

Energy-Efficient System Design for High-Volume Insurance Applications in Cloud-Native Environments

Keerthi Amistapuram

Net Developer

ORCID ID: 0009-0009-6408-1958

Abstract. Energy-efficient system design for high-volume cloud-native insurance applications has gained much attention from academia and practitioners over the past years. Energy-efficient design cannot be achieved through isolated optimization of single components without raising energy inefficiencies and reducing resilience. Energy efficiency must be considered holistically across architectural layers and the entire solution life-cycle whenever green software solutions are designed and constructed. Transportability and workload requirements play an important role in this context as these can have shaped energy and cost-efficient solutions with little variation across several products. Energy efficiency is particularly relevant during off-peak times when components should not just be scaled back, but rather spin down in order to minimize a potential negative influence on customer satisfaction. Reduced costs, resource savings through green software solutions, and meeting regulations such as the European GDPR or the SEC's new climate rules also contribute to the corporate agenda. However, energy consumption reporting remains an unsolved issue in the cloud industry. Insufficient cost-to-energy metrics and the lack of regulatory compliance supporting these calls underline the need for a set of blueprints focusing on energy efficiency in high-volume cloud-native insurance systems.

Keywords: Green software, cloud-native, autoscaling, data tiering, server-less, energy efficiency, insurance workloads, Cloud-Native Architecture, Energy Efficiency Optimization, Green Computing, Scalable Microservices, Workload Orchestration, Cost-to-Energy Ratio, High-Volume Transaction Processing, Sustainable IT Infrastructure, Dynamic Resource Allocation, Insurance Workflow Automation.

1 INTRODUCTION

Cloud-native designs ease the implementation of high-volume insurance workloads as usage and resource demand scale. Cloud providers offer pay-per-use pricing schemes but often rely on oversubscribed shared resources, wasting energy on idle instances; inefficient workloads further exacerbate energy waste. To contain costs, many organizations attempt energy-aware cloud strategies. However, electricity prices may not align with the cost of necessary resources, and energy usage often remains hidden and ungoverned behind clouds and containers. Regulatory frameworks such as the E.U. Taxonomy and the E.U. Corporate Sustainability Reporting Directive push for more transparent cost-to-energy ratios for customers, and insurance players are particularly sensitive to energy since many contracts cover natural catastrophes. Green software guidelines propose energy-aware design patterns, supporting cost-to-energy consciousness by revealing energy usage. Services such as Google Cloud's Sustainability Dashboard track resources for energy measurement. Therefore, these cost-to-energy regulations require careful consideration of cloud governance. While latent costs on idle large elastic compute (e.g., infrequent pricing or events) may not be energy-optimal, patterns such as storage and bandwidth tiering or per-use short-term resource consumption may yield significant savings. Recommendations on optimizing the energy efficiency of high-volume cloud-native insurance platforms should tackle the specific mechanical properties of the sector and offer design decisions addressing resource consumption, underlying technology for storage, and telemetry services enabling cost-to-energy governance in a manner aligned with existing patterns of green software.

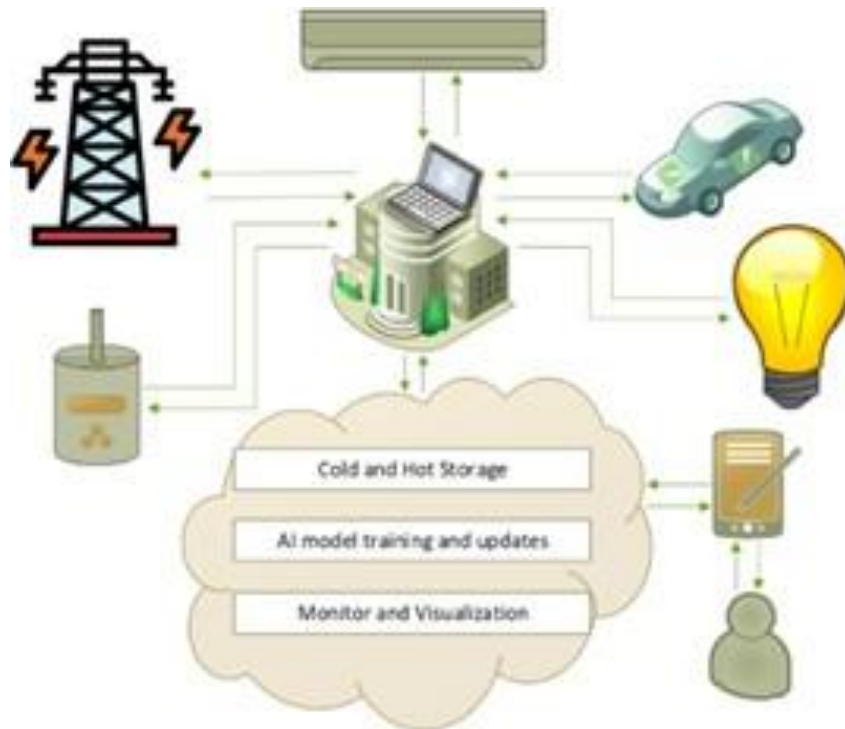


Fig. 1. Enhancing Building Energy Management

1.1 Problem Statement and Relevance

A cloud-native architecture for high-volume insurance workloads is typically designed for service availability and cost efficiency. However, these metrics do not directly consider energy supply or consumption in a cloud environment. Cost and energy can be seen as inverse aspects of the same global metric. An orchestration service is unaware, for example, that it incurs high costs when scaling out but consumes little power when scaling in. Under these conditions, reducing energy consumption becomes a major challenge. Societal pressure to report the carbon footprint of technology services, the increasing cost of energy, and news that data centers can be responsible for about 3

Scope and Definitions From an architectural perspective, the focus is on the Cloud Compute tier. Fortunately, innovative Cloud-IaaS providers offer hyper-scale Cloud services. However, actually saving energy requires more than buying such services. Conventional practices like perpetual over-provisioning and continuous operation (for fault-tolerance) can incur significant waste. The discussion therefore revolves around achieving the best possible Cost-to-Energy ratio for Infrastructure-as-a-Service-based Cloud services in high-volume, time-critical Insurance-scoring systems. The analysis indicates that improving Compute Efficiency by reducing other Cost-to-Energy components was insufficient. Cost-to-Energy only halved, resulting in waste of hundreds of thousands of Euros per year. Regulatory Energy-Use Reporting added further pressure for action. Cost-Energy reporting computed a Cost-to-Energy ratio since Cost-to-Customer was already available. Actual Costs of Insurance-scoring production workloads provided data for the 1st time, enabling the introduction of Energy-Awareness into the architecture, objectives, and Operation-Management Policies. Automating data locality decisions, continuously moving Data into the most cost-effective storage Tier, and hosting the HOT and WARM service in a co-located manner maximized Energy Efficiency. Against intuition, also preserving the data- replication Police of Scaling for Fault-tolerance, Judging Energy-Awareness and Energy-Efficiency patterns, SKETCHED pattern-Kites, SWH provide Practical design Patterns for Energy-Efficiency in high-volume Insurance-surveillance Systems developed in Cloud-Native Architectures.

2 FOUNDATIONS OF ENERGY-EFFICIENT CLOUD-NATIVE ARCHITECTURE

An effective architectural design is driven by a small set of foundational concepts. For energy-efficient cloud-native systems supporting fast-reacting high-volume insurance workloads, the following principles warrant particular attention. Green Software Principles. The design benefits from the principles of green software, one of which—power-aware

design—is associated with the general principle of reducing waste. Energy-related waste encompasses not only the direct consumption of excessive power during idle moments, which only manifests in workloads demanding high geographic coverage from their cloud backends, but also, indirectly, other costs that rise nonlinearly with load. An obvious example is energy costs, which typically account for only a small fraction of total operations expenses in application segments with lower cloud usage but assume greater importance as operational scales increase. In highly regulated industries such as insurance, growing regulatory concern about the environmental impact of digital services and demands for external disclosure of energy consumption (or at least carbon footprint)—such as Germany’s Energy-Consumption Label for Servers, which displays performance and electricity consumption metrics on a common scale—is reinforcing the focus on energy-efficient design and operation of cloud-native services. Cloud-native Architecture Patterns. Several well-established cloud-native architecture patterns contribute directly or indirectly to the energy efficiency of supporting platforms. Autoscaling and the sidecar pattern can reduce power waste during periods of low load and size transient resource requirements to match actual demand. The event-driven pattern can significantly help reduce compute-related energy consumption costs of all kinds, including the storage energy costs indirectly associated with excessive compute usage. Finally, the use of cloud IaaS platforms with a broad geographic coverage offers the opportunity to optimize data location and thus minimize I/O energy expenditure in widely distributed systems.

Equation 01: Definition (from first principles)

Let each component $i \in \{\text{compute, storage, network, overhead}\}$ contribute cost C_i and energy E_i

$$CER \equiv \sum_i E_i / \sum_i C_i \text{ [$/Wh]} \text{ With per-unit prices/coefficients } C_i = p_i q_i, E_i = e_i q_i \text{ This aligns with the artic}$$

(1)

Utilization	RPS	Container Instances	Per Instance Util
0.05	100	1	0.4
0.1	200	1	0.8
0.15	300	2	0.6
0.2	400	2	0.8
0.25	500	2	1
0.3	600	3	0.8
0.35	700	3	0.933333
0.4	800	4	0.8
0.45	900	4	0.9

Table 1. Cost-to-Energy Ratio (serverless vs containerized)

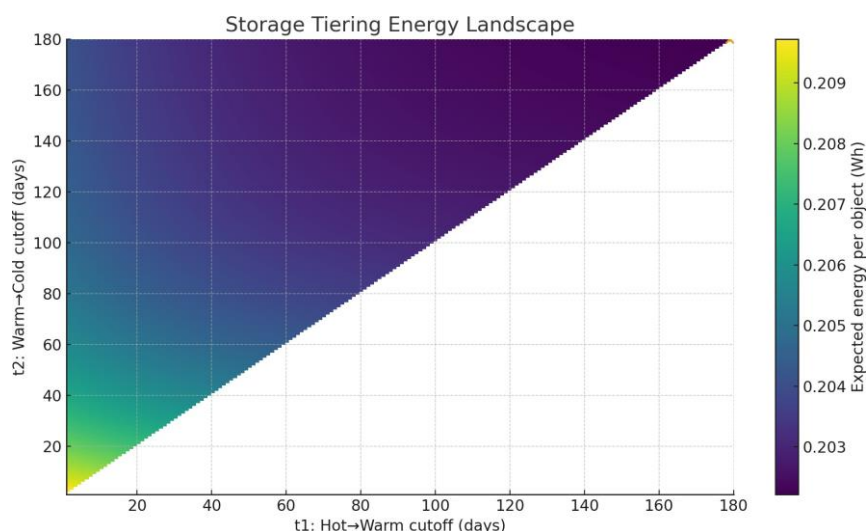


Fig. 2. Storage Tiering Energy Landscape

2.1 Principles of Green Software

Four principles will guide the architectural design into cloud-native systems with energy-cost efficiency. First, approach software design, development, and deployment using details and patterns that reduce the waste of resources, energy, and carbon during processing, storage, and transmission by the application. Most patterns exist, but they are not always considered due to lack of awareness or expertise. Second, consider energy, carbon, and cost as first-order metrics; consequently, autoscaling time windows must be sufficiently short even when operating in batch or near-batch mode. The energy of a high-volume insurance workload is composed of the individual energy cost of service layers and the shared power consumed by the cloud provider due to the capacity reserved by the workload. Therefore, minimizing power consumption while considering the underlying service layer costs is key to maximizing the energy-cost efficiency metric. A third guiding principle is to define a set of exhaustively documented and implemented measurement practices that allow for continuous evaluation and quantification of the energy, carbon, and cost metrics associated with the application. Finally, the metrics shall feed optimization loops capable of directing the application development and infrastructure configuration toward greater energy-cost efficiency. Hence, combinations of all determinants of energy, carbon, and cost undershoot shall conclude with well-defined and balanced autoscaling policies before official release.

2.2 Cloud-Native Patterns for Efficiency

Conventional wisdom contends that shifting workloads into the cloud naturally curtails energy utilization. This idea arises from the smart, meticulous, and flexible management of the resources in data centers. Resources in cloud systems are managed in an efficient manner, utilizing the automatic elasticity facility provided by scale-down techniques when the demand is low. However, a thorough analysis of conventional cloud-native architecture indicates a power-hungry architecture design. Such architectural designs are therefore not suitable for energy-sensitive systems. Moreover, many endeavors concentrate solely on enlarging the business, with little consideration for the environment. Green software engineering considers such issues and thus establishes four partly overlapping tenets of power-aware software: power-aware design, power waste reduction, measurement of power utilization, and optimization around achieved measurement. These principles have an impact on system design by examining the classical speedup theorem, which can be illustrated as follows. Nodes in a classical cloud system are either in an active state or a sleeping state with zero power overhead. The power conservation theorem states that a constant idle power consumption incurs no power-overhead penalty in terms of processing speedup. Moreover, an energy-efficient cloud-native architecture design specifies three architecture design patterns. It is necessary to examine the architectural patterns that fulfill the requirements in terms of cost, speed, and energy. In this context, dynamic autoscaling, the sidecar pattern, an event-driven computing architecture, and data-locality-aware event-driven architecture play crucial roles.

3 WORKLOAD AND DATA MANAGEMENT IN HIGH-VOLUME INSURANCE SYSTEMS

Architecture (see Section 4) is intimately connected to the workload characteristics and data management of a system. The workload characteristics include who generates the data when it is generated, how the data is stored—or in which tier it is stored—and how the data is processed. Two patterns can then be recognized in how such systems handle workloads and data: they can process workload and data in real time or in batch; and they can have an active data management strategy that follows the data lifecycle in a way that matches the changing data states to the tier that is optimal from a cost-and-latency perspective. Properly designing workload and data management makes it possible to optimize energy consumption.

3.1 5.1. Batch vs. Real-Time Processing

High-volume insurance workloads generate a large quantity of data, and generating data in large amounts is intrinsically latency insensitive. It is often possible to accept latencies of several minutes or even longer. Processing the workload with an event-driven batch process rather than in real time can therefore greatly improve the energy efficiency of the process. Processing the workload in real time leads to a much smaller latency. To benefit the most from a real-time process, it must be built on small amounts of data being processed at the same time. When flushing events into a data lake, it is important to flush events by session or by other means to make batch-processing jobs more efficient.

3.2 5.2. Data Lifecycle and Tiering

Whether data is stored in hot storage, warm storage, or cold storage has a significant impact on energy consumption. Transferring data between tiers also consumes energy, but this can be acceptable because of the savings achieved by moving data to an appropriate tier. Adding a data-management process that follows the data lifecycle allows savings by moving data to the correct tier. Data that has newly arrived before the closing of the process is stored in hot storage and



directly ingested by the data-processing jobs; data that is after the closing of the process can be transferred to warm or even cold storage, and ignored data can be deleted.

Fig. 3. Data Management Techniques

3.3 Batch vs. Real-Time Processing

Workload and Data Management in High-Volume Insurance Systems Designing energy-efficient systems requires understanding both workload characteristics and workload handling. In high-volume insurance systems, activity proceeds in two modes: real-time and batch. Both are necessary, and the tendency to overemphasize real-time processing should be avoided. Real-time processing generates immediate output, typically in the form of automated alerts and emails to users about key issues like parametrization errors in data-acquisition processes. Ideally, latencies should range from a few seconds to a couple of minutes. Batch processing consolidates accumulated events, sensors, etc. through time. By definition, some latency in generating the output is acceptable—typically minutes to hours for low-severity alerts, hours to days for periodic monitoring output, and days, weeks, or months for the generation of detailed figures like the full insurance indicators for the previous year. From an energy perspective, the trend is clear: batch activity usually consumes less energy and may involve a much larger scale of processing, enabling better energy efficiency of the underlying systems. For high-volume insurance systems, however, real-time activity is not just a novelty that can be put aside in favor of more energy-efficient batch processing.

3.4 Data Lifecycle and Tiering

Most insurance data has a predictable lifecycle: customers document their property to apply for insurance, subsequently use that property, and then retire it. Therefore, the recorded data can usually be partitioned into four lifecycle states: pre-event (when the insurance was not yet needed, as the customer had not yet applied), operation (when the insurance service was in use), post-event (when the event had already happened, and the customer was in the process of submitting a compensation application), and also when the insurance compensation was under analysis), and termination (after the agreement ended or was no longer active). Often, customers do not use certain coverages over long periods, so there is little data for those services. Therefore, the older the data is, the less active it tends to be. Nevertheless, backups must be preserved and retained to meet legal requirements. The data that insurances need to keep has different usages, so the storage media used must also be different. The data that are not being consulted frequently can be moved to less expensive storage media with higher access latency. Insurers should always try to reduce costs, not just for profit, but because it is a requirement by SOS-LIA. Users consume data, but they also want to reduce the expenditures from their own businesses. The data should thus flow through hot, warm, and cold storages along their lifecycle. Hot data lives in fast and expensive storage (e.g., SSDs), cold data is moved to cheaper storage (e.g., Amazon Glacier), and all the other states stay in regular media (e.g., AWS S3). Even the data that users are not accessing for a long time must be analyzed and possibly migrated to cold storage.

4 COMPUTE EFFICIENCY STRATEGIES

Delivering efficient solutions for complex problems generally incurs costs. An energy-efficient system should provide a similar analysis to minimize costs and investments and maximize profit. Particular solutions must be designed for each workload, and therefore, solutions to increase efficiency also vary with the application. The guidelines presented in this section suggest optimization options that reduce power without significant latency at different levels of the architecture. Autoscaling reduces power consumption by minimizing idle state duration. Power consumed during the processing phase is proportional to the number of resources allocated. Efficient resource scheduling reduces the time spent in the processing phase, optimizing energy during load peaks and minimizing total power consumption. Serverless execution models or containerized services running in a Kubernetes environment are available, and scaling up and down the number of instances at any time adapt execution resources to the load. Selecting the best option depends on stability and warm start duration. Serverless execution introduces latency during the first request of each function, which can affect the quality of service even with large instances and a significant number of processing resources, masking idle power consumption. Containerized services can run “always on” or can be scaled down to a single instance, but idle power is not suppressed. The overall advantage of serverless execution is reduced idle power—and even negative consumption during idle periods of stable workloads—whereas containerized execution increases portability and cold start time is minimized. Both options can coexist within the same application. Serverless execution costs also include a per-request price component, and in high-load scenarios this cost constitutes a significant part of the total operation cost. If the total demand of a service can be satisfied with a small number of instances, it is worth avoiding a serverless model and hosting the service in a container.

Equation 02: Containerized compute energy model

Incoming rate $\lambda(t)$ (req/s), per-instance capacity μ . Instances

$$n(t) = \max(1, \lceil \mu \lambda(t) \rceil), \quad u(t) = \min(1, n(t) \mu \lambda(t)) \quad \text{Per-instance power with concavity parameter } P_{inst}(u) = P_{id} \quad (2)$$

BatchSize	Energy Wh per event	Latency s	WithinSLO
1	0.0502	0.002	TRUE
6	0.008533	0.012	TRUE
11	0.004745	0.022	TRUE
16	0.003325	0.032	TRUE
21	0.002581	0.042	TRUE
26	0.002123	0.052	TRUE
31	0.001813	0.062	TRUE
36	0.001589	0.072	TRUE
41	0.00142	0.082	TRUE

Table 2. Batch vs Real-time: Energy & Latency

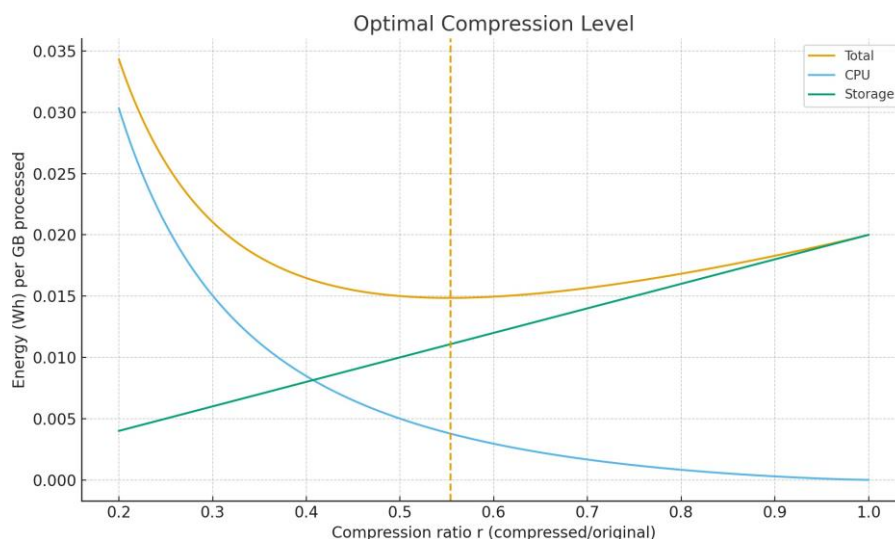


Fig. 4. Optimal Compression Level

4.1 Autoscaling and Resource Scheduling

In cloud systems, where demand is highly variable, resource-scheduling and autoscaling policies should minimize idle power and maintain sufficient compute power to fulfill user requests with the highest responsiveness possible. Resources can be configured according to workload patterns—for batch workloads, they should match peak usage and be scaled down with resources freed after execution; for real-time workloads, they should match capacity needs at all times. More generally, autoscaling policies can act to minimize spikes in resource consumption, while external scheduling can make use of tools, such as Kubernetes, to ensure the right number of nodes are available at a given time. Properly implemented, such patterns reduce energy consumption without incurring a latency penalty. Workload-specific metrics can also better indicate cloud charges to clients; such metrics can accurately capture the cost of idle resources in autoscaled groups and autoscaling groups can be placed in the same financial account for full monitoring capabilities. Fulfilling the Cost-efficient Cloud Software Principle not only helps clients financially, but also helps the environment, as the overall power used by the system is reduced. Reducing excess provisioning of resources, as maintained by the Infrastructure Sizing Principle, minimizes excess costs incurred by the cloud provider and helps support the Green Software Principle by reducing waste. Such techniques also link back to Cloud-native Pattern 1 (autoscaling), where properly tuned configuration ensures a more energy-aware deployment, thereby supporting the Energy-aware Service Classification Principle. Resources that are serverless in nature, such as Function as a Service, can be considered within the scope of autoscaling due to their zero-idle cost.

4.2 Serverless vs. Containerized Compute

Serverless compute resources can often remain warm even when idle, minimizing startup latency. However, the overhead induced by greater initialization time, added cold-start latency, and potential overcharging (e.g., charges incurred when simple functions are triggered extremely often) might outweigh energy savings compared to container-based compute resources. Containerized resources thus remain the preferred option when the ability to reach a completely idle state (and thus, a very low idle power consumption) is fundamental to the application. On the other hand, some workloads demand resources that will continuously process events in a sub-second timeframe and support a low latency target on the order of a few tens of milliseconds. For those workloads, utilizing serverless functions to handle requests is generally recommended. Analyzing these dimensions is important to avoid unstable resources that frequently start up and shut down, consuming substantial energy in transitions.

5 STORAGE AND I/O OPTIMIZATIONS

All design areas come together in I/O and, more importantly, in storage efficiency. Efficient physical databases consider multiple aspects, such as mechanics, battery consumption, speed of data access and freshness, consistency, and relevance. Variables related to storage and I/O are determined by all layers above: Data lifecycle states aggregate data into hot, warm, or cold sets, allowing the definition of associate tiers. Energy-aware caching addresses temporality and frequency of usage in a broader sense. Encoding, deduplication, and compression can also be optimized. Encoding scheme choices impact both data volume and processing energy, and therefore should be analyzed in this context. Furthermore, compression adds computational processing load, which may also impact energy consumption. Nevertheless, the trade-offs related to coding processing time and extra load versus disc storage and its associated energy need to be measured. Attention should be given to these aspects in preservation policies at both the business and operational levels: so-called Big Data scenarios frequently do not consider what data truly needs to be preserved and historical storage sometimes swells to an untenable size.

5.1 Efficient Data Encoding and Compression

Data encoding formats can improve disk space efficiency for core insurance processing, but they often incur an additional CPU load. Even higher compression may lead to improved storage requirements, and consequently energy savings, but it is typically accompanied by a stronger CPU load. Figure 5.1 depicts the trade-off between expected energy gains or consumption and imposed latency for the main available insurance data formats. The choice is thus application specific: XML format tends to be much heavier than others in CPU, but is best suited for very little data duplication. JSON compresses easily in disk space, while encoding

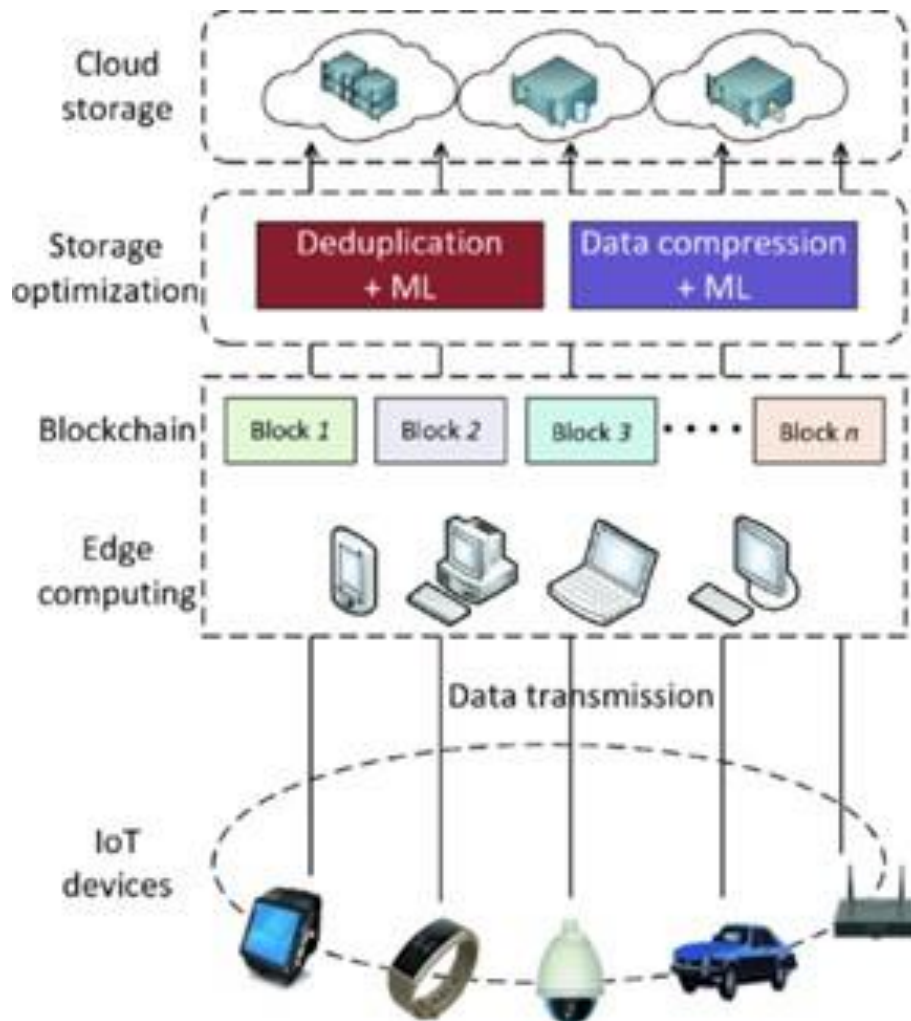


Fig. 5. An Optimization Scheme for IoT Data Storage Based on Machine Learning

sequential records with Avro, Parquet, or Protobuf sees low memory consumption. These patterns yield a low to medium energy consumption, associated with the use of sidecar queries to lookup status data. Other formats such as MsgPack, BSON, or Feather provide energy gains when decoding is cheaper than CPU tax, suggesting a greater focus on index space-saving. Bincode and CBOR support power with low CPU costs, indicating their use for CPU-constrained processes. A cost-effective solution relies on the use of either Avro or Parquet encoding to compress data in disk, internal table in a CBOR format to save external call, and MsgPack to share data among services or standby nodes. Heat map representations shall adopt the simpler pickling mechanism. The detailed redundancy in data encoding remains untransparent, specially in big codes. Hence, deduplication techniques can be applied to either detect redundancy in ID fields for further compression or the death of a provided asset in anonymization layers. Such procedures can incur a CPU overhead but accelerate the overall power while limiting data storage. Finally, the integration of a Compression/Decompression as a Service sidecar can further optimize power consumption for high-throughput tasks, taking profit of the data locality principle.

5.2 Optimized Storage Tiering

The data lifecycle of high-volume insurance applications introduces further opportunities for compute and storage efficiency. Data can be classified into three different states—hot, warm, and cold—reflecting their access frequency. Documents that originate in the system are written into hot storage, which usually comprises faster but more expensive storage systems, such as NoSQL databases. These hot data flows generate the basis for the continuous insurance processes of the system. The temporality of these workloads can generate data that is only needed for a shorter period of

time, such as fraud detection processes, and therefore care should be taken to size hot storage in a cost-efficient way. As data accumulates in hot storage, the necessity of keeping the complete historical data in hot storage becomes questionable. Past policyholders that have passed a certain date (for example four months after the end of coverage), in addition to the historical data of home-property statements are candidates to migrate to warm storage, usually a key-value store that has a higher latency and lower access frequency. For these type of inquiries, a slight increase in latencies is tolerable and even long-running operations can withstand higher latencies. Once certain processes become obsolete, or once these are requested less than anticipated, it is possible to start migrating data into cold storage, which is composed of cheaper ECG storage, and where latency is not an issue. A challenge in this logic is to define the conditions that should trigger the migration of data across lifecycle states. It is also important to define conditions to erase data that is not longer needed. These conditions should be devised in a way that they minimize costs, but they should also consider the energy aspects of data movements and ensure that the most adequate storage for that data is employed.

6 CONCLUSION

The results indicate that pursuing an architectural style designed for energy efficiency generally provides a significant gain in compute and operating cost through a net power saving, independent of the cost equipment level. Several patterns commonly associated with cloud-native software development lead to this type of gain. Unchecked, however, the energy impact also exposes the solution's user to regulatory reporting costs and risks arising from external power sources whose energy mix is outside the user's governance domain. It is advisable to track relevant metrics closely and, if necessary, define dedicated cloud area(s) responsible for reducing the ratio between energy cost and service price for the product segment. A high-volume data-processing solution designed for energy efficiency incorporates several known patterns. These patterns need not all be present in the solution, either concurrently or for its whole lifecycle. Moreover, other patterns not examined here may be necessary for reasons other than energy savings, and their use will not preclude or diminish the power savings on compute operations. Their application only implies that significant gains can be obtained in a cloud-native enterprise, such as an insurance or financial company, without sacrificing design principles of engineering control, governance security, data observability, reliability, and performance if regulatory overheads are also properly managed.

Equation 03: Break-even (serverless vs. containerized)

We compare cost/sec

$$\text{serverless} \lambda \text{creq} + \lambda \text{dpGB} \text{vs.} \text{container} 3600 n \text{cinst/hr} + \lambda \text{dpGB} \text{Network} \text{cancels,} \quad \text{so the break-even RPS} \quad (3)$$

λ solves

$$\lambda \text{creq} = 3600 n (\lambda) \text{cinst/hr}$$

6.1 Future Trends

The interplay between the software and hardware layers is crucial in shaping energy efficiency in cloud-native architectural environments. Emerging edge-specific hardware and software stack will further broaden the breadth of energy-efficient implementation in cloud-native environments and open more avenues to reduce carbon footprint for the coming years. Emerging technologies offer new paths to energy efficiency within financial services and banking. OpenAI will provide key power-aware building blocks and APIs. Emerging edge hardware will contribute to architectural efficiency by pushing service locations closer to end-user locations and by exploring alternative delivery methods with drones. Quantum computing, if realized, will deliver an unparalleled efficiency advantage. In the medium to long term, regulation can provide the conceiving nudge toward a real green economy. Power consumption regulators, such as energy use and carbon reporting through the EU Taxonomy, the SEC regulations on crypto currencies, and others will be pivotal.

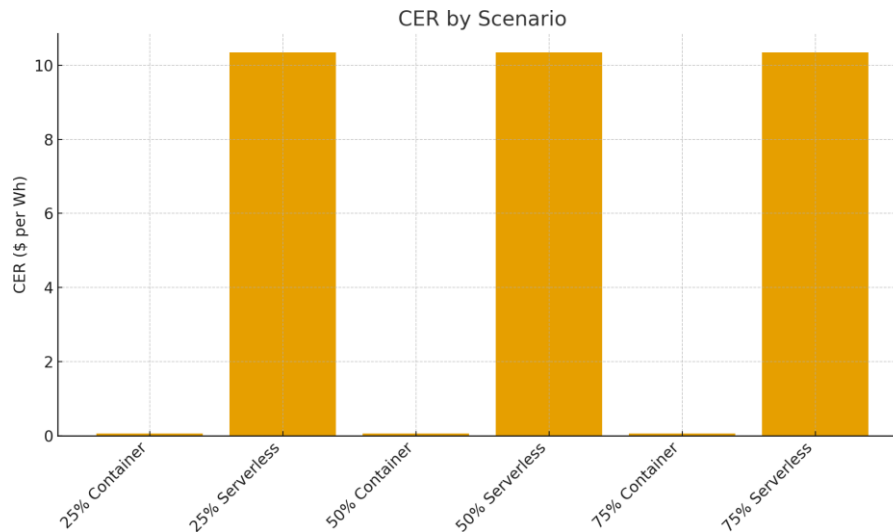


Fig. 6. CER by Scenario

REFERENCES

- [1]. Brooks, S. K., Webster, R. K., Smith, L. E., Woodland, L., Wessely, S., Greenberg, N., & Rubin, G. J. (2020). The psychological impact of quarantine and how to reduce it: Rapid review of the evidence. *The Lancet*, 395(10227), 912–920.
- [2]. Dong, E., Du, H., & Gardner, L. (2020). An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases*, 20(5), 533–534.
- [3]. Gadi, A. L. (2020). Evaluating Cloud Adoption Models in Automotive Manufacturing and Global Distribution Networks. *Global Research Development (GRD)* ISSN: 2455-5703, 5(12), 171-190.
- [4]. Wu, Z., & McGoogan, J. M. (2020). Characteristics of and important lessons from the coronavirus disease 2019 (COVID-19) outbreak in China. *JAMA*, 323(13), 1239–1242.
- [5]. Zhou, F., Yu, T., Du, R., Fan, G., Liu, Y., Liu, Z., . . . Cao, B. (2020). Clinical course and risk factors for mortality of adult inpatients with COVID-19 in Wuhan, China. *The Lancet*, 395(10229), 1054–1062.
- [6]. Richardson, S., Hirsch, J. S., Narasimhan, M., Crawford, J. M., McGinn, T., David-son, K. W., . . . the Northwell COVID-19 Research Consortium. (2020). Present- ing characteristics, comorbidities, and outcomes among 5,700 patients hospitalized with COVID-19 in the New York City area. *The New England Journal of Medicine*, 382(24), 2302–2311.
- [7]. Pandiri, L. (2020). Predictive Modeling of Claims in Flood and Mobile Home In- surance using Machine Learning. *Global Research Development (GRD)* ISSN: 2455- 5703, 5(12), 1-18.
- [8]. Li, Q., Guan, X., Wu, P., Wang, X., Zhou, L., Tong, Y., . . . Feng, Z. (2020). Early transmission dynamics in Wuhan, China, of novel coronavirus–infected pneumonia. *The New England Journal of Medicine*, 382, 1199–1207.
- [9]. Cucinotta, D., & Vanelli, M. (2020). WHO declares COVID-19 a pandemic. *Acta Biomedica*, 91(1), 157–160.
- [10]. Rubin, G. J., & Wessely, S. (2020). The psychological effects of quarantines and how to reduce them. *The Lancet*, 395(10227), 912–920.
- [11]. Botlagunta, P. N., & Sheelam, G. K. (2020). Data-Driven Design and Validation Techniques in Advanced Chip Engineering. *Global Research Development (GRD)* ISSN: 2455-5703, 5(12), 243-260.
- [12]. Polack, F. P., Thomas, S. J., Kitchin, N., Absalon, J., Gurtman, A., Lockhart, S., . . . Gruber, W. C. (2020). Safety and efficacy of the BNT162b2 mRNA Covid-19 vaccine. *The New England Journal of Medicine*, 383(27), 2603–2615.
- [13]. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., . . . Houlsby, N. (2020). An image is worth 16×16 words: Transformers for image recognition at scale. *arXiv Preprint arXiv:2010.11929*.
- [14]. He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE/CVF Confer- ence on Computer Vision and Pattern Recognition*, 9729–9738.
- [15]. Chakilam, C., Koppolu, H. K. R., Chava, K. C., & Suura, S. R. (2020). Integrating

- Big Data and AI in Cloud-Based Healthcare Systems for Enhanced Patient Care and Disease Management. Global Research Development (GRD) ISSN: 2455-5703, 5(12), 19-42.
- [16]. Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 1597–1607.
- [17]. Kolesnikov, A., Zhai, X., & Beyer, L. (2020). Big Transfer (BiT): General visual representation learning. *European Conference on Computer Vision (ECCV) 2020*, 491–507.
- [18]. Le Quéré, C., Jackson, R. B., Jones, M. W., Smith, A. J. P., Abernethy, S., Andrew, R. M., . . . Peters, G. P. (2020). Temporary reduction in daily global CO₂ emissions during the COVID-19 forced confinement. *Nature Climate Change*, 10(7), 647–653.
- [19]. Dwaraka Nath Kummari, Srinivasa Rao Challa, "Big Data and Machine Learning in Fraud Detection for Public Sector Financial Systems," *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, DOI: 10.17148/IJARCCE.2020.91221
- [20]. Hsiang, S., Allen, D., Annan-Phan, S., Bell, K., Bolliger, I., Chong, T., . . . Walsh, S. (2020). The effect of large-scale anti-contagion policies on the COVID-19 pandemic. *Nature*, 584(7820), 262–267.
- [21]. Chetty, R., Friedman, J. N., Hendren, N., Stepner, M., & the Opportunity Insights Team. (2020). How did COVID-19 and stabilization policies affect spending and employment? NBER Working Paper No. 27431.
- [22]. World Health Organization. (2020). Mask use in the context of COVID-19: Interim guidance (1 December 2020). WHO.
- [23]. Meda, R. (2020). Designing Self-Learning Agentic Systems for Dynamic Retail Supply Networks. *Online Journal of Materials Science*, 1(1), 1–20. Retrieved from <https://www.scipublications.com/journal/index.php/materials/article/view/1336>.
- [24]. United Nations. (2020). The Sustainable Development Goals Report 2020. United Nations.
- [25]. International Monetary Fund. (2020). World Economic Outlook, October 2020: A long and difficult ascent. IMF.
- [26]. OECD. (2020). Education at a Glance 2020: OECD indicators. OECD Publishing.
- [27]. Somu, B. (2020). Transforming Customer Experience in Digital Banking Through Machine Learning Applications. *International Journal Of Engineering And Computer Science*, 9(12).
- [28]. Piketty, T. (2020). Capital and ideology (A. Goldhammer, Trans.). Harvard University Press.
- [29]. Andersen, K. G., Rambaut, A., Lipkin, W. I., Holmes, E. C., & Garry, R. F. (2020). The proximal origin of SARS-CoV-2. *Nature Medicine*, 26(4), 450–452.
- [30]. Inala, R. (2020). Building Foundational Data Products for Financial Services: A MDM-Based Approach to Customer, and Product Data Integration. *Universal Journal of Finance and Economics*, 1(1), 1–18.
- [31]. Pan, A., Liu, L., Wang, C., Guo, H., Hao, X., Wang, Q., . . . Wu, T. (2020). Association of public health interventions with the epidemiology of the COVID-19 outbreak in Wuhan, China. *JAMA*, 323(19), 1915–1923.
- [32]. Howard, J., Huang, A., Li, Z., Tufekci, Z., Zdimal, V., van der Westhuizen, H. M., . . . Rimoin, A. W. (2020). Face masks against COVID-19: An evidence review. *PNAS*, 118(4), e2014564118.
- [33]. O’Neil, C. (2020). Weapons of math destruction: How big data increases inequality and threatens democracy.