetes-based Efficient Power Level Exporter (KEPLER) into scaling decisions governed by KEDA, the proposed framework introduces a threshold-based cost-control mechanism that regulates microservice scaling behaviour based on energy consumption rather than performance metrics alone. The system is evaluated through a formative proof-of-concept assessment using progressive workload scenarios representative of healthcare data processing demands, with empirical energy reductions mapped to estimated operational cost savings using published cloud compute pricing. The findings are intended to inform the design of cost-viable cloud-native

health information systems capable of supporting UHC objectives in resource-constrained settings.

METHODS

System Architecture

To evaluate the proposed energy-aware autoscaling approach, a cloud-native microservices architecture (MSA) was developed for cardiovascular disease (CVD) risk data processing. The system was designed to reflect a representative healthcare data pipeline comprising ingestion, cleansing, and clinical computation stages, operations common to health information systems managing chronic disease surveillance and population health analytics.

The architecture deploys two distinct microservices: (1) the *CleanseService*, developed in Python, which performs automated data cleansing operations on raw clinical datasets, handling missing values, inconsistent record formats, and data quality anomalies typical of healthcare data collected in resource-limited settings; and (2) the *ChronicDiseaseManager*, developed in C#, which calculates the CVD risk rate for each patient based on the cleansed clinical variables. The processed outputs are stored in CockroachDB, a distributed NewSQL database selected for its resilience and horizontal scalability, creating a structured data foundation suitable for downstream analytical applications.

Both microservices were containerised as Docker images and deployed on Kubernetes, the industry-standard container orchestration platform. Interservice communication is managed via HTTP protocols through an Istio service mesh, which provides traffic management, load balancing, and observability capabilities through envoy proxy sidecars attached to each microservice pod. The service mesh pattern reduces unnecessary data transfers between services and ensures that workloads are efficiently distributed, while providing the real-time performance visibility required by the energy-aware autoscaling mechanism described in the following subsection.

The system was deployed on a single-node Minikube environment. This deployment choice was deliberate: Minikube approximates the constrained compute infrastructure available in resource-limited healthcare settings, where multinode cloud clusters may not be available or affordable. By evaluating the system under these constraints, the results reflect performance characteristics relevant to the target deployment context rather than idealised cloud conditions. A low-level design of the architecture is presented in Figure 1, with descriptions of essential tools and components outlined in Table 1.

Figure 1. Cloud-native microservices architecture for cardiovascular disease risk data processing, deployed on Kubernetes with Istio service mesh and energy-aware autoscaling components.

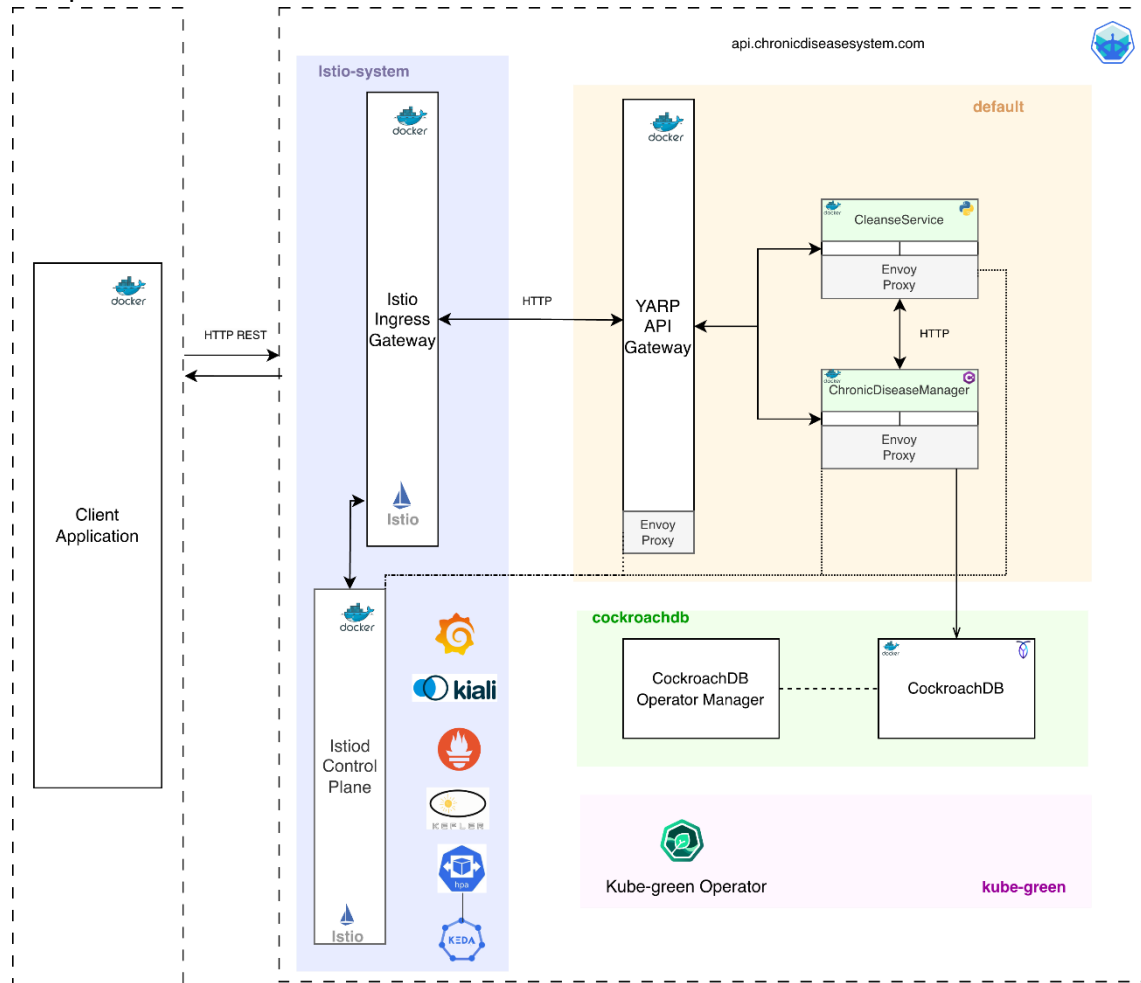


Table 1. Tools and Components Used in the System Architecture

Tool	Description
Kubernetes	Container orchestration platform providing automated deployment, scaling, and management of containerised microservices.
KEDA (Kubernetes Event-Driven Autoscaling)	Scales Kubernetes containers based on event-driven triggers, including external metrics such as energy consumption, extending beyond the capabilities of the native horizontal pod autoscaler.
KEPLER (Kubernetes-based Efficient Power Level Exporter)	Uses extended Berkeley Packet Filter (eBPF) technology to capture real-time energy consumption metrics at the container level, exporting them to Prometheus for monitoring and autoscaling integration [29].
Prometheus	Open-source monitoring and alerting toolkit that collects and stores time-series metrics, enabling real-time queries on energy consumption, CPU utilisation, and memory usage across microservice pods.
Grafana	Visualization platform used to create real-time dashboards from Prometheus and other data sources, providing operational visibility into energy consumption patterns during workload processing.
Istio	Service mesh platform providing traffic management, load balancing, and observability for interservice communication through envoy proxy sidecars.
CockroachDB	Distributed NewSQL database providing strong consistency, horizontal scalability, and fault tolerance for storing processed clinical data outputs.
ScaledObject CRD	Custom resource definition that specifies how KEDA should scale the application based on event triggers, including energy consumption thresholds.

Energy-Aware Autoscaling Mechanism

The core contribution of this study is the integration of real-time energy consumption metrics into Kubernetes autoscaling decisions, creating a cost-control mechanism that regulates microservice scaling behaviour based on energy consumption rather than performance metrics alone.

Monitoring Pipeline

The energy monitoring pipeline operates as follows. KEPLER, deployed as a DaemonSet on each Kubernetes node, uses eBPF probes to capture hardware-level energy consumption data at the container and pod level. These metrics are exported to Prometheus, which aggregates and stores them as time-series data. Prometheus serves as the central metric store from which both visualisation (via Grafana dashboards) and autoscaling decisions (via KEDA) are derived. This architecture ensures that energy consumption data is available in real time and can be queried programmatically to inform scaling decisions.

The integration of KEPLER with Prometheus provides granular visibility into the energy consumption of individual microservice pods, a level of observability that standard Kubernetes monitoring does not offer natively. This visibility is essential for enabling energy-aware scaling decisions, as it allows the orchestration platform to distinguish between microservices with different computational profiles and respond to their energy characteristics independently.

Threshold-Based Cost-Control Mechanism

The autoscaling logic is governed by a threshold mechanism designed to prevent resource over-provisioning. The mechanism operates on the principle that energy consumption at the microservice level serves as a direct proxy for operational cloud compute cost: higher energy consumption correlates with greater resource utilisation, which in turn translates to higher cloud infrastructure charges in pay-per-use pricing models.

The energy consumption proportion of a given microservice is calculated as follows:

$$E_proportion = E_microservice / E_namespace_total$$

where $E_microservice$ represents the energy consumed by all pods of a specific microservice, and $E_namespace_total$ represents the total energy consumed by all pods within the same Kubernetes namespace. This proportion quantifies the relative energy impact of each microservice on the overall cluster, providing a normalised metric suitable for threshold-based control.

A predefined energy consumption threshold (T) is established as the maximum acceptable energy proportion for any individual microservice. In this study, the initial baseline threshold (T) was set at 1% of the total Minikube cluster energy capacity, derived from observing the baseline idle energy consumption of the CleanseService microservice on the single-node cluster and selecting a ceiling marginally above it. However, rather than applying this value as a fixed ceiling, the threshold was configured dynamically based on the processing function of each microservice. Different processing functions exhibit distinct computational profiles: data cleansing operations involve iterative record-level validation with variable CPU intensity depending on data quality, whereas clinical risk calculation involves deterministic mathematical computation with a more predictable resource envelope. Accordingly, the energy threshold is parameterised per processing function, allowing the autoscaler to apply a higher energy ceiling to functions with inherently greater computational demands and a tighter ceiling to functions with lighter or more predictable profiles. This function-aware parameterisation ensures that scaling decisions reflect the operational characteristics of the work being

performed, rather than imposing a uniform constraint that may be too restrictive for intensive functions or too permissive for lightweight ones. In production deployments, this parameterisation would be calibrated based on the specific infrastructure capacity, workload characteristics, and budget constraints of the target health facility.

The autoscaling behaviour follows these rules:

1. When workload demand increases and additional processing capacity is required, KEDA scales out by adding pod replicas, subject to a maximum replica count of 10.
2. KEPLER continuously monitors energy consumption and exports metrics to Prometheus.
3. If the energy proportion of the microservice exceeds the threshold T , further scale-out is halted, regardless of incoming workload demand. This prevents the system from adding replicas that would push energy consumption, and by extension, operational cost, beyond sustainable limits.
4. When the workload subsides and energy consumption falls below the threshold, the system scales in by removing excess replicas, reducing resource consumption to the minimum required for baseline operation.
5. If a spike in energy consumption occurs during active processing but remains within the threshold, the system allows the existing pods to complete their workload without additional scaling, permitting energy levels to decline naturally.

This mechanism ensures that scaling decisions are bounded by an energy budget rather than driven solely by demand, providing a cost ceiling that prevents runaway resource consumption during workload surges. The workflow integrating the monitoring and autoscaling components is illustrated in Figure 2, and the system overview showing the threshold mechanism is presented in Figure 3.

Comparison Baseline

To contextualise the energy-aware autoscaler's behaviour, the system was initially deployed using the native Kubernetes horizontal pod autoscaler (HPA), configured to scale pod replicas when CPU utilisation exceeded 70% of the allocated resource limit. HPA served as the comparison baseline, representing conventional orchestration behaviour without energy or cost awareness. Observations from the HPA deployment, including latency degradation and uncontrolled resource growth under heavy workloads, informed the design of the energy-aware mechanism and provided qualitative comparison data for the evaluation.

Evaluation Design

This evaluation constitutes a formative proof-of-concept, not an exhaustive benchmarking study - designed to establish whether energy-aware orchestration can achieve meaningful resource reductions under progressively intensifying workloads while maintaining system performance and data processing integrity.

Figure 2. Autoscaling workflow integrating KEPLER energy telemetry, Prometheus metric collection, and KEDA event-driven scaling with threshold-based cost control.

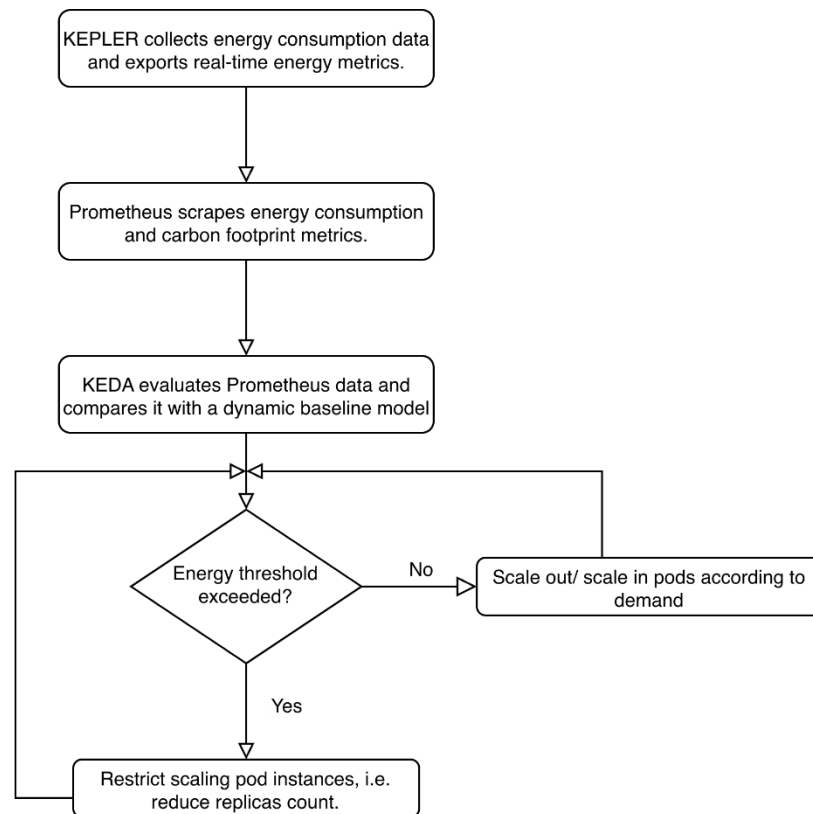
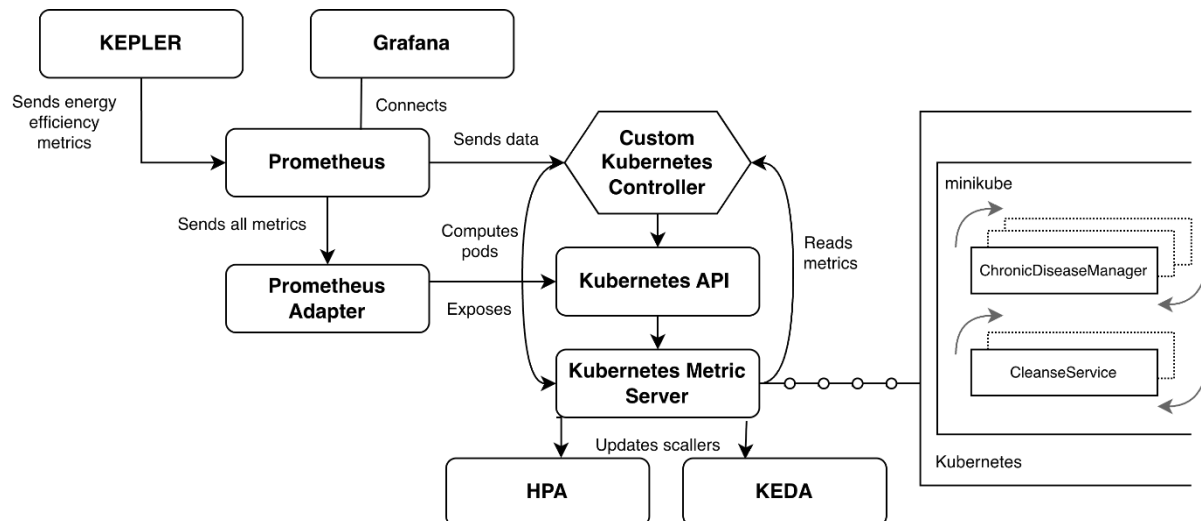


Figure 3. System overview of the energy-aware autoscaler, showing the threshold mechanism that halts pod scaling when energy consumption exceeds the predefined limit.



Workload Scenarios

Four workload scenarios of increasing intensity were designed to stress-test the autoscaling mechanism across a range of data processing demands representative of facility-to-district-level healthcare operations. Each scenario was defined by two parameters: dataset size (number of clinical records submitted for processing) and concurrency level (number of simultaneous requests). The scenarios were structured as a deliberate escalation,

progressing from routine facility-level workloads through to district-level surge conditions. While national-scale health information systems would process data volumes orders of magnitude larger (millions of records with hundreds of concurrent requests), the evaluated scenarios were designed to demonstrate the mechanism's effectiveness at a scale achievable within the constrained proof-of-concept environment. Table 2 describes each scenario and its healthcare context mapping.

Table 2. Progressive Workload Scenarios and Healthcare Context Mapping

Scenario	Records	Concurrent Requests	Healthcare Context	Rationale
1	10,000	30	Small health facility or clinic uploading daily patient encounter records from routine consultations	Tests system behaviour under moderate data volume with high request concurrency, representing distributed data submission from multiple points of care
2	20,000	15	District health office aggregating weekly chronic disease surveillance returns from a cluster of primary care clinics	Tests system response to larger datasets with moderate concurrency, reflecting periodic batch reporting cycles
3	50,000	15	Large district or sub-regional health authority consolidating monthly reporting data across multiple facilities	Tests system capacity under large data volumes, reflecting planned large-scale data operations
4	50,000	25	District-level surge during a localised disease outbreak, requiring rapid processing of increased clinical data volumes under time pressure	Tests system behaviour under extreme load combining large data volume with high concurrency, representing surge demand during public health emergencies

In each scenario, requests were submitted to the CleanseService microservice, which performed data cleansing operations on the synthetic cardiovascular disease (CVD) patient dataset before passing the processed records to the ChronicDiseaseManager for risk calculation. This workload path exercises the full data processing pipeline and represents the computationally intensive operations, data validation, missing value handling, format normalisation, and clinical computation, that characterise healthcare data processing in practice [37].

Measurements

For each workload scenario, the following metrics were captured:

- **Energy consumption rate:** The proportion of total cluster energy consumed by the CleanseService microservice, measured via KEPLER (Kubernetes-based Efficient Power Level Exporter) and Prometheus at two time points: during active processing (pre-scaling measurement) and after workload completion and any scaling events (post-scaling measurement). The difference between these values quantifies the energy reduction achieved by the autoscaling mechanism.
- **CPU utilisation:** Percentage of cluster CPU allocated to the microservice during and after workload processing, providing an indicator of computational demand responsiveness.
- **Memory utilisation:** Percentage of cluster memory consumed, monitored for stability to ensure that energy-aware scaling did not introduce memory-related performance degradation.
- **Scaling behaviour:** The number of pod replicas deployed before, during, and after each workload scenario, including any scale-out (adding replicas) or scale-in (removing replicas) events triggered by the energy threshold mechanism.

Cost Translation Methodology

To assess the practical financial implications of the observed energy reductions for the target deployment context, empirical results were mapped to estimated operational cost savings using published on-demand compute pricing from AWS cloud regions directly serving developing health systems: Africa (Cape Town, af-south-1) and Asia Pacific (Mumbai, ap-south-1). These regions were selected because they represent the cloud infrastructure options available to health facilities in Sub-Saharan Africa and South Asia, two of the regions where UHC goals are most actively pursued and where the financial barriers to digital health adoption documented in the literature are most acute [7-9]. The Minikube evaluation environment's resource profile (2 vCPUs, 4 GiB memory) was mapped to the AWS EC2 t3.medium instance type as the closest comparable virtual machine. The desk-based translation applied the observed energy reduction proportions to monthly compute cost baselines for each region across multiple deployment scales. This approach provides an indicative rather than definitive cost analysis, intended to demonstrate the direction and approximate magnitude of potential savings for resource-constrained deployments. The full methodology, assumptions, and projections are presented in the Estimated Operational Cost Implications subsection of the Results section.

Dataset

The system was evaluated using a synthetic patient dataset focused on chronic disease risk assessment. The dataset was generated to reflect the structure and variability of cardiovascular disease clinical data as encountered in primary care and population health settings, including patient demographics, clinical measurements (blood pressure, cholesterol levels, body mass index, blood glucose), lifestyle risk factors (smoking status, physical activity), and medical history variables.

The dataset was deliberately designed to include data quality characteristics typical of healthcare data collected in resource-limited settings [37,22]: missing values across multiple clinical fields, inconsistent record formatting, unexpected outliers in clinical measurements, and irregular data entry patterns. These characteristics were included because they directly influence processing demands; datasets with higher rates of missing or inconsistent data require additional computational effort during cleansing, causing fluctuations in CPU and memory utilisation that the autoscaling mechanism must accommodate. By incorporating these features, the evaluation tests the autoscaler's ability to manage the variable resource demands that arise from real-world data quality variability, rather than the predictable workloads generated by clean, uniform datasets.

The use of synthetic data was appropriate for this study for three reasons. First, the evaluation objective was to assess infrastructure-level behaviour (energy consumption, scaling dynamics, resource utilisation) rather than clinical accuracy or patient-level outcomes, making the provenance of the clinical values immaterial to the findings. Second, synthetic data eliminates the ethical and governance complexities associated with real patient data, including informed consent, data protection compliance, and institutional review requirements, which would be disproportionate for a proof-of-concept infrastructure evaluation. Third, synthetic generation allowed extreme and low-probability data quality conditions to be constructed deliberately, conditions that are rarely represented in publicly available clinical datasets and that may not surface in a real deployment except over an extended period of operation. This made it possible to stress-test the autoscaler against unpredictable and unprecedented workload conditions that a shorter real-world observation window would be unlikely to reveal. This approach is consistent with recent systems-level evaluations of healthcare microservices architectures that have similarly employed synthetic datasets to assess infrastructure performance independently of clinical data sensitivity [18].

Ethics approval was not required for this study as it involves the analysis of synthetic patient datasets that do not contain real human participant data.

RESULTS

Energy Consumption Across Progressive Workload Scenarios

The energy-aware autoscaler consistently reduced infrastructure energy consumption across all four progressive workload scenarios. Table 3 presents the quantitative outcomes

for each scenario, including pre- and post-scaling energy consumption rates, CPU and memory utilisation, scaling behaviour, and the percentage reduction in energy consumption achieved.

Table 3. Energy Consumption and Resource Utilisation Outcomes Across Progressive Workload Scenarios

Scenario	Healthcare Context	Records	Concurrent Requests	Energy Rate Pre-Scaling (%)	Energy Rate Post-Scaling (%)	Energy Reduction (%)	CPU Pre (%)	CPU Post (%)	Memory Pre (%)	Memory Post (%)	Scaling Behaviour
1	Clinic daily batch	10,000	30	1.53	1.32	13.7	6	7	75	75	Scale-in from 1 to 0 pods, then scale-out back to 1
2	District weekly reporting	20,000	15	5.54	3.97	28.3	5	5	74	70	No scaling; remained at 1 pod
3	Sub-regional aggregation	50,000	15	3.07	2.70	12.1	12	5	74	74	No scaling; remained at 1 pod
4	District outbreak surge	50,000	25	18.62	12.40	33.4	13	5	74	73	No scaling; remained at 1 pod

Across all scenarios, the autoscaler achieved energy reductions ranging from 12.1% to 33.4%. The most substantial reduction (33.4%) was observed during Scenario 4, the epidemic surge simulation, where the system processed 50,000 clinical records under 25 concurrent requests. During this high-intensity workload, energy consumption spiked to

18.62% of total cluster capacity during active processing. Rather than scaling out additional pod replicas, which would have further increased energy consumption, the threshold mechanism prevented unnecessary replication, allowing the existing pod to complete the workload. Upon task stabilisation, energy consumption declined to 12.40%, representing a reduction of 6.22 percentage points.

A similar pattern was observed in Scenario 2, the weekly surveillance aggregation, where 20,000 records processed across 15 concurrent requests produced a pre-scaling energy rate of 5.54%. The autoscaler's threshold control prevented redundant container replication, and post-scaling energy consumption fell to 3.97%, a reduction of 28.3%. This scenario demonstrated particularly effective budget-aware provisioning: despite a significant initial energy spike, the system managed the workload without scaling beyond a single pod, confirming that the energy threshold mechanism can regulate resource allocation without requiring additional replicas.

In Scenario 1, the routine district reporting simulation, the system exhibited active scaling behaviour: the autoscaler scaled in from 1 to 0 pods during a low-demand interval and subsequently scaled back out to 1 pod when new requests arrived. This dynamic scaling pattern reduced energy consumption by 13.7% and demonstrated the autoscaler's responsiveness to fluctuating demand levels typical of daily clinical data submission cycles.

Scenario 3, the public health campaign processing simulation, showed the smallest energy reduction (12.1%) despite processing the largest dataset at 50,000 records. This outcome reflects the moderate concurrency level (15 requests): the workload was data-intensive but not concurrency-intensive, producing a lower initial energy spike (3.07%) and consequently a smaller absolute reduction. This finding suggests that the magnitude of energy savings is influenced by the interaction between data volume and request concurrency rather than by data volume alone.

CPU utilisation remained responsive across all scenarios, increasing only as necessary to accommodate workload demand and declining promptly after processing completed. Post-scaling CPU values returned to between 5% and 7%, indicating that the energy-aware mechanism did not impair the system's ability to release computational resources after task completion. Memory utilisation remained stable at baseline levels (70%–75%) throughout all scenarios, confirming that the energy-aware autoscaling approach did not introduce memory-related performance degradation or resource contention.

Comparison With Conventional Autoscaling

The initial deployment of the cardiovascular disease (CVD) data processing system used the native Kubernetes horizontal pod autoscaler (HPA), configured to scale pod replicas when CPU utilisation exceeded 70% of the allocated resource limit. While HPA performed adequately under lighter workloads, the system exhibited progressively degraded behaviour as data volume and concurrency increased. Under heavy workload scenarios equivalent to Scenarios 3 and 4, HPA scaled out additional pod replicas in direct response to rising CPU utilisation without regard for the cumulative energy or cost implications of each additional replica. This resulted in resource over-provisioning, computational capacity exhaustion, and

network stack saturation as the high volume of concurrent requests overwhelmed the available buffer space. Latency degradation became pronounced, and the system's ability to allocate further microservice instances was impaired, reducing overall processing throughput.

Critically, the HPA deployment also produced significant spikes in energy consumption and associated carbon emissions during scale-out events, as each additional replica introduced incremental compute overhead without a proportionate improvement in processing performance under saturated conditions. The energy-aware autoscaler addressed these limitations by introducing a cost ceiling that bounded scale-out behaviour irrespective of CPU-driven demand signals. Table 4 presents a structured qualitative comparison of the two autoscaling approaches across key operational dimensions.

Table 4. Qualitative Comparison of Conventional HPA and Energy-Aware Autoscaling

Dimension	Horizontal Pod Autoscaler (HPA)	Energy-Aware Autoscaler (KEDA + KEPLER)
Scaling trigger	CPU and memory utilisation thresholds (70% CPU)	Energy consumption proportion relative to cluster capacity, with threshold parameterised per processing function (baseline 1%, adjusted by computational profile)
Scaling behaviour under high load	Unconstrained scale-out; adds replicas proportionally to CPU demand without upper bound on energy impact	Bounded scale-out; halts replica addition when energy threshold is exceeded, regardless of CPU demand
Resource over-provisioning	Observed under heavy workloads; excess replicas consumed resources without proportionate throughput gains	Mitigated; threshold mechanism prevents redundant replica deployment during energy spikes
Latency under surge conditions	Degraded; network stack and buffer saturation observed under high concurrency	Maintained; workload processed within existing pod capacity without additional scaling overhead
Energy consumption awareness	None; scaling decisions are performance-driven with no visibility into energy impact	Native; real-time energy telemetry informs every scaling decision via KEPLER and Prometheus
Cost awareness	None; no mechanism to limit operational expenditure during demand surges	Indirect; energy threshold serves as a proxy cost ceiling, bounding infrastructure spend

Dimension	Horizontal Pod Autoscaler (HPA)	Energy-Aware Autoscaler (KEDA + KEPLER)
Suitability for resource-constrained deployment	Limited; unconstrained scaling risks exceeding available infrastructure capacity and budget	Designed for constrained environments; scaling behaviour is bounded by an energy budget aligned with facility capacity

Estimated Operational Cost Implications

To assess the practical financial relevance of the observed energy reductions for the target deployment context, empirical results were translated to indicative operational cost projections using published on-demand compute pricing from AWS regions serving developing health systems. The Minikube evaluation environment's resource profile (2 vCPUs, 4 GiB memory) was mapped to the AWS EC2 t3.medium instance type. Pricing was sourced from two regions directly relevant to LMIC health system deployments: Africa (Cape Town, af-south-1) at US\$0.0520 per hour (US\$37.96 per node per month) and Asia Pacific (Mumbai, ap-south-1) at US\$0.0458 per hour (US\$33.43 per node per month). Cloud compute pricing in regions serving developing health systems typically carries a 10%–25% premium over baseline US pricing, reflecting the higher infrastructure costs of newer, lower-demand data centres, an additional cost burden borne by the very settings that can least afford it.

The observed energy reductions represent proportional savings in compute resource consumption during active data processing. In pay-per-use cloud pricing models, reduced resource consumption translates directly to reduced compute expenditure. The proportional reductions (12.1%–33.4%) are infrastructure-agnostic: they apply equally regardless of the specific instance type, cloud provider, or deployment scale. Table 5 presents estimated monthly savings across two LMIC deployment regions and three infrastructure scales, a single node, a 5-node cluster representative of a district health facility, and a 10-node cluster representative of a regional health authority.

Table 5. Estimated Monthly Operational Cost Savings Across LMIC Cloud Regions Based on AWS EC2 t3.medium On-Demand Pricing (June 2026)

Scenario	Energy Reduction (%)	Africa (Cape Town) Single Node (US\$)	Africa 5-Node (US\$)	Africa 10-Node Monthly (US\$)	Africa 10-Node Annual (US\$)	South Asia (Mumbai) Single Node (US\$)	South Asia 5-Node (US\$)	South Asia 10-Node Monthly (US\$)	South Asia 10-Node Annual (US\$)
1	13.7	5.20	26.00	52.01	624.06	4.58	22.90	45.80	549.65
2	28.3	10.74	53.71	107.43	1,289.12	9.46	47.31	94.62	1,135.42

Scenario	Energy Reduction (%)	Africa (Cape Town) Single Node (US\$)	Africa 5-Node (US\$)	Africa 10-Node Monthly (US\$)	Africa 10-Node Annual (US\$)	South Asia (Mumbai) Single Node (US\$)	South Asia 5-Node (US\$)	South Asia 10-Node Monthly (US\$)	South Asia 10-Node Annual (US\$)
3	12.1	4.59	22.97	45.93	551.18	4.05	20.23	40.46	485.46
4	33.4	12.68	63.39	126.79	1,521.44	11.17	55.83	111.67	1,340.03

Reference: Monthly baselines , Africa (Cape Town): US\$37.96/node; South Asia (Mumbai): US\$33.43/node. Savings calculated as baseline × energy reduction proportion, scaled linearly by node count.

These projections are intentionally conservative in three respects. First, the t3.medium is a modest, burstable instance; production healthcare data processing deployments would typically use larger instances with higher hourly rates, yielding proportionally greater absolute savings. Second, the projections assume on-demand pricing, which represents the highest per-hour cost; reserved instance or savings plan pricing would reduce the baseline but preserve the proportional saving. Third, the projections apply the energy reduction to continuous node operation; in practice, the savings during active processing surges , when the autoscaler prevents unnecessary pod replication , would be concentrated during peak periods, reducing the compute cost precisely when it would otherwise spike most sharply.

It is important to emphasise that the workload scenarios evaluated here represent facility-to-district-level data processing demands, not national-scale operations. A national health information system processing population-level data would handle millions of records with hundreds or thousands of concurrent requests, operating continuously across substantially larger infrastructure. The proof-of-concept scenarios were designed to demonstrate the mechanism's effectiveness at a scale achievable within the constrained evaluation environment; they do not represent the upper bound of its applicability. Indeed, the gap between the evaluated scale and production-scale national deployments strengthens the cost savings argument: if 12% to 33% energy reductions are achievable at modest facility-level workloads, the same proportional reductions applied across a national digital health information system, such as the cloud-native NDHIS architecture described by Zao et al [17] for Malawi, comprising dozens or hundreds of nodes, multiple microservices, and continuous data processing, would compound to yield substantially greater savings. For example, a 50-node deployment with a 33.4% energy reduction under the Africa (Cape Town) pricing baseline would yield estimated annual savings of approximately US\$7,607 on the conservative t3.medium instance alone, and this figure would increase considerably with larger instance types, additional microservices, and the higher workload volumes characteristic of national-scale operations. The cost savings reported here therefore represent a lower bound derived from deliberate methodological conservatism at the smallest viable deployment scale; the actual savings achievable in production health information system deployments would be considerably larger.

For a district hospital in Sub-Saharan Africa operating a 10-node cluster under Scenario 4 surge conditions, the energy-aware autoscaler would save approximately US\$1,521 annually. For a regional health authority in South Asia operating at equivalent scale, the annual saving would be approximately US\$1,340. In LMIC healthcare settings where annual digital health operating budgets for a district may range from US\$5,000 to US\$20,000, savings of this magnitude represent a meaningful proportion of available resources , particularly where cloud infrastructure is funded through time-limited donor grants or national health programme allocations and every dollar of operational saving extends the funded lifespan of the system [9,10].

It is important to note that these estimates are indicative. They are derived from published pricing applied to observed energy reductions, not from direct measurement of cloud billing under the energy-aware autoscaler. The actual financial impact would depend on the specific cloud provider, pricing model, deployment scale, workload frequency, and contractual arrangements. The purpose of this analysis is to demonstrate the direction and approximate magnitude of potential savings, establishing that the observed energy reductions carry practical financial implications for resource-constrained deployments in the regions where UHC goals are most urgently needed.

DISCUSSION

Principal Findings

This proof-of-concept study demonstrates that integrating real-time energy consumption metrics into Kubernetes autoscaling decisions can meaningfully reduce the operational resource demands of cloud-native healthcare microservices without compromising data processing performance. Across four progressive workload scenarios representing healthcare operational demands from routine district reporting to epidemic surge surveillance, the energy-aware autoscaler achieved post-scaling energy reductions of 12.1% to 33.4%. CPU utilisation remained responsive throughout, increasing only as necessary to accommodate workload demand and returning to baseline levels after processing completion. Memory utilisation remained stable across all scenarios, confirming that the energy-aware approach did not introduce resource contention or performance degradation.

The threshold-based cost-control mechanism proved effective at regulating scale-out behaviour under high-intensity workloads. In the most demanding scenario, 50,000 records under 25 concurrent requests, the autoscaler prevented unnecessary pod replication during a substantial energy spike (18.62% of cluster capacity), allowing the workload to complete within existing resources and yielding a 33.4% reduction upon task stabilisation. This behaviour stands in direct contrast to the conventional horizontal pod autoscaler (HPA), which under equivalent conditions produced uncontrolled resource growth, latency degradation, and network saturation without proportionate throughput gains.

The preliminary cost translation analysis suggests that these energy reductions, while modest in absolute terms for a single-node proof-of-concept deployment, translate to proportional savings that compound substantially at production scale. The 12% to 33% reductions are infrastructure-agnostic and would apply across every node and microservice in a real-world deployment. For national-scale health information systems operating across dozens of nodes with multiple microservices and continuous data processing, the cumulative annual savings would be considerably larger than those reported here from the deliberately conservative proof-of-concept baseline. For resource-limited health facilities where cloud compute costs directly compete with clinical expenditure, proportional reductions of this magnitude can materially influence the viability and sustainability of digital health system adoption.

Comparison With Prior Work

The findings of this study extend the emerging body of work on energy-aware and carbon-aware cloud autoscaling by addressing a deployment context that prior research has not examined. Where Gebreweld [30] and Hanafy et al [31] demonstrated carbon-aware scaling for low-priority and batch workloads respectively, our results show that energy-aware autoscaling is effective for the variable, on-demand workloads characteristic of healthcare data processing - achieving 12% to 33% reductions without the option of deferring processing to lower-demand periods. Where Cortellessa et al [32] and De Nardin et al [33] explored energy-performance trade-offs through architectural refactoring and proactive elasticity models, our approach operates at the orchestration level through a simple threshold mechanism that requires no application-level modification. And where the cloud-native healthcare literature [15-18,35,36] has addressed functionality, trustworthiness, privacy, and scalability, our study introduces the operational cost dimension - demonstrating that the question of whether resource-limited facilities can afford to run these systems is both answerable and addressable at the architectural level.

Implications for Health System Planning and UHC

The findings carry practical implications for digital health system planning in resource-constrained environments. The barriers literature consistently identifies financial constraints, including high startup costs, expensive maintenance, and reliance on time-limited external funding, as dominant obstacles to health information system adoption in resource-limited settings [7-9]. Meessen [10] has argued that digital health solutions must fit within coherent health financing architectures, and Wilson et al [6] have emphasised that technology for Universal Health Coverage (UHC) must be deployed with context-specific strategies that account for local capacity and sustainability.

Our results suggest that energy-aware, cost-mitigating orchestration can contribute to addressing these barriers by reducing the ongoing operational costs of cloud-native health information systems. If the proportional energy reductions observed in this study (12% to 33%) are achievable at production scale, they could lower the total cost of ownership for cloud-native health data processing infrastructure, making sustained operation more feasible for facilities operating under constrained budgets. This is especially relevant as

cloud-native microservices architectures are increasingly adopted for national-scale digital health systems [17], and as AI-driven analytical workloads, which are computationally intensive and therefore energy-expensive, become integral to health system operations [18].

The approach also aligns with the principle identified in our earlier review of HIS adoption barriers [9]: that sustainability and scalability must be addressed in tandem if health information systems are to support UHC goals in resource-limited environments. Designing for cost efficiency at the architectural level, rather than optimising retrospectively, ensures that sustainability is embedded as a foundational quality attribute rather than treated as an afterthought.

Limitations

Several limitations should be acknowledged when interpreting these findings.

The evaluation was conducted on a single-node Minikube environment. While this deployment was deliberately chosen to approximate the constrained compute infrastructure typical of resource-limited health facilities, it does not capture the complexity of multinode cloud deployments, including internode communication overhead, distributed scheduling, and network-level energy consumption. The observed energy reductions may differ in magnitude or pattern when the system operates across multiple nodes with heterogeneous workload distribution.

The system was evaluated using synthetic patient data. Although the dataset was designed to reflect the data quality characteristics of healthcare data collected in resource-limited settings, it does not capture the full variability, volume, or temporal patterns of real clinical data streams. The generalisability of the findings to production healthcare environments remains to be established through real-world evaluation.

The evaluation comprised four workload scenarios. While these were structured as a deliberate escalation through a representative operational range, they do not constitute an exhaustive exploration of the workload parameter space. The interaction between data volume, concurrency, data quality characteristics, and processing complexity warrants further investigation across a broader range of configurations.

While the energy consumption threshold was configured dynamically based on the processing function of each microservice, the per-function parameterisation was derived from baseline idle consumption observations and remained fixed within each function class throughout the evaluation. This approach does not adapt to changing workload patterns within a given function, seasonal demand variations, or evolving infrastructure capacity over time. A fully adaptive threshold mechanism that dynamically adjusts based on real-time workload intensity, historical consumption patterns, and function-level performance baselines would likely yield more nuanced and effective cost control.

The cost translation analysis is based on published cloud pricing applied to observed energy reductions, not on direct measurement of cloud billing under the energy-aware autoscaler. The estimates are indicative and assume proportional correlation between energy

consumption and compute cost, which may not hold precisely across all pricing models, instance types, or provider-specific billing mechanisms.

The energy consumption formulas currently calculate the proportion of energy consumed by each microservice relative to the total namespace consumption, but do not incorporate dynamic factors such as incoming request intensity, workload scaling velocity, or time-of-day pricing variations. Incorporating these factors would strengthen the precision of the cost-control mechanism.

Future Work

Future research should address the limitations identified above through several complementary directions.

Multinode cloud deployment on production-grade Kubernetes clusters (such as Amazon EKS, Azure AKS, or Google GKE) would test whether the energy reductions observed in this proof-of-concept translate to larger-scale environments with distributed workload scheduling and internode communication overhead. Such deployments would also enable direct measurement of cloud billing impact rather than reliance on pricing-based projections.

Evaluation with real-world healthcare data, subject to appropriate ethical approvals and data governance arrangements, would establish whether the energy consumption patterns observed with synthetic data generalise to the variability, volume, and temporal characteristics of production clinical data streams.

The static energy threshold should be replaced with an adaptive mechanism. Deep learning and reinforcement learning strategies have shown promise in optimising Kubernetes autoscaling decisions [38,39] and could be applied to dynamically adjust energy consumption thresholds based on workload intensity, historical patterns, and real-time cost signals. Control theory approaches may also offer formally grounded methods for threshold adaptation [14].

Integration with cloud provider cost APIs would enable real-time budget-aware scaling, where the autoscaler adjusts its behaviour not only based on energy metrics but also based on live pricing signals, reserved instance utilisation, and budget constraints specified by the health facility operator.

Finally, the system should be extended beyond the current two-microservice architecture to evaluate energy-aware orchestration across more complex microservices ecosystems comprising multiple interdependent services with heterogeneous computational profiles. As health information systems grow in scope and complexity, incorporating AI-driven analytics, interoperability layers, and patient-facing services, the energy and cost implications of orchestrating these services at scale will become increasingly significant.

CONCLUSIONS

Cloud-native microservices architectures offer the modularity, scalability, and resilience that health information systems require to support Universal Health Coverage, yet the operational costs of running these systems on cloud infrastructure remain a significant barrier to adoption in resource-limited settings. This proof-of-concept study demonstrates that integrating real-time energy consumption metrics into Kubernetes autoscaling decisions, using KEPLER for energy telemetry, Prometheus for metric aggregation, and KEDA for threshold-based scaling control, can reduce the operational resource demands of healthcare microservices by 12% to 33% without compromising data processing performance or system responsiveness.

By treating energy consumption as a direct proxy for operational cloud expenditure, the proposed framework introduces a cost-control dimension into container orchestration that is absent from conventional performance-driven autoscaling approaches. The cost savings demonstrated here are derived from a deliberately conservative proof-of-concept baseline; in production-scale health information systems operating across multiple nodes, microservices, and continuous processing cycles, the proportional reductions would compound to yield substantially greater savings. Cost projections for cloud regions serving Sub-Saharan Africa and South Asia confirm that these reductions carry practical financial implications for the settings where UHC investment is most urgently needed.

These findings support the broader principle that sustainability and cost awareness should be embedded as foundational architectural qualities in digital health systems, rather than pursued as retrospective optimisations. If cloud-native health information systems are to fulfil their potential in supporting UHC goals, particularly in low- and middle-income countries where financial barriers dominate the adoption landscape, they must be designed to be operationally affordable from the outset. Further validation through multinode cloud deployments, real-world clinical workloads, and adaptive threshold mechanisms is needed to translate these preliminary findings into production-ready architectural guidance for health system planners and implementers.

DECLARATIONS

Ethics Approval

Not applicable.

Consent to Participate

Not applicable.

Consent to Publish

Not applicable.

Data Availability

The synthetic patient dataset used in this study was generated using a custom-built data generation tool, the methodology for which has been previously published (DOI: 10.3233/SHTI250419). The source code for the data generation tool is not publicly deposited but is available from the corresponding author on reasonable request. The workload and scenario parameters used in the simulation-based stress-testing were generated dynamically during the evaluation process rather than drawn from a fixed, pre-defined set, and are available from the corresponding author on reasonable request.

Funding

The authors declare that no funds, grants, or other support were received for the submitted work.

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Authors' Contributions

TH and PK: Conceptualisation, Methodology, Writing — Original Draft, Writing — Review and Editing (equal contribution). TH: Software, Investigation, Formal Analysis, Visualisation. TH and PK: Data Curation. PK: Supervision, Project Administration. SK: Writing - Review and Editing. SB: Writing - Review and Editing. SdL: Writing - Review and Editing.

REFERENCES

1. Chiu WK, Fong BYF. Sustainable Development Goal 3 in healthcare. In: Fong BYF, Law R, eds. Environmental, Social and Governance and Sustainable Development in Healthcare. Springer Nature; 2023:33-45.
2. Roth S, Landry M, Parry J. Universal Health Coverage by Design: ICT-Enabled Solutions Are the Future of Equitable, Quality Health Care and Resilient Health Systems. Rockefeller Foundation; 2015.
3. Rasanathan K, O'Connell T, Yablonski J, Rees N. Universal health coverage. In: The UNICEF Health Systems Strengthening Approach. UNICEF; 2016.
4. Omaghomi TT, Akomolafe O, Ogugua JO, Daraojimba AI, Elufioye OA. Healthcare management in a post-pandemic world: lessons learned and future preparedness , a review. Int Med Sci Res J. 2024;4(2):210-223.
5. World Health Organization. Global Strategy on Digital Health 2020–2025. WHO; 2021.

6. Wilson D, Sheikh A, Görgens M, Ward K. Technology and universal health coverage: examining the role of digital health. *J Glob Health*. 2021;11:16006. doi:10.7189/jogh.11.16006
7. Derakhshani N, Doshmangir L, Ahmadi A, Fakhri A, Sadeghi-Bazargani H, Gordeev VS. Monitoring process barriers and enablers towards universal health coverage within the sustainable development goals: a systematic review and content analysis. *Clinicoecon Outcomes Res*. 2020;12:459-472. doi:10.2147/CEOR.S254946
8. Syaddam MM. Universal health coverage in developing countries: barriers and strategic reforms. *J Health Lit Qual Res*. 2021;1(2):99-112. doi:10.61194/jhlqr.v1i2.533
9. Haddad T, Kumarapeli P, de Lusignan S. Barriers and opportunities in developing health information system architectures to support universal health coverage in resource-limited settings. *Stud Health Technol Inform*. 2025;327:363-367. doi:10.3233/SHTI250405
10. Meessen B. The role of digital strategies in financing health care for universal health coverage in low- and middle-income countries. *Glob Health Sci Pract*. 2018;6(suppl 1):S29-S40. doi:10.9745/GHSP-D-18-00271
11. Jawhari B, Ludwick D, Keenan L, Zakus D, Hayward R. Benefits and challenges of EMR implementations in low resource settings: a state-of-the-art review. *BMC Med Inform Decis Mak*. 2016;16:1-12. doi:10.1186/s12911-016-0354-8
12. Pires LF, Cavalcante E, Batista T. Microservices in healthcare systems: a systematic mapping study. *J Biomed Inform*. 2021;118:103789. doi:10.1016/j.jbi.2021.103789
13. Haddad T, Kumarapeli P, de Lusignan S, Barman S, Khaddaj S. Advancing healthcare sustainability: embracing MACH architecture for health IT system transformation. *Stud Health Technol Inform*. 2024;316:1565-1566. doi:10.3233/SHTI240937
14. Xiao X. Architectural tactics to improve the environmental sustainability of microservices: a rapid review. Preprint. Posted online July 23, 2024. arXiv:2407.16706
15. Ianculescu M, Alexandru A. Microservices , a catalyzer for better managing healthcare data empowerment. *Stud Inf Control*. 2020;29(2):231-242. doi:10.24846/v29i2y202008
16. Akerele JI, Uzoka A, Ojukwu PU, Olamijuwon OJ. Improving healthcare application scalability through microservices architecture in the cloud. *Int J Sci Res Updates*. 2024;8(02):100-109.
17. Zao JKK, Wu JTS, Kanyimbo K, Delizy F, Gan TT, Kuo HI, Hsia CH, Lo CH, Yang SH, Richard CJA, Rajab B. Design of a trustworthy cloud-native national digital health information infrastructure for secure data management and use. *Oxf Open Digit Health*. 2024;2:oqae043. doi:10.1093/oodh/oqae043

18. Harshith M, Ansari ZA, Fatima S, Siddiqui S, Swarna S, Reddy DN, Mohsin SW. Federated microservices architecture with blockchain for privacy-preserving and scalable healthcare analytics. *Sci Rep.* 2026;16(1):9023. doi:10.1038/s41598-026-39837-1
19. Centofanti C, Santos J, Gudepu V, Kondepu K. Impact of power consumption in containerised clouds: a comprehensive analysis of open-source power measurement tools. *Comput Netw.* 2024;245:110371. doi:10.1016/j.comnet.2024.110371
20. Pijnacker B. Estimating container-level power usage in Kubernetes [master's thesis]. University of Groningen; 2024.
21. Pulimamidi R. Enhancing patient experience through IoT-based analytics and IT transformation in e-healthcare. *Measurement: Sensors.* 2024;33:101087. doi:10.1016/j.measen.2024.101087
22. Fidelia EN, Nkiru AG, Chinedu AD. Big data: a disruptive technology for sustainable development. *Int J Innov Sci Res Technol.* 2021;6(3):112-120.
23. Hameed A, Khoshkbarforousha A, Ranjan R, Jayaraman PP, Kolodziej J, Balaji P, Zomaya AY. A survey and taxonomy on energy-efficient resource allocation techniques for cloud computing systems. *Computing.* 2016;98:751-774. doi:10.1007/s00607-015-0457-1
24. Alharthi S, Alshamsi A, Alseiari A, Alwarafy A. Auto-scaling techniques in cloud computing: issues and research directions. *Sensors.* 2024;24(17):5551. doi:10.3390/s24175551
25. Ergasheva S, Khomyakov I, Kruglov A, Succil G. Metrics of energy consumption in software systems: a systematic literature review. *IOP Conf Ser Earth Environ Sci.* 2020;431(1):012051. doi:10.1088/1755-1315/431/1/012051
26. Li Y, Wen Y, Hu H. Carbon-aware scheduling in cloud data centers. *IEEE Trans Parallel Distrib Syst.* 2022;33(12):3496-3509. doi:10.1109/TPDS.2022.3156823
27. Lucivero F. Big data, big waste? A reflection on the environmental sustainability of big data initiatives. *Sci Eng Ethics.* 2020;26(2):1009-1030. doi:10.1007/s11948-019-00171-7
28. Rossi F, Cardellini V, Presti FL. Hierarchical scaling of microservices in Kubernetes. In: *Proc IEEE Int Conf Autonomic Computing and Self-Organizing Systems (ACSOS).* IEEE; 2020:28-37. doi:10.1109/ACSOS49614.2020.00015
29. Soldani D, Nahi P, Bour H, Jafarizadeh S, Soliman MF, Di Giovanna L, Risso F. eBPF: a new approach to cloud-native observability, networking and security for current and future mobile networks. *IEEE Access.* 2023;11:57174-57202. doi:10.1109/ACCESS.2023.3284211

30. Gebreweld HB. Evaluating the energy consumption impact of carbon-aware autoscaling in microservice-based applications on the public cloud: a sustainability perspective [master's thesis]. University of Groningen; 2023.
31. Hanafy WA, Liang Q, Bashir N, Irwin D, Shenoy P. CarbonScaler: leveraging cloud workload elasticity for optimizing carbon efficiency. *Proc ACM Meas Anal Comput Syst.* 2023;7(3):1-28. doi:10.1145/3605159
32. Cortellessa V, Di Pompeo D, Tucci M. Exploring sustainable alternatives for the deployment of microservices architectures in the cloud. Preprint. Posted online February 17, 2024. arXiv:2402.11238
33. De Nardin IF, da Rosa Righi R, Lopes TRL, da Costa CA, Yeom HY, Köstler H. On revisiting energy and performance in microservices applications: a cloud elasticity-driven approach. *Parallel Comput.* 2021;108:102858. doi:10.1016/j.parco.2021.102858
34. Araújo G, Barbosa V, Lima LN, Sabino A, Brito C, Fé I, Silva FA. Energy consumption in microservices architectures: a systematic literature review. *IEEE Access.* 2024;12:186710-186729. doi:10.1109/ACCESS.2024.3469123
35. Avireneni RT. Cloud-native middleware: architecting scalable and resilient healthcare delivery systems. *J Comput Sci Technol Stud.* 2025;7(3):901-908. doi:10.32996/jcsts.2025.7.3.100
36. Boda VVR, Immaneni J. Healthcare in the fast lane: how Kubernetes and microservices are making it happen. *Int J Emerg Res Eng Technol.* 2021;2(3):33-42.
37. Shickel B, Tighe PJ, Bihorac A, Rashidi P. Deep learning in healthcare: data preprocessing and representation challenges. *J Am Med Inform Assoc.* 2021;28(1):1-15. doi:10.1093/jamia/ocaa186
38. Pandey V. Reinforcement learning-based autoscaling for cost and performance optimisation in Kubernetes clusters. In: *Proc Int Conf P2P, Parallel, Grid, Cloud and Internet Computing.* Springer; 2025:25-37.
39. Zhang Q, Chen M, Li X. Reinforcement learning-based autoscaling for Kubernetes clusters. *Future Gener Comput Syst.* 2023;139:95-108. doi:10.1016/j.future.2022.09.024