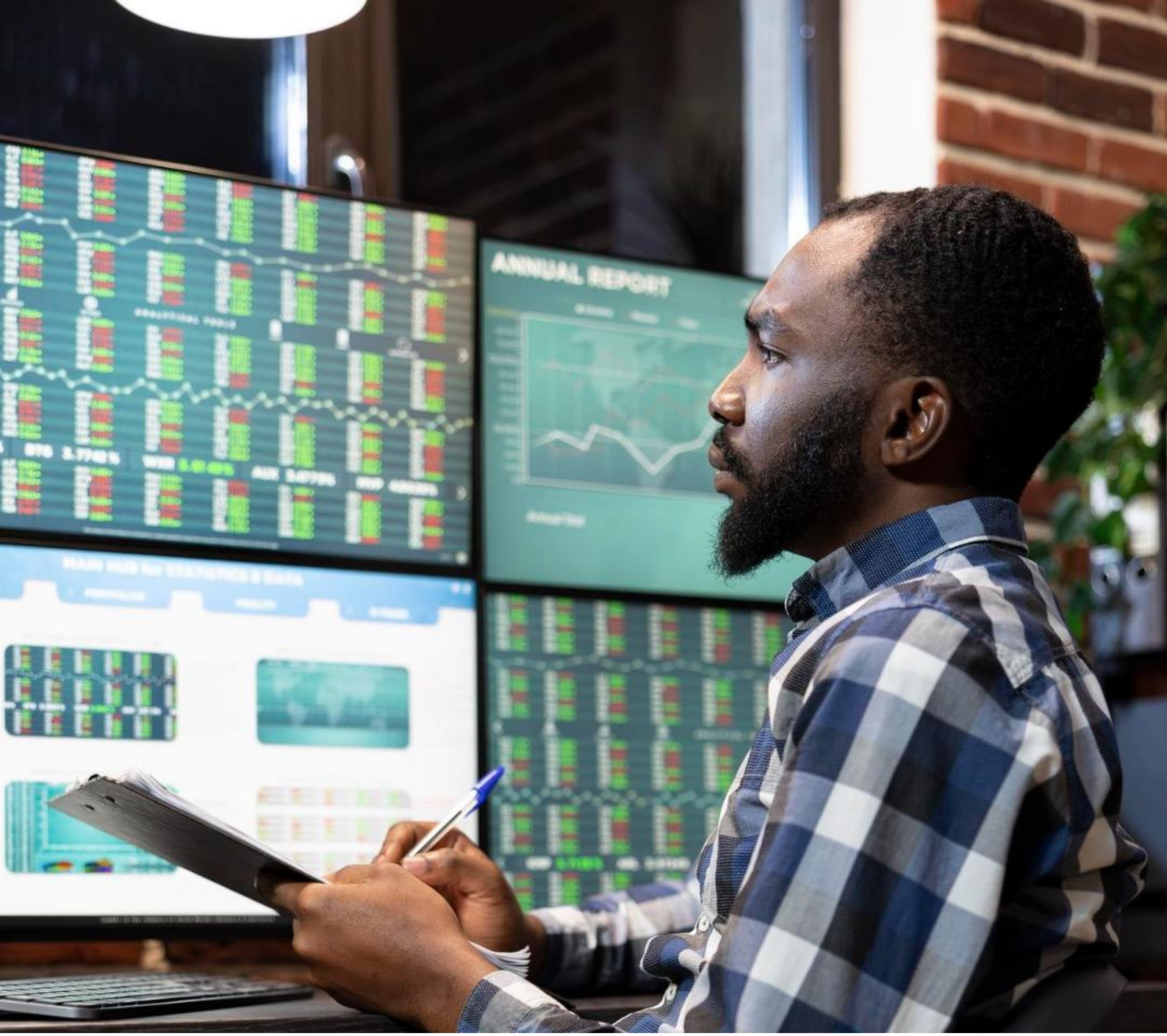


SUSTAINABLE CLOUD OPERATIONS FOR RESEARCH

(SCORE)

A Guideline for Cost-Efficient and Carbon-Aware Data Pipelines Design in Low- and Middle-Income Countries Research

SUSTAINABLE CLOUD OPERATIONS FOR RESEARCH (SCORE)

A Guideline for Cost-Efficient and Carbon-Aware Data Pipeline Design in Low- and Middle-Income Countries Research

Nigel Kamotho[1], Allan Zablon[1], Stephen Wong[1], Amos Bunde[1], Antony Kagure[1], Thembi Kiiru[1], Anthony K. Ngugi[2], Amina Abubakar[3], Akbar K. Waljee[4,5], Farhana Alarakhiya[1]

Affiliations

[1]Data Innovation Office, Aga Khan University, Nairobi, Kenya

[2]Department of Population Health, Aga Khan University, Nairobi, Kenya

[3]Institute for Human Development, Aga Khan University, Nairobi, Kenya

[4]Center for Global Health Equity, University of Michigan, Michigan, USA

[5]Department of Learning Health Sciences, Michigan Medicine, University of Michigan, Michigan, USA

Executive Summary

As cloud computing becomes a growing pillar for modern data intensive research, especially in fields like global health, many organizations across low- and middle-income countries (LMICs) remain hindered by operational constraints. These include limited DevOps capability, unreliable internet connectivity, fixed grant ceilings, and growing demands for environmental sustainability. Standard cloud architecture is based on stable infrastructure and high technical ability, making them neither viable in terms of cost nor sustainability within LMICs environments.

This report introduces the Sustainable Cloud Operations for Research (SCORE) guidelines. It is a guideline for context-specific cloud architecture models for LMICs research groups to optimize architecture decisions for cost, performance, and carbon emissions. These guidelines consist of three phases: Assessment, Selection, and Optimization. There are actionable opportunities at each step to guide pipeline design and resource management under constrained operating conditions.

The guidelines were validated using the National Institutes of Health Chest X-ray dataset (42 GB). Three Azure-native ingestion strategies based on the guidelines are compared: Synapse Pipelines (no-code), Azure Functions (serverless), and Synapse Notebooks (code-based). Execution time, cost, and emissions are tracked using Azure Monitor, Cost Management, and Emissions Insights tools. The results revealed that the serverless approach (Azure Functions) achieved the lowest cost (USD 0.01), lowest carbon emissions (0.00003 kg CO_2e), and shortest execution time (1.12 hours), whereas the Code approach (Synapse Notebooks) incurred the highest cost (USD 15.20) and emissions (0.16901 kg CO_2e), primarily due to a dedicated Spark pool. Additionally, geo-replication accounted for approximately 85% of storage costs, highlighting the need for clearer understanding of cloud pricing structures.

SCORE addresses a critical gap among existing cloud infrastructure guidelines. Rather than providing broad best practices, it provides systematic guidelines in a practical iterative model that addresses the fiscal, technical, and sustainability constraints that LMICs institutions face. The guidelines are designed to be practical, scalable, and uniform in a range of research environments where cloud integration must be efficient and sustainable.

Table of Contents

Glossary

AWS	Amazon Web Services
BI	Business Intelligence
CO_2	Carbon Dioxide
DLQ	Dead-Letter Queue
DSI-Africa	Data Science for Health Discovery and Innovation in Africa Consortium
ETL	Extract, Transform, Load
GB	Gigabyte
GHG	Greenhouse Gases
GHSP	Global Health and Science Practice
GCP	Google Cloud Platform
GPU	Graphics Processing Unit (inferred contextually)
HIC	High Income Country
IaC	Infrastructure as Code
I/O	Input/Output
IoT	Internet of Things
KB	Kilobyte
$kgCO_2e$	Kilograms of Carbon Dioxide Equivalent
LMICs	Low- and Middle-Income Countries
MB	Megabyte
ML	Machine Learning
NDAs	Non-Disclosure Agreements
NIH	National Institutes of Health
RAM	Random Access Memory
SCORE	Sustainable Cloud Operations for Research
USD	United States Dollar
UZIMA – DS	Utilizing Health Information for Meaningful Impact in East Africa
VM	Virtual Machine

Introduction

Research in the Cloud

Cloud computing has become an essential part of modern research, offering scalability, flexibility, and cost efficiency for data-intensive projects. For health research institutions, especially in low- and middle-income countries (LMICs), cloud platforms make it possible to store, manage, and analyze large datasets without the need to maintain costly on-prem infrastructure. [1,2] The cloud also supports secure data collaboration across borders through approaches such as data de-identification, federated learning, and digital twin modeling, which enables sensitive information to remain local while still being shared for joint research.

However, despite these advantages, implementing cloud-based research in LMICs remains a challenge. Teams often face limited DevOps expertise, unstable internet connectivity, rigid grant ceilings, and the pressure to operate sustainably.[3] Most cloud systems provided by major vendors such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure are designed for high-income countries with reliable infrastructure and specialized staff. As a result, LMIC institutions struggle to adapt these systems to their realities, often experiencing unpredictable costs and underutilized services.

At the same time, the environmental cost of cloud computing is growing. Data centers consume massive amounts of power, contributing significantly to global greenhouse gas emissions.[4] Although major providers now offer sustainability tools such as Microsoft's Emissions Impact Dashboard and Google's Carbon Footprint, which require advanced technical knowledge and infrastructure that are not always available in LMIC settings.[5]

To address these challenges, the Sustainable Cloud Operations for Research (SCORE) framework provides practical, context-specific guidelines for designing cloud data pipelines, as the first step in carrying out research in the cloud, balancing performance, affordability, and sustainability. SCORE builds on five years of implementation experience through initiatives such as Utilizing Health Information for Meaningful Impact in East Africa (UZIMA-DS), under the Data Science for Health Discovery and Innovation in Africa (DSI-Africa) Consortium, which revealed the limitations of existing cloud estimation tools and the need for structured, LMIC-oriented decision-making models.[6]

By focusing on three iterative phases, Assessment, Selection, and Optimization, SCORE helps research teams systematically evaluate trade-offs between cost, performance, and environmental impact, offering a scalable model for sustainable cloud adoption in LMIC research ecosystems.

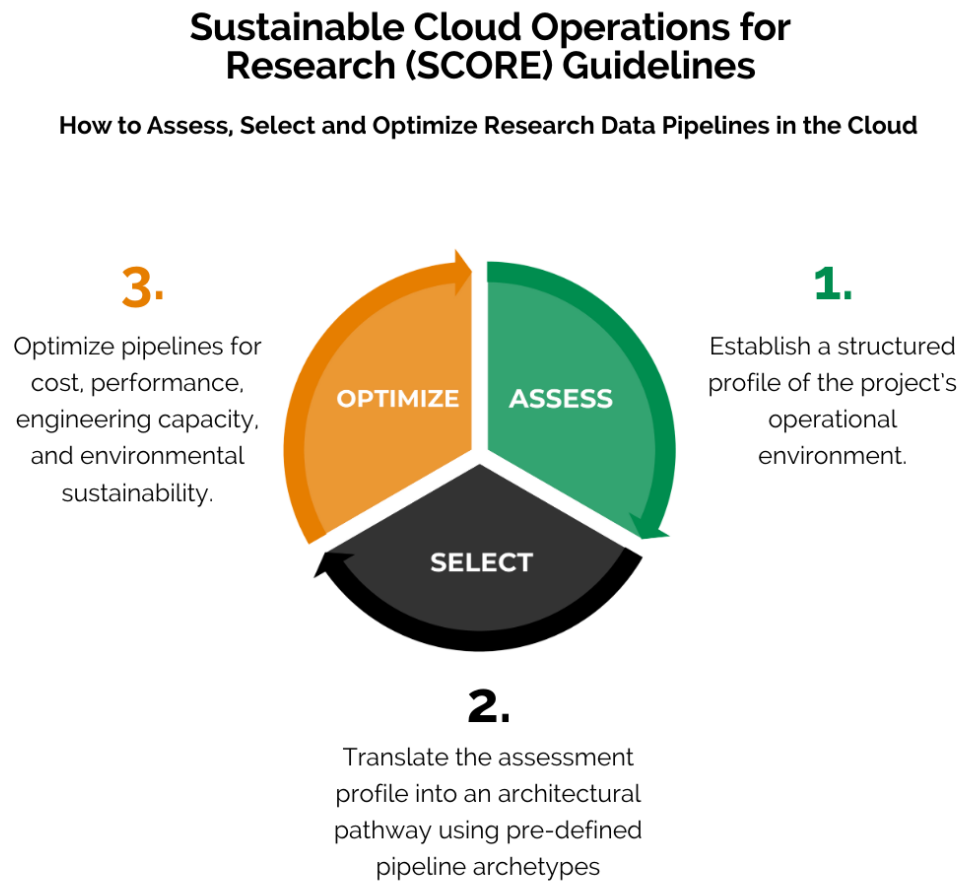
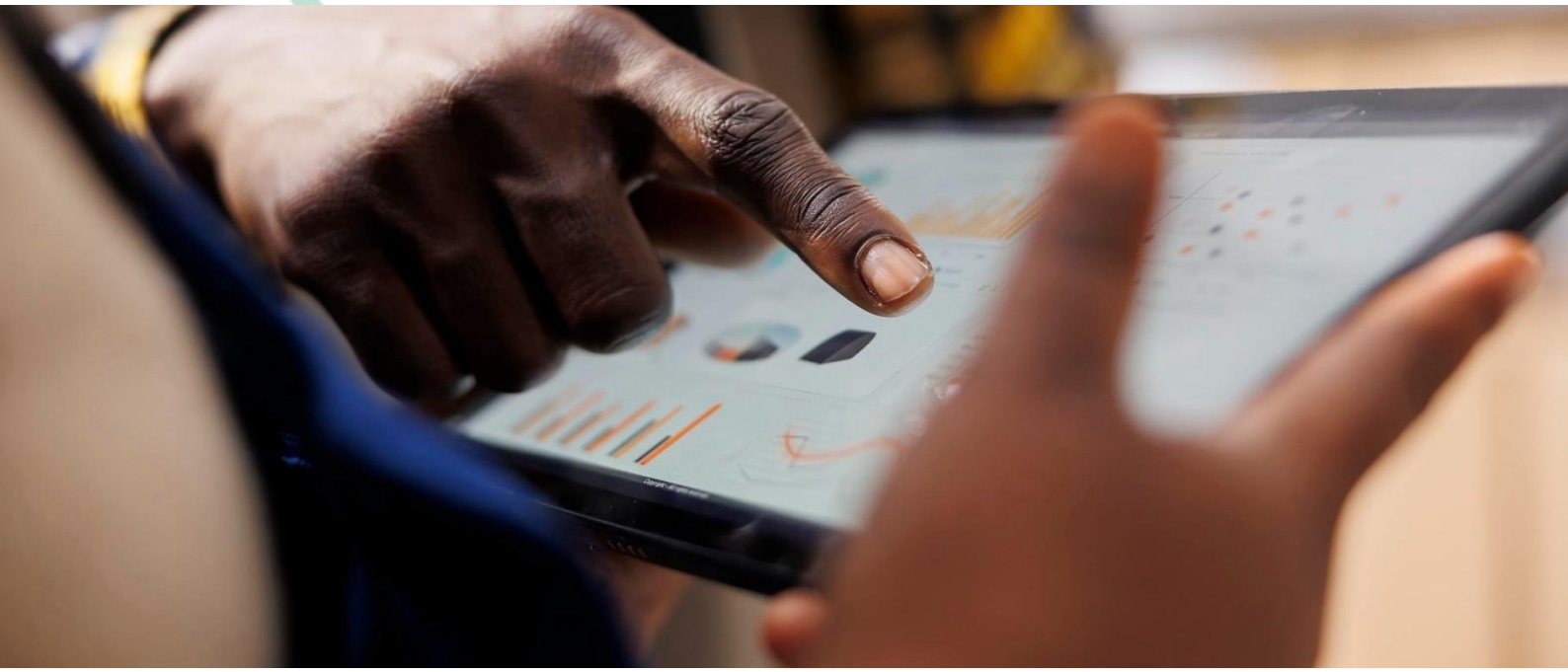


Figure 1: An Overview of the SCORE Guidelines



Sustainable Cloud Operations for Research (SCORE) Guidelines

The Sustainable Cloud Operations for Research (SCORE) guidelines are designed as a practical decision-making model to guide the design, selection and implementation of cloud-based data pipelines in low-resource research settings. It directly addresses the operational constraints faced by teams working in LMICs contexts including: tight budgets, limited DevOps capacity, and intermittent connectivity, by offering a structured process for making infrastructure decisions that balance performance, cost, and environmental sustainability. The SCORE guidelines include the assessment of the operational structure of the data profile, selection of the right ingestion pipeline, and optimization of the data pipelines in the cloud. The guide takes a three-phase approach to systematically evaluate the expected trade-offs of cost, performance, and carbon emissions. The guidelines are built around three phases:

Assessment Phase

The Assessment Phase forms the first step in the Sustainable Cloud Operations for Research (SCORE) framework, providing a systematic method for evaluating the operational environment in which cloud-based data pipelines will function. This phase identifies the contextual factors that influence whether a cloud ingestion model can be deployed effectively, cost-efficiently, and sustainably.

SCORE defines seven design parameters summarized in **Table 1** below, that ensure that LMIC research teams make informed, context-aware choices before implementation, aligning pipeline design with both technical realities and sustainability objectives.

Table 1. Data pipeline parameters for selecting an ingestion model that fits both the technical and organizational context.

Question	Design Dimension	Ingestion-Specific Implication	LMICs Use Case
1. How flexible must the cost model be?	Fixed (strict ceiling), Variable (consumption-based), Mixed.	Cost flexibility affects platform choice. Serverless works well for irregular patterns but may become expensive for large files or high throughput.	Mixed: Serverless ingestion with scheduled batch pre-processing to stay within grant budget.
2. What is the available engineering capacity?	Dedicated DevOps team, Generalist engineers, Low-code or GUI-only teams.	Engineering profile dictates tool complexity. No-code platforms reduce friction but limit control. DevOps teams can implement pipelines using code and CI/CD.	Low-code: Pipelines built with drag-and-drop orchestration tools, minimal scripting required.
3. What is the expected volume of data	Small (KB to MB), Medium (MB to GB), Large (GB to TB), Very Large (>TB).	Volume informs storage format, compression method, and transfer approach. Higher volumes may require chunked uploads, parallel transfer clients, or external drives.	Medium: 6 GB per week, accumulated from edge devices and uploaded nightly.
4. What is the data arrival pattern?	One-time, Periodic, Continuous, Intermittent.	Arrival patterns affect buffer design, queuing, and orchestration triggers. Intermittent uploads require retry logic and local caching.	Intermittent: Daily uploads from field teams using mobile data with frequent signal loss.
5. Is the workload Compute intensive?	No (pass-through only), Light (compression or validation), Heavy (pre-processing or transformation).	Determines whether ingestion can be function-based or if it requires a pre-processing job stage with dedicated compute.	Light: Files compressed and validated before storage, no transformation.
6. What is the frequency of execution?	Scheduled, On-demand, Event-triggered, Continuous.	Frequency shapes architecture. High-frequency or continuous ingest may need autoscaling; infrequent ingest should avoid idle cost via cold-start-optimized functions.	Event-triggered: Upload jobs initiated when new files are detected in a watched folder.
7. What is the state of internet reliability	Stable, Intermittent, Offline (manual sync).	Ingestion must tolerate sync failures. This affects timeout handling, retries, deduplication, and support for out-of-band uploads (e.g. USB).	Intermittent: Uploads fail several times per week and retry automatically using local buffer.

Selection Phase

Once a research team has assessed its context; budget limits, technical capacity, data size, and network reliability, the next step is to select the right data ingestion model. The Selection Phase of SCORE translates those findings into an actionable architectural decision tailored to the realities of low- and middle-income country (LMIC) research environments.

In LMIC projects, cloud spending can quickly exceed expectations due to hidden operational costs such as inter-region replication and network egress fees. For instance, during the setup of the UZIMA-DS cloud research hub, about 30% of the annual cloud budget was consumed by unexpected configuration costs. Studies show that unmanaged data transfer and replication can account for up to 6% of total spending in data-centric cloud environments.[7] These costs often go unnoticed in standard pricing calculators, making it difficult for teams with fixed budgets to experiment safely.

Therefore, pipeline selection in SCORE emphasizes understanding how data moves through the system, including upload frequency, ingestion triggers, and potential points of failure. This pragmatic approach ensures that infrastructure decisions are based on real-world performance and affordability rather than theoretical best practices. Ultimately, SCORE's Selection Phase prioritizes choosing a pipeline that is fit-for-purpose, sustainable, and operable under LMIC constraints, not merely the most sophisticated or feature-rich option.

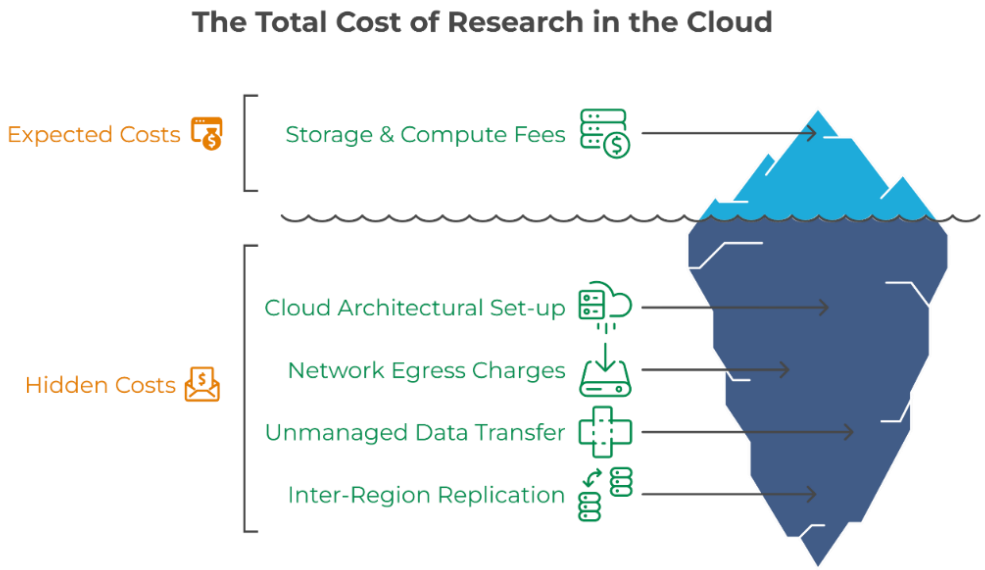


Figure 2:The Total Cost of Research

Table 2. Summarizes common ingestion pipeline archetypes, highlighting their best-fit use cases, trade-offs, and practical relevance for LMICs settings

Pipeline Type	Best Fit	Key Trade-offs	LMICs Use Case
Event-driven serverless (e.g. AWS Lambda, Google Cloud Functions, Azure Functions)	Lightweight ingestion triggered by file uploads, mobile syncs, or sensor pushes. Best for bursty, low-throughput data with cost sensitivity.	Short runtime limits (5 to 15 min), cold start latency, poor for stateful or multi-step logic.	A field survey app triggers a cloud function to upload encrypted form data after each submission. No infrastructure is kept live between uploads.
Low-code orchestration tools (e.g. Azure Data Factory, Google Cloud Data Fusion, Apache NiFi)	Scheduled or drag-and-drop ingestion from known sources. Useful for low-engineering environments with predictable workloads.	Limited control over logic, debugging overhead, connector-based billing can grow expensive. GUI reliance can slow collaboration.	A generalist team uses a visual tool to pull files nightly from shared folders into cloud storage. The job runs unattended on a fixed schedule.
Code-based batch pipelines (e.g. Apache Airflow, custom Python with SDKs, Spark ingestion)	High-volume, compute-heavy or customized data workflows. Allows deep control over retries, transformations, and error handling.	Requires skilled engineers, ongoing monitoring, and tuning to avoid resource waste. Complex to maintain in low-capacity teams.	A research lab ingests large imaging files weekly, transforms them using Spark, and compresses output to Parquet for downstream analysis.
Hybrid orchestration models (e.g. Functions with queues and notebooks, or Logic Apps plus batch processing)	Workloads with mixed frequencies, such as real-time ingestion from mobile devices and batch processing of uploads.	Requires tool integration, independent monitoring, and workflow orchestration. Harder to debug across components.	A telemetry pipeline triggers ingestion from devices in real time while scheduling batch jobs to run weekly for aggregation and cleanup.
Offline-first with batch sync (e.g. IoT edge, USB upload tools, mobile-to-cloud sync bridges)	Sites with intermittent or no connectivity. Data is staged locally and synced periodically. Suitable for edge devices or remote deployments.	Delayed monitoring, risk of data loss if sync fails. Requires retry logic, deduplication, and metadata tracking.	Remote sensors write data to SD cards. Data is uploaded in batches via mobile hotspot or USB connection once the site regains network access.

Optimization Phase

After selecting the right pipeline, the final step in the SCORE framework is the Optimization Phase. This stage focuses on refining performance, minimizing cost, and ensuring environmental sustainability, which are key priorities for research teams operating in resource-limited LMIC contexts.

Cost Optimization

Cost optimization begins with matching infrastructure to actual demand. Serverless tools like AWS Lambda or Azure Functions can eliminate idle costs for intermittent workloads, while reserved instances can reduce expenses by over 60% for continuous or 24/7 operations.[8,9] Data compression techniques such as GZIP or Parquet formats help lower storage and transfer fees, which is especially critical where data movement costs are high.[10] Regularly tagging resources by project or team exposes hidden charges, such as unused virtual machines, often up to 30% of cloud environments remain idle or abandoned, which can be mitigated through scheduled shutdowns.[11]

Performance Optimization

Performance optimization in LMIC environments often revolves around overcoming bandwidth limitations and hardware disparities. Proven strategies include: mitigating cold starts by pre-warming serverless functions, reducing latency by up to 90% for frequently invoked jobs;[12,13] accelerating data transfers using multi-part upload techniques, which can cut transfer times by 70–80% over high-latency networks;[14] using binary data formats like Parquet or Avro to reduce payload size by up to 75%; [15,16] dynamic batching, which adapts to network health, preventing bottlenecks during unstable connections.[17]

Low-Skill Engineering Capacity

In many LMIC research environments, teams have limited DevOps and software engineering skills, making it difficult to manage complex cloud systems effectively. The SCORE framework addresses this by promoting automation and managed services that simplify deployment and maintenance. Tools such as AWS Glue and Azure Data Factory automatically handle infrastructure scaling and monitoring, reducing the need for specialized staff.[14] To ensure consistent and rapid setup, SCORE recommends Infrastructure-as-Code (IaC) tools like Terraform or AWS CloudFormation, which use templates to standardize deployments. Monitoring tools such as CloudWatch and Azure Monitor provide visual dashboards for performance and cost tracking, removing reliance on command-line operations.

By adopting these managed and automated solutions, LMIC teams can maintain efficient, reliable, and cost-effective data pipelines without extensive technical expertise, strengthening long-term sustainability and scalability.[14]

Through these optimizations, SCORE ensures that cloud operations remain affordable, efficient, and environmentally responsible, making large-scale data science feasible in LMIC research ecosystems. Once a pipeline type is selected, this phase introduces targeted strategies for tuning performance, reducing cost, improving resilience, and minimizing environmental impact. After selecting pipeline architecture, apply the following prescriptive techniques to optimize cost, performance, resilience, and environmental impact. These are contextualized for research operations under LMICs constraints.

Table 3. Summarizes guidelines for cost, performance and low-skill engineering capacity optimization, giving context on when to apply these techniques and practical relevance for LMICs settings.

	Technique	When to Apply	Tactic	LMICs Use Case
Cost Optimization	Use serverless compute to avoid idle costs	For event-driven or bursty jobs	Use Azure Functions or Data Factory with consumption tier.	Sensor data from heart monitors
	Reserve compute only for consistent workloads	When usage exceeds 8–12 hrs./day or is 24/7	Use Reserved Instances or pre-warmed Spark pools.	Incoming EHR data
	Compress data before ingestion	Always	Use GZip or Parquet to reduce storage and egress fees.	Integrating an EHR system with a cloud service
	Avoid always-on dev/test resources	During development	Use auto-shutdown policies on dev VMs and sandboxes.	When using multiple cloud-based virtual machines
	Enable granular cost tagging	Always	Tag by pipeline stage, dataset, and team for audit-ready billing.	When managing multiple data pipelines/teams
Performance Optimization	*Pre-warm serverless	Frequent ingestion (>5/min)	Timer-triggered invocations; AWS Lambda Provisioned Concurrency	Real-time sensor telemetry
	Multi-part transfers	Files >100MB; high latency	AWS S3 Transfer Acceleration; Azure Blob parallel upload	Satellite imagery ingestion
	Binary serialization	Structured data ingestion	Parquet/Avro encoding; Protocol Buffers	Clinical trial data collection
	Adaptive batching	Unstable networks	AWS Kinesis dynamic batching; Azure Stream Analytics watermarking	Flood sensor networks
Low-Skill Engineering Capacity Optimization	Managed ETL services	Limited DevOps skills	AWS Glue, Azure Data Factory (serverless)	Genomics metadata ingestion

**Pre-warming serverless functions or pre-allocated Spark clusters may incur minimal constant cost. In LMICs projects with strict ceilings, weigh costs against latency benefits.*

Environmental Sustainability

Cloud computing has enabled rapid digital growth but also contributes significantly to global energy consumption and greenhouse gas emissions.[2,18] The SCORE framework emphasizes environmentally responsible design to balance performance with sustainability. Efficient data pipelines can reduce carbon impact by optimizing how and when computing resources are used. SCORE recommends scheduling workloads during low-carbon intensity periods using tools like Azure Emissions Dashboard or WattTime APIs, and prioritizing data centers powered by renewable energy. Regular workload profiling ensures that compute resources are "right sized," preventing waste from over-provisioned virtual machines. Tracking emissions through platforms such as *Microsoft Sustainability Manager* supports transparency and accountability.

By integrating these green engineering practices, LMIC research teams can reduce operational costs while minimizing environmental harm, advancing both **scientific and ecological sustainability** within their data-driven health research operations.[2,18]

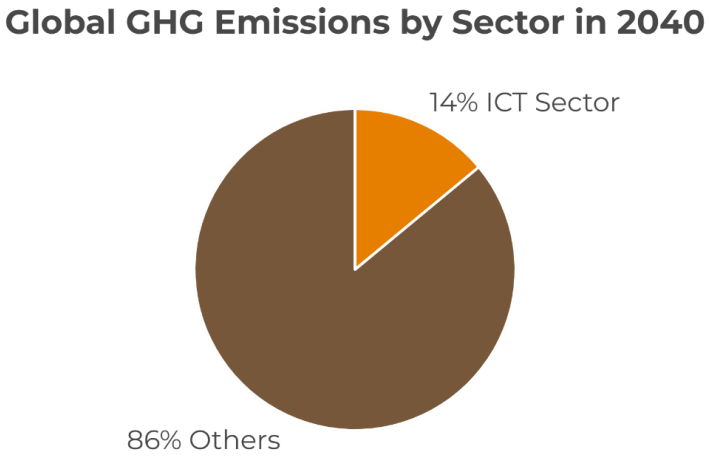


Figure 3: Global GHG Emissions per sector by 2040. Source: Hitesh Allam, Sustainable Cloud Engineering: Optimizing Resources for Green DevOps. International Journal of Artificial Intelligence, Data Science, and Machine Learning (2023)

Testing our Concept

Approach

To evaluate the practical relevance of the SCORE framework, we conducted an applied test simulating real-world cloud-based research conditions. The objective was to determine whether SCORE's recommendations could guide the design of cost-efficient and environmentally sustainable data ingestion pipelines in a low- and middle-income country (LMIC) context.

The test used the NIH Chest X-ray14 dataset, a large open-access dataset comprising 112,120 chest radiographs from 30,805 patients, totaling approximately 42 GB.[19] This dataset was selected because of its substantial size and suitability for benchmarking performance, even though its content was not directly relevant to the study's health outcomes. The experiment focused on a single, repeatable task: data ingestion, moving data from a source repository into Azure cloud storage. To produce consistent and reliable metrics, the ingestion process was repeated ten times under the same conditions.

Three Azure-native tools; Synapse Pipelines (no-code), Azure Functions (serverless), and Synapse Notebooks (code-based), were compared using the same dataset and parameters. To capture outcomes, the study employed Azure Cost Management to measure financial efficiency, Azure Monitor / Log Analytics to assess performance (execution time, CPU, and memory usage), and the Azure Carbon Optimization Tool to estimate emissions. Together, these tools provided a comprehensive view of the relationship between cloud architecture choices, cost, and environmental impact, validating SCORE as a practical, data-driven approach to sustainable cloud operations for LMIC research environments.

Table 4. Gives a description of each resource group and its purpose

Approach	Description	Purpose
Synapse Pipelines (No-Code)	A no-code data integration and orchestration tool within Azure Synapse, allowing workflow automation without extensive coding.	Evaluates efficiency of a fully managed, no-code orchestration system with automatic execution optimizations.
Synapse Notebooks (Code, Python)	A code-based interactive computing environment using Python, enabling custom scripting and fine-tuned control over data processing. Requires mounting to a compute resource.	Assesses resource usage and performance when users have direct control over the back-end code and computation.
Azure Functions (Serverless)	An event-driven, serverless compute service that dynamically scales based on workload demand, eliminating idle resource costs.	Explores efficiency of a fully serverless model where resources are allocated dynamically, reducing potential idle usage costs.

Although our three primary metrics were cost efficiency, time efficiency, and environmental impact, additional metrics such as write/storage costs and geo-replication v2 data transfer were also considered as they demonstrated a significant bearing on the overall costs **(Table 5).** Execution time was collected immediately after task completion for each approach. Cost and carbon emissions were retrieved from Azure once the monthly data was released.

Table 5. Primary metrics and associated measurements and considerations

Metric	Definition	Measurement	Measurement Tool	Considerations
Cost Efficiency	Reduction in operational costs without compromising performance.	Cost = Usage Quantity (standard) × Unit Price (non-standard).	Azure Cost Management + Billing	Unit price varies due to negotiated pricing, demand, location, etc. Actual costs may be under NDAs, limiting transparency.
Time Efficiency	Reduced execution time for data processing tasks.	Execution time measured from task initiation to completion.	Azure Monitor / Log Analytics	Measured in seconds/milliseconds; variability introduced by cold starts, system overhead, and queue wait times, i.e., may not measure CPU runtime.

Environmental Impact	Decrease in carbon emissions and energy consumption.	Carbon Emissions = Energy Consumption (standard) × Carbon Intensity (non-standard).	Azure Carbon Optimization Tool	Carbon intensity depends on grid mix, data center location, and regulations. Azure aggregates emissions at the monthly and resource group level, reducing granularity.
Write/Storage Cost	Charges incurred when data is added /updated and/or stored in the pipeline.	Based on the total volume of data written and stored (in GB).	Azure Cost Management + Billing	Rarely accessed data can accumulate significant charges if not optimized.
Geo-Replication v2 Data Transfer	Movement of data between geographically distributed Azure regions under the Version 2 geo-replication model.	Billing is per GB of data replicated across regions.	Azure Cost Management + Billing	The volume and frequency of data changes, the distance between regions, and the chosen storage tier have an impact on the overall cost.

*The hierarchy for Azure resources is as follows: Subscription > Resource Group > (Resource) Azure Synapse Analytics > Pipeline > Activity.

Outcome Metrics and Results

The Azure Functions (Serverless) approach delivered the best overall performance, recording the lowest cost (USD 0.01), lowest carbon emissions (0.00003 kg CO_2e), and shortest execution time (1.12 hours). This efficiency was attributed to serverless computing's ability to allocate resources dynamically, only when needed, avoiding idle costs. The Synapse Pipelines (No-Code) model performed moderately well, costing USD 1.68, emitting 0.01873 kg CO_2e, and taking 6.9 hours to complete. Its visual, managed environment simplified orchestration but added some latency due to additional backend processes like scheduling and logging.

In contrast, the Synapse Notebooks (Code-Based) approach was the least efficient, costing USD 15.20 and generating 0.16901 kg CO_2e, with a runtime of 5.74 hours. Most of these costs were driven by a dedicated Spark pool, which remained active even when idle. Additionally, geo-replication accounted for roughly 85% of storage costs, underscoring how data movement and redundancy can significantly affect budgets. Overall, these results confirm that serverless ingestion pipelines, when aligned with SCORE guidelines, offer the most sustainable and cost-effective approach for LMIC research operations.

Table 6. Performance Metrics for One-Time Data Ingestion and Storage and Data Transfer Costs

	Approach		
	Synapse Pipelines (No-Code)	**Azure Functions (Serverless Code)**	**Synapse Notebooks (Code, Python)**
Cost (USD)	1.68	0.01	15.20*
Carbon Emissions (kg CO_2e)	0.01873	0.00003	0.16901
Execution Time (hours)	6.90	1.12	5.74
Write/Storage Cost (USD)	1.56	1.75	1.55
Geo-Replication v2 Data Transfer (USD)	8.73	9.61	8.73

*The cost for Synapse Notebooks includes $0.01 for the Synapse Workspace and $15.19 for the Dedicated Spark Pool.

Discussion

The results of the SCORE proof of concept challenged common assumptions about the efficiency and sustainability of cloud-based data ingestion methods. The research team initially expected that code-based pipelines, such as Synapse Notebooks, would be the most cost-effective and environmentally sustainable due to their flexibility and potential for code-level optimization. However, the findings revealed the opposite: dedicated compute environments like Spark pools substantially increased both cost and carbon emissions, accounting for nearly all of the expenses in that approach.

This pattern mirrors ongoing research within LMIC research environments, where compute-intensive and security-heavy resources are often the largest contributors to both financial and environmental overheads. These insights highlight the importance of careful infrastructure planning and demonstrate that high-performance configurations are not always the most sustainable or affordable choices in resource-limited contexts. Conversely, serverless models, specifically Azure Functions, outperformed other configurations in every key metric: lowest cost (USD 0.01), lowest carbon emissions (0.00003 kg CO_2e), and shortest execution time (1.12 hours). Serverless architectures dynamically allocate computing power based on demand, eliminating idle resource costs and improving both economic and environmental efficiency.

Interestingly, storage costs remained fairly consistent across all methods, but geo-replication accounted for roughly 85% of those costs, emphasizing how easily overlooked pricing factors can inflate budgets. Teams frequently underestimate egress and replication fees, assuming these services are low-cost or included. The study's results reinforce the need for clearer understanding of cloud

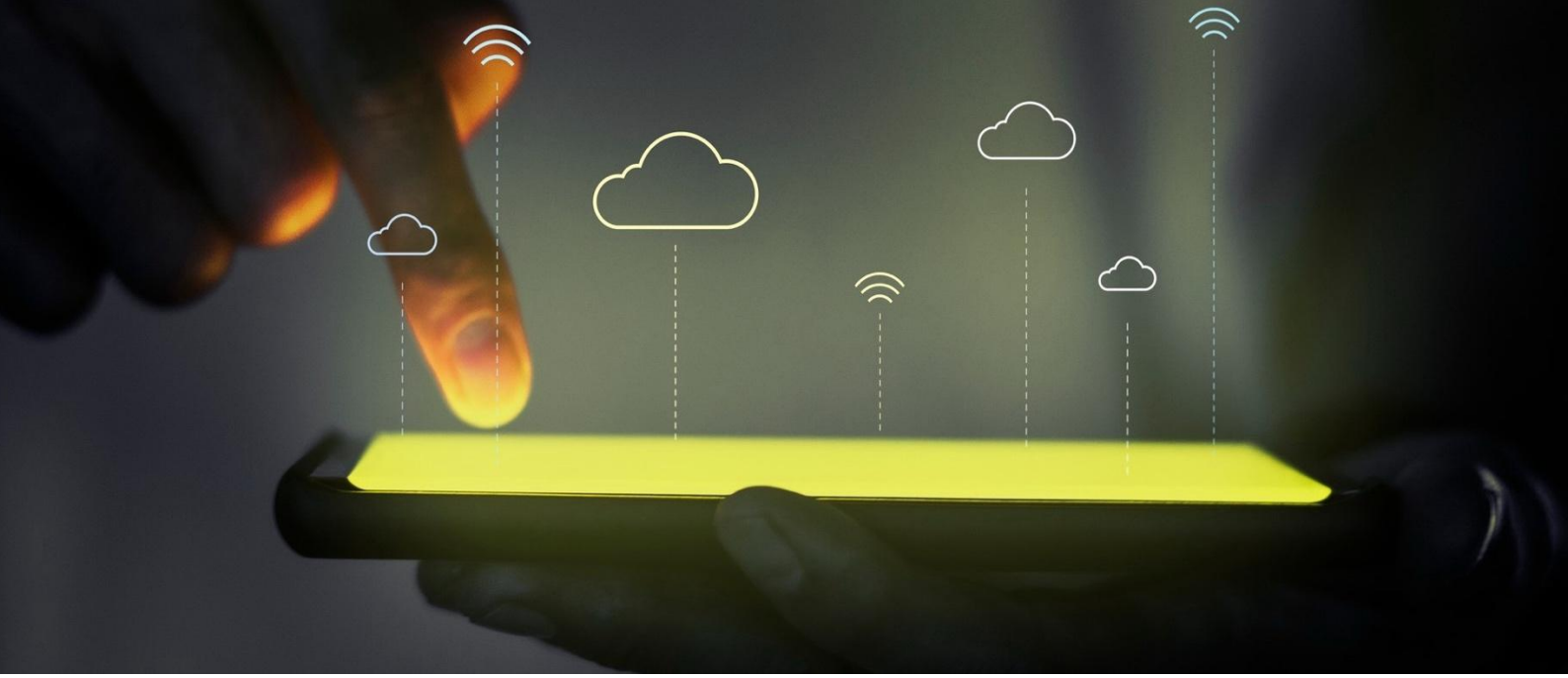
provider pricing models and for transparent budgeting frameworks suited to LMIC research institutions. Overall, the results affirm SCORE's value as a context-aware guideline that enables researchers to make evidence-based infrastructure decisions, balancing performance, cost control, and environmental responsibility across diverse research environments.

Considerations and Future Directions

While the SCORE framework offers a structured, evidence-based approach for building sustainable cloud operations in LMIC research environments, several considerations must guide future refinement and broader implementation. The current validation was conducted exclusively on Microsoft Azure, meaning that results may vary across other major platforms such as AWS and Google Cloud Platform (GCP), where architecture, pricing, and sustainability tools differ. Expanding SCORE testing across multiple cloud providers will help confirm its generalizability and reveal platform-specific nuances that affect cost and emissions.

Additionally, the validation focused primarily on data ingestion workflows. To fully evaluate SCORE's versatility, further studies should include more complex pipeline stages, such as data transformation, analysis, and GPU-intensive machine learning workloads. Such tasks often behave differently under variable compute and storage conditions. Another limitation involves the granularity of emissions data. Azure's current tools aggregate carbon metrics monthly at the resource-group level, which makes it difficult to capture emissions from individual tasks in real time. Until finer-grained tracking becomes available, LMIC teams may need to rely on proxy metrics for environmental optimization.

Finally, successful SCORE adoption depends on capacity building. Many research teams in LMICs have limited cloud literacy, emphasizing the need for training programs, documentation, and community-based support systems to ensure practical implementation.



Conclusion

This paper introduces SCORE guidelines that aim to enable LMIC research teams to navigate the triple constraints of cost efficiency, technical feasibility, and environmental sustainability in cloud-based data pipelines. Through a structured three-step approach, Assessment, Selection, and Optimization, SCORE addresses critical gaps in existing cloud guidance by contextualizing, quantifying tradeoffs, prioritizing practicality, and advancing sustainability. Future work should expand SCORE's validation to multi-cloud environments, other LMICs deployments, and advanced pipeline stages (e.g., distributed ML training). Integration with open-source monitoring tools could further reduce dependency on proprietary solutions. By democratizing sustainable cloud practices, SCORE empowers LMICs researchers to leverage cloud infrastructure as an equitable, scalable, and ecologically conscious foundation for scientific advancement.

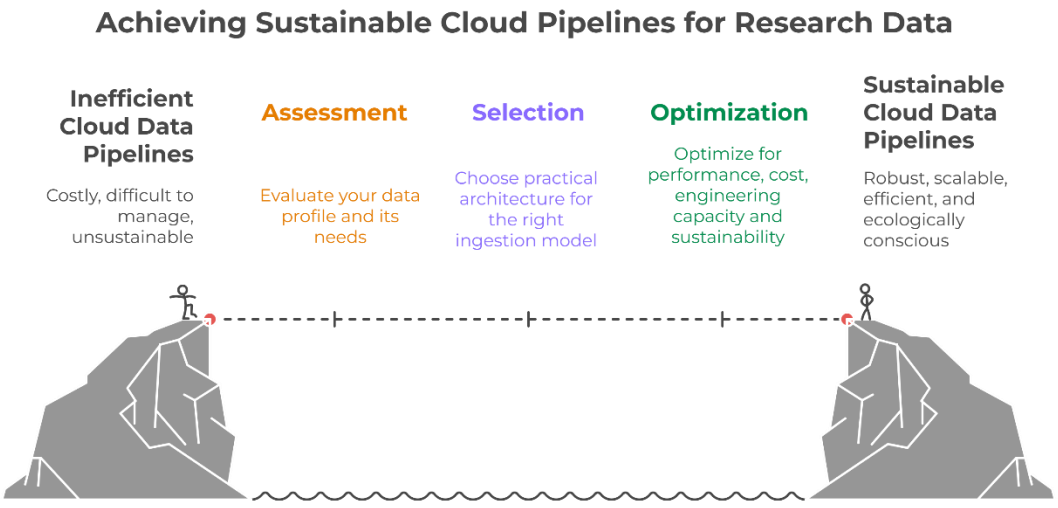


Figure 4: Bridging the Cloud Research Gap in LMICs

References

1. Kumar Pentyala D. Enhancing the Reliability of Data Pipelines in Cloud Infrastructures Through AI-Driven Solutions. Vol. 6, An International Peer Review Journal) VOlUME. 2020.

2. Allam H. Sustainable Cloud Engineering: Optimizing Resources for Green DevOps. International Journal of Artificial Intelligence, Data Science, and Machine Learning [Internet]. 2023;4:36–45. Available from: https://ijaidsml.org/index.php/ijaidsml/article/view/179

3. Franzen SRP, Chandler C, Lang T, Samuel D, Franzen RP. Health research capacity development in low and middle income countries: reality or rhetoric? A systematic meta-narrative review of the qualitative literature. BMJ Open [Internet]. 2017;7:12332. Available from: http://bmjopen.bmj.com/

4. Monserrate SG. The Staggering Ecological Impacts of Computation and the Cloud [Internet]. The Mit Press Reader. 2022 [cited 2025 Jul 10]. Available from: https://thereader.mitpress.mit.edu/the-staggering-ecological-impacts-of-computation-and-the-cloud/

5. Amazon Web Services. Sustainability Tools [Internet]. Amazon Web Services. 2025 [cited 2025 Jul 10]. Available from: https://aws.amazon.com/sustainability/tools/

6. Deochake S. Cloud Cost Optimization: A Comprehensive Review of Strategies and Case Studies. 2023 Jul 24; Available from: http://arxiv.org/abs/2307.12479

7. Fluence. 5 Case Studies of Cloud Egress Fee Reduction and Slashing Data Costs [Internet]. Fluence. 2025 [cited 2025 Jul 10]. Available from: https://www.fluence.network/blog/5-case-studies-of-cloud-egress-fee-reduction-and-slashing-data-costs/

8. Amazon Web Services. Cost Optimization Pillar - AWS Well-Architected Framework. 2023;

9. Microsoft. Azure Reserved Virtual Machine Instances [Internet]. 2025 [cited 2025 Jul 10]. Available from: https://azure.microsoft.com/en-us/pricing/reserved-vm-instances/

10. Pintaux N. Well-Architected Framework: Cost optimization pillar [Internet]. Google. 2024 [cited 2025 Jul 10]. Available from: https://cloud.google.com/architecture/framework/cost-optimization

11. Mazumdar S, Pranzo M. Power efficient server consolidation for Cloud data center. Future Generation Computer Systems. 2017 May;70:4–16.

12. Snyder P, Vastel A, Livshits B. Who Filters the Filters: Understanding the Growth, Usefulness and Efficiency of Crowdsourced Ad Blocking. Association for Computer Machinery [Internet]. 2020 Jun 12 [cited 2025 Jul 10]; Available from: https://dl.acm.org/doi/10.1145/3392144

13. Mahgoub A, Yi EB, Shankar K, Elnikety S, Chaterji S, Bagchi S. Orion and the Three Rights: Sizing, Bundling, and Prewarming for Serverless DAGs ORION and the Three Rights: Sizing, Bundling,

and Prewarming for Serverless DAGs [Internet]. Available from: https://www.usenix.org/conference/osdi22/presentation/mahgoub

14. Yildirim E, Kosar T. End-to-End Data-Flow Parallelism for Throughput Optimization in High-Speed Networks. J Grid Comput. 2012 Sep 10;10(3):395–418.

15. Maltsev E, Muliarevych O. Beyond JSON: Evaluating Serialization Formats for Space-Efficient Communication. Advances in Cyber-Physical Systems. 2024 May 10;9(1):9–15.

16. Morschel L, Adeyemi O, Garonne V, Litvintsev D, Millar P, Mkrtchyan T, et al. Efficient Message Encoding For Inter-Service Communication in dCache: Evaluation of Existing Serialization Protocols as a Replacement for Java Object Serialization. EPJ Web Conf. 2020 Nov 16;245:05017.

17. Amazon Web Services. Auto Scaling for AWS Glue ETL and streaming jobs. AWS Glue Developer Guide [Internet]. [cited 2025 Aug 1]. Available from: https://docs.aws.amazon.com/glue/latest/dg/auto-scaling.html

18. Raza M, KS S, K S, Mohamad A. Carbon footprint reduction in cloud computing: Best practices and emerging trends. International Journal of Cloud Computing and Database Management. 2024 Jan 1;5(1):25–33.

19. Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases [Internet]. Available from: https://uts.nlm.nih.gov/metathesaurus.html