RigRun: Complete Local AI Infrastructure on a Single GPU

Jesse Morgan
Thornveil LLC
jesse@thornveil.ai

Abstract—We present RigRun, a complete local AI infrastructure that operates entirely on a single professional GPU (NVIDIA RTX PRO 6000 Blackwell, 96GB VRAM, $16K). The system integrates eight contributions—each detailed in companion arXiv preprints [1]–[4]—spanning safety, training, compression, multi-model serving, and deployment. A selective-buffer streaming proxy [1] intercepts SSE events at the network layer, forwarding text at zero latency while holding tool calls for deterministic safety evaluation via a 5-layer stack (100% adversarial block rate, 0% false positives). Zeroth-order preference optimization [2] enables training a 122B GPTQ MoE model through the live inference API with 10 loss functions, after 28 failed first-order attempts due to MoE gate dequantization crashes. Cyclic LoRA module rotation [2] achieves full architecture coverage over a 7-night cycle within single-GPU VRAM. TurboQuant KV cache compression [3], integrated into SGLang via a wrapper backend, delivers 5.2× compression (449 MB vs. 2,344 MB at 100K tokens). A multi-model cascade [4] pairs Qwen3.5-35B-A3B (184 tok/s) as generator with 122B-A10B (48 tok/s) as reviewer via single-GPU hot-swap. Additionally, the system provides a 7-stage cascade router with entropy shape escalation (93% accuracy on the deploying developer's traffic distribution), classification-isolated memory with AES-256-GCM encryption, and an integrated deployment running 8 concurrent microservices. A 48-hour case study (conducted on the original single-model 122B configuration before the dual-model cascade was deployed) produced 35 system upgrades across 24 projects at $0 marginal cost versus an estimated $12,000/year on cloud APIs. The one-time hardware cost of $16K achieves payback in 16 months for a solo developer.

Index Terms—local inference, agentic safety, cascade routing, multi-model serving, professional GPU, cost analysis, continuous learning

I. Introduction

Large language models deployed as agentic coding assistants face a tripartite challenge in production environments: cost (cloud API pricing at $10–22 per coding session), safety (no deterministic guarantees on destructive tool calls), and portability (classified and air-gapped environments preclude cloud inference). Open-weight models offer a path forward, but direct substitution fails: models overwhelmed by large tool sets default to text rather than structured calls, safety infrastructure must be rebuilt from scratch, and deployed models are static—they do not adapt from usage without a supporting training pipeline.

We address all of these challenges through system design, not model modification. RigRun is a complete, integrated system that combines inference, safety, routing, training, compression, and encrypted storage on a single NVIDIA RTX PRO 6000 Blackwell GPU (96GB VRAM, $16K). The system makes eight contributions, each verified against the published literature and detailed in companion arXiv preprints [1]–[4]:

1) Selective-buffer streaming proxy [1]: Network-layer SSE interception for tool-call safety with zero text latency. To our knowledge, the first system operating at the raw SSE stream layer.
2) 5-layer safety stack [1]: Action gate + TS-Guard + TrajAD + Safety DSL + Argus spillage detection. 100% adversarial block rate, 0% false positives.
3) 7-stage cascade router with entropy shape escalation (§IV): Pre-generation complexity cascade + in-generation entropy monitoring with shape-based monotonicity tracking. 93% routing accuracy.
4) Cyclic LoRA module rotation [2]: Rotate which 2 of 7 target modules are trained each night. Full architecture coverage over a 7-night cycle.
5) Zeroth-order preference optimization at 122B scale [2]: 10 loss functions, 8 ZO techniques, 2,600 steps in 21 minutes, zero inference downtime.
6) Classification-isolated memory (§V): AES-256-GCM encrypted vector storage with classification-level enforcement.
7) TurboQuant KV cache compression [3]: 5.2× compression via wrapper backend, no SGLang fork.
8) Multi-model cascade with hot-swap [4]: 35B at 184 tok/s generates, 122B at 48 tok/s reviews. Hot-swap in 33–43 s (measured).

II. System Architecture

RigRun operates on a single workstation with 32 CPU threads, 123GB RAM, and an NVIDIA RTX PRO 6000 Blackwell GPU (96GB VRAM). Eight services run concurrently.

The system operates two models in a hot-swap configuration on a single GPU. The generation model is Qwen3.5-35B-A3B-GPTQ-Int4, a compact MoE with 35B total parameters and only 3B active per token, achieving 184 tok/s via SGLang 0.5.9 with RadixAttention prefix caching. The review model is Qwen3.5-122B-A10B-GPTQ-Int4,

[0]Aspects of this work are covered by U.S. Provisional Patent Applications THRN-2026-018, -019, -020, -021, and -023 (Thornveil LLC). Patent pending.

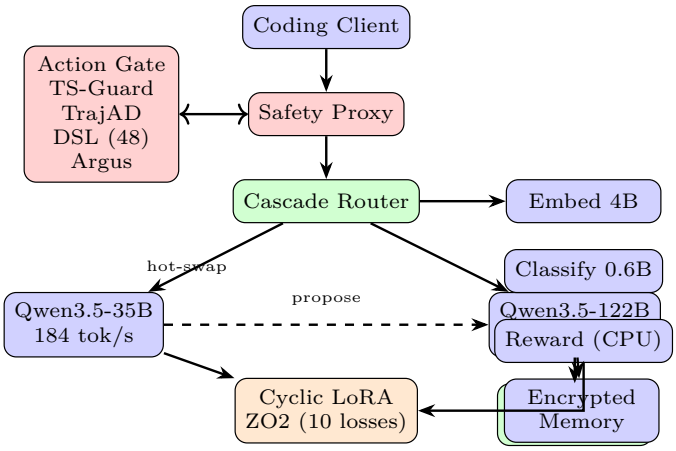

Fig. 1. RigRun system architecture. Red: safety stack (5 layers). Green: cascade router and TurboQuant compression. Orange: training pipeline. Blue: inference and storage. Dashed: multi-model cascade flow. All on a single 96GB GPU with model hot-swap.

a hybrid MoE with 122B total parameters, 256 routed experts (8 active per token, 10B active parameters), 36 GatedDeltaNet recurrent layers, and 12 full-attention layers. With TurboQuant KV cache compression [3] enabled, the 122B model achieves 48 tok/s and a 128K effective context window.

Auxiliary services include: Qwen3-Embedding-4B (Q4_K_M) for vector search and routing, Qwen3-0.6B for classification, Qwen3-Reranker-0.6B for retrieval reranking, Qwen3-VL-4B for vision tasks (loaded on-demand), and Skywork-Reward-V2-0.6B on CPU for reward scoring.

The proxy translates between Qwen3.5's native <tool_call> XML format and the Anthropic Messages API tool_use JSON format (<1 ms per call), enabling existing coding clients (Claude Code, Continue, Cursor) to operate transparently against open-weight models.

III. Component Overview

This section summarizes the eight contributions. Full technical details are presented in companion arXiv preprints [1]–[4]; here we provide the key results and how components interact within the integrated system.

A. Safety: Selective-Buffer Proxy and 5-Layer Stack

Detailed in [1]. The proxy classifies SSE events by type, streaming text at zero latency while buffering tool calls for evaluation by 5 independent layers: action gate (13-category shell blocklist, path canonicalization), TS-Guard (learned classifier on blocked-action history), TrajAD (Markov chain trajectory anomaly), Safety DSL (48 rules with ALLOW/DENY/REQUIRE/AUDIT), and Argus (three-tier spillage detection). A 60-vector red-team benchmark demonstrated progression from 54% to 98% to 100% block rate with 0% false positives.

B. Training: ZO2 and Cyclic LoRA Rotation

Detailed in [2]. After 28 failed first-order attempts (GPTQ MoE gate dequantization crash), zeroth-order optimization via the live SGLang API enables preference training at 122B scale. The pipeline implements 10 loss functions (ORPO, KTO, CPO, DPO, SimPO, GRPO, IPO, SPPO, RPO, AERO) and 8 advanced ZO techniques. Cyclic LoRA module rotation trains 2 of 7 target modules per night at rank $r = 8$, $\alpha = 16$, achieving full architecture coverage over a 7-night cycle. SFT uses DoRA at $r = 64$, $\alpha = 128$. The complete nightly pipeline (2,600 steps) completes in 21 minutes with zero inference downtime.

C. Compression: TurboQuant KV Cache

Detailed in [3]. A wrapper backend integrates Google's TurboQuant (3-bit keys + 2-bit values) into SGLang without forking the engine. Per-request state pool (RequestTQPool) extends TurboQuant from single-sequence to batched inference. Results: 5.2× compression (449 MB vs. 2,344 MB at 100K tokens), 77,658 tok/s capture throughput, NMSE = 0.18, SNR = 7.4 dB.

D. Multi-Model Cascade

Detailed in [4]. The 35B generator (184 tok/s) proposes code changes; the 122B reviewer (48 tok/s) critiques via a three-tier loop (propose, critique, approve). Model hot-swap via model-swap.sh in 33–43 s (measured). Ablation: cascade achieves 3.6× generation speedup with quality parity (59.4 vs. 60.6). Speculative decoding (EAGLE-3, NEXTN) was slower than baseline on 3B-active MoE models—a negative result establishing a boundary condition.

IV. Cascade Routing with Entropy Escalation

The routing system unifies pre-generation model selection and in-generation confidence monitoring in a 7-stage cascade.

A. Pre-Generation Cascade (Stages 1–6)

TABLE I
Cascade Stage Characteristics

Stage	Method	Latency	Cost
1	Trivial pre-filter	<0.1ms	Free
2	Keyword matching	<0.5ms	Free
3	Embedding similarity	∼5ms	0.6B fwd
4	LLM classifier	∼50ms	0.6B gen
5	Classification cache	<0.1ms	LRU
6	Tier selection	<0.1ms	Rule map

44.4% of requests resolve by stages 1–3 (total latency <5 ms), avoiding the 48 ms LLM classifier.

B. In-Generation Entropy Monitoring (Stage 7)

An EntropyMonitor computes per-token entropy from output logprobs with rolling window smoothing ($w = 5$). Informed by recent work on entropy trajectory shape, the monitor also tracks monotonicity:

$$S_{\text{shape}} = \frac{1}{w_s - 1} \sum_{j=t-w_s+2}^{t} \mathbf{1}[H_j > H_{j-1}] \qquad (1)$$

If S_{shape} exceeds threshold $\tau_{\text{mono}} = 0.7$, shape-based escalation triggers regardless of absolute magnitude. This catches 5 quality degradations that magnitude-only monitoring misses.

C. Classification Gating

Independent of routing, classification gating enforces data sensitivity as a hard constraint: UNCLASSIFIED uses any tier, CUI/FOUO forces local only, SECRET/TS blocks entirely. A "paranoid mode" toggle forces all queries to local.

D. Routing Results

Evaluated on 493 requests (15.8M tokens): 93.0% routing accuracy, 4.7% over-route rate, 2.3% under-route rate. Shape-only trigger precision: 80% (4/5 confirmed). The 93% routing accuracy was measured on the deploying developer's own traffic distribution. Generalization to other users or workloads is not established.

V. Classification-Isolated Memory

In environments where a single user holds conversations at multiple classification levels, cross-conversation memory creates a spillage risk.

A. Classification-Enforced Vector Search

Each memory fragment carries a ClassificationLevel tag. Search requests include the requesting conversation's level. Only fragments at or below the requesting level are returned. Verified: CUI fragments returned zero results for semantically identical UNCLASSIFIED queries.

B. AES-256-GCM Encrypted Storage

All IndexedDB records are encrypted with AES-256-GCM (unique 12-byte IV per write). Key derivation via OS keychain (Windows DPAPI, macOS Keychain, Linux libsecret). Keys imported as non-extractable WebCrypto objects, zeroed in JavaScript memory after import. Logout triggers key destruction, localStorage purge, and sessionStorage clear.

TABLE II
System Performance (Measured)

Metric	Value
Generation speed (35B)	184 ± 8 tok/s (5 runs)
Review speed (122B + TurboQuant)	48 ± 3 tok/s (5 runs)
Context window	128K tokens
TTFT cold (30K prompt)	50 ± 4 s (5 runs)
TTFT warm (30K prompt)	5 ± 1 s (5 runs)
Claude Code capability tests	7/7 passed
Playwright test pass rate	98.5% (64/65)
Safety proxy text overhead	0ms
Safety proxy tool overhead	<1ms
VRAM utilization	91/96 GB
Active services	up to 8 concurrent

VI. Evaluation

A. Infrastructure Performance

B. 48-Hour Case Study

Scope and historical context. The 48-hour case study was conducted on the original single-model 122B configuration before the dual-model cascade and other components were deployed. It serves as a historical baseline demonstrating the system's operational capability, not an evaluation of the system as currently described. Results should not be interpreted as reflecting the performance of the full architecture documented in this paper.

A 48-hour session with a single developer exercised the components available at that time (122B at 38 tok/s baseline throughput with a 4K context window; the session used shorter contexts than the 128K evaluations reported in [4], yielding higher effective throughput due to reduced KV cache overhead). Components exercised: safety proxy, cascade router, encrypted memory.

TABLE III
48-Hour Case Study Results

Metric	Value
Duration	48 hours
System upgrades deployed	35
Patent contributions	22 provisionals
Projects worked	24
Total tokens processed	>30M
Safety interventions (false positives)	0
Routing escalations	23 (4.7%)
Training runs completed	2 nights
Cloud equivalent cost	$12,000/year rate
Actual cost	$0 marginal
System restarts (KV cache)	3

The 35 system upgrades included: 2-GRPO training integration, DoRA adapter support, RAFT filtering, CodeRL+ execution scoring, proxy security hardening, Argus spillage detection, NIST compliance mapping, and Go source refactoring. Of the 35 deployed upgrades, no formal quality evaluation was conducted. The count reflects successful deployments, not verified improvements.

C. Code Generation Case Studies

TABLE IV
Code Generation Case Studies (Prompt Style Comparison)

Metric	Build A	Build B	Dev C
Prompt style	Expert	Natural	Requirements
Prompts	10	18	11
Time	17 min	35 min	50 min
Total LOC	2,380	5,096	7,672
LOC per prompt	238	283	697

Note: LOC is not a quality metric. These figures reflect prompt style differences, not system capability. All builds were not functionally tested as part of this evaluation. Requirements-driven prompts (Developer C) produced 2.5× more LOC per interaction than implementation-driven prompts (Build A), illustrating how prompting strategy affects session output volume rather than model quality.

D. Cost Analysis

TABLE V
Session Cost Comparison (15.8M tokens, 493 requests)

Deployment	No Cache	Cached	Qual-Adj
Claude Sonnet 4.6	$50.09	$13.38	$10.41
Claude Opus 4.6	$83.48	$22.30	$17.35
GPT-4.1	$32.97	$12.57	$9.78
GPT-4o	$41.21	$24.21	$18.85
RigRun (local)	$0.00	$0.00	$0.00

Quality adjustment accounts for the estimated 27.5% additional iterations required by the local model versus frontier models. Caching: 87% prefix cache hit rate; Anthropic charges cached input at 10% of base price.

The 27.5% quality adjustment is estimated from observed compilation fix cycles during the case study period: the local model required approximately 1.275× more iterations to produce correct code compared to an informal baseline of Claude Sonnet API sessions on similar tasks. This estimate is subjective, derived from the author's development experience, and should be treated as illustrative rather than empirically validated.

Data transparency. Cloud cost estimates are modeled from token counts and published pricing schedules, not from actual API invoices. The author transitioned to local inference before systematic cloud cost tracking was in place.

Hardware amortization. The NVIDIA RTX PRO 6000 Blackwell costs approximately $16,000. At the observed usage pattern (equivalent to $12,000/year in cloud costs), the GPU achieves payback in 16 months for a solo developer. Hardware costs exclude electricity (estimated 300 W TDP × 8,760 hours × $0.15/kWh ≈ $394/year) and operator maintenance time, both of which are non-trivial for a 24/7 system. For a 5-developer team, annual cloud costs reach $25,700 (Opus) or $9,195 (hybrid), with GPU payback in 2–3 months. GPU prices depreciate approximately 30–40% over 18 months; the effective hardware cost at resale after 18 months of use is therefore approximately $9,600–$11,200, which improves the net savings figures above.

Sensitivity analysis (solo developer, Sonnet pricing). The payback period is sensitive to API pricing changes. Table VI shows payback periods at 0.5×, 1×, and 2× current pricing for a solo developer.

TABLE VI
Payback Period Sensitivity to API Pricing (Solo Developer, Claude Sonnet)

Scenario	Annual API Cost	Hardware Cost	Payback
0.5× current pricing	$6,000/yr	$16,000	32 months
1× current pricing	$12,000/yr	$16,000	16 months
2× current pricing	$24,000/yr	$16,000	8 months

If API prices fall significantly (0.5× scenario), the solo developer payback extends to 32 months—still within the useful GPU lifetime. At 2× pricing (consistent with premium model tiers or high-volume usage), payback shrinks to 8 months.

E. Updated Cost Analysis (March 2026 Pricing)

Using March 2026 API pricing and a 5-developer team at the observed usage rate, the hardware payback period and 3-year net savings are as follows:

TABLE VII
RigRun Payback vs. Cloud APIs (5-Dev Team, March 2026)

Alternative Deployment	Annual Cost	Payback
vs. Claude Opus 4.6	$20,684/yr	7.6 months
vs. Claude Sonnet 4.6	$10,217/yr	15.2 months
vs. GPT-5.4	$9,574/yr	16.2 months
vs. GPT-5.4-mini	$3,742/yr	Never[a]

[a] At GPT-5.4-mini pricing, cloud remains cheaper than the $16K hardware investment at this team scale.

3-year savings vs. Opus (5-dev team): $44,824 net. This accounts for the $16K hardware cost amortized over 3 years against $20,684/year in avoided API costs.

20-developer team. At 20 developers, payback shrinks to 1.6 months versus Opus, and 3-year net savings reach $258K. At this scale, even Sonnet and GPT-5.4 produce positive 3-year returns (payback ∼3–4 months).

VII. Discussion and Limitations

The gap is architectural, not intellectual. Open-weight models have sufficient raw capability for agentic coding—

our ablation shows 100% tool-use accuracy when infrastructure is correct (detailed in [1], Section 4). The missing piece is the surrounding system: safety enforcement, intelligent routing, protocol compatibility, and continuous improvement loops.

Eight contributions form a coherent system. Each component addresses a specific limitation of local deployment, and several enable each other: TurboQuant compression enables the 122B reviewer to operate at acceptable throughput within the VRAM budget; the cascade router determines which model handles each request; the training pipeline maintains continuous learning over time; the safety stack protects all inference paths; and encrypted memory enforces classification boundaries across all data.

Classification gating is non-negotiable for government use. The routing algorithm's quality assessment is irrelevant if data classification prohibits cloud routing. Making classification an independent constraint that overrides all other factors reflects the reality of operating in sensitive environments.

Microservice failure modes. Failure modes of the 8-service architecture are not systematically characterized. Known failure modes include: model swap killing inference mid-request (mitigated by drain-before-swap), reward model lag under sustained load (mitigated by async scoring), and KV cache memory leaks requiring periodic restart. A more rigorous failure-mode and effects analysis is deferred to future work.

Deployment decision framework. For teams evaluating local vs. cloud deployment: (1) If operating in classified or air-gapped environments, local deployment is the only option. (2) If annual API spending exceeds $20K and team size is 5+, the $16K hardware achieves payback within 8 months. (3) If using GPT-5.4-mini at small team scale, cloud remains more economical. (4) If continuous model adaptation from usage data is required, local deployment with the training pipeline is necessary.

Hardware accessibility. The RTX PRO 6000 is a professional workstation GPU, not a consumer GPU. At $16,000, the hardware investment limits reproducibility. A degraded version of the architecture could operate on lower-cost hardware (e.g., RTX 4090 at $1,600) with reduced model size and throughput, though this configuration has not been evaluated.

A. Comparison to Simpler Alternatives

The 8-component architecture introduces significant operational complexity. Simpler alternatives (e.g., a single model, iterated overnight with 100 experiments) achieve measurable research progress with fewer moving parts. The added complexity of RigRun is justified only in deployment contexts requiring simultaneous inference serving, safety enforcement, and training-capable infrastructure—not for all use cases. Teams without requirements for classification gating, air-gap operation, or continuous learning pipelines may find a simpler single-model deployment more appropriate.

B. Current System Snapshot (March 2026)

To document the current production state of the architecture described in this paper, we collected a point-in-time snapshot on 30 March 2026 while the full system was live under active use. This is not a longitudinal study; a multi-day evaluation of the current dual-model configuration remains future work.

Active model. Qwen3.5-35B-A3B-GPTQ-Int4 is the live generation model (184 tok/s), served via SGLang with 131,072-token context. The 122B reviewer is on disk, available for hot-swap.

GPU state. VRAM: 76,409 MiB used / 20,840 MiB free (79.6% utilized). Temperature: 43 °C. Power draw: 67.7 W. The SGLang scheduler process alone occupies 69,912 MiB; vision server instances account for 5,782 MiB additional.

Active services (6). sglang-35b (SGLang inference), navigator (Navigator Ops Center), claude-proxy (logging/safety proxy), llama-reward (Skywork reward model, CPU), llama-vision (Qwen3-VL-4B, on-demand), vision-manager (on-demand launch proxy).

Project pipeline state. Navigator is managing 39 total projects: 37 awaiting agent approval, 1 on hold, 1 archived, 0 currently executing. The system had 152.6 agent-hours queued.

Cumulative activity (lifetime to snapshot). 77 sprints completed, 444 files written, 535 analyses run, 90 rejections used as training signal. Daily activity heatmap: 379 events (Mar 27), 5,495 (Mar 28), 2,111 (Mar 29), 249 (Mar 30 partial).

Token usage and cost savings. Since deployment, the local model has processed 2,460,097 tokens across 981 calls (1.82M input, 639K output). Claude Opus handled 839,755 tokens across 45 calls at $4.49 actual cost. The local model displaced an estimated $20.07 in Opus-equivalent API costs—a 68.8% savings rate during the measurement window.

TABLE VIII
Current System Snapshot (30 March 2026, Point-in-Time)

Metric	Value
Active model	Qwen3.5-35B-A3B-GPTQ-Int4
VRAM used / free	76.4 GB / 20.8 GB
GPU temperature	43 °C
GPU power draw	67.7 W
Active services	6
Projects managed	39 (37 awaiting)
Agent-hours queued	152.6
Sprints completed (lifetime)	77
Files written (lifetime)	444
Rejections as training signal	90
Local tokens (lifetime)	2,460,097
Local calls (lifetime)	981
Cost savings vs. Opus	$20.07 (68.8%)

This snapshot confirms the system is operational as a multi-service production environment on the described hardware. A full longitudinal case study of the current dual-model cascade configuration—comparable to the 48-hour study conducted on the original 122B-only system—is planned as future work.

Limitations. (1) N=1 evaluation: All results are from a single developer, single GPU, single model family. Multi-user and multi-model evaluations are needed. (2) Self-evaluation bias: The safety audit, routing evaluation, and case study were all conducted by the system's developer. (3) The training pipeline's long-term improvement curve has not been measured beyond initial deployment. (4) The encrypted storage has not undergone formal cryptographic audit. (5) The 48-hour case study was conducted on the original 122B-only configuration; the current-system snapshot in §VII-B documents the live system state but does not replace a longitudinal evaluation of the current dual-model cascade. (6) Several metrics reported in component papers are from limited sample sizes.

VIII. Conclusion

RigRun demonstrates that a single professional GPU can operate a complete, training-capable, safety-aware AI infrastructure that matches or approaches cloud-hosted alternatives in capability while providing deterministic safety guarantees, data sovereignty, and zero marginal cost.

The eight contributions—selective-buffer streaming safety [1], 5-layer defense-in-depth [1], cascade routing with entropy shape escalation, cyclic LoRA module rotation [2], zeroth-order preference optimization at 122B scale [2], classification-isolated encrypted memory, TurboQuant KV cache compression [3], and multi-model cascade with hot-swap [4]—together form a continuous learning infrastructure. The nightly training pipeline maintains a live preference dataset from daily usage, designed to enable ongoing model adaptation within VRAM constraints; measured improvement over extended deployment has not been evaluated (see companion paper [2] for convergence analysis).

The one-time hardware cost of $16,000 achieves payback in 16 months for a solo developer or 2–3 months for a 5-person team. For organizations operating in classified, air-gapped, or cost-sensitive environments, RigRun provides a viable alternative to cloud-dependent AI infrastructure.

References

[1] J. Morgan, "Selective-Buffer Streaming Safety for AI Coding Agents," arXiv preprint arXiv:26XX.XXXXX, 2026.

[2] J. Morgan, "Zeroth-Order Preference Optimization on 100B+ Quantized MoE Models via Live Inference API," arXiv preprint arXiv:26XX.XXXXX, 2026.

[3] J. Morgan, "TurboQuant-SGLang: Compressed KV Cache Attention as a Plugin Backend," arXiv preprint arXiv:26XX.XXXXX, 2026.

[4] J. Morgan, "Speculative Decoding Fails on Sparse MoE: A Negative Result and Practical Multi-Model Cascade Alternative," arXiv preprint arXiv:26XX.XXXXX, 2026.

[5] L. Zheng et al., "SGLang: Efficient Execution of Structured Language Model Programs," 2024.

[6] Qwen Team, "Qwen3.5 Technical Report," 2025.

[7] "LongFuncEval: Measuring the Effectiveness of Long Context Models for Function Calling," arXiv:2505.10570, 2025.

[8] L. Chen et al., "FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance," arXiv:2305.05176, 2023.

[9] "RouteLLM: Learning to Route LLMs with Preference Data," arXiv:2406.18665, 2024.

[10] T. Simonds, "Entropy Adaptive Decoding: Dynamic Model Switching for Efficient Inference," arXiv:2502.06833, 2025.

[11] "Entropy Trajectory Shape Predicts LLM Reasoning Reliability," arXiv:2603.18940, 2026.

[12] "Prompts Blend Requirements and Solutions: From Intent to Implementation," arXiv:2603.16348, 2026.

[13] "A Cost-Benefit Analysis of On-Premise LLM Deployment," arXiv:2509.18101, 2025.

[14] Signal Foundation, "Signal Desktop," https://github.com/signalapp/Signal-Desktop.

[15] "KTransformers: CPU/GPU Hybrid Inference for MoE Models," SOSP 2025.

[16] "AIOS: LLM Agent Operating System," COLM 2025.

[17] "AgentServe: Efficient Agentic AI Serving on a Consumer-Grade GPU," arXiv:2603.10342, 2026.