MoE-SpAc: Efficient MoE Inference Based on Speculative Activation Utility in Heterogeneous Edge Scenarios

Shuhuai Li
Shanghai Jiao Tong University
lishuhuai@sjtu.edu.cn

Jianghao Lin*
Shanghai Jiao Tong University
linjianghao@sjtu.edu.cn

Dongdong Ge
Shanghai Jiao Tong University
ddge@sjtu.edu.cn

Yinyu Ye
Shanghai Jiao Tong University
yinyuye@sjtu.edu.cn

Abstract

Mixture-of-Experts (MoE) models enable scalable performance but face severe memory constraints on edge devices. Existing offloading strategies struggle with I/O bottlenecks due to the dynamic, low-information nature of autoregressive expert activation. In this paper, we propose to repurpose Speculative Decoding (SD) not merely as a compute accelerator, but as an informative *lookahead sensor* for memory management, supported by our theoretical and empirical analyses. Hence, we introduce **MoE-SpAc**, an MoE inference framework that integrates a training-free *Speculative Utility Estimator* to track expert demand, a *Heterogeneous Workload Balancer* to dynamically partition computation via online integer optimization, and an *Asynchronous Execution Engine* to unify the prefetching and eviction in the same utility space. Extensive experiments on seven benchmarks demonstrate that MoE-SpAc achieves a **42%** improvement in TPS over the SOTA SD-based baseline, and an average **4.04**× speedup over all standard baselines. Code is available at https://github.com/lshAlgorithm/MoE-SpAc.

1 Introduction

Large Language Models (LLMs) have achieved remarkable capabilities by scaling to hundreds of billions or even trillions of parameters (Brown et al., 2020; Ouyang et al., 2022; Achiam et al., 2023). The Mixture-of-Experts (MoE) architecture has been instrumental in this growth, offering a path to vastly larger models while keeping the computational cost manageable (Jacobs et al., 1991; Zhou et al., 2022; Roller et al., 2021). By routing each input token through only a small subset of expert networks, MoE dramatically reduces the required floating-point operations (FLOPs), compared to dense models of similar size.

However, this parameter efficiency imposes a severe memory penalty. The immense parameter footprint creates a critical barrier to deployment in resource-constrained environments, such as personal devices and edge hardware, besides To this end, *heterogeneous offloading* has become the standard solution (Pan et al., 2026; Zhu et al., 2026a). Typically, the full set of expert weights resides in high-capacity CPU memory, while activated experts are transferred on-demand to the high-throughput GPU device.

Existing offloading strategies generally fall into two categories. *The first* strategy involves **GPU intensive** expert calculation. It offloads activated expert weights from CPU to GPU on demand,

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

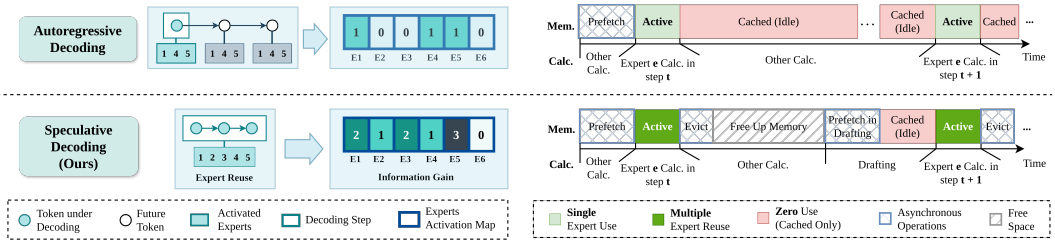


Figure 1: Two key observations how speculative decoding (SD, Bottom) better benefits MoE inference than autoregressive decoding (AR, Top). **Theoretically** (Left), SD enables expert reuse and transforms activation utility signals from AR into informative frequency-valued ones. **Practically** (Right), experts can be scheduled in parallel by proactively leaving more memory for experts to come and asynchronously prefetching experts during the drafting phase.

which significantly exacerbates the I/O bottleneck between heterogeneous devices (Fang et al., 2025a). While predictive *prefetching* attempts to hide this latency using auxiliary networks (Du et al., 2024; Hwang et al., 2024) or historical activation[2] patterns (Li et al., 2023a; Xue et al., 2024; Madan et al., 2026), these methods suffer from unavoidable prediction errors due to the binary, low-information signals during the autoregressive (AR) generation. *The second* strategy is **CPU-GPU hybrid** expert calculation, which allows CPU execution of experts that are missed in GPU VRAM to reduce expert loading overhead. Fiddler (Kamahori et al., 2024) and kTransformers (Chen et al., 2025a) adopt this method but rely on static expert allocation or profiling, failing to capture the dynamic nature of expert activation (Yu et al., 2025). Although HybriMoE (Zhong et al., 2025) introduces a dynamic strategy, its prefetching and caching algorithms remain decoupled, preventing a unified scheduling objective and resulting in suboptimal load balancing (Chen et al., 2026), more relevant works are attached in Appendix I.

In this paper, we identify a novel synergy between speculative decoding (SD) (Leviathan et al., 2023; Xia et al., 2022) and heterogeneous MoE inference. We propose to repurpose SD not merely as an accelerator, but as an informative lookahead sensor for memory management. Typically, SD accelerates inference by using a smaller draft model to generate candidate tokens, which the larger target model verifies in a parallel forward pass. On top of that, we observe that the SD paradigm offers decisive advantages for MoE inference from both theoretical and practical aspects.

Theoretically, SD *reuses* the expert weights and *enriches* the transmission of expert activation signals. As shown in Figure 1(Left), while autoregressive decoding yields a binary indicator (activated or not) for a single step. Hence, SD transforms the low-information binary signals of autoregressive decoding into non-binary, informative signals for expert scheduling. Furthermore, SD introduces inherent *fault tolerance*: exact frequency prediction is unnecessary; coarse-grained utility estimation is sufficient to guide effective scheduling policies.

Practically, as shown in Figure 1(Right), SD enables *parallel scheduling*, where experts can be prefetched in I/O while the GPU generates draft tokens. Additionally, SD presents hot and cold experts in a single step to optimize *heterogeneous compute utilization*: by offloading low-frequency experts to the CPU (sequential processing), we reserve the high-throughput GPU for high-frequency experts like E5 on left of Figure 1 (parallel processing), thereby balancing the load effectively across heterogeneous devices.

Leveraging these insights, we propose **MoE-SpAc**, an efficient **MoE** inference framework based on **sp**eculative **ac**tivation utility in heterogeneous edge scenarios. Rather than treating SD merely as a computation accelerator, MoE-SpAc introduces a training-free *Speculative Utility Estimator*. This estimator evaluates expert demand in a stable, discrete utility space via an inertial transition mechanism. Guided by these utilities, a *Heterogeneous Workload Balancer* solves an online integer optimization problem at each layer, dynamically determining a global threshold to partition experts between GPU and CPU based on real-time I/O and memory constraints. These decisions are executed by an *Asynchronous Execution Engine*, which unifies prefetching and eviction under the same utility metric to minimize synchronization overhead. In this way, MoE-SpAc not only masks I/O latency but also harmonizes the computational load across heterogeneous hardware, transforming the memory wall into a manageable expert scheduling problem.

[2]In this paper, unless otherwise specified, "activation" refers to the MoE gating decision (i.e., which experts are selected).

In summary, our contributions are as follows:

- **Paradigm Shift:** We redefine the role of speculative decoding in MoE inference, shifting its paradigm from a mere computation accelerator to an informative *lookahead sensor* for memory management, which is supported by both theoretical and empirical analysis.
- **Unified Scheduling Framework.** We propose *MoE-SpAc*, which integrates speculative decoding for online heterogeneous expert scheduling based on unified expert utilities. MoE-SpAc dynamically harmonizes CPU-GPU workloads, ensuring optimal throughput by adapting to strict I/O and memory constraints in real-time.
- **SOTA Performance:** Extensive experiments on seven benchmarks demonstrate that MoE-SpAc achieves a **42%** speedup in TPS over the best SD-based baseline, and an average **4.04**$\times$ speedup over all standard baselines.

2 Preliminary and Problem Formulation

2.1 Speculative Decoding

Speculative Decoding (SD) accelerates inference by leveraging a lightweight *draft model* to generate a sequence of γ candidate tokens, which are verified in parallel by the larger *target model*. This parallelism maximizes GPU compute utilization compared to the serial nature of autoregressive decoding, generating $[1, \gamma + 1]$ tokens at once. Let α denote the acceptance probability of a draft token. The expected number of tokens generated per decoding step, $\Omega(\gamma, \alpha)$, is given by Leviathan et al. (2023):

$$\Omega(\gamma, \alpha) := \mathbb{E}[\text{\# Generated Tokens}] = \frac{1 - \alpha^{\gamma+1}}{1 - \alpha}. \tag{1}$$

The latency of generating $\Omega(\gamma, \alpha)$ tokens using speculative decoding (T_{SD}) or autoregressive decoding (T_{AR}) is:

$$\begin{aligned} T_{SD} &= \gamma \cdot T_D + T_V, \\ T_{AR} &= \Omega(\gamma, \alpha) \cdot T_V, \end{aligned} \tag{2}$$

where T_D is the time elapsed for the draft model to generate a single token, and T_V is the latency of a single parallel verification pass by the target model. Hence, the wall-clock speedup factor is $T_{AR}/T_{SD} = \frac{1-\alpha^{\gamma+1}}{(1-\alpha)(\gamma c+1)}$, where $c = T_D/T_V$ represents the cost coefficient.

2.2 Mixture-of-Experts Architecture

Unlike dense models that utilize all parameters for every forward pass, MoE models conditionally activate only a subset of parameters per token. Formally, let $\mathbf{h} \in \mathbb{R}^d$ denote the input hidden state for a token at a specific MoE layer. This layer contains a set of N experts, $\mathcal{E} = \{E_1, E_2, \ldots, E_N\}$, where each expert $E_i(\cdot)$ is typically a feed-forward network (FFN). The output $\mathbf{y}$ is the weighted aggregation of activated experts under a Top-k ($k \ll N$) routing strategy:

$$\begin{aligned} \mathbf{s} &= \mathbf{W}_g \mathbf{h} \in \mathbb{R}^N \\ \mathcal{T}(\mathbf{h}) &= \text{Top-}k(\mathbf{s}) \subset \{1, \ldots, N\}, \\ g_i(\mathbf{h}) &= \frac{\exp(\mathbf{s}_i)}{\sum_{j \in \mathcal{T}(\mathbf{h})} \exp(\mathbf{s}_j)}, \quad i \in \mathcal{T}(\mathbf{h}), \\ \mathbf{y} &= \sum\nolimits_{i \in \mathcal{T}(\mathbf{h})} g_i(\mathbf{h}) \cdot E_i(\mathbf{h}), \end{aligned} \tag{3}$$

where $\mathbf{W}_g \in \mathbb{R}^{N \times d}$ is the learnable routing matrix, $\mathcal{T}(\mathbf{h})$ is the indices of k activated experts. and $g_i(\mathbf{h})$ is the normalized weight for expert E_i. Thus, the majority $(N - k)$ of experts remain inactive, creating significant sparsity.

2.3 Online Heterogeneous Expert Scheduling

We formalize the efficient inference of MoE models on memory-constrained edge devices as an online decision-making problem, termed *Heterogeneous Expert Scheduling*. The objective is to dynamically

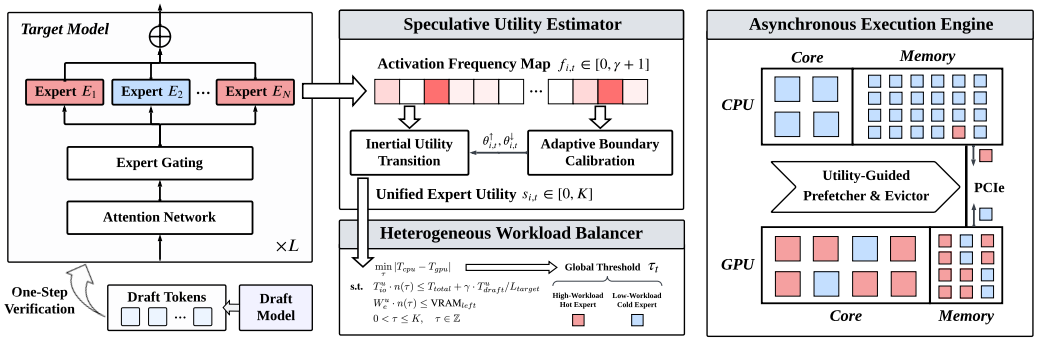


Figure 2: Overall framework of MoE-SpAc.

prioritize high-frequency experts (hot) to GPU while offloading the computation of low-frequency experts (cold) to CPU at each inference step to minimize the I/O-induced latency.

Inference Step. Let t denote a discrete inference step to generate a sequence of tokens $\mathbf{X}_t$. In autoregressive (AR) decoding, a step t corresponds to the generation of a single token, i.e., $|\mathbf{X}_t| = 1$. In speculative decoding (SD), a step t corresponds to a verification cycle that potentially generates multiple tokens, i.e., $\mathbb{E}|\mathbf{X}_t| = \Omega(\gamma, \alpha) \geq 1$.

Activation Frequency. For a given step t and input tokens $\mathbf{X}_t$, we define the *observed* activation frequency $f_{i,t}$ for expert E_i as the cumulative number of times that E_i is activated in Eq. 3 over tokens $\mathbf{X}_t$. For AR, $f_{i,t} \in \{0, 1\}$, providing a low-information, binary signal. For SD, $f_{i,t} \in [0, \gamma + 1]$, providing a more informative, non-binary signal that reflects the *intensity* of expert demand.

Activation Utility Estimation. To guide the expert scheduling, it requires a metric of expert priority for the *upcoming* step $t + 1$. Let $u_{i,t+1}$ denote the *ground-truth utility*, which is determined by a monotonically non-decreasing mapping function $G(\cdot)$ applied to the future (unknown) frequency:

$$u_{i,t+1} = G(f_{i,t+1}), \tag{4}$$

where $G(\cdot)$ maps frequencies to a discrete utility space (e.g., priority levels). Since $f_{i,t+1}$ is unobservable at step t, we must estimate the future utility $\hat{u}_{i,t+1}$ using a scoring function F based on the current utility and historical frequencies:

$$\hat{u}_{i,t+1} = F\left(\hat{u}_{i,t}, \{f_{i,j}\}_{j=1}^{t}\right). \tag{5}$$

Heterogeneous Resource Allocation. The final scheduling decision is a thresholding operation on estimated utilities. Let τ_t be a dynamic threshold determined by the system's resource constraints at step t. We partition the experts as:

- **Hot Experts** ($\hat{u}_{i,t+1} \geq \tau_t$): Experts with high activation frequency across the speculative window should be prioritized (prefetched) for high-throughput GPU.
- **Cold Experts** ($\hat{u}_{i,t+1} < \tau_t$): Experts with negligible or zero activation should be evicted from GPU to CPU-side execution for load balancing.

To make the scheduling decision with more reliable information, we theoretically demonstrate that beyond simple speedup, SD fundamentally alters the nature of the online heterogeneous expert scheduling by providing *Expert Reuse*, *Information Gain*, and *Fault Tolerance* in Appendix A.

3 MoE-SpAc

3.1 Overview

To resolve the online heterogeneous expert scheduling problem, we present **MoE-SpAc**, an efficient **MoE** inference framework based on **Sp**eculative **Ac**tivation utility for edge scenarios.

As shown in Figure 2, MoE-SpAc consists of three key components: **(1)** *Speculative Utility Estimator* predicts future expert demand for each verification step based on inertial utility transition and adaptive

boundary calibration; **(2)** *Heterogeneous Workload Balancer* harmonizes computation loads between devices by solving layer-wise online integer optimization to determine the optimal global threshold for GPU-CPU expert partitioning; and **(3)** *Asynchronous Execution Engine* actualizes prefetching and eviction decisions based on a *unified* utility metric, managing I/O operations asynchronously without stalling the critical computation. In this way, we adapt speculative decoding to MoE inference in edge scenarios, not merely as a compute accelerator, but also as a core lookahead sensor for critical memory management between heterogeneous devices.

3.2 Speculative Utility Estimator

To effectively guide the heterogeneous scheduling, we have to derive an activation utility that is both reflective of *next-step demand* and robust against *frequency fluctuations*. Due to the temporal locality and smooth evolving of the accumulated activation frequencies across speculative windows, we estimate the activation utility in a compressed discrete space, i.e., $u_{i,t} \in \{0, \dots, K\}$, where K is a fixed utility upper bound. We use $K = 4$, following the logarithmic guideline $\lceil \log_2(f_{\max}+1) \rceil$ for the maximum per-expert window count $f_{\max}$ (eight draft tokens give $\lceil \log_2 9 \rceil = 4$). Here, $u_{i,t} = 0$ indicates a dormant (extremely cold) expert, while $u_{i,t} = K$ represents a fully saturated (extremely hot) expert. As illustrated in Algorithm 1, we conduct *speculative utility estimation* based on the historical activation frequencies at each verification step t, which consists of two key components: *inertial utility transition* and *adaptive boundary calibration*.

Inertial Utility Transition. To better estimate the activation utility dynamically, we introduce an inertial update mechanism. At each step t, the utility $u_{i,t}$ of expert E_i remains unchanged unless the frequency fluctuation, defined as $\Delta_{i,t} = f_{i,t} - f_{i,t-1}$, exceeds a significant margin.[3] Formally, the transition is governed by:

$$u_{i,t+1} \leftarrow \begin{cases} \min(K, u_{i,t}+1) & \text{if } \Delta_{i,t} \geq \theta^{\uparrow}_{i,t} \\ \max(0, u_{i,t}-1) & \text{if } -\Delta_{i,t} \geq \theta^{\downarrow}_{i,t} \\ u_{i,t} & \text{otherwise} \end{cases} \tag{6}$$

where $\theta^{\uparrow}_{i,t}$ and $\theta^{\downarrow}_{i,t}$ are expert-specific fluctuation boundaries at step t to trigger the utility transition upward $(+1)$ or downward (-1). This inertia ensures that the system only incurs the high I/O cost of prefetching or eviction when there is a sustained shift in expert demand, effectively filtering out high-frequency noise.

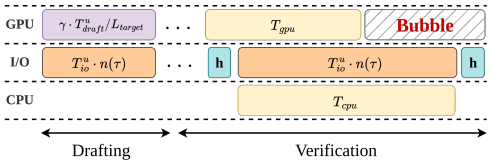
Algorithm 1 Speculative Utility Estimation

Hyperparameters: Utility Upper Bound K, Number of Experts N, Forgetting Factor λ.
Initialize: Utility $u_{i,1} = 0$ for all $i \in \{1, \dots, N\}$.
Initialize: Boundaries $\theta^{\uparrow}_{i,1} = \theta^{\downarrow}_{i,1} = \lfloor \gamma/2 \rfloor$.
for inference step $t = 1, 2, \dots$ **do**
  Observe activation frequencies $\mathbf{f}_t = \{f_{1,t}, \dots, f_{N,t}\}$
  for each expert $i \in \{1, \dots, N\}$ **do**
    $\Delta_{i,t} \leftarrow f_{i,t} - f_{i,t-1}$
    // Inertial Utility Transition
    if $\Delta_{i,t} \geq \theta^{\uparrow}_{i,t}$ **then**
      $u_{i,t+1} \leftarrow \min(K, u_{i,t}+1)$
    else if $-\Delta_{i,t} \geq \theta^{\downarrow}_{i,t}$ **then**
      $u_{i,t+1} \leftarrow \max(0, u_{i,t}-1)$
    else
      $u_{i,t+1} \leftarrow u_{i,t}$
    end if
    // Adaptive Boundary Calibration
    if $\Delta_{i,t} > 0$ **then**
      $\theta^{\uparrow}_{i,t+1} \leftarrow \lfloor (1-\lambda)\theta^{\uparrow}_{i,t} + \lambda \cdot \Delta_{i,t} \rfloor$
      $\theta^{\downarrow}_{i,t+1} \leftarrow \theta^{\downarrow}_{i,t}$
    else if $\Delta_{i,t} < 0$ **then**
      $\theta^{\uparrow}_{i,t+1} \leftarrow \theta^{\uparrow}_{i,t}$
      $\theta^{\downarrow}_{i,t+1} \leftarrow \lfloor (1-\lambda)\theta^{\downarrow}_{i,t} + \lambda \cdot |\Delta_{i,t}| \rfloor$
    else
      $\theta^{\uparrow}_{i,t+1} \leftarrow \theta^{\uparrow}_{i,t}$; $\theta^{\downarrow}_{i,t+1} \leftarrow \theta^{\downarrow}_{i,t}$
    end if
  end for
end for

Adaptive Boundary Calibration. The boundaries are initialized as $\lfloor \gamma/2 \rfloor$, but should be adaptively calibrated at each step to reflect the evolving trend of expert demand inertia. As shown in Algorithm 1, according to the positivity of frequency fluctuation, we update the boundaries via a moving average with a forgetting factor λ. In this way, the system adapts to the current workload characteristics, maintaining high responsiveness during distribution shifts while preserving stability during steady processes.

[3] Since the inertial utility transition does not require fitting to a ground-truth utility, we denote the estimated utility as $u_{i,t}$, instead of $\hat{u}_{i,t}$ for simplicity.

Symbol	Description
τ	Threshold, the decision variable
$n(\tau)$	Number of prefetched experts
$r_c(\tau)$, $r_g(\tau)$	Ratio processed on CPU (sequential) and GPU (parallel), respectively.
T^u_{cpu},T^u_{gpu}	Unit expert inference time on CPU, GPU
T^u_{io}	Unit expert I/O transmission time
T^u_{draft}	Unit time for drafting one token
L_{target}	Number of target model's layer



Pipeline of single layer forward in SD scenario. GPU and CPU stands for calculation on each device. h in I/O stands for the transmission of hidden states of tokens.

Figure 3: Notation and MoE-SpAc pipeline illustration.

We provide empirical and theoretical analysis to support the design of speculative utility estimation in Appendix B.

3.3 Heterogeneous Workload Balancer

With the expert utilities estimated above, now we have to determine the optimal global threshold τ_t that separates hot experts (prioritized on GPU) from cold experts (processed on CPU). To this end, we model this as an online integer optimization problem, solved for *each Transformer layer* of the target model at *each verification step*. For brevity, we omit the subscripts for step t and layer l in the following formulation.

Objective. With notations and illustration in Figure 3, our goal is to balance the computation load between heterogeneous devices to minimize synchronization overhead (bubbles). We can derive the total expert execution times for CPU (T_{cpu}) and GPU (T_{gpu}), respectively:

$$T_{cpu} = r_c(\tau) \cdot \gamma \cdot k \cdot T^u_{cpu}, \tag{7}$$

$$T_{gpu} = r_g(\tau) \cdot b \cdot T^u_{gpu}. \tag{8}$$

The system latency is dominated by the bottleneck device: $T_{total} = \max(T_{cpu}, T_{gpu})$. Hence, the objective is to minimize the difference between execution times of heterogeneous devices, i.e., $|T_{cpu} - T_{gpu}|$.

Constraints. The solution must satisfy the following constraints. (1) *I/O Constraint*: the time used to prefetch $n(\tau)$ new experts to GPU must not exceed the available computation window. As a bonus brought by SD, we can prefetch during the drafting phase. (2) *Memory Constraint*: the total size of newly prefetched experts must fit within the remaining VRAM. (3) *Decision Variable Constraint*: the decision variable τ should fall in the discrete utility space.

Formulation. The utility bound K controls the resolution of the scheduling decision space. The logarithmic guideline in Section 3.2 yields $K = 4$ for our speculative window. Figure 5 (Right) shows that performance stabilizes for $K \geq 4$: smaller bounds merge useful hot/cold distinctions, whereas larger bounds provide little additional benefit. We therefore use $K = 4$ throughout the default configuration.

Combining the objective and constraints, the complete optimization problem is:

$$\min_{\tau} |T_{cpu} - T_{gpu}| \tag{9}$$

$$\textbf{s.t.} \quad T^u_{io} \cdot n(\tau) \leq T_{total} + \gamma \cdot T^u_{draft}/L_{target} \tag{10}$$

$$W^u_e \cdot n(\tau) \leq \text{VRAM Left} \tag{11}$$

$$0 < \tau \leq K, \quad \tau \in \mathbb{Z} \tag{12}$$

where W^u_e stands for the memory space to store an unit expert. For an affine device-time model and a contiguous feasible interval, the optimal threshold is obtained by checking the integers adjacent to the analytical crossing and the feasible boundaries. Monotonicity alone does not imply convexity; Appendix C states the conditions and fallback search. The detailed formulation and solution derivation is provided in Appendix C due to the page limit.

3.4 Unified Asynchronous Execution Engine

To actualize the optimal threshold τ derived by the heterogeneous workload balancer, it requires an efficient mechanism to physically migrate expert weights without stalling the computation pipeline. Hence, we design an *asynchronous execution engine* that orchestrates prefetching and eviction based on the *unified* activation utility $u_{i,t}$.

Utility-Guided Prefetching is orchestrated via a multi-level priority queue that processes expert requests in descending order of utility (from K down to the dynamic threshold τ_t), while maintaining layer-wise temporal order through FIFO sub-queues. Complementarily, *Utility-Guided Eviction* is handled by a state-ordered pool, which indexes all GPU-resident experts by their activation utility. This unified architecture allows the system to achieve scheduling consistency and perform high-speed cache pruning in $O(logN)$ time, ensuring that only the most critical experts occupy the limited GPU VRAM. More implementation details are provided in Appendix D.

Runtime overhead. On Qwen3-30B-A3B, estimator state occupies approximately 30 KB and updating all N experts costs $O(N)$, measured at 0.063 ms per step. Threshold selection takes 0.529 ms per 512-token generation. These operations overlap hidden-state transfers and GPU attention, respectively; state size remains fixed with generation length. Appendix C details the solver assumptions.

4 Experiments

4.1 Experiment Setups

Hardware. We evaluate MoE-SpAc under constrained resources with a single NVIDIA GeForce RTX 4090 GPU, connected to CPU memory via a PCIe 4.0 interface (32GB/s).

Benchmarks. We adopt diverse benchmarks to cover different capabilities of LLMs, including MMLU-Pro (Wang et al., 2024), MT-bench (Zheng et al., 2023), HumanEval (Chen et al., 2021), GSM8K (Cobbe et al., 2021), Alpaca (Taori et al., 2023), CNN/DailyMail (Nallapati et al., 2016), and QA (Kwiatkowski et al., 2019).

Baselines. The baselines fall into two categories. (1) General inference engines, including `Accelerate` (Wolf et al., 2019), `vLLM` (Kwon et al., 2023), and `llama.cpp` (Ollama, 2025). (2) Inference systems tailored for MoE models, including `Mixtral Offloading` (Eliseev and Mazur, 2023), `MoE-Infinity` (Xue et al., 2024), `SP-MoE` (Chen et al., 2025b), `Fate` (Fang et al., 2025b), `HybriMoE` (Zhong et al., 2025), `MoE-SpeQ` (Wang et al., 2025a), and `SpecExp` (Madan et al., 2026) (Speculating Experts). Detailed descriptions of each baseline are given in Appendix E.

Configuration Details. We employ the MoE model `Qwen3-30B-A3B` as the target verifier and a quantized dense model `Qwen3-4B-FP8` as the draft model (Yang et al., 2025). We set the hyper-parameters as follows: draft length $\gamma = 8$, forgetting factor $\lambda = 0.1$. The maximum generation length per request is 512 tokens. We use `-cpu-offload-gb 60` for the vLLM baseline. Following previous works (Xue et al., 2024; Chen et al., 2025b; Svirschevski et al., 2024), we use a batch size of 1 to simulate edge deployment. Experiments on additional models and settings are presented in Section 4.6.

Metric. We adopt tokens per second (TPS) and latency (Lat.) as evaluation metrics, which capture the end-to-end efficiency of the inference system. Exact speculative sampling preserves the target distribution when acceptance/rejection is implemented correctly. Draft quantization affects acceptance, while heterogeneous numerical consistency requires separate validation (Appendix H).

4.2 Overall Performance

Table 1 shows that MoE-SpAc improves average TPS by **4.04**$\times$ over the original baselines and by **41.9%** over the strongest baseline, `llama.cpp-w/SD`. The latter isolates the benefit of heterogeneous expert scheduling beyond SD alone.

Table 1: The tokens per second (TPS) and latency (Lat.) performance of different inference methods. The best results are given in **bold** while the second-best values are underlined. *Rel.Imprv.* indicates the relative improvement of MoE-SpAc against the best baseline. Dashes denote unreported measurements; averages cover all seven benchmarks.

Method	MMLU-Pro		MT-bench		GSM8K		HumanEval		Alpaca		CNN/DM		QA		Average	
	TPS↑	Lat.↓	TPS↑	Lat.↓	TPS↑	Lat.↓	TPS↑	Lat.↓	TPS↑	Lat.↓	TPS↑	Lat.↓	TPS↑	Lat.↓	TPS↑	Lat.↓
Accelerate	1.13	495.37	1.15	435.55	1.21	237.45	1.15	393.60	1.09	350.83	1.27	222.47	1.16	138.42	1.17	324.81
vLLM	2.65	203.74	2.67	187.88	2.67	94.63	2.63	158.46	2.67	137.72	2.57	104.28	2.68	57.43	2.65	134.88
llama.cpp	15.17	37.92	14.34	36.08	15.01	19.21	14.74	30.57	16.86	22.40	15.42	19.46	15.09	11.07	15.23	25.24
llama.cpp-w/ SD	17.29	32.88	17.58	29.03	19.03	15.09	18.78	24.01	18.02	21.51	16.46	18.36	16.49	10.43	17.66	21.62
Mixtral Offload	2.43	230.87	2.36	212.70	2.35	120.25	2.48	177.87	2.33	162.32	2.23	131.72	2.33	70.86	2.36	158.08
MoE-Infinity	3.64	151.43	3.64	136.84	3.63	75.42	3.63	119.53	3.48	107.82	3.56	80.56	3.44	46.13	3.57	102.53
SpecExp	2.96	172.97	2.94	174.15	2.94	174.15	2.93	174.74	2.87	178.40	2.94	174.15	2.83	180.92	2.92	175.64
MoE-SpeQ	4.77	107.34	4.70	108.94	4.83	106.00	4.88	104.92	4.58	111.79	4.57	112.04	4.61	111.06	4.71	108.87
SP-MoE	4.75	116.73	4.46	111.34	4.87	55.78	4.98	86.93	4.83	76.16	4.38	65.24	4.55	34.62	4.69	78.11
Fate	7.93	73.65	7.89	66.89	7.82	39.42	7.82	60.24	7.42	54.83	7.83	40.78	7.83	24.23	7.79	51.43
HybriMoE	12.22	47.62	12.03	43.47	12.38	24.83	11.96	38.18	12.29	32.46	12.37	25.11	11.83	15.45	12.15	32.45
MoE-SpAc (Ours)	**25.80**	**23.11**	**24.10**	**22.26**	**27.82**	**11.34**	**28.24**	**16.93**	**23.15**	**17.88**	**22.78**	**14.31**	**23.56**	**8.36**	**25.06**	**16.31**
Rel.Imprv.	*49.2%*	*29.7%*	*37.1%*	*23.3%*	*46.2%*	*24.8%*	*50.4%*	*29.5%*	*28.5%*	*16.9%*	*38.4%*	*22.1%*	*42.9%*	*19.8%*	*41.9%*	*24.6%*

4.3 Precision of Speculative Utility Estimator

To demonstrate the effectiveness of regarding SD as a lookahead sensor for activation utility, we report the online hot-or-cold prediction accuracy of Speculative Utility Estimation from MoE-SpAc (SD), predicted at every step under different acceptance rates, and that of HybriMoE (AR), predicted at the layer level, in Figure 4. For fairness, we report our precision averaged across all layers with the threshold τ set to 1.

We observe that MoE-SpAc can rapidly learn activation patterns and consistently achieve a prediction accuracy of around 0.87, outperforming AR-based utility modeling approaches such as HybriMoE.

Furthermore, we conduct experiments using less capable draft models (`Qwen3-1.7B` and `Qwen3-1.7B-int4`), which yield lower acceptance rates, and the precision correspondingly declines. This supports the following conclusion: "The better the alignment between the draft model and the target model, the more effective SD is as a lookahead sensor."

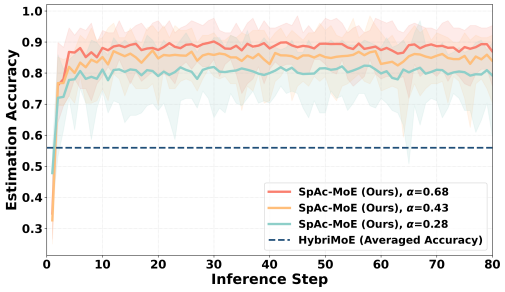


Figure 4: The hot-or-cold online prediction accuracy of MoE-SpAc (SD, step level) with different acceptance rate and HybriMoE (AR, layer level).

Frequency stability. Lower sMAPE accompanies higher prediction accuracy: as sMAPE decreases from 0.889 to 0.322, accuracy rises from 0.770 to 0.897. Appendix B gives the metric definition and cross-model measurements.

4.4 Ablation Study

To assess the impact of each component in MoE-SpAc, we conduct the ablation study on MMLU-Pro, HumanEval, and CNN/DM benchmarks. We define the variants as follows: **(1)** w/o `SpecUE` removes speculative utility estimator, falling back to binary utility modeling with $K = 1$; **(2)** w/o `AdaBC` removes adaptive boundary calibration and uses fixed boundaries ($\theta^{\uparrow}_{i,t} = 3$, $\theta^{\downarrow}_{i,t} = 1$); **(3)** w/o `HetWB` removes heterogeneous workload balancer and employs a static threshold ($\tau = 2$) instead of solving the online optimization; **(4)** w/o `SG-Prefetcher` removes utility-guided prefetcher, falling back to static expert allocation; **(5)** w/o `SG-Evictor` removes utility-guided evictor by retaining experts until VRAM is full; **(6)** w/o `SD` removes speculative decoding, reverting to standard autoregressive inference while retaining utility estimation over a sliding window of 10 tokens and heterogeneous scheduling. The results are summarized in Table 2, and the detailed analysis is attached in Appendix G.

4.5 In-Depth Analysis

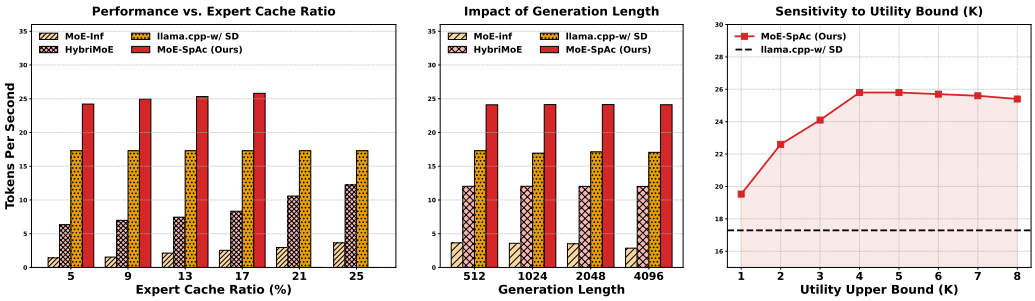


Figure 5: Stability and sensitivity analysis. Left: Impact of expert cache ratios; despite an OOM boundary at 21% due to draft model allocation, MoE-SpAc yields superior throughput compared to existing works. Middle: Scalability across generation lengths, showing consistent gains over baselines in long-context tasks. Right: Effect of the threshold cap K, where small values reduce performance by limiting the precision of the activation utility.

In Figure 5, we analyze the sensitivity and stability of MoE-SpAc to variations in critical system resources and configurations, including expert cache ratio (Left), generation length (Mid), utility upper bound (Right).

Table 2: Ablation study of MoE-SpAc.

Model Variants	MMLU-Pro		HumanEval		CNN/DM	
	TPS↑	Lat.↓	TPS↑	Lat.↓	TPS↑	Lat.↓
MoE-SpAc (Full)	**25.80**	**23.11**	**28.24**	**16.93**	**22.78**	**14.31**
w/o SpecUE	19.53	29.48	20.56	25.27	18.43	19.53
w/o AdaBC	25.33	23.48	28.03	19.03	22.31	15.98
w/o HetWB	24.32	24.32	26.95	19.97	20.16	17.69
w/o SG-Prefetcher	17.27	32.91	18.79	27.44	16.46	21.23
w/o SG-Evictor	18.31	31.23	20.03	25.72	17.16	21.01
w/o SD (AR Mode)	17.25	32.95	16.43	30.13	15.97	22.44

Expert Cache Ratio. As GPU cache capacity decreases, the workload balancer raises τ to satisfy Eq. 11, assigning more marginal experts to CPU. Despite the draft model's approximately 8% memory overhead, MoE-SpAc at a 17% expert-cache ratio outperforms baselines with 25% (Figure 5, Left).

Generation Length. The throughput advantage persists as maximum generation length increases from 512 to 4096 (Figure 5, Mid), supporting stable performance over longer requests.

Utility Upper Bound K. Figure 5 (Right) varies K from 1 to 8. Values below 4 reduce scoring precision and restrict threshold selection; larger values yield little additional benefit, supporting $K = 4$.

Table 3: TPS performance comparison across different models and datasets. We compare our `MoE-SpAc` against best baselines. *Rel. Improv.* is calculated against the strongest baseline.

Hardware	NVIDIA RTX 4090			NVIDIA A100-80GB								
Method	DeepSeek-V2-Lite (16B)			GLM-4.5-Air (106B)			Qwen3.5 (122B)			MiniMax-M2.5 (228B)		
	MT-bench	HumanEval	CNN/DM	MT-bench	HumanEval	CNN/DM	MT-bench	HumanEval	CNN/DM	MT-bench	HumanEval	CNN/DM
HybriMoE	18.43	17.95	18.43	15.34	15.28	16.83	9.25	9.25	9.27	4.47	4.48	4.44
Llama.cpp (w/ SD)	30.78	30.49	33.23	17.11	18.73	17.03	12.46	12.78	12.44	5.35	5.87	5.37
MoE-SpAc (Ours)	**47.10**	**47.26**	**50.17**	**27.47**	**27.79**	**26.98**	**20.76**	**21.42**	**20.27**	**10.13**	**10.89**	**10.03**
Rel. Improv.	*+53.0%*	*+55.0%*	*+51.0%*	*+60.5%*	*+48.4%*	*+58.4%*	*+66.6%*	*+67.6%*	*+62.9%*	*+89.3%*	*+85.5%*	*+86.8%*

4.6 Generalization to Batch Size, Backbone LLM, and Hardware

Batch Size. As depicted in Figure 6, CPU computation, memory bandwidth, and rejected speculative work limit batch scaling; MoE-SpAc benefits from parallel CPU–GPU execution. In the additional batch-size comparison (4–32), MoE-SpAc achieves approximately twice the throughput of HybriMoE, consistent with the benefit of parallel GPU execution for hot experts. Table 4 examines the joint effect of batch size and draft length.

Compatibility to Backbone LLMs and Hardware. We compare our method against the strongest baselines from both categories, using `DeepSeek-V2-Lite`DeepSeek-AI (2024), `GLM-4.5-Air`(Zeng et al., 2025), `Qwen3.5`(Qwen, 2026), and `MiniMax-M2.5`(MiniMaxAI, 2026), with quantized variants serving as draft models. We further include an additional hardware setting, NVIDIA A100-80GB GPUs connected to CPU via PCIe 4.0 (32 GB/s). The results are reported in Table 3. Across these

models, average TPS improvements over the strongest baseline are **52.9%**, **55.77%**, **65.7%**, and **87.2%**, respectively.

Smaller GPU memory. On an RTX 4060 (8 GB), using Qwen3-1.7B to draft for Qwen3-30B-A3B, MoE-SpAc achieves 15.46 TPS versus 11.36 without SD and 11.70 with SD for llama.cpp, a 32.1% gain over the SD baseline.

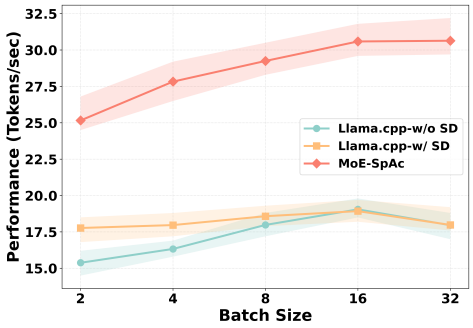


Figure 6: TPS under different batch sizes, with experiment configuration in Section 4.1.

Joint Effect of Batch Size and Draft Length. Table 4 shows that $\gamma = 8$ performs best across batch sizes 1–32. Larger batches increase memory traffic, while rejected drafts add verification work; the optimal pairing depends on acceptance, expert reuse, and memory capacity. At $\gamma = 8$, increasing the batch size from 1 to 32 improves throughput by 29.3%, indicating sublinear scaling. Extending the draft from 8 to 10 tokens yields no gain at any tested batch size; at batch size 32, the difference is only 0.4%. This pattern favors a moderate speculative window that amortizes weight movement without excessive verification work.

Table 4: MoE-SpAc throughput (TPS) across batch size and draft length γ. It shows the best combination of batch sizes and draft lengths in practice.

Batch size	$\gamma = 2$	$\gamma = 4$	$\gamma = 6$	$\gamma = 8$	$\gamma = 10$
1	20.79	22.23	23.92	24.10	23.88
2	20.67	23.42	24.13	26.01	24.93
4	23.83	23.94	25.78	27.52	26.63
8	24.62	24.89	28.01	29.23	28.44
16	25.37	26.41	29.39	30.48	29.31
32	27.50	27.22	29.64	31.15	31.03

5 Conclusion

We present *MoE-SpAc*, which repurposes speculative decoding as a lookahead signal for utility-guided heterogeneous expert scheduling in memory-constrained edge environments. By combining speculative utility estimation, dynamic CPU–GPU workload balancing, and unified asynchronous prefetching and eviction, MoE-SpAc reduces memory-transfer bottlenecks and improves resource utilization. Across seven benchmarks, it delivers a 42% throughput gain over the strongest SD-based baseline and a 4.04× average speedup over standard baselines, with additional evaluations demonstrating applicability across model backbones and hardware configurations.

Our scheduling approach is complementary to advances in speculative decoding. KV-cache compression (Sun et al., 2024) and improved draft–target alignment (Li et al., 2024a) offer opportunities to further increase prediction accuracy. Self-speculation (Zhang et al., 2024) can eliminate a separate draft model, freeing memory for expert caching. Another natural integration path is provided with built-in multi-token prediction (MTP) from frontier models including GLM (Z.ai, 2026), DeepSeek (Xu et al., 2026), Kimi (Team et al., 2026), and Qwen (Team, 2026). Beyond conventional MoE models, architectures such as MoLE (Jie et al., 2025) and Engram (Cheng et al., 2026) introduce additional forms of sparsity and structured memory access, creating opportunities for utility estimation and proactive data placement. As sparse architectures evolve, anticipating which parameters and memory resources will be needed next offers a promising direction for efficient local inference.

Acknowledgments and Disclosure of Funding

Lin's research is partially supported by the National Natural Science Foundation of China (NSFC) [Grant NSFC-624B2096, 72542012, 72595872]. Ge's research is partially supported by the National Natural Science Foundation of China (NSFC) [Grant NSFC-72225009, 72688301, 72394360, 72394365].

References

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. M. Dai, Q. V. Le, J. Laudon *et al.*, "Mixture-of-experts with expert choice routing," *Advances in Neural Information Processing Systems*, vol. 35, pp. 7103–7114, 2022.

S. Roller, S. Sukhbaatar, J. Weston *et al.*, "Hash layers for large sparse models," *advances in neural information processing systems*, vol. 34, pp. 17 555–17 566, 2021.

Y. Pan, Y. He, T. Han, L. Liu, S. Zhao, Z. Chen, M. Wang, C. Li, Y. Han, and Y. Wang, "Trimoe: Augmenting gpu with amx-enabled cpu and dimm-ndp for high-throughput moe inference via offloading," *arXiv preprint arXiv:2603.01058*, 2026.

Z. Zhu, G. Li, P. Wang, Z. Mo, M. Pei, Z. Song, X. Liang, and J. Cheng, "Dali: A workload-aware offloading framework for efficient moe inference on local pcs," *arXiv preprint arXiv:2602.03495*, 2026.

Z. Fang, Z. Hong, Y. Huang, Y. Lyu, W. Chen, Y. Yu, F. Yu, and Z. Zheng, "Accurate expert predictions in moe inference via cross-layer gate," *arXiv e-prints*, pp. arXiv–2502, 2025.

Z. Du, S. Li, Y. Wu, X. Jiang, J. Sun, Q. Zheng, Y. Wu, A. Li, H. Li, and Y. Chen, "Sida: Sparsity-inspired data-aware serving for efficient and scalable large mixture-of-experts models," *Proceedings of Machine Learning and Systems*, vol. 6, pp. 224–238, 2024.

R. Hwang, J. Wei, S. Cao, C. Hwang, X. Tang, T. Cao, and M. Yang, "Pre-gated moe: An algorithm-system co-design for fast and scalable mixture-of-expert inference," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2024, pp. 1018–1031.

J. Li, Y. Jiang, Y. Zhu, C. Wang, and H. Xu, "Accelerating distributed {MoE} training and inference with lina," in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, 2023, pp. 945–959.

L. Xue, Y. Fu, Z. Lu, L. Mai, and M. Marina, "Moe-infinity: Activation-aware expert offloading for efficient moe serving," *arXiv preprint arXiv:2401.14361*, vol. 3, 2024.

V. Madan, P. Singhania, A. Bhatele, T. Goldstein, and A. Panda, "Speculating experts accelerates inference for mixture-of-experts," *arXiv preprint arXiv:2603.19289*, 2026.

K. Kamahori, Y. Gu, K. Zhu, and B. Kasikci, "Fiddler: Cpu-gpu orchestration for fast inference of mixture-of-experts models," 2024.

H. Chen, W. Xie, B. Zhang, J. Tang, J. Wang, J. Dong, S. Chen, Z. Yuan, C. Lin, C. Qiu, Y. Zhu, Q. Ou, J. Liao, X. Chen, Z. Ai, Y. Wu, and M. Zhang, "Ktransformers: Unleashing the full potential of cpu/gpu hybrid inference for moe models," in *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*, 2025.

E. Yu, Z. Zhang, D. Dong, Y. Wu, and X. Liao, "Prescope: Unleashing the power of prefetching for resource-constrained moe inference," *arXiv preprint arXiv:2509.23638*, 2025.

S. Zhong, Y. Sun, L. Liang, R. Wang, R. Huang, and M. Li, "Hybrimoe: Hybrid cpu-gpu scheduling and cache management for efficient moe inference," *arXiv preprint arXiv:2504.05897*, 2025.

Z. Chen, T. Bu, C. Song, X. Lu, Y. Ye, and Z. Zhou, "A universal load balancing principle and its application to large language model serving," *arXiv preprint arXiv:2601.17855*, 2026.

Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19 274–19 286.

H. Xia, T. Ge, P. Wang, S.-Q. Chen, F. Wei, and Z. Sui, "Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation," *arXiv preprint arXiv:2203.16487*, 2022.

Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang *et al.*, "Mmlu-pro: A more robust and challenging multi-task language understanding benchmark," *Advances in Neural Information Processing Systems*, vol. 37, pp. 95 266–95 290, 2024.

L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in neural information processing systems*, vol. 36, pp. 46 595–46 623, 2023.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.

R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.

T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee *et al.*, "Natural questions: a benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th symposium on operating systems principles*, 2023, pp. 611–626.

Ollama, "Ollama," https://github.com/ollama/ollama, 2025, accessed: 2025-09-14.

A. Eliseev and D. Mazur, "Fast inference of mixture-of-experts language models with offloading," *arXiv preprint arXiv:2312.17238*, 2023.

L. Chen, Z. Wen, T. Wu, X. Zhang, and C. Wu, "Sp-moe: Speculative decoding and prefetching for accelerating moe-based model inference," *arXiv preprint arXiv:2510.10302*, 2025.

Z. Fang, Z. Hong, Y. Huang, Y. Lyu, W. Chen, Y. Yu, F. Yu, and Z. Zheng, "Fate: Fast edge inference of mixture-of-experts models via cross-layer gate," *arXiv preprint arXiv:2502.12224*, 2025.

W. Wang, J. Liu, X. Hou, X. Xia, P. Tang, M. Zhang, C. Li, and M. Guo, "Moe-speq: Speculative quantized decoding with proactive expert prefetching and offloading for mixture-of-experts," *arXiv preprint arXiv:2511.14102*, 2025.

A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

R. Svirschevski, A. May, Z. Chen, B. Chen, Z. Jia, and M. Ryabinin, "Specexec: Massively parallel speculative decoding for interactive llm inference on consumer devices," *Advances in Neural Information Processing Systems*, vol. 37, pp. 16 342–16 368, 2024.

DeepSeek-AI, "Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model," 2024.

A. Zeng, X. Lv, Q. Zheng, Z. Hou, B. Chen, C. Xie, C. Wang, D. Yin, H. Zeng, J. Zhang *et al.*, "Glm-4.5: Agentic, reasoning, and coding (arc) foundation models," *arXiv preprint arXiv:2508.06471*, 2025.

Qwen, "Qwen3.5-122b-a10b," https://huggingface.co/Qwen/Qwen3.5-122B-A10B, 2026, model card, accessed 2026-05-04.

MiniMaxAI, "Minimax-m2.7," https://huggingface.co/MiniMaxAI/MiniMax-M2.7, 2026, model card, accessed 2026-05-04.

H. Sun, Z. Chen, X. Yang, Y. Tian, and B. Chen, "Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding," *arXiv preprint arXiv:2404.11912*, 2024.

Y. Li, F. Wei, C. Zhang, and H. Zhang, "Eagle: Speculative sampling requires rethinking feature uncertainty," *arXiv preprint arXiv:2401.15077*, 2024.

J. Zhang, J. Wang, H. Li, L. Shou, K. Chen, G. Chen, and S. Mehrotra, "Draft& verify: Lossless large language model acceleration via self-speculative decoding," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 11 263–11 282.

Z.ai, "Glm-5.3: Frontier coding with emergent cyber capabilities," 2026. [Online]. Available: https://www.alphaxiv.org/abs/2608.glm-5

A. Xu, B. Lin, B. Xue, B. Wang, B. Xu, B. Wu, B. Zhang, C. Lin, C. Dong, C. Ling *et al.*, "Deepseek-v4: Towards highly efficient million-token context intelligence," *arXiv preprint arXiv:2606.19348*, 2026.

K. Team, T. Bai, Y. Bai, Y. Bao, J. Cai, X. Cai, P. Cao, Y. Cao, Z. Chai, Y. Charles *et al.*, "Kimi k3: Open frontier intelligence," *arXiv preprint arXiv:2607.24653*, 2026.

Q. Team, "Qwen3. 8-omni: Towards native omni-modal agents," *arXiv preprint arXiv:2609.25611*, 2026.

S. Jie, Y. Tang, K. Han, Y. Li, D. Tang, Z.-H. Deng, and Y. Wang, "Mixture of lookup experts," *arXiv preprint arXiv:2503.15798*, 2025.

X. Cheng, W. Zeng, D. Dai, Q. Chen, B. Wang, Z. Xie, K. Huang, X. Yu, Z. Hao, Y. Li, H. Zhang, H. Zhang, D. Zhao, and W. Liang, "Conditional memory via scalable lookup: A new axis of sparsity for large language models," *arXiv preprint*, 2026. [Online]. Available: https://arxiv.org/abs/2601.07372

Z. Huang, L. Zhu, Z. Zhan, T. Hu, W. Mao, X. Yu, Y. Liu, and T. Zhang, "Moesd: Unveil speculative decoding's potential for accelerating sparse moe," *arXiv preprint arXiv:2505.19645*, 2025.

O. Dietrich, J. G. Raya, S. B. Reeder, M. F. Reiser, and S. O. Schoenberg, "Measurement of signal-to-noise ratios in mr images: influence of multichannel coils, parallel imaging, and reconstruction filters," *Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine*, vol. 26, no. 2, pp. 375–385, 2007.

H. Kim, G. Papamakarios, and A. Mnih, "The lipschitz constant of self-attention," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5562–5571.

S. Agrawal, Z. Wang, and Y. Ye, "A dynamic near-optimal algorithm for online linear programming," *Operations Research*, vol. 62, no. 4, pp. 876–890, 2014.

X. Li, C. Sun, and Y. Ye, "Simple and fast algorithm for binary integer and online linear programming," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9412–9421, 2020.

S. Markidis, S. W. Der Chien, E. Laure, I. B. Peng, and J. S. Vetter, "Nvidia tensor core programmability, performance & precision," in *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*. IEEE, 2018, pp. 522–531.

N. Megiddo and D. S. Modha, "Outperforming lru with an adaptive replacement cache algorithm," *Computer*, vol. 37, no. 4, pp. 58–65, 2004.

M. Chrobak and J. Noga, "Lru is better than fifo," *Algorithmica*, vol. 23, no. 2, pp. 180–185, 1999.

Q. Zhu, X. Ye, Y. Liu, H. Ouyang, and C. Song, "Probe: Co-balancing computation and communication in moe inference via real-time predictive prefetching," 2026. [Online]. Available: https://arxiv.org/abs/2602.00509

Y. Xie, Z. Zhang, D. Zhou, C. Xie, Z. Song, X. Liu, Y. Wang, X. Lin, and A. Xu, "Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router," *arXiv preprint arXiv:2410.12013*, 2024.

J. Lee, A. Qiao, D. F. Campos, Z. Yao, Y. He *et al.*, "Stun: Structured-then-unstructured pruning for scalable moe pruning," *arXiv preprint arXiv:2409.06211*, 2024.

E. Frantar and D. Alistarh, "Qmoe: Practical sub-1-bit compression of trillion-parameter models," *arXiv preprint arXiv:2310.16795*, 2023.

H. Imani, A. Amirany, and T. El-Ghazawi, "Mixture of experts with mixture of precisions for tuning quality of service," in *2024 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 2024, pp. 1–6.

F. C. Salinas, K. Kumatani, R. Gmyr, L. Liu, and Y. Shi, "Knowledge distillation for mixture of experts models in speech recognition," Technical Report MSR-TR-2022-6, Microsoft Research, May 2022. https://www . . . , Tech. Rep., 2022.

F. Shu, Y. Liao, L. Zhuo, C. Xu, L. Zhang, G. Zhang, H. Shi, L. Chen, T. Zhong, W. He *et al.*, "Llava-mod: Making llava tiny via moe knowledge distillation," *arXiv preprint arXiv:2408.15881*, 2024.

C. Yang, Y. Sui, J. Xiao, L. Huang, Y. Gong, Y. Duan, W. Jia, M. Yin, Y. Cheng, and B. Yuan, "Moe-i ^{2}: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition," *arXiv preprint arXiv:2411.01016*, 2024.

P. Li, Z. Zhang, P. Yadav, Y.-L. Sung, Y. Cheng, M. Bansal, and T. Chen, "Merge, then compress: Demystify efficient smoe with hints from its routing policy," *arXiv preprint arXiv:2310.01334*, 2023.

S. Cao, S. Liu, T. Griggs, P. Schafhalter, X. Liu, Y. Sheng, J. E. Gonzalez, M. Zaharia, and I. Stoica, "Moe-lightning: High-throughput moe inference on memory-constrained gpus," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2025, pp. 715–730.

X. Zhuge, X. Shen, Z. Wang, F. Dang, X. Ding, D. Li, Y. Han, T. Hao, and Z. Yang, "Specoffload: Unlocking latent gpu capacity for llm inference on resource-constrained devices," *arXiv preprint arXiv:2505.10259*, 2025.

Z. Fang, Y. Huang, Z. Hong, Y. Lyu, W. Chen, Y. Yu, F. Yu, and Z. Zheng, "Klotski: Efficient mixture-of-expert inference via expert-aware multi-batch pipeline," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2025, pp. 574–588.

T. Xu, L. Xue, Z. Lu, A. Jackson, and L. Mai, "Moe-gen: High-throughput moe inference on a single gpu with module-based batching," *arXiv preprint arXiv:2503.09716*, 2025.

X. He, S. Zhang, Y. Wang, H. Yin, Z. Zeng, S. Shi, Z. Tang, X. Chu, I. Tsang, and O. Y. Soon, "Expertflow: Optimized expert activation and token allocation for efficient mixture-of-experts inference," *arXiv preprint arXiv:2410.17954*, 2024.

L. Wang, Y. Du, Y. Pan, S. C. Liew, J. Liu, and K. Chen, "Od-moe: On-demand expert loading for cacheless edge-distributed moe inference," 2025. [Online]. Available: https://arxiv.org/abs/2512.03927

S. Zhong, L. Liang, Y. Wang, R. Wang, R. Huang, and M. Li, "Adapmoe: Adaptive sensitivity-based expert gating and management for efficient moe inference," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024, pp. 1–9.

N. Gavhane, A. Mehrotra, R. Chawla, and P. Proenca, "Moe-beyond: Learning-based expert activation prediction on edge devices," 2025. [Online]. Available: https://arxiv.org/abs/2508.17137

P. Tang, J. Liu, X. Hou, Y. Pu, J. Wang, P.-A. Heng, C. Li, and M. Guo, "Hobbit: A mixed precision expert offloading system for fast moe inference," *arXiv preprint arXiv:2411.01433*, 2024.

Y. Xi, H. Wang, B. Chen, J. Lin, M. Zhu, W. Liu, R. Tang, Z. Wei, W. Zhang, and Y. Yu, "Efficiency unleashed: Inference acceleration for llm-based recommender systems with speculative decoding," in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025, pp. 1891–1901.

Z. Qin, R. Goel, M. Gagrani, R. Garrepalli, M. Lee, and Y. Sun, "Confu: Contemplate the future for better speculative sampling," *arXiv preprint arXiv:2603.08899*, 2026.

Y. Zhang, "A markov categorical framework for language modeling," *arXiv preprint arXiv:2507.19247*, 2025.

X. Liu, C. Daniel, L. Hu, W. Kwon, Z. Li, X. Mo, A. Cheung, Z. Deng, I. Stoica, and H. Zhang, "Optimizing speculative decoding for serving large language models using goodput," *arXiv preprint arXiv:2406.14066*, 2024.

Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, and D. Chen, "Snapkv: Llm knows what you are looking for before generation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 22 947–22 970, 2024.

X. Miao, G. Oliaro, Z. Zhang, X. Cheng, Z. Wang, R. Y. Y. Wong, Z. Chen, D. Arfeen, R. Abhyankar, and Z. Jia, "Specinfer: Accelerating generative llm serving with speculative inference and token tree verification," *arXiv preprint arXiv:2305.09781*, vol. 1, no. 2, p. 4, 2023.

T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, and T. Dao, "Medusa: Simple llm inference acceleration framework with multiple decoding heads," *arXiv preprint arXiv:2401.10774*, 2024.

D. Park and B. Egger, "Improving throughput-oriented llm inference with cpu computations," in *Proceedings of the 2024 International Conference on Parallel Architectures and Compilation Techniques*, 2024, pp. 233–245.

Y. Song, Z. Mi, H. Xie, and H. Chen, "Powerinfer: Fast large language model serving with a consumer-grade gpu," in *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, 2024, pp. 590–606.

S. Li, H. Lu, T. Wu, M. Yu, Q. Weng, X. Chen, Y. Shan, B. Yuan, and W. Wang, "Caraserve: Cpu-assisted and rank-aware lora serving for generative llm inference," *arXiv preprint arXiv:2401.11240*, 2024.

A. Saxena, P.-A. Tsai, H. Taneja, A. Jaleel, and M. Qureshi, "Utility-driven speculative decoding for mixture-of-experts," *arXiv preprint arXiv:2506.20675*, 2025.

Z. Wang, Z. Zhang, Y. Zhou, Z. Wang, M. Zhou, P. Jiang, W. Cai, C. Huan, R. Gu, S. Zhong *et al.*, "Accelerating mixture-of-experts inference by hiding offloading latency with speculative decoding," *arXiv preprint arXiv:2508.21706*, 2025.

J. Bang, E. Cho, R. Hwang, J. Chung, and M. Rhu, "Specmoe: A fast and efficient mixture-of-experts inference via self-assisted speculative decoding," *arXiv preprint arXiv:2604.10152*, 2026.

B. McDanel, S. Li, S. Surineni, and H. Khaitan, "Moe-spec: Expert budgeting for efficient speculative decoding," *arXiv preprint arXiv:2602.16052*, 2026.

R. Qin, Z. Li, W. He, J. Cui, H. Tang, F. Ren, T. Ma, S. Cai, Y. Zhang, M. Zhang *et al.*, "Mooncake: A kvcache-centric disaggregated architecture for llm serving," *ACM Transactions on Storage*, 2024.

P. Jaillet, J. Jiang, K. Mellou, M. Molinaro, C. Podimata, and Z. Zhou, "Online scheduling for llm inference with kv cache constraints," *arXiv preprint arXiv:2502.07115*, 2025.

Appendix Contents

A Theoretical Analysis on Advantages of SD for MoE Inference **16**
A.1 Expert Reuse . . . 16
A.2 Information Gain . . . 18
A.3 Fault Tolerance . . . 19

B Design Rationale for Speculative Utility Estimation **19**
B.1 Empirical Observation . . . 19
B.2 Theoretical Justification. . . . 20

C Optimization Formulation & Solution for Heterogeneous Workload Balancer **21**
C.1 Formulation Explanation . . . 21
C.2 Solution Derivation . . . 22

D Implementation Details of Unified Asynchronous Execution Engine **22**

E Baseline Information **24**

F Illustration of Inference Pipelines **24**

G Ablation Explanation **24**

H Implementation Validation **26**
H.1 Numerical Compatibility . . . 26
H.2 Additional Runtime Observations . . . 26

I Related Works **26**
I.1 Efficient MoE on Edge . . . 26
I.2 Speculative Decoding (SD) . . . 27
I.3 Heterogeneous Computing . . . 27
I.4 Concurrent Works . . . 27

J Future Works **27**

K Limitations **28**

L Broader Impacts **28**

A Theoretical Analysis on Advantages of SD for MoE Inference

We analyze the theoretical advantages of speculative decoding (SD). We demonstrate that beyond simple speedup, SD fundamentally alters the nature of the online heterogeneous expert scheduling problem by providing *Expert Reuse*, *Information Gain*, and *Fault Tolerance*.

A.1 Expert Reuse

We first address the rationale for allocating scarce edge memory to a draft model rather than caching more experts. Figure 7 illustrates the Tokens Per Second (TPS) performance of `Qwen3` models w.r.t. the percentage of total expert weights resident in GPU memory. The result reveals that marginally increasing the resident expert set yields diminishing returns in throughput for memory-bound regimes. Conversely, reserving a small partition of VRAM for a draft model (8% compared to all expert weights) enables SD, which unlocks substantial parallelism and expert reuse.

Although some works suggest SD is inefficient for MoE due to dynamic expert activation across different draft tokens (Huang et al., 2025; Li et al., 2024a), we show that when γ (draft length) is tuned correctly, the *expert reuse* across the speculation window amortizes the loading cost. Aside from the original improvement in wall time, we now illustrate the scenario where the expert reuse in resource-constrained MoE architecture can achieve speedup.

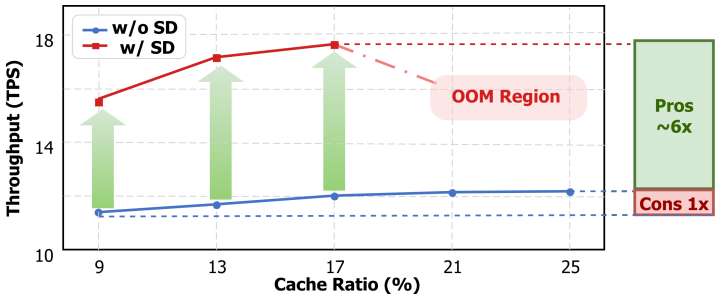


Figure 7: Illustration of TPS metric to the expert cache ratio(%) on NVIDIA GeForce RTX 4090 (24GB). Target model is `Qwen3-30B-A3B`, and draft one is `Qwen3-4B-FP8`. In edge scenario, less cache ratio precipitates only negligible decelerate, shown as red block with "Cons" label. Placing a small draft model can lead to huge speedup, shown as the green block with "Pros" label. Pros is about $6\times$ overweight Cons.

Formally, following the notions in Section 2.1, T_V is dominated by FFN calculation, which is presented as

$$T = (k \cdot N) \cdot (T_{LOAD} + T_{FFN}) \cdot L, \tag{13}$$

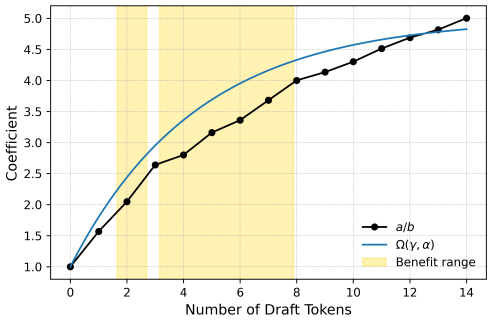
where k is the coefficient of the proportion of the activated experts, N is the total number of experts needed for γ tokens, T_{LOAD} is the time for loading one expert, T_{FFN} is the time for calculating through one expert, and L is the the number of layers.

Because of the *expert reuse*, ks for SD and AR are hard to describe with unified variable (not linear), thus denoting them as a and b, respectively. Therefore, we have

$$\begin{aligned} T_{SD} &= (a \cdot N) \cdot (T_{LOAD} + T_{FFN}) \cdot L + T_D \cdot \gamma \\ T_{AR} &= (b \cdot N) \cdot (T_{LOAD} + T_{FFN}) \cdot L \cdot \gamma \end{aligned} \tag{14}$$

γ	$\frac{a}{b}$	γ	$\frac{a}{b}$
1	1.57	8	4.00
2	2.05	9	4.13
3	2.64	10	4.30
4	2.80	11	4.51
5	3.16	12	4.69
6	3.36	13	4.82
7	3.68	14	5.00

(a) Table on experimental data points.



(b) Experimental vs theoretical curve.

Figure 8: The curve comparison between $\Omega(\gamma, \alpha)$ and $\frac{a}{b}$, experimented on MT-bench with `Qwen3-235B-A22B` as verification model and `Qwen3-4B-FP8`, taking α as 0.8. (a) shows the numerical result, while (b) illustrates the benefit area.

For $N \cdot (T_{LOAD} + T_{FFN}) \cdot L$, which is constant, let us denote Z. Then the Time Per Output Token (TPOT) for both SD and AR is:

$$\begin{aligned} TPOT_{SD} &= \frac{Z \cdot a + \gamma \cdot T_D}{\Omega(\gamma, \alpha)} \\ TPOT_{AR} &= \frac{Z \cdot b \cdot \gamma}{\gamma} = Z \cdot b \end{aligned} \tag{15}$$

We next describe under which scenario, $TPOT_{SD} < TPOT_{AR}$ will be established, i.e.

$$a + \gamma \cdot \frac{T_D}{Z} < b \cdot \Omega(\gamma, \alpha) \tag{16}$$

Since $\gamma \cdot \frac{T_D}{Z}$ is relatively small, it suffices to guaranteed:

$$\frac{a}{b} < \Omega(\gamma, \alpha) \tag{17}$$

As Figure 8 illustrates for `Qwen3` series, the advantage becomes more *pronounced* when $\gamma \in 2 \cup [3, 8]$. Thus, expert reuse benefits inference speed in theory and provide a guideline for setting the hyper-parameters.

Notice that $\Omega(\gamma, \alpha)$ has the limit:

$$\lim_{\gamma \to +\infty} \Omega(\gamma, \alpha) = \frac{1-0}{1-\alpha} = \frac{1}{1-\alpha} \tag{18}$$

which bounds the value of $\frac{a}{b}$ if an ideal speedup is desired.

Therefore, acceptance rate α is closely related to the alignment between draft model and target model. A series of models which share the same pretraining data are ideal for off-shelf model speculation usage.

In the end-to-end experiments, we report the key metric α and TPS performance across different datasets, using Equation 18 as a bridge to demonstrate their positive correlation. Details of the experimental settings are provided in Section 4.1.

Metric	CNN/DM	MT-Bench	GSM8K	HumanEval
Acceptance Rate (α)	0.56	0.68	0.83	0.85
Ω (Token yield) - a/b (Expert cost)	0.23	0.32	0.64	0.76
MoE-SpAc TPS	**22.78**	**24.10**	**27.82**	**28.24**

A.2 Information Gain

In Section 2.3, we illustrate how SD transforms the expert activation signals from binary indicators to frequency values. Here, we quantitatively demonstrate the information-theoretic advantage of this transformation.

Let $E_i(\cdot)$ be an expert with activation probability p_i. In standard AR inference, the expert activation for a single step is a random variable following a Bernoulli distribution: $X_i^{(1)} \sim \text{Bern}(p_i)$. In SD with a draft length γ, the activation generalizes to the aggregated count over the verification window, $X_i^{(\gamma+1)} = \sum_{t=1}^{\gamma+1} X_{i,t}^{(1)}$. By assuming weak inter-token correlation, this follows a Binomial distribution: $X_i^{(\gamma+1)} \sim \text{Bin}(\gamma + 1, p_i)$.

Entropy Dominance. The Shannon entropy of the aggregated signal $X_i^{(\gamma+1)}$ strictly dominates that of $X_i^{(1)}$:

$$H(X_i^{(\gamma+1)}) = -\sum_{k=0}^{\gamma+1} p_k^{(\gamma+1)} \log p_k^{(\gamma+1)} \geq H(X_i^{(1)}) \tag{19}$$

where $p_k^{(\gamma+1)} = \Pr(X_i^{(\gamma+1)} = k)$. This implies that $X_i^{(\gamma+1)}$ carries strictly more bits of information regarding expert demand, reducing uncertainty for utility estimation.

Signal-to-Noise Ratio (SNR). The stability of the offloading signal is critical. Following Dietrich et al. (2007), we define SNR as the ratio of the mean demand to its standard deviation (μ/σ). Then, we show that the SNR of expert activation signals brought by SD is $\sqrt{\gamma+1}$ times more robust compared with that of standard AR decoding:

$$\begin{aligned} \text{SNR}_{AR} &= \frac{p_i}{\sqrt{p_i(1-p_i)}}, \\ \text{SNR}_{SD} &= \frac{(\gamma+1)p_i}{\sqrt{(\gamma+1)p_i(1-p_i)}} = \sqrt{\gamma+1} \cdot \text{SNR}_{AR} \end{aligned} \tag{20}$$

The relative noise in the SD signal decreases by a factor of $\sqrt{\gamma+1}$. This mathematical stabilization allows for reliable expert activation scoring, whereas AR is dominated by stochastic noise, especially when $p_i \ll 1$.

A.3 Fault Tolerance

Finally, we demonstrate that the transition from AR to SD relaxes the precision requirements of utility estimation with larger safety margins, granting the system fault tolerance.

Scheduling Fault. Recall from Section 2.3 that the scheduling decision depends on whether the estimated utility $\hat{s}_{i,t+1}$ exceeds the threshold τ_t. We define a *scheduling fault* $\mathcal{E}_i$ as a discrepancy between the ground-truth optimal decision and the system's derived decision:

$$\mathcal{E}_i = \mathbb{I}\left(\mathbb{I}(s_{i,t+1} \geq \tau_t) \neq \mathbb{I}(\hat{s}_{i,t+1} \geq \tau_t)\right), \tag{21}$$

Specifically, a fault corresponds to either a cache miss (False Negative) or memory waste (False Positive).

Safety Margin Analysis. We analyze the fault tolerance in frequency domain by defining τ_t^* as the frequency threshold, such that $G(f_{i,t+1}) \geq \tau_t \iff f_{i,t+1} \geq \tau_t^*$. Hence, a scheduling fault is avoided if the estimation error does not exceed the distance between the ground truth and the threshold, which is defined as the *safety margin* $\Delta_{i,t}$:

$$|\hat{f}_{i,t+1} - f_{i,t+1}| \leq \underbrace{|f_{i,t+1} - \tau_t^*|}_{\text{Safety Margin } \Delta_{i,t}} \Rightarrow \mathcal{E}_i = 0, \tag{22}$$

where $\hat{f}_{i,t+1}$ is the estimated frequency implied by the score.

AR Rigidity. In AR with $f_{i,t+1} \in \{0, 1\}$, the threshold typically lies in the interval $\tau_t^* \in (0, 1)$. Consequently, the safety margin is strictly bounded by $\Delta_{i,t} < 1$. Since frequency counts are integers, any prediction error implies $|\hat{f}_{i,t+1} - f_{i,t+1}| \geq 1 > \Delta_{i,t}$, inevitably causing a scheduling fault (e.g., predicting 0 when truth is 1).

SD Tolerance. In SD with $f_{i,t+1} \in [0, \gamma + 1]$, the safety margin $\Delta_{i,t}$ can be larger than 1, especially for extremely hot or cold experts. For example, with $\gamma = 5$ and $\tau_t^* = 2$, an expert with demand $f_{i,t+1} = 5$ has a margin $\Delta_{i,t} = 3$. The system is tolerant to under-estimate the frequency by 1 or 2 units (noise) without crossing the boundary τ_t^*, still maintaining a correct scheduling decision ($\mathcal{E}_i = 0$).

B Design Rationale for Speculative Utility Estimation

In this section, we provide the analysis from the perspective of empirical observation and theoretical justification.

B.1 Empirical Observation

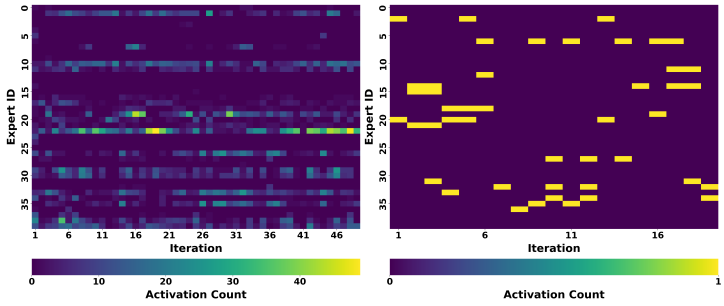


Figure 9: Activation frequency across iterations in SD is shown on the left, while the AR is shown on the right. It is much easier to estimate the light points where the experts are activated multiple times(hot experts).

We construct a sequence of activation frequency of a same expert along the steps, e.g. $f_{1,0}, f_{1,2}, \cdots, f_{1,t}$ for expert No.1. This sequences of different experts in speculative inference are depicted in Figure 9, with compared to those using AR. Empirically, these activation sequences in SD are more informative than AR and exhibit tighter correlation across steps. The frequency evolves gradually rather than fluctuate abruptly.

Frequency stability across workloads. For consecutive expert frequencies, define sMAPE as $2|f_t - f_{t-1}|/(|f_t| + |f_{t-1}|)$, assigning zero when both vanish. This empirical variation measure associates lower values with more stable frequencies. The measured sMAPE/accuracy pairs are $(0.889, 0.770)$, $(0.684, 0.801)$, $(0.490, 0.873)$, and $(0.322, 0.897)$, supporting the use of temporal stability in utility estimation. Across Qwen3-30B, DeepSeek-V4-Flash-284B, and GLM-4.7-358B, sMAPE is 0.490, 0.454, and 0.164, respectively, indicating that temporal locality also occurs in larger sparse models.

B.2 Theoretical Justification.

The validity of the Inertial Utility Transition relies on the premise that the underlying routing signal is smooth; otherwise, an inertial tracker would suffer from excessive lag. We formalize this smoothness via the bounded drift of expert gating scores.

Theorem 1. *(Bounded Drift of Expert Gating Scores) Let $\mathbf{h}_t^{(0)}$ and $\mathbf{h}_{t+1}^{(0)}$ be the input embeddings for two consecutive inference steps with bounded initial divergence $\delta_{in} = ||\mathbf{h}_t^{(0)} - \mathbf{h}_{t+1}^{(0)}||$. Assume the Attention and FFN modules in the routed model are α^l-Lipschitz and β^l-Lipschitz respectively (Kim et al., 2021). Let W_r^l be the routing weight matrix at layer l with spectral norm C_R^l. The divergence in expert gating scores $\mathbf{s}^l$ (before Top-k selection) is bounded by:*

$$||\mathbf{s}_t^l - \mathbf{s}_{t+1}^l|| \leq C_R^l \cdot \delta_{in} \prod_{j=0}^{l-1}(1 + \sigma^j), \tag{23}$$

where $\sigma^j = \beta^j(1 + \alpha^j)$ represents the layer-wise expansion factor.

Preliminaries. Let the hidden state update rule in the l-th Transformer layer be:

$$\mathbf{h}^{l+1} = \mathbf{h}^l + \text{FFN}^l(\mathbf{h}^l + \text{Attn}^l(\mathbf{h}^l)). \tag{24}$$

We assume the Lipschitz conditions for the sub-modules:

$$||\text{Attn}^l(\mathbf{h}) - \text{Attn}^l(\mathbf{h}')|| \leq \alpha^l||\mathbf{h} - \mathbf{h}'||, \tag{25}$$

$$||\text{FFN}^l(\mathbf{h}) - \text{FFN}^l(\mathbf{h}')|| \leq \beta^l||\mathbf{h} - \mathbf{h}'||. \tag{26}$$

Step 1: Layer-wise Divergence Propagation. Consider two input states $\mathbf{h}_t^l$ and $\mathbf{h}_{t+1}^l$ entering layer l. Let $\Delta^l = ||\mathbf{h}_t^l - \mathbf{h}_{t+1}^l||$. Subtracting the update equations Eq. (24) for both states:

$$||\mathbf{h}_t^{l+1} - \mathbf{h}_{t+1}^{l+1}|| \leq ||\mathbf{h}_t^l - \mathbf{h}_{t+1}^l|| + \beta^l \left(||\mathbf{h}_t^l - \mathbf{h}_{t+1}^l|| + ||\text{Attn}^l(\mathbf{h}_t^l) - \text{Attn}^l(\mathbf{h}_{t+1}^l)|| \right) \tag{27}$$

$$\leq \Delta^l + \beta^l(\Delta^l + \alpha^l\Delta^l) \tag{28}$$

$$= (1 + \beta^l + \alpha^l\beta^l)\Delta^l. \tag{29}$$

Let $\sigma^l \triangleq \beta^l(1 + \alpha^l)$. The recurrence relation becomes:

$$\Delta^{l+1} \leq (1 + \sigma^l)\Delta^l. \tag{30}$$

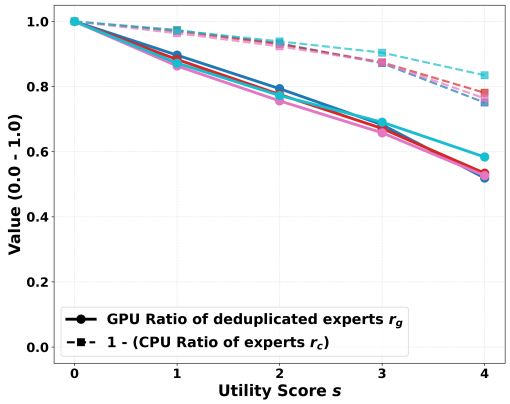
Step 2: Unrolling the Recursion. Recursively applying the inequality from layer 0 (input embedding) to layer l:

$$\Delta^l \leq \Delta^0 \prod_{j=0}^{l-1}(1 + \sigma^j), \tag{31}$$

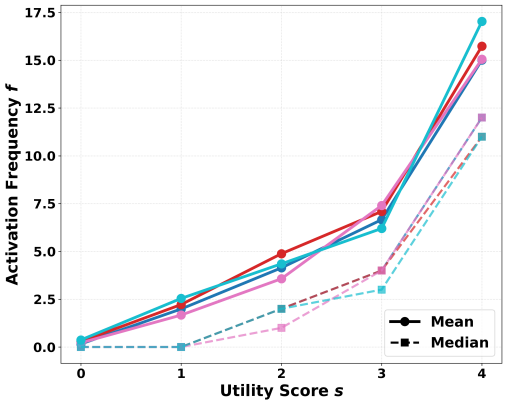
where $\Delta^0 = \delta_{in}$ is the initial divergence between the contexts.

Step 3: MoE Router Projection. In an MoE layer, the gating scores $\mathbf{s}^l \in \mathbb{R}^E$ (before softmax and Top-K) are computed via a linear projection W_r^l:

$$\mathbf{s}^l = W_r^l\mathbf{h}^l. \tag{32}$$



(a) Utility score to the ratio of γk activated experts that can be processed on GPU, $1 - r_c$ (Dotted line) and the ratio of b de-duplicated activated experts to be processed on GPU in parallel. r_g (Solid line) for a single layer.



(b) Utility Score to the average value (Solid line) and median (Dotted line) of activation frequency. Experiment conducted on MT-Bench with $K = 4$ on `Qwen` models.

Figure 10: Relationship between utility and activated frequency, ratio of activated experts, etc..

The magnitude of the difference in gating scores is bounded by the spectral norm of the weight matrix $C_R^l = ||W_r^l||_2$:

$$||\mathbf{s}_t^l - \mathbf{s}_{t+1}^l|| = ||W_r^l(\mathbf{h}_t^l - \mathbf{h}_{t+1}^l)|| \tag{33}$$

$$\leq ||W_r^l|| \cdot ||\mathbf{h}_t^l - \mathbf{h}_{t+1}^l|| \tag{34}$$

$$= C_R^l \Delta^l. \tag{35}$$

Conclusion. Substituting the unrolled bound for Δ^l, we obtain the final inequality:

$$||\mathbf{s}_t^l - \mathbf{s}_{t+1}^l|| \leq C_R^l \cdot \delta_{in} \prod_{j=0}^{l-1}(1 + \sigma^j). \tag{36}$$

This confirms that the expert routing scores exist within a bounded region around the previous step's scores, scaled by the depth of the network and the Lipschitz constraints of the layers. □

Eq. 36 bounds routing-score drift, not changes in the discrete Top-k set. If $\|\Delta\mathbf{s}\|_\infty \leq \epsilon$, Top-$k$ membership is preserved only when the gap between the k-th and $(k+1)$-th scores exceeds 2ϵ. Near ties, arbitrarily small perturbations can change membership. Multi-token aggregation can reduce sensitivity to individual flips empirically, but does not guarantee stable frequencies or error-free prediction. The inertial estimator is therefore motivated by the observed temporal correlation; CPU fallback handles prediction errors.

C Optimization Formulation & Solution for Heterogeneous Workload Balancer

C.1 Formulation Explanation

To illustrate Equation 7 and Equation 8, we first look at the two functions related to τ. As depicted in Figure. 10a, the ratio of γk activated experts processed on CPU $r_c(\tau)$ and the ratio of de-duplicated activated expert processed on GPU $r_g(\tau)$ remain stable in value for a given threshold k within a single layer. Therefore, this value can be learned from a warm up test.

To illustrate their relations with T_{CPU} and T_{GPU}, say an inference step that activates two experts per token with two draft tokens ($\gamma = 2$). If a layer forward pass prefetches expert No. 3 based on threshold τ, while the draft tokens activate experts {No. 0, No. 3} and {No. 2, No. 3}. Then $r_c(k) \approx 1 - \frac{2}{4}$ and $r_g(\tau) \approx 1/3$. The host calculation time would be the sum of computations for

missed experts 0 and 2, taking time $r_c(\tau) \times 4 \times T^u_{cpu}$. The device processes the stacked hidden states of the identified experts concurrently, which only takes $r_g(\tau) \times 3 \times T^u_{gpu}$.

Besides, Eq. 10 imposes a strict constraint to prevent I/O bottlenecks. Specifically, the prefetching budget for each layer is bounded by the aggregate duration of computation and the corresponding drafting phase. By prioritizing the placement of hot experts on the limited device memory, the optimization forces the thresholding policy to align with the inter-device transmission bandwidth. Equation 12 enforces a strictly positive threshold ($\tau > 0$) to differentiate our predictive approach from non-selective prefetching strategies. Regarding Eq. 11, minimizing the overhead is critical; querying the remaining memory $VRAM_{left}$ via `cudaMemGetInfo` entails non-negligible latency in CPU. Consequently, we can formulate an online linear programming problem to prioritize matching other constraints first, incorporating the memory constraint with latency-aware adjustments (Agrawal et al., 2014; Li et al., 2020).

Furthermore, Figure 10b empirically demonstrates that a higher score in utility estimator correlates with a higher frequency score in computation, thereby better utilizing the high-dimensional computational capacity of the GPU Markidis et al. (2018).

C.2 Solution Derivation

Since the domain of τ is restricted to integers $\tau \in [1, K]$, we determine the optimal τ^* by exploiting the structural properties of the objective function. First, we define the feasible set $\mathcal{S}$ constrained by physical resources. Let the available computation window be $C_{window} = T_{total} + \gamma \cdot T^u_{draft}/L_{target}$. Based on the I/O (Eq. 10) and Memory (Eq. 11) constraints, the feasible set is determined by the range of prefetching volume $n(\tau)$:

$$\mathcal{S} = \{\tau \in \mathbb{Z} \mid \tau_{min} \leq \tau \leq \tau_{max}\} \subseteq [1, K] \tag{37}$$

where the bounds τ_{min} and τ_{max} are derived from the limits on $n(\tau)$ imposed by C_{window} and VRAM_{left}. The objective function is defined as the absolute difference between the device execution times:

$$f(\tau) = \Big| \underbrace{r_c(\tau) \cdot \gamma \cdot k \cdot T^u_{cpu}}_{T_{cpu}(\tau)} - \underbrace{r_g(\tau) \cdot b \cdot T^u_{gpu}}_{T_{gpu}(\tau)} \Big| \tag{38}$$

Since a higher threshold τ reduces the number of hot experts, the GPU allocation ratio $r_g(\tau)$ is monotonically decreasing, causing $T_{gpu}(\tau)$ to decrease. Conversely, the CPU allocation ratio $r_c(\tau)$ is monotonically increasing, causing $T_{cpu}(\tau)$ to increase. Consequently, the difference $T_{cpu}(\tau) - T_{gpu}(\tau)$ is nondecreasing, so its absolute value is unimodal, but need not be convex. A constant-time analytical crossing requires an affine time model, as in the solver microbenchmark; it is not implied by monotonicity alone. With such a model and a contiguous feasible interval, only a constant number of candidates need evaluation. The optimal solution τ^* corresponds to either the integer closest to the theoretical intersection point τ_{cross} where $T_{cpu}(\tau_{cross}) = T_{gpu}(\tau_{cross})$, or one of the boundary points of $\mathcal{S}$ if the intersection lies outside the feasible range:

$$\tau^* = \underset{\tau \in \{\tau_{min}, \tau_{max}, \lfloor \tau_{cross} \rfloor, \lceil \tau_{cross} \rceil\} \cap \mathcal{S}}{\arg\min} f(\tau) \tag{39}$$

The candidate rule is exact under the stated model and feasibility assumptions. For arbitrary measured ratios, exhaustive evaluation of the K thresholds provides an $O(K)$ reference; a monotone crossing can be located in $O(\log K)$ when feasibility is an interval. If the feasible set is empty, prefetching must be disabled and missing experts executed on CPU; choosing $\tau = K$ alone does not ensure feasibility. These complexity bounds concern threshold selection, excluding utility updates and coefficient estimation.

D Implementation Details of Unified Asynchronous Execution Engine

Utility-Guided Prefetching. To manage the prefetching requests, we implement a *multi-level priority queue*, where each level contains pending requests for experts with the same utility $u_{i,t}$. Within each level, requests follow a First-In-First-Out (FIFO) order to respect the temporal sequence of layer execution for target model. The prefetcher continuously scans the queues at different levels in descending order of utility (from K down to τ). The execution logic is filtered by the dynamic

threshold τ_t: the prefetcher only processes queues with levels $u_{i,t} \geq \tau_t$, transferring weights in descending order of priority (from level K down to τ).

Utility-Guided Eviction. Complementarily, we manage the resident GPU cache using a *state-ordered pool*, implemented via a Red-Black tree. This structure indexes all GPU-resident experts by their current utility score $u_{i,t}$. When the load balancer updates the threshold to τ_t, the evictor can efficiently identify and remove resident experts with decreased scores $u_{i,t} < \tau_t$ in $O(\log N)$ time.

Crucially, unlike traditional policies such as Adaptive Replacement Cache Megiddo and Modha (2004) or Least Recently Used (LRU) Chrobak and Noga (1999) which rely on historical access timestamps, our approach employs a **unified metric** (expert utility) for both prefetching and eviction. This unification ensures scheduling consistency, effectively preventing the cache thrashing of marginally hot experts.

When it comes to the management of VRAM cache and score-based prefetching method, Figure 11 shows the details. Each expert ID is pushed into the queue labeled with corresponding score. With the online decision of threshold, the queue with lower lable is filtered out. Then the experts are prefetched according to the ID reported in the queue from high to low. When prefetching is finished, each expert weight is tagged with a score and recorded in a `set`. When the expert is in used, the score is modified to $K + 1$, which we call a frozen expert, because under no circumstance should it be evicted. Due to the limited resources, most experts should be evicted right after calculation, and leave the space for layers to come. Therefore, we change these scores to 0, thus quickly recycled and refilled by other weights. All the experts weight is cache is recorded dynamically with a `set`, reaching an ideal efficiency with fine-grained lock usage.

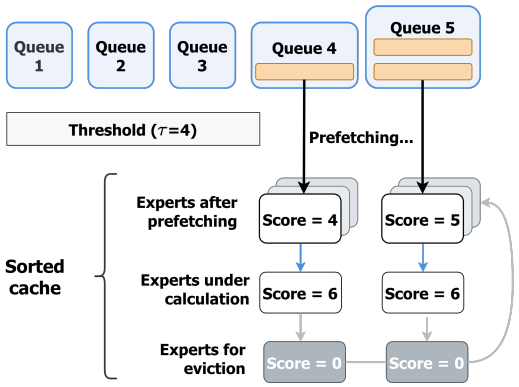


Figure 11: Scheduling and cache management of the MoE-SpAc in implementation when $K = 5$.

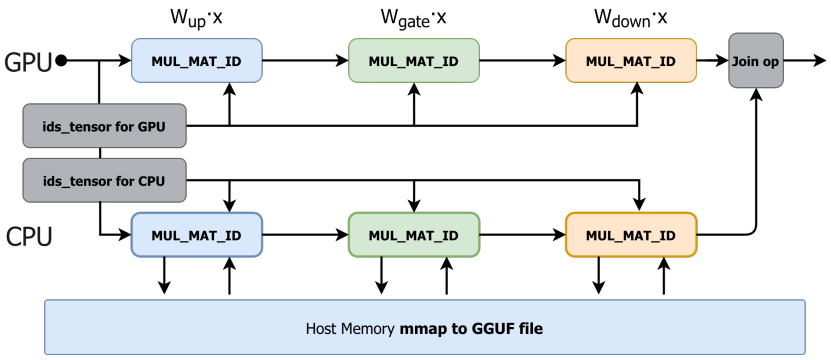


Figure 12: MoE-SpAc implementation atop `llama.cpp`, unchanged operators is omitted.

We implement the management and scheduling system asynchronously atop `llama.cpp`, which native support CPU specific or GPU specific expert calculation. As shown in Figure 12, we hack a `ids_tensor` modification in backend, rewrite the expert calculation kernel: skip the calculation of expert with special label in `ids_tensor`. This modification is applied to each process in expert

calculation in one layer forward, including $W_{up} \cdot x$, $W_{gate} \cdot x$, and $W_{down} \cdot x$. With a CUDA kernel joining up the result from the heterogeneous devices, we implement this process neatly without breaking the computation graph.

E Baseline Information

We compare MoE-SpAc against representative inference systems covering dense and MoE models, as well as different offloading and prediction strategies as follows:

- `Transformers Accelerate` (Wolf et al., 2019) supports layer-wise parameter offloading via device maps.
- `vLLM` (Kwon et al., 2023) is a high-throughput inference engine; our comparison enables CPU weight offloading with `-cpu-offload-gb 60`.
- `llama.cpp` (Ollama, 2025) is a lightweight inference engine optimized for constrained GPU environments.
- `llama.cpp-w/ SD` extends llama.cpp with speculative decoding.
- `Mixtral Offloading` (Eliseev and Mazur, 2023) offloads expert weights between CPU and GPU while maintaining an LRU cache.
- `MoE-Infinity` (Xue et al., 2024) is a cost-effective MoE inference library that predicts expert activation paths and evicts experts based on least visit count.
- `MoE-SpeQ` (Wang et al., 2025a) uses an on-device draft model to predict future expert demand and proactively prefetch weights, with an adaptive governor tuning speculation to the hardware.
- `SpecExp` (Madan et al., 2026), short for Speculating Experts, predicts future experts from internal model representations to overlap CPU–GPU weight transfers with computation; it also explores executing predicted experts directly.
- `SP-MoE` (Chen et al., 2025b) predicts expert activation by directly using a draft model.
- `Fate` (Fang et al., 2025b) predicts activated experts using the gating output of the subsequent layer and proactively offloads expert weights to target devices.
- `HybriMoE` (Zhong et al., 2025) is a hybrid CPU-GPU inference framework that improves resource utilization through a heterogeneous scheduling and cache management system, implemented atop `kTransformers`.

F Illustration of Inference Pipelines

In Figure 13, we compare the pipeline of our proposed MoE-SpAc with other scheduling baselines. We can observe that speculative decoding with heterogeneous computation allows parallel scheduling. With load balancing from Section 3.3, we can achieve much less time elapsed compared with other expert-scheduling baselines.

Layer latency. The layer profile takes 0.73–0.78 ms for most layers with MoE-SpAc, compared with 1.26–1.35 ms for llama.cpp, where CPU computation dominates the tail. Layer 47, including the fused output head, takes 1.51 ms with a CPU computation tail of 43%. These measurements explain the reduction in CPU bottlenecks within the verification pipeline.

Misprediction cost. A false-positive prediction consumes one expert's cache space and may incur an unnecessary weight transfer; its byte cost depends on expert dimensions and precision. A false negative executes the missing expert on CPU and transfers hidden states instead of the expert weights. CPU–GPU overlap absorbs part of this cost, while the remaining CPU work can extend the synchronization tail.

G Ablation Explanation

In this section, we provide a detailed analysis of the ablation results in Section 4.4.

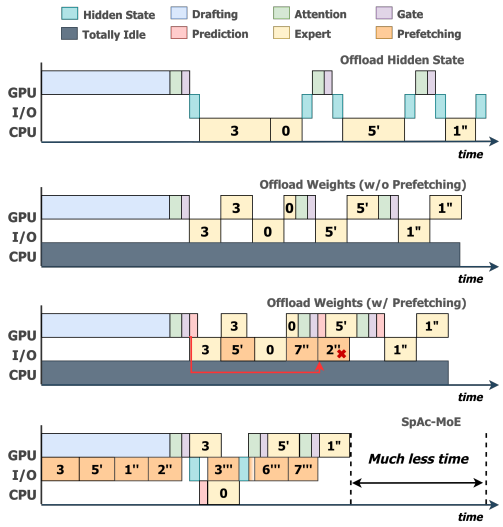


Figure 13: Comparison of MoE inference pipelines under Speculative Decoding (SD). 'GPU' and 'CPU' denote the compute device for expert layers, while prime notation (e.g., $1', 1''$) indicates the layer depth. MoE-SpAc (bottom) minimizes pipeline bubbles via optimized heterogeneous scheduling.

- The speculative utility estimator is the cornerstone of our framework. Removing it (i.e., w/o `SpecUE`) results in a significant drop in both throughput and latency, confirming that the informative frequency-valued signals brought by SD are crucial for expert demand estimation and memory management.

- Within the estimator, the adaptive boundary calibration contributes to robustness. However, its removal (w/o `AdaBC`) yields only a marginal degradation. This suggests that our default parameters ($\theta^{\uparrow} = 3, \theta^{\downarrow} = 1$) are sufficiently general for most expert activation patterns, though adaptation refines performance for corner cases.

- The heterogeneous workload balancer proves critical for I/O management. By fixing the threshold τ, the system fails to adapt to real-time I/O bandwidth constraints, leading to either I/O congestion or under-utilization of the GPU, causing a performance drop. There, we conduct more experiment with fixed τ to demonstrate the advantage of `HetWB`:

Table 5: Performance with adaptive and fixed thresholds.

	Adaptive (Ours)	**Fix** $\tau = 1$	**Fix** $\tau = 2$	**Fix** $\tau = 4$
TPS	**24.10**	21.73	23.48	10.16

- The utility-guided prefetcher and evictor are tightly coupled. Removing one of them (`SG-Evictor` or `SG-Prefetcher`) degrades the performance to the level of the baseline, since I/O is no longer masked. This confirms that the synergy between predictive scoring, dynamic balancing, and asynchronous execution is essential for realizing the full potential of MoE-SpAc.

Cache hits and baseline interpretation. Our scheduler combines proactive GPU placement with CPU execution for cold or missed experts. GPU expert-cache hit rates are 0.71, 0.73, 0.75, and 0.70 on MT-Bench, GSM8K, HumanEval, and CNN/DM, respectively. Cache hit rate differs from hot/cold prediction accuracy: assigning activated experts to CPU can reduce the synchronization tail even when it lowers cache hits. Table 1 includes the additional baseline comparisons; Accelerate covers on-demand weight loading with a simple cache.

H Implementation Validation

H.1 Numerical Compatibility

Exact speculative sampling preserves the target distribution under the original acceptance/rejection rule and a numerically consistent target evaluator. Quantizing the draft changes proposal quality and acceptance. We validate the heterogeneous kernels separately to check numerical consistency between CPU and GPU computation. AGGRE is bit-exact in the tested configurations, and all split-pipeline

Table 6: Numerical validation. E, U, and T denote expert count, experts per token, and token count. Split-pipeline tests cover cache ratios of 0/50/100%. NMSE is normalized mean squared error.

Test	Configuration	Worst NMSE
AGGRE (CUDA vs. CPU)	$E = 256, U = 8, T \in \{1, 32\}$	0 (bit-exact)
Small split MoE	$E = 8, U \in \{2, 8\}, T \in \{1, 7, 32\}$	4.1×10^{-6}
Qwen3-like split MoE	$E = 128, U \in \{2, 8\}, T \in \{1, 7, 32\}$	1.2×10^{-6}
Realistic layer	embedding 2048, FFN 768, $E = 128, U = 8, T = 32$	8.3×10^{-7}

groups pass with worst NMSE at most 4.1×10^{-6}. These tests establish numerical agreement at the kernel and layer levels; exact distributional preservation remains conditional on the sampling rule and target evaluation.

H.2 Additional Runtime Observations

The coefficient-update microbenchmark reports 4.31 ns per layer for the warmup arithmetic; this excludes collecting the profiling measurements. The author-response comparison reports that a single-activation CPU fallback using Intel AMX is within a 50% latency gap of GPU execution, while the AVX2 path is approximately $3\times$ slower. These relative costs depend on the processor and expert configuration. The response estimates approximately 600 MB for a false-positive expert placement in its cost example; the general cache cost is the actual expert weight size W_e^u, which varies across models and precisions.

I Related Works

I.1 Efficient MoE on Edge

MoE has emerged as a crucial architecture for scaling LLMs, prompting extensive research into optimizing its inference efficiency. Zhu et al. (2026b) co-balances computation and communication in real time in distributed system, which overlooks the stress from edge scenarios. Model compression techniques—including pruning (Xie et al., 2024; Lee et al., 2024), quantization (Frantar and Alistarh, 2023; Imani et al., 2024), distillation (Salinas et al., 2022; Shu et al., 2024), and decomposition (Yang et al., 2024; Li et al., 2023b)—have been successfully applied to MoEs. However, similar to their application in dense models, these methods inevitably trade generation quality for acceleration.

In scenarios where precision loss is not allowed and MoEs exceed GPU memory capacity, system-level offloading becomes essential. While latency in large-batch regimes can be amortized via micro-batch pipelining (Cao et al., 2025; Zhuge et al., 2025; Fang et al., 2025c; Xu et al., 2025), low-latency inference for single-batch requests remains challenging. To address this, *on-demand loading* methods, such as Lina (Li et al., 2023a), ExpertFlow (He et al., 2024), and OD-MoE (Wang et al., 2025b), exclusively utilize GPUs for expert computation. If an expert is not prefetched to the GPU, it must be loaded on demand. The downside of these approaches is the high I/O overhead, which causes prolonged GPU idling. *Expert Prefetching* (Xue et al., 2024; Zhong et al., 2024; Fang et al., 2025b; Gavhane et al., 2025) leverages historical activation patterns to preload experts, overlapping I/O with computation. Complementarily, *Expert Caching* (He et al., 2024; Tang et al., 2024; Zhong et al., 2024) exploits the temporal locality of expert activation to retain experts in high-bandwidth memory, mitigating offloading overheads. These research works are largely GPU-intensive and therefore do not fully utilize the computational resources of heterogeneous devices.

I.2 Speculative Decoding (SD)

SD allows for lossless latency reduction by verifying multiple drafted tokens in parallel (Leviathan et al., 2023; Xia et al., 2022; Xi et al., 2025; Qin et al., 2026). Recently, Zhang (2025) provided a precise information-theoretic rationale for SD by quantifying the information surplus in hidden states. However, the efficiency of SD is highly sensitive to the operational intensity. As batch size increases, the system transitions from memory-bound to compute-bound, making the verification of speculated tokens prohibitively expensive (Liu et al., 2024; Li et al., 2024b; Miao et al., 2023; Sun et al., 2024). Conversely, for personal or edge usage where batch sizes are small, inference is strictly memory-bound. In this regime, the additional arithmetic operations required for SD verification are strictly encompassed by the memory retrieval time, making SD an ideal candidate for acceleration (Cai et al., 2024).

I.3 Heterogeneous Computing

Previous offloading techniques have primarily focused on reducing memory transfer overhead by offloading certain computations to the CPU (Park and Egger, 2024). For instance, PowerInfer (Song et al., 2024) reduces GPU memory demand by executing less frequently activated neurons on the CPU, taking advantage of skewed activation patterns. Caraserve (Li et al., 2024c) addresses cold-start delays in LoRA serving by utilizing CPU assistance and employing rank-aware scheduling to reduce latency. However, they do not ensure skewed activations of neurons or parameter reuse. In the context of MoE models, techniques like Fiddler (Kamahori et al., 2024) and kTransformers (Chen et al., 2025a) extend this concept by offloading expert layer computation to the CPU during cache misses. Specifically, when an expert is not in the GPU cache, the CPU executes the corresponding expert layer instead of loading it from memory. These approaches aim to optimize memory usage by exploiting CPU-GPU parallelism and mitigating the overhead of loading large models onto the GPU, but still remains a huge waste. HybriMoE (Zhong et al., 2025) introduces a queue that re-orders experts by token count, dispatching hot experts to the GPU and cold ones to the CPU. It uses a greedy algorithm to minimize per-layer computation cost and prefetches experts for the next layer during non-expert computations to further reduce on-demand loading pressure. But its greedy algorithm and scoring-based cache strategy lacks theoretical foundation, and miss some opportunities to further accelerate the inference.

I.4 Concurrent Works

Recent studies have begun to explore the intersection of Speculative Decoding and MoE. Saxena et al. (2025) selectively enables speculation to avoid slowdowns and dynamically tunes number of draft tokens to accelerate MoE serving, which overlooks the activation utility generated in a single step. Huang et al. (2025) analyzed the efficacy of SD for sparse MoE, yet their theoretical analysis relies on the assumption of uniformly activated experts, ignoring certain correlations among drafted tokens. System-oriented works like those by Wang et al. (2025c) and Zhuge et al. (2025) utilize CPU resources primarily to interleave the execution of draft and target models to maximize throughput, rather than offloading specific experts to reduce latency. While Chen et al. (2025b) and Wang et al. (2025a) leverage information from the draft model for activation prediction, they overlook the continuous activation trends across iterations. Bang et al. (2026) remains constrained to GPU-centric execution, failing to fully unlock the potential of heterogeneous computing, despite its self-assisted drafting. McDanel et al. (2026) suffers from precision loss for limiting expert budget during verification.

J Future Works

It is important to note that our work is orthogonal to a broad class of efforts aimed at accelerating speculative decoding itself. Techniques such as KV-cache compression and reuse (Sun et al., 2024), as well as retraining- or alignment-based methods that improve draft–target agreement (Li et al., 2024a), can be readily integrated with MoE-SpAc to further reduce both memory footprint and verification overhead, thereby improving end-to-end efficiency. In addition, self-speculation approaches (Zhang et al., 2024), which eliminate the need for a separate draft model, can significantly reduce memory consumption and free additional GPU capacity for expert caching, effectively increasing the achievable expert cache ratio within our framework. Our workload balancer can further take KV

cache offloading and scheduling into account (Qin et al., 2024; Jaillet et al., 2025), to avoid overhead caused by KV cache from both draft model and target model.

Beyond standard MoE architectures, emerging sparse model designs open new opportunities for utility-guided scheduling. Advanced architectures such as Mixture-of-Lookup-Experts (MoLE) (Jie et al., 2025) and conditional memory systems like Engram (Cheng et al., 2026) introduce new axes of sparsity and structured lookup mechanisms. These designs naturally expose richer and more stable access patterns, which could enable more accurate expert utility estimation and prefetching decisions. As models become increasingly sparse across parameters, activations, and memory accesses, the speculative-utility-based scheduling paradigm proposed in this work is expected to play an increasingly central role in efficient inference systems.

K Limitations

Despite its effectiveness, our approach has several limitations. First, MoE-SpAc fundamentally relies on speculative decoding to provide informative, frequency-valued activation signals; when speculative decoding is ineffective (e.g., due to low acceptance rates or strict latency constraints), the benefits of speculative utility estimation are reduced. Second, the proposed system introduces additional scheduling and bookkeeping complexity, including utility tracking, online threshold optimization, and asynchronous execution management, which may incur nontrivial engineering complexity in practice.

L Broader Impacts

This work improves the efficiency of Mixture-of-Experts (MoE) inference in resource-constrained settings. A potential positive impact is to lower the hardware and energy cost of serving large sparse models, which may broaden access to advanced language models on commodity devices and edge platforms.

At the same time, improved inference efficiency can also facilitate wider deployment of such models in downstream applications, including those where generated content may be misleading, inappropriate, or over-relied upon. Our work does not introduce new model capabilities or safety mechanisms; it is a systems optimization method for existing models. Therefore, the societal impact of this work depends primarily on how the underlying models are deployed and governed in practice.

We hope this work contributes to more efficient and accessible AI systems, while recognizing that efficiency gains should be accompanied by appropriate safety evaluation and application-level oversight.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions of repurposing speculative decoding as a lookahead signal for heterogeneous expert scheduling, and these claims are supported by both the theoretical analysis and experimental results reported in the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated *Limitations* discussion in Appendix K, covering reliance on speculative decoding effectiveness, additional scheduling/bookkeeping complexity, and the current focus on single-GPU, batch-size-one edge scenarios.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper provides formal assumptions and derivations for the theoretical analysis in the appendix, including the analysis of expert reuse, information gain, fault tolerance, and the bounded-drift justification for utility estimation.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper specifies the evaluated model setup, hardware platform, benchmark suite, key hyperparameters, baselines, and evaluation metrics, and code is publicly available, providing a reasonable path to reproduce the main empirical results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is available at `https://github.com/lshAlgorithm/MoE-SpAc`. The experiments are conducted on public benchmarks and existing open or accessible model assets.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides the experimental hardware, benchmark suite, baselines, target and draft models, batch size, draft length, utility upper bound, forgetting factor, and evaluation metrics needed to understand the reported inference results. There is no model training in the proposed method itself.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We provide the results with five repeated experiments in tables. In Figure 4 and Figure 6, we report the error bar in colors with opacity.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper specifies the main hardware platform, including a single NVIDIA GeForce RTX 4090 GPU and PCIe 4.0 CPU–GPU interconnect. Memory and time of execution are all reported in Section 4.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is a systems and inference-efficiency study conducted on existing models and benchmarks, and the authors are not aware of any aspect of the research that conflicts with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix L states that potential positive impacts include making large MoE models more accessible on resource-constrained devices, while possible negative impacts include enabling more efficient deployment of powerful language models in misuse scenarios.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new generative model or dataset; it proposes an inference framework built on existing models and benchmarks. Any model-specific safeguards are inherited from the original model providers.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the existing benchmarks, models, and baselines used in the experiments.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The main new asset is the implementation of the proposed inference framework. The paper points to the public code repository.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research and therefore does not require IRB approval.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The work studies inference for existing large language models and explicitly uses a target MoE model and a draft model as part of the experimental and methodological setup. These model usages are described in the paper's experiment section and are central to the evaluation setting. LLMs are used solely as the research target in this paper.