Computing&AI Connect
ISSN: 3104-4719
2026, Article ID. 2026.0043, Cite as: https://doi.org/10.69709/CAIC.2026.142241

SCIFINITI PUBLISHING

Research Article

OPEN ACCESS

Device-First Continuum AI: Low-Latency, Resilient, and Cost-Efficient Autonomous Operations for the Energy Sector

Siavash M. Alamouti[1,*], **Fay Arjomandi**[1], **Michel Burger**[1], **Jeremy Hsu**[2], **Bashar Altakrouri**[3]

[1] mimik Technology Inc., Oakland, CA 94619, USA
[2] mimik Technology Inc., Vancouver, BC V6A 1B2, Canada
[3] Saudi Aramco, Dhahran 31311, Saudi Arabia

Article History
Received: April 11, 2026 Accepted: September 30, 2026

Abstract
Industrial automation in the energy sector requires AI that remains operational regardless of network conditions, a requirement that cloud-centric architectures cannot meet in remote oil fields, offshore platforms, and other disconnected environments. This paper evaluates Device-First Continuum AI (DFC-AI), an architecture in which intelligent agents originate on the end device and escalate across the continuum to a gateway or the cloud only when a task exceeds local capability. This study provides a benchmark-grounded comparison of cloud-centric, gateway-based, and device-first architectures on latency, resilience, and cost, along with a model that explains the results and the conditions under which the device-first advantage holds. Across simulations of drone inspection, sensor-network, and worker-safety deployments, DFC-AI delivers large reductions in both latency and energy, maintains near-full capability when the cloud is unreachable and when an individual node fails, and runs at a small fraction of the cost of either alternative. Cloud-centric systems fail completely on disconnection, while gateway-based systems, although they survive a cloud outage, concentrate risk at a single aggregation node whose loss disables every device it serves and can cost more than the cloud they were meant to replace. The advantage is largest for the predominantly local workloads typical of the energy sector and narrows only when work is inherently cloud-bound. The practical implication is that processing intelligence where data originates, rather than transmitting it to a distant tier, is what makes industrial AI available, affordable, and resilient at scale.

ACCEPTED

Keywords:
device-first continuum AI; Hybrid Edge Cloud; agentic AI; physical AI; industrial AI; on-device inference; network resilience; end-to-end latency; energy efficiency; total cost of ownership; microservices; industrial automation; energy sector

1. Introduction: The Imperative for Always-Available Intelligence

The promise of AI in industrial automation rests on a fundamental requirement: intelligent agents must be available anytime, anywhere, regardless of network conditions. In the energy sector, where operations span remote oil fields, offshore platforms, and hazardous environments, this requirement is not negotiable. It is essential for both operational continuity and human safety. Yet a cloud-centric AI architecture ceases to function when network connectivity is lost or degraded, rendering the intelligent system inoperative precisely when it may be needed most [1,2].

Consider the operational landscape within the energy sector. Autonomous robots inspecting pipelines, drones monitoring offshore platforms, smart sensors deployed across oilfields, and control systems within refineries all generate large amounts of contextual data directly

* **Corresponding Author:**
Siavash M. Alamouti, mimik Technology Inc., Oakland, CA 94619, USA, siavash.alamouti@mimik.com

© 2026 Copyright by the Authors.
Licensed as an open access article using a CC BY 4.0 license.

at the point of operation. For these devices to make accurate and timely decisions which may be crucial for safety, efficiency, and preventing costly downtime, they require immediate access to this local context. Transferring compressed video streams and sensor data to a centralized cloud for processing before a decision can be made introduces unacceptable latency, rendering real-time control and intervention impossible in many critical situations. The inspection deployment modeled here generates approximately 10.8 GB per drone per day at the capture rate and duty cycle stated in Section 5. Furthermore, industrial AI workloads exhibit the heavy-tailed structure documented in datacenter and cluster-trace studies [3–5]; we take the corresponding locality split in our scenarios to be approximately 80% locally resolvable, a design assumption whose effect on the results is examined in Table 1. Workloads of this kind can be categorized as physical AI: the intelligence is embodied in equipment that senses and acts on the physical world under real-time constraints, so the volume of data to be moved is set by the deployment itself rather than by user demand, and a late decision is of no operational use.

The challenge extends beyond current operations. As the industry transitions from millions of human-operated applications to agentic AI, in which billions of autonomous agents perceive, decide, and act continuously without a human in the loop for each step, the architectural limitations of centralized approaches become insurmountable. Simple mathematics illustrates this scalability crisis: one billion autonomous agents each requiring modest 1 MB/s communication with cloud services would demand 1000 TB/s of aggregate network bandwidth, exceeding the capacity of entire continental internet infrastructure. This fundamental barrier demands a complete architectural rethink.

To address these demands, Hybrid Edge Cloud architectures [6,7] have been proposed. Within this paradigm, DFC-AI [8] emerges as a specialized approach. DFC-AI is characterized by the implementation of intelligent agents, architected using a microservices framework, that are initiated and primarily reside on devices. These intelligent agents can then extend their operations and collaborate with gateways and cloud servers as needed, forming a cohesive and distributed intelligence continuum. Similar to how the human brain handles immediate reflexes and local tasks at the peripheral level while relying on the central nervous system for complex reasoning and global awareness, DFC-AI prioritizes intelligence at the device level for timely actions while leveraging the broader cloud continuum for more involved processes.

This paper presents comprehensive evidence for DFC-AI's advantages through mathematical modeling, detailed simulations of real-world energy sector scenarios with statistical validation, and verification of theoretical predictions.

The device-first continuum architecture itself was introduced in prior work [6–8]. The contribution of this paper is its rigorous evaluation: a model that explains where and why a device-first design wins, a benchmark-grounded simulation that compares it against the two dominant alternatives, and an explicit statement of the conditions under which its advantage holds and those under which it does not. To our knowledge, this is the first study to place the device-first option alongside cloud-centric and gateway-based baselines and to evaluate all three on latency, resilience, and cost together, rather than on latency or throughput alone, for autonomous operations in the energy sector.

The specific contributions are the following.

- We make the device-first continuum precise as the object of evaluation: AI agents implemented as microservices that originate and resolve on the end device, above the operating system, and escalate to gateway and cloud tiers only when a task exceeds local capability, so that the first service call a device makes is to itself.
- We give a latency model that ties the architecture to a single mechanism, avoidance of the network path. End-to-end latency is decomposed into round trip, transmission, and inference, which shows why on-device resolution dominates for the data-heavy, intermittently connected workloads characteristic of the energy sector.
- We compare DFC-AI quantitatively against centralized cloud and gateway-edge baselines across three representative energy-sector scenarios, drone inspection, sensor networks, and worker safety, using parameters drawn from published benchmarks and reporting latency distributions rather than single point estimates.
- We characterize resilience along two axes, the fraction of tasks served locally during a cloud outage and the fraction of the fleet that survives a single-node failure, deriving both per scenario from the deployment topology rather than asserting fixed figures.
- We show that the latency, resilience, and cost advantages all follow from one architectural choice, keeping work off the network, and we give a structural cost argument in which the on-device marginal cost is near zero while cloud cost is recurring and proportional to traffic.

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

Table 1: Sensitivity of DFC-AI latency to operating conditions (drone scenario); the cloud-centric reference stays at 490 ms median, 491 ms mean, and 505 ms p99.

Condition	Setting	Median (ms)	Mean (ms)	p99 (ms)	Local (%)
Local mesh bandwidth	1000 Mbps	34.6	61.8	498.1	—
	250 Mbps	34.6	78.3	498.1	—
	100 Mbps	34.6	111.2	498.1	—
	50 Mbps	34.6	119.4	500.9	—
	25 Mbps	34.6	119.4	500.9	—
Cloud-bound fraction	5%	34.6	61.8	498.1	94.9
	10%	35.4	85.0	499.5	89.9
	20%	36.9	131.7	502.3	79.6
	30%	40.3	181.9	507.2	68.7
Device capability	100%	34.6	61.8	498.1	—
	75%	35.6	62.8	498.1	—
	50%	37.6	64.9	498.1	—
	25%	43.6	71.0	498.1	—

- We bound the claims, identifying the workloads for which the device-first advantage holds and those for which it does not.

1.1. Relation to the Authors' Prior Work

The device-first continuum concept and the Hybrid Edge Cloud substrate on which it rests were introduced in our earlier publications [6–8]. In particular, the book chapter [8], an extended version of a paper presented at the Future Technologies Conference (FTC) 2025, defined DFC-AI and argued for it conceptually, describing the architecture, the microservice agent model, and the escalation principle. That work contained no quantitative evaluation: no latency, queueing, energy, or cost model, no simulation, no baselines, and no numerical results.

The present paper is an evaluation study that extends [8] rather than restating it. The new material comprises: (i) an analytical service-time model with device-dependent throughput scaling and a load-dependent (Erlang-C) queueing treatment, tied to a single mechanism, avoidance of the network path; (ii) a parameterization grounded in published on-device inference benchmarks rather than assumed constants; (iii) a discrete-event simulation of three energy-sector deployments with 30 independent runs per scenario, confidence intervals, and significance testing; (iv) the explicit introduction of a gateway-based baseline alongside the cloud baseline, and a two-axis resilience characterization that separates cloud-outage survival from single-node failure survival; (v) an arrival-rate sweep and a three-parameter sensitivity analysis; (vi) a structural cost model covering both payload-billed and token-billed workloads, with the gateway's fixed component taken from independent total-cost-of-ownership studies; and (vii) an explicit statement of the boundary conditions under which the device-first advantage narrows. No text, figure, table, or numerical result from [8] is reproduced here. The conceptual description in Sections 2 and 3 is a condensed restatement of [8], retained only to make the evaluated architecture precise, and is cited as such throughout.

2. Background: DFC-AI Architecture and Energy Sector Requirements

2.1. Device-First Continuum AI Overview

Device-First Continuum AI (DFC-AI) [8] differs from cloud-centric approaches in where execution begins: the first service call a device makes is to itself. As described in [8], DFC-AI implements intelligent agents using a microservices architecture that originates at end devices and extends across the computational continuum. The architecture is built on the Hybrid Edge Cloud paradigm [6,7] and enables devices to process data locally while maintaining the ability to collaborate with edge gateways and cloud servers when beneficial.

Key characteristics of DFC-AI relevant to energy sector applications include:

- **Intelligence at the Source:** AI capabilities embedded directly on devices enable real-time processing without network dependency, critical for remote energy operations.

ACCEPTED

- **Microservices Architecture:** Lightweight, modular agents that activate only required components, optimizing energy consumption in battery-powered devices.
- **Zero-Configuration Resource Discovery:** Automatic discovery and utilization of computational resources within the local environment, enabling dynamic clustering of inspection drones or sensor networks.
- **Network Resilience:** Full operational capability maintained during network outages, essential for safety-critical systems in isolated locations.
- **Heterogeneous Device Support:** Efficient utilization of mixed computational capabilities, where devices equipped with a graphics processing unit (GPU) handle intensive workloads for entire fleets.

2.2. Energy Sector Operational Requirements

The energy sector presents specific challenges that make traditional cloud-centric AI architectures inadequate:

1. **Remote and Hazardous Locations:** Oil fields, offshore platforms, and pipeline networks often operate in areas with limited or unreliable network connectivity. Systems must function autonomously for extended periods.
2. **Safety-Critical Operations:** Worker safety monitoring, gas leak detection, and emergency response systems cannot tolerate network-induced delays or failures. Immediate local response is essential.
3. **Substantial Data Generation:** Modern energy operations generate large data volumes. The inspection deployment modeled here produces approximately 10.8 GB per drone per day, derived from the capture rate, payload size, and duty cycle stated in Section 5.
4. **Real-Time Control Requirements:** Process control in refineries, valve adjustments in pipelines, and collision avoidance for autonomous vehicles require sub-second response times.
5. **Economic Constraints:** Energy companies require cost-effective solutions that leverage existing infrastructure investments rather than requiring continuous cloud service payments.

These requirements dictate an architectural approach that ensures autonomous operation, minimizes latency, reduces data transmission costs, and maintains functionality regardless of network availability.

3. Architectural Comparison: DFC-AI vs. Gateway-Based Edge AI vs. Centralized Cloud AI

The selection of an appropriate architectural paradigm is fundamental to the successful implementation of intelligent systems in industrial automation. This section outlines the main differences between three primary architectural approaches: Centralized Cloud AI, Gateway-Based Edge AI, and DFC-AI.

3.1. Centralized Cloud AI

In this architecture shown in Figure 1, all data from end devices is transmitted to a central cloud infrastructure for processing and analysis. The end device has no ability to host workloads and only includes a client agent with or without user interface depending on the type of device.

While the cloud offers computational resources and scalability, this approach suffers from several critical limitations. Most significantly, it exhibits complete operational failure during network outages; when internet connectivity is lost, all intelligent functionality ceases, rendering devices non-functional. This creates a single point of failure that is particularly problematic in remote industrial environments where network reliability cannot be guaranteed. Additionally, this approach suffers from high communication latency, making it unsuitable for applications requiring real-time responses. It requires continuous and reliable network connectivity and can lead to substantial bandwidth consumption and cloud hosting costs, especially with large deployments and high data volumes. The context and operational data that must be continuously transferred to the cloud can range from gigabytes to terabytes per day, amplifying bandwidth requirements and associated costs. Data privacy and security can also be a concern as sensitive operational data is transmitted over the network and resides in a centralized environment. End devices in this model typically have very low autonomy, acting primarily as data conduits.

3.2. Gateway-Based Edge AI

This approach shown in Figure 2 moves some computational power and intelligence closer to the end devices, typically residing on dedicated gateway devices. Gateways aggregate and pre-process data from multiple end devices before potentially forwarding it to the cloud for further analysis. This reduces latency compared to a purely cloud-based approach and can alleviate some bandwidth constraints.

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

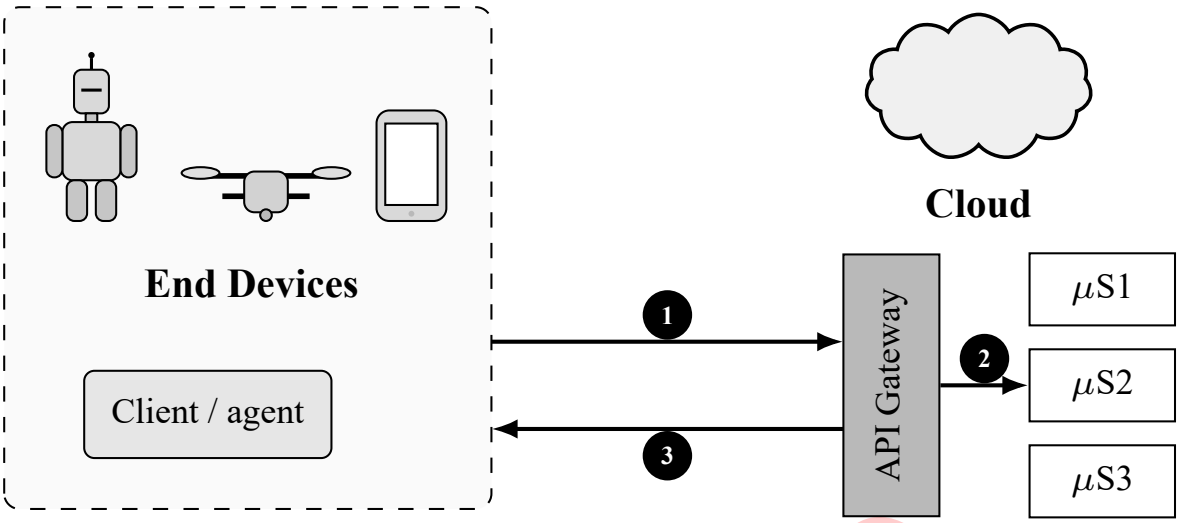


Figure 1: Centralized cloud AI. (1) the device sends raw context to the cloud; (2) the application programming interface (API) gateway dispatches to the microservices; (3) the result returns.

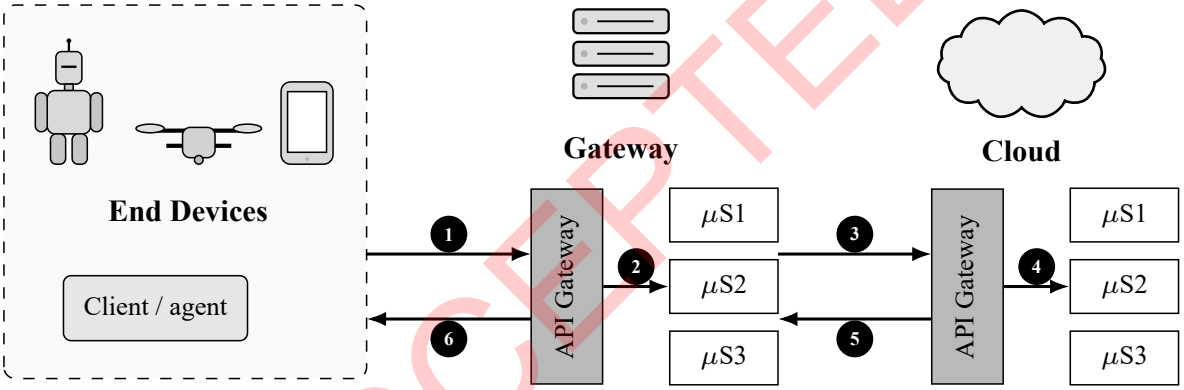


Figure 2: Gateway-based edge AI. (1) device offloads to the gateway; (2) gateway dispatches internally; (3) gateway escalates to the cloud when needed; (4) cloud dispatches internally; (5)–(6) results return via the gateway.

ACCEPTED

However, significant limitations remain. The intelligence is still centralized at the gateway level, the end-device context must be transmitted over the network to the gateway, and the end devices themselves remain comparatively simple. The device-to-gateway and gateway-to-cloud links both persist as dependencies, creating multiple potential points of failure. Critically, gateway tiers are typically operated as cloud-managed extensions rather than autonomous systems: in the dominant edge orchestration frameworks the control plane is kept in the cloud, and only a minimal subset of control-plane state is cached at the edge for use during connectivity loss [9]. A disconnected gateway can therefore continue serving already-deployed, pre-cached models, but it loses model and lifecycle updates, reconfiguration, and cross-node coordination until connectivity is restored, and it lacks dynamic resource discovery, requiring manual configuration of gateway resources. A further, often-overlooked point is that gateway-based architectures can paradoxically cost more than cloud solutions for some workloads, because the fixed edge-server infrastructure, including hardware amortization, power, cooling, and operational staff, must be maintained regardless of utilization, whereas cloud cost scales with actual consumption [10,11].

3.3. Device-First Continuum AI (DFC-AI)

DFC-AI, shown in Figure 3, places the agent runtime on the end devices themselves. These intelligent devices, leveraging microservices architecture, can process most of the data locally, make autonomous decisions based on their immediate context, and take real-time actions. This reduces communication latency and minimizes the reliance on continuous network connectivity, enabling operation in remote or challenging environments.

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

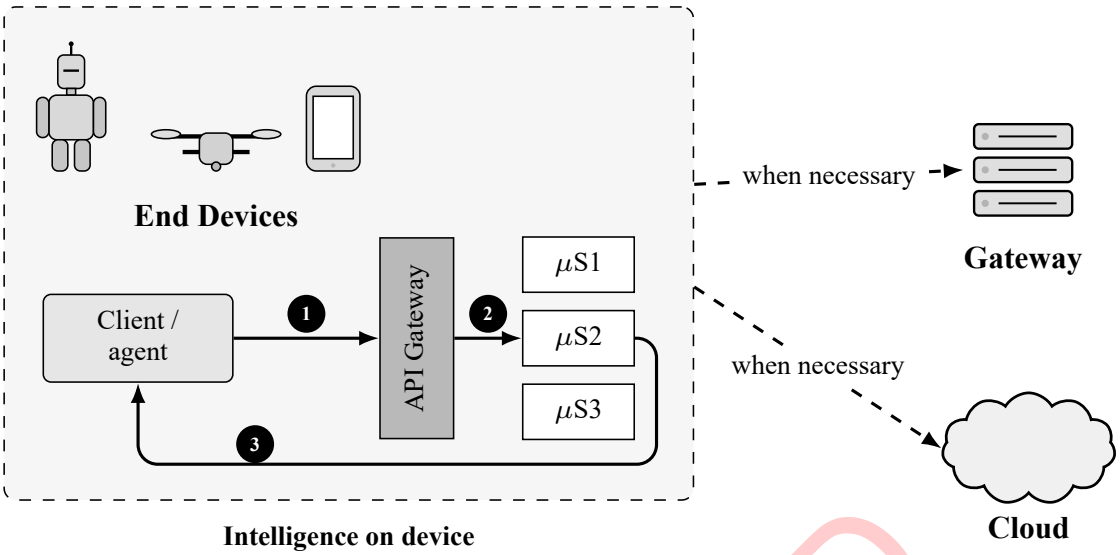


Figure 3: DFC-AI. The API gateway and microservices run on the end device: (1) the client invokes the local API gateway, (2) the gateway dispatches to a local microservice, (3) the result returns. Dashed arrows show optional escalation to a gateway or the cloud.

A key distinguishing feature of DFC-AI is its zero-configuration dynamic resource discovery capability. When a powerful computational node (such as an NVIDIA Jetson AGX [12] or similar GPU-enabled device) is introduced into the local network, all DFC-AI devices automatically discover and can immediately leverage this additional processing power without any manual configuration. For example, in a swarm of inspection drones, adding a single high-performance GPU node enhances the AI processing capabilities of the entire fleet, enabling more sophisticated real-time analytics like advanced computer vision or complex decision-making algorithms that would otherwise require cloud resources.

Furthermore, DFC-AI demonstrates superior network resilience compared to other architectures. Network resilience varies significantly based on workload characteristics. Cloud-centric systems fail completely during network outages. Gateway-based systems keep serving cached models during a cloud outage, but they concentrate risk at a single aggregation node, so a gateway or local-link failure removes intelligence from every device it serves. DFC-AI distributes intelligence across the devices, so it both rides out cloud outages and tolerates single-node failures: in our energy sector scenarios almost all tasks involve local sensor processing, immediate safety responses, or autonomous navigation, so DFC-AI retains 95% to 100% operational capability during a cloud outage and 90% to 100% under a single-node failure. This is because the core intelligence resides on the devices themselves, with cloud and edge resources serving as enhancement rather than dependency. Bandwidth consumption is lower as only essential insights or aggregated data are transmitted. Data privacy and security are enhanced as sensitive data can be processed and stored locally. DFC-AI offers real-time capabilities and high resilience due to the distributed nature of intelligence.

While DFC-AI may require more careful architectural planning, such as designing services to initiate local calls instead of relying on centralized gateways or servers, the overall development and deployment complexity tends to decrease. This is because developers can modularize workloads as microservices, test them independently on target devices, and avoid building and managing complex data pipelines between the edge and the cloud. In addition, reducing centralized dependencies streamlines system integration, simplifies maintenance, enhances device autonomy, and enables seamless direct collaboration between devices.

4. Mathematical Framework

To quantify the performance advantages of DFC-AI, we develop mathematical models that capture the key performance metrics: latency, energy consumption, and cost. These models enable us to compare DFC-AI against traditional cloud-centric and gateway-mediated architectures across realistic deployment scenarios.

4.1. Latency Analysis Model

End-to-end latency is critical for real-time industrial applications. We model the total latency for each architecture

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

by breaking down the components involved in processing a typical AI workload request.

4.2. Service Time and Load-Dependent Queueing

The time a task takes to complete has two parts: the time the chosen processor spends running it, called the service time, and the time it spends waiting behind other tasks before it can start, called the queueing delay. Most of the difference between the three architectures comes from these two quantities, so we define each before assembling the end-to-end latency.

4.2.1. Service Time

The service time of a task is the amount of computation it requires divided by the speed of the processor that runs it, plus the time to move its data, with two adjustments. For a task of complexity class c running on node d, the service time is

$$t_{\text{inf}}(c, d) = \left(\frac{W_c}{S_d} + R_{\text{io}}\, D \right) \eta_d\, \xi, \tag{1}$$

where:

- W_c is the amount of computation the task requires, fixed by its complexity class c. We set it to the single-stream inference latency of a representative model for each class on the reference device, an Arm Cortex-A class central processing unit (CPU), calibrated against published on-device inference benchmarks: about 2 ms for the simple class (a small model), from MLPerf Tiny [13]; about 30 ms for the moderate class (a compact image-classification network), from MLPerf Inference [14]; about 120 ms for the complex class (a standard image-classification network or an object detector), from edge object-detection benchmarks [15]; and about 400 ms for the cloud-only class (a large model), extrapolated from the same sources rather than quoted from a single reported result. These classes differ by close to two orders of magnitude in cost. Dividing W_c by S_d then reproduces the measured latencies of a GPU or neural processing unit (NPU) at the operating points that actually execute [16]: an object detector offloaded to a GPU runs in about 10 to 20 ms of compute time, matching the same benchmarks, while the same detector on the Arm CPU takes more than 100 ms.
- S_d is the processing throughput of node d, measured relative to that baseline device, for which $S_d = 1$. A small sensor or wearable is a fraction of one, while a GPU-equipped drone or edge server is fifteen to twenty times faster. Dividing the work W_c by S_d means the same task finishes proportionally sooner on a faster processor. This is the central refinement over a single fixed inference time: a model is quick on a GPU/NPU and slow on a weak CPU, and the routing decisions inherit that difference. A single throughput scalar per device is a first-order model, since the true speedup is larger for big models than for small ones; we set each scalar to match the heavier classes that dominate GPU/NPU use, which understates the speedup for the smallest models.
- $R_{\text{io}}\, D$ is the time to move the task's data into the model, for example decoding a video frame or reading a sensor buffer. It grows with the payload size D, at roughly 4 ms per megabyte. For a 5 MB drone image this term dominates the service time; for a 256-byte sensor reading it is negligible.
- η_d is a runtime overhead factor. It equals one when the model runs as native code, which is the case in the cloud, on a gateway, and on an on-device GPU/NPU. It rises to about 1.45, a roughly 45 percent slowdown, only on a device CPU, where the model runs as a portable microservice on a runtime above the operating system rather than as native code compiled for that processor. This figure agrees with published measurements of portable bytecode against native execution.
- ξ is a random multiplier that represents run-to-run timing variation, the small differences between repeated executions of the same task caused by caching, scheduling, and thermal effects. It is drawn from a lognormal distribution, so it is always positive and slightly skewed toward higher values: most runs are close to nominal and a few are slower. Its average value is exactly one, so it widens the spread of the results by about 18 percent without shifting the average. It changes the shape of the distribution, not the reported figures.

At the reference device, where $S_d = 1$ and ξ sits at its average, the compute term equals W_c, so each class takes its measured CPU inference latency; dividing by S_d then reproduces the measured GPU/NPU latencies on the faster nodes. The model therefore tracks published benchmark numbers at the operating points that execute and adds device-dependent scaling everywhere else.

4.2.2. Queueing Delay

A processor shared by many devices cannot run every request the instant it arrives. When several tasks arrive close together, the later ones wait, and the length of that wait depends on how heavily the processor is loaded rather than

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

on any fixed amount. We therefore model each processing tier as a queue with one or more servers. Writing λ for the average rate at which tasks arrive at a tier, $\mu = 1/\overline{t_{\text{inf}}}$ for the number of tasks one server can finish per second (the inverse of the average service time), and c for the number of servers working in parallel, the load on the tier is summarized by its utilization $\rho = \lambda/(c\mu)$, the fraction of capacity in use. The average wait is

$$T_{\text{queue}} = \frac{C(c, \lambda/\mu)}{c\mu - \lambda}, \quad C(c, a) = \frac{\frac{a^c}{c!}\frac{1}{1-\rho}}{\sum_{k=0}^{c-1}\frac{a^k}{k!} + \frac{a^c}{c!}\frac{1}{1-\rho}}, \quad (2)$$

where $a = \lambda/\mu$ and C is the Erlang-C probability that an arriving task finds every server busy; for a single server this reduces to $T_{\text{queue}} = \lambda/[\mu(\mu - \lambda)]$. The important feature is the denominator $c\mu - \lambda$. While the utilization ρ stays well below one the wait is small, and as ρ approaches one, meaning the tier is busy almost all the time, the wait climbs steeply.

This is where the architectures diverge. The cloud and the gateway are shared tiers that carry the combined traffic of the entire fleet, so their arrival rate λ is large and they can be driven toward saturation, especially when requests arrive in bursts. DFC-AI spreads the same total traffic across many device-local queues, so each device sees only its own small λ and stays in the region where the wait is negligible. The simulation does not assume any fixed wait; it drives each processor from the actual arrival times of the tasks, so that bursts produce real transient queueing and quiet periods produce none.

The end-to-end latency of each architecture, defined next, combines this service time and queueing delay with the network transmission time. The on-device path adds the runtime factor η and a small inter-process handoff of about 2.5 ms, and it pays no network transmission when the task is handled on the device that produced it.

For cloud-centric systems, the total latency includes network transmission to the cloud, queuing delays, cloud processing, and return transmission:

$$L_{\text{cloud}} = T_{\text{net}}^{\text{up}} + T_{\text{queue}} + T_{\text{proc}}^{\text{cloud}} + T_{\text{net}}^{\text{down}} \quad (3)$$

where:

- $T_{\text{net}}^{\text{up}}$ = uplink network transmission time from device to cloud
- T_{queue} = queuing delay at cloud servers due to concurrent requests
- $T_{\text{proc}}^{\text{cloud}}$ = processing time on cloud infrastructure
- $T_{\text{net}}^{\text{down}}$ = downlink network transmission time from cloud to device

For gateway-mediated systems, the latency includes local device-to-gateway communication, gateway processing, and escalation to the cloud when the gateway tier is predicted to be slower than the wide-area path:

$$L_{\text{gateway}} = T_{d\to g} + T_{\text{proc}}^{\text{gateway}} + T_{g\to d} + \alpha \cdot L_{\text{cloud}} \quad (4)$$

where:

- $T_{d\to g}$ = transmission time from device to gateway
- $T_{\text{proc}}^{\text{gateway}}$ = processing time at the gateway
- $T_{g\to d}$ = transmission time from gateway back to device
- α = the fraction of tasks that escalate to the cloud ($0 \le \alpha \le 1$); in the simulation this is an outcome of the same latency-prediction rule the device-first router uses, so the architectures differ in where work runs rather than in routing intelligence
- L_{cloud} = additional cloud processing latency when escalation occurs

For DFC-AI systems, most processing occurs locally within device clusters with occasional collaboration beyond the local network:

$$L_{\text{DFC}} = T_{\text{proc}}^{\text{local}} + \beta \cdot T_{\text{collab}} \quad (5)$$

where:

- $T_{\text{proc}}^{\text{local}}$ = local processing time within the device cluster (leveraging optimal local resources)
- β = collaboration factor representing the fraction of tasks requiring external collaboration ($0 \le \beta \le 1$)
- T_{collab} = additional time for collaboration beyond the local cluster when needed

The key advantage of DFC-AI is that β is typically small (<0.2) for the workloads modeled here, so the heterogeneous local cluster handles the vast majority of processing requirements locally. This locality split is a design assumption of the scenarios rather than a measured property of industrial workloads, and Table 1 reports the sensitivity of the results to it. Within the cluster, GPU-equipped devices automatically handle computationally intensive AI tasks for CPU-based devices, while T_{collab} represents rare instances requiring external resources beyond the local cluster capabilities.

Critically, when cloud connectivity is lost, $L_{\text{cloud}} \to \infty$ (complete failure), while the gateway and DFC-AI both continue serving locally cached tasks. The architectures diverge under a node failure: a gateway is a shared aggregation point whose loss disables every device it serves, whereas L_{DFC} depends only on the originating device, so a single failure is contained to that device.

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

4.3. Energy Consumption Model

Energy efficiency is crucial for battery-powered devices and overall system sustainability. DFC-AI's microservices architecture enables selective activation of AI components, so that only the modules a task requires are resident and running. On a resource-constrained device the path from this property to energy runs through memory occupancy: measurements on IoT gateways identify memory usage, together with container start-up time, as the decisive factors governing the performance and energy cost of deploying containerized microservices on such hardware [17]. The general modularity and selective-activation properties of microservice architectures are surveyed in [18]. Holding only the required modules resident therefore costs less than holding a monolithic image resident continuously, and it is this effect that the idle-power term $P^i_{\text{idle}}(1-\rho_i)$ below represents.

We are careful not to generalize the effect beyond that setting. In datacenter deployments, where services are replicated and orchestrated across nodes, coordination overhead can outweigh the benefit, and controlled experiments find monolithic deployments better performing at low replication [19]; that study measures throughput and response time rather than energy, so we treat it as bounding the performance claim rather than the energy claim. The direction of the result is therefore setting-dependent, and we claim it only for the device-resident case that this paper evaluates. The savings reported below accordingly arise from three effects that the model makes explicit: avoiding transmission over the wide-area path, executing on low-power local hardware instead of high-power shared hardware, and the reduced idle power that selective activation yields on the device, in that order of magnitude.

We model the energy consumption including both processing and transmission:

$$E_{\text{total}} = E_{\text{processing}} + E_{\text{transmission}} \tag{6}$$

For DFC-AI with microservices:

$$E_{\text{DFC}} = \sum_{i=1}^{N} \left(P^i_{\text{idle}} \cdot (1-\rho_i) + P^i_{\text{active}} \cdot \rho_i\right) \cdot T + E_{\text{trans}} \tag{7}$$

where:

- N = total number of microservices deployed on the device
- P^i_{idle} = idle power consumption of microservice i (watts)
- P^i_{active} = active power consumption of microservice i when processing (watts)
- ρ_i = utilization factor of microservice i (fraction of time active, $0 \le \rho_i \le 1$)
- T = time period over which energy is measured (seconds)
- $E_{\text{trans}} = E^{\ell}_{\text{pkt}} + e^{\ell}_{\text{byte}} D$ = transmission energy on link ℓ, the sum of a fixed radio cost per transmission (the connected-state promotion and tail that dominates for small frequent payloads) and a per-byte term whose slope is set from the per-gigabyte figures adopted for the three link types (0.6, 0.05, and 0.01 Wh/GB for the wide-area uplink, the on-site link, and the device-to-device mesh; see Section 5.2 for their derivation and status). The two-term form, a fixed radio cost plus a payload-proportional cost, follows the offloading energy models of [20]. The fixed term is largest for a cellular or satellite uplink and negligible for the local mesh. Processing energy is each node's active power times t_{inf}, a fraction of a watt to a few watts for a sensor or wearable, about 100 W for a shared gateway or GPU/NPU node, and a data-center figure on the order of 400 W for the cloud

The simulation evaluates the active term of this expression directly, charging each node's active power for the duration of each task it serves. In the deployments modeled here, the combination of local execution and selective activation yields the 90 to 95 percent reductions in total energy reported in Section 6, dominated by the avoided transmission term rather than by the idle-power term; a comparable ordering of magnitudes was obtained for traditional and agentic workloads in an earlier analysis by one of the authors [21].

4.4. Cost Model

We deliberately avoid absolute monetary figures, since cloud and hardware prices vary by vendor, region, and time and would date the analysis. Instead we model cost in relative terms.

The recurring cost of serving a task in the cloud has several components. The first is moving the payload over the network, which in remote energy sites means a costly cellular or satellite uplink rather than inexpensive data-center egress. The second is the cloud computation itself, measured in GPU time [22,23]. The third, when the cloud baseline is a hosted language model rather than a specialized model, is a per-token charge levied on every call. On the supply side, a gateway adds amortized infrastructure that is paid regardless of use, whereas already-owned devices contribute only a sunk, near-zero marginal cost. Which component dominates depends on the workload: for the data-heavy video of drone inspection and

ACCEPTED

worker safety, network movement and per-frame compute dominate; for token-billed agentic workloads the per-call token charge dominates and grows with every task and every reasoning step; and for a gateway the fixed infrastructure dominates whenever utilization is low.

The decisive observation is that a task resolved on the device incurs none of these cloud charges, neither network, nor cloud compute, nor tokens. The relative cost of an architecture therefore tracks the fraction of the workload it sends to the cloud, and is largely insensitive to the specific unit prices: moving any single rate by an order of magnitude leaves the comparison essentially unchanged. For the cloud and for device-first, neither of which provisions dedicated infrastructure for the task, this recurring cost is the whole cost, and we compute it relative to the cloud from the cloud-bound fraction of each workload. A gateway adds a fixed component, the amortized infrastructure that must be provisioned regardless of utilization, which dominates its total; for that component we rely on published total-cost-of-ownership studies rather than estimate prices ourselves [10,11]. The total cost combines the two:

$$C_{\text{total}} = C_{\text{recurring}} + C_{\text{fixed}} \tag{8}$$

- **Centralized cloud:** every task crosses the network and is served in the cloud, so the recurring component is maximal, while the fixed component is effectively zero under pay-as-you-go provisioning.
- **Gateway-edge:** only the escalated fraction reaches the cloud, so the recurring component is small, but a dedicated gateway tier must be acquired, powered, and maintained, giving a fixed component incurred even at low utilization.
- **DFC-AI:** only the escalated fraction reaches the cloud, and the bulk of computation runs on devices the enterprise already owns and powers, so the recurring cloud cost and the marginal device cost are near zero. Where a task exceeds device capability a shared GPU/NPU is used, but it is sized for that residual rather than the full workload, so its fixed component is a fraction of a gateway tier's and is zero where the devices suffice or the GPU/NPU is itself an owned device.

This treatment assumes that on-device computation is a sunk asset with spare capacity, so local inference adds negligible marginal cost. The assumption holds for the scenarios considered, where the devices are already deployed and powered for the operation. It would weaken in two cases: if serving the AI required provisioning new on-device compute, whose capital cost would then be amortized much as the gateway's is, and if the devices were severely battery-limited, where the added computation appears as an energy cost rather than a monetary one and is accounted for in the energy analysis instead.

For agentic workloads served by a hosted large language model (LLM), both compute and charge scale with tokens rather than payload bytes. Recent surveys of edge and on-device LLM execution document both the growing feasibility of running compact models on end devices and the privacy, latency, and offline-reliability pressures that motivate doing so [24,25]. The service time is $t^{\text{tok}} = n_{\text{pre}}/\tau_{\text{pre}} + n_{\text{dec}}/\tau_{\text{dec}}$ for n_{pre} prefill and n_{dec} decode tokens at the prefill and decode throughputs, and the cloud charge is $\pi_{\text{in}}\, n_{\text{pre}} + \pi_{\text{out}}\, n_{\text{dec}}\, n_{\text{steps}}$, growing with every reasoning step. This is the regime in which avoiding a cloud call removes the largest and most directly metered cost. The three scenarios evaluated here are vision and sensor workloads, for which compute scales with pixels and data volume through the W_c and R_{io} terms above, so we treat the token regime analytically rather than simulating it.

5. Simulation Framework and Methodology

We conducted comprehensive simulations with statistical rigor to validate the theoretical advantages of DFC-AI across realistic energy sector deployment scenarios. Our simulation framework encompasses three representative applications with 30 independent runs per scenario to establish 95% confidence intervals.

5.1. Simulation Scenarios

5.1.1. Autonomous Drone Fleet Inspection

- **Fleet Composition:** 10 drones, 1 GPU-equipped (Jetson AGX Orin, 20× processing power), 9 standard (Arm Cortex, 1× processing power)
- **Workload:** 5 MB compressed H.264/H.265 images, 6 captures/minute/drone (Poisson distributed)
- **Duty cycle:** six flight hours per day; battery endurance limits continuous flight to tens of minutes per sortie, so daily totals scale with flight hours rather than wall-clock time, giving approximately 10.8 GB per drone per day at the stated capture rate. The simulation models the active inspection period.
- **Task Complexity Distribution:** 80% simple (local processing), 15% complex (GPU required), 5% cloud-only (historical analysis)
- **Network Conditions:** 100 Mbps bandwidth (assuming high on-site network availability), 5–50 ms latency variability

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

5.1.2. Industrial Sensor Network Monitoring

- **Network Composition:** 500 sensors (350 simple, 150 smart) + 2 edge GPU servers
- **Data Generation:** 0.1Hz effective sampling rate (every 10 s), phase-staggered across sensors so that readings are not aligned to a common clock, 100 bytes to 1 KB per reading
- **Event Distribution:** 95% normal readings, 4% anomalies requiring collaboration, 1% critical events
- **Network Conditions:** Industrial ethernet (8–30 ms latency, 100 Mbps bandwidth)

5.1.3. Worker Safety Monitoring

- **System Composition:** 25 wearables, 5 vehicles, 10 cameras, 1 stationary GPU mini-PC
- **Workload:** Continuous vital signs (1Hz, 256 bytes), continuous personal protective equipment (PPE) and hazard video inference on a 2 MB frame every 3 s per camera, continuous activity and fall inference on a 50 kB inertial window every 5 s per wearable, and rare cloud-only incident reconstruction. Detection runs continuously rather than on a fixed check schedule, so that transient events are caught.
- **Network Conditions:** Remote site connectivity (50–80 ms latency, 10 Mbps)

5.2. Key Simulation Assumptions

The simulation combines inputs grounded in published measurements with parameters set as design assumptions for the scenarios. The inference work per complexity class is calibrated against on-device benchmarks [13–16], the two-term structure of the transmission-energy model from published offloading energy models [20] with per-gigabyte slopes adopted as assumptions consistent with published electricity-intensity studies [26,27], and the heavy-tailed structure of the workload from datacenter and cluster-trace studies [3–5]. The per-scenario network bandwidths, the complexity mix of each workload, the locality split, and the device counts are design assumptions chosen to represent typical energy-sector deployments rather than measured values; the sensitivity analysis in Table 1 shows how the results respond when the most consequential of these are varied. Because the inputs are measurements from commodity devices and networks, and the device-first design uses only standard, deployable components, a portable microservice runtime above the operating system together with ordinary network and interprocess calls, the comparison describes a system that can be built with current hardware rather than an idealized one. The specific assumptions are detailed below.

5.2.1. Heterogeneous Device Capabilities

Not all devices have equal processing power. In realistic deployments:

- 10–20% of devices are GPU-equipped for intensive AI workloads
- 80–90% are CPU-based devices for operational tasks
- Zero-configuration discovery enables automatic load distribution

5.2.2. Network Transmission Energy

Transmission energy combines a per-byte term, with the per-gigabyte slopes below, and a fixed per-transmission radio cost that dominates for small frequent payloads. The two-term structure follows the offloading energy models of [20]. The per-gigabyte slopes are the following:

- Cloud (wide-area uplink): 0.6 Wh/GB plus the largest fixed radio cost
- On-site link: 0.05 Wh/GB plus a small fixed cost
- Device-to-device mesh: 0.01 Wh/GB with negligible fixed cost

These three values are modeling assumptions, not measurements of our own or figures taken verbatim from a single source, and we state their provenance explicitly. Published electricity-intensity studies place fixed-line Internet transmission at 6×10^{-2} kWh/GB in 2015 and report that this intensity has halved approximately every two years, which extrapolates to a figure of order 10^{-3} kWh/GB, that is, of order 1 Wh/GB, for a contemporary access and core path [26]. Mobile access networks are consistently measured at a substantially higher intensity than fixed access over the same period [27]. We therefore adopt 0.6 Wh/GB for the wide-area uplink as a contemporary order-of-magnitude figure consistent with those trends, and set the on-site and device-to-device slopes an order and two orders of magnitude lower, respectively, reflecting the successively shorter reach and smaller shared infrastructure of those links. What the analysis depends on is this ordering across the three link types rather than the specific values: because the comparison is driven by the volume of data each architecture places on the wide-area path, scaling all three slopes by a common factor leaves the relative energy results unchanged, and the sensitivity analysis of Table 1 covers the effect of moving traffic between links. Absolute Wh/day figures should accordingly be read as indicative of magnitude rather than as measured consumption.

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

5.2.3. Inference, Queueing, and Correlated Load

Inference time scales with device throughput, so a task runs faster on a GPU/NPU than on a CPU device. The cloud is modeled as an elastic tier whose server count scales with the arrival rate to hold utilization near one half at mean load, so its typical latency is load-insensitive, while each device and each shared gateway or GPU/NPU node is a single-server queue that can saturate under load. All queues are driven by the actual arrival times, so latency responds to load rather than to a fixed mean, and each service time carries an independent multiplicative lognormal jitter of about eighteen percent. Heavy, incident-driven tasks, the complex and cloud-only classes, are generated by a self-exciting cluster process so that they arrive in correlated bursts, the regime that most stresses a shared tier; offsets from each cluster centre are symmetric, so a burst spreads on both sides of its centre. We additionally sweep the arrival rate to confirm that the ordering of the architectures is robust to load.

5.2.4. Device-First Routing

The DFC-AI router is device-first and uses only information a device can observe at decision time. A task runs on its source device whenever that device can serve its complexity class. Work that exceeds the device, the GPU-class tasks, is offloaded, and the router selects the target, a local GPU node or the cloud, by the lowest latency it can predict from the candidate throughput, the measured link bandwidth and payload size, and a causal estimate of each node's queue formed from the tasks already dispatched to it. These predictions use the expected service time and do not use the per-task jitter or any knowledge of future arrivals, so the router is not an oracle; the realized latency, including the actual queueing and jitter, follows from the queue model above. Because the cloud is always available as a fallback, when a local GPU/NPU is saturated or a local link is too slow for a large payload the offloaded work moves to the cloud, so DFC-AI tail latency tracks cloud performance to within the local dispatch overhead of about 2.5 ms rather than diverging from it. Tasks that require cloud-scale resources are sent to the cloud by definition.

5.2.5. Cost Model Assumptions

- **Recurring cost driver:** the cloud-bound data volume, taken directly from the workload as payload size times task rate, which determines cloud ingest, storage, and computation.
- **Device marginal cost:** treated as near zero, since the devices are enterprise-owned assets already provisioned and powered for the operation.
- **Gateway fixed cost:** a dedicated tier whose cost is incurred regardless of utilization, on top of its escalated cloud traffic.
- **Scope:** the comparison covers both specialized models served on owned or rented compute and hosted language models billed per token; the device-first advantage is largest under per-token billing, where every avoided call removes a direct charge.

Because the index is built from data volume rather than prices, the resulting comparisons are illustrative and architectural. They show how cost scales with each design, not what a particular deployment would pay.

5.2.6. Inference Work and Device Scaling

Inference time follows the service-time model $t_{\text{inf}} = (W_c/S_d + R_{\text{io}}D)\,\eta_d$ of equation (1):

- Base work W_c of about 2, 30, 120, and 400 ms for the simple, moderate, complex, and cloud-only classes on an Arm Cortex-A reference CPU, calibrated against published benchmarks [13–16]; dividing by device throughput reproduces measured GPU/NPU latencies
- Data-movement term R_{io} of about 4 ms per megabyte of payload
- Compute term scaled by the node's throughput, so a GPU/NPU runs a task faster than a CPU device, and a portable-runtime factor on the device CPU path
- Relative device throughputs used in the scenarios, with the reference CPU set to one: GPU-class nodes (the inspection drone's onboard GPU, the sensor-network edge GPUs, and the worker-safety mini-PC) at fifteen to twenty, vehicle compute at two, smart sensors at one half, wearables at one fifth, simple sensors and cameras at one tenth, and the cloud at one hundred

5.3. Statistical Methodology

Each scenario was run 30 times with different random seeds (a fixed base seed plus the run number) to characterize variability. We computed 95% confidence intervals using Student's t-distribution and used two-sample Welch t-tests to check that the differences between architectures are not due to chance; the achieved significance is $p < 0.001$ for both the latency and the energy comparisons. The simulation is additionally cross-checked against the closed-form latency model of Section 4.

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

6. Simulation Results and Analysis

To validate our theoretical models, we developed a simulation framework covering three representative energy sector scenarios. These simulations incorporate realistic operational parameters, network conditions typical of remote industrial sites, and workload distributions based on established patterns from cloud computing literature. Across the three scenarios, DFC-AI reduced the typical latency by 79 to 93 percent and energy use by 90 to 95 percent, kept 90 to 100 percent of capability when a local node failed, and cost a small fraction of either alternative. The rest of this section shows where these gains come from, how they hold up under stress, and the conditions under which they narrow.

6.1. Latency Performance

DFC-AI delivers the lowest typical latency in every scenario. Table 2 reports the median, the latency a typical request experiences, and DFC-AI cuts it by 79 to 93 percent against the cloud. The differences between architectures are large and statistically significant (two-sample Welch t-test over 30 runs, $p < 0.001$), with 95% confidence half-widths on the per-run DFC-AI median of 3.4, 0.2, and 4.9 percent for the drone, sensor, and worker-safety scenarios respectively.

Two further views matter alongside the median. The tail, the 99th-percentile latency in Table 3, is the slowest request in a hundred, the figure a safety deadline must tolerate. The mean, used in the sensitivity analysis below, is the simple average, which can rise when heavy tasks are pushed off the device even while the typical request is unchanged. A system can have a fast typical case and a slow tail at the same time, so we report all three.

Table 2: Median latency by scenario (ms). Values are the mean of the per-run median over 30 independent runs.

Architecture	Drone	Sensor	Safety
Cloud-Centric	490.1	40.0	80.2
Gateway-Edge	445.9	8.2	20.2
DFC-AI	**34.8**	**4.6**	**16.6**
DFC-AI Improvement	93%	88%	79%

The worker-safety scenario is where the tail separates the architectures. In the drone scenario all three tails sit near 500 ms, because the slowest requests there are the five percent of tasks that are cloud-bound by design and that every architecture sends to the cloud; this leaves the DFC-AI median untouched. On the worker-safety site, whose remote uplink is only 10 Mbps, shipping a 2 MB video frame to the cloud costs roughly 1.7 s, so the cloud

Table 3: Tail latency by scenario, 99th percentile (ms). Values are the mean of the per-run 99th percentile over 30 independent runs.

Architecture	Drone	Sensor	Safety
Cloud-Centric	505.1	43.0	1693.3
Gateway-Edge	498.1	11.4	208.6
DFC-AI	**497.5**	13.5	**49.6**

carries second-scale tails for the frames it sends over that uplink. The gateway and DFC-AI both process the frame on the local GPU/NPU instead, so both keep sub-second tails: the gateway holds its tail at 209 ms and DFC-AI at 50 ms. The gateway tail is higher because it routes every task through the shared node over the on-site link, whereas DFC-AI runs routine work on the originating device and sends only the heavy frames to the shared node, leaving it less loaded. In the sensor network the DFC-AI tail of 13.5 ms sits marginally above the gateway's 11.4 ms, because the slowest one percent there are the critical events that both architectures send to the cloud, and the device-first path pays a small local dispatch overhead before doing so.

To examine behavior under contention rather than at nominal load, we sweep the arrival rate of the worker-safety workload. Table 4 reports median and 99th-percentile latency as the rate rises from half to eight times nominal.

Table 4: Worker-safety latency under load: median/p99 (ms) versus arrival-rate multiplier. Tail values here are pooled across runs at each load level, a different estimator from Table 3, which averages the per-run tail; the two therefore differ slightly at the nominal rate.

Rate	Cloud	Gateway	DFC-AI
0.5×	80.2/1692.0	20.2/202.0	**16.6/42.5**
1×	80.2/1692.8	20.2/206.2	**16.6/46.9**
2×	80.2/1695.9	20.2/223.7	**16.6/66.1**
4×	80.2/1714.2	20.2/1069.8	**16.8/910.3**
8×	80.2/2458.6	26.0/1702.0	**17.0/1697.8**

The DFC-AI median is flat and lowest at every load, holding between 16.6 and 17.0 ms across the entire range, because routine work stays on per-device queues that never approach saturation; the figure beside it, the tail, grows only as the load climbs. As the rate rises, both the gateway and DFC-AI keep their tails bounded by cloud performance: each sends overflow to the cloud once its shared GPU/NPU is predicted to be slower than the wide-area path, so neither tail climbs past the cloud's. The DFC-AI tail stays below the gateway's at every rate, from 42.5 against 202.0 at half nominal to 1697.8 against 1702.0 at eight times nominal, because DFC-AI runs rou-

ACCEPTED

tine work on the originating device and offloads only the heavy frames, so its shared node saturates later than the gateway's, which must absorb every task. At eight times nominal both edge tails sit below the cloud's 2458.6 ms. The median is unaffected throughout because it is set by on-device work, where DFC-AI keeps its lead. Adding a second shared GPU/NPU would lower both edge tails further by reducing the share that spills to the cloud. Each entry is pooled over many independent runs to give a stable estimate of the tail.

The drone inspection scenario reveals the critical impact of local processing: with 5 MB compressed images transmitted at 100 Mbps, DFC-AI processes these images locally in under 40 ms, achieving the sub-second response times essential for autonomous navigation and immediate defect detection. In the worker safety scenario, routine vital-sign checks resolve on the wearables in under 20 ms, while the stationary GPU mini-PC handles video-analytics frames in roughly 30 ms, demonstrating the importance of strategic GPU placement in DFC-AI deployments.

6.2. Sensitivity to Operating Conditions

The result of this subsection is robustness: as the operating conditions worsen, the typical DFC-AI request is essentially unaffected, the worst case stays close to the cloud path, and only the average creeps up, and only modestly. To show this, we vary three parameters one at a time in the drone scenario and report the effect on DFC-AI latency. Table 1 summarizes the sweeps. The cloud-centric reference, whose latency is set by the wide-area uplink, stays at a median of 490 ms, a mean of 491 ms, and a 99th percentile of 505 ms throughout.

The median and the tail barely move, for a simple reason. The typical request is handled on the device whatever these conditions are, so the median stays near 35 ms, and the worst case is always the same small slice of inherently cloud-bound tasks, so the tail stays near the cloud value of 500 ms. What changes is the average, because these conditions act on the heavy tasks that get offloaded, which sit between those two fixed points. As the local mesh bandwidth falls from 1000 to 25 Mbps the mean rises from 61.8 to 119.4 ms, as more of the large 5 MB images are sent to the cloud rather than across a slow local link; once the link is slow enough that the cloud is always the faster choice, further reduction changes nothing, which is why the 50 and 25 Mbps rows coincide. As the cloud-bound fraction rises from 5 to 30 percent the mean rises from 61.8 to 181.9 ms and the share of work handled locally falls from 94.9 to 68.7 percent, the quantity that sets autonomy under a cloud outage. As device throughput is cut to a quarter the mean rises only from 61.8 to 71.0 ms and the median to 43.6 ms.

The cloud-bound sweep also shows the one condition under which the DFC-AI tail rises above the cloud reference. A task that is ultimately cloud-bound is triaged on the device before dispatch, so it pays the local interprocess handoff of about 2.5 ms in addition to the full cloud path. As the cloud-bound fraction rises, these tasks come to populate the 99th percentile, and the DFC-AI tail moves from 498.1 to 507.2 ms against a cloud reference of 505 ms, crossing it between 20 and 30 percent. The tail therefore tracks the cloud path to within the dispatch overhead rather than being strictly bounded by it. In every case the DFC-AI mean stays far below the cloud-centric mean of 491 ms, because the router keeps work on the device when local execution is faster and falls back to the cloud when it is not.

6.3. Network Resilience and Autonomy

The resilience difference that matters is not surviving a cloud outage, where the gateway and DFC-AI both cope, but surviving the loss of a single local node, where they diverge sharply. Table 5 compares operational capability under two distinct failure modes: a cloud outage with the local network intact, and the failure of a single local node.

Table 5: Operational capability under cloud outage (share of tasks still served locally) and single-node failure (share of the fleet retaining intelligence).

Failure Mode	Scenario	Gateway	DFC-AI
Cloud outage	Drone	94.7%	94.8%
	Sensor	99.0%	99.0%
	Safety	99.9%	100.0%
Single-node failure	Drone	0.0%	90.0%
	Sensor	50.0%	99.8%
	Safety	0.0%	97.6%

A cloud-centric system retains no capability under either failure mode, so it is omitted from the table. This separates two failure modes that are often conflated. When cloud connectivity is lost but the local network is intact, a gateway keeps serving its cached models and retains 94.7 to 99.9 percent of capability, essentially matching DFC-AI's 94.8 to 100 percent. The architectures diverge under a local failure. A gateway is a shared aggregation point: in the worker-safety and drone scenarios a single gateway serves the entire group, so its loss removes intelligence from every device at once, while with two aggregation nodes in the sensor network a single failure costs half the fleet. DFC-AI distributes intelligence across the devices, so a single node failure removes only that node and

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

leaves 90.0 to 99.8 percent of the fleet operational. The single-node figure counts the share of devices that remain reachable and powered after one failure; devices whose heaviest tasks exceed their own capability lose that portion of their work when the shared node fails, while retaining their local work. This single-point-of-failure exposure, rather than the cloud-outage fraction, is the substantive resilience difference, and for safety-critical operations in remote energy facilities it can be decisive. These figures hold for the workload mix of the modeled scenarios, in which most tasks are local. The share of work that must reach the cloud sets an upper bound on autonomy, and Table 1 shows the locally handled fraction falling from 94.9 to 68.7 percent as the cloud-bound share rises from 5 to 30 percent, so the resilience claims should be read against a workload of this kind rather than as architecture-independent guarantees.

6.4. Energy Efficiency

DFC-AI uses 90 to 95 percent less energy than the cloud in every scenario, for one underlying reason: it keeps data off the power-hungry wide-area path. Table 6 shows total energy consumption including both processing and transmission.

Table 6: Daily energy consumption by scenario (Wh/day), extrapolated to a continuous 24-h basis for comparability; for the duty-cycled drone deployment the absolute figures scale with flight hours, leaving the percentage savings unchanged.

Architecture	Drone	Sensor	Safety
Cloud-Centric	462.5	636.1	1074.9
Gateway-Edge	95.5	110.0	233.0
DFC-AI	**36.0**	**63.8**	**56.5**
DFC-AI Savings	92.2%	90.0%	94.7%
Gateway Savings	79.3%	82.7%	78.3%

The energy savings are large in all three scenarios. For the data-heavy drone and worker-safety workloads they come primarily from eliminating the transmission of 5 MB images and 2 MB video frames to the cloud, which carries a high per-gigabyte energy cost (the 0.6 Wh/GB assumed for the wide-area uplink) that local processing avoids. For the sensor network the savings instead arise from performing a very large number of routine inferences on low-power devices rather than on high-power cloud GPUs/NPUs, and from avoiding the fixed radio cost incurred on every one of those many small cloud transmissions. In every case the dominant effect is the same, keeping work off the wide-area path. Multiplied across thousands of continuously operating devices, the effect is substantial, lowering the energy footprint of the deployment and extending battery life for remote installations.

6.5. Economic Analysis

DFC-AI costs a small fraction of either alternative, and a gateway can cost more than the cloud it was meant to replace. Table 7 presents the relative total cost of ownership of deploying the AI capability under each architecture, normalized so that the cloud-centric case equals one.

The recurring component, computed from the cloud-bound volume of each workload, is small for both edge architectures and identical between them: 0.050 for the drone scenario, 0.010 for the sensor network, and 0.001 for worker safety, in both the gateway and device-first cases, since both send the same inherently cloud-bound work onward. What separates them is the fixed component. Device-first adds none, because the work runs on devices already owned and powered. A gateway adds a dedicated tier, and for that component we take a three-year total-cost-of-ownership comparison of hyperscale edge configurations against regional data centers, which places the edge platform 35 to 55 percent above the cloud baseline, the difference arising from real estate, cooling, and maintenance staffing [10], corroborated by [11]. The first of these is an industry analysis rather than a peer-reviewed measurement, and the comparison is workload-dependent: where gateway hardware is heavily and continuously used, its amortized cost falls below the cloud.

Table 7: Relative total cost of ownership for deploying the AI capability (cloud-centric = 1.00).

Architecture	Relative TCO
Cloud-centric	1.00 (baseline)
Gateway-edge	1.35–1.55
DFC-AI	0.001–0.05

The three values reflect how each architecture incurs cost. The cloud bills per use and sets the baseline. A gateway adds dedicated hardware whose amortized cost, together with power, cooling, and maintenance, is paid regardless of utilization, so independent studies place its total above the cloud at the moderate utilization typical of provisioned-for-peak deployments, and below the cloud only when the hardware is heavily and continuously used. Device-first runs the bulk of the work on devices already owned and amortized for their primary function and escalates only the small remainder, and where a task exceeds device capability it uses a shared GPU/NPU sized for that residual rather than a full-workload gateway. Its cost therefore falls in the range of the recurring index alone, from 0.001 to 0.05 across the three scenarios. The

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

advantage is largest for data-heavy workloads, where local processing displaces a large cloud-egress bill; for a data-light workload there is little cloud cost to displace, so a dedicated GPU/NPU there, like a gateway, does not beat the cloud.

These cost savings, although large, are not the dominant economic argument for the data-light scenarios. For sensor networks and worker safety the primary value is the prevention of outages and accidents. A single avoided production shutdown or prevented worker injury, made possible by intelligence that keeps operating through a network outage, outweighs the cost difference by a wide margin. The low incremental cost therefore strengthens rather than weakens the case for DFC-AI, because it makes resilience the decisive economic factor.

6.6. The Role of Data Sovereignty

If a gateway's total cost is, at the utilization typical of real deployments, no lower than the cloud's, its appeal must rest on something other than saving money.

Surveys of industrial edge adoption report that the leading driver of gateway deployment is data sovereignty, the requirement to keep regulated or sensitive data within a jurisdiction or on the premises, rather than cost reduction [28]. Latency and autonomy follow, while cost and complexity are cited as barriers rather than motivations. A gateway delivers data sovereignty by keeping processing local, but pays for it with the fixed infrastructure described above. DFC-AI provides the same property, since data is processed on the originating device and need never leave it, yet it does so without a dedicated tier, because the compute already exists in equipment the enterprise owns and operates. The device-first approach therefore offers the sovereignty that genuinely motivates edge deployments while avoiding the fixed cost that, in practice, often makes a gateway more expensive than the cloud it was meant to replace.

6.7. Processing Location Distribution

Table 8 shows where DFC-AI processes work in each scenario, as a share of tasks and as a share of data volume.

The shared GPU/NPU is an owned device in the drone fleet, a GPU carried on one of the drones, and dedicated hardware in the other two scenarios, two edge GPUs in the sensor network and one mini-PC in worker safety. Two patterns stand out. The device handles the large majority of tasks everywhere, from three quarters in worker safety to more than four fifths in the drone fleet, while the cloud receives almost nothing, at most five percent. The task share and the volume share diverge sharply wherever a few heavy items dominate the bytes: in worker safety the device handles three quarters of the tasks but a tenth of a percent of the volume, because the video frames are rare but large and all route to the GPU/NPU. This is why the cost advantage tracks the workload. Where the GPU/NPU is an owned device, or where it displaces a large cloud-egress bill as in the drone and safety cases, device-first cost falls to the low end of its range; where a dedicated GPU/NPU serves a data-light workload, as it can in the sensor network, less cloud cost is displaced and the advantage narrows.

6.8. Comprehensive Performance Comparison

Figure 4 compares the architectures on latency, energy, and resilience, averaged across scenarios. Latency and energy are median reductions relative to cloud, and resilience is the fraction of the fleet that retains intelligence under a single-node failure. Operating cost is treated separately in the economic analysis above.

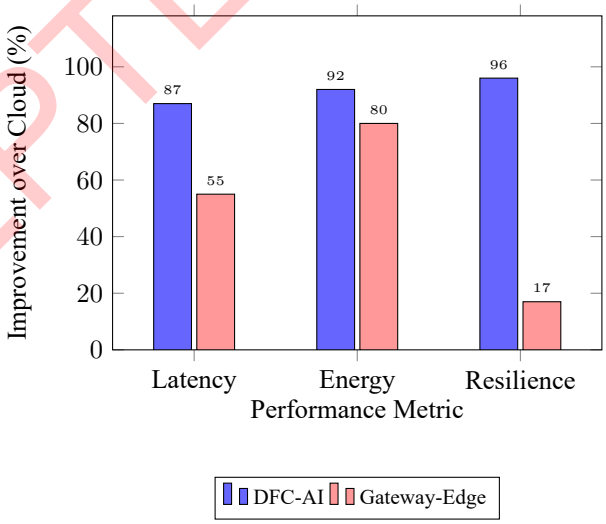


Figure 4: Improvement over cloud in latency, energy, and resilience, averaged across scenarios (%).

6.9. Validation of Theoretical Models

Our simulation results strongly validate the theoretical predictions, as shown in Table 9.

The analytical median latency, taken at the low-load limit where queueing is negligible, matches the simulated median to within three percent in every scenario (1.2, 0.3, and 2.4 percent for the drone, sensor, and worker-safety cases), and the small residual reflects the stochastic jitter present in the simulation but absent from the closed-form prediction. Because the analytical and simulated models

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

Table 8: Where DFC-AI processes tasks, by scenario (%).

Scenario	Basis	Device	GPU/NPU	Cloud
Drone	tasks, volume	81.7	13.3	5.0
Sensor	by task count	28.5	70.5	1.0
	by data volume	77.1	21.9	1.0
Safety	by task count	75.0	25.0	0.0
	by data volume	0.1	99.9	0.0

Table 9: Theoretical model versus simulation: DFC-AI median latency (ms).

Scenario	Theory	Simulation	Error
Drone	34.4	34.8	1.2%
Sensor	4.6	4.6	0.3%
Safety	17.0	16.6	2.4%

share the same generative assumptions, this agreement establishes internal consistency rather than external validation. Robustness to the conditions the closed form omits is shown separately by the load sweep of Table 4, where the ordering of the architectures and the flatness of the DFC-AI median persist as the arrival rate rises by more than an order of magnitude.

6.10. Statistical Significance

All performance improvements were tested for statistical significance using two-sample Welch t-tests with 30 independent runs per scenario (df $= 29$):

- Latency reduction: $p < 0.001$ (highly significant)
- Energy savings: $p < 0.001$ (highly significant)

The cost comparison is structural rather than stochastic: the cloud and device-first values follow directly from the workload and the gateway range is taken from published studies, so cost is reported as a relative comparison rather than a quantity requiring a significance test.

7. Discussion

7.1. The Primacy of Autonomy

While our results demonstrate performance and economic advantages, the fundamental value of DFC-AI lies in ensuring continuous intelligent operation. The ability to maintain 95% to 100% functionality during network outages is essential for:

- **Safety-Critical Operations:** Worker safety systems must function regardless of connectivity. A fall detection system that fails during a network outage defeats its purpose.
- **Remote Operations:** Equipment in isolated locations cannot depend on network availability. Offshore platforms, remote pipelines, and desert installations often experience extended connectivity loss.
- **Time-Critical Decisions:** Real-time control requires immediate response. The roughly 500 ms cloud latency for drone image processing makes collision avoidance challenging.
- **Operational Continuity:** Production systems cannot halt due to network issues. Refineries and processing plants require continuous operation.

7.2. The Gateway Paradox

Our finding that gateway-based architectures can have comparable or higher costs than cloud solutions for certain workloads challenges conventional wisdom but reflects operational reality reported in independent total-cost-of-ownership studies [10,11]:

- Edge infrastructure requires fixed costs (hardware, maintenance, operations) regardless of utilization
- Both edge servers and cloud resources may be needed, potentially doubling infrastructure
- Edge servers require physical security, cooling, and maintenance in industrial environments
- Scaling edge infrastructure requires significant capital investment
- Gateway systems are typically cloud-dependent extensions with limited autonomous capability

DFC-AI reduces infrastructure costs by maximizing the utilization of devices enterprises already own for operational purposes, while still allowing for strategic edge deployments where centralized processing provides clear benefits.

7.3. Heterogeneous Fleet Reality

Our simulation's assumption that only 10–20% of devices need GPU capabilities reflects industrial reality and economic practicality:

ACCEPTED

- Not every drone needs expensive GPU processing. One GPU drone can serve a fleet.
- Simple sensors perform adequately with basic processors for routine monitoring
- Strategic placement of GPU resources (e.g., stationary mini-PCs for worker safety) optimizes performance
- Intelligent task routing ensures GPU resources are used where most beneficial
- Zero-configuration discovery means GPU resources are automatically shared

This heterogeneous approach optimizes cost-performance while maintaining full fleet capability. For example, in our worker safety scenario, a single stationary GPU mini-PC provides video analytics and complex processing for 25 workers, 10 cameras, and 5 vehicles, demonstrating efficient resource utilization.

7.4. Scalability for Billions of Agents

The transition to billions of autonomous agents makes DFC-AI not just advantageous but essential:

- **Network Mathematics:** 1 billion agents × 1 MB/s = 1000 TB/s bandwidth requirement
- **Processing Load:** Centralized processing of billions of agents would require unprecedented data center capacity beyond current global infrastructure
- **Latency Cascade:** Even small delays multiply across billions of interdependent agents
- **Failure Propagation:** Centralized failures affect all agents simultaneously

DFC-AI's distributed model scales linearly with agent count, avoiding these exponential scaling challenges.

8. Related Work

Edge computing moves computation closer to data sources to cut latency and bandwidth [2], and the subsequent work on edge intelligence has pushed deep learning training and inference toward the network edge in industrial and consumer settings [29]. These approaches reduce dependence on the cloud, yet most retain centralized control and assume connectivity for coordination, so the end devices remain managed clients rather than autonomous participants.

Fog computing introduced a distributed tier between devices and the cloud [30] and framed a research agenda for low-latency, geographically distributed services [31, 32]. Fog deployments generally rely on dedicated intermediate infrastructure rather than the end devices themselves, which carries the fixed-infrastructure cost discussed in the economic analysis. Gateway-based edge architectures [33] place intelligence at aggregation points but leave end devices as data collectors and remain cloud-dependent for control and updates [9].

Collaborative edge inference partitions a model between a device and a server to balance latency and energy. Neurosurgeon selects a layer-level split point between the device and the datacenter [34], and distributed deep neural networks extend this idea across cloud, edge, and end devices with early exits [35]. These methods improve a single offloading decision but still presume a reachable server tier, so they degrade when connectivity is lost. The same limitation carries into the current generation of agentic workloads: recent reviews of edge and on-device LLM execution catalogue compression, partitioning, and device-cloud collaboration techniques, and identify offline reliability as an open requirement rather than a solved one [24,25]. Microservice designs for the Internet of Things (IoT) show the modularity and selective-activation benefits that DFC-AI also uses [18,19], and studies of industrial IoT document the challenges that motivate local processing [1], yet both typically assume cloud or gateway orchestration. Resource discovery in edge and IoT systems is usually handled by a cloud-managed control plane that caches only a subset of state at the edge [9]. Across these lines of work, resilient operation under disconnection or node loss is rarely treated as a first-class design goal, an observation we draw from the surveyed literature rather than from any single study.

DFC-AI differs from these approaches in three respects. First, intelligence is device-first: agents run as microservices that originate on the end device and extend outward only when a task exceeds local capability, rather than starting at a gateway, a fog node, or the cloud. Second, devices form clusters through zero-configuration discovery, without a cloud-managed control plane, so a cluster keeps functioning when the wide-area link is down. Third, resilience is structural, because intelligence is distributed across the devices: a cloud outage or a single-node failure removes only the affected element rather than disabling a shared tier, which the modeled scenarios quantify at 94.8 to 100 percent capability under a cloud outage and 90.0 to 99.8 percent under a single-node failure. The combination of device-first microservices, zero-configuration discovery, and distributed resilience is what separates DFC-AI from edge, fog, and collaborative-inference designs that keep control, coordination, or the first point of execution off the device.

ACCEPTED

2026, Vol. 3, Article ID. 2026.0043
https://doi.org/10.69709/CAIC.2026.142241

9. Conclusions

This paper evaluated Device-First Continuum AI (DFC-AI) for autonomous operations in the energy sector through mathematical modeling and comprehensive simulations of drone inspections, sensor networks, and worker safety systems.

Our analysis demonstrates that DFC-AI fundamentally changes how industrial AI systems operate by processing data where context naturally resides. Rather than moving massive amounts of contextual information to distant processors, DFC-AI enables devices to make decisions using their immediate environmental understanding. For instance, a drone interprets its own visual feed, a sensor analyzes its own readings, a wearable monitors its own worker. This context-first approach cuts the typical latency by 79 to 93 percent and energy use by 90 to 95 percent across the modeled scenarios, while keeping near-full operational capability both when the cloud is unreachable and when an individual node fails. Because the device-first router falls back to the cloud whenever local execution would be slower, DFC-AI tail latency tracks cloud-centric performance to within the local dispatch overhead across every condition we tested, including heavy load and weak local networks, while staying ahead on median latency, energy, resilience, and cost. The advantage is largest for the predominantly local workloads typical of the energy sector and narrows toward parity only when the workload is inherently cloud-bound.

A second mechanism accounts for the results in the heterogeneous scenarios. Zero-configuration discovery lets devices form local clusters in which a GPU- or NPU-equipped node serves the tasks that exceed a weaker device, so a single GPU-equipped drone handles the complex vision tasks of a fleet while the remaining drones resolve routine work locally. In the processing-location results of Table 8 this appears as a task share concentrated on the originating device and a data-volume share concentrated on the shared node.

The cost consequence follows from the same structure rather than from any new expenditure: the bulk of the computation runs on devices that the operator has already acquired, powered, and amortized for their primary function, so the marginal cost of the residual is small relative to either a metered cloud bill or a dedicated gateway tier.

Two limitations bound these conclusions. First, the results are model-based; the analytical and simulated models share their generative assumptions, so their agreement establishes internal consistency and not empirical validation. Second, the per-scenario device counts, workload mixes, locality splits, and network parameters are design assumptions rather than measurements, and the transmission-energy slopes are adopted values, so the absolute figures should be read as indicative of magnitude while the ordering of the architectures, which the sensitivity and load sweeps show to be robust, carries the weight of the argument.

Future work should therefore extend these evaluations to field trials in operational energy facilities, measuring latency, energy, and availability directly on deployed equipment, and should extend the workload set to token-billed agentic tasks, predictive maintenance, and grid optimization, where the cost structure differs from the vision and sensor workloads modeled here.

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ARM	Advanced RISC Machines, now Arm Ltd.
CPU	Central Processing Unit
DFC-AI	Device-First Continuum AI
FTC	Future Technologies Conference
GB	Gigabyte
GPU	Graphics Processing Unit
IoT	Internet of Things
LLM	Large Language Model
MB	Megabyte
Mbps	Megabits per second
ms	Millisecond
NPU	Neural Processing Unit
p99	99th percentile
PPE	Personal Protective Equipment
TB	Terabyte
TCO	Total Cost of Ownership
Wh	Watt-hour

Author Contributions

Conceptualization: S.M.A., F.A., J.H., and M.B.; Methodology: S.M.A. and B.A.; Software, Validation, and Formal Analysis: S.M.A. and M.B.; Visualization and Writing—Original Draft: S.M.A.; Writing—Review and Editing: S.M.A., F.A., M.B., B.A., and J.H.; Supervision: S.M.A. All authors have read and agreed to the published version of the manuscript.

Availability of Data and Materials

The study uses no third-party datasets and no data from human or animal subjects. The simulation code and the generated result files are not publicly available, because the simulator is proprietary to mimik Technology Inc. and forms part of the company's internal tooling. Every input required to reproduce the reported results is nonethe-

ACCEPTED

less specified in the manuscript, and we have taken care that this is so: the service-time, queueing, energy and cost models with their governing equations in Section 4; the per-scenario device counts, workload rates, payload sizes, complexity mixes and network parameters in Section 5.1; and the benchmark-derived inference work per complexity class, the relative device throughputs, the runtime overhead and inter-process handoff factors, the jitter distribution, the transmission-energy slopes and the random-seeding scheme in Sections 5.2 and 5.3. The parameters and algorithms are therefore described in sufficient detail to permit independent reimplementation, and the generated datasets are available from the corresponding author on reasonable request.

Consent for Publication

Not applicable. The manuscript contains no identifiable photographs, videos, individual-level data, or other personal information relating to human participants. The worker-safety scenario is a simulated deployment; no human subjects were involved and no personal data were collected or processed.

Conflicts of Interest

S.M.A., F.A., M.B. and J.H. are employed by mimik Technology Inc., the company that develops the Device-First Continuum AI technology evaluated in this manuscript, and they hold equity in that company. S.M.A. is Executive Chairman and a co-founder, F.A. is Chief Executive Officer and founder, M.B. is Chief Technology Officer, and J.H. is Chief Architect. They are named inventors on patents assigned to the company. B.A. is employed by Saudi Aramco, a prospective industrial user of architectures of the type evaluated here, and holds no financial interest in mimik Technology Inc. No sponsorship, consultancy arrangement, or third-party payment was received in connection with this work. We state explicitly that no competing interests exist beyond these employment, ownership and inventorship interests.

Funding

No external funding was received for this research. The work was carried out using the internal resources of the authors' affiliated organizations. There are no funders or grant numbers to report.

Acknowledgments

The authors recognize the entire mimik team for their relentless dedication, ingenuity, and passion.

AI Declaration

The authors used Claude (Anthropic) for language editing of author-written text in LaTeX, for refining the figures from drafts drawn by the authors and for rendering figures from the authors' own simulation output, and for debugging and documenting the Python simulation code. The architecture, the analytical models, the simulation design, the parameter choices, the results, their interpretation, and the conclusions are the authors' own. All citations were verified by the authors against the original sources, and the authors reviewed all AI-assisted output and accept full responsibility for the content of this manuscript. No other language-editing, proofreading, or writing-support service was used.

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Inform.*, vol. 14, no. 11, pp. 4724–4734, 2018. [CrossRef]

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016. [CrossRef]

[3] L. Barroso, J. Clidaras, and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, 2nd ed. San Rafael, CA, USA: Morgan & Claypool, 2013. [CrossRef]

[4] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*. New York, NY, USA: Association for Computing Machinery, 2012, Art. no. 7. [CrossRef]

[5] C. Reiss, J. Wilkes, and J. Hellerstein, *Google cluster-usage traces: Format + schema*. Mountain View, CA, USA: Google Inc., 2011. [Online]. Available: https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md

[6] S. M. Alamouti, F. Arjomandi, and M. Burger, "Hybrid Edge Cloud: A pragmatic approach for decentralized cloud computing," *IEEE Commun. Mag.*, vol. 60, no. 9, pp. 16–29, Sep. 2022. [CrossRef]

[7] F. Arjomandi, S. Alamouti, M. Burger, and B. Altakrouri, "Building blocks to empower cognitive internet with Hybrid Edge Cloud," in *Proceedings of the Future Technologies Conference (FTC) 2024, Volume 3*, Lecture Notes in Networks and Systems, vol. 1156. Cham, Switzerland: Springer, 2024. [CrossRef]

[8] S. Alamouti, F. Arjomandi, M. Burger, and H. Gün, "Device First Continuum AI (DFC-AI): Realizing human-like AI," in *Proceedings of the Future Technologies Conference (FTC) 2025*, Lecture Notes in Networks and Systems. Cham, Switzerland: Springer, 2025, pp. 482–495. [CrossRef]

[9] R. Vaño, I. Lacalle, P. Sowiński, R. S-Julián, and C. E. Palau, "Cloud-native workload orchestration at the edge: A deployment review and future directions,"

ACCEPTED

Sensors, vol. 23, no. 4, 2023, Art. no. 2215. [CrossRef] [PubMed]

[10] J. Madden, "The economics of edge computing," *Mobile Experts Industry Analysis*, Oct. 2020. Accessed: Aug., 2026. [Online]. Available: https://www.edgecomputing-news.com/news/analysis-economics-of-edge-computing/.

[11] P. Nikolaou et al., "Total cost of ownership perspective of Cloud vs Edge deployments of IoT applications," in *Computing at the EDGE: New Challenges for Service Provision*, G. Karakonstantis and C. J. Gillan, Eds., Cham, Switzerland: Springer, 2022, pp. 141–161. [CrossRef]

[12] NVIDIA, "Jetson AGX Orin developer kit," 2024. Accessed: Aug., 2026. [Online]. Available: https://developer.nvidia.com/embedded/jetson-agx-orin-developer-kit.

[13] C. Banbury et al., "MLPerf Tiny Benchmark," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), Track Datasets and Benchmarks*, 2021, vol. 1. [CrossRef]

[14] V. J. Reddi *et al.*, "MLPerf Inference Benchmark," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Computer Architecture (ISCA)*, Valencia, Spain, May 30–Jun. 3, 2020, pp. 446–459. [CrossRef]

[15] D. K. Alqahtani, M. A. Cheema, and A. N. Toosi, "Benchmarking deep learning models for object detection on edge computing devices," in *Service-Oriented Computing (ICSOC 2024)*, Lecture Notes in Computer Science, vol. 15404. Singapore: Springer, 2025, pp. 142–150. [CrossRef] [PubMed]

[16] NVIDIA, *Jetson benchmarks*. Santa Clara, CA, USA: NVIDIA Developer, 2024. Accessed: Aug., 2026. [Online]. Available: https://developer.nvidia.com/embedded/jetson-benchmarks.

[17] S. M. Raza, J. Jeong, M. Kim, B. Kang, and H. Choo, "Empirical performance and energy consumption evaluation of container solutions on resource constrained IoT gateways," *Sensors*, vol. 21, no. 4, 2021, Art. no. 1378. [CrossRef] [PubMed]

[18] C. Pahl and P. Jamshidi, "Microservices: A systematic mapping study," in *Proc. 6th Int. Conf. Cloud Comput. Serv. Sci. (CLOSER)*, Rome, Italy, 2016, vol. 1, pp. 137–146. [CrossRef]

[19] G. Blinowski, A. Ojdowska, and A. Przybyłek, "Monolithic vs. microservice architecture: A performance and scalability evaluation," *IEEE Access*, vol. 10, pp. 20,357–20,374, 2022. [CrossRef]

[20] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016. [CrossRef]

[21] S. Alamouti, "Quantifying energy and cost benefits of Hybrid Edge Cloud: Analysis of traditional and agentic workloads," *arXiv*, 2025. [CrossRef]

[22] Amazon Web Services, "Amazon EC2 G4 instances pricing," 2024. Accessed: Aug., 2026. [Online]. Available: https://aws.amazon.com/ec2/instance-types/g4/.

[23] Microsoft Azure, "GPU-optimized virtual machine sizes pricing," 2024. Accessed: Aug., 2026. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/.

[24] Y. Zheng, Y. Chen, B. Qian, X. Shi, Y. Shu, and J. Chen, "A review on edge large language models: Design, execution, and applications," *ACM Comput. Surv.*, vol. 57, no. 8, 2025, Art. no. 209. [CrossRef]

[25] O. Friha, M. A. Ferrag, B. Kantarci, B. Cakmak, A. Ozgun, and N. Ghoualmi-Zine, "LLM-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 5799–5856, 2024. [CrossRef]

[26] J. Aslan, K. Mayers, J. G. Koomey, and C. France, "Electricity intensity of internet data transmission: Untangling the estimates," *J. Ind. Ecol.*, vol. 22, no. 4, pp. 785–798, 2018. [CrossRef] [PubMed]

[27] H. Pihkola, M. Hongisto, O. Apilo, and M. Lasanen, "Evaluating the energy consumption of mobile data transfer—From technology development to consumer behaviour and life cycle thinking," *Sustainability*, vol. 10, no. 7, 2018, Art. no. 2494. [CrossRef]

[28] N. Jahnke, M. Rohde, and T. Kraus, "Edge computing for digital sovereignty in the data economy," in *New Digital Work II: Digital Sovereignty of Companies and Organizations*, U. Schmuntzsch, A. Shajek, and E. A. Hartmann, Eds., Cham, Switzerland: Springer, 2025, pp. 247–263. [CrossRef]

[29] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019. [CrossRef]

[30] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. Mcc Workshop Mob. Cloud Comput.*, Helsinki, Finland, 2012, pp. 13–16. [CrossRef]

[31] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016. [CrossRef]

[32] M. A. Hassan, M. Xiao, Q. Wei, and S. Chen, "Help your mobile applications with fog computing," in *Proc. 12th Annu. IEEE Int. Conf. Sensing, Commun.-Netw.–Work. (SECON Workshops)*, Seattle, WA, USA, Jun. 22–25, 2015, pp. 1–6. [CrossRef]

[33] A. Glikson, S. Nastic, and S. Dustdar, "Deviceless edge computing: Extending serverless computing to the edge of the network," in *Proc. 10th ACM Int. Syst. Storage Conf. (SYSTOR)*, Haifa, Israel, May 2017, Art. no. 28. [CrossRef]

[34] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. 22nd Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Xi'an, China, Apr. 8–12, 2017, pp. 615–629. [CrossRef]

ACCEPTED

[35] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, Jun. 5–8, 2017, pp. 328–339. [CrossRef]

Disclaimer/Publisher's Note: The views expressed in this article are those of the author(s) and do not necessarily reflect the views of the publisher or editors. The publisher and editors assume no responsibility for any injury or damage resulting from the use of information contained herein.

ACCEPTED