

Large-Scale LLM Inference with Heterogeneous Workloads: Prefill-Decode Contention and Asymptotically Optimal Control

Ruihan Lin^a, Zean Han^b, Zezhen Ding^c, Jiheng Zhang^d

Department of Industrial Engineering and Decision Analytics, The Hong Kong University of Science and Technology
^arlinah@connect.ust.hk, ^bzhanax@connect.ust.hk, ^czdingah@connect.ust.hk, ^djiheng@ust.hk

Abstract.

Large Language Models (LLMs) are rapidly becoming critical infrastructure for enterprise applications, driving unprecedented demand for GPU-based inference services. A key operational challenge arises from the two-phase nature of LLM inference: a compute-intensive *prefill* phase that processes user input, followed by a memory-bound *decode* phase that generates output tokens. When these phases share GPU resources, prefill tasks throttle the processing speed of concurrent decodes, creating state-dependent contention. This contention is further complicated by workload heterogeneity, as different applications exhibit vastly different input and output lengths. We develop a stochastic control framework for scheduling heterogeneous LLM workloads across large GPU clusters. We formulate LLM inference as a multiclass many-server queueing network with state-dependent service rates, grounded in empirical iteration-time measurements. We analyze the fluid approximation of this system and solve steady-state linear programs that characterize optimal resource allocation. We design gate-and-route policies that regulate prefill admission and decode routing, and prove that they are asymptotically optimal in the many-GPU limit under both bundled and separate token-pricing schemes. We further extend the framework to incorporate Service Level Indicators (SLIs) such as latency and fairness, providing a general approach to constrained scheduling. Numerical experiments calibrated to empirical iteration-time data demonstrate that our policies outperform standard serving heuristics.

Key words: Stochastic control, Queueing network, Large language models, Revenue management

1. Introduction

Large Language Models (LLMs) have emerged as a foundational technology in contemporary artificial intelligence, leading to a substantial increase in computational demand (Zhao et al. 2025). To accommodate this growth, production-scale infrastructures have expanded correspondingly, frequently requiring the concurrent utilization of thousands of GPUs to sustain worldwide inference workloads (Kwon et al. 2023, Aminabadi et al. 2022). Since commercial LLM services predominantly adopt per-token pricing (OpenAI 2025, Google Cloud 2025, Anthropic 2025, Google 2025, DeepSeek 2025), revenue is closely tied to the volume and composition of tokens processed, making efficient resource allocation critical for both profitability and user experience.

Prefill–decode contention. A distinguishing feature of LLM inference is that each request proceeds in two stages: a *prefill* stage, in which the model processes the user’s input prompt, followed by a *decode* stage,

in which it generates output tokens autoregressively. Modern serving systems batch multiple requests on the same GPU to exploit the GPU’s parallel processing capability, but these two stages interact in nontrivial ways when sharing GPU resources. Prefill is compute-intensive and, when present in a batch, dominates the iteration time and thereby throttles the processing speed of co-located decode tasks; decode, in contrast, is memory-bound and proceeds faster when running alone (Kwon et al. 2023, Agrawal et al. 2024). Moreover, iteration time grows roughly linearly in the amount of prefill work, so adding a second prefill to a batch would nearly double the iteration time without improving parallelism. For this reason, practical systems process at most one prefill per GPU at a time. This *prefill-decode contention* creates a fundamental scheduling tension: admitting more prefills increases the rate at which new requests enter the system but slows down all concurrent decodes, and the scheduler must carefully balance how many GPUs are devoted to prefill versus decode-only operation.

Workload heterogeneity. Real-world LLM services do not see a single homogeneous stream of requests. They serve a mix of applications, such as summarization, creative writing, and question answering, which differ widely in typical input and output lengths (Sun et al. 2024, Zheng et al. 2024, Zhao et al. 2024). For example, summarization tasks average over 1,000 input tokens with moderate output, while creative writing requires fewer than 100 input tokens yet generates over 900 output tokens on average (see Table EC.1 in the e-companion for detailed statistics). This heterogeneity amplifies the scheduling challenge: a class with long prefills and short decodes (e.g., summarization) consumes GPU time during prefill but releases decode capacity quickly, whereas a class with short prefills and long decodes (e.g., creative writing) admits quickly but occupies decode slots for extended periods. Maximizing revenue requires serving an appropriate mix of both to balance pipeline utilization. Consequently, this multiclass resource allocation problem is not well served by simple static priority rules such as first-come-first-served or shortest-job-first; instead, it calls for finer control over how resources are distributed across classes.

This paper addresses the following question: ***How should a large-scale LLM inference system jointly control admission and scheduling across multiple request classes to maximize token-based revenue while respecting service level indicators (SLIs)?*** Answering this question requires overcoming several challenges. First, the state-dependent service rates arising from prefill-decode contention create analytical difficulties that preclude direct application of standard queueing results. Second, heterogeneous workloads induce a multiclass resource allocation problem where the optimal policy depends on the composition of the workload mix. Third, practical systems must balance revenue maximization against SLIs such as latency and fairness across request classes.

Our approach. We model the system as a multiclass many-server queueing network where each GPU operates in one of two modes: *mixed* (running one prefill alongside decodes) or *solo* (decode-only). Service rates are state-dependent, specifically, decodes run slower in mixed mode due to intensive computation

of prefill, and we derive these rates from empirical iteration-time measurements in Section 2.2, capturing essential GPU physics in a tractable analytical framework.

Because production LLM clusters typically comprise hundreds to thousands of GPUs, we study the system through fluid approximation in the many-server regime, which is a standard and well-established approach in operations research for analyzing large-scale stochastic networks. In this scaling, stochastic fluctuations average out and the system trajectory converges to a deterministic limit, yielding both analytical tractability and high accuracy at scale. The fluid model reduces to a steady-state linear program (LP) whose solution prescribes how to partition cluster capacity between mixed and solo modes and how to distribute workload across request classes.

We translate the fluid solution into implementable control via a *gate-and-route* architecture: a *prefill gate* regulates admission by tracking class-level occupancy targets from the LP, and a *decode router* directs completed prefills to available GPU slots. This decomposition into static planning (solving the LP) and dynamic control (enforcing LP targets) is central to our design and underpins the asymptotic optimality results we establish.

Our contributions. We make the following contributions.

1. *Multiclass many-server model with prefill-decode contention.* We develop a queueing network where each GPU operates in mixed or solo mode, with state-dependent service rates that capture how prefill operations throttle co-located decodes. The service-rate parameters are calibrated from controlled experiments on production hardware (A100 GPUs) to closely reflect real system behavior.
2. *Fluid approximation and LP-based planning.* We establish convergence of the scaled stochastic system to a deterministic fluid limit in the many-server regime. The steady-state analysis reduces to a linear program that prescribes optimal capacity partitioning between mixed and solo modes and class-level occupancy targets. This LP formulation provides analytical tractability and serves as the foundation for the control policies we develop.
3. *Asymptotically optimal control policies.* We translate the LP solution into implementable gate-and-route policies and prove their asymptotic optimality. Under bundled charging (revenue credited upon completion), an occupancy-tracking gate suffices; under separate charging (revenue credited after each phase), a priority-based policy is needed to counteract incentive distortions that would otherwise shift congestion downstream. The framework naturally extends to incorporate service-level indicators (fairness, latency) as constraints or penalties, and we show that enforcing fairness at the prefill stage is more revenue-costly than at the decode stage. Simulation experiments confirm that per-GPU revenue converges to the fluid optimum as the cluster scales and that our policies consistently outperform industry-standard heuristics.

1.1. Literature Review

Efficient LLM inference serving has become a critical systems challenge as production deployments face mounting demands for throughput, latency, and resource efficiency. Early systems work concentrated on improving performance on single GPUs through architectural innovations. Iteration-level batching made it practical to schedule at token granularity (Yu et al. 2022), while paged attention reduced KV-cache fragmentation and enabled high utilization under dynamic workloads (Kwon et al. 2023). Chunked prefill further enabled interleaving prefill chunks with ongoing decodes (Agrawal et al. 2024).

As deployments scaled from single-device prototypes to large-scale production clusters, designers began allowing the prefill and decode stages to be executed on different GPUs to better match the compute-bound prefill phase with the memory-bandwidth-bound decode phase (Patel et al. 2024, Zhong et al. 2024). This transition from single-GPU optimization to cluster-scale orchestration raises new questions that go beyond engineering heuristics: how should a service provider allocate GPU capacity and route requests across GPUs under heterogeneous workloads to maximize long-run revenue while considering customized service-level indicators? Existing serving stacks acknowledge these trade-offs but do not yet provide a formal framework for revenue-driven resource allocation at scale.

This operational challenge has attracted growing attention from the operations research (OR) and operations management (OM) communities. Multiple survey articles underscore the growing intersection of AI/LLM and OR/OM, covering foundational frameworks for AI and operations (Dai and Swaminathan 2026), LLM foundations and algorithmic innovations (Zhao et al. 2025), efficient inference techniques (Zhou et al. 2024), serving system advances and opportunities (Li et al. 2024), and the intersection of queueing theory and predictive scheduling for LLMs (Mitzenmacher and Shahout 2025). Within this broad landscape, research efforts unfold along two complementary directions. On one hand, researchers are exploring how LLMs can enhance traditional OR/OM workflows. For instance, Simchi-Levi et al. (2025) leverage generative AI to democratize optimization, Huang et al. (2025) train LLMs for automated optimization modeling, and Simchi-Levi et al. (2026) apply LLMs to supply chain decision-making. On the other hand, and more directly relevant to our work, is the growing body of research that applies OR/OM methodologies to improve LLM inference itself. Within this OR-for-LLM strand, efforts can be further categorized into two areas: improving the quality of LLM outputs, such as aggregating responses via higher-order information beyond majority voting (Ai et al. 2025) or applying online learning frameworks to address missing covariates in pre-trained AI model assisted decision-making (Hu and Simchi-Levi 2025), and accelerating LLM inference through principled resource allocation and scheduling, which is the central focus of the present paper.

In the context of accelerating LLM inference, a recent strand of work employs competitive analysis from the online algorithms community to formalize LLM serving. Zhou and coauthors model KV-cache-constrained batching and benchmark online schedulers against a hindsight integer program. They show that under fully adversarial arrivals, no deterministic online algorithm achieves a constant competitive ratio; under additional

assumptions, they provide a polynomial-time scheduler with a constant competitive ratio (Jaillet et al. 2025). Follow-up work obtains constant-competitive policies when heterogeneous prefill/decode lengths are modeled directly (Wang et al. 2025) and logarithmic-competitive guarantees under interval predictions of decode length via an adaptive policy (A_min) contrasted with a conservative upper-bound policy (A_max) (Chen et al. 2025).

Meanwhile, the advancement of queueing theory provides an analytical framework for stochastic systems and control design. In many-server settings, fluid approximations provide tractable first-order descriptions for capacity planning and performance analysis (Whitt 2006, Zhang 2013), including accuracy guarantees for sizing under impatience (Bassamboo and Randhawa 2010) and asymptotically optimal scheduling structures for multiclass systems with abandonment (Atar et al. 2010, Long et al. 2020, 2024). Subsequent work enriches these models by exploiting within-queue heterogeneity, allowing dependence between service requirements and patience (Bassamboo and Randhawa 2016, Wu et al. 2019), and analyzing state-dependent service rates and slowdowns (Dong et al. 2015). In parallel, delay estimation and information-sharing frameworks have been developed to manage latency considerations in complex service systems (Ibrahim and Whitt 2009, Ibrahim 2018). Finally, the literature addresses methodological concerns such as robustness to model/input uncertainty in simulation (Lam 2016, Ghosh and Lam 2019), alongside control-oriented heavy-traffic perspectives on dynamic admission/sequencing (Harrison and Zeevi 2004, Ata 2006) and state-space-collapse analyses in parallel-server networks (Dai and Tezcan 2011).

Applying these queueing tools to LLM inference, there are growing contemporary theoretical works that employ stochastic modeling and fluid approximations. Ao et al. (2025) model KV-memory growth and batch-time linearity and propose threshold-based policies (WAIT and Nested WAIT) that approach fluid-optimal throughput on a single GPU while keeping latency and time-to-first-token bounded; their analysis highlights how batching composition and memory constraints jointly drive performance. Complementary results argue that work-conserving rules achieve optimal throughput under simplified token-time abstractions (Li et al. 2025), and empirical iteration-time models justify treating per-iteration latency as a piecewise linear function of tokens advanced (Li et al. 2025). These works provide valuable theoretical foundations for understanding single-GPU scheduling dynamics.

In contrast, our work focuses on cluster-scale serving with multiple GPUs, which introduces distinct modeling considerations. First, in disaggregated multi-GPU deployments where decode tasks can be isolated on dedicated servers, the performance characteristics differ from single-GPU mixed batching: decode throughput becomes primarily memory-bandwidth bound rather than compute-bound, exhibiting minimal degradation with batch size. This motivates the piecewise characterization of prefill and decode service rates described by Li et al. (2025), which aligns more closely with the hardware-level asymmetry between the two phases. Second, cluster-scale optimization often calls for a many-server framework where capacity scales by increasing the number of GPUs ($n \rightarrow \infty$) while maintaining constant per-device performance, as

opposed to scaling the service rate of a single GPU to infinity as in traditional heavy-traffic analysis. Finally, our emphasis on revenue maximization under heterogeneous workloads and customizable SLI constraints complements the throughput-centric perspective of prior work, offering service providers a framework for balancing commercial objectives with quality-of-service guarantees.

To realize this vision, we construct a multiclass many-server fluid approximation where service rates are derived directly from an empirical token-time law. By modeling job classes via expected rather than exact lengths, our framework remains robust to per-request mispredictions while capturing the essential system dynamic: admitting prefills forces co-located decodes into a slower mixed mode. The resulting steady-state optimization problem parametrizes a two-level control policy, consisting of a prefill gate and a decode router, that provably achieves asymptotic optimality in the many-GPU limit.

1.2. Organization

The remainder of the paper is organized as follows. Section 2 introduces the iteration-time abstraction, the multiclass many-server stochastic network with mixed and solo decode modes, and the bundled revenue objective. Section 3 develops the many-GPU fluid limit, formulates the steady-state linear program (LP) over per-GPU occupancies, and establishes structural properties such as decode-buffer elimination. Section 4 constructs the occupancy-anchored gate-and-route policy, proves its asymptotic optimality under bundled pricing, and then analyzes the separate prefill/decode charging scheme together with the corresponding prioritize-and-route policy. Section 5 extends the fluid LP to an SLI-aware formulation that integrates fairness and latency proxies, and shows convergence of the stochastic occupancies to the SLI-optimal fluid solution. Section 6 presents numerical experiments with calibrated parameters and synthetic multiclass workloads, validating the fluid predictions and exploring revenue–SLO tradeoffs. Section 7 concludes and discusses directions for extending the framework. Technical proofs and additional lemmas are collected in the electronic companion.

2. Problem Formulation

This section develops the queueing model in three parts. Section 2.1 provides background on modern LLM inference systems. Section 2.2 characterizes GPU iteration time. Section 2.3 derives the resulting service rates and embeds them into a many-server stochastic network, specifying the state, flows, admissible controls, capacity coupling, and revenue objectives.

2.1. Preliminaries: Modern LLM Inference Systems

Two stages of LLM inference. We formally introduce the two stages in LLM inference, namely *prefill* and *decode*. In the *prefill* stage, the model reads the entire input prompt once and, layer by layer, builds an internal representation of all input tokens. Implementations store per-token intermediate states in a key–value (KV) cache so that, in the subsequent *decode* stage, the model can generate output tokens one by one by

attending to this cache instead of recomputing all past KV values (Shazeer 2019, Dai et al. 2019, Kwon et al. 2023). Empirical studies show that these two stages stress the hardware differently: prefill behaves like a large, single-shot matrix computation, while decode repeatedly accesses and extends the KV cache and is more sensitive to memory traffic (Kwon et al. 2023, Agrawal et al. 2024). This asymmetry is the root cause of the scheduling trade-offs that we model.

Continuous batching. State-of-the-art serving systems do not run one request at a time per GPU. Instead, they use *continuous* or *iteration-level* batching: in each short iteration, the GPU advances many active requests by one output token in parallel, then repeats this process for the next token (Yu et al. 2022, Kwon et al. 2023). In practice, these systems impose a fixed upper bound on how many requests can be batched on a GPU, chosen for engineering reasons such as avoiding out-of-memory errors and reducing dynamic memory management overhead (Kwon et al. 2023, Agrawal et al. 2024). We denote this constant by B .

Chunked prefill and GPU modes. Long prompts make it inefficient to run prefill in isolation. Recent systems therefore *chunk* prefills into smaller pieces (with each C tokens) that can be interleaved with decode work on the same GPU (Agrawal et al. 2024). Measurements show that once a prefill chunk is present in an iteration, it tends to dominate the iteration time; running additional prefills in parallel on the same GPU brings no extra benefit because compute capacity is already saturated (Agrawal et al. 2024). Following this evidence, we adopt the standard assumption that each GPU runs at most one prefill chunk at a time.

Figure 1 summarizes the resulting GPU-level architecture, which resembles key elements of systems such as vLLM and Sarathi-Serve (Kwon et al. 2023, Agrawal et al. 2024). New requests from all classes first enter *prefill queues*. A host-side *Prefill Scheduler* selects some queued requests and starts their prefill stage on GPUs that are not currently running another prefill. While a prefill is in progress, that GPU is in a *mixed* mode: one slot is occupied by the prefill chunk, and the remaining at most $B - 1$ slots can be used to advance decodes from decode-ready requests of various classes. When the prefill finishes, the request moves into the *decode buffer*, representing the set of requests ready to decode, with their KV caches typically kept resident in GPU memory. The GPU then returns to a *solo* decode mode, where all activated slots are devoted to decode. A separate *Decode Scheduler* continuously fills empty slots on both mixed and solo GPUs from the multiclass decode buffer. Because a mixed-mode GPU shares its compute and memory bandwidth between a large prefill and several decodes, the per-token progress of those decodes is slower than on a solo GPU; admitting more prefills thus increases the rate at which new requests enter decode but slows down decodes sharing the same GPU.

2.2. Token Processing on GPUs

To build a tractable stochastic model, we first characterize GPU iteration time. In each iteration, the GPU processes a batch that may contain both prefill and decode tasks: all decode tasks in the batch each generate one output token, and if a prefill chunk is present, the GPU processes one chunk of its input tokens. The iteration time is defined as the duration required to complete this batch processing step.

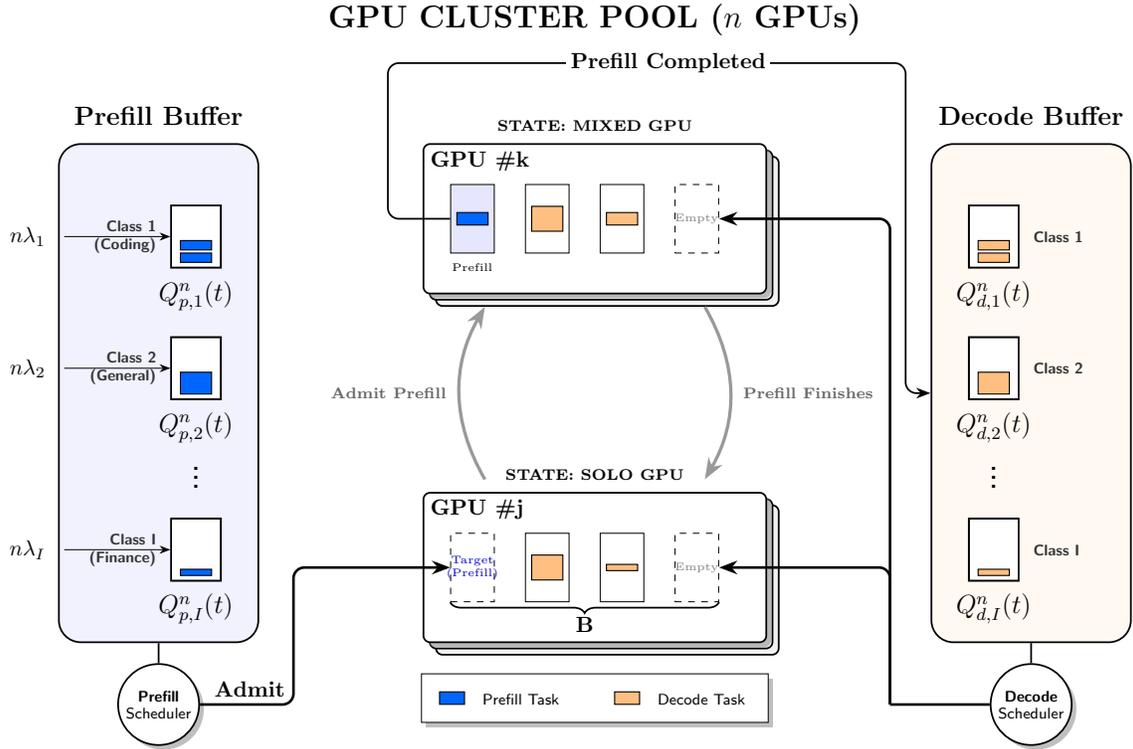


Figure 1 Schematic of the Dynamic GPU Scheduling Architecture. The system manages a cluster of n GPUs with batch size B . GPUs transition between the Solo State (decode-only) and Mixed State (one prefill + decodes) based on assignments from the Prefill Scheduler. Completed prefills enter a virtual Decode Buffer, from which the Decode Scheduler populates available slots in either state.

Iteration time. A key empirical finding, documented by Li et al. (2025), is that iteration time depends on the total number of tokens processed in a batch:

$$\tau(b') = c + a \cdot \max\{0, b' - b_0\}. \quad (1)$$

Here, b' denotes the effective token count for the iteration, equal to the prefill chunk size plus the number of concurrent decode tasks. The constant $c > 0$ captures fixed overheads (e.g., kernel launches), $a > 0$ is the marginal cost per token, and $b_0 \geq 0$ is a threshold below which overheads dominate. This formula captures two operating regimes:

- **Decode-only iteration:** In a decode-only batch, the effective token count b' equals the batch size, which is typically smaller than b_0 . The max term in Equation (1) then vanishes, and the iteration time reduces to a near-constant value

$$\tau_{\text{solo}} = \tau(b') = c, \quad \text{for } b' \leq b_0. \quad (2)$$

- **Mixed-batch iteration:** When a batch includes a prefill chunk of size C , the chunk dominates the effective token count (i.e., $b' \approx C$). For practical chunk sizes where $C > b_0$, iteration time grows linearly

in C . This is consistent with empirical observations from Sarathi-Serve, where decode iteration time is largely independent of batch size but prefill iteration time scales linearly with chunk size (Agrawal et al. 2024).

Our experiments in Section 6.1 confirm both regimes: Figure 3 shows that iteration time remains nearly flat for small batch sizes and grows linearly in C once a prefill chunk is present.

Since practical chunk sizes typically satisfy $C \geq b_0$, we adopt the linear form for mixed-batch iteration time:

$$\tau_{\text{mix}}(C) = \alpha + \beta C, \quad \text{where } \alpha := c - ab_0 \text{ and } \beta := a > 0. \quad (3)$$

This two-regime abstraction captures the key prefill–decode interaction while remaining analytically tractable.

2.3. The Stochastic Model

We now embed the iteration-time characterization from Section 2.2 into a multiclass many-server stochastic network. The model is indexed by the number of GPUs $n \in \mathbb{N}$.

Primitives and service parameters. Requests belong to a finite set of classes $\mathcal{I} := \{1, \dots, I\}$. A class- i request is characterized by its representative prompt length P_i and decode length D_i (in tokens). The system consists of n homogeneous GPUs. Each GPU can host at most $B \in \mathbb{N}$ concurrent decode streams and at most one prefill at a time. Prefill is executed in fixed-size chunks of $C > 0$ tokens per iteration.

Recall from Section 2.2 that the iteration time is $\tau_{\text{solo}} = c$ in decode-only mode and $\tau_{\text{mix}}(C) = \alpha + \beta C$ when a prefill chunk is present. For notational convenience, write $\tau := \tau_{\text{mix}}(C)$ for the mixed-iteration time. Service rates are derived as follows.

- **Prefill rate.** A prefill job of length P_i tokens advances C tokens per iteration, each taking time τ . Completing the prefill requires (P_i/C) iterations, so the mean service time is $(P_i/C)\tau$ and the rate is

$$\mu_{p,i} = \frac{C}{P_i \tau}.$$

- **Mixed decode rate.** In mixed mode, a decode job produces one token per iteration. A class- i job needs D_i tokens, so the mean service time is $D_i\tau$ and the rate is

$$\mu_{m,i} = \frac{1}{D_i \tau}.$$

- **Solo decode rate.** In decode-only mode, each token takes τ_{solo} seconds. Defining $\gamma := 1/\tau_{\text{solo}}$ to be the token generation rate per slot, the mean service time for D_i tokens is D_i/γ and the rate is

$$\mu_{s,i} = \frac{\gamma}{D_i}.$$

We collect these rates as

$$\mu_{p,i} = \frac{C}{P_i \tau}, \quad \mu_{m,i} = \frac{1}{D_i \tau}, \quad \mu_{s,i} = \frac{\gamma}{D_i}. \quad (4)$$

For analytical tractability, we model prefill, mixed decode, and solo decode service times as independent exponential random variables with rates $\mu_{p,i}$, $\mu_{m,i}$, and $\mu_{s,i}$, respectively, for each class $i \in \mathcal{I}$.

We assume Poisson arrivals: in the n th system, class- i arrivals form a Poisson process with rate $\lambda_i^n := n\lambda_i$, where $\lambda_i > 0$ is the nominal arrival rate per GPU, so the total offered load grows proportionally with n . Customers are impatient in both the prefill and decode queues: any class- i job that is waiting is endowed with an independent exponential patience time with rate $\theta_i \geq 0$. Interarrival times, service times, and patience times are assumed mutually independent across all jobs and classes.

State and control processes. Fix $n \in \mathbb{N}$. For each class $i \in \mathcal{I}$ and time $t \geq 0$, denote by $Q_{p,i}^n(t)$ the number of class- i jobs waiting for prefill, by $X_i^n(t)$ the number in prefill service, by $Q_{d,i}^n(t)$ the number waiting for decode (prefill completed), and by $Y_{m,i}^n(t)$ and $Y_{s,i}^n(t)$ the numbers in mixed and solo decode, respectively. These processes are right-continuous, integer-valued, and change by unit jumps when individual jobs enter or leave the corresponding stage. The total class- i content in prefill and decode is

$$Z_{p,i}^n(t) := Q_{p,i}^n(t) + X_i^n(t), \quad Z_{d,i}^n(t) := Q_{d,i}^n(t) + Y_{m,i}^n(t) + Y_{s,i}^n(t). \quad (5)$$

The cumulative primitive counting processes are defined as follows. Let $A_i^n(t)$ be the total number of exogenous arrivals of class- i jobs by time t . Let $B_{p,i}^n(t)$ and $B_{d,i}^n(t)$ denote the total abandonments from the prefill and decode queues of class i . Let $S_{p,i}^n(t)$ be the total prefill completions, and $S_{d,m,i}^n(t)$ and $S_{d,s,i}^n(t)$ the total mixed and solo decode completions; the total decode completions are

$$S_{d,i}^n(t) := S_{d,m,i}^n(t) + S_{d,s,i}^n(t). \quad (6)$$

These counting processes admit a standard random time-change representation. Let

$$\{N_{A,i}, N_{B_{p,i}}, N_{B_{d,i}}, N_{p,i}, N_{d,m,i}, N_{d,s,i}\}_{i \in \mathcal{I}}$$

be mutually independent unit-rate Poisson processes. Then, for each $i \in \mathcal{I}$ and $t \geq 0$,

$$A_i^n(t) = N_{A,i}(\lambda_i^n t), \quad B_{p,i}^n(t) = N_{B_{p,i}}\left(\int_0^t \theta_i Q_{p,i}^n(s) ds\right), \quad (7)$$

$$B_{d,i}^n(t) = N_{B_{d,i}}\left(\int_0^t \theta_i Q_{d,i}^n(s) ds\right), \quad S_{p,i}^n(t) = N_{p,i}\left(\int_0^t \mu_{p,i} X_i^n(s) ds\right), \quad (8)$$

$$S_{d,m,i}^n(t) = N_{d,m,i}\left(\int_0^t \mu_{m,i} Y_{m,i}^n(s) ds\right), \quad S_{d,s,i}^n(t) = N_{d,s,i}\left(\int_0^t \mu_{s,i} Y_{s,i}^n(s) ds\right). \quad (9)$$

Equations (7)–(9) state that each cumulative count is driven by a unit-rate Poisson process with the corresponding integrated intensity.

Scheduling decisions are encoded through cumulative control processes. For each class i , let $U_{p,i}^n(t)$ be the number of jobs admitted into prefill service by time t , and $U_{d,m,i}^n(t)$ and $U_{d,s,i}^n(t)$ the numbers admitted into mixed and solo decode. Mode switches between decode submodes are counted by

$$M_{s \rightarrow m,i}^n(t) \text{ and } M_{m \rightarrow s,i}^n(t), \quad (10)$$

the cumulative numbers of class- i decodes switched from solo to mixed and from mixed to solo by time t . These mode-switch processes are endogenous: they have no external Poisson clocks and are induced by changes in prefill activity on each GPU. In particular, solo-to-mixed switches occur when a prefill is admitted to a GPU that was previously in pure decode mode, and mixed-to-solo switches occur when that prefill completes; these transitions are structural consequences of the prefill dynamics rather than direct control actions of the scheduling policy.

For later use, define the aggregate in-service counts

$$X^n(t) := \sum_i X_i^n(t), \quad Y_m^n(t) := \sum_i Y_{m,i}^n(t), \quad Y_s^n(t) := \sum_i Y_{s,i}^n(t), \quad (11)$$

and similarly $M_{s \rightarrow m}^n(t) := \sum_i M_{s \rightarrow m,i}^n(t)$ and $M_{m \rightarrow s}^n(t) := \sum_i M_{m \rightarrow s,i}^n(t)$. The per-GPU physical constraints imply

$$0 \leq X^n(t) \leq n, \quad (12)$$

$$0 \leq Y_m^n(t) \leq (B-1) X^n(t), \quad (13)$$

$$0 \leq Y_s^n(t) \leq B(n - X^n(t)). \quad (14)$$

Equation (12) enforces at most one prefill per GPU, while (13)–(14) bound mixed and solo decodes according to whether a GPU is running a prefill.

Admissible policies. We now formalize the notion of a policy. Let

$$\pi^n := \left(Q_p^n, Q_d^n, X^n, Y_m^n, Y_s^n, A^n, B_p^n, B_d^n, S_p^n, S_{d,m}^n, S_{d,s}^n, U_p^n, U_{d,m}^n, U_{d,s}^n, M_{s \rightarrow m}^n, M_{m \rightarrow s}^n \right)$$

denote the collection of all non-primitive processes in the n th system (state, cumulative flows, and control processes), where each symbol stands for the vector over classes $i \in \mathcal{I}$. Let Π^n denote the set of policies that satisfy the admissibility conditions below:

(i) the resulting state processes satisfy the capacity constraints (12)–(14) and balance equations (15)–(19) for all $t \geq 0$;

(ii) the policy is *event-driven*, i.e., each control process $U_{\cdot,i}^n(t)$ can change only at arrival epochs, abandonment epochs, service-completion epochs, or at $t = 0$;

(iii) within each class, prefill and decode queues are served in first-come-first-served order, and service is non-preemptive.

We say that any $\pi^n \in \Pi^n$ is an *admissible policy*, under which the state processes are then uniquely determined from the primitives and the controls via the balance equations (15)–(19).

Balance equations. Under any policy $\pi^n \in \Pi^n$, the state and cumulative processes satisfy the following flow-balance identities for all $i \in \mathcal{I}$ and $t \geq 0$:

$$Q_{p,i}^n(t) = Q_{p,i}^n(0) + A_i^n(t) - U_{p,i}^n(t) - B_{p,i}^n(t), \quad (15)$$

$$X_i^n(t) = X_i^n(0) + U_{p,i}^n(t) - S_{p,i}^n(t), \quad (16)$$

$$Q_{d,i}^n(t) = Q_{d,i}^n(0) + S_{p,i}^n(t) - U_{d,m,i}^n(t) - U_{d,s,i}^n(t) - B_{d,i}^n(t), \quad (17)$$

$$Y_{m,i}^n(t) = Y_{m,i}^n(0) + U_{d,m,i}^n(t) - S_{d,m,i}^n(t) + M_{s \rightarrow m,i}^n(t) - M_{m \rightarrow s,i}^n(t), \quad (18)$$

$$Y_{s,i}^n(t) = Y_{s,i}^n(0) + U_{d,s,i}^n(t) - S_{d,s,i}^n(t) + M_{m \rightarrow s,i}^n(t) - M_{s \rightarrow m,i}^n(t). \quad (19)$$

Equation (15) says that the prefill queue-length equals its initial content plus arrivals, minus admissions and abandonments. Equation (16) tracks prefill jobs in service as admissions minus completions. Equation (17) balances the decode queue as completed prefills minus admissions into decode and abandonments. Equations (18)–(19) track mixed and solo decodes as admissions minus completions, plus net inflow from mode switches.

Adding (18) and (19) eliminates the mode-switch terms and yields

$$Y_{m,i}^n(t) + Y_{s,i}^n(t) = Y_{m,i}^n(0) + Y_{s,i}^n(0) + U_{d,m,i}^n(t) + U_{d,s,i}^n(t) - S_{d,m,i}^n(t) - S_{d,s,i}^n(t), \quad (20)$$

so mode switches only redistribute ongoing decodes between mixed and solo, without changing their total number.

Revenue models and objective functions. Commercial LLM services predominantly use token-based pricing. We consider two revenue models that differ in when revenue is recognized.

(1) *Bundled charging scheme.* The provider charges a single price per request based on the total number of tokens, and revenue is recognized only when the entire request completes (after decode). For class i ,

$$w_i := c_p P_i + c_d D_i, \quad (21)$$

where $c_p, c_d \geq 0$ are unit prices per prefill and decode token. The per-GPU average reward over $[0, T]$ under policy π^n is

$$R^n(T; \pi^n) := \frac{1}{nT} \mathbb{E}^{\pi^n} \left[\sum_{i=1}^I w_i S_{d,i}^n(T) \right]. \quad (22)$$

Only completed requests contribute to (22); prefill work without decode completion yields no revenue.

(2) *Separate charging scheme.* Alternatively, prefill and decode tokens may be billed and recognized separately. The corresponding per-GPU average reward is

$$\tilde{R}^n(T; \pi^n) := \frac{1}{nT} \mathbb{E}^{\pi^n} \left[\sum_{i=1}^I (c_p P_i S_{p,i}^n(T) + c_d D_i S_{d,i}^n(T)) \right]. \quad (23)$$

Both objectives depend on token throughput but induce different scheduling incentives: in particular, the separate scheme (23) may encourage aggressive prefill admissions to harvest immediate prefill revenue at the expense of downstream decode congestion. Asymptotically optimal control for (22) and (23) will be studied in Section 4.

3. Fluid Approximation and Steady-State Planning

In large-scale LLM deployments, providers typically operate hundreds or thousands of GPUs in parallel. At this scale, the system state is high-dimensional and stochastic, with arrivals, service completions, and abandonments fluctuating across time and devices, making direct stochastic optimization analytically intractable and hard to interpret. We therefore adopt a many-GPU fluid approximation: consider a sequence of systems indexed by the number of GPUs n , scale all queue lengths and in-service counts by $1/n$, and let $n \rightarrow \infty$. In this limit, random fluctuations average out and the network is described by a deterministic set of flow-balance equations and capacity constraints. Steady-state solutions of this fluid model specify per-GPU occupancies, which serve as planning targets for the stochastic control policies in Section 4.

Fluid-scaled processes. For any stochastic process $W^n(t)$ in the n -th system, we define its fluid-scaled version by

$$\bar{W}^n(t) := \frac{1}{n} W^n(t), \quad t \geq 0.$$

We use an overbar to indicate such scaled processes (e.g., $\bar{Q}_{p,i}^n(t)$, $\bar{X}_i^n(t)$, $\bar{Y}_{m,i}^n(t)$), and we write the corresponding lowercase letters (e.g., $q_{p,i}(t)$, $x_i(t)$, $y_{m,i}(t)$) for generic deterministic fluid trajectories that arise as limits of these scaled processes in Section 2.3. These functions satisfy, for all $t \geq 0$ and all $i \in \mathcal{I}$, the flow-balance equations:

$$q_{p,i}(t) = q_{p,i}(0) + a_i(t) - u_{p,i}(t) - b_{p,i}(t), \quad (24)$$

$$x_i(t) = x_i(0) + u_{p,i}(t) - s_{p,i}(t), \quad (25)$$

$$q_{d,i}(t) = q_{d,i}(0) + s_{p,i}(t) - u_{d,m,i}(t) - u_{d,s,i}(t) - b_{d,i}(t), \quad (26)$$

$$y_{m,i}(t) = y_{m,i}(0) + u_{d,m,i}(t) - s_{d,m,i}(t) + m_{s \rightarrow m,i}(t) - m_{m \rightarrow s,i}(t), \quad (27)$$

$$y_{s,i}(t) = y_{s,i}(0) + u_{d,s,i}(t) - s_{d,s,i}(t) + m_{m \rightarrow s,i}(t) - m_{s \rightarrow m,i}(t). \quad (28)$$

Here $q_{p,i}(t)$ and $q_{d,i}(t)$ are the prefill and decode queue contents, $x_i(t)$, $y_{m,i}(t)$ and $y_{s,i}(t)$ are the in-service masses in prefill, mixed decode and solo decode, $u_{p,i}(t)$, $u_{d,m,i}(t)$ and $u_{d,s,i}(t)$ are the cumulative admissions into these stages, $b_{p,i}(t)$ and $b_{d,i}(t)$ are cumulative abandonments, $s_{p,i}(t)$, $s_{d,m,i}(t)$ and $s_{d,s,i}(t)$ are cumulative service completions, and $m_{s \rightarrow m,i}(t)$, $m_{m \rightarrow s,i}(t)$ are cumulative mode switches between decode submodes.

The primitive arrivals, abandonments, and service completions evolve at their mean rates:

$$a_i(t) = \lambda_i t, \quad (29)$$

$$b_{p,i}(t) = \int_0^t \theta_i q_{p,i}(s) ds, \quad b_{d,i}(t) = \int_0^t \theta_i q_{d,i}(s) ds, \quad (30)$$

$$s_{p,i}(t) = \int_0^t \mu_{p,i} x_i(s) ds, \quad (31)$$

$$s_{d,m,i}(t) = \int_0^t \mu_{m,i} y_{m,i}(s) ds, \quad s_{d,s,i}(t) = \int_0^t \mu_{s,i} y_{s,i}(s) ds. \quad (32)$$

Equation (29) gives the deterministic arrival rate, (30) the abandonment flows from the prefill and decode queues, and (31)–(32) the prefill and decode completion flows driven by the in-service masses.

Let

$$x(t) := \sum_i x_i(t), \quad y_m(t) := \sum_i y_{m,i}(t), \quad y_s(t) := \sum_i y_{s,i}(t),$$

and similarly $u_p(t) := \sum_i u_{p,i}(t)$ and $s_p(t) := \sum_i s_{p,i}(t)$. The per-GPU capacity constraints in the fluid model are

$$0 \leq x(t) \leq 1, \quad (33)$$

$$0 \leq y_m(t) \leq (B-1)x(t), \quad (34)$$

$$0 \leq y_s(t) \leq B(1-x(t)), \quad (35)$$

which mirror the prefill and decode caps in (12)–(14) after scaling by n .

Finally, admission feasibility holds at the fluid level. For all $0 \leq s \leq t$ and all $i \in \mathcal{I}$,

$$u_{d,m,i}(s,t) + u_{d,s,i}(s,t) \leq q_{d,i}(s) + s_{p,i}(s,t) - b_{d,i}(s,t), \quad (36)$$

$$u_{p,i}(s,t) \leq q_{p,i}(s) + a_i(s,t) - b_{p,i}(s,t), \quad (37)$$

where $u_{p,i}(s,t) := u_{p,i}(t) - u_{p,i}(s)$ and $a_i(s,t)$, $s_{p,i}(s,t)$, $b_{p,i}(s,t)$, etc. are defined analogously. Inequalities (36)–(37) state that, over any time interval $[s,t]$, admissions into each stage cannot exceed the fluid already present in the corresponding buffer plus the net inflow into that buffer.

ASSUMPTION 1 (Convergence of initial state). *The initial states of fluid-scaled processes converge to a deterministic state: $(\bar{Q}_{p,i}^n(0), \bar{Q}_{d,i}^n(0), \bar{X}_i^n(0), \bar{Y}_{m,i}^n(0), \bar{Y}_{s,i}^n(0)) \Rightarrow (q_{p,i}(0), q_{d,i}(0), x_i(0), y_{m,i}(0), y_{s,i}(0))$;*

THEOREM 1 (Fluid limit). *Fix any finite horizon $T > 0$. The sequence of fluid-scaled stochastic processes*

$$\begin{aligned} \bar{\mathcal{X}}^n(t) := & (\{\bar{Q}_{p,i}^n, \bar{Q}_{d,i}^n, \bar{X}_i^n, \bar{Y}_{m,i}^n, \bar{Y}_{s,i}^n\}_{i \in \mathcal{I}}, \{\bar{S}_{p,i}^n, \bar{S}_{d,m,i}^n, \bar{S}_{d,s,i}^n\}_{i \in \mathcal{I}}, \{\bar{B}_{p,i}^n, \bar{B}_{d,i}^n\}_{i \in \mathcal{I}}, \\ & \{\bar{U}_{p,i}^n, \bar{U}_{d,m,i}^n, \bar{U}_{d,s,i}^n\}_{i \in \mathcal{I}}, \{\bar{M}_{s \rightarrow m,i}^n - \bar{M}_{m \rightarrow s,i}^n\}_{i \in \mathcal{I}}) \end{aligned}$$

is tight in $\mathbb{D}([0, T], \mathbb{R}^d)$ under the Skorokhod J_1 topology, where d is the total dimension of the vector above. Moreover, any subsequential weak limit $\bar{\mathcal{X}}(t)$ is almost surely continuous and, on $[0, T]$, satisfies the fluid model equations (24)–(28) with (29)–(32), the capacity constraints (33)–(35), with initial state given by \mathbf{z}_0 .

Fluid Reward Objectives. Consistent with the revenue models defined in Section 2, we formulate the fluid control objectives for the two charging schemes separately.

(1) *Bundled Objective.* Under the bundled scheme, value is realized only upon request completion. Thus, the objective maximizes the weighted throughput of the decode phase:

$$R(T) := \frac{1}{T} \int_0^T \sum_{i=1}^I w_i (y_{m,i}(\tau) \mu_{m,i} + y_{s,i}(\tau) \mu_{s,i}) d\tau, \quad (38)$$

where $w_i = c_p P_i + c_d D_i$ is the total reward for a completed class- i request. Note that the prefill activity $x_i(\tau)$ contributes to the objective only indirectly by feeding the decode queue.

(2) *Separate Objective.* Under the separate scheme, the system accumulates value continuously as tokens are processed in both phases. The objective becomes:

$$\tilde{R}(T) := \frac{1}{T} \int_0^T \sum_{i=1}^I \left[\underbrace{(c_p P_i)}_{\text{prefill value}} \mu_{p,i} y_{p,i}(\tau) + \underbrace{(c_d D_i)}_{\text{decode value}} (y_{m,i}(\tau) \mu_{m,i} + y_{s,i}(\tau) \mu_{s,i}) \right] d\tau. \quad (39)$$

3.1. Fluid Control Problem

We first consider the bundle objective (the separate charging scheme will be discussed in Section 4.2) and solve a steady-state (fluid) optimization to choose the optimal long-run occupancy shares and the routing of prefilled tasks across mixed and solo decode pools. This formulation intentionally abstracts away the transient effects of stochastic variability in interarrival, prefill, and decode times (and abandonment/patience, if present), and instead enforces constraints only in terms of average arrival and service rates. The solution delivers capacity-splitting targets that will guide the stochastic control policy developed in the next section.

$$\begin{aligned} \max_{\{x_i, y_{m,i}, y_{s,i}, q_{p,i}, q_{d,i}\}} & \sum_{i=1}^I w_i (y_{m,i} \mu_{m,i} + y_{s,i} \mu_{s,i}) \\ \text{s.t.} & \sum_{i=1}^I x_i \leq 1, && \text{(Prefill Capacity)} \\ & \sum_{i=1}^I y_{m,i} \leq (B-1) \sum_{i=1}^I x_i, && \text{(Mixed Decode Capacity)} \\ & \sum_{i=1}^I y_{s,i} \leq B \left(1 - \sum_{i=1}^I x_i\right), && \text{(Solo Decode Capacity)} \\ & \lambda_i - \theta_i q_{p,i} = \mu_{p,i} x_i, \quad \forall i, && \text{(Prefill Flow Balance)} \\ & \mu_{p,i} x_i - \theta_i q_{d,i} = \mu_{m,i} y_{m,i} + \mu_{s,i} y_{s,i}, \quad \forall i, && \text{(Decode Flow Balance)} \\ & x_i, y_{m,i}, y_{s,i}, q_{d,i}, q_{p,i} \geq 0, \quad \forall i. && \text{(Non-negativity)} \end{aligned} \quad (40)$$

The linear program (40) describes the steady-state fluid model for routing prefill and decode work of multiple classes across GPUs with batch size B . The decision variables are long-run, per-GPU averages: x_i is the fraction of time a GPU devotes to class- i prefill; $y_{m,i}$ and $y_{s,i}$ are the average class- i decode occupancies in mixed mode and solo mode; and $q_{p,i}$ and $q_{d,i}$ are the average prefill and decode queue masses.

The capacity constraints in the first three lines of (40) enforce the per-GPU limits: at most one prefill can run on a GPU, and, conditional on whether a prefill is present, at most $B - 1$ (mixed) or B (solo) decodes can be served in parallel. The flow-balance constraints for each class i state that, in steady state, arrivals net of prefill abandonments equal the prefill completion rate, and that prefill completions, net of decode abandonments, are exactly matched by the total decode completion rate. The objective in the first line of (40) maximizes the per-GPU long-run reward by weighting class- i decode completions in mixed and solo modes with $w_i = c_p P_i + c_d D_i$, the total value of a completed request under bundled pricing.

PROPOSITION 1 (Decode-buffer elimination). *Assume $\gamma \tau \geq (B - 1)/B$ (solo decode more efficient). Then the steady-state fluid LP admits an optimal solution with $q_{d,i}^* = 0$ for all i .*

The condition in Proposition 1 is natural: solo decode is strictly more efficient than mixed decode. Since revenue is generated only when a request *finishes* decode, fluid mass held in the decode buffer yields no reward and only delays completions. We prove that any LP solution with $q_{d,i} > 0$ can be improved by moving this backlog upstream while keeping the capacity constraints, which weakly increases the completion rate; hence, at optimality, the decode buffer is empty in steady state.

A key implication for control is that the optimal reward rate is determined by *occupancy proportions*, i.e. how much GPU time is spent on prefill and how decode slots are filled. In a backlogged system, the fluid-optimal plan keeps both prefill and decode fully utilized and fixes the fraction of GPUs running prefills at its target level. The remaining design question is how to route completed prefills between mixed and solo decode so that capacity is saturated without building a decode backlog. This leads to a simple gate-and-route architecture: a *prefill gate* that regulates the target prefill occupancy (and class mix) and a *decode router* that splits work between mixed and solo decodes to keep slots busy while preventing persistent decode queues.

4. Stochastic Control Policy

Building upon the fluid-optimal solution, we now develop an implementable control framework for the stochastic n -GPU system. Our approach operationalizes the fluid prescriptions by decomposing the scheduling problem into two hierarchical stages: a static resource partitioning phase that fixes the cluster configuration, and a dynamic control phase that manages job admission and routing in real time. We first introduce the occupancy-based *Gate-and-Route* policy in Section 4.1, which regulates prefill occupancies and decode routing to attain the fluid-optimal throughput under the bundled objective. We then extend this architecture to the separate charging scheme in Section 4.2, where we develop a counterpart optimal policy and analyze how stage-based revenue recognition fundamentally reshapes scheduling incentives across heterogeneous request classes relative to the bundled objective.

4.1. Bundled Charging Scheme

The design of this policy is inspired by the structural insight from Proposition 1, which reveals that the relative efficiency gain of solo decoding is class-independent. This property suggests a decomposition of the complex scheduling problem. We can *statically* partition the cluster resources to ensure that the aggregate decode capacity is critically loaded in the fluid limit, capable of fully digesting the downstream workload generated by the optimal prefill throughput. With the decode stage dimensioned to clear the traffic naturally, the burden of optimization shifts upstream. Consequently, we focus our *fine-grained dynamic control* on the prefill admission to strictly regulate the job mix and occupancy, allowing the decode stage to operate under a simple work-conserving discipline (FCFS) while still guiding the system toward the fluid-optimal state.

Static Planning Let n be the number of GPUs and B the per-GPU decode stream cap. Take any optimal per-GPU solution of the steady-state fluid LP, denoted by $(x_i^*, y_{m,i}^*, y_{s,i}^*, q_{p,i}^*)_{i \in \mathcal{I}}$.

Fix the number of mixed GPUs as

$$M := \left\lceil n \sum_{i \in \mathcal{I}} x_i^* \right\rceil,$$

choose any subset $\mathcal{G}_{\text{mix}} \subset \{1, \dots, n\}$ with $|\mathcal{G}_{\text{mix}}| = M$, and set $\mathcal{G}_{\text{solo}} := \{1, \dots, n\} \setminus \mathcal{G}_{\text{mix}}$. A GPU $g \in \mathcal{G}_{\text{mix}}$ permanently reserves one slot for prefill (or equivalently, those GPUs prioritize new prefill over decode jobs) and may run at most $(B - 1)$ decodes; a GPU $g \in \mathcal{G}_{\text{solo}}$ never runs prefills and may run at most B decodes.

Dynamic Control Let $X_i^n(t)$ denote the number of class- i prefill tasks currently in service and $Q_{p,i}^n(t)$ denote the prefill queue length at time t .

Upstream (prefill) gate on mixed GPUs. Prefills run only on \mathcal{G}_{mix} . Whenever a mixed GPU $g \in \mathcal{G}_{\text{mix}}$ has its reserved prefill slot idle, identify the set of classes with waiting jobs, $\mathcal{I}_{\text{wait}} = \{i \in \mathcal{I} : Q_{p,i}^n(t^-) > 0\}$. If $\mathcal{I}_{\text{wait}}$ is empty, the slot remains idle. Otherwise, compute the occupancy deviation index for each candidate class:

$$\xi_i(t^-) := \frac{1}{x_i^*} (X_i^n(t^-) - n x_i^*).$$

The scheduler admits the head-of-line prefill of the class i^* that minimizes this deviation:

$$i^* \in \arg \min_{i \in \mathcal{I}_{\text{wait}}} \xi_i(t^-).$$

If there are multiple classes achieving the minimum deviation, ties are broken by selecting the class with the largest queue deviation $\delta_i(t^-) := Q_{p,i}^n(t^-) - Q_{p,i}^\dagger$. Service is non-preemptive and FCFS within each class.

The gate is a negative-feedback rule around the fluid targets x_i^* : classes whose prefill occupancy exceeds their target (large ξ_i) are held back, while under-served classes (smallest ξ_i) are pulled up by being admitted first. Since the total mixed prefill capacity $\sum_i x_i^*$ is fixed by the static planning, not all classes can be above target at once, and repeatedly correcting the most deviated class keeps the long-run average occupancies fluctuating in a small neighborhood of the fluid-optimal levels.

Downstream (decode) routing. Maintain a single decode buffer with class- i queue length $Q_{d,i}^n(t)$. When a class- i job requires decode (either immediately after prefill completion or upon a decode completion elsewhere), route as follows:

1. If some GPU in $\mathcal{G}_{\text{solo}}$ has a free decode slot, place the job uniformly at random among such GPUs.
2. Otherwise, if some GPU in \mathcal{G}_{mix} has a free decode slot, place it randomly among such GPUs.
3. Otherwise, append the job to the decode buffer (FCFS across class).

The key insight is that, from a token-level viewpoint, the decode stage only needs to keep up with the stream of decode tokens created by the prefill gate. Once the policy stabilizes this token production rate and keeps decode compute fully utilized, the decode workload is always consumable, and the detailed class mix becomes secondary. This is why a simple work-conserving rule such as FCFS is sufficient at decode.

Two mechanisms make this intuition rigorous. Static planning fixes the mixed versus solo partition so that decode is never overloaded in the fluid limit, and in the binding case it is critically loaded at an LP-optimal point with zero steady-state decode buffer. GPU physics further implies that the relative speed advantage of solo decoding over mixed decoding is the same across classes, which lets us translate capacity between solo and mixed in a class-agnostic way and treat decode as effectively homogeneous in heavy traffic. Theorem 2 formalizes this insight by proving that the resulting gate-and-route policy, with FCFS decoding, achieves asymptotic optimality.

THEOREM 2 (Asymptotic optimality of occupancy-based Gate-and-Route Policy). *Let R^* denote the optimal objective value of the steady-state fluid routing LP, and let Assumptions 1 hold. Then the occupancy-based Gate-and-Route policy $\pi^{n,*}$ is asymptotically optimal:*

$$\liminf_{T \rightarrow \infty} \liminf_{n \rightarrow \infty} R^n(T; \pi^{n,*}) = R^*.$$

4.2. Separate Charging Scheme

The bundled-revenue formulation studied above serves as our primary benchmark and aligns with the objective of maximizing the throughput of completed requests. To complement this view, we also study a *separate charging* objective in which value is recognized separately at prefill and decode. This variant lets us derive a counterpart optimal policy and clarify how the timing of revenue recognition changes the incentives for admission and routing. Formally, we define the per-GPU time-averaged reward as follows.

$$\tilde{R}_n(T; \pi^n) := \frac{1}{nT} \mathbb{E}_{\pi^n} \left[\sum_{i=1}^I \left(c_p P_i S_{p,i}^n(T) + c_d D_i S_{d,i}^n(T) \right) \right], \quad (41)$$

where $c_p, c_d \geq 0$ are unit prices, and $S_{p,i}^n(T)$ and $S_{d,i}^n(T)$ are cumulative prefill and decode completions for class i by time T .

In steady state, we optimize the corresponding fluid objective over the same feasibility constraints as the bundled LP (i.e., (40)). Substituting the service-rate definitions shows that the objective coefficients

are class-independent, so the separate-charging LP depends on $(x_i, y_{m,i}, y_{s,i})$ only through the aggregate occupancies:

$$\max_{(x,y,q)} c_p \frac{C}{\tau} \sum_{i=1}^I x_i + \frac{c_d}{\tau} \sum_{i=1}^I y_{m,i} + c_d \gamma \sum_{i=1}^I y_{s,i}. \quad (42)$$

These structural properties yield three key insights for policy design under separate charging. First, since prefill occupancy earns the same marginal reward $c_p C/\tau$ across all classes, the pricing structure itself does not prioritize any specific class mix in the prefill stage. Second, because solo-mode decode iterations are faster than mixed-mode ones ($\gamma > 1/\tau$), solo decode occupancy is strictly more valuable per unit time. A revenue-maximizing controller should thus prioritize saturating solo capacity.

Finally, unlike the bundled scheme where revenue is deferred, separate charging incentivizes the system to maintain a high "inventory" of downstream work. To keep the more valuable decode slots busy, the prefill gate should favor classes with a larger decode-to-prefill ratio D_i/P_i , as they generate more future decode revenue per unit of prefill capacity consumed. This logic motivates the static priority index used in the *Prioritize-and-Route* policy described below. Notably, the separate-charging optimum may tolerate persistent decode backlogs, as these backlogs serve as a buffer to ensure continuous decode revenue.

4.2.1. Prioritize-and-Route policy Under the separate-charging objective we reuse the same gate-and-route architecture as in the bundled scheme; the only change is the prefill admission rule (priority). We briefly summarize the modifications.

Static planning. Solve the separate-charging LP (42). We define the prefill queue targets and partition the GPUs into G_{mix} and G_{solo} following the identical procedure as in the bundled case, using the optimal prefill occupancies \tilde{x}_i^* to determine the partition size \tilde{M} .

Dynamic control. The downstream decode routing is unchanged, and the only modification is the upstream prefill gate on G_{mix} . Whenever a reserved prefill slot becomes idle and some prefill queue is nonempty, we admit the head-of-line job from the class with the largest decode-to-prefill ratio:

$$i^* \in \arg \max_{i \in I} \left\{ \frac{D_i}{P_i} \right\},$$

breaking ties arbitrarily. Service remains FCFS within each class.

THEOREM 3 (Asymptotic optimality under separate charging). *Let \tilde{R}^* be the optimal value of the steady-state fluid LP (42), and let $\tilde{R}_n(T; \pi^n)$ be the per-GPU separate-charging reward in (41). Assume Assumption 1 holds.*

For each $n \in \mathbb{N}$, define the prioritize-and-route policy $\tilde{\pi}^{n,}$ as above, then $\tilde{\pi}^{n,*}$ is asymptotically optimal for the separate-charging objective:*

$$\liminf_{T \rightarrow \infty} \liminf_{n \rightarrow \infty} \tilde{R}_n(T; \tilde{\pi}^{n,*}) = \tilde{R}^*.$$

The Revenue-Congestion Trade-off and Operational Risks. While Theorem 3 guarantees asymptotic optimality for the separate-charging objective, the underlying incentive shift comes from the structure of (42): revenue is recognized at prefill and decode separately, so a revenue-driven controller may exploit any available prefill capacity even when decode is already congested. This decoupling changes where congestion accumulates. Under bundled charging, the policy tends to regulate admissions so that the prefill buffer absorbs overload while the downstream decode buffer remains comparatively lean. Under separate charging, the system may instead build substantial decode backlogs to keep decode slots continuously busy, as illustrated in Figure 2. Operationally, this is problematic because it can lead to memory pressure and create long post-prefill waits and, in extreme cases, requests that complete prefill (and generate revenue) but experience severely delayed decode or abandon before completion. This motivates augmenting the revenue objective with explicit service constraints. Accordingly, in Section 5 we extend the framework to maximize revenue subject to SLI constraints, and in Section 6 we use shadow-price analysis to quantify the resulting economic trade-offs.

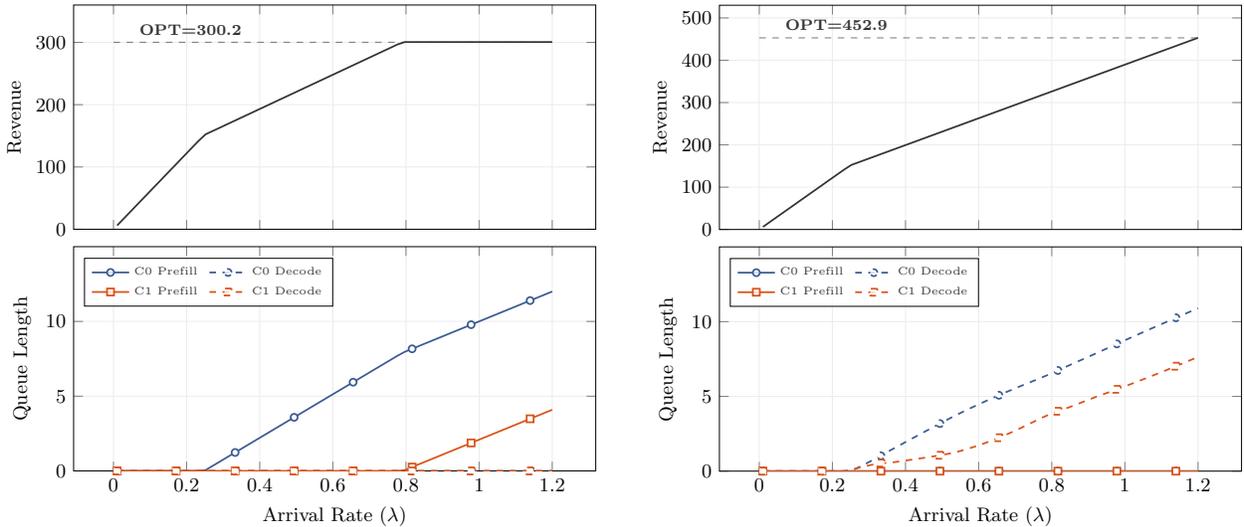


Figure 2 Comparison of Revenue and Queue Lengths under Bundled vs. Separate Charging Schemes (C0: class 0, C1: class 1).

5. SLI-Aware Optimization and Control Policy

While the preceding sections focused on revenue maximization, practical deployments must also adhere to different Service Level Indicators (SLIs), such as fairness and latency limits. In this section, we show that our fluid-based planning framework can naturally accommodate these operational requirements. By formulating SLIs as explicit constraints (or penalty terms) within the steady-state optimization, we can generate SLI-aware target occupancies without altering the fundamental structure of the control policy. This approach offers a flexible way to trade off revenue against diverse service guarantees. We focus our exposition on the bundled charging setting, though the methodology extends directly to other objectives.

SLIs are widely used in practice, but we emphasize that in our framework they are defined at the level of the steady-state fluid variables. A *service-level indicator (SLI)* is a user-facing performance metric (e.g., fairness or latency) computed from the system’s steady-state behavior. In our framework, an SLI is modeled as either (i) a *hard constraint* of the form $g(\mathbf{x}, \mathbf{y}, \mathbf{q}) \leq \eta$, or (ii) a *soft penalty* term subtracted from the revenue objective, both expressed in terms of the steady-state variables in the fluid optimization problem.

A central modeling principle we adopt throughout this section is that *a reasonable SLI should not rely on persistent decode-buffer buildup*. Indeed, in our setting decode-buffer mass generates no value under completion-based reward, but it increases waiting time and, more importantly, can lead to severe GPU-memory pressure due to KV-cache residency and migration. Accordingly, we impose the following standing assumption for the SLI-aware planning problem: the chosen SLI specification is such that the optimization admits an optimal solution with $q_{d,i}^* = 0$ for all i (decode-buffer elimination). For completeness, we provide an extension that allows $q_{d,i}^* > 0$ in the electronic companion; this case is particularly relevant under separate charging. To this end, we illustrate several canonical SLI specifications and then present the corresponding SLI-aware planning problem and control policy.

Resource fairness (prefill and decode) Fairness SLIs control the dispersion of prefill occupancies $\{x_i\}_{i \in \mathcal{I}}$ and solo decode occupancies $\{y_{s,i}\}_{i \in \mathcal{I}}$. While attractive at a high level, fairness constraints can be costly in our setting because they directly constrain the workload mix (\mathbf{x} and/or \mathbf{y}_s). This can force the system away from a hardware-efficient operating point and create a structural mismatch between prefill output and downstream decode capacity, leading to idling/under-utilization and a reduction in revenue. This effect is quantified in the Pareto frontiers in Section 6 (Fig. 8): Prefill Fairness has a steep shadow price (Fig. 8a), whereas Decode Fairness is comparatively cheap (Fig. 8b).

Prefill Fairness.

$$\max_{i,j \in \mathcal{I}} \{x_i - x_j\} \leq \eta_1, \quad q_{d,i} = 0, \quad \forall i \in \mathcal{I}. \quad (43)$$

Equivalently, the same fairness preference can be modeled in the objective via a penalty term

$$l_1 = \eta'_1 \max_{i,j \in \mathcal{I}} \{x_i - x_j\}, \quad (44)$$

with weight $\eta'_1 > 0$ that tunes the trade-off between revenue and Prefill Fairness.

Decode Fairness.

$$\max_{i,j \in \mathcal{I}} \{y_{s,i} - y_{s,j}\} \leq \eta_2, \quad q_{d,i} = 0, \quad \forall i \in \mathcal{I}. \quad (45)$$

In penalty form, we instead add

$$l_2 = \eta'_2 \max_{i,j \in \mathcal{I}} \{y_{s,i} - y_{s,j}\}, \quad (46)$$

with weight $\eta'_2 > 0$; larger η'_2 places more emphasis on equalizing solo decode usage across classes.

Average Time per Output Token While the worst-case Time per Output Token is governed by the chunk size C , the average TPOT depends on the cluster-wide balance between prefill and decode activity. Each unit of prefill occupancy x_i introduces mixed-mode iterations that slow co-resident decodes. A natural SLI is to cap the average TPOT:

$$\frac{\tau(B-1) \sum_{i=1}^I x_i + \frac{1}{\gamma} B(1 - \sum_{i=1}^I x_i)}{(B-1) \sum_{i=1}^I x_i + B(1 - \sum_{i=1}^I x_i)} \leq \eta_3, \quad (47)$$

for some target $\eta_3 > 0$. In this formulation we retain the standard capacity constraints in (40), so idling is permitted when the TPOT cap is tight; the revenue objective still discourages unnecessary idling whenever additional work can be served.

Alternatively, we can incorporate TPOT directly into the objective by penalizing total prefill load:

$$l_3 = \eta'_3 \frac{\tau(B-1) \sum_{i=1}^I x_i + \frac{1}{\gamma} B(1 - \sum_{i=1}^I x_i)}{(B-1) \sum_{i=1}^I x_i + B(1 - \sum_{i=1}^I x_i)}, \quad (48)$$

where $\eta'_3 > 0$ controls the strength of the revenue–latency trade-off.

5.1. SLI-Aware Gate-and-Route Control Policy

To incorporate SLIs into the steady-state optimization, we augment the objective of Section 3.1 (or Section 4.2 for separate charging) by subtracting a weighted sum of penalty terms:

$$\max_{(\mathbf{x}, \mathbf{y}, \mathbf{q}) \in \mathcal{F}_{\mathcal{K}}} \sum_{i=1}^I w_i (\mu_{m,i} y_{m,i} + \mu_{s,i} y_{s,i}) - \sum_{k \in \mathcal{K}} l_k, \quad (49)$$

where \mathcal{K} indexes the active SLIs and l_k are chosen from (44), (46), (48), or other application-specific penalties. We append the SLI-specific constraints to the feasibility constraints from Section 3.1, and denote the feasibility region by $\mathcal{F}_{\mathcal{K}}$.

To realize the targets $(y_{m,i}^*, y_{s,i}^*)$ derived from the SLI-aware planning problem, we employ a randomized decode router. For clarity of exposition, we focus on the zero-buffer case ($q_{d,i}^* = 0$) in this section, which avoids memory pressure and simplifies the tracking mechanism. An extension that accommodates persistent decode queues ($q_{d,i}^* > 0$) is provided in Section EC.6.

Static planning. This phase follows the identical procedure as defined in Section 4, determining the cluster-level queue targets $Q_{P,i}^\dagger$ and the GPU partition sets (\mathcal{G}_{mix} and $\mathcal{G}_{\text{solo}}$) based on the optimal solution $(x_i^*, y_{m,i}^*, y_{s,i}^*, q_{p,i}^*)$ of the corresponding SLI-aware optimization problem in (49).

Dynamic control. The prefill gate remains the same as the *Gate-and-Route* policy in Section 4.1. The decode router splits the decode buffer into mixed and solo components and computes the class- i solo probability

$$p_{s,i} := \begin{cases} \frac{\mu_{s,i} y_{s,i}^*}{\mu_{m,i} y_{m,i}^* + \mu_{s,i} y_{s,i}^*}, & \text{if } \mu_{m,i} y_{m,i}^* + \mu_{s,i} y_{s,i}^* > 0, \\ 1, & \text{otherwise.} \end{cases}$$

Upon prefill or decode completion of class i , draw $U \sim \text{Unif}(0, 1)$; route to $\mathcal{G}_{\text{solo}}$ if $U \leq p_{s,i}$ and to \mathcal{G}_{mix} otherwise, placing the decode uniformly at random among GPUs in the targeted group with free slots (or queuing in the corresponding buffer if none available). Note that here we logically split the decode buffer into mixed buffer and solo buffer instead of a single decode buffer mentioned in Section 4.

The router implements a randomized load split that mirrors the fluid targets: each class- i decode is sent to solo or mixed with probability $p_{s,i}$ chosen so that the long-run fraction of class- i service provided by solo vs. mixed matches $(y_{s,i}^*, y_{m,i}^*)$. As many decodes are routed over time, the law of large numbers forces the realized occupancies $(y_{m,i}^n/n, y_{s,i}^n/n)$ to track these target proportions, effectively “reshuffling” decode work until the stochastic system hovers around the desired steady-state levels.

The asymptotic optimality guarantee of Theorem 2 extends to the SLI-aware policy under mild regularity on the optimization problem (bounded penalties, etc.).

THEOREM 4 (Occupancy Convergence and Asymptotic Optimality of SLI-Aware Policy). *Let $(x_i^*, y_{m,i}^*, y_{s,i}^*, q_{p,i}^*, q_{d,i}^*)_{i \in \mathcal{I}}$ denote an optimal solution of the SLI-aware LP (49) with active SLI set \mathcal{K} and corresponding penalties $\{l_k\}_{k \in \mathcal{K}}$ and constraints. Assume the SLI-aware LP satisfies Slater’s condition and each penalty l_k is bounded and Lipschitz continuous in the decision variables. Further assume that the selected optimal solution satisfies $q_{d,i}^* = 0$ for all i (e.g., by including $q_{d,i} = 0$ in the constraints). Under the SLI-aware control policy, the scaled steady-state occupancies converge:*

$$\lim_{n \rightarrow \infty} \frac{1}{n} \mathbb{E}[X_i^{(n)}] = x_i^*, \quad \lim_{n \rightarrow \infty} \frac{1}{n} \mathbb{E}[Y_{m,i}^{(n)}] = y_{m,i}^*, \quad \lim_{n \rightarrow \infty} \frac{1}{n} \mathbb{E}[Y_{s,i}^{(n)}] = y_{s,i}^*,$$

for all $i \in \mathcal{I}$, and the per-GPU SLI-aware objective value converges to optimality:

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{1}{n} \mathbb{E} \left[\sum_{i=1}^I w_i \left(\mu_{m,i} Y_{m,i}^{(n)} + \mu_{s,i} Y_{s,i}^{(n)} \right) - n \sum_{k \in \mathcal{K}} l_k(\mathbf{X}^{(n)}/n, \mathbf{Y}_m^{(n)}/n, \mathbf{Y}_s^{(n)}/n) \right] \\ = \sum_{i=1}^I w_i \left(\mu_{m,i} y_{m,i}^* + \mu_{s,i} y_{s,i}^* \right) - \sum_{k \in \mathcal{K}} l_k(\mathbf{x}^*, \mathbf{y}_m^*, \mathbf{y}_s^*), \end{aligned}$$

where $\mathbf{X}^{(n)} = (X_1^{(n)}, \dots, X_I^{(n)})$ and similarly for $\mathbf{Y}_m^{(n)}, \mathbf{Y}_s^{(n)}$.

6. Numerical Experiments

In this section, we evaluate the performance of the proposed *Gate-and-Route* policy through event-driven simulations calibrated with real-world LLM inference profiles. We compare our approach against standard industry heuristics across varying cluster sizes, demonstrating that our fluid-based control effectively maximizes system throughput and revenue while satisfying SLI constraints in the long run. We also conduct sensitivity analysis to understand the tradeoff between SLIs, GPU configurations, and revenue.

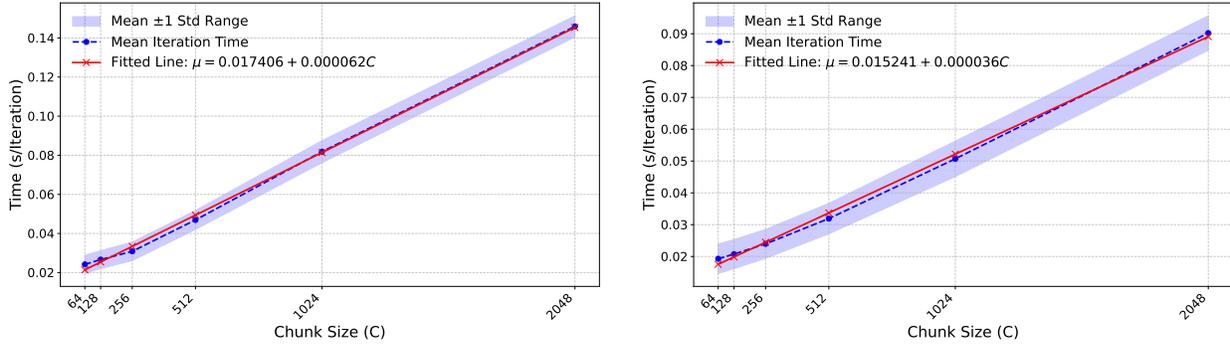


Figure 3 Calibration of mixed-iteration hyperparameters (α, β) for Qwen-8B (left) and Qwen-4B (right). Dots show the empirical mean iteration time τ for each chunk size C , and the solid line is the OLS fit $\tau = \alpha + \beta C$. For Qwen-8B, the fitted line is $\tau = 0.0174 + 6.2 \times 10^{-5} C$ with $R^2 = 0.998$; for Qwen-4B, it is $\tau = 0.0152 + 3.6 \times 10^{-5} C$ with $R^2 = 0.997$.

6.1. Calibration of Hyperparameters

We begin by calibrating the iteration-time parameters α , β , and γ used in the subsequent numerical experiments. Our measurements are conducted on a server equipped with $4 \times$ NVIDIA A100-SXM4-40GB GPUs; detailed hardware and software specifications are reported in the EC (Table EC.3). In practice, we use vLLM version 0.11.0 to evaluate two models: Qwen-4B and Qwen-8B, and for each model we measure both mixed and solo decode modes.

For mixed-mode calibration, we vary the prefill chunk size C and record the mean iteration time τ on a single GPU, then fit a linear model $\tau \approx \alpha + \beta C$ via ordinary least squares. As shown in Figure 3, the fitted lines closely track the empirical means: for Qwen-8B we obtain $\alpha \approx 0.0174$ and $\beta \approx 6.2 \times 10^{-5}$, and for Qwen-4B we obtain $\alpha \approx 0.0152$ and $\beta \approx 3.6 \times 10^{-5}$, with coefficients of determination $R^2 > 0.99$ in both cases. This supports the linear mixed-iteration model used in Section 2.2.

For solo decode, we set the per-stream token speed γ to the empirical mean across runs. In our setup, this yields $\gamma \approx 45.45$ tokens/s for Qwen-8B and $\gamma \approx 52.63$ tokens/s for Qwen-4B.

6.2. Convergence Analysis

We evaluate the asymptotic performance of our proposed policies using a two-class instance ($I = 2$) calibrated to the hardware specifications derived in Section 6.1. The system parameters are $B = 16$, $C = 256$, $\alpha = 0.0174$, and $\beta = 6.2 \times 10^{-5}$. The solo decode speed is set to $\gamma \approx 45.45$ tokens/s (corresponding to a per-token latency of 0.022s).

We define two distinct workload classes to represent the heterogeneity discussed in Table EC.1:

- **Class 0 (Decode-Heavy):** $P_0 = 300$, $D_0 = 1000$. Represents tasks like code generation.
- **Class 1 (Prefill-Heavy):** $P_1 = 3000$, $D_1 = 400$. Represents tasks like paper summarization or context-heavy analysis.

Arrival rates are symmetric with $\lambda = [0.5, 0.5]$, and abandonment rates are $\theta = [0.1, 0.1]$. We utilize the separate charging scheme with prices $c_p = 0.1$ and $c_d = 0.2$. We simulate the system across varying scales $n \in \{5, 20, 50, 200, 500\}$, running 5 random seeds per configuration to capture stochastic variability.

Revenue and Queue Convergence. We first validate the asymptotic optimality of the standard gate-and-route policy. Figure 4 illustrates the per-GPU revenue and queue lengths as the system scale n increases.

As predicted by the fluid limit, the average per-GPU revenue (Left) converges to the optimal value R^* derived from the steady-state LP. The shaded error bands, representing the standard deviation across seeds, narrow significantly as $n \rightarrow \infty$, confirming the concentration of measure. Simultaneously, the normalized queue lengths for both Class 0 (Top Right) and Class 1 (Bottom Right) stabilize near their fluid targets. Notably, the prefill gate effectively regulates the admission process, ensuring that the stochastic queue lengths track the fluid-optimal backlog required to maintain utilization, while the decode buffer remains negligible.

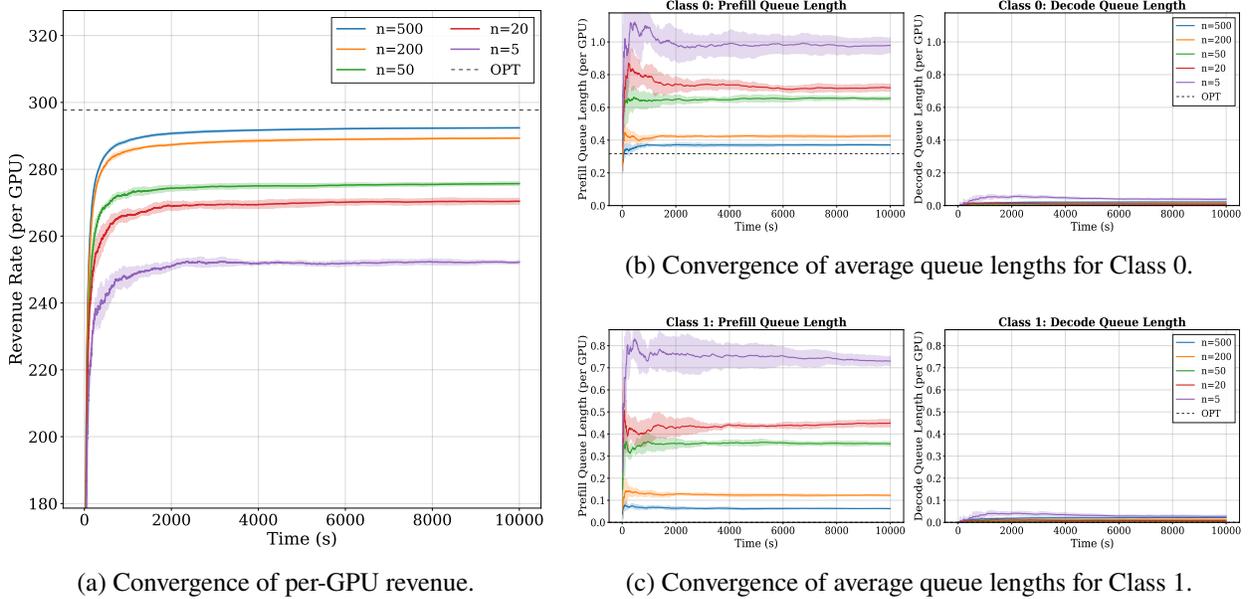


Figure 4 Asymptotic convergence of Revenue and Queue Lengths under the Gate-and-Route policy.

Occupancy Convergence Analysis. While the aggregate revenue converges, a deeper inspection reveals a discrepancy in the convergence behavior of specific resource occupancies under the standard policy. Figure 5 plots the fluid-scaled prefill occupancy $X_i^{(n)}/n$ and decode occupancy $Y_i^{(n)}/n$.

We observe that the prefill occupancy (Top) converges tightly to the LP solution x_i^* . This is expected, as the prefill gate explicitly targets these values. However, the decode occupancy (Bottom) exhibits persistent variance and deviates from the specific class-wise breakdown $y_{m,i}^* + y_{s,i}^*$ predicted by the LP, even though the total reward is optimal. This occurs because the standard router prioritizes *any* available solo slot to maximize throughput (work-conservation) but is indifferent to *which* class occupies that slot. Consequently, the specific mix of Class 1 vs. Class 2 in decode slots fluctuates, satisfying the aggregate capacity constraints but failing to converge to the specific fluid vector y^* .

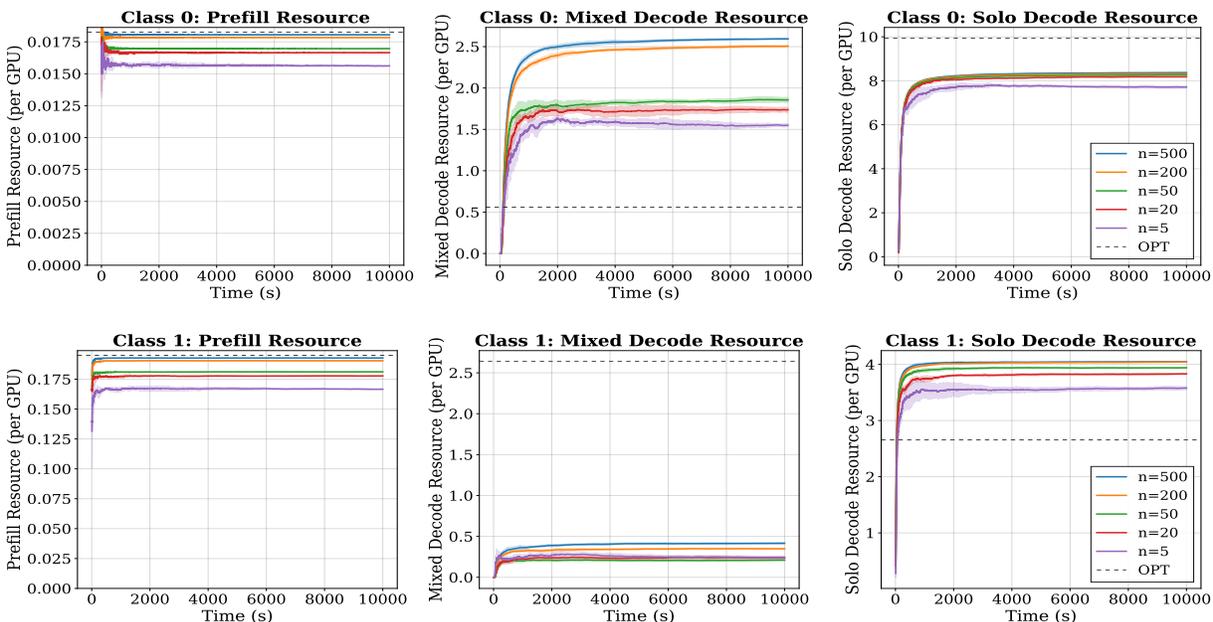


Figure 5 Occupancy convergence under the *Gate-and-Route* Policy. Top: Class 0 (Decode-Heavy). Bottom: Class 1 (Prefill-Heavy). Note the loose convergence for decode occupancy (y).

SLI-Aware Convergence. To address the decode occupancy drift and enforce strict adherence to the fluid plan (crucial for fairness SLIs), we employ the SLI-aware gate-and-route policy described in Section 5.1. This policy utilizes a stochastic router with probabilities $p_{s,i}$ derived from the LP solution to probabilistically route jobs to mixed or solo pools.

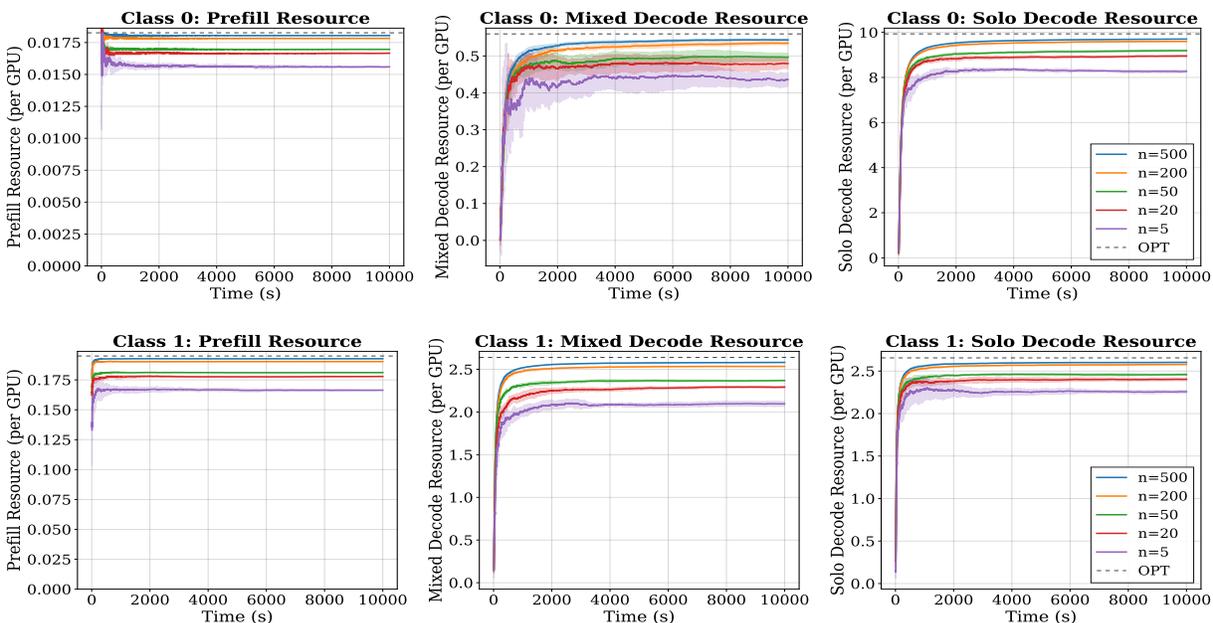


Figure 6 Occupancy convergence under the *SLI-Aware Gate-and-Route* Policy. Top: Class 0 (Decode-Heavy). Bottom: Class 1 (Prefill-Heavy). Both prefill (x) and decode (y) occupancies now converge tightly to the LP targets.

Figure 6 demonstrates the impact of this policy. Unlike the standard case, both the prefill occupancy and the decode occupancy now converge strictly to the optimization solution for both classes. By enforcing the specific routing split $y_{m,i}^*$ vs. $y_{s,i}^*$, the SLI-aware policy ensures that the stochastic system structurally mimics the fluid limit. This validates Theorem 4, confirming that we can achieve precise control over resource allocation distributions at scale, which is a prerequisite for satisfying multi-objective SLIs in the long run.

6.3. Policy Performance: Baselines and Ablations

To evaluate the efficacy of our proposed framework, we compare several alternative scheduling policies across a variety of instances. These instances encompass diverse infrastructure hyperparameters (α from 0.15 to 0.02, β from 10^{-5} to 10^{-3} , and γ from 10 to 50), distinct target user compositions (P_i, D_i from 200 to 3000), and different arrival rates λ from 0.25 to 0.5, and all under GPU number $n = 500$. We report the normalized revenue rate (scaled from 0 to 1) along with standard deviations to demonstrate robustness.

Candidate Policies We evaluate the following five policies together with our proposed policy to isolate the contributions of our admission control (Gate), decode router (Route), and static planning:

- **Gate-and-Route (Greedy Router) with Static Planning (GG-SP):** This is our proposed policy. It utilizes the queue-length based admission control (Gate) derived from our theoretical analysis. For routing, it uses a Greedy router in the gate-and-route architecture; specifically, the router prioritizes dispatching waiting decode tasks to available solo GPUs over initiating new prefill tasks.
- **FCFS-and-Immediate without Static Planning (FI-WSP):** This represents the standard industry baseline in Sarathi-serve (Agrawal et al. 2024). It employs First-Come-First-Served (FCFS) admission without any queue-length based control. Crucially, it uses an “Immediate” execution model where prefill and decode phases are coupled: the decode phase commences immediately on the same GPU slot following the prefill completion, occupying the resource continuously until the request is finished.
- **Gate-and-Immediate without Static Planning (GI-WSP):** This policy introduces our queue-length based Gate control to the coupled architecture. Like FI-WSP, it executes decode immediately after prefill on the same slot. Comparing GI-WSP against GG-SP highlights the specific performance gains achieved by *decoupling* the prefill and decode phases.
- **Gate-and-FCFS without Static Planning (GF-WSP):** This policy utilizes the same decoupled architecture and queue-length based Gate as our proposed method. However, the routing logic is inverted to a naive FCFS approach: whenever a GPU slot becomes available, the system prioritizes admitting a new prefill task rather than processing a waiting decode task. Comparing this against GG-SP demonstrates the necessity of the *decode-first* (Greedy) prioritization in a decoupled system.
- **FCFS-and-Route (Greedy Router) with Static Planning (FG-SP):** This policy mirrors our proposed gate-and-route architecture with the same Greedy router, but removes the admission control mechanism and accepts all arrivals via FCFS. Comparing FG-SP against GG-SP isolates the role of the *Gate* in preventing system overload and maintaining revenue stability.

The comparative results, illustrated in Figure 7, reveal a critical hierarchy of design elements. We find that while the admission gate alone (GI-WSP) provides a meaningful improvement over the industry baseline (FI-WSP), other components like static planning can actually be counter-productive when applied in isolation (as seen in FG-SP), where rigid resource partitioning without traffic regulation may lead to localized imbalances. The full revenue potential is only unlocked in our integrated GG-SP policy: the gate controls the decode workloads, while the static planning enables the greedy router to extract maximal efficiency from the hardware pools. Additionally, the consistently low standard deviations observed for GG-SP across varied instances underscore the robustness of this integrated design.

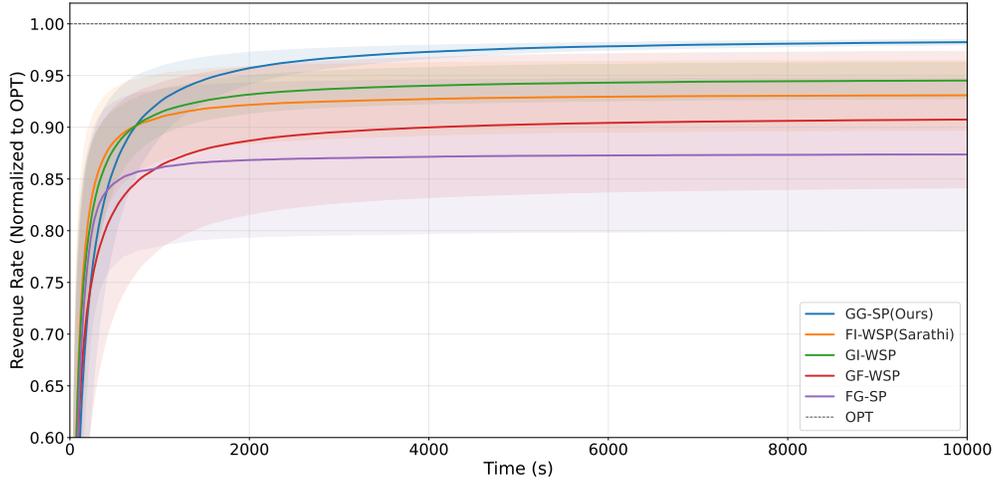


Figure 7 Comparison of normalized average revenue across different policies. The results aggregate performance over various infrastructure settings (α, β, γ) and user class combinations. Error bars indicate the standard deviation.

6.4. The Cost of Service Quality and Hardware Resources

To investigate the cost of service quality and fairness, we analyze the trade-off between total revenue and specific SLIs. We utilize the same system instance defined in the convergence analysis to ensure consistency, which is representative since it consists of long-prefill-short-decode and short-prefill-long-decode classes. We formulate this as a constrained optimization problem where we maximize revenue subject to a strict constraint on exactly one SLI metric at a time: Prefill Fairness, Decode Fairness, or Time Per Output Token (TPOT), while relaxing the others. This approach allows us to isolate the “price” of each specific constraint in terms of lost revenue.

The Shadow Price of SLIs Figure 8 illustrates the Pareto frontiers for three operational constraints. We interpret the slope of these curves as the **shadow price**, which is the marginal revenue sacrificed to achieve a stricter SLI target. It reveals distinct economic sensitivities:

- **Asymmetric Costs of Fairness (Figs. 8a and 8b):** We observe a stark contrast between the shadow prices of fairness at different stages of the inference pipeline. Prefill fairness (η_1) incurs a steep revenue

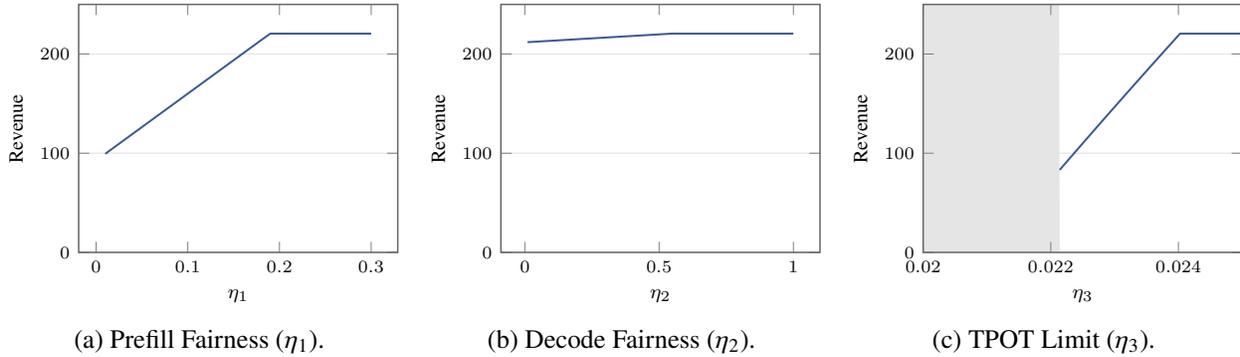


Figure 8 Pareto frontiers illustrating the shadow price of Service Level Indicators (SLIs).

penalty because the prefill stage acts as the primary bottleneck; imposing rigid class-mix constraints here prevents the scheduler from aligning admissions with the hardware’s optimal operating point, leading to a structural mismatch between job arrivals and downstream capacity. In contrast, the nearly flat frontier for decode fairness (η_2) suggests a negligible shadow price. This implies that once a request has entered the system, rebalancing its processing speed relative to other classes barely degrades total revenue.

- **The Price of Low TPOT (Fig. 8c):** The shadow price of the TPOT constraint increases significantly as the target latency approaches 0.022 s, a lower bound determined by the solo-decode rate γ . Near this threshold, the feasible region for prefill throughput shrinks rapidly, leading to a substantial decrease in optimal revenue as the system prioritizes meeting the stringent latency requirement over throughput.

GPU configurations and Revenue We next analyze the sensitivity of the system’s performance to the maximum batch size (B) and the hyperparameters (α, β, γ). Figure 9 illustrates the trade-off between total Revenue (primary objective, left y-axis) and TPOT (latency cost, right y-axis).

The results indicate distinct operational trends. First, revenue increases with batch size B but saturates around $B = 16$, suggesting diminishing returns on memory scaling beyond this point. Second, the system is highly sensitive to the computational penalty β ; as β increases, revenue drops sharply. Finally, γ acts as a strong incentive, positively correlating with higher revenue and lower latency.

Figure 10a visualizes the revenue landscape across the joint configuration space of memory capacity (proxied by B) and computational speed (proxied by β). This mapping serves two critical strategic functions for infrastructure management.

First, the heatmap provides a quantitative basis for hardware selection by allowing operators to overlay GPU market prices onto the revenue landscape to identify configurations with the highest return on investment (ROI). Second, its gradient directs hardware upgrades by revealing the steepest path to revenue growth. By identifying whether this path favors increasing B or decreasing β , decision-makers can pinpoint the primary bottleneck (memory or compute) and prioritize investments to maximize marginal gains.

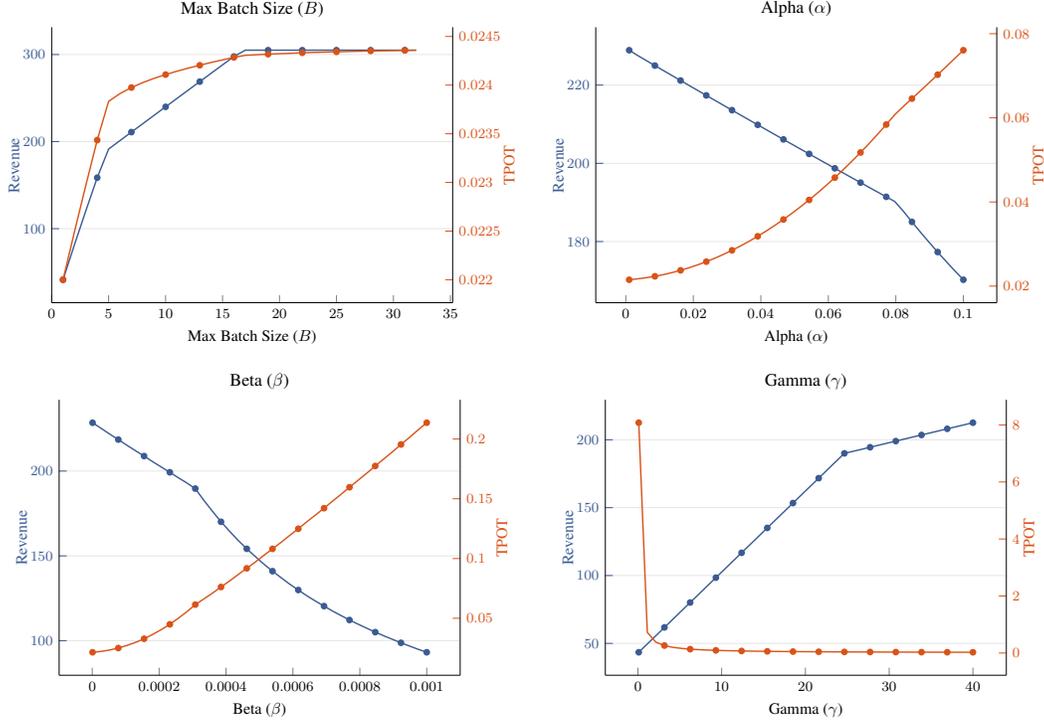


Figure 9 Parameter sensitivity analysis showing the impact of maximum batch size (B), α , β , and γ on revenue and TPOT. Blue lines indicate revenue (higher is better); red lines indicate TPOT (lower is better).

Optimal Token Pricing We also examine the optimal pricing structure by analyzing the relationship between the prefill price c_p and the decode price c_d . Specifically, if we investigate the common revenue maximization problem subject to a total price constraint $c_p + c_d = k$, the heatmap (Figure 10b) results reveal a striking invariance: regardless of the magnitude of the budget k , the revenue-maximizing prices consistently yield a unique, constant ratio c_p/c_d . This indicates that the optimal economic balance between prefill and decode is scale-invariant. Consequently, for pricing strategy, managers should focus on maintaining this intrinsic cost structure ratio, as the optimal split between prefill and decode prices remains stable even as the total price level varies.

7. Conclusions

Efficient LLM inference at scale hinges on resolving the resource contention between compute-bound prefill and memory-bound decode. In this work, we connect an empirically grounded iteration-time abstraction with stochastic control to study admission and scheduling in large GPU clusters under token-based revenue objectives and service constraints. A multiclass many-GPU fluid approximation yields a tractable steady-state linear program that prescribes how to split capacity between mixed and solo modes (γ) and how to allocate prefill occupancy across classes.

Building on this planning formulation, we develop a *gate-and-route* control architecture with a prefill admission gate that tracks the fluid occupancy targets and a decode router that keeps downstream capacity work-conserving. Our analysis establishes a set of structural properties that enable this decomposition,

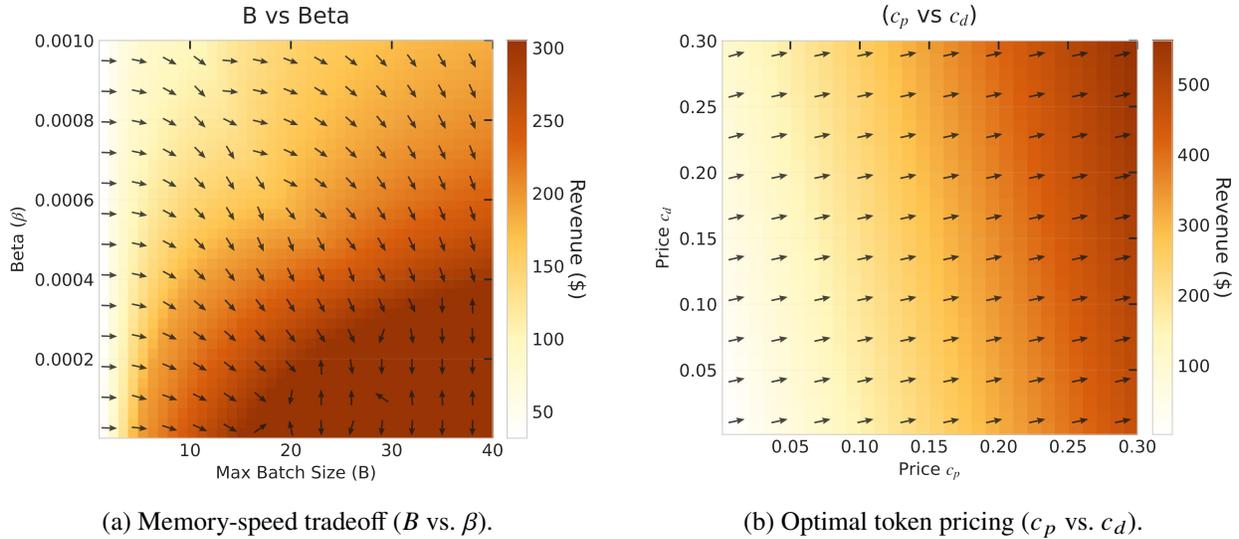


Figure 10 Sensitivity analysis of hardware configurations and pricing structures on per-GPU revenue.

including the existence of an optimal fluid plan with no steady-state decode buffering (Proposition 1) and a corresponding asymptotic optimality guarantee for the proposed policy in the many-GPU limit.

Our numerical evaluation is empirically calibrated and intended to illustrate the mechanisms highlighted by the theory: we calibrate the iteration-time model using real deployments on a modern inference stack, and then run calibrated event-driven simulations to study the resulting control behavior. In particular, simulations confirm two qualitative predictions from the asymptotic analysis: per-GPU revenue approaches the fluid optimum as the cluster scales, and persistent decode backlogs are avoided under the proposed control. We further compare against representative heuristic baselines and ablations in our calibrated setting, with the goal of clarifying mechanisms rather than providing a production benchmark.

Our results also yield actionable implications for service providers. First, billing and scheduling objectives need not coincide: while separate charging for prefill and decode tokens is natural for accounting, optimizing the scheduler against a separate objective can encourage overly aggressive prefill admission and shift congestion downstream. A practical recommendation is therefore to bill by phase if desired, while scheduling against an end-to-end (bundled) completion objective to align incentives with user-perceived performance. Second, incorporating SLIs as constraints or penalties in the planning problem provides a systematic way to study the revenue implications of fairness and latency requirements; in our setting, enforcing fairness at the prefill stage is typically more revenue-costly than at the decode stage.

Several avenues exist to extend the theoretical depth and practical scope of this work. First, one can relax the assumption of exponential service times by employing measure-valued processes. This formulation would accommodate general service distributions, enabling more granular, state-dependent control policies. Second, to move beyond mean-value analysis, developing diffusion approximations that characterize stochastic variability will provide rigorous guarantees for tail-latency SLIs. Finally, the model can be generalized to

heterogeneous infrastructures, orchestrating inference across clusters composed of diverse GPU generations and distinct agent architectures.

References

- Agrawal A, Kedia N, Panwar A, Mohan J, Kwatra N, Gulavani BS, Tumanov A, Ramjee R (2024) Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*.
- Ai R, Pan Y, Simchi-Levi D, Tambe M, Xu H (2025) Beyond majority voting: LLM aggregation by leveraging higher-order information.
- Aminabadi RY, Rajbhandari S, Awan AA, Li C, Li D, Zheng E, Ruwase O, Smith M, Zhang M, Rasley J, et al. (2022) DeepSpeed-Inference: Enabling efficient inference of transformer models at unprecedented scale. *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–15 (IEEE).
- Anthropic (2025) Claude API pricing. URL <https://claude.com/pricing#api>.
- Ao R, Luo G, Simchi-Levi D, Wang X (2025) Optimizing LLM inference: Fluid-guided online scheduling with memory constraints.
- Ata B (2006) Dynamic control of a multiclass queue with thin arrival streams. *Operations Research* 54(5):876–892.
- Atar R, Giat C, Shimkin N (2010) The $c\mu/\theta$ rule for many-server queues with abandonment. *Operations Research* 58(5):1427–1439.
- Bassamboo A, Randhawa RS (2010) On the accuracy of fluid models for capacity sizing in queueing systems with impatient customers. *Operations Research* 58(5):1398–1413.
- Bassamboo A, Randhawa RS (2016) Scheduling homogeneous impatient customers. *Management Science* 62(7):2129–2147.
- Chen Z, Ye Y, Zhou Z (2025) Adaptively robust LLM inference optimization under prediction uncertainty. *arXiv preprint* URL <http://dx.doi.org/10.48550/arXiv.2508.14544>.
- Conover M, Hayes M, Mathur A, Xie J, Wan J, Shah S, Ghodsi A, Wendell P, Zaharia M, Xin R (2023) Free dolly: Introducing the world’s first truly open instruction-tuned LLM. URL <https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm>.
- Dai JG, Tezcan T (2011) State Space Collapse in Many-Server Diffusion Limits of Parallel Server Systems. *Mathematics of Operations Research* 36(2):271–320.
- Dai T, Swaminathan JM (2026) Artificial intelligence and operations: A foundational framework of emerging research and practice. *Production and Operations Management* .
- Dai Z, Yang Z, Yang Y, Carbonell JG, Le Q, Salakhutdinov R (2019) Transformer-XL: Attentive language models beyond a fixed-length context. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2978–2988.

-
- DeepSeek (2025) Models & pricing — DeepSeek API docs. URL https://api-docs.deepseek.com/quick_start/pricing/.
- Dong J, Feldman P, Yom-Tov GB (2015) Service systems with slowdowns: Potential failures and proposed solutions. *Operations Research* 63(2):305–324.
- Ghosh S, Lam H (2019) Robust Analysis in Stochastic Simulation: Computation and Performance Guarantees. *Operations Research* 67(1):232–249.
- Google (2025) Billing — Gemini API — Google AI for developers. URL <https://ai.google.dev/gemini-api/docs/billing/>.
- Google Cloud (2025) Vertex AI generative AI pricing. URL <https://cloud.google.com/vertex-ai/generative-ai/pricing>.
- Harrison JM, Zeevi A (2004) Dynamic scheduling of a multiclass queue in the Halfin-Whitt heavy traffic regime. *Operations Research* 52(2):243–257.
- Hu H, Simchi-Levi D (2025) Pre-trained AI model assisted online decision-making under missing covariates: A theoretical perspective.
- Huang C, Tang Z, Hu S, Jiang R, Zheng X, Ge D, Wang B, Wang Z (2025) ORLM: A customizable framework in training large models for automated optimization modeling. *Operations Research* Articles in Advance.
- Ibrahim R (2018) Sharing delay information in service systems: A literature survey. *Queueing Systems* 89:49–79.
- Ibrahim R, Whitt W (2009) Real-Time delay estimation in overloaded multiserver queues with abandonments. *Management Science* 55(10):1729–1742.
- Jaillet P, Jiang J, Mellou K, Molinaro M, Podimata C, Zhou Z (2025) Online scheduling for LLM inference with KV cache constraints. *arXiv preprint* URL <http://dx.doi.org/10.48550/arXiv.2502.07115>.
- Kwon W, Li Z, Zhuang S, Sheng Y, Zheng L, Yu CH, Gonzalez J, Zhang H, Stoica I (2023) Efficient memory management for large language model serving with PagedAttention. *Proceedings of the 29th Symposium on Operating Systems Principles*, 611–626.
- Lam H (2016) Robust Sensitivity Analysis for Stochastic Systems. *Mathematics of Operations Research* 41(4):1248–1275.
- Li B, Jiang Y, Gadepally V, Tiwari D (2024) LLM inference serving: Survey of recent advances and opportunities. *2024 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–8.
- Li Y, Dai J, Peng T (2025) Throughput-optimal scheduling algorithms for LLM inference and AI agents. *arXiv preprint* URL <http://dx.doi.org/10.48550/arXiv.2504.07347>.
- Long Z, Shimkin N, Zhang H, Zhang J (2020) Dynamic scheduling of multiclass many-server queues with abandonment: The generalized $c\mu/h$ rule. *Operations Research* 68(4):1218–1230.
- Long Z, Zhang H, Zhang J, Zhang ZG (2024) The generalized c/μ rule for queues with heterogeneous server pools. *Operations Research* 72(6):2488–2506.

- Mitzenmacher M, Shahout R (2025) Queueing, predictions, and large language models: Challenges and open problems. *Stochastic Systems* Articles in Advance.
- OpenAI (2025) OpenAI API pricing. URL <https://openai.com/api/pricing>.
- Patel P, Choukse E, Zhang C, Shah A, Goiri I, Maleki S, Bianchini R (2024) SplitWise: Efficient generative LLM inference using phase splitting. *Proceedings of the 51st International Symposium on Computer Architecture (ISCA 2024)*, 3775–3801.
- Shazeer N (2019) Fast transformer decoding: One write-head is all you need. *arXiv preprint* URL <http://dx.doi.org/10.48550/arXiv.1911.02150>.
- Simchi-Levi D, Dai T, Menache I, Wu MX (2025) Democratizing optimization with generative AI. *SSRN preprint* URL <http://dx.doi.org/10.2139/ssrn.5511218>.
- Simchi-Levi D, Mellou K, Menache I, Pathuri J (2026) Large language models for supply chain decisions. *AI in Supply Chains*, volume 27 of *Springer Series in Supply Chain Management* (Springer).
- Sun B, Huang Z, Zhao H, Xiao W, Zhang X, Li Y, Lin W (2024) Llumnix: Dynamic scheduling for large language model serving. *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 173–191.
- Wang M, Ye Y, Zhou Z (2025) LLM serving optimization with variable prefill and decode lengths. *arXiv preprint* URL <http://dx.doi.org/10.48550/arXiv.2508.06133>.
- Whitt W (2006) Fluid models for multiserver queues with abandonments. *Operations Research* 54(1):37–54.
- Wu C, Bassamboo A, Perry O (2019) Service system with dependent service and patience times. *Management Science* 65(3):1151–1172.
- Yu GI, Jeong J, Kim G, Kim S, Chun BG (2022) Orca: A distributed serving system for transformer-based generative models. *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, 521–538 (USENIX Association).
- Zhang J (2013) Fluid models of many-server queues with abandonment. *Queueing Systems* 73(2):147–193.
- Zhao W, Ren X, Hessel J, Cardie C, Choi Y, Deng Y (2024) WildChat: 1m ChatGPT interaction logs in the wild. *The Twelfth International Conference on Learning Representations*.
- Zhao WX, Zhou K, Li J, Tang T, Wang X, Hou Y, Min Y, Zhang B, Zhang J, Dong Z, Du Y, Yang C, Chen Y, Chen Z, Jiang J, Ren R, Li Y, Tang X, Liu Z, Liu P, Nie JY, Wen JR (2025) A survey of large language models.
- Zheng L, Chiang WL, Sheng Y, Li T, Zhuang S, Wu Z, Zhuang Y, Li Z, Lin Z, Xing E, et al. (2024) LMSYS-Chat-1M: A large-scale real-world LLM conversation dataset. *The Twelfth International Conference on Learning Representations*.
- Zhong Y, Liu S, Chen J, Hu J, Zhu Y, Liu X, Jin X, Zhang H (2024) DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 193–210.
- Zhou Z, Ning X, Hong K, Fu T, Xu J, Li S, Lou Y, Wang L, Yuan Z, Li X, Yan S, Dai G, Zhang XP, Dong Y, Wang Y (2024) A survey on efficient inference for large language models.

EC.1. Proof of Proposition 1

We mainly adopt the strategy that given an optimal solution with $q_{d,i} > 0$, we can always construct an optimal solution with $q'_{d,i} = 0$ based on the solution given above. The proof follows from the following steps:

First, we eliminate q_d in the original optimization problem to get a new optimization problem (P') which better characterizes our target. Fix a class i with $\theta_i > 0$. From the second constraint in (40), we have:

$$\mu_{p,i}x_i - \theta_i q_{d,i} = \mu_{m,i}y_{m,i} + \mu_{s,i}y_{s,i},$$

Considering $q_{d,i} \geq 0$, we obtain the following inequality

$$\mu_{m,i}y_{m,i} + \mu_{s,i}y_{s,i} \leq \mu_{p,i}x_i, \quad (\text{EC.1})$$

with

$$q_{d,i} = \frac{\mu_{p,i}x_i - \mu_{m,i}y_{m,i} - \mu_{s,i}y_{s,i}}{\theta_i} \geq 0. \quad (\text{EC.2})$$

Here $q_{d,i}$ is exactly the slack variable of (EC.1). Without loss of generality, we focus on $\theta_i > 0$.

Define the problem (P'):

$$\begin{aligned} & \max_{\{x_i, y_{m,i}, y_{s,i}, q_{p,i}\}_{i=1}^I} \sum_{i=1}^I w_i (\mu_{m,i}y_{m,i} + \mu_{s,i}y_{s,i}) \\ & \text{s.t.} \quad \sum_{i=1}^I x_i \leq 1, \\ & \quad \sum_{i=1}^I y_{m,i} \leq (B-1) \sum_{i=1}^I x_i, \\ & \quad \sum_{i=1}^I y_{s,i} \leq B \left(1 - \sum_{i=1}^I x_i\right), \\ & \quad \lambda_i - \theta_i q_{p,i} = \mu_{p,i}x_i, \quad \forall i, \\ & \quad \mu_{m,i}y_{m,i} + \mu_{s,i}y_{s,i} \leq \mu_{p,i}x_i, \quad \forall i, \\ & \quad x_i, y_{m,i}, y_{s,i}, q_{p,i} \geq 0, \quad \forall i. \end{aligned} \quad (\text{EC.3})$$

For any feasible solution of (EC.3) we can recover a feasible solution of (40) by defining $q_{d,i}$ via (EC.2). Conversely, any feasible solution of (40) satisfies (EC.1) and thus yields a feasible solution of (EC.3) by dropping $q_{d,i}$. Hence the two problems are equivalent and share the same optimal value.

In particular, $q_{d,i} = 0$ for a given class i is equivalent to the inequality (EC.1) being *tight*:

$$q_{d,i} = 0 \iff \mu_{m,i}y_{m,i} + \mu_{s,i}y_{s,i} = \mu_{p,i}x_i.$$

Second, we characterize the above optimization problem by using the KKT condition:

Because (EC.3) is a linear program with a nonempty relative interior (Slater point exists: e.g., take $x_i = y_{m,i} = y_{s,i} = 0$ and $q_{p,i} = \lambda_i/\theta_i$), the Karush–Kuhn–Tucker (KKT) conditions are necessary and sufficient for optimality.

The Lagrangian of (EC.3) is

$$\begin{aligned} \mathcal{L} = & \sum_{i=1}^I w_i (\mu_{m,i} y_{m,i} + \mu_{s,i} y_{s,i}) + \alpha \left(1 - \sum_{i=1}^I x_i\right) + \beta \left((B-1) \sum_{i=1}^I x_i - \sum_{i=1}^I y_{m,i} \right) \\ & + \delta \left(B - \sum_{i=1}^I y_{s,i} - B \sum_{i=1}^I x_i \right) + \sum_{i=1}^I \phi_i (\lambda_i - \theta_i q_{p,i} - \mu_{p,i} x_i) \\ & + \sum_{i=1}^I \eta_i (\mu_{p,i} x_i - \mu_{m,i} y_{m,i} - \mu_{s,i} y_{s,i}) + \sum_{i=1}^I (\sigma_{x,i} x_i + \sigma_{m,i} y_{m,i} + \sigma_{s,i} y_{s,i} + \sigma_{p,i} q_{p,i}). \end{aligned}$$

The KKT conditions at an optimal primal–dual pair $(x_i^*, y_{m,i}^*, y_{s,i}^*, q_{p,i}^*; \alpha^*, \beta^*, \delta^*, \phi_i^*, \eta_i^*, \sigma_{x,i}^*, \sigma_{m,i}^*, \sigma_{s,i}^*, \sigma_{p,i}^*)$ are:

(i) *Stationarity.* For each i ,

$$\frac{\partial \mathcal{L}}{\partial y_{m,i}} = w_i \mu_{m,i} - \beta - \eta_i \mu_{m,i} + \sigma_{m,i} = 0, \quad (\text{EC.4})$$

$$\frac{\partial \mathcal{L}}{\partial y_{s,i}} = w_i \mu_{s,i} - \delta - \eta_i \mu_{s,i} + \sigma_{s,i} = 0, \quad (\text{EC.5})$$

$$\frac{\partial \mathcal{L}}{\partial x_i} = -\alpha + (B-1)\beta - B\delta - \phi_i \mu_{p,i} + \eta_i \mu_{p,i} + \sigma_{x,i} = 0, \quad (\text{EC.6})$$

$$\frac{\partial \mathcal{L}}{\partial q_{p,i}} = -\phi_i \theta_i + \sigma_{p,i} = 0. \quad (\text{EC.7})$$

(ii) *Complementary slackness.*

$$\alpha \left(1 - \sum_{j=1}^I x_j\right) = 0, \quad (\text{C1})$$

$$\beta \left(\sum_{j=1}^I y_{m,j} - (B-1) \sum_{j=1}^I x_j \right) = 0, \quad (\text{C2})$$

$$\delta \left(\sum_{j=1}^I y_{s,j} + B \sum_{j=1}^I x_j - B \right) = 0, \quad (\text{C3})$$

$$\eta_i (\mu_{m,i} y_{m,i} + \mu_{s,i} y_{s,i} - \mu_{p,i} x_i) = 0, \quad \forall i, \quad (\text{C4})$$

$$\sigma_{x,i} x_i = 0, \quad \sigma_{m,i} y_{m,i} = 0, \quad \sigma_{s,i} y_{s,i} = 0, \quad \sigma_{p,i} q_{p,i} = 0, \quad \forall i. \quad (\text{C5})$$

(iii) *Primal feasibility.* All constraints in (EC.3) hold.

(iv) *Dual feasibility.*

$$\alpha, \beta, \delta, \eta_i, \sigma_{x,i}, \sigma_{m,i}, \sigma_{s,i}, \sigma_{p,i} \geq 0, \quad \forall i, \quad \phi_i \in \mathbb{R}, \quad \forall i.$$

Third, starting from the characterization above, we can begin to construct a satisfying optimal solution.

Let

$$(x_i^*, y_{m,i}^*, y_{s,i}^*, q_{p,i}^*)$$

be any optimal solution of (EC.3) (hence also of (40)) and let $(\alpha^*, \beta^*, \delta^*, \phi_i^*, \eta_i^*, \dots)$ be a corresponding dual optimal solution satisfying the KKT conditions above.

Suppose that there exists an index i_0 such that the following constraint is *slack* at the optimum, i.e.,

$$\mu_{m,i_0} y_{m,i_0}^* + \mu_{s,i_0} y_{s,i_0}^* < \mu_{p,i_0} x_{i_0}^*. \quad (\text{EC.8})$$

Then, by complementary slackness (C4), we must have

$$\eta_{i_0}^* = 0.$$

We now show that, under the assumption $\gamma\tau \geq (B-1)/B$, we can *reallocate* prefill and decode occupancy across modes so that:

- the global capacity constraints remain feasible,
- the per-class decode completion rates $\mu_{m,i} y_{m,i}^* + \mu_{s,i} y_{s,i}^*$ remain unchanged for all i ,
- and the slack in (EC.8) is reduced, ultimately to 0,

without changing the objective value. This yields a new optimal solution in which the i_0 -th decode constraint is tight. By repeating the same procedure for every class with slack inequality (EC.8), we obtain an optimal solution with

$$\mu_{m,i} y_{m,i}^* + \mu_{s,i} y_{s,i}^* = \mu_{p,i} x_i^*, \quad \forall i,$$

which is equivalent to $q_{d,i}^* = 0$ in the original formulation.

For each i with $\mu_{m,i} y_{m,i}^* + \mu_{s,i} y_{s,i}^* < \mu_{p,i} x_i^*$ (i.e., $q_{d,i}^* > 0$ in the original variables), define the gap

$$\Delta_i := \frac{\mu_{p,i} x_i^* - \mu_{m,i} y_{m,i}^* - \mu_{s,i} y_{s,i}^*}{\mu_{p,i}} \geq 0.$$

Consider the modified active prefill and prefill queue

$$\tilde{x}_i := x_i^* - \Delta_i = \frac{\mu_{m,i} y_{m,i}^* + \mu_{s,i} y_{s,i}^*}{\mu_{p,i}}, \quad \tilde{q}_{p,i} := q_{p,i}^* + \frac{\mu_{p,i} \Delta_i}{\theta_i} = q_{p,i}^* + \frac{\mu_{p,i} x_i^* - \mu_{m,i} y_{m,i}^* - \mu_{s,i} y_{s,i}^*}{\theta_i}.$$

(For classes where (EC.1) already holds with equality, set $\Delta_i = 0$ and $\tilde{x}_i = x_i^*$, $\tilde{q}_{p,i} = q_{p,i}^*$.)

By construction,

$$\begin{aligned} \mu_{p,i} \tilde{x}_i &= \mu_{p,i} (x_i^* - \Delta_i) = \mu_{p,i} x_i^* - (\mu_{p,i} x_i^* - \mu_{m,i} y_{m,i}^* - \mu_{s,i} y_{s,i}^*) \\ &= \mu_{m,i} y_{m,i}^* + \mu_{s,i} y_{s,i}^*. \end{aligned}$$

Hence the inequality(EC.8) becomes tight:

$$\mu_{m,i}y_{m,i}^* + \mu_{s,i}y_{s,i}^* = \mu_{p,i}\tilde{x}_i.$$

Moreover,

$$\begin{aligned} \lambda_i - \theta_i \tilde{q}_{p,i} - \mu_{p,i}\tilde{x}_i &= \lambda_i - \theta_i \left(q_{p,i}^* + \frac{\mu_{p,i}x_i^* - \mu_{m,i}y_{m,i}^* - \mu_{s,i}y_{s,i}^*}{\theta_i} \right) - \mu_{p,i}(x_i^* - \Delta_i) \\ &= (\lambda_i - \theta_i q_{p,i}^* - \mu_{p,i}x_i^*) + \mu_{m,i}y_{m,i}^* + \mu_{s,i}y_{s,i}^* - \mu_{m,i}y_{m,i}^* - \mu_{s,i}y_{s,i}^* \\ &= 0, \end{aligned}$$

so the first constraint remains satisfied. Nonnegativity holds because $x_i^* \geq \Delta_i$ is equivalent to $\mu_{p,i}x_i^* \geq \mu_{m,i}y_{m,i}^* + \mu_{s,i}y_{s,i}^*$, which already holds.

At this stage we have decreased $\sum_i x_i$ to $\sum_i \tilde{x}_i = \sum_i x_i^* - \sum_i \Delta_i$ while keeping $(y_{m,i}, y_{s,i})$ unchanged. Thus:

- The constraint $\sum_i x_i \leq 1$ remains feasible;
- The solo-decode capacity $\sum_i y_{s,i} \leq B(1 - \sum_i x_i)$ becomes *less* restrictive, since the right-hand side increases when $\sum_i x_i$ decreases;
- The mixed-decode capacity

$$\sum_{i=1}^I y_{m,i} \leq (B-1) \sum_{i=1}^I x_i$$

may become *more* restrictive, because the right-hand side decreases as $\sum_i x_i$ decreases.

If the mixed-capacity constraint is still satisfied with the new \tilde{x}_i , we are done with this step. Otherwise, let

$$\Delta := \sum_{i=1}^I y_{m,i}^* - (B-1) \sum_{i=1}^I \tilde{x}_i > 0$$

denote the gap of mixed-mode occupancy between the new capacity.

We now show that, thanks to the assumption $\gamma\tau \geq (B-1)/B$, we can reduce the total mixed occupancy by Δ while increasing solo occupancy accordingly, without violating capacity and without changing decode completion rates.

For each i , define perturbations $(\delta y_{m,i}, \delta y_{s,i})$ such that

$$\sum_{i=1}^I \delta y_{m,i} = -\Delta, \quad \delta y_{m,i} \leq 0, \quad \forall i.$$

We choose $\delta y_{s,i}$ to preserve per-class decode completion rates:

$$\mu_{m,i} \delta y_{m,i} + \mu_{s,i} \delta y_{s,i} = 0, \quad \forall i. \tag{EC.9}$$

Equation (EC.9) implies

$$\delta y_{s,i} = -\frac{\mu_{m,i}}{\mu_{s,i}} \delta y_{m,i}.$$

From the speed abstraction, we have

$$\mu_{m,i} = \frac{1}{D_i \tau}, \quad \mu_{s,i} = \frac{\gamma}{D_i}, \quad \Rightarrow \quad \frac{\mu_{m,i}}{\mu_{s,i}} = \frac{1}{\gamma \tau},$$

which is *independent* of i . Hence

$$\sum_{i=1}^I \delta y_{s,i} = -\frac{1}{\gamma \tau} \sum_{i=1}^I \delta y_{m,i} = \frac{\Delta}{\gamma \tau}. \quad (\text{EC.10})$$

Thus we have decreased total mixed occupancy by Δ and increased total solo occupancy by $\Delta/(\gamma \tau)$.

Mixed-decode capacity. By construction, the new mixed occupancy is

$$\sum_{i=1}^I (y_{m,i}^* + \delta y_{m,i}) = \sum_{i=1}^I y_{m,i}^* - \Delta = (B-1) \sum_{i=1}^I \tilde{x}_i,$$

so the mixed capacity is now exactly tight and hence feasible.

Solo-decode capacity. Let $X^* := \sum_{i=1}^I x_i^*$ and $\tilde{X} := \sum_{i=1}^I \tilde{x}_i$. Then

$$\tilde{X} = X^* - \sum_{i=1}^I \Delta_i.$$

Originally, the solo capacity constraint was

$$\sum_{i=1}^I y_{s,i}^* \leq B(1 - X^*).$$

Conducting the steps above, with $y_{s,i}$ unchanged and x_i replaced by \tilde{x}_i , we had

$$\sum_{i=1}^I y_{s,i}^* \leq B(1 - X^*) \leq B(1 - \tilde{X}),$$

so solo capacity was slack. After applying $(\delta y_{m,i}, \delta y_{s,i})$, the new solo occupancy becomes, using (EC.10),

$$\sum_{i=1}^I (y_{s,i}^* + \delta y_{s,i}) = \sum_{i=1}^I y_{s,i}^* + \frac{\Delta}{\gamma \tau}.$$

We want to ensure

$$\sum_{i=1}^I y_{s,i}^* + \frac{\Delta}{\gamma \tau} \leq B(1 - \tilde{X}). \quad (\text{EC.11})$$

Observe that

$$\begin{aligned} \Delta &= \sum_{i=1}^I y_{m,i}^* - (B-1) \sum_{i=1}^I \tilde{x}_i \\ &= \left[\sum_{i=1}^I y_{m,i}^* - (B-1) \sum_{i=1}^I x_i^* \right] + (B-1) \sum_{i=1}^I (x_i^* - \tilde{x}_i). \end{aligned}$$

By feasibility of the original solution, $\sum_i y_{m,i}^* \leq (B-1) \sum_i x_i^*$, hence

$$\Delta \leq (B-1) \sum_{i=1}^I (x_i^* - \tilde{x}_i) = (B-1) \sum_{i=1}^I \Delta_i.$$

Therefore,

$$\frac{\Delta}{\gamma\tau} \leq \frac{B-1}{\gamma\tau} \sum_{i=1}^I \Delta_i.$$

Using the assumption $\gamma\tau \geq (B-1)/B$, we obtain

$$\frac{B-1}{\gamma\tau} \leq B, \quad \Rightarrow \quad \frac{\Delta}{\gamma\tau} \leq B \sum_{i=1}^I \Delta_i.$$

Finally, note that

$$B(1 - \tilde{X}) = B(1 - X^*) + B \sum_{i=1}^I \Delta_i.$$

Combining these inequalities,

$$\begin{aligned} \sum_{i=1}^I y_{s,i}^* + \frac{\Delta}{\gamma\tau} &\leq \sum_{i=1}^I y_{s,i}^* + B \sum_{i=1}^I \Delta_i \\ &\leq B(1 - X^*) + B \sum_{i=1}^I \Delta_i \\ &= B(1 - \tilde{X}), \end{aligned}$$

which is exactly (EC.11). Thus the solo-decode capacity constraint remains feasible after the reallocation.

By (EC.9), for each i ,

$$\mu_{m,i}(y_{m,i}^* + \delta y_{m,i}) + \mu_{s,i}(y_{s,i}^* + \delta y_{s,i}) = \mu_{m,i}y_{m,i}^* + \mu_{s,i}y_{s,i}^*.$$

Hence all class-level decode completion rates are unchanged, and the objective

$$\sum_{i=1}^I w_i (\mu_{m,i}y_{m,i} + \mu_{s,i}y_{s,i})$$

remains the same. Consequently, we have constructed a new feasible solution with the *same* objective value, but with mixed occupancy reduced to exactly match the capacity available under the updated \tilde{x}_i .

Moreover, for any class i with initial slack in (EC.1), we now have

$$\mu_{m,i}y_{m,i}^* + \mu_{s,i}y_{s,i}^* = \mu_{p,i}\tilde{x}_i,$$

i.e., its decode inequality is tight.

Applying the above procedure to each class i whose decode inequality (EC.1) is slack at an optimal solution, we obtain another optimal solution of (EC.3) such that

$$\mu_{m,i}y_{m,i}^* + \mu_{s,i}y_{s,i}^* = \mu_{p,i}x_i^*, \quad \forall i.$$

Returning to the original formulation (40), this is equivalent (via (EC.2)) to

$$q_{d,i}^* = 0, \quad \forall i.$$

Thus the steady-state fluid LP admits an optimal solution in which the decode buffer is empty in the fluid limit, completing the proof. \square

EC.2. Proofs of Theorem 1

Proof Fix $T > 0$. We denote the scaled processes as $\bar{W}^n := W^n/n$. From the model above, we have:

$$\begin{aligned} A_i^n(t) &= N_{a,i}(\lambda_i n t), & B_{p,i}^n(t) &= N_{b_p,i} \left(\int_0^t \theta_i Q_{p,i}^n(s) ds \right), & B_{d,i}^n(t) &= N_{b_d,i} \left(\int_0^t \theta_i Q_{d,i}^n(s) ds \right), \\ D_{p,i}^n(t) &= N_{d_p,i} \left(\int_0^t \mu_{p,i} X_i^n(s) ds \right), & D_{d,m,i}^n(t) &= N_{d_m,i} \left(\int_0^t \mu_{m,i} Y_{m,i}^n(s) ds \right), \\ D_{d,s,i}^n(t) &= N_{d_s,i} \left(\int_0^t \mu_{s,i} Y_{s,i}^n(s) ds \right) \end{aligned}$$

First we prove the tightness. The GPU capacity implies that $0 \leq X^n \leq n$, $0 \leq Y_m^n \leq (B-1)X^n \leq (B-1)n$, $0 \leq Y_s^n \leq B(n - X^n) \leq Bn$, so the time-changes for $D_{p,i}^n, D_{d,m,i}^n, D_{d,s,i}^n$ are bounded. Since $Q_{p,i}^n(s) \leq Q_{p,i}^n(0) + A_i^n(s)$ and $Q_{d,i}^n(s) \leq Q_{d,i}^n(0) + D_{p,i}^n(s)$, we have:

$$\frac{1}{n} \int_0^t \theta_{p,i} Q_{p,i}^n(s) ds \leq \theta_{p,i} \left(t \bar{Q}_{p,i}^n(0) + t \bar{A}_i^n(t) \right), \quad \frac{1}{n} \int_0^t \theta_{d,i} Q_{d,i}^n(s) ds \leq \theta_{d,i} \left(t \bar{Q}_{d,i}^n(0) + t \bar{D}_{p,i}^n(t) \right),$$

hence these time-changes are stochastically bounded on $[0, T]$. Controls are nondecreasing and satisfy

$$U_{p,i}^n(t) \leq Q_{p,i}^n(0) + A_i^n(t), \quad U_{d,m,i}^n(t) + U_{d,s,i}^n(t) \leq Q_{d,i}^n(0) + D_{p,i}^n(t),$$

and structural transfers obey $M_{m \rightarrow s}^n \leq B D_p^n$, $M_{s \rightarrow m}^n \leq B U_p^n$. By the FSLLN for Poisson processes, the family $\{\bar{A}_i^n\}_n$ is tight.

Then by the random time-change theorem, together with the bounds above, we have the tightness of all other time-changed Poisson coordinates. Also, by Billingsley's criterion for nondecreasing processes, the full vector \bar{X}^n is tight in $\mathbb{D}([0, T], \mathbb{R}^d)$ under the J_1 -topology.

Second, we focus on the subsequence convergence and continuity. By tightness, we may pick a subsequence $\bar{X}^n \Rightarrow \bar{X}$. By Skorokhod representation theorem, we may assume a.s. convergence u.o.c. on $[0, T]$. Since all jumps are $\frac{1}{n}$, the limit \bar{X} is a.s. continuous.

Finally, we prove that the stochastic processes satisfy the fluid model constraints. The FSLLN gives $\bar{A}_i^n \rightarrow a_i(t) = \lambda_i t$ u.o.c. For the time-changed Poisson processes, if $L^n(t) := \frac{1}{n} \int_0^t \ell^n(s) ds \rightarrow L(t)$ u.o.c., then $N(L^n(\cdot)n)/n \rightarrow L(\cdot)$ u.o.c. Hence

$$\bar{B}_{p,i}^n \Rightarrow b_{p,i}(t) = \int_0^t \theta_{p,i} q_{p,i}(s) ds, \quad \bar{B}_{d,i}^n \Rightarrow b_{d,i}(t) = \int_0^t \theta_{d,i} q_{d,i}(s) ds,$$

$$\begin{aligned}\bar{D}_{p,i}^n \Rightarrow d_{p,i}(t) &= \int_0^t \mu_{p,i} x_i(s) ds, & \bar{D}_{d,m,i}^n \Rightarrow d_{d,m,i}(t) &= \int_0^t \mu_{m,i} y_{m,i}(s) ds, \\ \bar{D}_{d,s,i}^n \Rightarrow d_{d,s,i}(t) &= \int_0^t \mu_{s,i} y_{s,i}(s) ds.\end{aligned}$$

Taking limits in the flow balances yields (24)–(28) with (29)–(32).

The remaining inequalities are easy to check since the prelimit constraints satisfy:

$$0 \leq \bar{X}^n \leq 1, \quad 0 \leq \bar{Y}_m^n \leq (B-1)\bar{X}^n, \quad 0 \leq \bar{Y}_s^n \leq B(1-\bar{X}^n),$$

and the inequalities are preserved under the limit. Hence (33)–(35) hold. \square

EC.3. Proof of Optimality of Gate and Route Policy

EC.3.1. Proof of prefill convergence in Theorem 2

We focus on the LP solutions where $q_{p,i}^* > 0$. The proof for $q_{p,i}^* = 0$ is straightforward.

LEMMA EC.1. *There exists time t_0 , such that for $t \geq t_0$, we have: $x_i(t) \leq x_i^*$ for all i .*

Proof Suppose not, then for any t_0 , there exists t_1 , such that $x_i(t_1) > x_i^*$. Take $\tilde{t} = \sup\{\tilde{t} < t_0 : x_i(\tilde{t}) \leq x_i^*\}$. Then it is clear that $x_i(\tilde{t}) = x_i^*$ and $x_i(t) > x_i^*$ for $t \in (\tilde{t}, t_1]$.

But by our policy, if $x_i(t) > x_i^*$, we must have $x_i'(t) < 0$. This means that $x_i(t_1) = x_i(\tilde{t}) + \int_{\tilde{t}}^{t_1} dx_i(t) dt < x_i(\tilde{t}) = x_i^*$, contradiction. \square

LEMMA EC.2. *In the fluid system, there exists time t_i , such that $q_{p,i}(t_i) > 0$. And $q_{p,i}(t) > 0$ for all $t \geq t_i$ and $i \in I$.*

Proof Suppose not, then for some i , we have $q_{p,i}(t) = 0$ for all t . In which case, $x_i(t) \leq x_i^*$. By the balance equation, we have:

$$\begin{aligned}\lambda_i &= \theta_i q_{p,i}(t) + \mu_{p,i} x_i(t) \\ &= \mu_{p,i} x_i(t) \\ &\leq \mu_{p,i} x_i^*\end{aligned}$$

However, by the structure of the LP solution, $\lambda_i = \theta_i q_{p,i}^* + \mu_{p,i} x_i^*$, contradiction.

If there exists some index i and \tilde{t}_i such that $q_{p,i}(\tilde{t}_i) = 0$, we can take a small enough interval $(\tilde{t} - \epsilon, \tilde{t})$ such that $q_{p,i}(t)$ is decreasing in the interval and $q_{p,i}(t) < q_{p,i}^*$. Since in the interval $q_{p,i}(t) > 0$, $x_i(t) = x_i^*$. Then by the balance equations, we have:

$$\dot{q}_{p,i}(t) = \theta_i [q_{p,i}^* - q_{p,i}(t)] > 0$$

contradiction! Then $q_{p,i}(t) > 0$ for all $t \geq t_i$ and $i \in I$. \square

LEMMA EC.3. *Under our policy, $q_{p,i}(t) \rightarrow q_{p,i}^*$ and $x_i(t) \rightarrow x_i^*$ as $t \rightarrow \infty$.*

Proof By the above lemma, we can assume that $q_{p,i}(t) > 0$. $x_i(t) = x_i^*$ as a result. Then the convergence of $x_i(t)$ follows. By the balance equation,

$$\begin{aligned}\dot{q}_{p,i}(t) &= \lambda_i - \mu_{p,i}x_i^* - \theta_i q_{p,i}(t) \\ &= \theta_i(q_{p,i}^* - q_{p,i}(t))\end{aligned}$$

Solving the above ODE, the convergence of $q_{p,i}(t)$ follows in standard. \square

EC.3.2. Proof of decode convergence in Theorem 2

We now show that, under these conditions, the aggregate decode buffer $q_d(t) := \sum_{i=1}^I q_{d,i}(t)$ converges to zero along any fluid trajectory of the gate-and-route policy.

LEMMA EC.4. *Consider the fluid model under the gate-and-route policy with the static GPU partition induced by the LP solution: a fraction $x^* := \sum_{i=1}^I x_i^*$ of GPUs are mixed and a fraction $1 - x^*$ are solo. Let*

$$y_m(t) := \sum_{i=1}^I y_{m,i}(t), \quad y_s(t) := \sum_{i=1}^I y_{s,i}(t).$$

If at some regular time t we have $q_d(t) > 0$, then decode capacity is fully utilized:

$$y_m(t) = x^*(B - 1), \quad y_s(t) = (1 - x^*)B.$$

Proof By construction of the static planning step, exactly a fraction x^* of the GPUs are permanently assigned to the mixed group, and the remaining fraction $1 - x^*$ to the solo group. Each mixed GPU holds at most $(B - 1)$ decodes, while each solo GPU holds at most B decodes. After fluid scaling by n , the maximal aggregate decode occupancies are

$$y_m(t) \leq x^*(B - 1), \quad y_s(t) \leq (1 - x^*)B.$$

The gate-and-route decode router is work-conserving: as long as the global decode buffer is nonempty (i.e., $q_d(t) > 0$), any free decode slot in either group is immediately filled with a waiting job. Hence no decode slot can be idle whenever $q_d(t) > 0$. Therefore,

$$y_m(t) = x^*(B - 1), \quad y_s(t) = (1 - x^*)B,$$

whenever $q_d(t) > 0$. \square

We next introduce a Lyapunov function that measures the total remaining decode work in “mixed-mode time units.”

DEFINITION EC.1 (WEIGHTED DECODE WORK). Define

$$W_d(t) := \sum_{i=1}^I \frac{q_{d,i}(t) + y_{m,i}(t) + y_{s,i}(t)}{\mu_{m,i}}.$$

Here $1/\mu_{m,i}$ is the mean decode service time of a type- i job in mixed mode. Thus $W_d(t)$ is the total remaining decode work expressed in mixed-mode time units.

We assume the mode speed ratio

$$\mu_{s,i} = \gamma\tau \mu_{m,i} =: \kappa \mu_{m,i} \quad \text{for all } i \in \{1, \dots, I\},$$

with $\kappa = \gamma\tau > 0$ constant across classes.

LEMMA EC.5. *Along any fluid trajectory, for $t \geq 0$,*

$$\dot{W}_d(t) = \sum_{i=1}^I \frac{\mu_{p,i}}{\mu_{m,i}} x_i(t) - (y_m(t) + \kappa y_s(t)) - \sum_{i=1}^I \frac{\theta_i}{\mu_{m,i}} q_{d,i}(t). \quad (\text{EC.12})$$

Proof We use the fluid balance equations and the random time-change representation for arrivals, completions, and abandonments; all fluid coordinates are absolutely continuous, so their derivatives exist.

Arrivals into decode. For class i , prefills complete at rate $\mu_{p,i}x_i(t)$. Each completion creates one decode job of that class, with mixed-mode work $1/\mu_{m,i}$. Hence the instantaneous inflow into W_d from class i is

$$\mu_{p,i}x_i(t) \cdot \frac{1}{\mu_{m,i}},$$

and summing over i yields the first term on the right-hand side of (EC.12).

Service in mixed mode. For class i , mixed decodes are in service at rate $\mu_{m,i}y_{m,i}(t)$, each completion removing $1/\mu_{m,i}$ units of W_d . Thus the mixed-mode drain is

$$\sum_{i=1}^I \mu_{m,i}y_{m,i}(t) \cdot \frac{1}{\mu_{m,i}} = \sum_{i=1}^I y_{m,i}(t) = y_m(t).$$

Service in solo mode. Using $\mu_{s,i} = \kappa\mu_{m,i}$, solo-mode completions for class i occur at rate $\mu_{s,i}y_{s,i}(t)$, each also removing $1/\mu_{m,i}$ units of work. Thus

$$\sum_{i=1}^I \mu_{s,i}y_{s,i}(t) \cdot \frac{1}{\mu_{m,i}} = \sum_{i=1}^I \kappa\mu_{m,i}y_{s,i}(t) \cdot \frac{1}{\mu_{m,i}} = \kappa \sum_{i=1}^I y_{s,i}(t) = \kappa y_s(t),$$

which gives the second term in (EC.12).

Abandonments. Class- i decode abandonments occur at rate $\theta_i q_{d,i}(t)$. Each abandonment removes $1/\mu_{m,i}$ units of W_d . Thus the total drain from abandonments is

$$\sum_{i=1}^I \theta_i q_{d,i}(t) \cdot \frac{1}{\mu_{m,i}},$$

which yields the third term in (EC.12) with a minus sign.

Combining these three contributions establishes (EC.12). □

LEMMA EC.6. Assume that under the gate-and-route policy, the prefill occupancies satisfy $x_i(t) \rightarrow x_i^*$ as $t \rightarrow \infty$ for each i , and let $x^* := \sum_{i=1}^I x_i^*$. Define the per-GPU decode capacity (in mixed-mode time units)

$$C^* := \kappa(1 - x^*)B + x^*(B - 1).$$

Then there exists $T_0 < \infty$ such that for all $t \geq T_0$,

$$q_d(t) > 0 \implies \dot{W}_d(t) \leq - \sum_{i=1}^I \frac{\theta_i}{\mu_{m,i}} q_{d,i}(t).$$

Proof Since $x_i(t) \rightarrow x_i^*$, we have

$$A(t) := \sum_{i=1}^I \frac{\mu_{p,i}}{\mu_{m,i}} x_i(t) \longrightarrow A^* := \sum_{i=1}^I \frac{\mu_{p,i}}{\mu_{m,i}} x_i^*.$$

We claim that $A^* \leq C^*$. Indeed, by Proposition 1 we can choose an LP-optimal solution with $q_{d,i}^* = 0$ for all i . Summing the LP decode flow-balance constraints and using $\mu_{s,i} = \kappa\mu_{m,i}$ gives

$$A^* = \sum_{i=1}^I y_{m,i}^* + \kappa \sum_{i=1}^I y_{s,i}^* \leq x^*(B - 1) + \kappa(1 - x^*)B = C^*,$$

where the inequality is exactly the LP mixed/solo decode capacity constraints.

By Lemma EC.1, there exists t_0 such that $x_i(t) \leq x_i^*$ for all i and all $t \geq t_0$. Hence

$$A(t) \leq A^* \leq C^*, \quad t \geq t_0.$$

By Lemma EC.4, if $q_d(t) > 0$ then decode capacity is fully utilized and $y_m(t) = x^*(B - 1)$, $y_s(t) = (1 - x^*)B$. Substituting into (EC.12) gives, for such t ,

$$\dot{W}_d(t) = A(t) - \left[x^*(B - 1) + \kappa(1 - x^*)B \right] - \sum_{i=1}^I \frac{\theta_i}{\mu_{m,i}} q_{d,i}(t).$$

Then by using the inequality $A(t) \leq A^* \leq C^*$, we have:

$$\dot{W}_d(t) \leq - \sum_{i=1}^I \frac{\theta_i}{\mu_{m,i}} q_{d,i}(t),$$

as claimed. □

We are now ready to show that the decode buffer vanishes in the fluid limit.

PROPOSITION EC.1. Assume $\theta_i > 0$ for all i , and that under the gate-and-route policy the prefill occupancies satisfy $x_i(t) \rightarrow x_i^*$ as $t \rightarrow \infty$, with the associated capacity constant $C^* := \kappa(1 - x^*)B + x^*(B - 1)$. Then along any fluid trajectory,

$$\lim_{t \rightarrow \infty} q_d(t) = 0.$$

Proof Define

$$\underline{\theta} := \min_{1 \leq i \leq I} \frac{\theta_i}{\mu_{m,i}} > 0,$$

using the assumption $\theta_i > 0$ for all i . Then

$$\sum_{i=1}^I \frac{\theta_i}{\mu_{m,i}} q_{d,i}(t) \geq \underline{\theta} \sum_{i=1}^I q_{d,i}(t) = \underline{\theta} q_d(t).$$

Then by Lemma EC.6, we have:

$$\dot{W}_d(t) \leq -\underline{\theta} q_d(t). \quad (\text{EC.13})$$

which means that $W_d(t)$ is nonincreasing whenever $q_d(t) \geq 0$.

We now argue by contradiction. Suppose $q_d(t)$ does not converge to zero. Then there exists $\delta > 0$ and a sequence $t_k \rightarrow \infty$ such that $q_d(t_k) \geq \delta$ for all k . Because of the Lipschitz property of the fluid processes, there exists $\eta > 0$ and subintervals $[t'_k, t''_k]$ with

$$t'_k \leq t_k \leq t''_k, \quad t''_k - t'_k \geq \eta,$$

Then for all $t \in [t'_k, t''_k]$, (EC.13) gives $\dot{W}_d(t) \leq -\underline{\theta} \delta/2 =: -c < 0$. Take the integral of the inequality, we have:

$$W_d(t''_k) - W_d(t'_k) \leq -c(t''_k - t'_k) \leq -c\eta < 0,$$

which means that in each interval determined by the sequence t_k , the decrease of $W_d(t)$ is at least $c\eta$, which contradicts the fact that $W_d(t) \geq 0$.

Therefore, our assumption is false, and we must have

$$\lim_{t \rightarrow \infty} q_d(t) = 0.$$

□

Proof of Theorem 2 Combining Proposition EC.1 with the previously established convergence of the prefill occupancies $x_i(t) \rightarrow x_i^*$ and the fixed decode capacities in Lemma EC.4, we conclude that the fluid occupancies converge to the LP-optimal point $(x_i^*, y_{m,i}^*, y_{s,i}^*)_{i \in [I]}$, and the corresponding per-GPU reward attains the LP optimum. This is the key ingredient in the proof of Theorem 2 (asymptotic optimality of the gate-and-route policy).

□

EC.4. Proof of the Optimality of the Prioritize-and-Route Policy

EC.4.1. Proof of Theorem 3

Proof Let $\tilde{\pi}^{n,*}$ be the prioritize-and-route policy defined in Section 4.2, parameterized by an optimal solution of the steady-state fluid LP (42).

Under separate charging, the steady-state objective (42) by which the reward only depends on aggregate occupancies $(\sum_i x_i, \sum_i y_{m,i}, \sum_i y_{s,i})$. What is left is to prove that the prioritize-and-route policy reaches the maximal workload of the system.

The prioritize-and-route policy is work-conserving at the prefill side, so the prefill occupancy is always full. The priority index $\phi_i = D_i/P_i$ is proportional to the decode-work generation rate per unit prefill occupancy, since

$$\frac{\mu_{p,i}}{\mu_{m,i}} = \frac{C/(\tau P_i)}{1/(\tau D_i)} = C \frac{D_i}{P_i} = C \phi_i.$$

Therefore, among all feasible ways to allocate a given amount of prefill occupancy across classes, the static priority rule maximizes the instantaneous inflow of downstream decode workload. This ensures that the aggregate decode occupancies are maximized whenever sufficient workload is available; if workload is insufficient, no policy can keep more decode capacity busy.

Consequently, the limiting fluid reward under the prioritize-and-route policy achieves the optimal value \tilde{R}^* of the steady-state LP (42), which yields the asymptotic optimality of the policy, i.e.

$$\liminf_{T \rightarrow \infty} \liminf_{n \rightarrow \infty} \tilde{R}_n(T; \tilde{\pi}^{n,*}) = \tilde{R}^*.$$

which is exactly Theorem 3. □

EC.5. Proofs of the Optimality of the SLI-Aware Policy

In this section, we prove the optimality of the SLI-aware policy. Since the SLI-aware policy deals with the prefill stage the same as the Gate-and-Route policy, we will only prove the optimality of the SLI-aware policy for the decode stage. The optimality of the prefill stage follows from the optimality of the Gate-and-Route policy.

EC.5.1. Convergence of the decode occupancies under the SLI-aware policy

In this subsection, we focus on the convergence of the decode occupancies under the SLI-aware policy.

PROPOSITION EC.2. *Fix an optimal solution $(x_i^*, y_{m,i}^*, y_{s,i}^*, q_{p,i}^*, q_{d,i}^*)_{i \in \mathcal{I}}$ of the SLI-aware steady-state program (49) with $q_{d,i}^* = 0$ for all i . Let $x^* := \sum_{i=1}^I x_i^*$ and consider the corresponding static planning of the system. Denote $q_{d,m,i}(t)$ and $q_{d,s,i}(t)$ for the mixed and solo decode buffers under the virtual buffer split of Section 5.1 respectively. Then along any fluid trajectory of the SLI-aware policy,*

$$\lim_{t \rightarrow \infty} y_{m,i}(t) = y_{m,i}^*, \quad \lim_{t \rightarrow \infty} y_{s,i}(t) = y_{s,i}^*, \quad \lim_{t \rightarrow \infty} q_{d,s,i}(t) = 0, \quad \lim_{t \rightarrow \infty} q_{d,m,i}(t) = 0, \quad \forall i \in \mathcal{I}.$$

Proof Denote the total mixed and solo decode capacities as

$$C_m := x^*(B-1), \quad C_s := (1-x^*)B.$$

First we show that the decode buffer converges to zero as $t \rightarrow \infty$. Define the weighted decode work:

$$W_m(t) := \sum_{i=1}^I \frac{q_{d,m,i}(t) + y_{m,i}(t)}{\mu_{m,i}}, \quad W_s(t) := \sum_{i=1}^I \frac{q_{d,s,i}(t) + y_{s,i}(t)}{\mu_{s,i}}.$$

Take the derivative of the formulas above, we have:

$$\dot{W}_m(t) = \sum_{i=1}^I \frac{(1-p_{s,i})\mu_{p,i}}{\mu_{m,i}} x_i(t) - y_m(t) - \sum_{i=1}^I \frac{\theta_i}{\mu_{m,i}} q_{d,m,i}(t), \quad (\text{EC.14})$$

$$\dot{W}_s(t) = \sum_{i=1}^I \frac{p_{s,i}\mu_{p,i}}{\mu_{s,i}} x_i(t) - y_s(t) - \sum_{i=1}^I \frac{\theta_i}{\mu_{s,i}} q_{d,s,i}(t). \quad (\text{EC.15})$$

Here $y_m(t) := \sum_i y_{m,i}(t)$ and $y_s(t) := \sum_i y_{s,i}(t)$.

Whenever $q_{d,m}(t) > 0$, the mixed group is work-conserving, so all mixed decode slots are busy and $y_m(t) = C_m$; similarly, $q_{d,s}(t) > 0$ implies $y_s(t) = C_s$. Since $x_i(t) \rightarrow x_i^*$ and the LP feasibility constraints include $\sum_i y_{m,i}^* \leq C_m$ and $\sum_i y_{s,i}^* \leq C_s$, the constants

$$A_m^* := \sum_{i=1}^I \frac{(1-p_{s,i})\mu_{p,i}}{\mu_{m,i}} x_i^*, \quad A_s^* := \sum_{i=1}^I \frac{p_{s,i}\mu_{p,i}}{\mu_{s,i}} x_i^*$$

satisfy $A_m^* \leq C_m$ and $A_s^* \leq C_s$, since they are both the admission limit of the policy which should be less than the total decode capacity. Therefore,

$$q_{d,m}(t) > 0 \implies \dot{W}_m(t) \leq -\underline{\theta}_m q_{d,m}(t), \quad q_{d,s}(t) > 0 \implies \dot{W}_s(t) \leq -\underline{\theta}_s q_{d,s}(t),$$

where $\underline{\theta}_m := \min_i \theta_i / \mu_{m,i} > 0$ and $\underline{\theta}_s := \min_i \theta_i / \mu_{s,i} > 0$. The same contradiction argument as in Proposition EC.1 implies

$$\lim_{t \rightarrow \infty} q_{d,m}(t) = 0, \quad \lim_{t \rightarrow \infty} q_{d,s}(t) = 0.$$

Now we are ready to show the convergence of the decode occupancies. By our policy, we can compute the instant admission rates of the mixed and solo decode pools:

$$\dot{u}_{d,m,i}(t) = (1-p_{s,i})(\mu_{p,i}x_i(t) - \theta_i q_{d,s,i}(t)), \quad \dot{u}_{d,s,i}(t) = p_{s,i}(\mu_{p,i}x_i(t) - \theta_i q_{d,m,i}(t))$$

Take the derivative of the decode occupancies of the balance equations, we have:

$$\dot{y}_{m,i}(t) = (1-p_{s,i})\mu_{p,i}x_i(t) - \mu_{m,i}y_{m,i}(t), \quad \dot{y}_{s,i}(t) = p_{s,i}\mu_{p,i}x_i(t) - \mu_{s,i}y_{s,i}(t),$$

Since $x_i(t) \rightarrow x_i^*$ and $q_{d,m,i}(t) \rightarrow 0$ and $q_{d,s,i}(t) \rightarrow 0$ and the LP feasibility condition:

$$\mu_{p,i}x_i^* = \mu_{m,i}y_{m,i}^* + \mu_{s,i}y_{s,i}^*$$

then for any $\epsilon > 0$, there exists T_ϵ such that for all $t \geq T_\epsilon$,

$$\begin{aligned} |\dot{\mu}_{d,s,i}(t) - \mu_{s,i}y_{s,i}^*| &\leq \epsilon \\ |\dot{\mu}_{d,m,i}(t) - \mu_{m,i}y_{m,i}^*| &\leq \epsilon \end{aligned}$$

Then we have:

$$\begin{aligned} \mu_{m,i}y_{m,i}^* - \epsilon &\leq \dot{y}_{s,i}(t) + \mu_{m,i}y_{m,i}(t) \leq \mu_{m,i}y_{m,i}^* + \epsilon \\ \mu_{s,i}y_{s,i}^* - \epsilon &\leq \dot{y}_{s,i}(t) + \mu_{s,i}y_{s,i}(t) \leq \mu_{s,i}y_{s,i}^* + \epsilon \end{aligned}$$

Solving the above inequalities and let $\epsilon \rightarrow 0$ and $t \rightarrow \infty$, we have:

$$\lim_{t \rightarrow \infty} y_{m,i}(t) = y_{m,i}^*, \quad \lim_{t \rightarrow \infty} y_{s,i}(t) = y_{s,i}^*, \quad \forall i \in \mathcal{I}.$$

which means that the decode occupancies converge to the LP targets. \square

EC.6. General SLI-aware Policy

In this section, we analyze the general SLI-aware setting under mild regularity on the optimization problem (bounded penalties, etc.), in which case the SLI-aware LP may admit an optimal solution with $q_{d,i}^* > 0$ for some i .

General SLI-aware Policy Fix an optimal solution of the SLI-aware LP $(x_i^*, y_{m,i}^*, y_{s,i}^*, q_{p,i}^*, q_{d,i}^*)_{i \in \mathcal{I}}$, allowing $q_{d,i}^* > 0$. As in Section 5.1, we partition GPUs into $G_{\text{mix}}^{(n)}$ and $G_{\text{solo}}^{(n)}$ according to the static planning. We split the decode buffer into mixed buffer and solo buffer and track the queue lengths $Q_{d,m,i}^n(t)$ and $Q_{d,s,i}^n(t)$ accordingly. Upon each class- i prefill completion, we route the resulting decode job to the solo buffer with probability $p_{s,i}$ and to the mixed buffer with probability $1 - p_{s,i}$.

We set the pool-selection probabilities

$$p_{s,i} := \begin{cases} \frac{\mu_{s,i}y_{s,i}^*}{\mu_{m,i}y_{m,i}^* + \mu_{s,i}y_{s,i}^*}, & \text{if } \mu_{m,i}y_{m,i}^* + \mu_{s,i}y_{s,i}^* > 0, \\ 1, & \text{otherwise.} \end{cases}$$

When $q_{d,i}^* > 0$, the LP flow-balance implies

$$\mu_{m,i}y_{m,i}^* + \mu_{s,i}y_{s,i}^* = \mu_{p,i}x_i^* - \theta_i q_{d,i}^*,$$

so the definition of $p_{s,i}$ above can be rewritten as

$$p_{s,i} = \frac{\mu_{s,i}y_{s,i}^*}{\mu_{p,i}x_i^* - \theta_i q_{d,i}^*},$$

i.e., $p_{s,i}$ is the solo share of the *net* decode completion rate. Accordingly, we have the consistent pool-level queue split

$$q_{d,s,i}^* := p_{s,i} q_{d,i}^*, \quad q_{d,m,i}^* := (1 - p_{s,i}) q_{d,i}^*.$$

We additionally define pool class selection weights (with the convention $0/0 := 0$)

$$\varpi_{m,i} := \frac{\mu_{m,i} y_{m,i}^*}{\sum_{j \in \mathcal{I}} \mu_{m,j} y_{m,j}^*}, \quad \varpi_{s,i} := \frac{\mu_{s,i} y_{s,i}^*}{\sum_{j \in \mathcal{I}} \mu_{s,j} y_{s,j}^*}.$$

Different from the baseline SLI-aware policy, in the general SLI-aware policy, we introduce a pool class selection rule within each pool.

(i) Solo-pool class selection. Whenever a solo decode slot becomes available and $\sum_i Q_{d,s,i}^n(t^-) > 0$, the decode scheduler selects a class according to the weights $\{\varpi_{s,i}\}_{i \in \mathcal{I}}$ restricted to currently nonempty solo decode buffer:

$$\mathbb{P}(\text{select class } i \mid \{Q_{d,s,j}^n(t^-)\}_j) = \frac{\varpi_{s,i} \mathbf{1}\{Q_{d,s,i}^n(t^-) > 0\}}{\sum_{j \in \mathcal{I}} \varpi_{s,j} \mathbf{1}\{Q_{d,s,j}^n(t^-) > 0\}}.$$

It then routes the head-of-line job of that class to the freed solo slot. If the solo decode buffer is empty, the slot idles.

(ii) Mixed-pool class selection. Whenever a *mixed* decode slot becomes available and $\sum_i Q_{d,m,i}^n(t^-) > 0$, the scheduler selects a class according to $\{\varpi_{m,i}\}$ restricted to nonempty mixed decode buffer (defined analogously) and routes the corresponding head-of-line job to the freed mixed slot.

This policy is work-conserving within each pool and preserves FCFS within each class buffer. It differs from the baseline SLI-aware router only in the cross-class selection rule *within* the mixed/solo decode pools. Then we focus on the fluid dynamics induced by the general SLI-aware policy. We focus on the case where the SLI-aware LP solution satisfies $q_{d,i}^* > 0$ for all i and both decode capacity constraints bind. The proof for boundary cases (some $q_{d,i}^* = 0$, etc.) is analogous.

First, we introduce some conventions for the general SLI-aware policy. Let $q_{d,m,i}(t)$ and $q_{d,s,i}(t)$ denote the mixed/solo decode queue contents, and $u_{d,m,i}(t)$ and $u_{d,s,i}(t)$ denote the cumulative admissions into mixed/solo decode service. Abandonments occur from each pool buffer at rate θ_i , i.e.,

$$\dot{b}_{d,m,i}(t) = \theta_i q_{d,m,i}(t), \quad \dot{b}_{d,s,i}(t) = \theta_i q_{d,s,i}(t),$$

and the total queue length is $q_{d,i}(t) = q_{d,m,i}(t) + q_{d,s,i}(t)$.

LEMMA EC.7. *Fix a regular time t at which all fluid coordinates are differentiable. Define the aggregate decode admission and completion rates*

$$\dot{u}_{d,m}(t) := \sum_{i \in \mathcal{I}} \dot{u}_{d,m,i}(t), \quad \dot{u}_{d,s}(t) := \sum_{i \in \mathcal{I}} \dot{u}_{d,s,i}(t), \quad \dot{d}_{d,m}(t) := \sum_{i \in \mathcal{I}} \mu_{m,i} y_{m,i}(t), \quad \dot{d}_{d,s}(t) := \sum_{i \in \mathcal{I}} \mu_{s,i} y_{s,i}(t).$$

Define the pool-level decode backlogs

$$q_{d,m}(t) := \sum_{i \in \mathcal{I}} q_{d,m,i}(t), \quad q_{d,s}(t) := \sum_{i \in \mathcal{I}} q_{d,s,i}(t).$$

If the corresponding pool buffer is positive at time t , then that pool is work-conserving:

$$q_{d,m}(t) > 0 \implies \dot{u}_{d,m}(t) = \dot{d}_{d,m}(t), \quad q_{d,s}(t) > 0 \implies \dot{u}_{d,s}(t) = \dot{d}_{d,s}(t).$$

Moreover, under the within-pool class selection rule,

$$\begin{aligned} q_{d,m}(t) > 0 &\implies \dot{u}_{d,m,i}(t) = \dot{u}_{d,m}(t) \frac{\varpi_{m,i} \mathbf{1}\{q_{d,m,i}(t) > 0\}}{\sum_{j \in \mathcal{I}} \varpi_{m,j} \mathbf{1}\{q_{d,m,j}(t) > 0\}}, \\ q_{d,s}(t) > 0 &\implies \dot{u}_{d,s,i}(t) = \dot{u}_{d,s}(t) \frac{\varpi_{s,i} \mathbf{1}\{q_{d,s,i}(t) > 0\}}{\sum_{j \in \mathcal{I}} \varpi_{s,j} \mathbf{1}\{q_{d,s,j}(t) > 0\}}. \end{aligned}$$

Proof If $q_{d,\bullet}(t) > 0$ for $\bullet \in \{m, s\}$, then the corresponding pool buffer is nonempty and the policy is work-conserving on that pool: If a decode slot is free, then a job will be routed to the slot immediately. Hence $\dot{u}_{d,\bullet}(t) = \dot{d}_{d,\bullet}(t)$. The instant admission formulas follow directly from the definition of pool selection rule: at the fluid scale, each pool's admissions inherit the fixed weights $\varpi_{\bullet,i}$ restricted to nonempty class buffers. \square

LEMMA EC.8. Fix $\mu \in \mathbb{R}_{++}^I$ and $v \in \mathbb{R}_{++}^I$ with $\sum_{i=1}^I v_i = 1$. Let $D := \text{diag}(\mu_1, \dots, \mu_I)$ and $A := v\mu^\top - D$. Then:

- (i) 0 is a simple eigenvalue of A and all other eigenvalues have strictly negative real parts.
- (ii) For the ODE $\dot{y}(t) = Ay(t)$, $\mathbf{1}^\top y(t)$ is conserved and

$$y(t) \longrightarrow y^\infty := (\mathbf{1}^\top y(0)) \frac{D^{-1}v}{\mathbf{1}^\top D^{-1}v} \quad \text{as } t \rightarrow \infty.$$

Proof (i): By the matrix determinant lemma,

$$\det(\lambda I - A) = \det(\lambda I + D - v\mu^\top) = \det(\lambda I + D) \left(1 - \mu^\top (\lambda I + D)^{-1} v\right).$$

Hence λ is an eigenvalue of A if and only if

$$1 = \mu^\top (\lambda I + D)^{-1} v = \sum_{i=1}^I v_i \frac{\mu_i}{\lambda + \mu_i}.$$

At $\lambda = 0$, the right-hand side equals $\sum_i v_i = 1$, so $\lambda = 0$ is an eigenvalue. Its simplicity follows because the derivative of the right-hand side at $\lambda = 0$ equals $-\sum_i v_i / \mu_i \neq 0$. Next, let λ satisfy $\text{Re}(\lambda) \geq 0$ and $\lambda \neq 0$. Then for each i ,

$$\left| \frac{\mu_i}{\lambda + \mu_i} \right| < \frac{\mu_i}{\text{Re}(\lambda) + \mu_i} \leq 1,$$

with strict inequality because $\lambda \neq 0$. Therefore

$$\left| \sum_{i=1}^I v_i \frac{\mu_i}{\lambda + \mu_i} \right| \leq \sum_{i=1}^I v_i \left| \frac{\mu_i}{\lambda + \mu_i} \right| < \sum_{i=1}^I v_i = 1,$$

so the eigenvalue equation cannot hold. This shows that 0 is the only eigenvalue with nonnegative real part.

(ii): Since $A = v\mu^\top - D$ and $\sum_i v_i = 1$, we have

$$\mathbf{1}^\top A = \mathbf{1}^\top (v\mu^\top) - \mathbf{1}^\top D = (\mathbf{1}^\top v)\mu^\top - \mu^\top = \mathbf{0}^\top.$$

Therefore, along any absolutely continuous solution of $\dot{y}(t) = Ay(t)$,

$$\frac{d}{dt} \mathbf{1}^\top y(t) = \mathbf{1}^\top \dot{y}(t) = \mathbf{1}^\top Ay(t) = 0$$

so the total mass $M := \mathbf{1}^\top y(t)$ is conserved: $M = \mathbf{1}^\top y(0)$ for all t . The equilibrium set is the kernel of A . If $Ay = 0$, then

$$0 = Ay = v(\mu^\top y) - Dy \iff Dy = (\mu^\top y)v.$$

Let $s := \mu^\top y$; since D is invertible, this implies

$$y = s D^{-1}v.$$

Hence $\text{Ker}(A) = \text{span}\{r\}$ where $r := D^{-1}v \in \mathbb{R}_{++}^I$. Imposing the conserved mass $M = \mathbf{1}^\top y$ determines s uniquely:

$$M = \mathbf{1}^\top y = s \mathbf{1}^\top D^{-1}v \implies s = \frac{M}{\mathbf{1}^\top D^{-1}v}.$$

Therefore the hyperplane $\{\mathbf{1}^\top y = M\}$ contains a *unique* equilibrium, namely

$$y^\infty = M \frac{D^{-1}v}{\mathbf{1}^\top D^{-1}v} = (\mathbf{1}^\top y(0)) \frac{D^{-1}v}{\mathbf{1}^\top D^{-1}v}.$$

By part (i), all eigenvalues of A other than 0 have strictly negative real parts, and 0 is a simple eigenvalue. Let $r := D^{-1}v$ be the (right) eigenvector associated with eigenvalue 0, and note that $\mathbf{1}^\top$ is a (left) eigenvector since $\mathbf{1}^\top A = \mathbf{0}^\top$. The corresponding spectral projector is

$$P := \frac{r \mathbf{1}^\top}{\mathbf{1}^\top r} = \frac{D^{-1}v \mathbf{1}^\top}{\mathbf{1}^\top D^{-1}v}.$$

Standard linear-systems theory then implies $e^{At} \rightarrow P$ as $t \rightarrow \infty$. Consequently,

$$y(t) = e^{At} y(0) \longrightarrow P y(0) = (\mathbf{1}^\top y(0)) \frac{D^{-1}v}{\mathbf{1}^\top D^{-1}v} = y^\infty,$$

□

PROPOSITION EC.3. *Assume $\theta_i > 0$ for all $i \in \mathcal{I}$. Under the refined SLI-aware policy above, every fluid solution satisfies*

$$\lim_{t \rightarrow \infty} x_i(t) = x_i^*, \quad \lim_{t \rightarrow \infty} y_{m,i}(t) = y_{m,i}^*, \quad \lim_{t \rightarrow \infty} y_{s,i}(t) = y_{s,i}^*, \quad \lim_{t \rightarrow \infty} q_{d,i}(t) = q_{d,i}^*,$$

for all $i \in \mathcal{I}$.

Proof The prefill part mainly follows from the proof of Theorem 2. Therefore the prefill argument in the proof of Theorem 2 applies verbatim, and there exists $T_p < \infty$ such that for all $t \geq T_p$ and all $i \in \mathcal{I}$,

$$q_{p,i}(t) > 0, \quad x_i(t) = x_i^*. \quad (\text{EC.16})$$

We analyze the decode dynamics on $[T_p, \infty)$ and shift the time origin so that (EC.16) holds for all $t \geq 0$.

Then the resulting constant pool arrival rates are

$$\alpha_{s,i} := p_{s,i} \mu_{p,i} x_i^*, \quad \alpha_{m,i} := (1 - p_{s,i}) \mu_{p,i} x_i^*, \quad i \in \mathcal{I},$$

and the pool capacities

$$y_s^* := \sum_{i \in \mathcal{I}} y_{s,i}^*, \quad y_m^* := \sum_{i \in \mathcal{I}} y_{m,i}^*,$$

which coincide with the binding decode capacity constraints at the LP optimum.

The differential version of the balance equations give:

$$\dot{q}_{d,s,i}(t) = \alpha_{s,i} - \dot{u}_{d,s,i}(t) - \theta_i q_{d,s,i}(t), \quad \dot{q}_{d,m,i}(t) = \alpha_{m,i} - \dot{u}_{d,m,i}(t) - \theta_i q_{d,m,i}(t),$$

together with the corresponding in-service dynamics $\dot{y}_{s,i}(t) = \dot{u}_{d,s,i}(t) - \mu_{s,i} y_{s,i}(t)$ and $\dot{y}_{m,i}(t) = \dot{u}_{d,m,i}(t) - \mu_{m,i} y_{m,i}(t)$ under the static planning.

We first show that there exists $T_0 < \infty$ such that for all $t \geq T_0$,

$$q_{d,s}(t) > 0, \quad q_{d,m}(t) > 0, \quad y_s(t) = y_s^*, \quad y_m(t) = y_m^*, \quad (\text{EC.17})$$

where $q_{d,s}(t) := \sum_i q_{d,s,i}(t)$ and $q_{d,m}(t) := \sum_i q_{d,m,i}(t)$, and $y_s(t) := \sum_i y_{s,i}(t)$ and $y_m(t) := \sum_i y_{m,i}(t)$.

Fix the solo pool (the mixed pool is identical). Using the LP balances and the definitions of $q_{d,s,i}^* := p_{s,i} q_{d,i}^*$, we have for each i ,

$$\alpha_{s,i} = \mu_{s,i} y_{s,i}^* + \theta_i q_{d,s,i}^* \implies \frac{\alpha_{s,i}}{\mu_{s,i}} \geq y_{s,i}^*,$$

with strict inequality for any i with $p_{s,i} > 0$ (equivalently $y_{s,i}^* > 0$). Summing gives

$$\sum_{i \in \mathcal{I}} \frac{\alpha_{s,i}}{\mu_{s,i}} > \sum_{i \in \mathcal{I}} y_{s,i}^* = y_s^*. \quad (\text{EC.18})$$

Suppose, toward a contradiction, that $q_{d,s}(t) = 0$ for $t \geq T'$ for some $T' < \infty$. Then $q_{d,s,i}(t) = 0$ for $t \geq T'$ and all i . The solo-buffer balance implies, for $t \geq T'$,

$$0 = \dot{q}_{d,s,i}(t) = \alpha_{s,i} - \dot{u}_{d,s,i}(t) - \theta_i q_{d,s,i}(t) = \alpha_{s,i} - \dot{u}_{d,s,i}(t),$$

so $\dot{u}_{d,s,i}(t) = \alpha_{s,i}$ for $t \geq T'$. Therefore for $t \geq T'$,

$$\dot{y}_{s,i}(t) = \dot{u}_{d,s,i}(t) - \mu_{s,i} y_{s,i}(t) = \alpha_{s,i} - \mu_{s,i} y_{s,i}(t).$$

Solving this linear ODE yields $y_{s,i}(t) \rightarrow \alpha_{s,i}/\mu_{s,i}$ as $t \rightarrow \infty$. Taking sums and using (EC.18) implies $y_s(t) = \sum_i y_{s,i}(t) \rightarrow \sum_i \alpha_{s,i}/\mu_{s,i} > y_s^*$, contradicting the capacity constraint $y_s(t) \leq y_s^*$ for all t . Therefore the solo-pool buffer cannot be eventually empty. Moreover, by (EC.18), there exists $T_s < \infty$ such that

$$q_{d,s}(t) > 0, \quad \forall t \geq T_s.$$

Applying the same reasoning to the mixed pool yields $T_m < \infty$ such that $q_{d,m}(t) > 0$ for all $t \geq T_m$. Let $T_0 := \max\{T_s, T_m\}$. Then for all $t \geq T_0$, both pool buffers are positive. By the work-conserving property of the policy within each pool, $y_s(t) = y_s^*$ and $y_m(t) = y_m^*$ for all $t \geq T_0$, proving (EC.17).

Second, we focus on the convergence of the decode occupancies. We consider $t \geq T_0$ for which (EC.17) holds. Since $q_{d,m}(t) > 0$, Lemma EC.7 gives $\dot{u}_{d,m}(t) = \dot{d}_{d,m}(t)$. Moreover, since $q_{d,m}(t) > 0$ for all $t \geq T_0$ by (EC.17), the selection rule is active for all such t . For any class i with $\alpha_{m,i} > 0$, if $q_{d,m,i}(t) = 0$ when $t \geq T_0$, then no class- i job can be pulled from the mixed buffer at time t , so $\dot{u}_{d,m,i}(t) = 0$ and the mixed-queue balance gives

$$\dot{q}_{d,m,i}(t) = \alpha_{m,i} - \dot{u}_{d,m,i}(t) - \theta_i q_{d,m,i}(t) = \alpha_{m,i} > 0.$$

Therefore $q_{d,m,i}(t) > 0$ for $t \geq T_0$ whenever $\alpha_{m,i} > 0$. Hence the class-level admission rate simplifies to the unconditional proportion $\varpi_{m,i}$ for almost every $t \geq T_0$.

$$\dot{u}_{d,m,i}(t) = \varpi_{m,i} \dot{u}_{d,m}(t) = \varpi_{m,i} \sum_{j \in \mathcal{I}} \mu_{m,j} y_{m,j}(t).$$

Under the static planning, the mixed in-service masses satisfy

$$\dot{y}_{m,i}(t) = \dot{u}_{d,m,i}(t) - \mu_{m,i} y_{m,i}(t) = \varpi_{m,i} \sum_{j \in \mathcal{I}} \mu_{m,j} y_{m,j}(t) - \mu_{m,i} y_{m,i}(t).$$

This is a linear system $\dot{y}_m(t) = A_m y_m(t)$, where

$$A_m := \varpi_m \mu_m^\top - \text{diag}(\mu_{m,1}, \dots, \mu_{m,I}), \quad \varpi_m := (\varpi_{m,i})_{i \in \mathcal{I}}, \quad \mu_m := (\mu_{m,i})_{i \in \mathcal{I}}.$$

Applying Lemma EC.8 to A_m yields $y_{m,i}(t) \rightarrow y_{m,i}^*$ as $t \rightarrow \infty$. The same argument applies to the solo pool which yields $y_{s,i}(t) \rightarrow y_{s,i}^*$.

Finally, we prove the convergence of the decode queues. For each class i , the pool-specific decode queues satisfy, for $t \geq T_0$,

$$\dot{q}_{d,s,i}(t) = \alpha_{s,i} - \dot{u}_{d,s,i}(t) - \theta_i q_{d,s,i}(t), \quad \dot{q}_{d,m,i}(t) = \alpha_{m,i} - \dot{u}_{d,m,i}(t) - \theta_i q_{d,m,i}(t).$$

Since both pools are work-conserving for all $t \geq T_0$, the admission rates satisfy

$$\dot{u}_{d,s,i}(t) = \varpi_{s,i} \sum_{j \in \mathcal{I}} \mu_{s,j} y_{s,j}(t), \quad \dot{u}_{d,m,i}(t) = \varpi_{m,i} \sum_{j \in \mathcal{I}} \mu_{m,j} y_{m,j}(t),$$

for $t \geq T_0$. Substituting gives, for $t \geq T_0$,

$$\dot{q}_{d,s,i}(t) = \alpha_{s,i} - \varpi_{s,i} \sum_{j \in \mathcal{I}} \mu_{s,j} y_{s,j}(t) - \theta_i q_{d,s,i}(t), \quad \dot{q}_{d,m,i}(t) = \alpha_{m,i} - \varpi_{m,i} \sum_{j \in \mathcal{I}} \mu_{m,j} y_{m,j}(t) - \theta_i q_{d,m,i}(t).$$

Since $y_{s,i}(t) \rightarrow y_{s,i}^*$ and $y_{m,i}(t) \rightarrow y_{m,i}^*$, the pool-level LP balances satisfy

$$p_{s,i} \mu_{p,i} x_i^* - \mu_{s,i} y_{s,i}^* = \theta_i q_{d,s,i}^*, \quad (1 - p_{s,i}) \mu_{p,i} x_i^* - \mu_{m,i} y_{m,i}^* = \theta_i q_{d,m,i}^*.$$

Using the definition of $\varpi_{\bullet,i}$, we also have $\varpi_{s,i} \sum_{j \in \mathcal{I}} \mu_{s,j} y_{s,j}^* = \mu_{s,i} y_{s,i}^*$ and $\varpi_{m,i} \sum_{j \in \mathcal{I}} \mu_{m,j} y_{m,j}^* = \mu_{m,i} y_{m,i}^*$.

Solving the standard linear ODE, we have for any $t \geq T_0$,

$$q_{d,s,i}(t) = e^{-\theta_i(t-T_0)} q_{d,s,i}(T_0) + \int_{T_0}^t e^{-\theta_i(t-u)} \left(\alpha_{s,i} - \varpi_{s,i} \sum_{j \in \mathcal{I}} \mu_{s,j} y_{s,j}(u) \right) du.$$

Since $y_{s,i}(u) \rightarrow y_{s,i}^*$ for each i , the integrand converges to $\alpha_{s,i} - \varpi_{s,i} \sum_{j \in \mathcal{I}} \mu_{s,j} y_{s,j}^* = \alpha_{s,i} - \mu_{s,i} y_{s,i}^* = \theta_i q_{d,s,i}^*$. Standard arguments for linear convolution with an exponentially decaying kernel (or directly applying dominated convergence) then yield $q_{d,s,i}(t) \rightarrow q_{d,s,i}^*$ as $t \rightarrow \infty$. The same argument applied to the mixed pool queue gives $q_{d,m,i}(t) \rightarrow q_{d,m,i}^*$. Summing gives $q_{d,i}(t) = q_{d,m,i}(t) + q_{d,s,i}(t) \rightarrow q_{d,i}^*$. \square

EC.7. Additional Experiments and Empirical Validation

This section provides additional empirical validations that complement the main text. We first provide a comprehensive analysis of workload heterogeneity through task-level statistics and visualization in the (P, D) space. We then validate the exponential distribution assumption for output lengths by fitting empirical data and calculating divergence metrics. Finally, we detail the computing infrastructure used for our measurements and simulations.

Workload heterogeneity and task characteristics. Table EC.1 reports the average prompt (input) and output lengths for eight task categories from the Databricks Dolly-15k instruction-tuning dataset (Conover et al. 2023). The data reveals substantial variation across categories. For example, summarization and information extraction tasks average over 1,000 input tokens with moderate output. In contrast, creative writing and open QA require fewer than 100 input tokens yet generate longer outputs on average.

To further illustrate this heterogeneity, Figure EC.1 plots the input and output length pairs for two representative categories: Information Extraction and Creative Writing. These two classes exhibit nearly opposite characteristics. Information Extraction tasks cluster in the high-input and low-output region. Creative Writing tasks concentrate in the low-input and high-output zone. This contrast confirms that a uniform scheduling policy cannot efficiently balance resources across diverse workloads, which motivates the multiclass framework developed in this paper.

Task Category	Avg. P (tokens)	Avg. D (tokens)	Samples
Brainstorming	61	331	1,764
Classification	123	142	2,136
Closed QA	992	182	1,738
Creative writing	89	915	699
General QA	69	572	2,187
Information extraction	1,139	273	1,462
Open QA	45	293	3,739
Summarization	1,177	436	1,150

Table EC.1 Workload heterogeneity: average prompt (P) and output (D) lengths across task categories from the Databricks Dolly-15k instruction-tuning dataset (Conover et al. 2023).

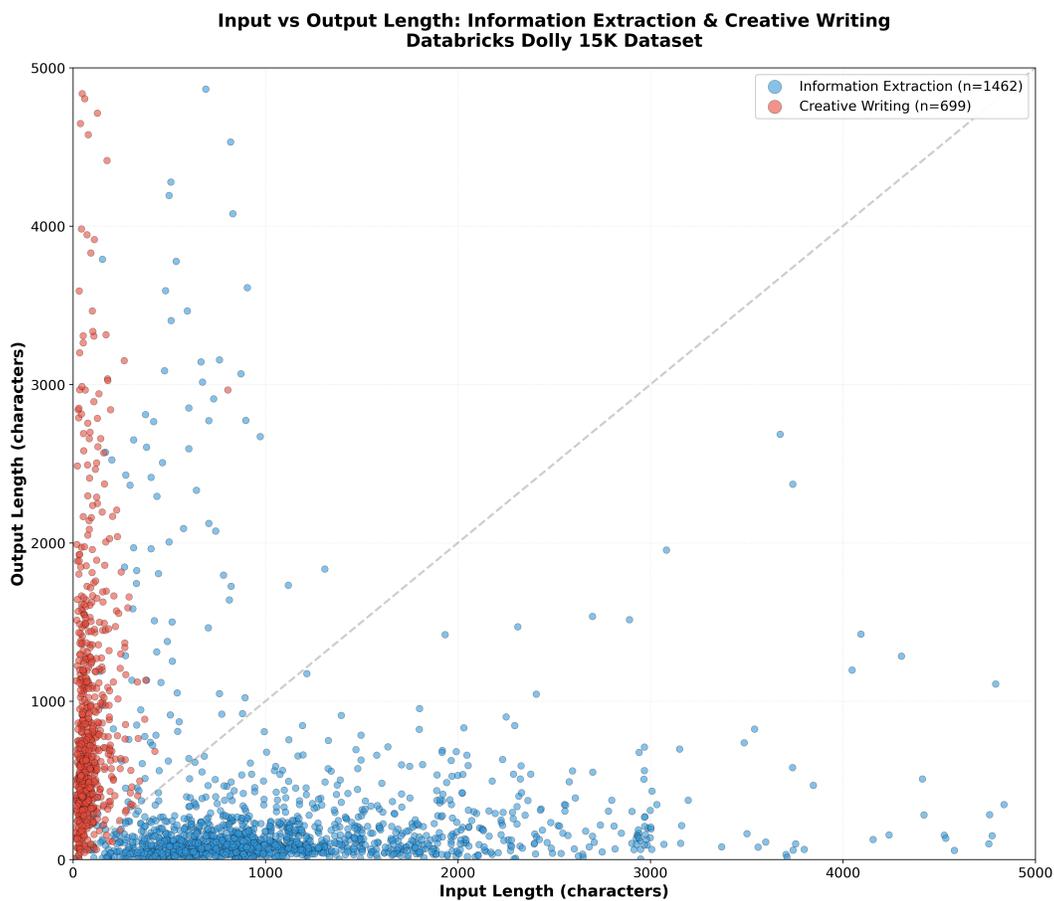


Figure EC.1 Scatter plot of input versus output lengths for Information Extraction (blue) and Creative Writing (red) in the Databricks Dolly-15k dataset. The two categories occupy nearly opposite regions of the (P, D) space.

Validation of the exponential distribution assumption. Our model assumes exponentially distributed output lengths for each task class. This assumption directly determines the decode service rate in our stochastic network. To check whether this approximation is reasonable, we fit exponential distributions to the actual output length data from the Databricks Dolly-15k dataset using Maximum Likelihood Estimation.

Figure EC.2 compares the empirical CDFs with the fitted exponential CDFs for all eight task categories. The exponential fit is good for most categories. Brainstorming, Classification, Closed QA, Information Extraction, Open QA, and Summarization all track closely. General QA shows reasonable agreement despite some mid-range deviation. Creative Writing deviates more noticeably in the upper tail. This is likely because its output distribution is heavier-tailed than a pure exponential, which is consistent with its high mean and variance.

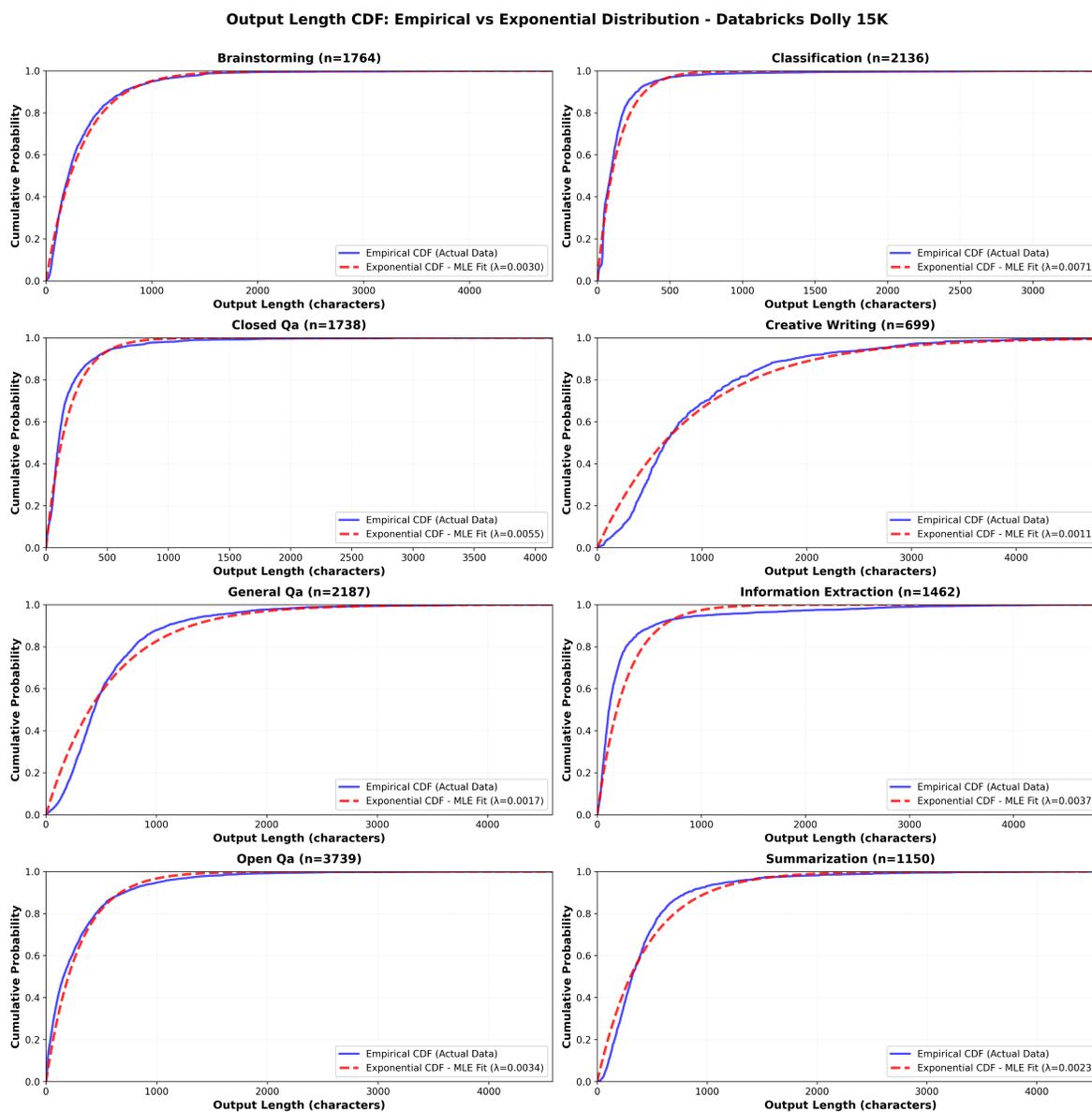


Figure EC.2 Empirical output length CDFs (solid blue) versus fitted exponential CDFs (dashed red) for the eight task categories in the Databricks Dolly-15k dataset.

To complement the visual comparison, Table EC.2 reports the Kullback-Leibler (KL) divergence between the empirical output-length distribution and its fitted exponential approximation for each category. Smaller values indicate closer agreement. Overall, the exponential assumption captures the main characteristics of the distributions well enough to justify its use for analytical tractability.

Table EC.2 KL divergence between empirical output lengths and fitted exponential distributions across task categories (smaller is better).

Category	KL divergence
Open QA	0.0584
Brainstorming	0.0652
Closed QA	0.1245
Summarization	0.1255
General QA	0.1309
Classification	0.1491
Creative Writing	0.1595
Information Extraction	0.2631

Computing infrastructure. Table EC.3 summarizes the hardware and software environment used for the empirical measurements and numerical experiments in the paper.

Table EC.3 Computing infrastructure specifications.

Component	Specification
Hardware	
CPU	AMD EPYC 7H12 64-core processor
Cores / threads	2 sockets \times 64 cores \times 2 threads (256 hardware threads)
Clock frequency	1.5–2.6 GHz
Cache	L1d/L1i: 4 MiB (128 \times), L2: 64 MiB (128 \times), L3: 512 MiB (32 \times)
GPU	4 \times NVIDIA A100-SXM4-40GB
GPU memory	40 GB per GPU (160 GB total)
System memory	1.0 TiB RAM
Swap	8.0 GiB
Software	
Operating system	Ubuntu 22.04.5 LTS (Jammy Jellyfish)
Python	Version 3.9.23
CUDA	Version 12.6
NVIDIA driver	Version 560.28.03