MOSAIC: Efficient Mixture-of-Agent Scheduling via Adaptive Aggregation and Inference Concurrency

Anonymous ACL submission

Abstract

Mixture-of-Agents (MoA) systems improve reasoning accuracy by routing each query to multiple expert LLMs and aggregating their outputs. Efficiently executing this workload on limited GPU resources has bottlenecks. Skill-based routing creates skewed expert demand, and combining instruction-tuned LLMs with long-reasoning models results in extreme variability in generation lengths. Consequently, traditional scheduling strategies suffer from significant GPU idling and throughput collapse due to load imbalances. We present MOSAIC, a scheduling framework to accelerate MoA workloads. First, we formulate an Integer Linear Program (ILP) based scheduler that jointly optimizes expert placement and per-worker prompt assignment from offline-profiled costs, replicating reasoning experts across workers while pinning lightweight ones. Second, MOSAIC uses confidence-aware adaptive aggregation, leveraging inter-expert agreement to bypass the heavy final aggregator LLM for consensus queries. In our 4-GPU system, MOSAIC achieves up to $\mathbf{2.5}\times$ expert-stage, $\mathbf{4.23}\times$ aggregator-stage and $\mathbf{1.7}$~$\mathbf{2.3}\times$ end-to-end speedups over the baseline scheduler, while matching accuracy within ± 0.1pp.

1 Introduction

Large language models (LLMs) (Touvron et al., 2023) yielding state-of-the-art (SoTA) accuracy on certain category of tasks, often fail to provide reasonable performance on different domains (*out-of-distribution*), due to their inherent training set and architectural bias towards certain tasks. Skill based adoption of LLMs (Chen et al., 2024; Wang et al., 2025a,b; Du et al., 2024; Wang et al., 2024a) has significantly boosted their performance on diverse set of tasks, including mathematical reasoning (Yu et al., 2024), commonsense and medical reasoning (Lobo et al., 2025). In specific, earlier works demonstrated benefits of *mixture-of-agents* (MoA), that harnesses the collective expertise of multiple LLMs to yield a unified robust framework. They are able to perform effectively avoiding task 'out-of-distribution' concern (Azizi et al., 2026; Chu et al., 2025). Here, the task-based expert models or agents[1] are meticulously chosen to form the candidate models or layers, creating ensemble of models (or expert layers) to form the test-time mixture. However, they often suffer from iterative generation from the *proposer models* (Wang et al., 2025a) that often follows an *aggregation* overhead (Wang et al., 2025b).

To improve the skill-based decision making of MoA, Symbolic-MoE (Chen et al., 2026) proposes adaptive, instance-level, skill-based routing. Additionally, it demonstrates the effectiveness of task-specific aggregator selection, avoiding the hard selection of best proposing model as aggregator. The skill-based router performs offline proposer and aggregator selection from a pool of LLMs and large reasoning models (LRMs) (Tian et al., 2026), which produce much longer, verbose outputs. Even though they introduce a batched inference strategy, inference time suffers from sequential model loading time for all solvers and high synchronization cost when deployed in multi-GPU setup. Additionally, as the aggregator LLM must process the combined output or reasoning traces of its experts, the prefill of aggregation can be extremely lengthy and computationally expensive at times.

Our Contributions. Motivated by this gap in MoA serving, we present MOSAIC. In specific, we make the following contributions:

- **We characterize heterogeneous MoA workloads on limited compute resources** and systematically identify the bottlenecks. We highlight how combining instruction-

[1] We alternatively refer a monolithic expert agent as solver or model in this paper.

tuned LLMs with long-reasoning models causes extreme variability in generation lengths (upto 13×) and how skill-based routing induces popularity skew across expert pools (upto 92×).

- **We formulate the expert-generation phase as an ILP** that jointly optimizes model-to-worker placement, per-worker prompt counts, and selective replication of heavy reasoning experts under a per-worker capacity limit.
- **We introduce a confidence-aware adaptive aggregation gate** that bypasses the aggregator LLM on questions where the experts converge on a majority answer.
- We evaluate MOSAIC on a 4-GPU server across MMLU-Pro, MedMCQA, and GPQA. Against a round-robin baseline at the first-seen-order of queries, MOSAIC delivers up to 2.54× expert-stage and 4.23× aggregator-stage speedup, translating into 1.71-2.34× end-to-end wall-clock reduction while matching baseline accuracy within ±0.1pp.

2 Background and Related Work

Recent work has shown that using multiple LLM responses can improve answer quality on challenging reasoning tasks. Mixture-of-Agents (MoA) (Wang et al., 2025b) proposes a layered inference procedure in which multiple LLM agents generate responses, and later agents use previous responses as auxiliary context to produce a stronger final answer. Symbolic-MoE (Chen et al., 2026) selects expert LLMs at the instance level using symbolic skill descriptions, then aggregating their generated reasoning traces into a final answer.

We focus on the scheduling problem created by such multi-agent inference workloads. We formulate it as an offline scheduling task where a batch of questions and their assigned experts are known in advance. Specifically, our problem begins after the expert selection step: once the skill-based router assigns a subset of experts to a question, the system must efficiently execute these independent model calls on limited GPU resources. Such workloads naturally arise when evaluating multi-expert reasoning systems over fixed benchmark suites; for example, Symbolic-MoE (Chen et al., 2026) uses MMLU-Pro (Wang et al., 2024b), GPQA (Rein et al., 2024), and MedMCQA (Pal et al., 2022).

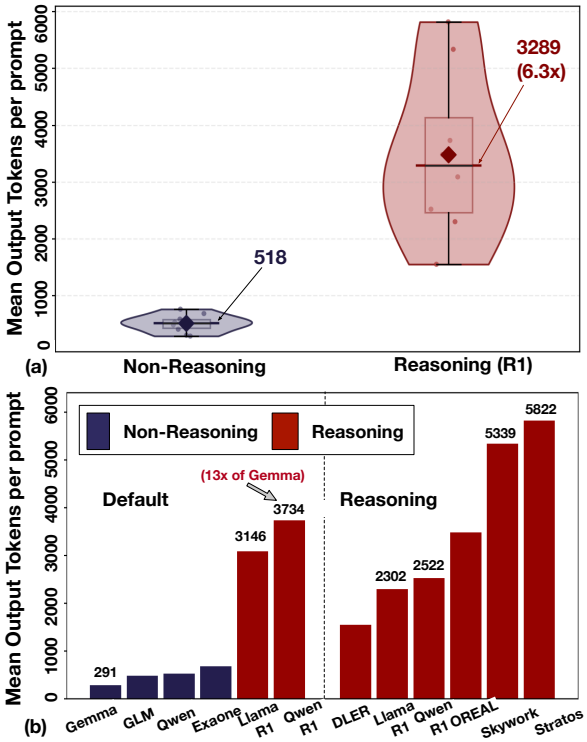

Figure 1: Output token distribution (a); and mean output tokens per prompt (b) across reasoning and non-reasoning models for MMLU pro.

This setting is different from standard LLM serving. The pool of expert models can exceed available GPU capacity, and the skill-based router produces highly skewed selection frequencies across models. Furthermore, mixing standard instruction-tuned LLMs with long-reasoning models introduces highly variable generation lengths. These dynamics complicate resource allocation.

A natural approach to serving such workloads is *static solver-per-worker placement*: deploy one high-throughput LLM serving instance per GPU and pin each expert to a fixed worker. This follows the single-model serving regime optimized by systems such as vLLM and SGLang (Kwon et al., 2023; Zheng et al., 2024). While this avoids redundant model loading and preserves large per-expert batches, router-induced skew in solver demand and variability in generation length can leave some workers idle and others as stragglers. On the other hand, *data-parallel execution* can mitigate load imbalance by placing popular models on multiple workers, a strategy explored in traditional DNN serving systems such as INFaaS (Romero et al., 2021). While DNN workloads often have predictable execution costs, LLM workloads differ fundamentally: generation is autoregressive, output lengths are input-dependent, and decod-

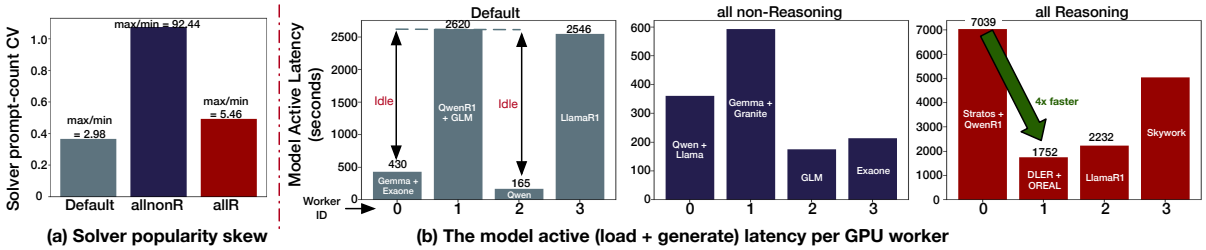

Figure 2: (a) Solver popularity skew across model pools; (b) Overall state of GPU idling for MMLU-Pro

ing cost can vary substantially across prompts and models. Recent multi-LLM serving systems such as MuxServe (Duan et al., 2024) exploit LLM-specific prefill/decode behavior to colocate and multiplex multiple LLMs on a fixed GPU cluster. However, they target online serving settings with SLO-oriented objectives, whereas our setting is a finite offline batch of pre-selected expert calls. Finally, while *tensor/model parallelism* is critical for serving models that exceed single-GPU capacity (Li et al., 2023; Aminabadi et al., 2022), we assume each expert fits within a single GPU. Therefore, the core challenge in our setting is the optimal placement and scheduling of independent experts across a worker cluster.

3 Motivation

Serving multi-agent workloads presents a unique set of scheduling challenges. Given a batch of requests with pre-assigned experts, our objective is to minimize makespan and maximize hardware utilization. We examine the distinct characteristics of these workloads below:

Observation 1. *Heterogeneous model categories and solver pools yield highly diverse generation token count, with LRMs often showing long tail distribution of token count across candidate solvers.*

Figure 1(a) summarizes an experiment with MMLU-Pro, across 12 models of different classes (6 reasoning and 6 non-reasoning). The LRMs' average generation length (3289 tokens/prompt) is 6.3× longer than that of an LLM (518 tokens/prompt). Further, LRMs often show long tail distribution of token generation, ranging from 1500 to 6000 tokens per prompt. Additionally, the token-budget profile of a solver is not static but dependent on the composition of the broader solver pool. Figure 1b shows two different solver pools, the default has two LRMs and four LLMs while the all reasoning (allR) pool has all six solver candidates as LRMs. In default pool, LLamaR1 and QwenR1 being the only LRMs, generate 3146 and 3734 tokens per prompt on average respectively. But on allR, they get a broader, mixed-difficulty subset of questions where their specific profile excels, with their token costs per prompt dropping to 2302 (-27%) and 2522 (-32%). Meanwhile, the most-demanding questions migrate to specialized long-CoT solvers (Skywork: 5339 tokens/prompt; Stratos: 5822 tokens/prompt). **This shows that a model's compute profile is dynamically shaped by its peers, since the solver pool affects the difficulty of the questions routed to that model**.

Observation 2. *Popularity of different solvers across model pools largely vary as a result of skill-weighted sampling by the router.*

With the default reasoning + non-reasoning mix on the MMLU-Pro test set (2100 questions × 3 solvers = 6300 selections), Gemma is selected most often (1632) and Exaone least (547), giving a max/min ratio of 2.98×. In Figure 2a, y-axis represents population coefficient-of-variance for each model pool. In all non-Reasonings (allnonR), the strongest remaining models dominate (Gemma 53%, Qwen 24%) while Llama and Granite each fall below 2%, pushing max/min to 92.4×. **This shows that the popularity skew of solvers is sensitive to the pool composition.**

Figure 2b shows what happens when these skewed solver demands combine with token-budget disparity. We consider a round-robin allocation (in first-seen-solver order in the available query queue), a scheme that commits each solver to a worker before any cost or count signal is observed. In the default pool, reasoning workers dominate wall time: worker 1 (QwenR1+GLM, 2620s) and worker 3 (LlamaR1, 2546s) take 6-16× longer than worker 0 (Gemma+Exaone, 430s) and worker 2 (Qwen, 165s). Notably, worker 0 (W0) holds the most popular solver (Gemma) in the entire pool, yet finishes earlier; because reasoning models generate 6-13× more tokens per prompt. Per-prompt cost dominates popularity-

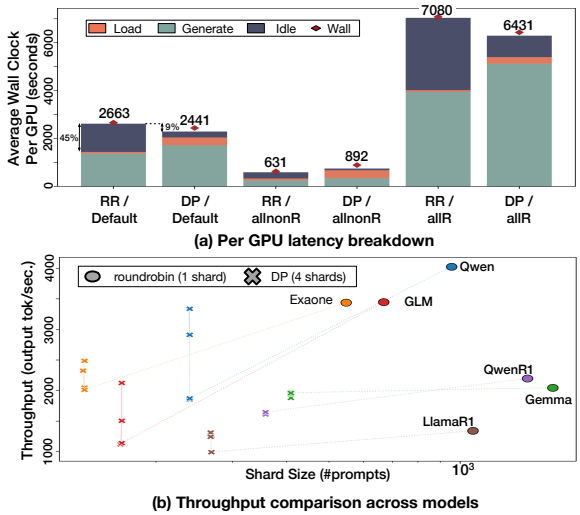

Figure 3: Comparing data-concurrent replication with round-robin allocation

count whenever the two solver classes mix. The idle GPU time is also significant for allR and allnonR cases. No single factor explains the imbalance in GPU idling: a cost-aware scheduler must balance $\sum$(prompts $\times$ per-prompt cost) per worker rather than either quantity alone.

Observation 3. *Data-concurrent replication trades straggler-idle time for two new costs: serialized model loads and per-shard throughput collapse.* Data parallelism (DP) can counter the previous GPU-idling by keeping a complete copy of each solver on all GPUs and shard its prompts evenly across workers. Figure 3a confirms the tradeoff is pool-dependent. DP cuts the ~45% idle share of roundrobin runtime to give a overall ~9% wall-time improvement in default model-pool, and similarly improves allR (7080s to 6431s). In allnonR, however, DP is ~29% slower (631s vs 892s), the per-solver generate budget is too small to amortize serialized loads. DP also incurs a second cost from over-sharding: once each model's prompts are split four ways, per-shard throughput drops 40–60% (in Figure 3b) for the smaller instruction-tuned solvers (Qwen, Exaone, GLM). **Taken together, DP only wins when each solver's generate budget is large enough to absorb both the serialized load tax and the per-shard throughput loss**. Full per-solver breakdown and critical-path analysis are in Section A.6.

4 MOSAIC: Methodology

Figure 4 shows the proposed MOSAIC framework. The input prompts are routed to specific experts based on skill-based selection, and then executed on a multi-GPU server to generate the expert outputs. We aggregate all expert outputs to generate the final output. The key components of MOSAIC flow are: (a) an ILP-based multi-model scheduler for expert generation, and (b) an adaptive aggregation mechanism.

4.1 MOSAIC: Multi-Model Scheduling

4.1.1 Problem Setting

We consider an MoA deployment that serves a fixed prompt set on N workers ($n \in \{1, \ldots, N\}$) drawn from a pool of M heterogeneous experts $\mathcal{M}$ with $M > N$. Each worker holds one model in GPU at once and executes its tasks as sequential vLLM sessions. For every expert m, let ℓ_m be the session-start (load) cost. The upstream skill-based recruit step is completed before scheduling and supplies the per-expert total prompt count N_m. Each worker's wall time decomposes into a setup term $\sum_m \ell_m \mathbf{1}[\text{worker } n \text{ loads } m]$ and a generation term; the scheduling objective is to minimise the makespan.

$$C = \max_{n \in \{1,\ldots,N\}} T_n. \tag{1}$$

4.1.2 Workload Characterisation

Output length is the dominant cost driver of MoA serving: every recruited expert must emit a long-form rationale, so generation time grows linearly with output tokens once the batch has reached steady state. Crucially, each expert exhibits a stable, model-specific output-token distribution: instruction-tuned experts produce short responses while reasoning-tuned experts produce heavy long tails. So a per-model summary suffices to predict per-session wall time.

We therefore profile each expert m on a small calibration subset of the training data, extracting its mean and maximum output-token counts

$$\bar{O}_m = \mathbb{E}[O \mid m], \qquad O_m^{\max} = \max_q O_{m,q}. \tag{2}$$

A per-model regression $T_m(N_m, \bar{O}_m, O_m^{\max})$ is then fitted from a short multi-N profiling sweep on the calibration set; the regression captures how vLLM's continuous-batching scheduler trades parallelism against tail latency, so the predicted session wall time remains accurate as the per-session prompt count N_m varies across experts. Dividing T_m by N_m gives the per-prompt scheduling cost

$$\tau_m \triangleq \frac{T_m(N_m, \bar{O}_m, O_m^{\max})}{N_m}, \tag{3}$$

which is a constant coefficient at solve time.

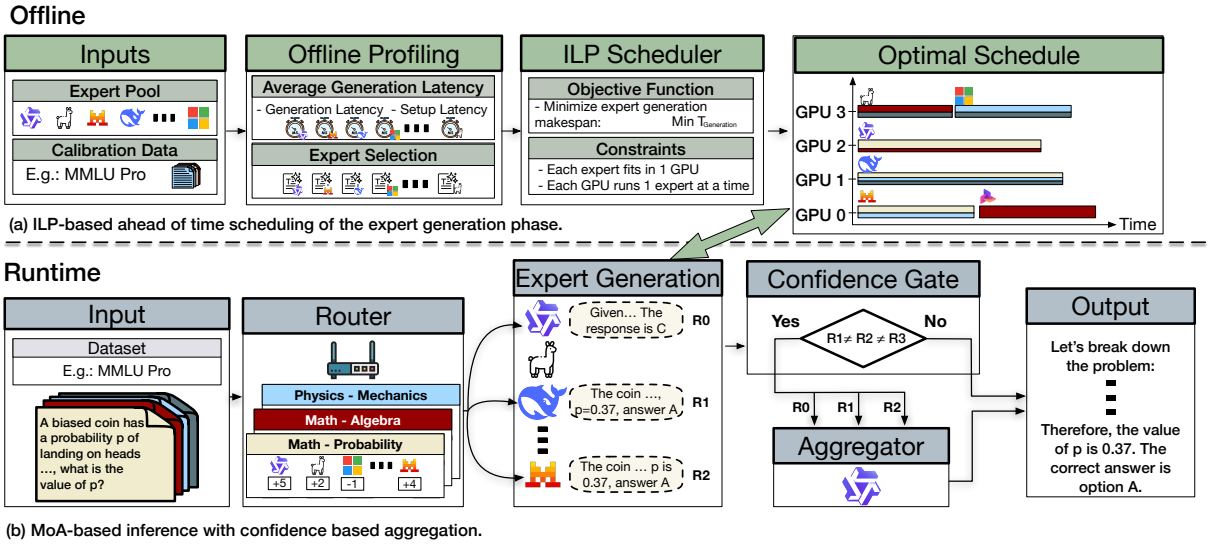

Figure 4: Detailed overview of the MOSAIC framework.

4.1.3 Scheduling Algorithm Formulation

In our mixture-of-agent setup, the scheduling decisions are discrete: model placement on a worker is binary, prompt counts are integer, and the capacity bounds are small integer caps. Therefore, the schedule is naturally cast as an ILP. A greedy list-scheduling heuristic would forfeit the joint optimisation of placement and per-worker prompt count under the replica cap; at the problem sizes encountered here ($M \leq 6$ experts, $N = 4$ workers), the search space is small enough that a modern CP-SAT solver returns provably optimal solutions in milliseconds, so the price of exactness over a heuristic is negligible.

Decision variables. For every (m, n) we introduce $Y_{m,n} \in \{0, 1\}$ to indicate that worker n loads model m, and $X_{m,n} \in \mathbb{Z}_{\geq 0}$ to count the number of model-m prompts placed on worker n.

Worker time.

$$T_n = \sum_m \ell_m Y_{m,n} + \sum_m \tau_m X_{m,n}. \tag{4}$$

Constraints. Four constraint families restrict the feasible region.

First, every prompt of every model must be placed on some worker:

$$\sum_n X_{m,n} = N_m, \quad \forall m. \tag{5}$$

Second, prompts of model m may only be served by workers that load m, coupling the count variables to the placement variables:

$$X_{m,n} \leq N_m Y_{m,n}, \quad \forall m, n. \tag{6}$$

Third, each model is loaded on at least one worker and at most on a per-model cap $R_m^{\max}$ defined below:

$$1 \leq \sum_n Y_{m,n} \leq R_m^{\max}, \quad \forall m. \tag{7}$$

The cap is set by a heuristic that respects each model's setup/execution economics. Writing $\hat{T}_m \triangleq T_m(N_m, \bar{O}_m, O_m^{\max})$, we define the load-to-execution ratio and the replica cap as

$$\rho_m = \frac{\ell_m}{\hat{T}_m}, \qquad R_m^{\max} = \text{clamp}\left(1, \left\lfloor \frac{\hat{T}_m}{\ell_m} \right\rfloor, N\right), \tag{8}$$

admitting a model to as many workers as its execution time can amortise one additional load each: reasoning experts with $\rho_m \ll 1$ are eligible for replication up to N workers, whereas models with comparable load and execution costs are pinned to a single worker.

Finally, no worker hosts more than K distinct models:

$$\sum_m Y_{m,n} \leq K, \quad \forall n. \tag{9}$$

With $K < M$ the solver cannot place every expert on its own worker; this is the lever that forces the longest-running experts to be replicated across multiple workers.

Objective.

$$\min C \quad \text{s.t.} \quad C \geq T_n, \ \forall n. \tag{10}$$

Handling heterogeneity. The integer prompt count $X_{m,n}$ is the central degree of freedom of the formulation, permitting asymmetric per-worker

splits of a single expert that no whole-model or fixed-bucket allocation can express in closed form. The per-model cap $R_m^{\max}$ translates per-model setup/execution economics into placement budgets, and the per-worker cap K binds the joint optimisation so that the longest-running experts are replicated only as far as the makespan demands. The resulting topology that each reasoning expert paired with a distinct lightweight partner on each of several workers, emerges from the joint minimisation rather than being designed in.

4.2 MOSAIC: Adaptive Aggregation

In expert selection phase, solvers are recruited per-question against varied skill profiles. The k experts answering a question are deliberately diverse. When such skill-aligned experts converge on the same answer, that convergence is a strong signal that the answer is correct. It is potentially stronger than k independent samples from a single model (self-consistency). We exploit this signal to skip the aggregator LLM on questions where experts agree.

Observation. On MMLU-Pro, 960 of 2100 questions (45.71%) get a unanimous 3:0 vote among the three recruited experts. In this 3:0 bucket, the un-aggregated majority answer matches the golden output 82.92% of the time. It is essentially similar (or better) in accuracy when all queries in this bucket are fully passed to the aggregator LLM (82.40%). Similar trend holds across all four benchmarks: 3:0-bucket hit rates are 47.48% (MedMCQA), 43.33% (AIME24), and 37.37% (GPQA), with majority accuracy matching or exceeding the aggregator in every case (Table 1).

Let $\{a_1, \ldots, a_k\}$ be the answers of k recruited experts to a question q, and let $C(q) \in [0, 1]$ be a confidence function over those answers. We define a confidence gate with threshold τ:

$$\hat{y}(q) \;=\; \begin{cases} \text{majority}\big(\{a_i\}_{i=1}^{k}\big) & \text{if } C(q) \geq \tau, \\ \text{Aggregator}\big(q, \{a_i\}_{i=1}^{k}\big) & \text{otherwise.} \end{cases} \tag{11}$$

Here we instantiate C as the *plurality fraction*

$$C_{\text{vote}}(q) \;=\; \max_{a} \frac{\big|\{i : a_i = a\}\big|}{k},$$

So for k=3 the buckets 3:0, 2:1, and 1:1:1 correspond to C=1, $C = \frac{2}{3}$, and $C = \frac{1}{3}$ respectively. We report two operating points: a conservative gate τ=1 (gate 3:0 only) and an aggressive gate τ=2/3 (gate 3:0 and 2:1). Algorithm is available in Section A.5.

The formulation in Equation (11) is agnostic to the choice of confidence estimator. Any signal correlated with response quality (for example response-level uncertainty, or context-aware features) can be substituted for inter-expert agreement in the same gate.

5 Evaluation and Results

5.1 Evaluation Methodology

Platform and datasets All experiments run on a single host equipped with four NVIDIA A100 80 GB GPUs, using vLLM 0.7.1. One model is loaded per GPU at a time and swapped sequentially across vLLM sessions. The single-host, multi-GPU configuration is the minimal setting that exposes the cross-worker scheduling problem the paper formalises while excluding network-level confounds; the 80 GB envelope per GPU is large enough to hold any one of our experts together with its full 32 k-token KV cache without sharding, so the only setup cost incurred by the scheduler is the per-session model load ℓ_m. All three benchmarks are served from the same hybrid experts pool (reasoning-tuned and instruction-tuned experts; full list in Appendix A.2); per-expert prompt slot counts on each benchmark are reported in Appendix A.3.

Calibration and baseline. The per-model output statistics $\bar{O}_m$, $O_m^{\max}$ are estimated from a small train-only subset of each benchmark (350 questions for MMLU_Pro and 504 for MedMCQA); the scheduler never observes the test set. The comparison baseline is round-robin (RR) placement, $m \mapsto m \bmod N$, which requires no calibration and incurs exactly one load per model. RR is the natural multi-GPU baseline against which any non-trivial scheduling policy must demonstrate value.

5.2 MOSAIC Accuracy Result and Analysis

Table 1: Accuracy (Δpp) and aggregator-call savings (skip%) under confidence-gated aggregation.

Task	Baseline	τ = 1 (3:0)		τ = 2/3 (3:0 + 2:1)	
	acc. (%)	Δpp	skip%	Δpp	skip%
MMLU-Pro	63.19	+0.24	45.71	−0.62	84.62
MedMCQA	59.69	−0.04	47.48	+0.32	90.18
GPQA	56.57	+1.00	37.37	+0.49	87.37

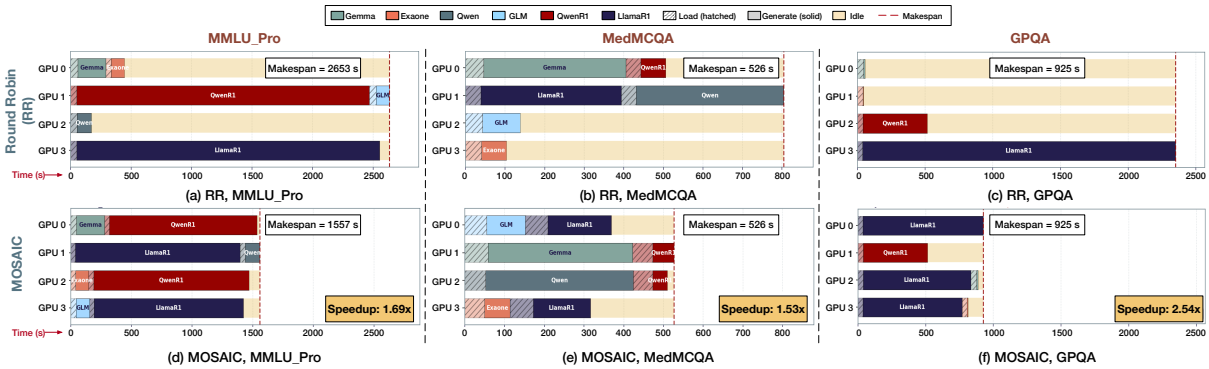

Figure 5: Per-GPU execution timeline of our schedule (bottom row) and the baseline round-robin baseline (top row) on 3 selected datasets: MMLU_Pro (left), MedMCQA (middle), and GPQA (right). Each bar represents one GPU worker session. MOSAIC reduces the workload's makespan by assigning balanced pairs of reasoning and non-reasoning experts to the workers.

We evaluate the accuracy change for a task and the latency savings in the aggregation phase from the gating mechanism described in Equation (11). Table 1 reports accuracy under the conservative (τ=1, skip 3:0) and aggressive (τ=2/3, skip 3:0 and 2:1) gates against the full-aggregator baseline. The conservative gate matches the baseline within ±0.1pp on large benchmarks like MMLU-Pro, MedMCQA and *improves* accuracy on GPQA while bypassing 37-48% of aggregator calls. The aggressive gate bypasses 73-90% of aggregation calls while closely matching the baseline performance of all tasks considered.

5.3 MOSAIC Speedup Result and Analysis

Expert Scheduling Speedup Figure 5 reports the expert-phase per-GPU timelines of MOSAIC against the round-robin baseline on three benchmarks. Each configuration is repeated over three random seeds to control for the stochastic component of vLLM's continuous-batching scheduler; per-seed walls are listed in Appendix A.4. MOSAIC delivers a 1.69× speedup on MMLU_Pro, 1.53× on MedMCQA, and 2.54× on GPQA, with setup cost in every case comparable to RR's. The small number of additional model loads required to replicate the heaviest experts is paid back many times over by the resulting drop in worker-wall imbalance. The mechanism is visible in the figure: in every RR panel one or two workers dominate the wall, idling the others (the dotted yellow regions); MOSAIC pulls the heaviest expert onto two or more workers, pairs each replicated slice with a single-worker instruction-tuned partner, and so contracts the makespan to near the parallel lower bound. The same structural decision drives all three speedups despite a ∼20× variation in workload size (594 prompts on GPQA, 12 549 on MedMCQA) and very different recruit distributions, indicating that the win is not specific to any single benchmark's τ_m profile.

The capacity cap K=2 adopted throughout is the smallest value that satisfies the two requirements implied by Equation (8). First, a feasibility lower bound $K \geq \lceil M/N \rceil = 2$ holds independently of any calibration. Second, the per-model replica caps measured from calibration sit at $R_m^{\max}$=N for the two reasoning experts ($\rho_m \approx 0.02$, since their generation time dwarfs the load cost) but $R_m^{\max}$=1 for the four instruction-tuned experts ($\rho_m \in [0.2, 0.95]$, where an extra load would consume most of the savings). K=2 is therefore precisely the capacity that grants the reasoning experts the room they need to split without admitting a second load of any instruction-tuned expert: K=1 is infeasible and K=3 would over-replicate models whose $R_m^{\max}$=1 already forbids the extra load. The solver treats K=2 as a soft budget, saturating it on workers that benefit from a reasoning slice and leaving it slack elsewhere; this is most pronounced on GPQA, where the recruiter sends 72% of all prompts to LlamaR1 and MOSAIC consequently replicates LlamaR1 across three workers, yielding the largest observed speedup, because RR's single-worker LlamaR1 allocation is the bottleneck.

Adaptive Aggregation Speedup On MMLU-Pro, gating both the 3:0 and 2:1 buckets reduces the aggregator latency from 1514.7s to 870.5s, when we ran the aggregator with tensor-parallelism across 4 available GPUs. This results a **1.74**× aggregator-stage speedup. On MedMCQA, the same gate bypasses ~90% of questions and

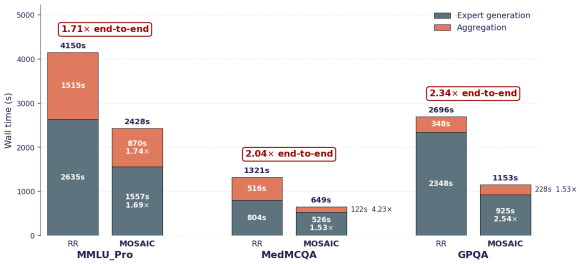

Figure 6: End-to-end wall-time decomposition. For each benchmark a pair of stacked bars compares the round-robin baseline (left) with MOSAIC (right)

yields **4.23**× speedup (516.49s to 122.09s). For GPQA, skipping ~87% questions give speedup of **1.53**× (348s to 227.73s).

Overall Speedup End-to-end pipeline runtime is the sum of expert-generation and aggregator wall time. MOSAIC accelerates the two stages by orthogonal mechanisms: workload-aware ILP scheduling on the expert side, and confidence-based gating on the aggregator side. Figure 6 reports the absolute wall-time decomposition for both schedules on the three benchmarks. Combining the two stages, MOSAIC reduces the end-to-end wall time from 4149.9 s to 2427.6 s on MMLU_Pro (**1.71**×), from 1321.0 s to 648.6 s on MedMCQA (**2.04**×), and from 2696.4 s to 1153.1 s on GPQA (**2.34**×). On MedMCQA, the medical-MCQ questions reach a clear majority among the three expert votes most of the time, so the aggregator gate alone gives 4.23× and accounts for most of the end-to-end gain. On GPQA the R1-dominated workload is where the scheduler pays off most (2.54× expert).

5.4 Ablation on aggregation skip results

Speedup in our experiments is sub-linear to the gating rate in aggregator calls. We bypass 84.62% of MMLU-Pro questions but recover only 1.74× rather than about ~6×. The reason is that the residual 1:1:1 bucket has the longest reasoning traces from the experts. For MMLU-Pro, the mean output length is 2696 tokens for 1:1:1 vs. 1064 tokens for 3:0, a 2.53× ratio. The gating mechanism ensures that hard questions requiring expert-generated long traces are handled by the aggregator. Figure 7a shows the relationship between CoT length and expert agreement. The mean output tokens per question decrease monotonically across agreement buckets and within each solver class. The ECDFs (in Figure 7b) show that questions having unanimous answer resolve in shorter traces well before the ones having most disagreement do (Spearman $\rho = -0.27$, $p < 10^{-30}$).

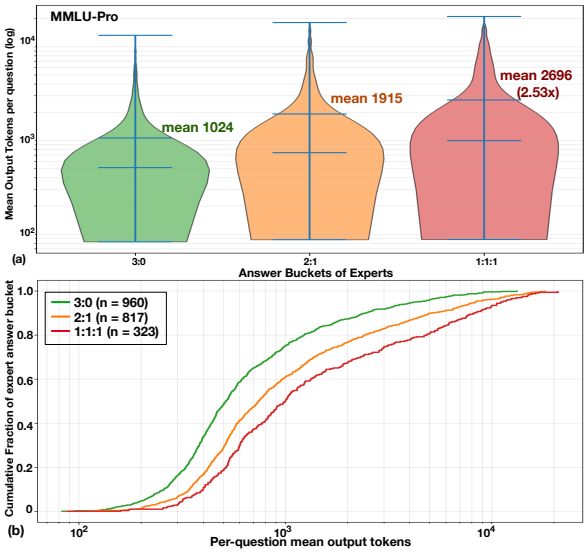

Figure 7: Per-question mean output tokens by agreement bucket. a: violin plot on a log scale, all solvers pooled. b: ECDF by bucket. Unanimous (3:0) questions resolve in markedly fewer tokens than disagreement (1:1:1) questions (2.53× ratio, Spearman $\rho = -0.27$).

6 Conclusions

MoA-style inference creates a distinct scheduling problem. MOSAIC addresses this setting by jointly optimizing the expert-generation phase and the aggregation phase. For expert generation, it uses offline profiling and ILP-based scheduling to balance model load cost, generation cost, and selective replication across GPUs. For aggregation, MOSAIC exploits inter-expert agreement as a confidence signal to skip unnecessary aggregator calls. Across benchmarks, it reduces expert-stage makespan and substantially lowers aggregation overhead while preserving accuracy. Overall, MOSAIC demonstrates the need for workload-aware scheduling in MoA serving.

7 Limitations

While in MOSAIC, we demonstrate the benefits in accelerating the mixture of model serving, this work faces limitations in the static selection of the 'confidence criterion'. In specific, for any task type, the user needs to select the aggregation skip criteria in advance that does not change during the inference serving stage. Static 'confidence selection' along with associated hyperparameter choices of thresholding remains as a limitation of the current work. Further, the ILP solver can take higher bene-

fit for tasks that has categorical partitioning based on task sub-type or criticality (example: MMLU-Pro), compared to the tasks that has no further sub-partitioning (example: GPQA).

Our MoA serving acceleration may benefit the expert model based task processing in an Agentic framework. However, extending this work of LLMs-based monolithic mixture-of-agent serving may not remain straightforward to heterogeneous agentic workload execution as many of the tasks may not be bottle-necked by the LLM inference execution latency. Further investigation on fruitful adoption of MOSAIC in such workloads remains as an interesting future research.

8 LLM Usage

Generative AI tools (Gemini, Claude) were used to assist in writing scripts for plotting results and automating the experimental workflow. All core research code was developed solely by the authors.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, and 1 others. 2022. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15. IEEE.

Seyedarmin Azizi, Erfan Baghaei Potraghloo, Minoo Ahmadi, Souvik Kundu, and Massoud Pedram. 2026. Power-smc: Low-latency sequence-level power sampling for training-free llm reasoning. *arXiv preprint arXiv:2602.10273*.

Bespoke Labs. 2025. Bespoke-stratos-7b. Hugging Face model card. Accessed: 2026-05-25.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, and 81 others. 2024. Internlm2 technical report. *Preprint*, arXiv:2403.17297.

Justin Chen, Swarnadeep Saha, and Mohit Bansal. 2024. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7066–7085.

Justin Chih-Yao Chen, Sukwon Yun, Elias Stengel-Eskin, Tianlong Chen, and Mohit Bansal. 2026. Symbolic mixture-of-experts: Adaptive skill-based routing for heterogeneous reasoning. *International Conference on Machine Learning*.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*.

ContactDoctor. 2024. ContactDoctor-Bio-Medical: A high-performance biomedical language model. Hugging Face model card.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first international conference on machine learning*.

Jiangfei Duan, Runyu Lu, Haojie Duanmu, Xiuhong Li, Xingcheng Zhang, Dahua Lin, Ion Stoica, and Hao Zhang. 2024. Muxserve: flexible spatial-temporal multiplexing for multiple llm serving. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, and 37 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. Openthoughts: Data recipes for reasoning models. *Preprint*, arXiv:2506.04178.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang,

Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. 2025. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

IBM Granite Team. 2024. Granite-3.1-8b-instruct. Hugging Face model card. Accessed: 2026-05-25.

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

Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E Gonzalez, and 1 others. 2023. {AlpaServe}: Statistical multiplexing with model parallelism for deep learning serving. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, pages 663–679.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, and 1 others. 2025. Dler: Doing length penalty right-incentivizing more intelligence per token via reinforcement learning. *arXiv preprint arXiv:2510.15110*.

Elita Lobo, Chirag Agarwal, and Himabindu Lakkaraju. 2025. On the impact of fine-tuning on chain-of-thought reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11679–11698.

Mistral AI Team. 2024a. Mathstral-7b-v0.1. Hugging Face model card. Accessed: 2026-05-25.

Mistral AI Team. 2024b. Mistral-nemo-instruct-2407. Hugging Face model card. Accessed: 2026-05-25.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

LG AI Research. 2024. Exaone 3.5: Series of large language models for real-world use cases. *arXiv preprint arXiv:2412.04862*.

Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. 2021. {INFaaS}: Automated model-less inference serving. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 397–411.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Gemma Team. 2024a. Gemma.

Qwen Team. 2024b. Qwen2.5: A party of foundation models.

Jiayi Tian, Seyedarmin Azizi, Yequan Zhao, Erfan Baghaei Potraghloo, Sean McPherson, Sharath Nittur Sridhar, Zhengyang Wang, Zheng Zhang, Massoud Pedram, and Souvik Kundu. 2026. Skipkv: Selective skipping of kv generation and storage for efficient inference with large reasoning models. *Ninth Annual Conference on Machine Learning and Systems*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Chenyu Wang, Zishen Wan, Hao Kang, Emma Chen, Zhiqiang Xie, Tushar Krishna, Vijay Janapa Reddi, and Yilun Du. 2025a. Slm-mux: Orchestrating small language models for reasoning. *arXiv preprint arXiv:2510.05077*.

Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. 2024a. Fusing models with complementary expertise. In *International Conference on Learning Representations*, volume 2024, pages 45284–45306.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y Zou. 2025b. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*, volume 2025, pages 33944–33963.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, volume 2024, pages 45040–45061.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. Sglang: efficient execution of structured language model programs. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

A Appendix

A.1 Full Model Pool

Table 5 lists the full set of models considered in our study, separated into reasoning and non-reasoning models.

A.2 Hybrid Model Pool Used in Evaluation

The six experts that constitute the hybrid pool used throughout Section 5 are listed in Table 2. Two of the six are reasoning-tuned and emit long ⟨think⟩ rationales; the remaining four are instruction-tuned and produce concise answers. The mean output length differs by roughly an order of magnitude across the two families, which is the heterogeneity our scheduler is designed to absorb.

Each expert runs as a single vLLM session at tensor-parallel rank 1. Per-session load cost ℓ_m ranges between 38–47 s and is dominated by weight transfer and KV-cache pre-allocation on a single A100 80 GB.

Table 2: Six-expert hybrid pool. Output statistics are measured on the MMLU_Pro test set and rounded to representative values; analogous ranges hold on MedMCQA.

Expert	Family	Mean output (tok)
LlamaR1	reasoning-tuned	$\sim$ 3 000
QwenR1	reasoning-tuned	$\sim$ 3 000
Gemma	instruction-tuned	$\sim$ 200
Exaone	instruction-tuned	$\sim$ 200
GLM	instruction-tuned	$\sim$ 200
Qwen	instruction-tuned	$\sim$ 200

A.3 Datasets

All three benchmarks are used in their standard test splits. Each test question is routed to three experts by the upstream skill-based recruit, producing a per-expert slot count N_m summarised in Table 3.

Table 3: Per-expert prompt slots on the three benchmarks at seed 0. The three workloads exercise very different recruit shapes: MMLU_Pro spreads slots across all six experts, MedMCQA concentrates on the instruction-tuned family, and GPQA is dominated by LlamaR1.

Expert	MMLU_Pro N_m	MedMCQA N_m	GPQA N_m
LlamaR1	1 070	620	429
QwenR1	1 428	144	158
Gemma	1 632	4 195	4
Exaone	547	625	3
GLM	667	1 283	0
Qwen	956	5 682	0
Total prompts	**6 300**	**12 549**	**594**
Test questions	2 100	4 183	198

MMLU_Pro contains 2 100 multiple-choice questions across 14 STEM and humanities categories. MedMCQA contributes 4 183 medical multiple-choice questions. GPQA contributes 198 graduate-level science questions whose narrow skill profile drives the recruiter to concentrate $\sim$72% of its slots on LlamaR1; the two lowest-scoring experts (GLM, Qwen) are never selected. All three benchmarks are served by the same six-expert hybrid pool described in Appendix A.2.

A.4 Multi-seed Validation

Each configuration is repeated over three random seeds (0, 1, 2) to control for the stochastic component of vLLM's continuous-batching scheduler. Table 4 reports the per-seed expert-phase wall time of MOSAIC on the three benchmarks.

Table 4: Per-seed wall time of MOSAIC. All seed wall times are within $\pm$6% of the seed-0 value, confirming that the schedule is not over-fit to a particular recruit realisation.

Dataset	Seed 0 (s)	Seed 1 (s)	Seed 2 (s)
MMLU_Pro	1601	1668	1637
MedMCQA	585	551	527
GPQA	925	945	896

A.5 Adaptive Aggregation Algorithm

Algorithm 1 Confidence-Aware Adaptive Aggregation

Require: question q, experts $E_1, \ldots, E_k$ from skill recruitment, threshold τ

$a_i \leftarrow \textsc{Extract}(E_i(q))$ for $i = 1, \ldots, k$
$C \leftarrow \max_a |\{i : a_i = a\}| / k$
if $C \geq \tau$ **then**
  return majority($\{a_i\}$) ▷ aggregator is skipped
else
  return Aggregator(q, $\{a_i\}$)
end if

A.6 The Pitfalls of Data Concurrency and Over-Sharding

In data parallelism (DP), a complete copy of each solver is loaded on all 4 GPUs. Prompts requiring that solver are evenly shared across workers. Solvers are processed sequentially with a sync barrier between them. This strategy differs from solver-per-worker or round-robin placement in critical path: runs each worker's assigned solvers serially with model loads overlapped across workers, giving expert_latency $\approx \max_w(\sum_m \text{load} + \text{gen})$ and exposing within-worker stragglers as idle time. DP serializes over models but parallelizes within each, giving expert_latency $\approx \sum_m \max_w(\text{load} + \text{gen})$ and replacing straggler-idle with a replicated load tax (every model loaded on every worker). Figure 2a confirms the tradeoff; DP cuts the ~45% idle share of roundrobin runtime to give a overall ~9% wall-time improvement in default model-pool, and similarly improves allR (7080s to 6431s). In allnonR, however, DP is ~29% slower (631s vs 892s), the per-solver generate budget is too small to amortize serialized loads.

DP incurs a second cost due to throughput collapse from over-sharding. vLLM(Kwon et al., 2023) relies on continuous batching, which only reaches peak tok/s when enough prompts are in flight to keep the model fully utilized. Below a model and GPU-dependent threshold, throughput scales sub-linearly with batch size, and at very small batches, the GPU starves. Figure 2b makes this visible by plotting the throughput of each model in the default model-pool for both roundrobin (full data shard) and DP (1/4 shard). The full-shard points sit at the right for each color (solver), showing peak-throughput. The DP points are located on the left (137-239 prompts per shards) and throughput is reduced by 40-60% for Qwen, Exaone, and GLM. Gemma is an exception because skill-weighted recruitment over-picked it in the default pool, even its 1/4 shard (~408 prompts) stays above its saturation threshold. Thus, DP barely costs throughput for Gemma. Reasoning models cluster around 1000-2000 tok/s, regardless of shard size. Their long autoregressive decode phase is memory-bandwidth bound, so a handful of in-flight prompts already saturate HBM and additional batch yields little.

Taken together, DP trades off straggler-idle time for two new costs: serialized load and reduced throughput per shard. It only wins when the per-solver generate budget is large enough to absorb both. Solver-per-worker placement, by contrast, operates at peak per-shard throughput and pays the model load only once per worker. Its sole weakness is idle time from popularity and cost-skew.

Model Name	Size	Hugging Face Checkpoint
Non-reasoning models		
BioLlama (ContactDoctor, 2024)	8B	ContactDoctor/Bio-Medical-Llama-3-8B
DeepSeekMath (Shao et al., 2024)	7B	deepseek-ai/deepseek-math-7b-instruct
Exone (Research, 2024)	7.8B	LGAI-EXAONE/EXAONE-3.5-7.8B-Instruct
Gemma2 (Team, 2024a)	9B	google/gemma-2-9b-it
GLM4 (GLM et al., 2024)	9B	THUDM/glm-4-9b
Granite (IBM Granite Team, 2024)	8B	ibm-granite/granite-3.1-8b-instruct
InternLM3 (Cai et al., 2024)	8B	internlm/internlm3-8b-instruct
Llama3.1 (Grattafiori et al., 2024)	8B	meta-llama/Llama-3.1-8B-Instruct
Mathstral (Mistral AI Team, 2024a)	7B	mistralai/Mathstral-7B-v0.1
Mistral (Mistral AI Team, 2024b)	12B	mistralai/Mistral-Nemo-Instruct-2407
Phi3.5-mini (Abdin et al., 2024)	3.5B	microsoft/Phi-3.5-mini-instruct
Qwen2.5 (Team, 2024b)	7B	Qwen/Qwen2.5-7B-Instruct
Qwen2.5-Coder (Hui et al., 2024)	7B	Qwen/Qwen2.5-Coder-7B-Instruct
Qwen2.5-Math (Yang et al., 2024)	7B	Qwen/Qwen2.5-Math-7B-Instruct
Reasoning models		
LlamaR1 (DeepSeek-AI, 2025)	8B	deepseek-ai/DeepSeek-R1-Distill-Llama-8B
QwenR1 (DeepSeek-AI, 2025)	7B	deepseek-ai/DeepSeek-R1-Distill-Qwen-7B
DLERR1 (Liu et al., 2025)	7B	nvidia/DLER-R1-7B-Research
ThinkerR1 (Guha et al., 2025)	7B	open-thoughts/OpenThinker-7B
StratosR1 (Bespoke Labs, 2025)	7B	bespokelabs/Bespoke-Stratos-7B
SkyworkR1 (He et al., 2025)	7B	Skywork/Skywork-OR1-7B

Table 5: Model pool.