Efficient Request Scheduler for LLM Inference

Chen Chen[1,3], Lei Jiao[2], and Richard Mortier[3]

[1]Nottingham Trent University, UK
[2]University of Oregon, USA
[3]University of Cambridge, UK

Abstract

The deployment of large language models (LLMs) in resource-constrained edge server clusters introduces a fundamental systems challenge: sustaining high token throughput and low end-to-end latency under tight GPU memory capacity and bandwidth limits. Unlike centralized data center environments, edge clusters operate with limited device parallelism and stricter service-level objectives (SLOs), amplifying inefficiencies in request scheduling. Existing LLM serving schedulers are ill-suited for this regime. They treat requests as an independent workload, overlooking application-level structure and phase-specific execution characteristics. This design results in two systemic inefficiencies. First, many edge workloads share identical, lengthy system prompts across requests, yet current schedulers redundantly execute prefill computations for each request, wasting GPU compute and key–value (KV) cache memory. Second, phase-agnostic batching policies mix compute-intensive prefill operations with memory-bound decode steps without coordination, producing suboptimal batches that fail to overlap complementary resource demands, thereby underutilizing GPU execution pipelines.

This paper presents ACK, an application-aware and length-predictive scheduler for edge LLM inference. ACK rethinks the queuing layer as an intelligent orchestration component that exploits semantic commonality and phase heterogeneity to improve resource efficiency. ACK introduces two core techniques. (1) Semantic-aware request grouping: ACK hashes requests by their system prompt and co-locates matching requests within the scheduling window, enabling global KV cache reuse. By eliminating redundant prefix computation, ACK reduces both prefill latency and memory pressure while preserving correctness and isolation. (2) Phase-aware scheduling: ACK incorporates a lightweight output token bucket predictor that estimates each request's decode length and corresponding decode-to-prefill ratio. Leveraging these predictions, ACK strategically reorders requests to construct hybrid continuous batches that maximize overlap between compute-bound prefill stages and memory-bound decode stages, improving GPU pipeline utilization without modifying the underlying LLM engine.

To meet latency targets in multi-tenant edge deployments, ACK further integrates SLO-aware preemption. The scheduler dynamically adjusts batching and execution order to prevent head-of-line blocking and mitigate tail-latency amplification under bursty workloads. Importantly, ACK is designed as a practical, deployable scheduling layer built atop vLLM, requiring no model retraining or architectural modification. This design choice ensures compatibility with existing model implementations and deployment pipelines while enabling immediate performance gains.

We implement ACK and evaluate it on a real-world edge GPU testbed using diverse open-source LLMs and production-level workloads characterized by shared system prompts and heterogeneous output lengths. Across a range of arrival rates and SLO configurations, ACK consistently improves system efficiency. Compared to state-of-the-art schedulers, ACK increases token throughput by up to 1.6× and reduces end-to-end request latency by up to 2.4×, while maintaining SLO compliance under high load. Our results demonstrate that application awareness and lightweight token reordering at the scheduling layer can substantially improve the performance in edge LLM serving.

Keywords— Large language model, inference, queueing, output token prediction