Optimizing LLM Inference: Fluid-Based Online Scheduling under Memory Constraints

Ruicheng Ao
Institute for Data, Systems, and Society
Massachusetts Institute of Technology
ao@mit.edu

Gan Luo
School of Mathematical Sciences
Peking University
luogan@stu.pku.edu.cn

David Simchi-Levi
Institute for Data, Systems, and Society
Massachusetts Institute of Technology
dslevi@mit.edu

Xinshang Wang
Alibaba Group
xinshang.w@alibaba-inc.com

Abstract

Large Language Model (LLM) inference faces unique scheduling challenges due to the dynamically growing Key-Value (KV) cache during token generation, making traditional scheduling algorithms ineffective. We develop a fluid dynamics approximation to establish an optimal throughput benchmark and propose the WAIT (Waiting for Accumulated Inference Threshold) algorithm that achieves near-optimal performance with near-optimal throughput gap. For practical scenarios with unknown output lengths, we introduce Nested WAIT that maintains asymptotic optimality through hierarchical segmentation. Experiments on Llama-7B demonstrate 20-30% throughput improvements over state-of-the-art systems like vLLM.

1 Introduction

LLM inference presents a fundamental scheduling challenge: the Key-Value (KV) cache grows dynamically during token generation, with memory requirements increasing unpredictably from initial prompt processing (prefill) through autoregressive decoding. This dynamic memory growth invalidates classical scheduling approaches like Shortest Job First, as jobs with initially small memory footprints can expand to dominate GPU resources during execution.

Current system-level solutions Kwon et al. [2023], Agrawal et al. [2023] employ engineering optimizations but lack theoretical foundations. Our work bridges operations research and machine learning by developing a queueing-theoretic framework for LLM inference scheduling. Unlike prior theoretical work that assumes known output lengths, we address the realistic setting where output lengths are unknown at arrival.

Contributions: We make three key contributions:

- **Fluid benchmark:** We establish a fluid dynamics model that characterizes optimal throughput **Thoughput**$^* = \sum_j \lambda_j(l'_j + 1)$ under memory constraints, providing a theoretical performance bound.
- **WAIT algorithm:** For known output lengths, our threshold-based algorithm achieves near-optimal throughput with gap $O(1/T)$ under heavy traffic.
- **Nested WAIT:** For unknown output lengths, we develop a hierarchical segmentation approach that maintains asymptotic optimality while "learning" prompt types during execution.

Preprint. Under review.

Our algorithms achieve 20-30% throughput improvements over production systems on real workloads, demonstrating that principled scheduling can substantially improve LLM serving efficiency. More detailed literature reviews and problem background can be found in Appendix A

2 Model

We consider an LLM inference system processing prompts on a single GPU with memory capacity C. Prompts arrive stochastically as m types, where type $j \in [m]$ has:

- Arrival rate λ_j (Poisson process)
- Input length l_j tokens (prefill phase)
- Output length l'_j tokens (decode phase)

During inference, each prompt undergoes one prefill iteration followed by l'_j decode iterations. A prompt at **stage** k ($k \in \{0, 1, \ldots, l'_j\}$) has processed its input and generated k output tokens, consuming memory $l_j + k$ for its KV cache.

Batching dynamics: The GPU processes batches containing prompts at various stages. For a batch with n_1 prompts in prefill (lengths $k_1, \ldots, k_{n_1}$) and n_2 prompts in decode (current sizes $s_1, \ldots, s_{n_2}$), the iteration time is:

$$\tau = d_0 + d_1 \cdot \left(\sum_{i=1}^{n_1} k_i + \sum_{i=1}^{n_2} s_i \right) \tag{1}$$

where d_0 is fixed overhead and d_1 is per-token memory cost. This linear model, validated experimentally, reflects memory-bound computation in attention mechanisms.

Optimization objective: Maximize throughput subject to memory and latency constraints:

$$\begin{aligned} \max_{\pi \in \Pi} \quad & \mathbb{E}[\textbf{Thoughput}^{(T,\pi)}] \\ \text{s.t.} \quad & \sum_{i \in B^t} (l_{j(i)} + k_i^t) \leq C \quad \forall t \end{aligned} \tag{2}$$

where Π denotes non-preemptive scheduling policies and B^t is the batch at time t.

3 Fluid Dynamics and Benchmark

To establish a performance benchmark, we develop a fluid model where discrete stochastic arrivals are approximated by continuous deterministic flows. In equilibrium, the system maintains n_j^* active prompts of type j, with exactly $n_j^*/(l'_j + 1)$ prompts at each stage $k \in \{0, 1, \ldots, l'_j\}$.

Equilibrium analysis: The total memory usage for type j is:

$$M_j^* = n_j^* \left(l_j + \frac{l'_j}{2} \right) \tag{3}$$

In steady state, arrivals equal completions during each iteration:

$$(d_0 + d_1 M^*) \lambda_j = \frac{n_j^*}{l'_j + 1} \tag{4}$$

Solving this system yields the equilibrium memory:

$$M^* = \frac{d_0 \sum_{j=1}^m \lambda_j (l'_j + 1)(l_j + \frac{l'_j}{2})}{1 - d_1 \sum_{j=1}^m \lambda_j (l'_j + 1)(l_j + \frac{l'_j}{2})} \tag{5}$$

The optimal throughput benchmark is:

$$\textbf{Thoughput}^* = \sum_{j=1}^m \lambda_j (l'_j + 1) \tag{6}$$

Key insight: This fluid benchmark represents the maximum achievable throughput for any non-preemptive policy. We prove that $\mathbb{E}[\textbf{Thoughput}^{(T,\pi)}] \leq \textbf{Thoughput}^*$ for all policies $\pi \in \Pi$, establishing this as the fundamental performance limit for LLM inference scheduling.

4 WAIT Algorithm for Known Output Lengths

The WAIT (Waiting for Accumulated Inference Threshold) algorithm uses threshold-based batching to approach fluid equilibrium. For each prompt type j, we set threshold n_j and process type j only when at least n_j prompts are waiting at stage 0 (prefill).

Algorithm 1 WAIT Algorithm

```
Initialize inventories n_jk = 0 for all types j, stages k
while true do
    Wait for arrivals or batch completion
    for each type j with n_j0 ≥ n_j do
        Add min{n_j, n_jk} prompts at each stage k to batch
    end for
    if batch is non-empty then
        Process batch and update inventories
    end if
end while
```

Threshold selection: Choose n_j satisfying:

$$\Delta T(n_1, \ldots, n_m) := d_0 + d_1 \sum_j n_j(l'_j + 1)(l_j + \frac{l'_j}{2}) < \frac{n_j}{\lambda_j} \tag{7}$$

This ensures arrivals during processing don't exceed completions, maintaining stability.

Theorem 4.1 (WAIT Optimality)**.** *Under heavy traffic scaling with arrival rates $\lambda_j^{(\zeta)} = \zeta\lambda_j$, suppose the total memory $M > M^*$, where M^* is given in (5), WAIT achieves:*

$$\textbf{\textit{Thoughput}}^* - \mathbb{E}[\textbf{\textit{Thoughput}}^{(\zeta,\pi)}] = O((\zeta T)^{-1}) \tag{8}$$

$$\mathbb{E}[\textit{Latency}^{(\zeta,\pi)}] = O(1) \tag{9}$$

The proof uses coupling arguments from queueing theory, constructing a dominant process to bound performance gaps. WAIT achieves the optimal $O(1/T)$ convergence rate, matching theoretical lower bounds.

5 Nested WAIT for Unknown Output Lengths

Real-world LLM inference faces unknown output lengths at arrival. Nested WAIT addresses this by organizing processing into m segments corresponding to potential output lengths $l'_1 < l'_2 < \ldots < l'_m$.

Key idea: Prompts become distinguishable as they progress through decoding. Initially, all prompts enter segment 1 (stages 0 to l'_1). After l'_1 iterations, type-1 prompts complete while others advance to segment 2, revealing their true types progressively.

Segment dynamics:

- Segment i contains prompts at stages $\{l'_{i-1} + 1, \ldots, l'_i\}$
- Arrival rate for segment i: $\sum_{j=i}^m \lambda_j$
- Threshold n_i triggers processing when segment i has sufficient prompts

Theorem 5.1 (Nested WAIT Optimality)**.** *With thresholds satisfying $n_{i+1}/n_i > (\sum_{j=i+1}^m \lambda_j)/(\sum_{j=i}^m \lambda_j)$ and the total memory satisfying equation (9) in Ao et al. [2025], Nested WAIT achieves:*

$$\textbf{\textit{Thoughput}}^* - \mathbb{E}[\textbf{\textit{Thoughput}}^{(\zeta,\pi)}] = O((\zeta T)^{-1})$$

Despite unknown types, Nested WAIT maintains the same asymptotic optimality as WAIT, demonstrating robustness to uncertainty.

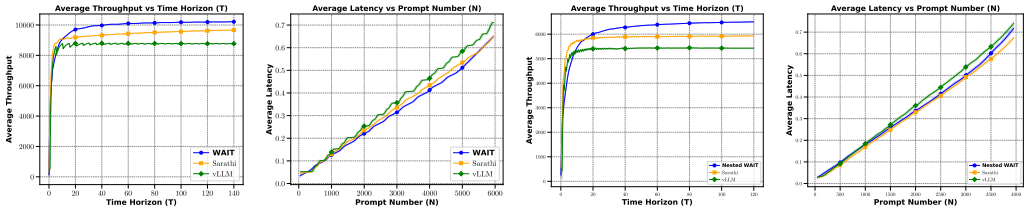
Algorithm 2 Nested WAIT Algorithm

```
Initialize segment inventories, thresholds [n_1, ..., n_m]
while true do
    Find largest i where all segments 1 to i meet thresholds
    if such i exists then
        Form batch from segments 1 to i
        Process batch, advance prompts to next segments
    else
        Wait for more arrivals
    end if
end while
```

6 Experiments

We evaluate (Nested) WAIT against vLLM Kwon et al. [2023] and Sarathi Agrawal et al. [2023] using Llama-7B on A100 GPU (Microsoft Vidur simulator Agrawal et al. [2024]).

Workloads: (1) *Synthetic:* For WAIT with known lengths, we test low demand (2 types, $l' = (10, 20)$, $\lambda = (1000, 1000)$) and high demand (3 types, $l' = (100, 200, 300)$, $\lambda = (6000, 4000, 2000)$). For Nested WAIT, we use 4 types with $l' = (20, 40, 80, 160)$. (2) *Real:* LMSYS-Chat-1M Zheng et al. [2023] with 50K prompts, grouped into 10 bins (50 tokens each) for Nested WAIT.



(a) WAIT: Throughput (b) WAIT: Latency (c) Nested: Throughput (d) Nested: Latency

Figure 1: Throughput and latency comparison under high load: (a-b) WAIT on synthetic data, (c-d) Nested WAIT on real data.

Results: Figures (a) and (c) demonstrate consistent 20-30% throughput improvements across both algorithms. WAIT achieves peak gains under high synthetic load, while Nested WAIT maintains similar performance on real workloads despite unknown output lengths. Latency plots (b,d) show the tradeoff: while our algorithms incur slightly higher latency at low loads due to batching, they achieve superior throughput-latency balance under heavy traffic—the target operating regime for production systems. The threshold-based approach effectively approaches theoretical fluid limits in practice.

7 Conclusion

We developed a theoretically grounded framework for LLM inference scheduling that bridges operations research and machine learning. Our fluid dynamics model establishes fundamental performance limits, while the (Nested) WAIT algorithms achieve near-optimal throughput with provable guarantees. The hierarchical segmentation in Nested WAIT elegantly handles unknown output lengths—a critical challenge in practical deployments—while maintaining asymptotic optimality.

Future directions: Extensions to multi-GPU systems with pipeline parallelism, handling time-varying arrival patterns, and incorporating recent system optimizations like multi-head latent attention present promising research opportunities. Our framework provides a foundation for principled scheduling in increasingly complex LLM serving scenarios.

References

Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S Gulavani, and Ramachandran Ramjee. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills. *arXiv preprint arXiv:2308.16369*, 2023.

Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav Gulavani, Ramachandran Ramjee, and Alexey Tumanov. Vidur: A large-scale simulation framework for llm inference. *Proceedings of Machine Learning and Systems*, 6:351–366, 2024.

Ruicheng Ao, Gan Luo, David Simchi-Levi, and Xinshang Wang. Optimizing llm inference: Fluid-guided online scheduling with memory constraints, 2025. URL https://arxiv.org/abs/2504.11320.

Søren Asmussen, Soren Asmussen, and Sren Asmussen. *Applied probability and queues*, volume 2. Springer, 2003.

Zixi Chen, Yinyu Ye, and Zijie Zhou. Adaptively robust llm inference optimization under prediction uncertainty. *arXiv preprint arXiv:2508.14544*, 2025.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Yichao Fu, Siqi Zhu, Runlong Su, Aurick Qiao, Ion Stoica, and Hao Zhang. Efficient llm scheduling by learning to rank. *Advances in Neural Information Processing Systems*, 37:59006–59029, 2025.

Patrick Jaillet, Jiashuo Jiang, Chara Podimata, and Zijie Zhou. Online scheduling for llm inference with kv cache constraints, 2025. URL https://arxiv.org/abs/2502.07115.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

Yueying Li, Jim Dai, and Tianyi Peng. Throughput-optimal scheduling algorithms for llm inference and ai agents, 2025. URL https://arxiv.org/abs/2504.07347. Accessed: 2025-04-11.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. in 2024 acm/ieee 51st annual international symposium on computer architecture (isca). *IEEE Computer Society, Los Alamitos, CA, USA*, pages 118–132, 2024.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624, 2023.

Peggy Strait. On the maximum and minimum of partial sums of random variables. *Pacific Journal of Mathematics*, 52(2):585–593, 1974.

Meixuan Wang, Yinyu Ye, and Zijie Zhou. Llm serving optimization with variable prefill and decode lengths. *arXiv preprint arXiv:2508.06133*, 2025.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538, 2022.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P Xing, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*, 2023.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. {DistServe}: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 193–210, 2024.

A Extended Introduction and Related Work

A.1 LLM Inference Process: Detailed Description

Understanding the LLM inference process is essential for tackling its associated scheduling challenges. Figure 2 provides a step-by-step illustration of how a query is processed during inference.

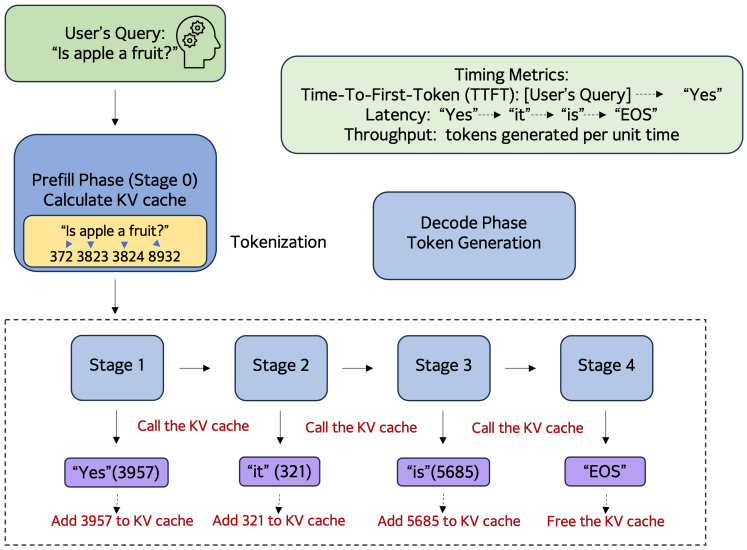


Figure 2: An example of LLM inference. The process begins with a user prompt ("Is apple a fruit?") that undergoes prefill (Stage 0) to initialize the KV cache, followed by sequential decode stages (1-4) where output tokens are generated one at a time. The figure shows how memory usage grows during decoding and illustrates key metrics: TTFT (Time to First Token), Latency, and Throughput.

The process begins when a user submits a prompt—e.g., "Is apple a fruit?"—which is then passed through several computational stages. The inference pipeline consists of two main phases:

Prefill Phase (stage 0): Upon receiving the prompt, the model first tokenizes the input into a sequence of discrete units (e.g., "Is", "apple", "a", "fruit", "?"). These tokens are then embedded and simultaneously processed in a single forward pass to compute the Key-Value (KV) cache, which stores intermediate representations (i.e., attention keys and values) for each token. These precomputed values are critical for enabling efficient reuse during subsequent decoding steps. This phase corresponds to Stage 0 in Figure 2, where all prompt tokens are embedded and their KV representations are added to the cache.

Decode Phase (stages 1 to l'): After the prefill phase, the model enters the decode phase, where it generates the output one token at a time. At each stage, the model queries the existing KV cache to compute the next token, appends the new token to the output sequence, and updates the KV cache with its key-value pair. For instance, the model might first generate "Yes" (Stage 1), then "it" (Stage 2), followed by "is" (Stage 3), and finally the End of Sequence (EOS) token (Stage 4). This progression is depicted in the lower part of Figure 2. Each token generation step involves both reading from and writing to the KV cache, resulting in a memory footprint that grows linearly with the length of the generated sequence.

Figure 2 also highlights key performance metrics in LLM inference:

- **Time-To-First-Token (TTFT)** measures the latency from user input to the first generated token.
- **Latency** refers to the total time required to complete the generation of all output tokens.
- **Throughput** captures the average number of tokens generated per unit time.

This inference structure, particularly the growing memory requirements and sequential decoding pattern, introduces fundamental constraints on scheduling and batching. Efficient scheduling must account for both prompt heterogeneity and KV cache dynamics in order to optimize latency and resource utilization.

A.2 Why Traditional Scheduling Fails for LLM Inference

The KV cache is essential for efficiency, as it prevents the model from recalculating the attention history for each new token. Without it, the computational cost would scale quadratically with the sequence length, making real-time inference infeasible. However, the expanding KV cache poses a significant scheduling challenge, as the memory footprint grows unpredictably during the decode phase. This variability, combined with stochastic prompt arrivals and differing response lengths, complicates resource allocation on hardware with finite capacity, such as GPUs. Exceeding memory limits can force the system to offload data to slower storage mediums, degrading performance and increasing latency.

Scheduling LLM inference tasks involves grouping prompts into batches processed concurrently on a GPU, combining prefill and decode phases to optimize resource use. This process is constrained by GPU memory limits, as the KV cache grows with each generated token, restricting the number of active prompts. Traditional scheduling methods, such as Shortest Job First (SJF), assume fixed job sizes and known processing times, making them unsuitable for LLM inference where memory demands increase dynamically and output lengths are often unknown.

For example, consider two prompts: one with a short input but a long, unpredictable output (e.g., "Summarize the history of artificial intelligence"), and another with a longer input but a short output (e.g., "Is 5 a prime number?"). In a traditional setting, SJF would prioritize the second prompt, expecting its shorter processing time to minimize average wait times. However, in LLM inference, the first prompt's decode phase could generate hundreds of tokens, causing its KV cache to balloon over time and occupy significant memory, while the second prompt's quick decode phase releases resources almost immediately after a longer prefill. Prioritizing the second prompt might reduce initial latency, but the first prompt's prolonged memory usage could block other tasks, leading to GPU under-utilization and reduced throughput.

Beyond SJF, other traditional methods like priority scheduling falter as they prioritize prompts without accounting for the KV cache's unpredictable growth, potentially exhausting memory. While batching remains effective, it requires dynamic adaptation—e.g., adjusting batch sizes based on current memory usage—to accommodate the KV cache's expansion, unlike static batching in traditional settings. These characteristics—stochastic prompt arrivals, multi-phase processing, and unpredictable resource needs—render classical operations research approaches inadequate, necessitating tailored scheduling strategies that account for dynamic memory growth and phase-specific demands.

A.3 Recent Work on LLM Inference Scheduling

Recent system-level optimizations for LLM inference Yu et al. [2022], Kwon et al. [2023], Agrawal et al. [2023], Pope et al. [2023], DeepSeek-AI [2024], Patel et al. [2024], Zhong et al. [2024] focus on engineering solutions, such as batching and memory compression, but lack a rigorous mathematical foundation.

Notable theoretical contributions include:

- Jaillet et al. [2025] models LLM inference as an online scheduling problem with KV cache constraints, developing an adapted shortest-job-first algorithm with near-optimal regret bounds, though it assumes known output lengths at arrival—a limitation for real-world scenarios.
- Wang et al. [2025] optimizes serving with variable prefill and decode lengths through a stochastic optimization framework.
- Chen et al. [2025] proposes robust optimization techniques to handle prediction uncertainty in LLM inference scheduling.
- Li et al. [2025] proposes a stochastic processing model showing that work-conserving scheduling algorithms achieve optimal throughput for both individual requests and multi-agent AI workloads.

In practice, output length predictions are imprecise or costly. For example, as shown in Fu et al. [2025], high-accuracy prediction is sensitive to the bucket size of output lengths for classification. When the output length is unknown, the performance of algorithms may degenerate significantly and performance guarantees obtained in full knowledge scenarios no longer hold. To address this problem, we develop a fluid dynamics approximation to establish a tractable benchmark, providing

insights for devising effective scheduling algorithms that can handle unknown output lengths while maintaining theoretical guarantees.

A.4 Detailed Contributions

This work contributes to LLM inference scheduling through the following:

Mathematical Model: We develop a multi-stage online scheduling model for LLM inference, accounting for queueable prompts under memory constraints. To analyze performance and inform practical algorithms, we employ a fluid dynamics approximation based on queueing theory, which serves as both an analytical tool and a performance benchmark.

Asymptotically Optimal Algorithms: We develop the WAIT algorithm for known output lengths as a warm-up and extend it to the Nested WAIT algorithm for unknown output lengths, both achieving asymptotic optimality under heavy traffic via novel waiting time and coupling analyses.

Experimental Validation: We evaluate our algorithms on synthetic and real-world datasets, outperforming benchmarks like vLLM and Sarathi, which are widely recognized high-performance inference engines, in average throughput using Llama2-7B on a single A100 GPU.

These contributions bridge operations research and machine learning, addressing analytical challenges in stochastic, memory-constrained online scheduling problems.

B Extensions

In this section, we present two extensions to our generalize our algorithms and theoretic analysis. First, we investigate the adaptation of Algorithm 2 to accommodate time-varying arrival rates. Second, we propose a generalized segment design for Algorithm 2, extending beyond the existing m-segment parameterization. We establish that these extensions preserve comparable theoretical guarantees as in Theorem 5.1.

B.1 Time-Varying Arrival Rates

In this part, we analyze the asymptotic optimality of the Nested WAIT algorithm under time-varying arrival rates, denoted λ_j^t for prompt type $j \in [m]$ at time $t \in [0, T]$. We assume that λ_j^t are continuous functions of time, uniformly bounded by $\lambda_{\max} < \infty$. Define the accumulated arrivals of type j over the interval $[t_1, t_2]$ as $\lambda_j[t_1, t_2] := \int_{t_1}^{t_2} \lambda_j^t \, dt$. Notice that $\Delta T_{[1,2,\ldots,m]}(n_1, \cdots, n_m)$ represents the per-iteration time cost when processing exactly n_i prompts at each stage in segment i, for all $i \in [m]$ (n_i is the threshold defined in Algorithm 2).

For the Nested WAIT policy defined by $\pi = [n_1, \ldots, n_m]$, we consider the following conditions for all $t \in [0, T]$, $0 < \Delta t \le \Delta T_{[1,2,\cdots,m]}(n_1, \cdots, n_m)$ and $i = 1, 2, \ldots, m-1$.

$$\begin{aligned} &\sum_{j=1}^{m} \lambda_j \big[t, t + \Delta t\big] \le \sum_{j=1}^{m} \lambda_j \big[t, t + \Delta T_{[1,2,\cdots,m]}(n_1, \cdots, n_m)\big] < n_1, \\ &n_{i+1}/n_i > p_i[t, t + \Delta t], \end{aligned} \tag{10}$$

where $p_i[t, t + \Delta t] = (\sum_{j=i+1}^{m} \lambda_j[t, t + \Delta t])/(\sum_{j=i}^{m} \lambda_j[t, t + \Delta t])$.

We define that the total number of iterations over time horizon $[0, T]$ as S. Therefore, we have $T \ge \sum_{s=1}^{S} \Delta T_{s\text{-th Batch}} \ge S \cdot \min \Delta T_{\text{Batch}}$, where $\Delta T_{s\text{-th Batch}}$ denotes the time cost of the s-th iteration and $\Delta T_{\min} = \min_s\{\Delta T_{\text{Batch}}\}$. We now present the following result that extends Theorem 5.1.

Theorem B.1. *Suppose the thresholds $\pi = [n_1, \ldots, n_m]$ satisfy (10), and the memory $M^{(\zeta,\pi)}$ fulfills:*

$$\begin{aligned} M^{(\zeta,\pi)} &\ge M^{\pi} + \sum_{j=2}^{m} (l + l'_{j-1}) \left(n_j + \theta_j^{-1} \ln\Big(\tfrac{m\zeta S}{\delta}\Big) \right) \\ &= O\left(2M^{\pi} + \sum_{j=1}^{m} \theta_j^{-1} (l + l'_{j-1}) \ln\Big(\tfrac{m\zeta S}{\delta}\Big) \right), \end{aligned}$$

where θ_j is given in (14) with the lower bound in Equation (15). Then, the following asymptotic bounds hold:

$$\textbf{\textit{Thoughput}}^* - \mathbb{E}\left[\textbf{\textit{Thoughput}}^{(\zeta,\pi)}\right] = O\left((\zeta T)^{-1}\right),$$
$$\mathbb{E}\left[\textbf{\textit{Latency}}^{(\zeta,\pi)}\right], \mathbb{E}\left[\textbf{\textit{TTFT}}^{(\zeta,\pi)}\right] = O(1).$$

Additionally, the memory usage remains below $M^{\zeta,\pi}$ with probability at least $1-\delta$ over the time hotizon $[0,T]$.

The proof constructs a coupled dominant stochastic process, extending the approach of Theorem 5.1 with adjustments for time-varying arrival rates. A detailed proof is provided in Appendix E.

B.2 Segment Design

In this subsection, we generalize the segment design in Algorithm 2. We assume constant arrival rates, though the approach can be extended to time-varying rates as discussed earlier. Consider m prompt types with decode lengths $l_1' < \cdots < l_m'$ and arrival rates $\lambda_1, \lambda_2, \cdots, \lambda_m$. Our objective is to merge these into L segments, where $L \le m$. For notational convenience, let $\Delta L := m/L$. We partition the m types into L segments as follows: types 1 to ΔL, types $\Delta L + 1$ to $2\Delta L$, ..., types $(L-1)\Delta L + 1$ to m.

For each segment $i \in \{1, \ldots, L\}$, the total arrival rate is defined as $\lambda_i' = \sum_{(i-1)\Delta L+1 \le j \le i\Delta L} \lambda_j$. Additionally, we define $\lambda'(i_1 \to i_2) := \sum_{j=i_1}^{i_2} \lambda_j'$ for $1 \le i_1 \le i_2 \le L$.

Given a policy $\pi = [n_1, \ldots, n_L]$ with threshold n_i for the i-th segment in Algorithm 2, we formulate a linear system:

$$\begin{aligned} &\Delta T_{[1:L]}(n_{1:L}) \le \frac{n_1}{\sum_{j=1}^{L} \lambda_j'}, \\ &\frac{n_{i+1}}{n_i} > p_i, \ \forall i < L, \end{aligned} \tag{11}$$

where $p_i = \frac{\lambda'(i+1 \to L)}{\lambda'(i \to L)}$, $\Delta T_{[1,\ldots,L]}(n_1, \ldots, n_L) = d_0 + d_1 M^\pi(n_1, \ldots, n_L)$, and the memory M^π running with n_i at each stage in segment i is given by:

$$M^\pi(n_1, \ldots, n_L) = \sum_{i=1}^{L} n_i \left(l + \frac{i}{2}\Delta L\right) \Delta L.$$

We define that the total number of iterations over time horizon $[0,T]$ as S. Therefore, we have $T \ge \sum_{s=1}^{S} \Delta T_{s\text{-th Batch}} \ge S \cdot \min \Delta T_{\text{Batch}}$, where $\Delta T_{s\text{-th Batch}}$ denotes the time cost of the s-th iteration and $\Delta T_{\min} = \min_s\{\Delta T_{\text{Batch}}\}$. We demonstrate that the Nested WAIT algorithm (Algorithm 2) with L segments and thresholds $[n_1, \ldots, n_L]$ satisfying Equation (11) achieves asymptotic optimality under the heavy-traffic conditions.

Theorem B.2. *For any fixed number of segments L and thresholds $\pi = [n_1, \ldots, n_L]$ satisfying Equation (11), if the memory capacity $M^{(\zeta,\pi)}$ satisfies:*

$$\begin{aligned} M^{(\zeta,\pi)} &\ge M^\pi + \sum_{j=2}^{L} (l + l_{j-1}')n_j + \sum_{j=2}^{L} (l + l_{j-1}')\theta_j^{-1} \ln\left(\frac{m\zeta S}{\delta}\right) \\ &= O\left(2M^\pi + \sum_{j=1}^{L} (l + l_{j-1}')\theta_j^{-1} \ln\left(\frac{m\zeta S}{\delta}\right)\right), \end{aligned} \tag{12}$$

where θ_j is given by Equation (13) with lower bound $8(n_j - n_{j-1}p_j)/n_j$.. , then the performance of Nested WAIT is guaranteed by:

$$\textbf{\textit{Thoughput}}^* - \mathbb{E}\left[\textbf{\textit{Thoughput}}^{(\zeta,\pi)}\right] = O\left((\zeta T)^{-1}\right),$$
$$\mathbb{E}\left[\textbf{\textit{Latency}}^{(\zeta,\pi)}\right], \mathbb{E}\left[\textbf{\textit{TTFT}}^{(\zeta,\pi)}\right] = O(1).$$

Moreover, memory is not exceeded with probability at least $1-\delta$ over the total batch counts $s \in \{1, \cdots, S\}$ as well as the time horizon $t \in [0,T]$.

The proof follows a similar structure to that of Theorem 5.1, substituting the original arrival rates $(\lambda_1, \ldots, \lambda_m)$ with the clustered rates $(\lambda'_1, \ldots, \lambda'_L)$ as in Appendix D. We omit here for simplicity.

To show how the memory requirements in (12) varies with different model parameters, we utilize the dataset from Zheng et al. [2023], with $m = 500$ prompt types, decode lengths ranging from 1 to 500, and a fixed prefill length of 62. The distribution of prompt types is shown in Figure 3, where arrival rates are set proportional to normalized frequencies.

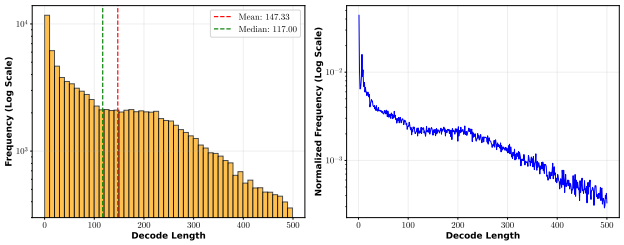


Figure 3: Arrival rate construction from real data, with rates proportional to normalized frequencies.

We analyze the three components of the memory bound in Equation (12): (1) the peak batch memory usage $\sum_{j=1}^{L} n_j \left(l + \frac{j}{2}\Delta\right)\Delta$, (2) the queue length bound $\sum_{j=2}^{L} n_j(l + l'_{j-1})$, and (3) the high-probability queue length bound $\sum_{j=2}^{L} \theta_j^{-1}(l + l'_{j-1}) \ln\left(\frac{(L-1)(T+1)}{\delta}\right)$.

Simulations under varying total arrival rates, time horizons T, and confidence levels δ are presented in Figures 4, 5, and 6. These figures depict the memory usage proportions of the three terms across different segment numbers L, revealing that the overall memory demand is primarily driven by the first term (fluid equilibrium memory) and follows a U-shaped trend with respect to L, suggesting that moderate segment number can reduce waste of memory usage. Additionally, Nested WAIT effectively manages unknown prompt types with only a marginal increase in memory beyond the fluid equilibrium requirement. In practice, a moderate number of segments, such as $L = 10$ as used, is sufficient.

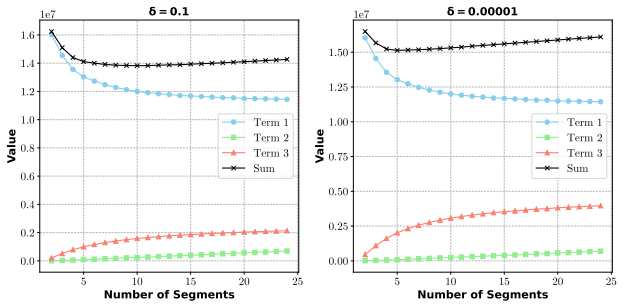


Figure 4: Memory usage proportions under low arrival rate (total rate = 50, $T = 200$). Left: $\delta = 0.1$; Right: $\delta = 10^{-5}$.

C Proof of Theorem 4.1

First we consider the single-type case, and prove some useful lemmas.

C.0.1 Single-Type

First, consider the single-type scenario. For the booking limit policy, it can be inferred from the recursion that the waiting time is necessarily spent on awaiting the accumulation of sufficient prompts for prefill.

Lemma C.1. *The number of incoming prompts X^t at each time step n follows a Poisson distribution with parameter λ. The service processes up λ prompts at each time step when there are at least λ prompts available. Let W^t represent the number of prompts in the queue at time t, starting with*

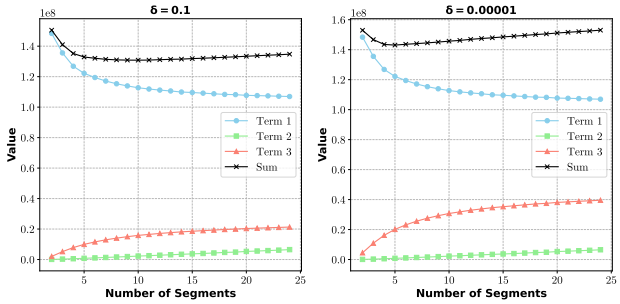


Figure 5: Memory usage proportions under high arrival rate (total rate = 500, $T = 200$). Left: $\delta = 0.1$; Right: $\delta = 10^{-5}$.

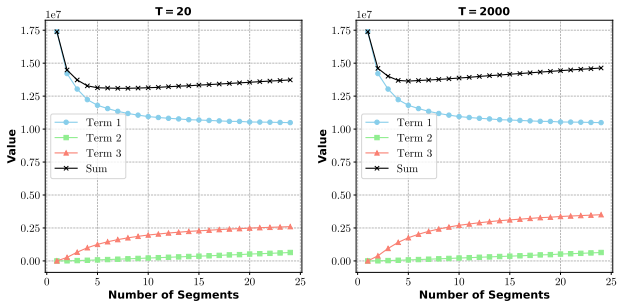


Figure 6: Memory usage proportions with varying time horizons (total rate = 50, $\delta = 0.001$). Left: $T = 20$; Right: $T = 2000$.

$W^0 = 0$. *The state transition rule is given by*

$$W^{t+1} = W^t + X^t - \lambda \cdot \mathbf{1}\{W^t + X^t \geq \lambda\}$$

Define the total number of stuck time steps as T_{stuck}*, which is also the total number of time steps of* $W^t + X^t < \lambda$*, and the following relationship holds*

$$\mathbb{E}[W^T] = \lambda \cdot \mathbb{E}[T_{stuck}].$$

Detailed proof is in Appendix F.1. In order to estimate the expected upper bound, we need to construct a coupled process to dominate W^t.

Lemma C.2. *Define a new process* $\{\tilde{W}^t\}$ *as a coupled version of* $\{W^t\}$ *as follows:*

$$\tilde{W}^0 = 2\lambda, \quad \tilde{W}^{t+1} = \max\{2\lambda, \tilde{W}^t + X^t - \lambda\}.$$

Then, we have $\tilde{W}^t \geq W^t + \lambda, \ \forall t \in \mathbb{N}$.

Detailed proof is in Appendix F.2. Next, by Proposition 6.3 in Asmussen et al. [2003], we have that

Lemma C.3.

$$\tilde{W}^T = 2\lambda + \max(S^T, S^T - S^1, \cdots, S^T - S^{T-1}, 0), \ S^k = \sum_{r=1}^{k}(X^r - \lambda)$$

Let $\xi^k = X^k - \lambda$, and notice that

$$\max(S^T, S^T - S^1, \cdots, S^T - S^{T-1}, 0) = \max_{1\leq k\leq T}(\sum_{r=k}^{T}\xi^r)^+ \overset{d}{=} \max_{1\leq s\leq T}(\sum_{r=1}^{s}\eta^r)^+, \ \eta^r \overset{d}{=} X^1 - \lambda$$

So we have that $\tilde{W}^T \overset{d}{=} 2\lambda + \max_{1\leq s\leq T}(\sum_{r=1}^{s}\eta^r)^+$.

Since $\tilde{W}^T \geq \lambda^T$, we have $\mathbb{E}[W^T] \leq \mathbb{E}[\tilde{W}^T] - \lambda = \mathbb{E}[\max_{1\leq s\leq T}(\sum_{r=1}^{s}\eta^r)^+] + \lambda$. Let $Y^T = \max_{1\leq s\leq T}(\sum_{r=1}^{s}\eta^r)^+$, consider the expectation $\mathbb{E}\left[Y^T\right]$. By Lemma 2 in Strait [1974], we get that

Lemma C.4.

$$\mathbb{E}\left[Y^T\right] = \mathbb{E}\left[\max_{1\le s\le T}(\sum_{r=1}^{s}\eta^r)^+\right] = \mathbb{E}\left[\max_{1\le k\le T}(S^k)^+\right] = \sum_{k=1}^{T}\frac{1}{k}\mathbb{E}\left[(S^k)^+\right]$$

Then we consider the $\mathbb{E}[(S^k)^+]$, and just use Cauchy's inequality

$$\mathbb{E}[(S^k)^+] = \mathbb{E}[\max\{S^k,0\}] \le \mathbb{E}[|S^k|] \le \sqrt{\mathbb{E}[(S^k)^2]} = \sigma\sqrt{k} = \sqrt{\lambda k} \Rightarrow \mathbb{E}[Y^T] \le \sqrt{\lambda}\sum_{k=1}^{T}\frac{1}{\sqrt{k}} \sim \sqrt{\lambda T}$$

So we have that
Lemma C.5.

$$\mathbb{E}[W^T] - \lambda \le \mathbb{E}[Y^T] \le \sqrt{\lambda}\sum_{k=1}^{T}\frac{1}{\sqrt{k}} \sim \sqrt{\lambda T},\ T\to\infty$$

Then consider the average throughput. In the deterministic scenario, the average throughput **Thoughput*** $= \lambda$. Thus, in the stochastic case, the expected gap between the average throughput **Thoughput** and **Thoughput*** is given by $\frac{1}{T}\mathbb{E}[W^T]$.
In **Conventional Heavy-Traffic** case,

$$\frac{1}{T^\zeta}\mathbb{E}[W^{T,\zeta}] = \frac{\lambda^\zeta}{T^\zeta} + \frac{(\lambda T)^{\zeta/2}}{T^\zeta} = \left(\frac{\lambda}{T}\right)^\zeta + \lambda^{\zeta/2}T^{-\zeta/2} \to 0 \quad \text{as } \zeta\to\infty$$

C.0.2 Multiple-Type

When the memory constraint $C \ge M^*$, consider the WAIT policy π, which is defined by the $[n_1,\cdots,n_m]$, where n_j represents the threshold of the number of prompts of type j in each stage.

Solving the linear system:

$$d_0 + d_1\sum_{j=1}^{m} n_j(l_j'+1)\left(l_j+\frac{l_j'}{2}\right) \le \frac{n_j}{\lambda_j},\ M^\pi = \sum_{j=1}^{m} n_j(l_j'+1)\left(l_j+\frac{l_j'}{2}\right) \ge M^*$$

we obtain the policy class Π, each $\pi = [n_1,\cdots,n_m] \in \Pi$ with corresponding memory capacity $M^\pi \ge M^*$.

$\forall\pi\in\Pi$, when prompts of all m types are in the same batch, Arrival$_j \le$ Completion$_j$. Indeed, when type j is in the batch, no matter what the other types are in the batch, we have that Arrival$_j \le$ Completion$_j$. Specifically, when $M^\pi = M$, the policy is $\pi = [\frac{n_1^*}{l_1'+1},\cdots,\frac{n_m^*}{l_m'+1}]$.

Now we define some notations. Let s denote the batch index, J^s denote the types contained in the s-th batch, $s(J^s)$ denote the time at the start of the inference of the s-th batch and $e(J^s)$ denote the time at the end of the inference of the s-th batch. Note that $s(J^{s+1}) - e(J^s)$ denotes the waiting time before $s+1$-th batch, this waiting condition only occurs when all m types do not satisfy the threshold.

At $s+1$-th batch, the state transition rule is given by

$$W_{(j)}^{e(J^{s+1})} = W_{(j)}^{e(J^s)} + X_{(j)}^{s(J^{s+1})-e(J^s)} + Y_{(j)}^{e(J^{s+1})-s(J^{s+1})} - n_j\cdot\mathbf{1}\{j\in J^{s+1}\},\ \forall j=1,\cdots,m$$

where $X_{(j)}^{s(J^{s+1})-e(J^s)}$ and $Y_{(j)}^{e(J^{s+1})-s(J^{s+1})}$ are independent with

$$X_{(j)}^{s(J^{s+1})-e(J^s)} \sim P(\lambda_j\cdot(s(J^{s+1})-e(J^s))),\ Y_{(j)}^{e(J^{s+1})-s(J^{s+1})} \sim P(\lambda_j\cdot(e(J^{s+1})-s(J^{s+1})))$$

Since $s(J^{s+1}) - e(J^s) > 0$ if and only if when all m types do not satisfy the threshold, so $\forall t\in[e(J^s),s(J^{s+1})]$, $W_{(j)}^{e(J^s)} + X_{(j)}^t \le n_j, \forall j\in\{1,\cdots,m\}$. We will use this property to construct a coupled process to dominate this.

Lemma C.6. *Define a coupled process $\tilde{W}^t_{(j)}$ with continuous time:*

$\tilde{W}^0_{(j)} = 2n_j, \tilde{W}^{t+\Delta T_{[1,\cdots,m]}}_{(j)} = \max\{2n_j,\ \tilde{W}^t_{(j)} + X^{\Delta T_{[1,\cdots,m]}}_{(j)} - n_j\},\ X^{\Delta T_{[1,\cdots,m]}}_{(j)} \sim P(\lambda_j \cdot \Delta T_{[1,\cdots,m]})$

We have that $\tilde{W}_t \geq W_t,\ \forall t \geq 0$

Detailed proof is in Appendix F.3. For the coupled process $\tilde{W}^t_{(j)}$, since the processing time of a batch containing all m types $\Delta T_{[1,\cdots,m]}$ is the same under fixed thresholds $[n_1, \cdots, n_m]$:

$$\Delta T_{[1,\cdots,m]} = d_0 + d_1 \cdot \sum_{j=1}^{m} n_j(l'_j + 1)\left(l_j + \frac{l'_j}{2}\right),$$

we can use batch index s for simplicity. So the coupled process is equivalent to

$\tilde{W}^0_{(j)} = 2n_j,\ \tilde{W}^{s+1}_{(j)} = \max\{2n_j,\ \tilde{W}^s_{(j)} + X^s_{(j)} - n_j\} = \max\{2n_j,\ \tilde{W}^s_{(j)} + Y^s_{(j)}\},\ X^s_{(j)} \sim P(\lambda_j \cdot \Delta T_{[1,\cdots,m]})$

1. When $\Delta T_{[1,\cdots,m]} = \frac{n_j}{\lambda_j}$, we have that

$$\lambda_j \cdot \Delta T = n_j \Rightarrow \mathbb{E}\left[Y^s_{(j)}\right] = 0.$$

So by Lemma C.5, we have that

$$\mathbb{E}\left[W^S_{(j)}\right] \leq \mathbb{E}\left[\tilde{W}^S_{(j)}\right] \leq \lambda_j + \sqrt{\lambda_j S},$$

2. When $\Delta T_{[1,\cdots,m]} < \frac{n_j}{\lambda_j}$, the expectation $\mathbb{E}\left[Y^s_{(j)}\right] < 0$. The coupled process $\tilde{W}^s_{(j)}$ has a negative drift. Since

$$\text{Var}(Y^s_{(j)}) = \text{Var}(X^s_{(j)}) = \lambda_j \cdot \Delta T_{[1,\cdots,m]},\ \left|\mathbb{E}\left[Y^s_{(j)}\right]\right| = n_j - \lambda_j \cdot \Delta T_{[1,\cdots,m]}$$

By Kingman, we have that

$$\mathbb{E}\left[W^S_{(j)}\right] \leq \mathbb{E}\left[\tilde{W}^S_{(j)}\right] \leq 2n_j + \frac{\text{Var}(Y^s_{(j)})}{2\left|\mathbb{E}\left[Y^s_{(j)}\right]\right|} = 2n_j + \frac{\lambda_j \cdot \Delta T_{[1,\cdots,m]}}{2 \cdot \left(n_j - \lambda_j \cdot \Delta T_{[1,\cdots,m]}\right)}$$

So when $C = M^*$, we have that $\pi = [\frac{n^*_1}{l'_1+1}, \cdots, \frac{n^*_m}{l'_m+1}]$. This is a special case where all types satisfy the condition $\Delta T_{[1,\cdots,m]} = \frac{n_j}{\lambda_j}$. So with the union bound, we have the expected upper bound of the throughput gap to the Throughput* under the fluid system.

$$\frac{1}{S}\sum_{j=1}^{m}\mathbb{E}\left[W^S_{(j)}\right] \leq \frac{1}{S}\sum_{j=1}^{m}\lambda_j + \sum_{j=1}^{m}\sqrt{\frac{1}{S}\lambda_j}$$

In **Conventional Heavy-Traffic** case,

$$\frac{1}{S^\zeta}\sum_{j=1}^{m}\lambda^\zeta_j + \sum_{j=1}^{m}\sqrt{\frac{1}{S^\zeta}\lambda^\zeta_j} = \frac{1}{\zeta S}\sum_{j=1}^{m}\lambda_j + \sum_{j=1}^{m}\sqrt{\frac{1}{\zeta S}\lambda_j} \to 0,\ \zeta \to \infty.$$

With the expected end-to-end latency

$$\frac{1}{S} \cdot \Delta T_{[1,\cdots,m]} \cdot \left(\sum_{j=1}^{m}\lambda_j + \sum_{j=1}^{m}\sqrt{\lambda_j \cdot S}\right) \sim \Omega(\sqrt{\frac{1}{S}})$$

When all m types satisfy the condition $\Delta T_{[1,\cdots,m]} < \frac{n_j}{\lambda_j}$, the expected upper bound of the throughput gap is bounded:

$$\frac{1}{S}\sum_{j=1}^{m}\mathbb{E}\left[W^S_{(j)}\right] \leq \frac{1}{S}\sum_{j=1}^{m}\left(2n_j + \frac{\lambda_j \cdot \Delta T_{[1,\cdots,m]}}{2 \cdot \left(n_j - \lambda_j \cdot \Delta T_{[1,\cdots,m]}\right)}\right)$$

with the bounded expected end-to-end latency

$$\Delta T_{[1,\cdots,m]} \cdot \frac{1}{S}\sum_{j=1}^{m}\mathbb{E}\left[W^S_{(j)}\right] \leq \Delta T_{[1,\cdots,m]}\frac{1}{S}\sum_{j=1}^{m}\left(2n_j + \frac{\lambda_j \cdot \Delta T_{[1,\cdots,m]}}{2 \cdot \left(n_j - \lambda_j \cdot \Delta T_{[1,\cdots,m]}\right)}\right) \sim \Omega(\frac{1}{S})$$

D Proof of Theorem 5.1

For the first segment, the state transition rule is given by

$$W_{(1)}^{s+1} = W_{(1)}^{s} + Y_{(1)}^{s} - n_1 \cdot \mathbf{1}\{W_{(1)}^{s} + Y_{(1)}^{s} \geq n_1\},\ Y_{(1)}^{s} \sim P(\Delta T_{s\text{-th Batch}} \cdot \sum_{j=1}^{m} \lambda_j),$$

where s denotes the index of all batches, and $\Delta T_{s\text{-th Batch}}$ denotes the time cost of the inference of the t-th batch. Since $\Delta T_{s\text{-th Batch}} \leq \Delta T_{[1,\cdots,m]}(n_1, \cdots, n_m)$, where $\Delta T_{[1,\cdots,m]}(n_1, \cdots, n_m)$ is the time cost of the inference of the batch containing all m types, we can obtain a dominated process.

$$\bar{W}_{(1)}^{s+1} = \bar{W}_{(1)}^{s} + \tilde{Y}_{(1)}^{s} - n_1 \cdot \mathbf{1}\{\bar{W}_{(1)}^{s} + \tilde{Y}_{(1)}^{s} \geq n_1\},\ Y_{(1)}^{s} \sim P(\Delta T_{[1,\cdots,m]}(n_1, \cdots, n_m) \cdot \sum_{j=1}^{m} \lambda_j),$$

Compared to process $\{W_{(1)}^{s}\}$, the process $\{\bar{W}_{(1)}^{s}\}$ use $Y_{(1)}^{s} \sim P(\Delta T_{[1,\cdots,m]}(n_1, \cdots, n_m) \cdot \sum_{j=1}^{m} \lambda_j)$ as arrival. And this process is dominated by a Lindley process

$$\tilde{W}_{(1)}^{0} = 2n_1, \tilde{W}_{(1)}^{s+1} = \max\{2n_1, \tilde{W}_{(1)}^{s} + \tilde{Y}_{(1)}^{s} - n_1\}$$

Since $\Delta T_{[1,\cdots,m]}(n_1, \cdots, n_m) \cdot \sum_{j=1}^{m} \lambda_j < n_1$, the coupled process has negative drift. Since $\text{Var}(X_{(1)}^{s} - n_1) = \Delta T_{[1,\cdots,m]}(n_1, \cdots, n_m) \sum_{j=1}^{m} \lambda_j$ and $\left|\mathbb{E}\left[X_{(1)}^{s} - n_1\right]\right| = n_1 - \Delta T_{[1,\cdots,m]}(n_1, \cdots, n_m) \sum_{j=1}^{m} \lambda_j$, by Kingman Inequality, we have that

$$\mathbb{E}\left[W_{(1)}^{s}\right] \leq 2n_1 + \frac{\Delta T_{[1,\cdots,m]} \cdot \sum_{j=1}^{m} \lambda_j}{2\left(n_1 - \Delta T_{[1,\cdots,m]} \cdot \sum_{j=1}^{m} \lambda_j\right)}$$

Notice that the prompts waiting before the first segment is stored in CPU, so do not consume the GPU memory capacity, so $W_{(1)}^{s}$ does not affect the memory capacity.

Then we consider the k-th segment, $2 \leq k \leq m$, the arrival process is that

$$Y_{(k)}^{s} \sim \text{Binomial}\left(n_{k-1},\ \frac{\lambda_k + \cdots + \lambda_m}{\lambda_{k-1} + \lambda_k + \cdots + \lambda_m}\right) = \text{Binomial}(n_{k-1},\ p_k),\ \forall s \in \mathbb{N}$$

The k-th segment can be modeled as a process where the counting of steps is triggered by the process of the former segments, since the arrival of the k-th segment occurs if and only if the $k-1$-th segment is processed in the former batch. And the state transition rule is given by

$$W_{(k)}^{s+1} = W_{(k)}^{s} + Y_{(k)}^{s} - n_k \cdot \mathbf{1}\{W_{(k)}^{t} + Y_{(k)}^{t} \geq n_k\}$$

where s denotes the s-th batch which contains segment $k-1$. So here the index s of each segment are indeed different, but are bounded by the total number of the batches.

We set $X_{(k)}^{s} = Y_{(k)}^{s} - n_k$, $\forall s \in \mathbb{N}$, and we can construct a coupled process $\tilde{W}_{(k)}^{s}$ to dominate $W_{(k)}^{s}$.

Lemma D.1. *Define a new process $\{\tilde{W}_{(k)}^{s}\}$ as a coupled version of $\{W_{(k)}^{s}\}$ as follows:*

$$\tilde{W}_{(k)}^{0} = n_k,\ \tilde{W}_{(k)}^{s+1} = \max\{n_k, \tilde{W}_{(k)}^{s} + X_{(k)}^{s}\},\ \forall s \in \mathbb{N}$$

Then, we have that

$$\tilde{W}_{(k)}^{s} \geq W_{(k)}^{s},\ \forall s \in \mathbb{N}.$$

Detailed proof is in the Appendix F.4, and with the help of Lemma D.1, when estimating the upper bound corresponding to $\{W_{(k)}^{s}\}$, we can estimate $\tilde{W}_{(k)}^{s}$ instead. Note that the process $\{\tilde{W}_{(k)}^{s}\}$ is equivalent to

$$\tilde{W}_{(k)}^{0} = n_k,\ \tilde{W}_{(k)}^{s+1} = n_k + \max\{0, X_{(k)}^{1}, X_{(k)}^{1} + X_{(k)}^{2},\ X_{(k)}^{1} + \cdots + X_{(k)}^{s}\},\ \forall s \in \mathbb{N}$$

Define

$$S_{(k)}^{i} = X_{(k)}^{1} + \cdots + X_{(k)}^{i}$$

and we have that

$$\tilde{W}^0_{(k)} = n_k,\ \tilde{W}^{s+1}_{(k)} = n_k + \max\{0, S^1_{(k)}, \cdots, S^s_{(k)}\},\ \forall t \geq 0, \text{ where } S^0_{(k)} = 0$$

Next, we will estimate the $\mathbb{E}\left[W^s_{(k)}\right]$ for $2 \leq k \leq m$, since the prompts in segments $2 \to m$ consume the memory of the GPU. So we consider the expectation of $\tilde{W}^s_{(k)}$, we have that

$$\mathbb{E}\left[W^s_{(k)}\right] \leq \mathbb{E}\left[\tilde{W}^s_{(k)}\right] \leq n_k + \mathbb{E}\left[\max_{0\leq i\leq s-1}\{S^i_{(k)}\}\right]$$

Since the expectation

$$\mathbb{E}\left[X^s_{(k)}\right] = n_{k-1}\cdot p_k - n_k < 0,\ \forall s \in \mathbb{N}$$

the coupled process $\tilde{W}^s_{(k)}$ has negative drift. Since $\text{Var}(X^i_{(k)}) = \text{Var}(Y^i_{(k)}) = n_{k-1}\cdot p_k(1-p_k)$, and $\left|\mathbb{E}\left[X^i_{(k)}\right]\right| = n_k - n_{k-1}\cdot p_k$, by Kingman Inequality, we have that

$$\mathbb{E}\left[\max_{0\leq i\leq s-1}\{S^i_{(k)}\}\right] \leq \frac{\text{Var}(X^i_{(k)})}{2\left|\mathbb{E}\left[X^i_{(k)}\right]\right|} \leq \frac{n_{k-1}\cdot p_k(1-p_k)}{2\left(n_k - n_{k-1}\cdot p_k\right)}$$

Thus, we obtain the expected upper bound:

$$\mathbb{E}\left[W^t_{(k)}\right] \leq n_k + \frac{n_{k-1}\cdot p_k(1-p_k)}{2\left(n_k - n_{k-1}\cdot p_k\right)}$$

The expectation bound derived above ensures that the expected queue length $\mathbb{E}\left[W^s_{(k)}\right]$ of the k-th segment is finite and controlled, which also provides a useful estimate for the expected memory consumption of the k-th segment.

As we have discussed before, the index s of each segment are different, but are all bounded by the total number of all the batches. So if we use the total number of all the batches S as the upper bound of the index s of different segments, the expected gap of the throughput $\textbf{Thoughput}^* - \mathbb{E}[\textbf{Thoughput}^{(\zeta S,\pi)}]$ is bounded by

$$\frac{1}{S}\sum_{j=1}^{m}\mathbb{E}\left[W^S_{(j)}\right] \leq \frac{2n_1}{S} + \frac{1}{S}\cdot\frac{\Delta T_{[1,\cdots,m]}\cdot\sum_{j=1}^m\lambda_j}{2\left(n_1 - \Delta T_{[1,\cdots,m]}\cdot\sum_{j=1}^m\lambda_j\right)} + \frac{1}{S}\sum_{j=2}^{m}n_k + \frac{1}{S}\sum_{j=2}^{m}\frac{n_{k-1}\cdot p_k(1-p_k)}{2\left(n_k - n_{k-1}\cdot p_k\right)}$$

In **Conventional Heavy Traffic**, we have the estimation of $\textbf{Thoughput}^* - \mathbb{E}[\textbf{Thoughput}^{(\zeta,\pi)}]$:

$$\frac{2n_1}{\zeta S} + \frac{1}{\zeta S}\cdot\frac{\Delta T_{[1,\cdots,m]}\cdot\sum_{j=1}^m\lambda_j}{2\left(n_1 - \Delta T_{[1,\cdots,m]}\cdot\sum_{j=1}^m\lambda_j\right)} + \frac{1}{\zeta S}\sum_{j=2}^{m}n_k + \frac{1}{\zeta S}\sum_{j=2}^{m}\frac{n_{k-1}\cdot p_k(1-p_k)}{2\left(n_k - n_{k-1}\cdot p_k\right)} = O((\zeta S)^{-1})$$

With the end-to-end expected latency and TTFT:

$$\mathbb{E}\left[\textbf{Latency}^{(\zeta,\pi)}\right] \leq \Delta T_{[1,\cdots,m]}\cdot\sum_{j=1}^{m}\frac{1}{\zeta S}\sum_{t=1}^{T}\mathbb{E}\left[W^{(j)}_t\right] \sim O(1)$$

$$\mathbb{E}\left[\textbf{TTFT}^{(\zeta,\pi)}\right] \leq \Delta T_{[1,\cdots,m]}\cdot\frac{1}{\zeta S}\sum_{t=1}^{T}\mathbb{E}\left[W^{(1)}_t\right] \sim O(1)$$

However, while the average is bounded, this does not prevent the queue length $W^s_{(k)}$ from exceeding this limit in specific sample paths due to stochastic fluctuations. In practical terms, there remains a positive probability of memory overflow, where $W^s_{(k)}$ grows large enough to exhaust available resources, potentially disrupting system performance. To address this, we next derive a high-probability bound to quantify the likelihood of such overflows.

As a result, we need to estimate

$$P\left(\max\{0, S^1_{(k)}, \cdots, S^{s-1}_{(k)}\} \geq c\right) = P\left(\max_{0\leq i\leq s-1}\{S^i_{(k)}\} \geq c\right), \forall c > 0, \text{ where } S^0_{(k)} = 0$$

We construct an exponential martingale $M_{(k)}^i = e^{\theta S_{(k)}^i}$. And to make sure it is a martingale, we first consider the moment generating function

$$\phi_{(k)}(\theta) = \mathbb{E}\left[e^{\theta X_{(k)}^i}\right] = e^{-\theta n_k}(1 - p_k + p_k e^{\theta})^{n_{k-1}},$$

and find $\theta > 0$ to make the moment generating function equal 1 to make sure the $M_{(k)}^i$ is a martingale. The following lemma shows that the solution $\theta > 0$ exists. And detailed proof is in the Appendix F.5.

Lemma D.2. *Suppose $n_{k-1} > n_k > n_{k-1}p_k$ and $p_k \in (0,1)$. Then there exists a unique solution $\theta_k > 0$ to the equation*

$$e^{-\theta_k n_k}(1 - p_k + p_k e^{\theta_k})^{n_{k-1}} = 1, \tag{13}$$

where θ_k is strictly increasing with respect to $D = n_k - n_{k-1}p_k$ and satisfies the lower bound

$$\theta_k \geq \frac{8(n_k - n_{k-1}p_k)}{n_{k-1}}.$$

So when $\theta_k > 0$ satisfies $\phi(\theta_k) = 1$, $M_{(k)}^i$ is a martingale that starts at $M_{(k)}^0 = e^{\theta_k \cdot 0} = 1$.

By Doob's inequality, we have that

$$P(\max_{1 \leq i \leq s-1}\{e^{\theta_k S_{(k)}^i}\} \geq a) \leq \frac{\mathbb{E}\left[M^i\right]}{a} = \frac{\mathbb{E}\left[M^0\right]}{a} = \frac{1}{a}.$$

So we have that

$$P(\max_{1 \leq i \leq s-1}\{e^{\theta S_{(k)}^i}\} \geq e^{\theta c}) \leq \frac{1}{e^{\theta c}} \Rightarrow P(\max_{1 \leq i \leq s-1}\{S_{(k)}^i\} \geq c) \leq \frac{1}{e^{\theta c}}.$$

So consider the probability, we have

$$P\left(W_{(k)}^s \geq c\right) < P\left(\tilde{W}_{(k)}^s \geq c\right) = P(\max_{1 \leq i \leq s-1}\{S_{(k)}^i\} \geq c - n_k) \leq e^{-\theta_k(c-n_k)}$$

So for $2 \leq k \leq m, 0 \leq t \leq T$, we have that

$$P\left(W_{(k)}^t \geq c\right) < e^{-\theta_k(c-n_k)}$$

where $\theta_k > 0$ is the solution to (13).

To determine the memory lower bound, we need to ensure that the memory consumption does not exceed the bound at any $s \in \{1, \ldots, S\}$ with probability at least $1 - \delta$. For each segment $k \in \{2, \ldots, m\}$ and total batch index $s \in \{1, \ldots, S\}$, we have:

$$P\left(W_{(k)}^s \geq c\right) < e^{-\theta_k(c-n_k)},$$

There are $m - 1$ segments (from $k = 2$ to m) and S batch counts (from $s = 1$ to S). We allocate the total failure probability δ across all segments and batch counts. The total number of events is $(m-1) \cdot S$. Set the overflow probability for each segment and batch counts to:

$$P\left(W_{(k)}^s \geq c_k\right) \leq \frac{\delta}{(m-1)\cdot S} \Rightarrow e^{-\theta_k(c_k-n_k)} \leq \frac{\delta}{(m-1)\cdot S} \Rightarrow c_k \geq n_k + \frac{\ln\left(\frac{(m-1)\cdot S}{\delta}\right)}{\theta_k}$$

If we choose

$$c_k = n_k + \frac{\ln\left(\frac{(m-1)\cdot S}{\delta}\right)}{\theta_k},$$

then we have that

$$P\left(W_{(k)}^s \geq c_k\right) \leq \frac{\delta}{(m-1)\cdot S}.$$

Next, we compute the joint probability of overflow across all segments and batch counts using a union bound:

$$P\left(\bigcup_{k=2}^{m}\bigcup_{s=1}^{S}\{W_{(k)}^s \geq c_k\}\right) \leq \sum_{k=2}^{m}\sum_{s=1}^{S} P\left(W_{(k)}^s \geq c_k\right) \leq (m-1)\cdot S \cdot \frac{\delta}{(m-1)\cdot S} = \delta$$

Thus:

$$P\left(W^s_{(k)} \le c_k \text{ for all } \begin{matrix} k \in \{2,\dots,m\}, \\ s \in \{1,\dots,S\} \end{matrix}\right) \ge 1-\delta.$$

So the memory consumption is bounded:

$$\text{Memory Consumption}^s = M^\pi + \sum_{j=2}^{m} W^s_{(j)} \cdot (l + l'_{j-1}) \le M^\pi + \sum_{j=2}^{m} c_j \cdot (l + l'_{j-1})$$

where M^π is the peak batch memory usage and is defined by threshold $\pi = [n_1, \cdots, n_m]$

$$M^\pi = \sum_{j=1}^{m} n_j \left(l + \frac{l'_1 + \cdots + l'_j}{2}\right)\left(l'_j - l'_{j-1}\right),\ l'_0 = 0$$

Substitute $c_j = n_j + \frac{\ln\left(\frac{(m-1)\cdot S}{\delta}\right)}{\theta_j}$, we have that

$$\text{Memory Consumption}^s \le M^\pi + \sum_{j=2}^{m} \left(n_j + \frac{\ln\left(\frac{(m-1)\cdot S}{\delta}\right)}{\theta_j}\right) \cdot (l + l'_{j-1}).$$

Define the memory bound $\mathbf{M}^S$:

$$\mathbf{M}^S = M^\pi + \sum_{j=2}^{m} n_j \cdot (l + l'_{j-1}) + \sum_{j=2}^{m} \frac{\ln\left(\frac{(m-1)\cdot S}{\delta}\right)}{\theta_j} \cdot (l + l'_{j-1}).$$

By the lower bound in Lemma D.2, we can obtain the upper bound of the third term:

$$\sum_{j=2}^{m} \frac{\ln\left(\frac{(m-1)\cdot S}{\delta}\right)}{\theta_j} \cdot (l + l'_{j-1}) \le \sum_{j=2}^{m} \frac{n_{k-1}}{8(n_k - n_{k-1}p_k)} \ln\left(\frac{(m-1)\cdot S}{\delta}\right) \cdot (l + l'_{j-1})$$

And this $\mathbf{M}^S$ ensures that with probability at least $1-\delta$, the memory consumption does not exceed $\mathbf{M}^S$ at any batch counts $s \in \{1,\dots,S\}$.

E Proof of Theorem B.1

For the first segment, the state transition rule is

$$W^{s+1}_{(1)} = W^s_{(1)} + Y^s_{(1)} - n_1 \cdot \mathbf{1}\{W^s_{(1)} + Y^s_{(1)} \ge n_1\}, Y^s_{(1)} \sim P(\sum_{j=1}^{m} \lambda_j [t(s), t(s) + \Delta T_{s-\text{th Batch}}])$$

where s denotes the total batch counts, $t(s)$ denotes the start time of s-th batch, and $\Delta T_{s-\text{th Batch}}$ denotes the cost of the inference of the $s-th$ batch. Since $\Delta T_{s-\text{th Batch}} \le \Delta T_{[1,\dots,m]}(n_1,\dots,n_m)$, where $\Delta T_{[1,\dots,m]}(n_1,\dots,n_m)$ is the time cost of the inference of the s-batch containing all m types, we can obtain a dominated process $\{\bar{W}^s_{(1)}\}$. Also notice that when thresholds $\pi = [n_1, \cdots, n_m]$ is fixed, the time cost $\Delta T_{[1,\dots,m]}(n_1,\dots,n_m)$ is time invariant.

$$\bar{W}^{s+1}_{(1)} = \bar{W}^s_{(1)} + \bar{Y}^s_{(1)} - n_1 \cdot \mathbf{1}\{\bar{W}^s_{(1)} + \bar{Y}^s_{(1)} \ge n_1\}, Y^s_{(1)} \sim P(\sum_{j=1}^{m} \lambda_j [t(s), t(s) + \Delta T_{[1,\dots,m]}(n_1,\dots,n_m)])$$

By (10), we have that

$$\sum_{j=1}^{m} \lambda_j [t(s), t(s) + \Delta T_{[1,\dots,m]}(n_1,\dots,n_m)] \le \sup_{\forall s} \left\{\sum_{j=1}^{m} \lambda_j [t(s), t(s) + \Delta T_{[1,\dots,m]}(n_1,\dots,n_m)]\right\} < n_1$$

We denote $\sup_{\forall s} \left\{\sum_{j=1}^{m} \lambda_j [t(s), t(s) + \Delta T_{[1,\dots,m]}(n_1,\dots,n_m)]\right\}$ as $\mathbf{\Lambda}^\pi < n_1$, and we can construct a dominate process with time invariant arrival

$$\tilde{W}^{s+1}_{(1)} = \tilde{W}^s_{(1)} + \tilde{Y}^s_{(1)} - n_1 \cdot \mathbf{1}\{\tilde{W}^s_{(1)} + \tilde{Y}^s_{(1)} \ge n_1\}, \tilde{Y}^s_{(1)} \sim P(\mathbf{\Lambda}^\pi)$$

We next define that $X^s_{(1)} = \tilde{Y}^s_{(1)} - n_1$, and $\{\tilde{W}^s_{(1)}\}$ is dominated by a Lindley process

$$\overline{W}^0_{(1)} = 2n_1,\ \overline{W}^{s+1}_{(1)} = \max\{2n_1, \overline{W}^s_{(1)} + X^s_{(1)}\},\ \forall s \in \mathbb{N}$$

Then we have that

$$W^s_{(1)} \le \bar{W}^s_{(1)} \le \tilde{W}^s_{(1)} \le \overline{W}^s_{(1)}, \forall s \in \mathbb{N}.$$

Since $n_1 > \mathbf{\Lambda}^\pi$, the process $\{\overline{W}^s_{(1)}\}$ has negative drift. Since

$$\text{Var}(X^s_{(1)}) = \mathbf{\Lambda}^\pi,\ \left|\mathbb{E}\left[X^s_{(1)}\right]\right| = n_1 - \mathbf{\Lambda}^\pi$$

By Kingman Inequality, we have that

$$\mathbb{E}\left[W^s_{(1)}\right] \le 2n_1 + \frac{\mathbf{\Lambda}^\pi}{2\left(n_1 - \mathbf{\Lambda}^\pi\right)}$$

Next we consider the k-th segment, $2 \le k \le m$. The arrival for the k-th segment is from the last batch containing the $k-1$-th segment, so the index s to the k-th segment denotes the batch counts of the batch containing the $k-1$-th batch. Notice that, the index s to each segment are different, but are bounded by the total batch counts.

Now consider the s-th arrival of the k-th segment, it comes from the s-th batch containing the $k-1$-th batch and the parameter $p_k(s)$ is defined by the components of this batch. Although we do not exactly know the arrival time of these prompts, but we know that the arrival happens from t to $t + \Delta t$, where $t < t + \Delta t \le t(s, k-1)$, $t(s, k-1)$ is the starting time of the s-th batch containing the $k-1$-th segment. By (10), we define that

$$p_k(s) = \frac{\sum_{j=k+1}^m \lambda_j[t, t+\Delta t]}{\sum_{j=k}^m \lambda_j[t, t+\Delta t]} \le p^*_k < \frac{n_k}{n_{k-1}}, \forall s \in \mathbb{N}.$$

So the arrival process is

$$Y^s_{(k)} \sim \text{Binomial}(n_{k-1}, p_k(s)),\ \forall s \ge 0$$

So the state transition rule is given by

$$W^{s+1}_{(k)} = W^s_{(k)} + Y^s_{(k)} - n_k \cdot \mathbf{1}\{W^s_{(k)} + Y^s_{(k)} \ge n_k\}$$

where s denotes the s-th batch which contains segment $s-1$, since there is new arrival to segment s if and only if the segment $s-1$ is contained in the former batch.

We set $X^s_{(k)} = Y^s_{(k)} - n_k, \forall s \in \mathbb{N}$. Although here the distribution of $Y^s_{(k)} \sim \text{Binomial}(n_{k-1}, p_k(s))$ is time-varying, the coupling construction is still the same as Lemma D.1

Lemma E.1. *Define a new process $\{\tilde{W}^s_{(k)}\}$ as a coupled version of $\{W^s_{(k)}\}$ as follows:*

$$\tilde{W}^0_{(k)} = n_k,\ \tilde{W}^{s+1}_{(k)} = \max\{n_k,\ \tilde{W}^s_{(k)} + X^s_{(k)}\},\ \forall t \ge 0,\ X^s_{(k)} = Y^s_{(k)} - n_1.$$

Then, we have that

$$\tilde{W}^s_{(k)} \ge W^s_{(k)},\ \forall s \in \mathbb{N}$$

However, in order to use the Kingman , we should construct another coupled process with time-invariant arrival. So consider

$$\bar{W}^0_{(k)} = n_k,\ \bar{W}^{s+1}_{(k)} = \max\{n_k,\ \bar{W}^s_{(k)} + \bar{X}^s_{(k)}\},\ \bar{X}^s_{(k)} = \bar{Y}^s_{(k)} - n_k,\ \bar{Y}^s_{(k)} \sim \text{Binomial}(n_{k-1}, p^*_k),\ \forall s \in \mathbb{N}.$$

Then we have that

$$\bar{W}^s_{(k)} \ge \tilde{W}^s_{(k)} \ge W^s_{(k)},\ \forall s \in \mathbb{N}$$

Note that the process $\bar{W}^s_{(k)}$ is equivalent to

$$\bar{W}^0_{(k)} = n_k,\ \bar{W}^{s+1}_{(k)} = n_k + \max_{0 \le i \le t}\{\bar{S}^i_{(k)}\},\ \text{ where } \bar{S}^0_{(k)} = 0, \bar{S}^i_{(k)} = \sum_{j=1}^i \bar{X}^j_{(k)}, 1 \le i \le s$$

Since the expectation

$$\mathbb{E}\left[\bar{X}^s_{(k)}\right] = n_{k-1} \cdot p^*_k - n_k < 0,\ \forall t \in \mathbb{N}$$

the coupled process $\{\bar{W}^s_{(k)}\}$ has a negative drift. Since

$$\text{Var}(\bar{X}^s_{(k)}) = n_{k-1}p^*_k(1-p^*_k),\ \left|\mathbb{E}\left[\bar{X}^s_{(k)}\right]\right| = n_k - p^*_k n_{k-1}$$

by Kingman, we have that

$$\mathbb{E}\left[\max_{0\le i\le t}\{\bar{S}^i_{(k)}\}\right] \le \frac{n_{k-1}p^*_k(1-p^*_k)}{2(n_k - n_{k-1}p^*_k)}$$

Thus, we obtain the expected upper bound:

$$\mathbb{E}\left[W^s_{(k)}\right] \le \mathbb{E}\left[\tilde{W}^s_{(k)}\right] \le \mathbb{E}\left[\bar{W}^s_{(k)}\right] \le n_k + \frac{n_{k-1}p^*_k(1-p^*_k)}{2(n_k - n_{k-1}p^*_k)}$$

The discussion of the asymptotic optimal throughput under bounded latency and TTFT is the same as the discussion in Appendix D. For the high probability bound, it is also similar to the discussion in Appendix D considering the coupled process $\{\bar{W}^s_{(k)}\}$. Recall the definition of $\{\bar{W}^s_{(k)}\}$:

$$\bar{W}^0_{(k)} = n_k,\ \bar{W}^{s+1}_{(k)} = n_k + \max_{0\le i\le t}\{\bar{S}^i_{(k)}\},\ \text{ where } \bar{S}^0_{(k)} = 0, \bar{S}^i_{(k)} = \sum_{j=1}^{i}\bar{X}^j_{(k)}, 1 \le i \le s$$

where $\bar{X}^j_{(k)} = \bar{Y}^j_{(k)} - n_k$, $\bar{Y}^j_{(k)} \sim \text{Binomial}(n_{k-1}, p^*_k)$ and $p^*_k < \frac{n_k}{n_{k-1}}$. The same to the discussion in Appendix D after the Lemma D.2, the memory bound M_S here is:

$$\mathbf{M}_S = M^\pi + \sum_{j=2}^{m} n_j \cdot (l + l'_{j-1}) + \sum_{j=2}^{m}\theta_j^{-1}\ln\left(\frac{(m-1)\cdot S}{\delta}\right)\cdot(l + l'_{j-1})$$

where $\theta_j, 2 \le j \le m$ here is the solution to the equation similar to (13)

$$e^{-\theta_j n_j}(1 - p^*_j + p^*_j e^{\theta_j})^{n_{j-1}} = 1 \tag{14}$$

Since

$$p^*_j = \arg\min_{0<p_j<1}\{n_j - p_j n_{j-1}\},$$

by the strictly increasing property of θ_j in Lemma D.2, the corresponding θ_j is the smallest solution under different $0 < p_j < 1$, thus obtain the biggest term $\sum_{j=2}^{m}\theta_j^{-1}\ln\left(\frac{(m-1)\cdot(S+1)}{\delta}\right)\cdot(l + l'_{j-1})$. And the following discussion considering the lower bound of θ_j is the same in Appendix D.

F Proof of Lemmas

F.1 Proof of Lemma C.1

We take the expectation on both sides of the state transition equation

$$\mathbb{E}\left[W^{t+1}\right] = \mathbb{E}\left[W^t\right] + \lambda - \lambda\mathbb{P}(W^t + X^t \ge \lambda)$$

Summing from $t = 0$ to $T - 1$, and noting that $W_0 = 0$, we get the lemma

$$\mathbb{E}\left[W^T\right] = \lambda T - \lambda\mathbb{E}\left[T - T_{\text{stuck}}\right] = \lambda\cdot\mathbb{E}\left[T_{\text{stuck}}\right]$$

F.2 Proof of Lemma C.2

We prove this by induction. First, for $t = 0$, we have $\tilde{W}^0 = 2\lambda \ge \lambda = W^0 + \lambda$. Next, assume that $\tilde{W}^t \ge W^t + \lambda$ holds for some $t \ge 0$. We now prove that $\tilde{W}^{t+1} \ge W^{t+1} + \lambda$. Consider the following two cases:

1. If $W^t + X^t \ge \lambda$, then $W^{t+1} = W^t + X^t - \lambda$. In this case,

$$\tilde{W}^{t+1} = \max\{2\lambda, \tilde{W}^t + X^t - \lambda\} \ge \tilde{W}^t + X^t - \lambda \ge (W^t + \lambda) + X^t - \lambda = W^{t+1} + \lambda.$$

2. If $W^t + X^t < \lambda$, then $W^{t+1} = W^t + X^t$. In this case,

$$\tilde{W}^{t+1} = \max\{2\lambda, \tilde{W}^t + X^t - \lambda\} \ge 2\lambda = \lambda + \lambda > W^{t+1} + \lambda.$$

Hence, $\tilde{W}^{t+1} \ge W^{t+1} + \lambda$. By the principle of induction, we have $\tilde{W}^t \ge W^t + \lambda$ for all $t \in \mathbb{N}$.

F.3 Proof of Lemma C.6

Since $\tilde{W}^t_{(j)} \geq 2n_j$, when $W^t_{(j)} < n_j$, we have $\tilde{W}^t_{(j)} \geq W^t_{(j)}$.

At first, $W^0_{(j)} = 0$, but $W^t_{(j)}$ will achieve n_j and type j starts to join the batch. And after a period of time, $W^t_{(j)}$ drops back below n_j, and repeats the process above.

In each period $T_0 \leq t \leq T_1$ when $W^t_{(j)} \geq n_j$, we set $\tau = \max\{t \leq T_0 | W^t_{(j)} = n_j\}$. So $\forall t \in [T_0, T_1]$, the number of iterations of $W^t_{(j)}$ and $\tilde{W}^t_{(j)}$ from τ to t is that

$$I_{\tau\to t} \geq \lfloor \frac{t-\tau}{\Delta T_{[1,\cdots,m]}} \rfloor,\ \tilde{I}_{\tau\to t} \leq 1 + \lfloor \frac{t-\tau}{\Delta T_{[1,\cdots,m]}} \rfloor \Rightarrow \tilde{I}_{\tau\to t} \leq 1 + I_{\tau\to t}.$$

So the coupled process at most makes one more iteration than the real process after τ. For example, at τ, the real process is just at a batch inference without type j, but the coupled process just finished a batch inference with all the types, which yields a n_j drop on $\tilde{W}^t_{(j)}$. So at τ, $\tilde{W}^t_{(j)} \geq n_j = W^t_{(j)}$. And for time $(\tau, t]$, since the arrival dynamic is the same between the two processes, at the real process type j joins every batch, and the batch may not contain all types compared with the coupled process. So the numbers of the finish of a batch of coupled process is at most the same as the real ones. So $\tilde{W}^t_{(j)} \geq W^t_{(j)}$ holds during $(\tau, t]$, hence $\forall t \geq 0$.

F.4 Proof of Lemma D.1

We prove the dominance by induction. When $t = 0$, it is obvious that $\tilde{W}^0_{(k)} = n_k \geq 0 = W^t_{(k)}$. Next, we assume that $\tilde{W}^t_{(k)} \geq W^t_{(k)}$ holds for some $t \geq 0$. We now prove that $\tilde{W}^{t+1}_{(k)} \geq W^{t+1}_{(k)}$. Consider the following two cases:

1. If $W^t_{(k)} + Y^t_{(k)} \geq n_k$, then $W^t_{(k+1)} = W^t_{(k)} + Y^t_{(k)} - n_k$. In this case,

$$\tilde{W}^{t+1}_{(k)} = \max\{n_k, \tilde{W}^t_{(k)} + X^t_{(k)}\} \geq \tilde{W}^t_{(k)} + X^t_{(k)} = W^t_{(k)} + Y^t_{(k)} - n_k = W^t_{(k+1)}$$

2. If $W^t_{(k)} + Y^t_{(k)} < n_k$, then $W^t_{(k+1)} = W^t_{(k)} + Y^t_{(k)}$. In this case,

$$\tilde{W}^{t+1}_{(k)} = \max\{n_k, \tilde{W}^t_{(k)} + X^t_{(k)}\} \geq n_k > W^t_{(k+1)}$$

F.5 Proof of Lemma D.2

First, we prove that the solution $\theta_k > 0$ exists and is unique and for simplicity, we denote $\theta = \theta_k$ here. Define the function

$$f(\theta) = e^{-\theta n_k}(1 - p_k + p_k e^{\theta})^{n_{k-1}}.$$

We need to show that there exists $\theta > 0$ such that $f(\theta) = 1$. First, evaluate $f(\theta)$ at $\theta = 0$:

$$f(0) = e^0(1 - p_k + p_k \cdot 1)^{n_{k-1}} = 1.$$

Thus, $\theta = 0$ is a solution, but we seek the solution $\theta > 0$. Next, we consider the logarithm of $f(\theta)$

$$g(\theta) = \ln f(\theta) = -\theta n_k + n_{k-1} \ln(1 - p_k + p_k e^{\theta}),$$

$$g'(\theta) = -n_k + n_{k-1} \cdot \frac{p_k e^{\theta}}{1 - p_k + p_k e^{\theta}}.$$

At $\theta = 0$,

$$g(0) = 0,\ g'(0) = -n_k + n_{k-1} p_k.$$

Since $\mathbb{E}[X^s_{(k)}] = n_{k-1}p_k - n_k < 0$, we have $g'(0) < 0$, so $f(\theta)$ is decreasing near $\theta = 0$. Consider the second derivative:

$$g''(\theta) = n_{k-1} \cdot \frac{p_k e^{\theta}(1 - p_k)}{(1 - p_k + p_k e^{\theta})^2} > 0 \quad \text{for} \quad \theta > 0,$$

since $p_k \in (0, 1)$ and $n_{k-1} > 0$. Thus, $g(\theta)$ is convex, and so is $f(\theta)$. Now, consider the limit as $\theta \to \infty$:

$$f(\theta) \approx p_k^{n_{k-1}} e^{\theta(n_{k-1} - n_k)}.$$

By our settings, $n_{k-1} > n_k$, this ensures that $f(\theta) \to \infty, \theta \to \infty$. Thus, $f(\theta)$ is continuous, convex, starts at $f(0) = 1$, decreases below 1 for small $\theta > 0$, and increases to infinity. By the intermediate value theorem, there exists only one $\theta > 0$ such that $f(\theta) = 1$.

Next, we prove the monotonicity of the solution $\theta > 0$ with respect to $D = n_k - n_{k-1}p_k$. Recall that $g(0) = 0$, $g'(0) = -D < 0$, and $g''(\theta) > 0$ for $\theta > 0$, so $g(\theta)$ is strictly convex. The unique positive solution $\theta > 0$ occurs where $g(\theta)$ crosses zero after initially decreasing from $g(0) = 0$. Consider two values D_1 and D_2 such that $D_1 < D_2$, with corresponding functions $g_1(\theta) = -\theta(n_{k-1}p_k + D_1) + n_{k-1}\ln(1 - p_k + p_k e^{\theta})$ and $g_2(\theta) = -\theta(n_{k-1}p_k + D_2) + n_{k-1}\ln(1 - p_k + p_k e^{\theta})$, and solutions θ_1 and θ_2, respectively.

At $\theta = 0$, $g_1'(0) = -D_1 > g_2'(0) = -D_2$, meaning $g_2(\theta)$ decreases more steeply from zero than $g_1(\theta)$. For a fixed $\theta > 0$, the partial derivative $\frac{\partial g}{\partial D} = -\theta < 0$. Thus, increasing D decreases $g(\theta)$ pointwise. Since $g(\theta)$ is convex and approaches infinity as $\theta \to \infty$, the zero crossing shifts to a larger θ to restore $g(\theta; D) = 0$. Hence, if $D_2 > D_1$, then $\theta_2 > \theta_1$.

Finally, we provide asymptotic behavior and bounds for the solution $\theta > 0$ in terms of D, n_{k-1}, and p_k. When D is small, θ is also small. Expand $g(\theta)$ around $\theta = 0$, we have that $g(\theta) = g(0) + g'(0)\theta + \frac{1}{2}g''(0)\theta^2 + O(\theta^3) = 0 - D\theta + \frac{1}{2}n_{k-1}p_k(1 - p_k)\theta^2 + O(\theta^3)$. Setting $g(\theta) = 0$ and neglecting higher-order terms, we obtain that $-D\theta + \frac{1}{2}n_{k-1}p_k(1 - p_k)\theta^2 \approx 0 \implies \theta \approx \frac{2D}{n_{k-1}p_k(1-p_k)}$. Thus, for small D, $\theta = O(D)$, and specifically, $\theta \approx \frac{2(n_k - n_{k-1}p_k)}{n_{k-1}p_k(1-p_k)}$.

Next we apply Taylor's theorem with remainder to find the general lower bound of the solution $\theta > 0$.

$$g(\theta) = g(0) + g'(0)\theta + \frac{1}{2}g''(c)\theta^2 = -D\theta + \frac{1}{2}g''(c)\theta^2 = 0 \implies \theta = \frac{2D}{g''(c)}. \quad \text{for some } c \in (0, \theta)$$

So we need to find an upper bound for $g''(c)$, since $g''(x) = n_{k-1} \cdot \frac{p_k e^x (1-p_k)}{(1-p_k+p_k e^x)^2}$, we define $h(y) = \frac{p_k y(1-p_k)}{(1-p_k+p_k y)^2}$ where $y = e^x \geq 1$. The function $h(y)$ achieves a maximum of $\frac{1}{4}$ over $y \geq 1$ and $0 < p_k < 1$. Thus we have that $g''(x) \leq n_{k-1} \cdot \frac{1}{4} = \frac{n_{k-1}}{4} \implies \theta = \frac{2D}{g''(c)} \geq \frac{2D}{n_{k-1}/4} = \frac{8D}{n_{k-1}}$. So a general lower bound is:

$$\theta \geq \frac{8(n_k - n_{k-1}p_k)}{n_{k-1}}. \tag{15}$$