Task-Level Admission for Agent Workloads

Admitting agent tasks whole completed 50% more tasks per GPU-hour at 1.5× load, with no background SLO loss

Pranav Varshney
Nodus Compute

Viswa Kotra
Nodus Compute

Abstract

Agent workloads issue tasks that fan out into several parallel model requests, and a task is only useful when all of them finish in time. Admission control that judges each request alone can admit part of a task, spend GPU time on it and still lose the task. In this study, a research prototype from Nodus, the infrastructure execution layer for AI workloads, admitted or declined each task as a whole, a capability not yet in the product. On Qwen3-0.6B on one GPU, task-level admission completes more tasks by their deadline per GPU-hour than per-request admission at every load we measured: +3.8% at 0.8× load, +11.6% at 1.0×, +25% at 1.2× and +50% at 1.5×. Background interactive traffic loses 0 points of SLO attainment at all loads. The result comes from one seed on a 0.6B model. A companion streaming-credit variant measured in the same campaign failed its pre-registered gate, and we report it as a negative result.

1 Introduction

An agent step rarely sends one request. It sends several in parallel, such as tool calls, candidate plans or retrieval branches, and the step completes only when all of them return before the task's deadline. Under load, a router that admits or rejects each request independently can accept some branches of a task and reject others. The accepted branches consume GPU time, the task misses its deadline anyway, and that GPU time is lost.

The research prototype measured here makes the admission decision once per task. A task is either admitted with all of its branches or declined up front, so GPU time goes to tasks that can finish. We compare this with per-request admission on the same engine, the same model and the same arrival schedule, and we measure completed tasks per GPU-hour together with the SLO of background interactive traffic that shares the GPU. The serving engine is stock vLLM [1] and the model is Qwen3-0.6B [2]. Deadline-aware admission for inference has a long history [3], and our contribution is applying it at the level of the whole agent task inside a research admission router built for this study.

2 Experimental setup

Workload. Multi-branch agent tasks, each with a deadline, arrive alongside background single-stream interactive requests. Arrivals are open loop at 0.8×, 1.0×, 1.2× and 1.5× of the GPU's calibrated capacity, and each load replays one identical arrival schedule through both arms. The run uses Qwen3-0.6B on 1 GPU with 1 seed.

Arms. Per-request admission, the baseline, judges each branch on its own and rejects it if that branch alone would miss the deadline. Task-level admission judges the task once and admits or rejects all of its branches together.

Metrics. The primary metric is tasks per GPU-hour, counting only tasks whose branches all completed by the deadline. The guard metric is the SLO attainment of background interactive traffic, reported as the drop relative to per-request admission. The measured run's manifest digest is 186d2d4232f9c03b0f617fa010781677eb5e3515bdbc71e6747f241ef2884d8e.

3 Results

Table 1: Tasks completed by their deadline per GPU-hour on one GPU, task-level vs per-request admission.

Load	Per-request admission	Task-level admission	Gain
0.8×	24,780	25,728	+3.8%
1.0×	n/a	n/a	+11.6%
1.2×	21,900	27,372	+25%
1.5×	16,320	24,540	+50%

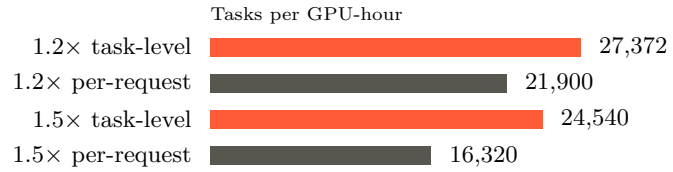


Figure 1: Tasks per GPU-hour at the loads where both values were reported.

Task-level admission completes more tasks per GPU-hour at every load. The gain grows with load, from +3.8% at 0.8× (25,728 against 24,780) and +11.6% at 1.0× to +25% at 1.2× (27,372 against 21,900) and +50% at 1.5× (24,540 against 16,320).

Background interactive traffic is unaffected. Its SLO attainment drops by 0 points relative to per-request admission at all loads.

Streaming-credit variant (negative result). A companion streaming-credit variant was measured in the same campaign and failed its pre-registered gate. It is not part of the result above.

4 Admission at the task boundary

A multi-branch task has an all-or-nothing completion condition. If a task has k branches and one misses its deadline, work spent on the other k−1 branches does not produce a completed task. Per-request admission sees only branch-local feasibility and can therefore create partial admission. Task-level admission evaluates the grouped demand before any branch enters the engine and either reserves the task as a unit or rejects it before GPU work begins.

This changes the optimization target from accepted requests to completed tasks. It does not require changing model execution or vLLM scheduling. Both arms use the same engine, GPU, model, arrivals, and branch work; only the semantic unit presented to admission changes. The background stream remains an independent guard workload so a task gain cannot be purchased by silently displacing ordinary interactive requests.

5 Load-dependent behavior

At 0.8× load, capacity is loose enough that both policies finish most feasible work and the gain is 3.8%. As offered load reaches and exceeds calibrated capacity, partial admission

Table 2: Semantic difference between the two admission units.

Property	Per-request	Task-level
Decision unit	One branch	All branches of one task
Partial work possible	Yes	No
Primary objective	Accepted requests	Deadline-complete tasks
Background guard	Measured	Measured

becomes more expensive. The gain rises to 11.6% at 1.0×, 25% at 1.2×, and 50% at 1.5×. The monotonic increase is consistent with wasted partial tasks becoming a larger share of GPU time near saturation.

Table 3: Absolute task throughput where both arm values are present.

Load	Baseline tasks/GPU-h	Task-level tasks/GPU-h	Additional completed tasks
0.8×	24,780	25,728	948
1.2×	21,900	27,372	5,472
1.5×	16,320	24,540	8,220

The background SLO delta is 0 points at every load. This guard matters most at 1.5×, where a throughput-only policy could improve the task metric by starving the background stream. The measured gain instead comes from spending fewer cycles on tasks that cannot finish.

6 Ablation and evidence boundary

The companion streaming-credit variant fails its preregistered gate and is retained as a negative result. That outcome prevents attributing the task-level gain to any generic notion of more aggressive admission. The passing mechanism is specifically grouped all-or-nothing admission under the measured deadline model.

The evidence covers one GPU, Qwen3-0.6B, one seed, one branch-count distribution, and four offered loads. It does not establish the same ratios for larger models, different fan-out, variable tool latency, multi-GPU serving, or admission estimates with model error. The 1.0× row reports only the measured percentage gain because the frozen summary does not expose both absolute counts; no values are reconstructed.

7 Reproducibility

A reproduction must replay the identical open-loop arrivals through both arms, preserve task-to-branch membership and deadlines, and count a task only when every branch completes before its deadline. It must charge the full GPU window, retain rejected and partial-task accounting, and report background SLO separately from task throughput.

The measured artifact is identified by manifest digest 186d2d4232f9c03b0f617fa010781677eb5e3515bdbc71e6747f241ef2884d8e. The load calibration, one-seed scope, Qwen3-0.6B revision, GPU platform, and failed streaming-credit arm are part of the result rather than optional metadata.

8 Conclusion

For agent workloads that fan out into parallel requests, admitting whole tasks rather than single requests delivered +25% completed tasks per GPU-hour at 1.2× load and +50% at 1.5× on one GPU, with 0 points of background SLO loss. The measurement covers one seed on a 0.6B model, and the companion streaming-credit variant failed its gate. For teams running agents on shared inference capacity, the result suggests that the same GPUs finish more tasks on time when the router decides at the level of the task.

References

[1] W. Kwon et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. SOSP 2023.

[2] Qwen Team. Qwen3 Technical Report. arXiv preprint, 2025.

[3] A. Gujarati et al. Serving DNNs like Clockwork: Performance Predictability from the Bottom Up. OSDI 2020.