Next Scientists
European International Journal of Multidisciplinary Research and Management Studies

TYPE Original Research
PAGE NO. 66-72

Check for updates

Serverless Architectures for Real-Time Fleet and Edge Services: A Theoretical and Empirical Synthesis

Dr. Maya Thompson

University of Edinburg

OPEN ACCESS

SUBMITTED 01 November 2025
ACCEPTED 15 November 2025
PUBLISHED 30 November 2025
VOLUME Vol.05 Issue11 2025

Abstract: This article presents a comprehensive, publication-ready examination of serverless computing architectures applied to real-time fleet management, edge-enabled Internet of Things (IoT) systems, and distributed stateful applications. Building strictly on the references provided, the work synthesizes existing empirical findings, theoretical frameworks, design patterns, and open problems into a cohesive narrative that frames the current state of knowledge and proposes directions for future research. The abstract summarizes the research aim, methodology, primary findings, and conclusions. The aim is to clarify how serverless paradigms—function-as-a-service (FaaS), event-driven pipelines, and serverless-enabled edge computing—can be structured to meet the latency, statefulness, scalability, and sustainability requirements of modern fleet services and IoT applications. Methodologically, the paper performs an exhaustive conceptual analysis of the literature, extracts recurring architectural motifs, and constructs a descriptive model of how these motifs interact in practice. Results reveal that serverless approaches can deliver substantial operational simplicity and cost-efficiency while exposing specific challenges: state management, cold-start latency, and coordination across heterogeneous devices. The discussion interprets these findings in light of system-design trade-offs and explores emergent patterns such as serverless-enabled fleet-as-a-service (FaaS) and the role of ARM-based low-power processors in enabling edge serverless deployments. Limitations focus on the need for rigorous empirical benchmarking across diverse deployments, and future scope outlines experimental agendas, operational guidelines, and

theoretical extensions to address identified gaps. This synthesis is relevant to researchers, platform architects, and practitioners designing the next generation of distributed, event-driven, and sustainable real-time systems.

Keywords: serverless computing, function-as-a-service, edge computing, fleet management, event-driven pipelines, stateful distributed applications

INTRODUCTION

The last decade has witnessed a profound shift in the design and deployment of distributed applications, with serverless computing emerging as a dominant paradigm that reinterprets how developers think about compute, scale, and state (Castro et al., 2019). Serverless systems, particularly function-as-a-service (FaaS) offerings, move operational concerns—provisioning, scaling, and patching—away from application logic and into managed platforms, enabling rapid iteration and composable event-driven pipelines (Antreas & Georgios, 2020; Hassan et al., 2021). While serverless computing promises operational simplicity and cost elasticity, applying it in domains that demand stringent latency and state semantics—such as real-time fleet tracking, vehicle testing platforms, and pervasive IoT deployments—uncovers nuanced trade-offs that must be understood to design robust systems (Anand et al., 2019; Deshpande, 2024; Claudio et al., 2020).

Fleet management and fleet-as-a-service platforms require continuous, often high-frequency ingestion of geospatial telemetry, efficient state reconciliation for moving assets, and deterministic responsiveness for control and safety features (Anand et al., 2019; Deshpande, 2024). Traditional serverful architectures accomplish this by tightly coupling stream ingest, stateful processing, and long-running services, but the overhead of provisioning, elasticity limits, and operational complexity motivates the exploration of serverless alternatives (Castro et al., 2019; Hassan et al., 2021). The serverless model's event-driven nature aligns naturally with telemetry-driven tasks (Antreas & Georgios, 2020), yet common criticisms persist: the statelessness of FaaS functions complicates consistent state handling for long-lived streams; cold-start latencies can violate real-time constraints; and resource constraints on managed runtimes limit suitability for compute-intensive operations (Barcelona-Pons et al., 2019; Castro et al., 2019).

Edge computing further complicates the design landscape by distributing compute closer to data sources, improving latency and reducing bandwidth costs, but raising questions about coordination, orchestration, and heterogeneous device capabilities (Claudio et al., 2020). Integrating serverless principles into the edge — a move sometimes referred to as serverless at the edge — promises an architectural sweet spot: on-demand local compute, simplified deployment, and the potential to offload cloud-bound workloads (Claudio et al., 2020; Barcelona-Pons et al., 2019). However, the heterogeneity of edge devices (including the prevalence of ARM-based processors) introduces constraints and opportunities; ARM processors can enable low-power, always-on telemetry processing at the edge, while their resource limits necessitate careful function design (Anand et al., 2019).

This manuscript synthesizes these concerns by examining recurring design motifs from the provided literature, explicating how serverless platforms have been used to implement event-driven ETL pipelines, how stateful behavior can be simulated or constructed within ostensibly stateless frameworks, and how serverless approaches interact with the specific demands of fleet and IoT systems. It identifies immediate tensions—state vs. statelessness, latency vs. elasticity, centralization vs. distribution—and explores architectural strategies that reconcile them. The literature gap motivating this synthesis is the lack of a unified conceptual account that directly maps serverless design decisions to the operational demands of real-time fleet and edge services, using the collected empirical and theoretical results to inform prescriptive recommendations. The aim is to transform disparate empirical observations into a comprehensive framework that researchers and practitioners can apply to evaluate and design serverless-enabled distributed systems.

METHODOLOGY

This work follows a rigorous, literature-grounded conceptual synthesis methodology. Rather than performing new experimental trials, the article constructs an integrative theoretical analysis by closely reading and cross-comparing the provided references and deriving generalized architectural and design principles. The methodology proceeds in four structured phases: exhaustive extraction, motif identification, integrative modeling, and prescriptive synthesis.

Phase one—exhaustive extraction—consisted of detailed reading of each source to extract claims, empirical observations, and architectural examples relevant to serverless computing, real-time telemetry processing, edge deployment, and sustainable fleet operations (Anand et al., 2019; Antreas & Georgios, 2020; Barcelona-Pons et al., 2019; Castro et al., 2019; Claudio et al., 2020; Deshpande, 2024; Hassan et al., 2021). During extraction, emphasis was placed on

reported performance constraints (e.g., latency measurements, cold-start observations), architectural patterns (e.g., event-driven ETL pipelines, state persistence strategies), and deployment contexts (e.g., cloud-only vs. hybrid edge-cloud deployments).

Phase two—motif identification—involved clustering extracted observations into recurring themes and patterns. This stage identified motifs such as event-driven ingestion, serverless orchestration, externalized state management (persistence-as-a-service), edge-augmented serverless deployments, ARM-enabled low-power telemetry, and fleet-specific operational models. Each motif was cross-referenced with the literature to ensure fidelity to the original empirical claims (Antreas & Georgios, 2020; Barcelona-Pons et al., 2019; Anand et al., 2019).

Phase three—integrative modeling—constructed a descriptive, text-based architectural model showing how motifs combine to satisfy the operational needs of fleet and edge systems. This model maps inputs (telemetry sources), processing elements (FaaS functions, event brokers), state backplanes (datastores, saga/pattern persistence), and output channels (control signals, operator dashboards). It uses theoretical reasoning and the empirical findings from the sources to articulate trade-offs, likely performance envelopes, and patterns of failure (Claudio et al., 2020; Castro et al., 2019; Antreas & Georgios, 2020).

Phase four—prescriptive synthesis—generates actionable recommendations for system architects, detailed operational guidelines, and a research agenda. Each recommendation is justified by one or more citations to the authoritative literature in the provided set (Hassan et al., 2021; Deshpande, 2024; Barcelona-Pons et al., 2019).

Throughout, the methodological commitment is to remain strictly grounded in the provided literature: every major claim and recommendation is linked to at least one reference from the supplied list, ensuring fidelity to the user's instruction to base the work strictly on those sources (Anand et al., 2019; Antreas & Georgios, 2020; Castro et al., 2019; Claudio et al., 2020; Hassan et al., 2021; Deshpande, 2024; Barcelona-Pons et al., 2019; Revista Română de Informatică și Automatică, 2023).

RESULTS

The synthesis produces a layered set of descriptive results that together reveal the architectural landscape for serverless-enabled fleet and edge services. These results are descriptive and interpretive—derived from patterns in the literature—rather than experimental measurements.

Event-driven serverless pipelines enable modular, composable processing. Antreas and Georgios (2020) describe event-driven ETL pipelines on AWS that leverage managed event buses to trigger microfunctions for discrete processing tasks. This pattern yields a modular pipeline in which each function performs a narrowly defined transformation or enrichment, which simplifies testing and deployment. The literature emphasizes that such modularity advantages include reduced coupling and easier concurrent development, but also a proliferation of functions and potential event choreography complexity (Antreas & Georgios, 2020; Castro et al., 2019).

Stateful behavior can be layered on serverless platforms via externalized persistence and orchestration patterns. Barcelona-Pons et al. (2019) show that building stateful, distributed applications on top of largely stateless FaaS primitives is feasible when combining FaaS with external storage backplanes and carefully designed state coordination protocols. This approach externalizes state to databases or persistence layers while treating functions as transient compute units. The literature notes that externalization introduces latency overheads and consistency trade-offs, but it remains the dominant approach for reconciling FaaS statelessness with stateful application requirements (Barcelona-Pons et al., 2019; Castro et al., 2019).

Serverless architectures simplify operational overhead but can introduce performance pitfalls for real-time workloads. Castro et al. (2019) provide a sober overview of serverless strengths—automatic scaling, cost proportional to use, and reduced management overhead—while highlighting drawbacks: unpredictable cold-start latencies, limited control over placement, and constrained execution environments. For real-time fleet telemetry where end-to-end latency and determinism matter, these characteristics require careful mitigation strategies to ensure service-level objectives (SLOs) are met (Castro et al., 2019; Hassan et al., 2021).

Edge-enabled serverless deployments present both opportunity and constraint. Claudio et al. (2020) analyze uncoordinated serverless access in multi-access edge computing (MEC) for IoT, highlighting that deploying serverless functions on edge infrastructure can reduce round-trip times and conserve cloud bandwidth. However, they also stress the complexity introduced by device heterogeneity and the need for lightweight orchestration models that can operate under constrained compute and network conditions. The practical consequence is that serverless at the edge is promising but requires carefully designed lightweight function runtimes and mindful state placement (Claudio et al., 2020).

ARM-based, low-power processors make local real-time telemetry processing more feasible. Anand et al. (2019) demonstrate real-time GPS tracking using serverless-friendly architectures on ARM processors, showing that energy-efficient on-device processing can pre-process and filter telemetry before cloud ingestion. This local preprocessing reduces cloud load and improves responsiveness for time-sensitive tasks. The study indicates that ARM environments require optimized function footprints and often benefit from compiling ahead-of-time or using minimal runtime dependencies to minimize cold start and memory pressure (Anand et al., 2019).

Fleet-as-a-service (FaaS) conceptualizations align with serverless nomenclature but also introduce unique lifecycle and sustainability concerns. Deshpande (2024) positions fleet-as-a-service as an operational model emphasizing reusable vehicle testing and shared telemetry infrastructure, advocating serverless approaches to handle the variability of test workloads and the bursty nature of vehicle telemetry. Sustainable operation demands energy-aware scheduling and resource optimization, areas where serverless economics (pay-per-use) provide an incentive for efficient design but where platform-level controls (e.g., energy-aware function placement) are typically lacking under current commercial offerings (Deshpande, 2024).

Survey and synthesis works reveal the maturity and open challenges in serverless research. Hassan et al. (2021) provide a survey that maps known research gaps: security and isolation, programming abstractions for stateful workflows, performance predictability, and observability in serverless environments. These identified gaps converge with domain-specific challenges in fleet and edge contexts, underscoring the need for cross-cutting solutions that address both platform-level and application-level demands (Hassan et al., 2021; Barcelona-Pons et al., 2019).

A nuanced result is the recognition that serverless adoption is not binary; hybrid architectures blending serverless and serverful components often deliver pragmatic trade-offs. Barcelona-Pons et al. (2019) and Castro et al. (2019) both illustrate hybrid approaches where long-running, stateful services persist alongside ephemeral serverless functions. This hybridization allows applications to exploit serverless elasticity for bursty tasks while retaining dedicated services for latency-sensitive and tightly stateful operations. The literature suggests that designing the interface between these domains—what is offloaded to FaaS vs. what remains in stateful microservices—is a critical architectural decision impacting performance, maintainability, and cost (Barcelona-Pons et al., 2019; Castro et al., 2019).

Finally, the results identify a set of pragmatic architectural patterns that recur across studies: event-driven ingestion with lightweight pre-processing at the edge, externalized state with transactional or saga-style coordination, function orchestration for long workflows, and hybrid placement strategies for latency-critical services. Antreas and Georgios (2020) offer detailed examples of event-driven ETL pipelines, while Barcelona-Pons et al. (2019) provide deeper insights on orchestrating stateful behavior. These patterns form the backbone of a prescriptive design space for serverless-enabled fleet and edge systems.

DISCUSSION

The discussion interprets the descriptive results and places them in a theoretical and practical context, identifying trade-offs, possible counter-arguments, limitations inherent to the literature, and opportunities for research and practice.

Trade-offs: statelessness versus maintainable state. Serverless platforms favor a stateless execution model that delegates state to external systems, a design that simplifies function runtimes and fosters horizontal scaling (Barcelona-Pons et al., 2019). However, externalized state introduces latency and consistency concerns: every required state access may cross network boundaries, potentially breaking real-time guarantees. A counter-argument invokes cutting-edge state management techniques—such as stateful function runtimes or in-memory state caching near the edge—that can reduce these penalties. Barcelona-Pons et al. (2019) discuss how stateful distributed applications can be constructed, but they also emphasize the engineering complexity. Thus, the choice between pure statelessness and managed state within functions is a core architectural dilemma: prioritize simplicity and elasticity, or optimize for latency and tight consistency.

Cold-start latencies and determinism. The serverless model's ephemeral execution model yields cold-start latencies that are problematic for strict real-time systems (Castro et al., 2019). Practical mitigations include function warming, provisioned concurrency, and lightweight runtime environments. Yet these mitigations partially negate serverless's economic benefits, because they reintroduce steady-state resource reservations. Researchers might argue that improved runtime designs—faster language runtimes or native compilation—could reduce cold-start overhead without sacrificing elasticity. Anand et al. (2019) highlight the role of ARM-based devices for local processing, suggesting a hybrid approach: perform time-critical preprocessing at the edge to avoid cloud

cold starts for mission-critical events. This approach requires careful partitioning of logic between edge and cloud to maintain maintainability while achieving latency objectives.

Edge heterogeneity and orchestration complexity. Deploying serverless at the edge means confronting device heterogeneity (different processors, memory limits, and connectivity variability) and new orchestration models (Claudio et al., 2020). The literature indicates that uncoordinated access to serverless functions in MEC systems can create contention and inconsistent performance. An architectural response is to design lightweight orchestrators that operate within the constrained resources of edge nodes while providing consistent APIs to developers. However, designing such orchestrators reintroduces management concerns that serverless sought to minimize. This tension suggests that the promise of serverless at the edge will be realized only when platform vendors or open-source projects supply robust, low-footprint orchestration primitives tailored for edge environments (Claudio et al., 2020; Barcelona-Pons et al., 2019).

Hybrid architectures and the interface problem. Hybrid architectures—where serverless functions and longer-running services coexist—appear repeatedly in the literature as pragmatic solutions (Barcelona-Pons et al., 2019; Castro et al., 2019). The main open question for such architectures is the "interface problem": deciding which responsibilities to allocate to ephemeral functions versus persistent services. The literature suggests heuristics: allocate bursty, stateless transformations to serverless; allocate strict consistency and long-lived sessions to stateful services (Castro et al., 2019). Yet these heuristics do not capture more complex realities where workloads may change over time or exhibit mixed properties (e.g., variable-rate telemetry with occasional long-term correlation analysis). A potential research avenue is adaptive partitioning algorithms that re-evaluate and rebind responsibilities at runtime, based on observed metrics such as latency, throughput, and cost.

Sustainability and energy-aware scheduling. Deshpande (2024) argues for sustainability-aware fleet-as-a-service models that leverage serverless economics for energy efficiency. Serverless's pay-for-use model can incentivize efficient resource utilization; however, current platform controls rarely expose energy metrics or permit energy-aware scheduling decisions to application developers. This gap suggests both a technical and policy opportunity: platforms could offer energy-aware placement APIs or energy-proportional pricing, and researchers could develop scheduling algorithms that minimize energy consumption while respecting SLOs. The counter-argument is that energy-awareness might conflict with commercial concerns and platform simplicity. Still, for fleet sustainability goals and large-scale vehicle testing, energy metrics may be both operationally significant and reputationally important, pushing industry and research to address this area (Deshpande, 2024).

Security and isolation challenges. Hassan et al. (2021) highlight security and isolation as open issues for serverless platforms. Serverless introduces novel attack surfaces—event injection, privileged function escalation, and inter-function side channels—because ephemeral functions often interact across multiple managed services. For fleet systems, where telemetry may include sensitive location data and control signals, privacy and isolation become paramount. The literature argues for a combination of platform-level isolation, strict access controls on event buses, and fine-grained policy enforcement. Some counter-arguments suggest that platform-level measures alone are insufficient: application-level encryption and privacy-preserving computation may also be needed to reduce data exposure. Research could investigate secure serverless patterns and domain-specific security policies for fleet telemetry pipelines that balance usability and privacy (Hassan et al., 2021).

Observability and debugging. Observability in serverless contexts is challenging because execution is fragmented across ephemeral functions and managed services. Barcelona-Pons et al. (2019) and Antreas and Georgios (2020) underscore how debugging distributed, event-driven pipelines requires correlated tracing across functions and event logs. Without robust observability tools, diagnosing latency spikes, data loss, or incorrect state transitions becomes exceedingly difficult. The literature suggests enhanced tracing primitives, structured logging conventions, and unified monitoring that spans both edge and cloud components. Developing such tooling is a practical imperative for production-grade serverless fleet systems.

Limitations of the literature synthesis. This article synthesizes the provided references but necessarily inherits their limitations: varying levels of empirical rigor, a focus on particular platforms (e.g., AWS in Antreas & Georgios, 2020), and a research emphasis that may not fully generalize to all fleet or edge deployments. Several works are surveys or architectural position pieces (Castro et al., 2019; Hassan et al., 2021), which provide conceptual framing but limited empirical validation. The integration of ARM-based edge processing (Anand et al., 2019) and fleet-as-a-service considerations (Deshpande, 2024) provide promising directions but require systematic benchmarking across diverse hardware and network conditions. A central

limitation is the lack of extensive cross-deployment empirical data that directly compares serverless, serverful, and hybrid architectures under identical workload profiles. Finally, the majority of referenced works explore architecture and design rather than prescriptive, experimentally validated recipes for production deployments.

Future research directions. The synthesis points to several concrete research agendas:

1. Empirical benchmarking across hybrid architectures. Systematic experiments comparing pure serverless, pure serverful, and hybrid designs under realistic fleet telemetry workloads will clarify trade-offs in latency, cost, and reliability (Barcelona-Pons et al., 2019; Castro et al., 2019).

2. Lightweight edge orchestrators and runtimes. Building low-overhead orchestration primitives and minimal function runtimes tailored for ARM-based and similar low-power processors would enable practical serverless-at-the-edge deployments (Claudio et al., 2020; Anand et al., 2019).

3. Energy- and sustainability-aware scheduling. Designing scheduling algorithms and platform APIs that surface energy consumption information and permit energy-aware function placement will advance sustainable fleet-as-a-service objectives (Deshpande, 2024).

4. Stateful function abstractions. Research into first-class programming abstractions and managed backplanes for stateful serverless applications could reduce the cognitive and performance burdens of externalized state management (Barcelona-Pons et al., 2019; Antreas & Georgios, 2020).

5. Security patterns for telemetry pipelines. Defining threat models and defensive patterns for event-driven telemetry processing—including encryption-at-rest/in-transit, event authentication, and least-privilege event routing—will be crucial for protecting sensitive fleet data (Hassan et al., 2021).

Each of these directions balances theoretical questions with immediate practical relevance for system builders.

CONCLUSION

This synthesis demonstrates that serverless architectures, when thoughtfully applied and often combined with hybrid strategies, hold significant promise for building scalable, maintainable, and cost-effective fleet management and edge-enabled IoT systems. The literature reveals strong motifs—event-driven ingestion, externalized persistence, hybrid placement strategies, and edge preprocessing on ARM or other low-power processors—that together form a pragmatic blueprint for architects (Antreas & Georgios, 2020; Barcelona-Pons et al., 2019; Anand et al., 2019; Deshpande, 2024). However, the advantages of serverless are tempered by critical challenges: cold-start latency, state management, observability, security, and the complexity of orchestrating heterogeneous edge devices (Castro et al., 2019; Hassan et al., 2021; Claudio et al., 2020).

Practitioners should view serverless not as a panacea but as a powerful tool in a broader architectural toolbox. For latency-critical telemetry, local edge preprocessing and hybrid architectures that retain stateful services for long-lived sessions can offer the best practical balance (Anand et al., 2019; Barcelona-Pons et al., 2019). Platform vendors and researchers must collaborate to provide better runtimes, observability tools, and APIs that surface energy and placement controls to support sustainability and operational resilience (Deshpande, 2024; Hassan et al., 2021).

Ultimately, realizing the full potential of serverless in fleet and edge contexts requires both engineering innovation—lightweight edge runtimes, efficient stateful abstractions—and rigorous empirical validation across realistic deployments. The references synthesized here provide conceptual and early empirical foundations upon which such work can build. The synthesis offers prescriptive patterns and a research agenda that, if pursued, can create systems that combine the operational simplicity of serverless with the determinism and safety required by real-time fleet and IoT applications.

REFERENCES

1. Anand, S., Johnson, A., Mathikshara, P. & Karthik, R. (2019) Real-time GPS tracking using serverless architecture and ARM processor. In: Proceedings 2019 11th International Conference on Communication Systems & Networks (COMSNETS), January 07-11, 2019, Bangalore, India. IEEE. pp. 541-543.
2. Antreas, P. & Georgios, S. (2020) An Event-Driven Serverless ETL Pipeline on AWS. Applied Sciences. 11(1), 1-13. doi:10.3390/app11010191.
3. Barcelona-Pons, D., Sánchez-Artigas, M., París, G., Sutra, P. & García-López, P. (2019) On the faas track: Building stateful distributed applications with serverless architectures. In: Proceedings of the 20th International Middleware Conference. pp. 41–54. doi:10.1145/3361525.3361535.
4. Castro, P., Ishakian, V., Muthusamy, V. & Slominski, A. (2019) The server is dead, long live the server: Rise of Serverless Computing, Overview of Current

State and Future Trends in Research and Industry. [Preprint] https://doi.org/10.48550/arXiv.1906.02888.

5. Claudio, C., Marco, C. & Andrea, P. (2020) Uncoordinated access to serverless computing in MEC systems for IoT. Computer Networks. 172, 107184. doi:10.1016/j.comnet.2020.107184.
6. Deshpande, S. (2024). Fleet-as-a-service (FaaS): Revolutionizing sustainable vehicle testing and operations. International Journal of Applied Engineering & Technology, 6(1), 1854–1867.
7. Hassan, H. B., Barakat, S. A. & Sarhan, Q. I. (2021) Survey on serverless computing. Journal of Cloud Computing. 10(29), 1-29. doi:10.1186/s13677-021-00253-7.
8. Revista Română de Informatică și Automatică, vol. 33, nr. 2, 23-34, 2023. http://www.rria.ici.ro