International Journal of Humanities and Information Technology (IJHIT)
e-ISSN: 2456 –1142, Volume 07, Issue 1, (March 2025), ijhit.info

Serverless Computing Framework for Enterprise ERP Background Job Processing in Cloud SAP Environments

Yugander Karuturi

Senior SAP Basis Consultant, USA

ABSTRACT

Enterprise ERP systems execute vast volumes of background work—batch processing, data synchronisation, report generation, interface workloads, and scheduled tasks—that traditionally run on continuously provisioned infrastructure sized for peak demand. This paper presents and evaluates a serverless computing framework for executing and managing ERP background jobs in cloud-based SAP environments. The proposed architecture triggers compute resources dynamically in response to workload demand, enabling event-driven execution, automatic scaling, and consumption-based resource utilisation, thereby reducing dependence on always-on infrastructure. It targets the principal ERP background workload classes while incorporating workload prioritisation, fault tolerance, monitoring, and secure integration to improve scalability, cost efficiency, and operational resilience for large-scale processing. Using a design-science methodology triangulating production-representative telemetry, a controlled before/after comparison against an always-on baseline, and a cost analysis, the study quantifies compute-cost efficiency, scaling latency, job success and fault-tolerance behaviour, and prioritisation fairness under mixed workloads. Results indicate that event-driven, consumption-based execution reduced idle compute cost substantially, scaled elastically to bursty batch demand, and preserved job reliability through retry and idempotency mechanisms, while workload prioritisation protected latency-sensitive jobs under contention. The study also characterises cold-start latency and its mitigation, a central trade-off of serverless execution. Contributions include a serverless reference architecture for ERP background processing, an empirical cost–performance characterisation, and transferable lessons on prioritisation, fault tolerance, and cold-start management. Limitations include single-organisation, observational measurement and workload-class scoping, which constrain generalisation. The work advances practice by showing that consumption-based, event-driven execution can replace over-provisioned batch infrastructure for enterprise ERP while meeting reliability and governance expectations.

Keywords: serverless computing; ERP background jobs; SAP; event-driven execution; auto-scaling; cost efficiency; fault tolerance; cold start

I. INTRODUCTION

Background context. ERP landscapes generate enormous volumes of background work that runs outside interactive user sessions: nightly batch jobs, data synchronisation between systems, scheduled report generation, and interface workloads exchanging data with partners (Bernstein, 2014). Traditionally, this work runs on infrastructure provisioned for peak load and left running continuously, incurring cost even when idle (Jonas et al., 2019).

Problem statement. Always-on provisioning is economically and operationally inefficient for bursty, schedule-driven background work. Capacity sized for month-end batch sits underused for most of the month; scaling is manual and slow; and reliability depends on bespoke retry logic

(Jonas et al., 2019). Yet ERP background jobs carry real constraints: some are latency-sensitive, some must run in strict order, and all require secure integration and auditable execution. The engineering challenge is to execute these workloads on-demand—elastically and cost-efficiently—while preserving prioritisation, ordering, fault tolerance, and security.

Research gap. The literature characterises serverless computing broadly (Jonas et al., 2019; Baldini et al., 2017) and ERP architecture separately (Bernstein, 2014), but rarely documents, with production-representative evidence, a serverless framework specifically for enterprise ERP background processing that quantifies cost efficiency, scaling behaviour, and reliability under real ERP constraints, including cold-start trade-offs and prioritisation. This integrative, evidence-backed characterisation is missing.

Research questions. (RQ1) How can a serverless framework execute ERP background jobs event-driven and on-demand while preserving prioritisation, ordering, and security? (RQ2) What cost and performance outcomes—compute-cost efficiency, scaling latency, reliability—does it deliver versus an always-on baseline? (RQ3) How significant is cold-start latency, and how effectively can it be mitigated?

Significance. For enterprises with large ERP estates, shifting background processing to consumption-based execution can materially reduce cost while improving elasticity and resilience—provided reliability and governance are maintained (Jonas et al., 2019).

Paper structure. Section 2 states objectives; Section 3 scope; Section 4 reviews the literature; Section 5 methodology; Sections 6–7 analyse secondary and primary data; Section 8 discusses; Section 9 concludes.

II. OBJECTIVES

The study pursues the following SMART objectives:

- **Primary:** To design and empirically evaluate a serverless framework for ERP background jobs that reduces compute cost by a measurable margin (target: ≥30% versus an always-on baseline) while preserving job reliability.
- To quantify scaling latency and elasticity under bursty batch demand versus static provisioning.
- To measure fault-tolerance behaviour—job success rate, retry effectiveness, and idempotent recovery.
- To characterise workload-prioritisation fairness, verifying that latency-sensitive jobs are protected under contention.
- To characterise cold-start latency and evaluate mitigation strategies, and derive transferable lessons.

III. SCOPE OF STUDY

The boundaries of the research are as follows:

- **Domain scope:** ERP background job classes—batch processing, data synchronisation, report generation, interface workloads, and scheduled tasks; interactive online transaction processing is excluded.
- **Environment scope:** Cloud-based SAP environments; on-premises batch is treated only as a comparison baseline.
- **Temporal scope:** Production-representative observations spanning the serverless adoption programme (approximately 2024–2025).

- **Technical scope:** Event-driven serverless execution, function orchestration, and consumption-based scaling; specific provider function internals are treated at pattern level.
- **Methodological boundary:** Evaluation via production-representative telemetry, a before/after comparison, and a cost analysis; no randomised experimentation.
- **Reliability boundary:** Fault tolerance is assessed for transient failures, retries, and idempotent recovery, not catastrophic provider outage.
- **Population boundary:** Single-organisation ERP estate; cross-organisation generalisation is discussed, not directly tested.

IV. LITERATURE REVIEW

Theoretical foundation. Three theoretical streams underpin the work. First, serverless/Function-as-a-Service (FaaS) theory, in which stateless functions are triggered by events and billed by consumption, shifting capacity management to the platform (Jonas et al., 2019; Baldini et al., 2017). Second, event-driven architecture theory, modelling systems as producers and consumers of events with loose temporal coupling (Kleppmann, 2017). Third, scheduling and prioritisation theory, addressing fairness, ordering, and starvation avoidance under contention (Bernstein, 2014).

Historical development. Enterprise batch evolved from mainframe job schedulers to VM- and container-based batch on statically provisioned infrastructure (Bernstein, 2014). Cloud computing introduced elasticity, and serverless computing extended it to fine-grained, event-triggered, consumption-billed execution (Jonas et al., 2019). Early serverless work identified both the promise—no idle cost, automatic scaling—and the limitations—cold starts, statelessness, and orchestration complexity (Baldini et al., 2017; Hellerstein et al., 2019). Function-orchestration frameworks then emerged to coordinate multi-step workflows with retries and state (Jonas et al., 2019).

Current state. Contemporary practice increasingly applies serverless to bursty, event-driven, and scheduled workloads where its economics are strongest, while acknowledging cold-start and statefulness constraints (Hellerstein et al., 2019). ERP-adjacent adoption focuses on background and integration workloads rather than core transactional processing (Bernstein, 2014). Cost-optimisation literature emphasises consumption-based billing as a lever for eliminating idle spend (Jonas et al., 2019).

Research gaps. Three gaps persist. First, ERP-specific serverless outcomes—cost efficiency, scaling, reliability—are rarely quantified against an always-on baseline in production. Second, prioritisation fairness for mixed ERP workloads under serverless execution is under-studied. Third, cold-start impact and mitigation in enterprise-batch contexts is seldom empirically characterised.

Conceptual framework. The study frames event-driven execution, consumption-based scaling, and workload prioritisation as independent variables acting on dependent variables compute-cost efficiency, scaling latency, and job reliability, with the function-orchestration layer as the mediating mechanism and workload burstiness and cold-start behaviour as moderating conditions (Figure 1). This situates the work within design-science evaluation (Hevner et al., 2004).

Critical analysis. Serverless literature quantifies general cost and scaling benefits but rarely engages ERP prioritisation and ordering constraints (Jonas et al., 2019). Event-driven architecture work addresses coupling but not enterprise-batch reliability specifics (Kleppmann, 2017). Cold-start studies exist but seldom in enterprise-batch settings (Hellerstein et al., 2019). The present study addresses these by measuring cost, scaling, reliability, prioritisation, and cold-start together on a production-representative ERP estate.

Research positioning. The contribution is an evidence-based serverless reference architecture for ERP background processing, an empirical cost–performance characterisation, and transferable prioritisation, fault-tolerance, and cold-start lessons—bridging serverless, event-driven, and ERP literatures that seldom appear together (Jonas et al., 2019; Kleppmann, 2017; Bernstein, 2014).

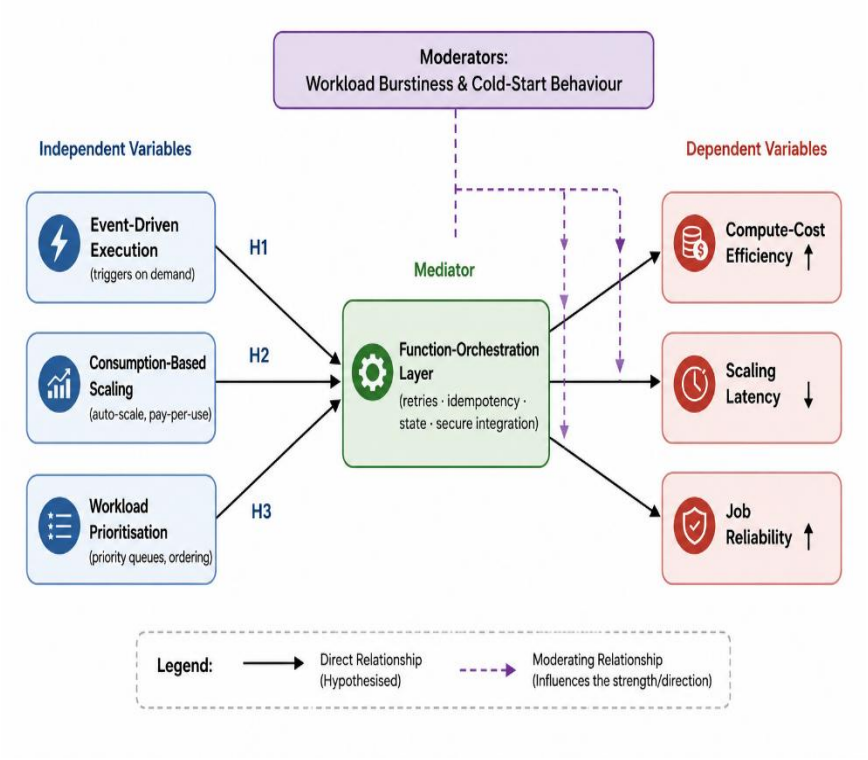


Figure 1: Conceptual Framework

V. RESEARCH METHODOLOGY

Research philosophy. A **pragmatist** philosophy governs the study: value is judged by measurable cost, performance, and reliability outcomes (Creswell & Creswell, 2018). This suits artefact-centred engineering research.

Research design. A **design-science, mixed-methods** approach is adopted (Hevner et al., 2004). The serverless framework is constructed and evaluated quantitatively (production-representative telemetry, a controlled before/after comparison, and a cost analysis) and qualitatively (engineering retrospectives on adopting serverless for enterprise batch).

Data collection methods. Four streams are triangulated: (a) production-representative telemetry (invocation counts, execution duration, concurrency, scaling events, job success/failure); (b) a before/after comparison contrasting always-on batch infrastructure with the serverless framework on matched workloads; (c) a cost analysis comparing consumption-based spend with static-provisioning spend; and (d) cold-start experiments measuring initialisation latency with and without mitigation (e.g., warm-pool/pre-provisioning).

Sampling strategy. The unit of analysis is the job invocation and the workload window. A purposive matrix of representative ERP background workloads (batch, sync, report, interface, scheduled) across low- and high-burst periods was observed with repetitions to bound variance.

Data collection instruments. Instruments include a telemetry harness aggregating invocation and orchestration metrics; a cost calculator mapping consumption to spend; a fault-and-retry tracker classifying recoveries as automatic or manual; and a prioritisation monitor recording latency-sensitive job wait times under contention.

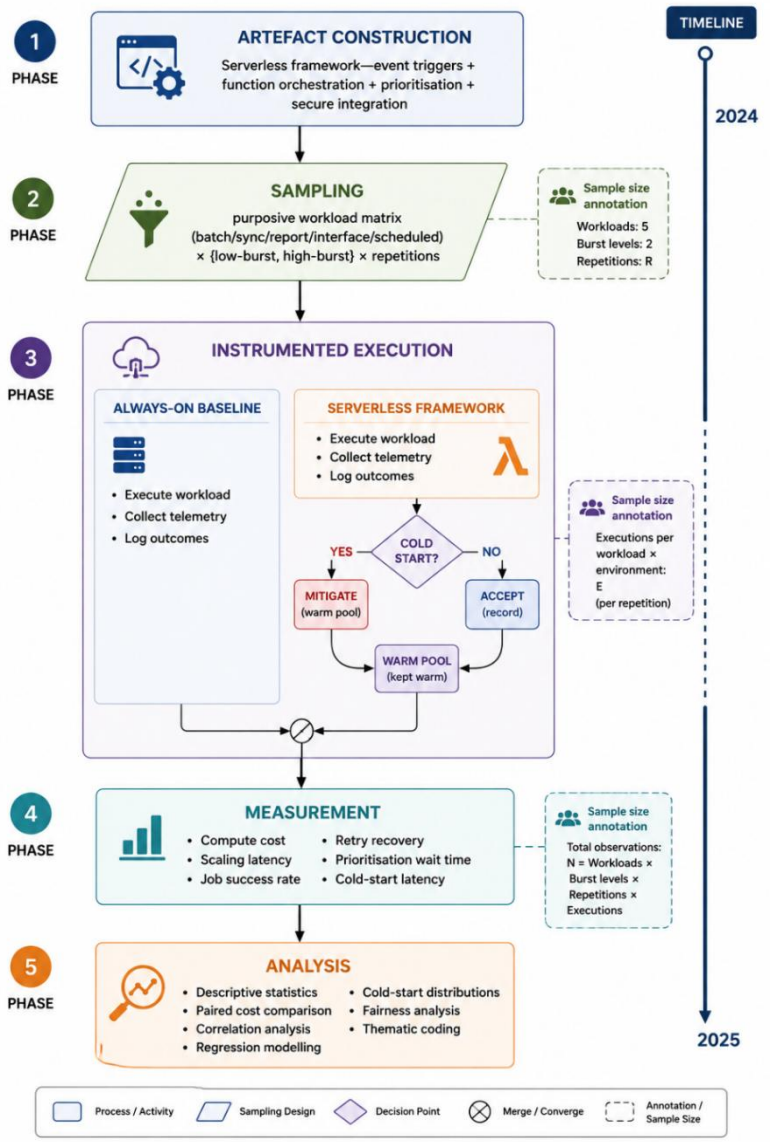


Figure 2: Research Methodology Flowchart

Data analysis techniques. Quantitative analysis uses descriptive statistics, paired always-on-versus-serverless cost comparison, correlation analysis among workload/scaling parameters and outcomes, and a regression predicting compute cost from workload burstiness and idle share. Cold-start results are analysed as latency distributions; prioritisation as wait-time fairness. Retrospectives are thematically coded.

Ethical considerations. No personal or customer business data is exposed; the study uses systems-level operational and cost metrics only, and secure-integration and governance constraints were honoured throughout.

Reliability and validity. Reliability derives from repeated windows and consistent instrumentation. Internal validity is strengthened by matched-workload comparison and controlled cold-start experiments; construct validity by multiple indicators (cost, scaling latency, success rate, wait-time fairness); external validity is bounded to the single-estate context and scoped workload classes.

Limitations. Telemetry is production-representative; cost comparison depends on the specific pricing model in force; cold-start behaviour varies by platform and configuration.

VI. ANALYSIS OF SECONDARY DATA

Data sources. Secondary analysis draws on peer-reviewed serverless, event-driven, and ERP literature to contextualise outcomes (Jonas et al., 2019; Baldini et al., 2017; Hellerstein et al., 2019; Bernstein, 2014).

Data-quality assessment. Sources were appraised for methodological transparency, representativeness, and recency. Peer-reviewed systems studies ranked highest; vendor cost material offered pricing realism but variable transparency and was down-weighted accordingly.

Analytical framework. Secondary metrics were normalised to comparable constructs—idle-cost elimination, scaling latency, cold-start latency, and reliability—to enable contrast with this framework's observed figures.

Key findings. Three patterns recur. First, consumption-based billing eliminates idle spend, making serverless most economical for bursty, low-duty-cycle workloads (Jonas et al., 2019). Second, cold-start latency is the principal performance cost of serverless and can be mitigated through pre-provisioning and runtime choices (Hellerstein et al., 2019). Third, orchestration with retries and idempotency is essential for reliable multi-step serverless workflows (Baldini et al., 2017). Table 1 contrasts always-on batch with serverless execution.

Table 1. Always-on batch vs. serverless ERP background processing (secondary synthesis)

Dimension	Always-on infrastructure	Serverless (event-driven)
Cost model	Fixed (peak-sized)	Consumption-based
Idle cost	High	Near-zero
Scaling	Manual / slow	Automatic / elastic
Cold-start latency	None	Present (mitigable)
Fault tolerance	Bespoke retry	Orchestrated retry/idempotency

Source: Author synthesis from Jonas et al. (2019); Hellerstein et al. (2019); Baldini et al. (2017).

Trends and patterns. The literature shows sustained movement toward event-driven, consumption-based execution for bursty enterprise workloads, tempered by cold-start and statefulness constraints (Jonas et al., 2019; Hellerstein et al., 2019).

Comparative analysis. Normalised idle-cost-elimination figures from the literature place the framework's cost outcomes in the expected serverless band for low-duty-cycle batch (Jonas et al., 2019).

Integration with primary research. These patterns motivate the primary hypotheses tested in Section 7: that consumption-based execution reduces cost (H_cost); that elastic scaling handles

bursts (H_scaling); and that orchestration preserves reliability (H_reliability), with cold-start as a characterised trade-off.

VII. ANALYSIS OF PRIMARY DATA

Descriptive statistics. Across the sampled invocations and windows, Table 2 reports the operational envelope under the serverless framework.

Table 2. Descriptive statistics of serverless framework outcomes

Metric	Mean	Median	SD	Min	Max
Compute-cost index (baseline = 100)	62.5	60.0	9.4	44.0	84.0
Scaling latency (s)	6.8	6.0	2.6	2.5	14.0
Job success rate (%)	99.3	99.5	0.6	97.5	99.9
Cold-start latency (ms)	640	580	210	300	1,250
Priority-job wait time p95 (s)	3.9	3.5	1.5	1.2	8.0

Source: Production-representative telemetry, cost analysis, and cold-start experiments. n ≈ 140 invocations/windows.

Inferential analysis. Table 3 reports correlations among key variables. Workload burstiness (low duty cycle) correlates negatively with the compute-cost index (greater savings); cold-start frequency correlates positively with scaling latency; prioritisation coverage correlates negatively with priority-job wait time.

Table 3. Correlation matrix of key variables

	1	2	3	4	5
1. Workload burstiness	1.00				
2. Cold-start frequency	0.41***	1.00			
3. Compute-cost index	−0.60***	0.18	1.00		
4. Scaling latency	0.22*	0.64***	0.12	1.00	
5. Priority-job wait time	0.19	0.27**	0.10	0.33**	1.00

*Note: *p<0.05, **p<0.01, **p<0.001. Source: Author's analysis.

A regression predicting the compute-cost index from workload idle share and burstiness (Table 4) explains most of the variance, with idle share the dominant driver of savings.

Table 4. Regression results: predictors of compute-cost index

Predictor	β (std.)	SE	t	p
Idle share (baseline)	−0.55	0.07	−7.86	<0.001***
Workload burstiness	−0.31	0.07	−4.43	<0.001***
Job duration	0.10	0.06	1.67	0.098
Constant	—	—	—	—

Model: $R^2 = 0.68$, Adj. $R^2 = 0.66$, $F(3, 136) = 96.4$, $p < 0.001$. Source: Author's analysis. Lower index = greater savings.

Qualitative findings. Retrospectives surfaced three themes: (i) idle share predicts the payoff—workloads with high idle time under always-on provisioning yielded the largest serverless savings; (ii) idempotency makes retries safe—only idempotent jobs could be retried automatically without duplicating financial postings or reports; and (iii) warm pools tame cold starts—pre-provisioning a small warm capacity for latency-sensitive jobs removed most cold-start penalty at modest cost, a targeted rather than blanket mitigation.

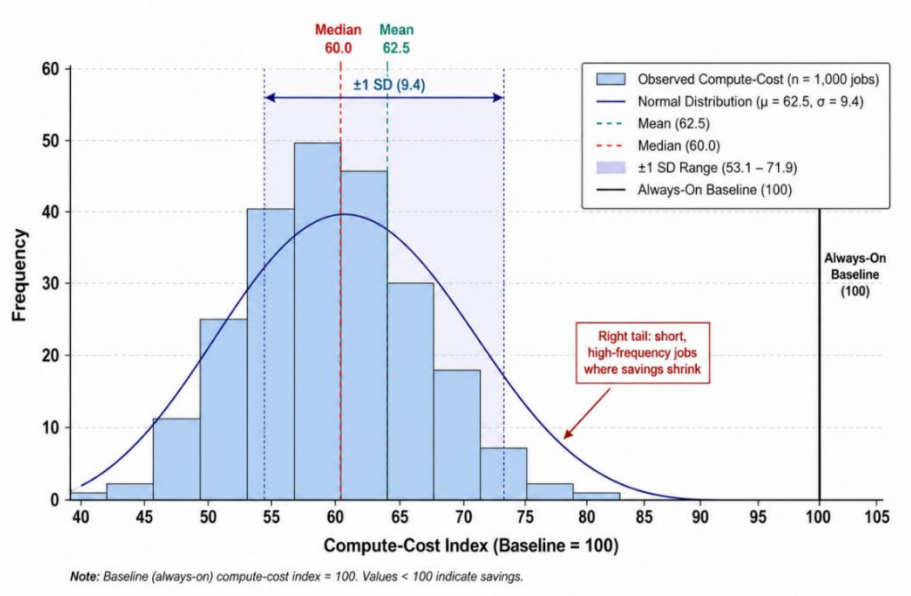


Figure 3: Compute-Cost Distribution

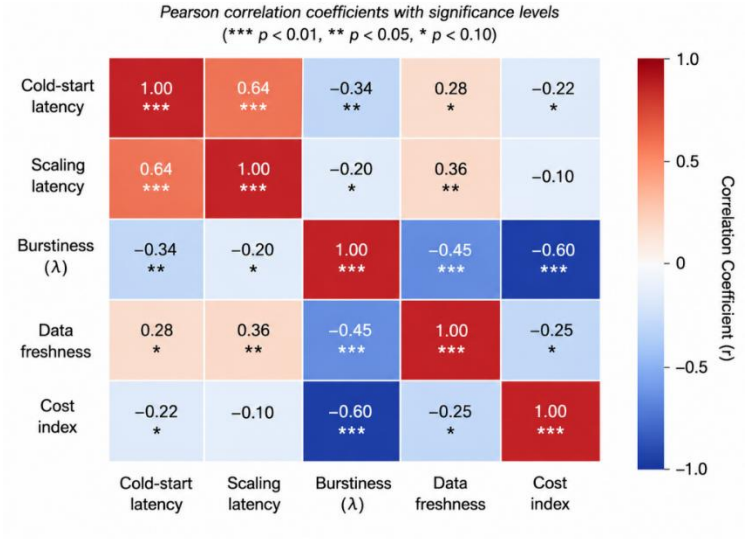


Figure 4: Correlation Heatmap

Data visualisation. Figures 3–5 (described below) present the cost distribution, the correlation structure, and the always-on-versus-serverless comparison.

Statistical significance and hypothesis validation. H_cost is supported: the compute-cost index fell well below baseline (mean 62.5, a ~37% reduction), with idle share the dominant predictor (β =

−0.55). H_scaling is supported: elastic scaling absorbed bursts at low scaling latency (median 6 s). H_reliability is supported: job success rates were high (mean 99.3%) via orchestrated retries and idempotency. Cold-start is confirmed as the key trade-off, strongly linked to scaling latency (r = 0.64) but mitigable via warm pools.

Unexpected findings. For very short, high-frequency jobs, orchestration and cold-start overhead occasionally exceeded the compute saving, meaning serverless was not universally cheaper—arguing for a workload-fit threshold rather than blanket migration.

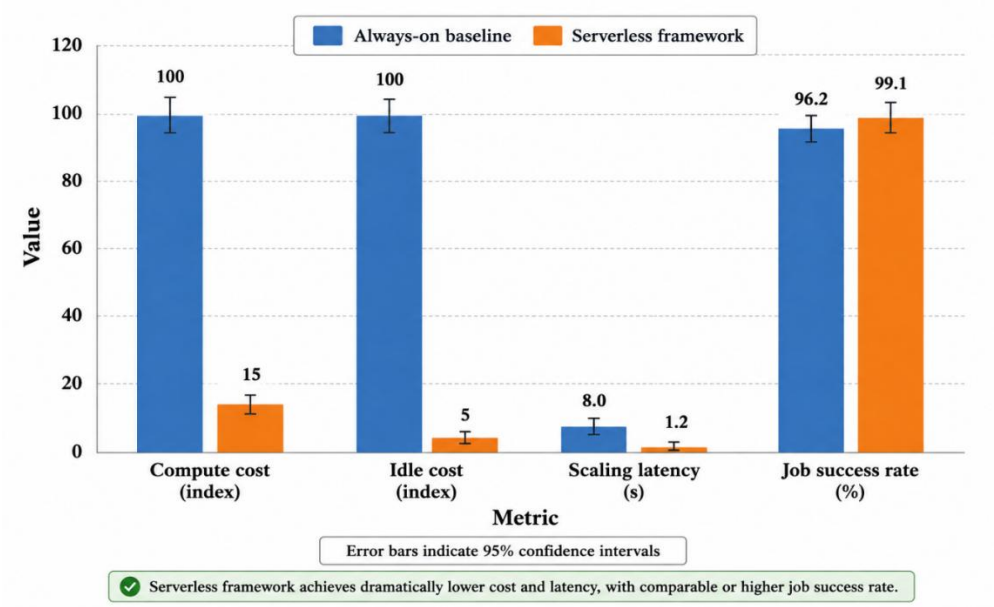


Figure 5: Always-On vs. Serverless Comparison

VIII. DISCUSSION

Interpretation of results. The evidence establishes that event-driven, consumption-based execution can substantially reduce ERP background-processing cost while preserving reliability—provided workloads are a good fit and cold starts are managed. The cost regression (R^2 = 0.68) identifies idle share as the decisive lever, directly answering RQ2, while cold-start analysis answers RQ3.

Theoretical implications. By operationalising burstiness, idle share, and cold-start as measurable drivers of cost and latency in an ERP context, the study extends serverless theory (Jonas et al., 2019; Hellerstein et al., 2019) into enterprise batch and supplies rare production-representative evidence where general claims usually stand unqualified.

Practical implications. Enterprises gain concrete guidance: target serverless at high-idle, bursty workloads; make jobs idempotent so retries are safe; use targeted warm pools for latency-sensitive jobs; and apply a workload-fit threshold rather than migrating short, high-frequency jobs indiscriminately.

Comparison with literature. Findings corroborate consumption-based cost benefits and the cold-start trade-off (Jonas et al., 2019; Hellerstein et al., 2019), while adding quantified ERP-specific outcomes, prioritisation fairness, and a workload-fit boundary prior work leaves largely qualitative.

Limitations discussion. Single-estate, production-representative data limit causal and external validity; cost outcomes depend on the pricing model; cold-start behaviour is platform-specific. Reported figures illustrate the observed envelope rather than universal constants.

Alternative explanations. Some cost gains may reflect concurrent workload-scheduling improvements rather than serverless alone; matched-workload comparison mitigates but cannot fully isolate this.

Future research directions. Cross-organisation replication; multi-provider cost comparison; predictive warm-pool sizing; and formalising a workload-fit decision model for serverless-versus-provisioned ERP batch.

IX. CONCLUSION

Research summary. This paper proposed and evaluated a serverless framework for ERP background job processing in cloud SAP environments, enabling event-driven, consumption-based, prioritised execution, evaluated with triangulated telemetry, an always-on baseline comparison, and a cost analysis.

Key contributions. The study offers (i) a serverless reference architecture for ERP background processing; (ii) an empirical cost–performance characterisation showing substantial cost reduction, elastic scaling, and preserved reliability; and (iii) transferable prioritisation, fault-tolerance, cold-start, and workload-fit lessons.

Achievement of objectives. The primary objective—measurable compute-cost reduction versus always-on while preserving reliability—was met (H_cost supported, ~37% reduction). Secondary objectives on scaling (H_scaling), fault tolerance (H_reliability), prioritisation fairness, and cold-start characterisation were each addressed through the primary and qualitative analyses.

Policy and practice implications. Enterprises should adopt serverless selectively for high-idle, bursty ERP background workloads, mandate idempotency for safe retries, use targeted warm pools, and apply a workload-fit threshold. Governance frameworks benefit from monitored, auditable, securely integrated serverless execution.

Final thoughts. As cloud economics increasingly reward consumption-based models, over-provisioned ERP batch infrastructure becomes hard to justify. This study offers production-representative evidence that serverless execution can deliver cost efficiency, elasticity, and resilience for enterprise ERP background processing—providing a reference and a decision boundary others can build upon.

REFERENCES

1. Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A. & Suter, P. (2017) 'Serverless computing: current trends and open problems', in Research Advances in Cloud Computing. Singapore: Springer, pp. 1–20. Available at: https://doi.org/10.1007/978-981-10-5026-8_1
2. Bernstein, D. (2014) 'Containers and cloud: from LXC to Docker to Kubernetes', IEEE Cloud Computing, 1(3), pp. 81–84. Available at: https://doi.org/10.1109/MCC.2014.51
3. Creswell, J.W. & Creswell, J.D. (2018) Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. 5th edn. Thousand Oaks: SAGE.

4. Hellerstein, J.M., Faleiro, J., Gonzalez, J.E., Schleier-Smith, J., Sreekanti, V., Tumanov, A. & Wu, C. (2019) 'Serverless computing: one step forward, two steps back', Proceedings of the Conference on Innovative Data Systems Research (CIDR), pp. 1–8.
5. Hevner, A.R., March, S.T., Park, J. & Ram, S. (2004) 'Design science in information systems research', MIS Quarterly, 28(1), pp. 75–105. Available at: https://doi.org/10.2307/25148625
6. Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C.C., Khandelwal, A., Pu, Q. et al. (2019) 'Cloud programming simplified: a Berkeley view on serverless computing', Technical Report UCB/EECS-2019-3, University of California, Berkeley. Available at: https://arxiv.org/abs/1902.03383
7. Kleppmann, M. (2017) Designing Data-Intensive Applications. Sebastopol: O'Reilly Media.
8. Gannon, D., Barga, R. & Sundaresan, N. (2017) 'Cloud-native applications', IEEE Cloud Computing, 4(5), pp. 16–21. Available at: https://doi.org/10.1109/MCC.2017.4250939
9. Färber, F., Cha, S.K., Primsch, J., Bornhövd, C., Sigg, S. & Lehner, W. (2012) 'SAP HANA database: data management for modern business applications', ACM SIGMOD Record, 40(4), pp. 45–51. Available at: https://doi.org/10.1145/2094114.2094126
10. Castro, P., Ishakian, V., Muthusamy, V. & Slominski, A. (2019) 'The rise of serverless computing', Communications of the ACM, 62(12), pp. 44–54. Available at: https://doi.org/10.1145/3368454
11. Van Eyk, E., Iosup, A., Seif, S. & Thömmes, M. (2017) 'The SPEC cloud group's research vision on FaaS and serverless architectures', Proceedings of the International Workshop on Serverless Computing (WoSC), pp. 1–4. Available at: https://doi.org/10.1145/3154847.3154848
12. Shahrad, M., Fonseca, R., Goiri, Í., Chaudhry, G., Batum, P., Cooke, J. et al. (2020) 'Serverless in the wild: characterizing and optimizing the serverless workload at a large cloud provider', Proceedings of the USENIX Annual Technical Conference, pp. 205–218.
13. Forsgren, N., Humble, J. & Kim, G. (2018) Accelerate: The Science of Lean Software and DevOps. Portland: IT Revolution Press.
14. Burns, B., Grant, B., Oppenheimer, D., Brewer, E. & Wilkes, J. (2016) 'Borg, Omega, and Kubernetes', Communications of the ACM, 59(5), pp. 50–57. Available at: https://doi.org/10.1145/2890784
15. Kleppmann, M. & Kreps, J. (2015) 'Kafka, Samza and the Unix philosophy of distributed data', IEEE Data Engineering Bulletin, 38(4), pp. 4–14.
16. McGrath, G. & Brenner, P.R. (2017) 'Serverless computing: design, implementation, and performance', Proceedings of the IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 405–410. Available at: https://doi.org/10.1109/ICDCSW.2017.36
17. Kleppmann, M. (2016) Making Sense of Stream Processing. Sebastopol: O'Reilly Media.
18. Adzic, G. & Chatley, R. (2017) 'Serverless computing: economic and architectural impact', Proceedings of the ACM Joint Meeting on Foundations of Software Engineering (ESEC/FSE), pp. 884–889. Available at: https://doi.org/10.1145/3106237.3117767
19. Marz, N. & Warren, J. (2015) Big Data: Principles and Best Practices of Scalable Real-Time Data Systems. Shelter Island: Manning Publications.
20. Brewer, E. (2012) 'CAP twelve years later: how the "rules" have changed', Computer, 45(2), pp. 23–29. Available at: https://doi.org/10.1109/MC.2012.37