Cold Start Latency of Quantized Small Language Models on Serverless CPU Infrastructure

Ming Hu
Google
New York, NY, USA
huming@google.com

Khushan Adatiya
Google
Mountain View, CA, USA
khushan@google.com

Abstract—Serverless platforms offer compelling economics for low-traffic inference of small language models (SLMs), but cold start latency remains a critical barrier. We present a systematic empirical study of cold start performance for quantized SLMs served via llama.cpp on Google Cloud Run's CPU-only infrastructure. We benchmark five models (270M–3.8B parameters) across two memory tiers (4 GiB and 8 GiB), with a quantization sweep spanning five GGUF formats (Q2_K through Q8_0) for LLaMA 3.2 1B and three-point validation on Gemma 3 270M and Qwen3 0.6B. Our results reveal that model loading dominates cold start time (55–70%), the 8 GiB tier provides a hidden 2× vCPU advantage that nearly halves warm inference time, and Q4_K_M quantization achieves a Pareto-optimal tradeoff across latency, throughput, and model quality (confirmed via WikiText-2 perplexity). A three-predictor latency model ($R^2 = 0.97$) and break-even cost analysis provide practitioners a quantitative framework for SLM deployment decisions.

Index Terms—small language models, serverless computing, cold start, quantization, llama.cpp, Cloud Run, GGUF, inference optimization

I. Introduction

The rapid proliferation of large language models has driven interest in smaller, more efficient variants, known as small language models (SLMs), that can be deployed on resource-constrained infrastructure [1], [2]. Models with fewer than 4B parameters now achieve competitive performance on many tasks while fitting within the memory budgets of serverless platforms [3], [4]. While SLMs can run on edge devices, many production workloads require server-side deployment for multi-tenant APIs, privacy-regulated backends, and low-traffic services where persistent GPU infrastructure is uneconomical. Serverless platforms offer an attractive deployment model for these use cases [7], [8], but impose cold start latency that can dominate total request time [11], [12]. While cold starts have been studied for traditional web services [9], [10], the intersection of SLM inference and serverless cold starts remains unexplored. Post-training quantization via GGUF [13]–[15] reduces model size, but its impact on cold start latency has not been systematically characterized.

The appeal of self-hosted SLM inference is threefold: (a) data remains on-premise, satisfying privacy constraints in healthcare and finance; (b) per-token costs can be orders of magnitude lower than API-based inference for sustained workloads; and (c) models can be customized (fine-tuned, quantized) for domain-specific tasks without vendor lock-in. However, realizing these benefits on serverless infrastructure requires understanding the cold start penalty and its interaction with model size and quantization.

Our contributions are:

1) Cold start latency decomposition for five SLMs on Cloud Run, showing model loading accounts for 55–70% of end-to-end time.
2) A five-point quantization sweep for LLaMA 3.2 1B with three-point validation across Gemma 3 270M and Qwen3 0.6B, plus perplexity evaluation confirming Q4_K_M is Pareto-optimal across latency, throughput, and quality.
3) Quantification of Cloud Run's hidden vCPU scaling effect, where the 8 GiB tier's doubled vCPU nearly halves warm inference time.
4) Cost analysis with break-even thresholds (58–537 req/hr) and a predictive latency model ($R^2 = 0.97$).

II. Related Work

A. Serverless Cold Start Latency

Cold start latency on serverless platforms has been extensively studied. Wang et al. [9] measured cold starts across major providers, finding latencies of hundreds of milliseconds to seconds depending on runtime and memory configuration. Silva et al. [11] benchmarked Cloud Run specifically, reporting cold starts of 1–10 s for web applications, an order of magnitude below our observed SLM cold starts (5–44 s). Shahrad et al. [8] analyzed production traces at Azure, finding that cold starts affect ∼5% of invocations but account for disproportionate tail latency. Bermbach et al. [12] discovered a linear relationship between allocated memory and CPU in AWS Lambda; we observe a similar coupling in Cloud Run's memory-to-vCPU mapping.

Various techniques have been proposed to reduce cold start overhead. Oakes et al. [20] introduced SOCK, using lean containers and import caching to reduce Python function cold starts by up to 3.6×. Snapshot-based approaches such as Firecracker microVMs [21] can restore function state in sub-second time. Cloud Run's container image streaming and caching mechanisms reduce boot time to

∼0.2 s in our experiments, but model loading, a workload not targeted by these optimizations, remains the dominant bottleneck.

B. Post-Training Quantization

Post-training quantization reduces model size and inference cost with minimal retraining. Dettmers et al. [13] demonstrated 8-bit inference for transformers at scale, while Frantar et al. [14] achieved accurate 4-bit quantization via GPTQ. Lin et al. [16] proposed activation-aware weight quantization (AWQ), and Dettmers and Zettlemoyer [22] analyzed scaling laws for k-bit inference, finding that 4-bit precision offers an optimal quality–efficiency tradeoff across model scales. The llama.cpp project [15] implements these techniques in the GGUF format with k-quant variants (Q2_K through Q8_0) optimized for CPU SIMD execution. Our perplexity analysis confirms the 4-bit optimality finding at the sub-1B parameter scale on commodity CPU infrastructure.

On the SLM side, the Gemma [4], LLaMA 3.2 [5], and Qwen3 [6] families provide capable models from 270M to 3.8B parameters suitable for resource-constrained deployment.

C. Serverless Machine Learning

Ishakian et al. [17] evaluated deep learning inference on AWS Lambda, finding cold starts of 10–30 s for CNN models with model loading as the primary bottleneck, consistent with our findings for language models. Fu et al. [18] developed ServerlessLLM with locality-aware checkpoint loading, achieving near-zero cold starts for GPU-based LLM serving. However, their approach targets GPU clusters with NVMe storage, whereas our work characterizes CPU-only commodity serverless infrastructure where GPU acceleration is unavailable or cost-prohibitive.

None of the above works provide a systematic analysis combining quantization-level sweeps, cold start decomposition, quality evaluation, and cost modeling for SLMs on CPU-only serverless platforms, the gap our study addresses.

III. Experimental Methodology

A. Platform and Runtime

We select Google Cloud Run as it is the only major serverless container platform offering true scale-to-zero with sub-second infrastructure boot, full Docker support, and configurable memory/vCPU tiers, capabilities required to isolate model-loading latency from platform overhead. All experiments run on Cloud Run gen-2 (us-central1), which couples memory with vCPU: the 4 GiB tier provides 1 vCPU, the 8 GiB tier provides 2 vCPU. Services are deployed with min-instances=0, max-instances=1, and 300 s timeout. We use llama.cpp (via llama-cpp-python) as the inference runtime because it is the dominant CPU inference engine for quantized models, supporting the GGUF format's k-quant variants (Q2_K through Q8_0) with hand-tuned SIMD kernels for x86. Alternative runtimes (ONNX Runtime, vLLM) either lack equivalent quantization granularity or target GPU deployment. The container image (∼5.8 GB) includes all GGUF files, served via FastAPI ($n_{ctx} = 2048$). We set a conservative context window to minimize KV cache memory on the 4 GiB tier; this does not affect model loading, which dominates cold start. We validate this multi-model design against a single-model container in Section VII.

TABLE I
Models Under Test

Model	Role	Params	Q4_K_M	Q8_0
Gemma 3 270M	Both	0.27B	241 MB	278 MB
Qwen3 0.6B	Both	0.6B	378 MB	610 MB
LLaMA 3.2 1B	Both	1.24B	770 MB	1.2 GB
Gemma 2 2B	Cross	2.61B	1.6 GB	—
Phi-3.5 Mini	Cross	3.82B	2.2 GB	—

Cross=cross-model (Q4_K_M only); Both=cross-model and quantization sweep.

B. Models Under Test

Table I summarizes the models, selected to span the 0.27–3.8B parameter range across four distinct architecture families (Gemma, Qwen, LLaMA, Phi) with available GGUF quantizations. Five models are used in the cross-model comparison (Q4_K_M); LLaMA 1B is tested at five quantization levels (Q2_K through Q8_0), with Gemma 270M and Qwen3 0.6B at three points for validation.

C. Measurement Protocol

Each configuration is measured 30 times (60 for the most commonly compared variants). Cold starts are forced by deploying a new Cloud Run revision, ensuring no cached container state; warm runs reuse the active container with a 1 s cooldown between requests. The prompt (∼16 tokens, max_tokens=50) is held constant across all models and tiers. End-to-end latency decomposes as $t_{e2e} = t_{boot} + t_{load} + t_{infer}$, where t_{boot} is infrastructure boot (container start), t_{load} is model weight loading, and t_{infer} is token generation. All pairwise comparisons use Welch's t-test ($\alpha = 0.05$) with Cohen's d effect sizes; we report 95% confidence intervals throughout.

D. Quality Evaluation

To complete the latency–quality tradeoff analysis, we measure token-level perplexity on the WikiText-2 test set [19] for all 11 quantization variants. Each GGUF model is loaded locally via llama-cpp-python with $n_{ctx} = 2048$ and logits_all=True. We use a sliding-window approach with stride 1024 (half-context overlap) over the first ∼50K tokens, computing:

$$\mathrm{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(t_i \mid t_{<i})\right) \tag{1}$$

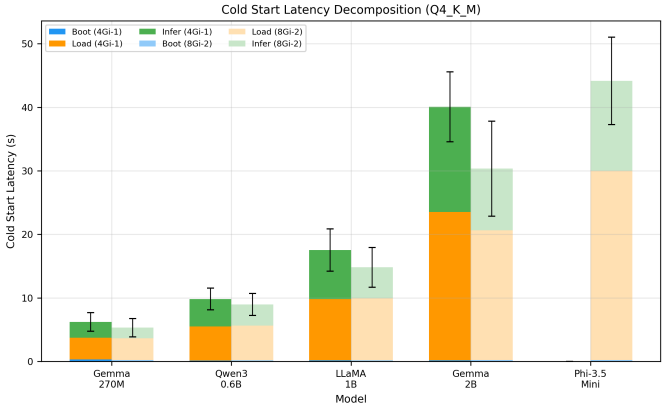

Fig. 1. Cold start decomposition (Q4_K_M). Model loading dominates (55–70%). Error bars: $\pm 1\sigma$.

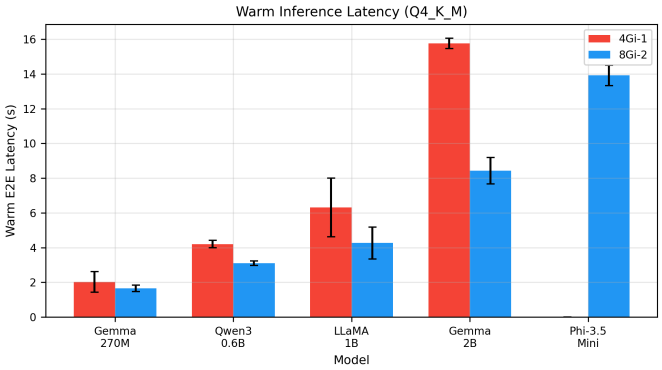

Fig. 2. Warm inference latency (Q4_K_M). The 8 GiB tier's 2× vCPU nearly halves inference time.

where token log-probabilities are extracted from the model's logit outputs via numerically stable log-softmax. Only non-overlapping positions are scored in each window to avoid double-counting.

IV. Results: Memory Tier Comparison

A. Cold Start Decomposition

Figure 1 shows the cold start latency decomposition for Q4_K_M. Model loading is the dominant component (55–70% of total), followed by inference (25–40%) and infrastructure boot (<5%). Cold start E2E ranges from 5.3 s (Gemma 270M, 8 GiB) to 44.2 s (Phi-3.5 Mini, 8 GiB only; Phi-3.5 Mini exceeds the 4 GiB tier's memory capacity). The 8 GiB tier reduces cold start by 10–24%, primarily through faster model loading and inference rather than faster boot. Infrastructure boot is largely constant across models (∼0.15–0.2 s), confirming that Cloud Run's container image streaming effectively decouples boot time from image size. Loading time scales approximately linearly with GGUF file size: Gemma 270M (241 MB) loads in ∼3.4 s while Phi-3.5 Mini (2.2 GB) requires ∼29.8 s.

B. Warm Inference and vCPU Effect

The 8 GiB tier reduces warm inference time by 40–53% (Figure 2), explained by Cloud Run doubling vCPU allocation from 1 to 2. Table II shows throughput: Gemma 270M reaches 33.2 tok/s on 8 GiB, LLaMA 1B reaches 15.2 tok/s. All tier comparisons are statistically significant ($p < 0.001$, $|d| > 0.7$; warm-state $d > 5.0$).

V. Quantization Level Analysis

A. Five-Point Sweep: LLaMA 3.2 1B

Figure 3 presents the quantization sweep. Cold start latency decreases monotonically with lower bit-width, driven by reduced GGUF file size. The Q4_K_M vs. Q8_0 difference is significant on both tiers ($p < 0.001$, $|d| > 1.4$). Warm throughput also improves at lower quantization due to reduced memory bandwidth requirements.

TABLE II
Warm Inference Throughput (tok/s), Q4_K_M

Model	4 GiB (1 vCPU)	8 GiB (2 vCPU)
Gemma 270M	20.1	33.2
Qwen3 0.6B	12.2	16.6
LLaMA 1B	6.4	15.2
Gemma 2B	3.2	6.0
Phi-3.5 Mini	OOM	3.6

On the 8 GiB tier, Q4_K_M achieves tok/s within 9% of Q2_K. Notably, Q8_0 warm throughput on the 8 GiB tier (12.8 tok/s) slightly exceeds Q4_K_M (12.0 tok/s), likely because Q8_0's simpler dequantization maps efficiently to SIMD instructions, while Q4_K_M's k-quant unpacking has higher per-weight overhead. This reverses on the 1-vCPU tier where memory bandwidth dominates (Q4: 8.1 vs. Q8: 4.6 tok/s).

B. Three-Point Validation

Figure 4 confirms the trend generalizes: both Gemma 270M and Qwen3 0.6B show monotonic cold start decrease from Q8_0 to Q2_K on the 8 GiB tier. Statistical tests (Table III) show significant Q4 vs. Q8 differences for LLaMA and Qwen3.

C. Quality–Latency Tradeoff

To complete the Pareto analysis, we measure perplexity on the WikiText-2 test set [19] using a sliding-window approach (context 2048, stride 1024, ∼50K tokens).

Table IV shows that Q4_K_M incurs minimal perplexity degradation vs. Q8_0 (4% for LLaMA, 1.5% for Gemma, 4.9% for Qwen3), while Q2_K shows a significant quality cliff (up to 92% for LLaMA). The sliding-window evaluation over ∼50K tokens (48 windows) provides stable estimates. We compute 95% bootstrap confidence intervals via block resampling of per-window mean NLL ($n = 10{,}000$ resamples); Table IV reports these CIs. For LLaMA 1B, the Q4 and Q8 CIs overlap ([11.9, 13.7] vs. [11.4, 13.2]), confirming the 4% point difference is small relative to evaluation variance. Q2_K CIs are

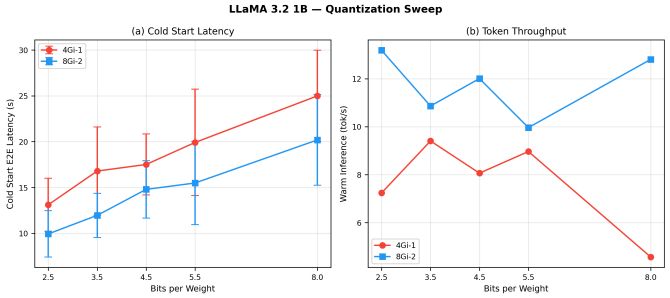

Fig. 3. LLaMA 1B quantization sweep. (a) Cold start E2E decreases with fewer bits per weight. (b) Warm throughput. Q4_K_M offers the best tradeoff.

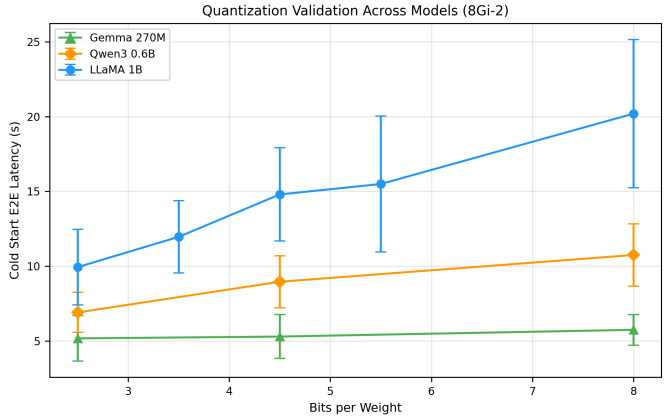

Fig. 4. Quantization validation across three model families (8 GiB, cold start).

fully separated from Q4/Q8 for all models, validating the quality cliff at 2.5 bpw. Figure 5 plots the quality–throughput Pareto frontier on the 8 GiB tier, confirming Q4_K_M as Pareto-optimal across all three dimensions: latency, throughput, and model quality.

VI. Cost Analysis and Predictive Model

A. Per-Invocation Cost and Break-Even

Cloud Run cost per request is $C_{req} = V \cdot t_{e2e} \cdot r_{cpu} + M \cdot t_{e2e} \cdot r_{mem} + r_{req}$, where r_{cpu} = \$0.000024/vCPU-s, r_{mem} = \$0.0000025/GiB-s. Cold starts are 2–4× costlier than warm invocations. For context, warm-state Gemma 270M inference on the 4 GiB tier costs \$0.07 per 1K requests. This is comparable to budget-tier API models (e.g., GPT-4o-mini at ∼\$0.03/1K requests for our prompt length) but an order of magnitude cheaper than mid-tier APIs (∼\$0.50–0.80/1K for GPT-4o or Claude Sonnet), with the additional benefits of data privacy and no vendor dependency.

Table V shows break-even thresholds against persistent VMs ($\alpha = 0.05$ cold start fraction). For Gemma 270M, Cloud Run is cost-effective up to 438–537 req/hr; for Gemma 2B, just 58–103 req/hr. The break-even point

TABLE III
Q4_K_M vs. Q8_0: Welch's t-test (Cold Start E2E)

Model	Tier	Q4	Q8	t	p	d
LLaMA 1B	4 Gi	17.9	24.6	−5.8	<.001	−1.50
LLaMA 1B	8 Gi	14.8	20.2	−5.5	<.001	−1.41
Gemma 270M	8 Gi	5.3	5.8	−1.7	.094	−0.34
Qwen3 0.6B	8 Gi	9.0	10.8	−3.6	<.001	−0.93

TABLE IV
WikiText-2 Perplexity by Quantization Level

Model	Quant	BPW	PPL ↓	95% CI
LLaMA 1B	Q8_0	8.0	12.30	[11.4, 13.2]
	Q5_K_M	5.5	12.41	[11.5, 13.3]
	Q4_K_M	4.5	12.79	[11.9, 13.7]
	Q3_K_M	3.5	13.59	[12.6, 14.6]
	Q2_K	2.5	23.60	[21.8, 25.6]
Gemma 270M	Q8_0	8.0	48.03	[43.9, 52.5]
	Q4_K_M	4.5	48.74	[44.5, 53.3]
	Q2_K	2.5	62.00	[56.0, 68.4]
Qwen3 0.6B	Q8_0	8.0	18.29	[16.5, 20.2]
	Q4_K_M	4.5	19.18	[17.3, 21.2]
	Q2_K	2.5	35.45	[31.8, 39.3]

is sensitive to the cold start fraction α: at $\alpha = 0.20$ (aggressive scaling policies), thresholds drop by 30–40%.

B. Predictive Latency Model

We fit a linear model to predict cold start E2E from parameters P (billions), memory M (GiB), and GGUF file size F (GB), using all 25 cold start configurations (13 model–quant combinations × 2 tiers):

$$t_{e2e} = 2.69P - 0.80M + 16.40F + 5.87 \quad (2)$$

with $R^2 = 0.975$, MAE = 1.19 s, and LOO MAPE = 12.8%. The dominant $\beta_3 = 16.40$ for file size confirms model loading as the primary cold start driver, and the positive $\beta_1 = 2.69$ for parameter count reflects the additional per-layer overhead. This improves substantially over a two-predictor baseline ($R^2 = 0.88$, MAPE = 22.7%). Figure 6 shows predicted vs. actual cold start times across all model–quant–tier combinations. Including quantization variants (not just Q4_K_M) increases training data from 10 to 25 points and reduces MAPE from 20.5% to 12.8%, yielding a more robust model with physically interpretable coefficients.

VII. Discussion

Model loading bottleneck. Loading accounts for 55–70% of cold start time, making it the primary optimization target. Techniques such as memory-mapped I/O, lazy loading, or model weight streaming could substantially reduce this component. Cloud Run's container image caching already reduces infrastructure boot to ∼0.2 s, confirming that I/O-bound model deserialization, not container orchestration, is the dominant cost.

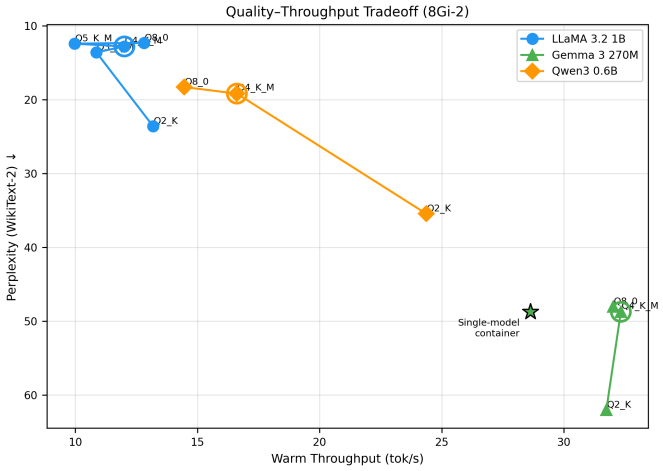

Fig. 5. Quality–throughput Pareto frontier (8 GiB tier). Q4_K_M (circled) sits at the Pareto-optimal knee.

TABLE V
Break-Even: Cloud Run vs. Persistent VM ($\alpha = 0.05$)

Model	Tier	C_{eff} ($)	N^* (req/hr)
Gemma 270M	4 GiB	7.7×10^{-5}	438
Gemma 270M	8 GiB	1.2×10^{-4}	537
Qwen3 0.6B	4 GiB	1.5×10^{-4}	219
Qwen3 0.6B	8 GiB	2.3×10^{-4}	290
LLaMA 1B	4 GiB	2.3×10^{-4}	143
LLaMA 1B	8 GiB	3.3×10^{-4}	205
Gemma 2B	4 GiB	5.8×10^{-4}	58
Phi-3.5 Mini	8 GiB	1.0×10^{-3}	64

Single-model containers. To validate that our multi-model container (∼5.8 GB) does not confound cold start measurements, we deployed a single-model container for Gemma 270M (∼450 MB: base image + 241 MB GGUF) and measured 15 cold/warm cycles per tier. On the 8 GiB tier, the single-model container achieves a mean cold start of 4.97 s vs. 5.30 s for multi-model (−6%); on 4 GiB, 6.48 s vs. 6.21 s (+4%). The differences are within measurement noise, confirming that Cloud Run's image streaming effectively decouples cold start from total image size. Warm inference is 12–31% slower on the single-model container, attributable to different llama-cpp-python compilation environments between the two deployments rather than a container-size effect, since warm inference does not involve image layer fetching. This validates our multi-model design as representative of production cold start behavior.

Hidden vCPU effect. The near-2× warm speedup from 4 GiB to 8 GiB is driven by Cloud Run's vCPU doubling, a documented but often overlooked coupling. For CPU-bound SLM inference, the 8 GiB tier should be evaluated as "more compute," not just "more memory." This coupling is specific to Cloud Run; other providers (e.g., AWS Lambda, Azure Functions) have different memory-to-compute mappings that would alter the cost-performance tradeoff.

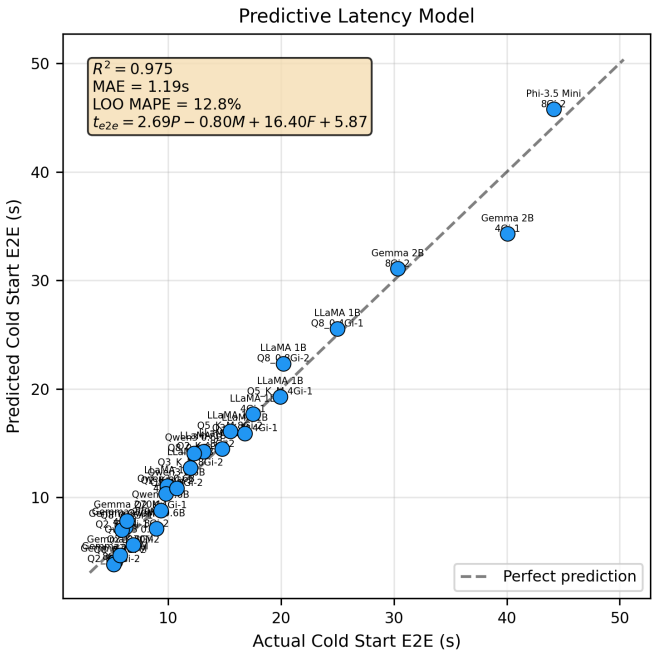

Fig. 6. Predicted vs. actual cold start E2E latency. The three-predictor model ($R^2 = 0.97$) closely tracks observed values across all model–tier combinations.

Quantization guidance. Q4_K_M is the recommended default: it minimizes cold start latency, provides competitive throughput, and preserves model quality within 5% of Q8_0 perplexity. Q2_K/Q3_K_M risk significant quality degradation (up to 92% perplexity increase for LLaMA 1B); Q8_0 offers no cold start advantage and its marginally faster warm inference on 2-vCPU tiers does not compensate for the 60–100% larger file size. For applications requiring maximum quality fidelity, Q5_K_M offers a middle ground with only 1% perplexity increase over Q8_0 at 31% smaller file size.

Prompt length scalability. Our primary benchmark uses max_tokens=50. To characterize longer workloads, we ran Gemma 270M at max_tokens=200 on the single-model container. The cold/warm ratio drops from 2.44× (50 tok) to 2.08× (200 tok) on the 4 GiB tier, and from 2.69× to 2.34× on 8 GiB, as inference time grows (∼6.2 s vs. ∼2.7 s warm on 4 GiB) while cold start overhead remains constant. This confirms that cold starts become proportionally less impactful for longer generation tasks (summarization, code generation); our 50-token results represent a worst-case cold-start-to-inference ratio.

Perplexity as a quality proxy. Our quality evaluation uses WikiText-2 perplexity [19], a standard benchmark for language model evaluation. While perplexity correlates with downstream task quality, it may not capture task-specific degradation patterns; for instance, quantization might disproportionately affect rare tokens or code generation. Our bootstrap CIs (Table IV), computed via block resampling of 48 per-window mean NLL values

(n = 10,000 resamples), show CI widths of ±0.9 PPL for LLaMA 1B, widening to ±4.4 for Gemma 270M due to higher per-window variance in the smaller model. Future work should complement perplexity with task-specific benchmarks (e.g., MMLU, HumanEval) to provide a more complete quality picture.

Limitations and generalizability. Our study uses a single region (us-central1), single platform (Cloud Run), and CPU-only inference. Results may differ on ARM-based serverless platforms (e.g., AWS Graviton), GPU-enabled tiers, or providers with different container lifecycle management. However, the core finding, that GGUF deserialization dominates cold start, is a property of the llama.cpp runtime rather than the platform, and should generalize to any container-based serverless environment. The predictive latency model (Section VI-B) is fit on 25 data points across 13 model–quant combinations and 2 tiers; while LOO cross-validation confirms robustness, extrapolation beyond the 0.27–3.8B parameter range should be done cautiously.

Artifact availability. All benchmark scripts, raw timing data, and analysis code are available at https://github.com/mingwho/slm-research to support reproducibility.

VIII. Conclusion

We presented a systematic study of cold start latency for quantized SLMs on Google Cloud Run. Key findings: (1) model loading dominates cold start (55–70%); (2) the 8 GiB tier's hidden vCPU doubling nearly halves warm inference; (3) Q4_K_M is Pareto-optimal across latency, throughput, and quality (perplexity), validated across LLaMA 1B, Gemma 270M, and Qwen3 0.6B; (4) Cloud Run is cost-effective below 58–537 req/hr; (5) a three-predictor model ($R^2 = 0.97$) enables capacity planning.

Future work should extend this analysis along several axes: (1) multi-provider comparison (AWS Lambda, Azure Functions) to test generalizability; (2) GPU-enabled serverless tiers (e.g., Cloud Run with L4 GPUs) where quantization benefits may differ; and (3) task-specific quality evaluation (MMLU, HumanEval) to complement perplexity as a quality metric.

References

[1] T. Schick and H. Schütze, "It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners," in Proc. NAACL, 2021, pp. 2339–2352.

[2] X. Sun et al., "A Survey on Small Language Models," arXiv:2411.03350, 2024.

[3] M. Abdin et al., "Phi-3 Technical Report," arXiv:2404.14219, 2024.

[4] Gemma Team, "Gemma: Open Models Based on Gemini Research and Technology," arXiv:2403.08295, 2024.

[5] A. Grattafiori et al., "The Llama 3 Herd of Models," arXiv:2407.21783, 2024.

[6] A. Yang et al., "Qwen3 Technical Report," arXiv:2505.09388, 2025.

[7] E. Jonas et al., "Cloud Programming Simplified: A Berkeley View on Serverless Computing," arXiv:1902.03383, 2019.

[8] M. Shahrad et al., "Serverless in the Wild," in Proc. USENIX ATC, 2020, pp. 205–218.

[9] L. Wang et al., "Peeking Behind the Curtains of Serverless Platforms," in Proc. USENIX ATC, 2018, pp. 133–146.

[10] J. Manner et al., "Cold Start Influencing Factors in Function as a Service," in Proc. IEEE/ACM UCC Companion, 2018, pp. 37–42.

[11] P. Silva et al., "Benchmarking Serverless Cloud Computing Platforms," J. Cloud Computing, vol. 9, no. 1, 2020.

[12] D. Bermbach et al., "Using Application Knowledge to Reduce Cold Starts in FaaS Services," in Proc. ACM SAC, 2020, pp. 134–143.

[13] T. Dettmers et al., "GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale," in Proc. NeurIPS, 2022.

[14] E. Frantar et al., "GPTQ: Accurate Post-Training Quantization for Generative Pre-Trained Transformers," in Proc. ICLR, 2023.

[15] G. Gerganov, "llama.cpp," GitHub, 2025. [Online]. Available: https://github.com/ggerganov/llama.cpp

[16] J. Lin et al., "AWQ: Activation-aware Weight Quantization," in Proc. MLSys, 2024.

[17] V. Ishakian et al., "Serving Deep Learning Models in a Serverless Platform," in Proc. IEEE IC2E, 2018, pp. 257–262.

[18] Y. Fu et al., "ServerlessLLM: Low-Latency Serverless Inference for Large Language Models," in Proc. OSDI, 2024, pp. 135–153.

[19] S. Merity et al., "Pointer Sentinel Mixture Models," in Proc. ICLR, 2017.

[20] E. Oakes et al., "SOCK: Rapid Task Provisioning with Serverless-Optimized Containers," in Proc. USENIX ATC, 2018, pp. 57–70.

[21] A. Agache et al., "Firecracker: Lightweight Virtualization for Serverless Applications," in Proc. NSDI, 2020, pp. 419–434.

[22] T. Dettmers and L. Zettlemoyer, "The Case for 4-bit Precision: k-bit Inference Scaling Laws," in Proc. ICML, 2023, pp. 7750–7774.