XShare: Collaborative in-Batch Expert Sharing for Faster MoE Inference

Daniil Vankov[*1] **Nikita Ivkin**[2] **Kyle Ulrich**[2] **Xiang Song**[*3] **Ashish Khetan**[2] **George Karypis**[*4]

Abstract

Mixture-of-Experts (MoE) architectures are increasingly used to efficiently scale large language models. However, in production inference, request batching and speculative decoding significantly amplify expert activation, eroding these efficiency benefits. We address this issue by modeling batch-aware expert selection as a modular optimization problem and designing efficient greedy algorithms for different deployment settings. The proposed method, namely `XShare`, requires no retraining and dynamically adapts to each batch by maximizing the total gating score of selected experts. It delivers end-to-end throughput improvements of 10–37% across diverse deployment scenarios (single- or mixed-batch, with or without speculative decoding) while preserving baseline accuracy. On mixed-batch workloads, our method Pareto-dominates the baseline on both throughput and accuracy.

1. Introduction

Mixture-of-Experts (MoE) architectures (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022) have become the dominant paradigm for scaling large language models, decoupling total parameter count from per-token FLOPs by activating only a sparse subset of experts at each token. Recent scaling laws (Abnar et al., 2025) confirm that, at fixed compute, sparse MoE models with 10–100$\times$ more parameters than dense counterparts achieve strictly better quality during *training*.

However, the training-time efficiency gains of MoE architectures do not directly translate to inference-time efficiency. Production serving systems (e.g., Kwon et al. 2023) that batch requests under latency constraints introduce two key challenges. First, to maximize hardware utilization, serving systems batch multiple requests together. While each individual token activates only k experts (typically $k \in \{1, 4, 8\}$), the union of experts activated across a batch grows substantially with batch size. Second, many modern serving systems employ speculative decoding (Leviathan et al., 2023; Li et al., 2025), where a draft model generates candidate tokens verified in parallel by the target model. For a speculative length of s, each request effectively multiplies the batch size by $(s + 1)$: a batch of B requests becomes $B(s + 1)$ tokens processed simultaneously. The combination of request batching and speculative decoding results in expert activation patterns where, for moderate batch sizes, the majority of experts must be loaded into memory, and for large batches, nearly all experts are activated. To quantify this effect, under the simplifying assumption of uniform independent expert selection, the expected number of activated experts in a batch follows $\mathbb{E}[N_a] = N\left(1 - (1 - k/N)^B\right)$, where N is the total number of experts, k is the top-k routing, and B is the effective batch size (including speculative tokens). For DeepSeek-R1 ($N = 256$, $k = 8$), this predicts that at batch size 8, approximately 57 experts are activated, and at batch size 32, approximately 163 experts, closely matching our empirical observations (Figure 1). With speculative decoding of length s, the effective batch becomes $B(s + 1)$, further accelerating expert activation growth. While a sparse MoE model remains computationally more efficient than a dense model with equivalent parameters (only activated experts consume FLOPs), the memory bandwidth bottleneck dominates during the autoregressive decode phase: all activated experts must be loaded from memory regardless of their utilization, making inference memory-IO-bound rather than compute-bound.

Core idea. Our method is motivated by the observation that expert selection should be optimized for the batch as a whole rather than on a per-token basis. We first aggregate gating scores across all tokens and select the top-K experts according to this batch-level utility. This selected subset then constrains per-token routing, from which each token chooses its own top-k experts as depicted in Figure 2. Our approach further extends to production-critical settings: expert-parallel deployments across GPUs and speculative decoding, which existing methods do not address.

[*]Work was done while at Amazon Web Services [1]Arizona State University [2]Amazon Web Services [3]Google [4]University of Minnesota. Correspondence to: Daniil Vankov <dvankov@asu.edu>, Nikita Ivkin <ivkin@amazon.com>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

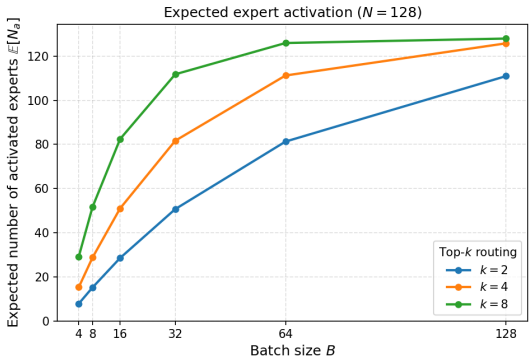


Figure 1. Average number of activated experts

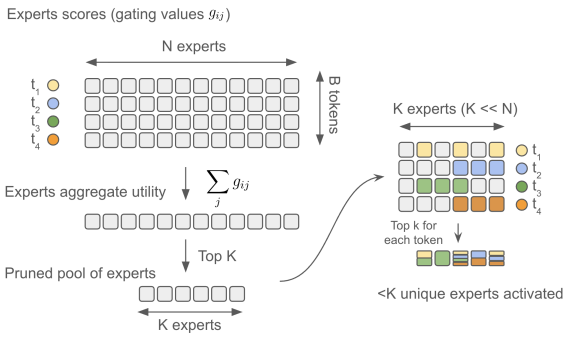


Figure 2. Batch utility expert pruning

Related works. Existing expert-pruning approaches (Liu et al., 2024; Bai et al., 2025; Xie et al., 2024) produce *static* decisions calibrated offline, often requiring retraining or distillation, and cannot adapt to per-batch workload composition or speculative decoding patterns. Such retraining-based methods are orthogonal to our approach and can be combined with it. Closer to our work, Lu et al. (2024)'s Dynamic Skipping prunes per-token below a calibrated gating ratio but ignores batch composition. Lynx (Gupta et al., 2024) first formalized the batch-level problem, but its frequency-based heuristic drops a fixed offline-chosen number of experts and ignores their per-token rank. Concurrent work (Oncescu et al., 2025) builds a shared pool from per-token top-k' picks (similar to our warm-up); this pool is not optimized for batch-level utility.

Contributions. (i) We formulate batch-aware expert selection as a modular optimization with optimal greedy guarantees. (ii) We give a practical, retraining-free three-stage algorithm (warm-up, greedy, refine) that adapts per batch and delivers up to $+23\%$ OTPS at $c \geq 8$ within 1 pt of baseline; under mixed-batch workloads it strictly Pareto-dominates baseline at $c \in \{8, 16, 32, 64\}$. (iii) For speculative decoding, we exploit intra-request expert correlation via a hierarchical per-request variant, gaining up to $+37\%$ OTPS and remaining the unique accuracy-preserving accelerator at high effective batch.

2. Problem Formulation

Preliminaries. For a token hidden state $x \in \mathbb{R}^d$, an MoE layer routes x through N expert FFNs $\{E_i\}_{i=1}^N$ (and optional shared experts $\{E_j^s\}$) via a top-k router with weights W_g: it computes logits $h(x) = W_g x$, picks the indices $\mathcal{T}$ of the k largest entries, and outputs $y = \sum_{i \in \mathcal{T}} g_i(x) E_i(x) + \sum_j E_j^s(x)$, where $g_i(x) = \exp(h_i(x)) / \sum_{j \in \mathcal{T}} \exp(h_j(x))$ for $i \in \mathcal{T}$ and 0 otherwise. With $k \ll N$ (typically $k \in \{1, 4, 8\}$), inference can skip E_i for all $i \notin \mathcal{T}$; efficient MoE inference therefore reduces to identifying $\mathcal{T}$ and *loading* the corresponding expert weights.

Consider a batch $B = \{x_1, \ldots, x_n\}$ of n tokens at the decode phase. At MoE layer $l \in \{1, \ldots, L\}$ the router produces scores $G_i^{(l)} \in \mathbb{R}^N$ for each token x_i, with $g_{i,j}^{(l)}$ the score of expert j. Per-token top-k routing activates k experts, but the batch union $|\bigcup_i \text{TopK}(G_i^{(l)})|$ grows to 62%/95% of all experts at batch sizes 32/64 on DeepSeek-R1 (Sec. 1), creating a memory-IO bottleneck during decode.

Optimization Objective. We seek a small subset $S_l \subseteq E$ at each layer l such that routing tokens to their top-k experts within S_l preserves accuracy:

$$\begin{aligned} \max_{S_1, \ldots, S_L \subseteq E} \quad & \text{Accuracy}\big(f(S_1, \ldots, S_L; B)\big) \\ \text{s.t.} \quad & \sum_{l=1}^{L} |S_l| \leq K, \end{aligned} \tag{1}$$

where $f(S_1, \ldots, S_L; B)$ is the model output when layer l is restricted to S_l, and $K \ll L \cdot N$ is the global budget. The search space is combinatorial ($2^{N \cdot L}$) and evaluating accuracy requires a forward pass, so we use a proxy.

Assumption 2.1 (Router Score Reliability)**.** Higher gating scores $g_{i,j}^{(l)}$ indicate experts that contribute more to model output for token x_i.

This is supported by prior work (Gupta et al., 2024) and verified in Sec. 3. Since each layer was trained to activate the same number of experts, we treat layers in isolation and, under Assumption 2.1, replace (1) by its per-layer proxy:

$$\begin{aligned} \max_{S_l \subseteq E} \quad & f_l(S_l; G^{(l)}) = \sum_{j \in S_l} \sum_{i=1}^{n} g_{i,j}^{(l)} \\ \text{s.t.} \quad & |S_l| \leq m_l. \end{aligned} \tag{2}$$

We allocate budget uniformly $m_l = K/L$.

Modularity. Since $f_l(S; G^{(l)}) = \sum_{j \in S} \sum_{i=1}^{n} g_{i,j}^{(l)}$ is modular in S, the marginal gain of adding expert e equals

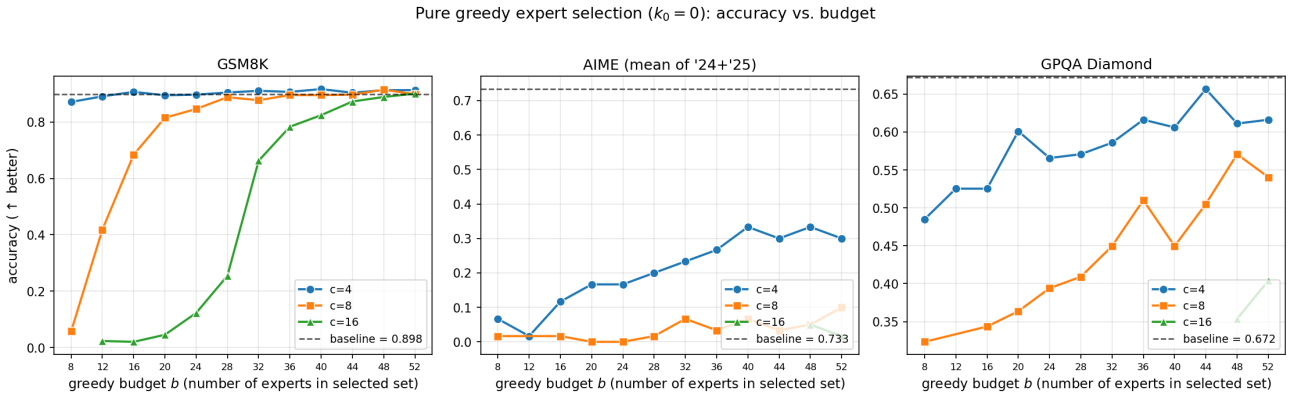


Figure 3. **Pure greedy expert selection (k_0=0, no warm-up) on GPT-OSS-120B.** For each dataset, accuracy increases monotonically with the greedy budget m_l (number of experts in the selected set S_l) and converges to the unpruned baseline (dashed). Curves shown for concurrencies $c \in \{4, 8, 16\}$. Pure greedy without warm-up is most fragile at small m_l in the high-concurrency regime — the selected set may not yet cover the per-row top picks — which motivates pairing greedy selection with a top-k_0 warm-up (Algorithm 2).

$\sum_i g_{i,e}^{(l)}$ regardless of S. Hence selecting the top-m_l experts by aggregate score $\sum_i g_{i,j}^{(l)}$ is *optimal* for (2), and the per-layer optimum yields a globally optimal proxy solution under uniform budget. The extra top-k call is negligible in the memory-bound decode regime.

3. Batch-Aware Expert Selection Algorithm

While the modular argument above gives the optimal proxy solution, a variant of Algorithm 1 with a fixed per-layer budget and warm-up initialization performs better for a more difficult task. We *warm-up* S_0 with the per-token top-k_0 experts so each token's highest-confidence picks are guaranteed, then *refine* by routing each token to its top-k within the selected S_l. Algorithm 2 runs at every MoE layer.

Figure 3 confirms empirically that the pure-greedy variant (k_0=0) on GPT-OSS-120B is monotone in m_l: accuracy on GSM8K, AIME, and GPQA grows with the budget and approaches the unpruned baseline. Higher-concurrency curves shift right — the same budget m_l yields more aggressive in-batch pruning — which motivates pairing greedy selection with the warm-up.

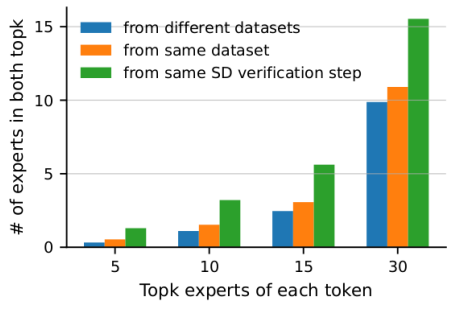


Figure 4. Overlap between the top-k ($k = 5, 10, 15, 30$) experts ($|E_i \cap E_j|$) for (1) two speculative tokens; (2) two tokens from the same dataset; and (3) two tokens from different datasets.

3.1. Speculative Decoding Aware Pruning

Speculative decoding inflates the effective batch from b to $b(1 + L_s)$ tokens, sharpening the expert-activation problem. It also creates an opportunity: tokens from the same request — consecutive steps of one sequence — exhibit much higher overlap in their top-k expert sets than tokens drawn from the same or different datasets (Fig. 4; overlap is 2–3× larger between speculative siblings).

Assumption 3.1 (Intra-Request Correlation)**.** Tokens within the same request exhibit correlated expert preferences: if expert e has high gating scores for token $x_r^{(0)}$, it is likely to have high scores for $x_r^{(1)}, \ldots, x_r^{(L_s)}$ as well.

Hierarchical Proxy. With each request r contributing $(1 + L_s)$ tokens T_r, define the per-request proxy $f_l(S; G^{(l)}; r) = \sum_{x \in T_r} \sum_{j \in S} g_{x,j}^{(l)}$. The batch proxy decomposes as $f_{\text{spec}}(S; G^{(l)}) = \sum_{r \in B} f_l(S; G^{(l)}; r)$. Under Assumption 3.1, optimizing per-request first and then aggregating exploits the shared structure: Algorithm 3 solves the per-request subproblem and Algorithm 4 combines them.

A small per-request budget m_r (typically 4–5) captures most of the within-request gating mass; Sec. 4 shows up to 37% OTPS gain on GPT-OSS-120B at L_s=5 while preserving baseline accuracy.

Algorithm 1 GREEDYSELECT (per layer)

Require: Scores $G^{(l)}$, budget m, init set S_0
$scores[j] \leftarrow \sum_{i=1}^{n} g_{i,j}^{(l)}$ for $j \in E$
return $S_0 \cup \text{Top}_{m-|S_0|}(scores,\ E \setminus S_0)$

Algorithm 2 Batch-Aware Expert Selection (per layer)

Require: Batch B, scores $G^{(l)}$, budget m_l, warm-up k_0
$S_0 \leftarrow \bigcup_{i=1}^{n} \text{Top}_{k_0}(G_i^{(l)})$ # warm-up
$S_l \leftarrow \text{GREEDYSELECT}(G^{(l)}, m_l, S_0)$
Route each $x_i \in B$ via $\text{Top}_k(G_i^{(l)}, S_l)$ # refine

Algorithm 3 PERREQUESTSELECT

Require: Request r with tokens T_r, scores $G^{(l)}$, per-request budget m_r, warm-up k_0
$S_0 \leftarrow \bigcup_{x \in T_r} \text{Top}_{k_0}(G_x^{(l)})$
$scores[j] \leftarrow \sum_{x \in T_r} g_{x,j}^{(l)}$ for $j \in E$
return $S_0 \cup \text{Top}_{m_r - |S_0|}(scores, E \setminus S_0)$

Algorithm 4 Speculative-Decoding-Aware Expert Selection

Require: Batch B, scores $G^{(l)}$, batch budget m, per-request budget m_r, warm-up k_0
$S_{\text{batch}} \leftarrow \bigcup_{r \in B} \text{PERREQUESTSELECT}(r, G^{(l)}, m_r, k_0)$
$S_{\text{batch}} \leftarrow \text{GREEDYSELECT}(G^{(l)}, m, S_{\text{batch}})$
Route each $x_i \in B$ via $\text{Top}_k(G_i^{(l)}, S_{\text{batch}})$

4. Experiments

We evaluate all our methods for GPT-OSS 120B across multiple datasets (AIME2025, AIME2024, GPQA, HumanEval, ShareGPT, GSM8K), and deployment scenarios (autoregressive serving, speculative decoding). For details on experimental setup and baselines refer to Appendix B and for the full list of comprehensive ablation studies refer to Appendix D.

4.1. Scenario 1 — Single GPU, No Spec., Single-Load

On GPT-OSS-120B at $c \in \{4, 8, 16, 32, 64, 128\}$, Algorithm 2 traces a smooth Pareto curve. At $c \in \{16, 32\}$ the warm-up-only (k_0=2, b=0) already gains +20–23% OTPS within 0.5 pt of baseline; for larger c the greedy budget takes over the frontier: (b=28, k_0=2) at c=64 (+14% OTPS, −0.2 pt) and (b=52, k_0=2) at c=128 (+7% OTPS, +0.3 pt). We compare against the concurrent OEA (Oncescu et al., 2025), recovered as Algorithm 2 with b=0.

4.2. Scenario 2 — Single GPU, Spec. Dec., Single-Load

Under Eagle3, at $c \in \{4, 8, 16\}$ (effective batch ≤ 96) the warm-up-only (k_0=1, b=0) already gains +25–37% OTPS within 1 pt of baseline. As the effective batch grows, warm-up alone collapses (−3.2 pt at c=32); the per-request variant (Algorithm 4) takes over as the unique accuracy-preserving accelerator at c=32: (g=4, k_0=2) delivers +7.4% OTPS within 0.5 pt of baseline, while every fixed-budget global config either loses ≥ 1 pt or slows down.

4.3. Scenario 3 — Mixed-Batch, No Spec.

Production traffic is heterogeneous. We model this by reserving $c/4$ slots for graded reasoning prompts (AIME, GPQA, GSM8K) and filling the remaining $3c/4$ with cycling ShareGPT and HumanEval prompts. This is the strongest case for our methods: OEA routes the rare graded tokens through filler-chosen experts, while Algorithm 2 clamps the active set independent of batch composition. (b=12, k_0=2) wins in 12 of 20 cells across $c \in \{4, 8, 16, 32\}$, holding AIME-25 at 0.77–0.83 while OEA k_0=1 collapses below 0.50.

4.4. Scenario 4 — Mixed-Batch with Spec. Dec.

Combining mixed-load with Eagle3 stabilizes Scenario 3's cross-request dilution: every frontier point stays within ±2 pt of baseline at $c \in \{8, 16\}$. At c=8 the per-request (g=5, k_0=1) wins (+13% OTPS, +1.5 pt); at c=16 global greedy (b=12, k_0=1) takes over (+25% OTPS, +1.5 pt). Both strictly Pareto-dominate baseline.

5. Conclusion

We address expert activation explosion in batched MoE inference, where batching and speculative decoding activate 60–95% of experts despite each token using only k. We introduce the first formal framework for batch-aware expert selection, where maximizing gating mass yields a modular objective with optimal greedy solutions. We develop algorithms for three deployment scenarios: standard serving (batch-aware selection with warm-start), speculative decoding (hierarchical intra-request), and expert-parallelism (peak-load minimizing). All operate at inference time without retraining and integrate easily into serving frameworks like vLLM. Without spec. dec., (k_0=2, b=0) delivers +14–23% OTPS at $c \in \{8, 16, 32\}$ within 1 pt of baseline; with Eagle3 this rises to +25–37% at $c \in \{4, 8, 16\}$, and the per-request variant is the unique accuracy-preserving accelerator at c=32. On mixed-load, our fixed-budget greedy strictly Pareto-dominates baseline at $c \in \{8, 16, 32, 64\}$. On DeepSeek-R1 with EP, we cut active experts by 73% and peak GPU load 3× while preserving accuracy.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

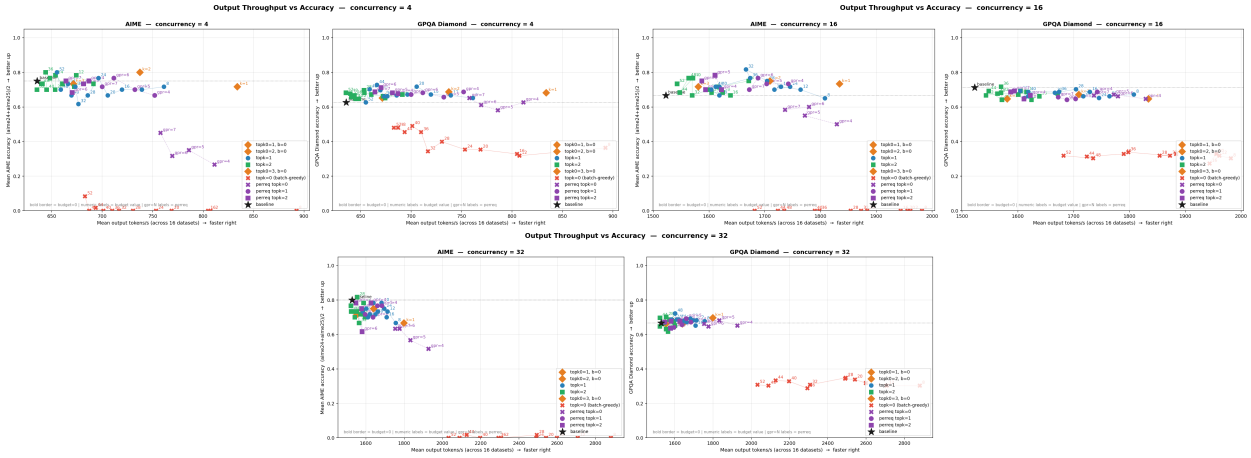


Figure 5. **Scenario 2 (Eagle3 spec. dec., single-load).** OTPS vs. joint AIME∪GPQA accuracy at c=4, c=16, and c=32 (left to right). Spec. dec. inflates the effective batch by $n{+}1$, so the gap to baseline grows with c — at c=32 our methods deliver up to 3.6× speedup (e.g. `global` b=48, k_0=0 on GSM8K) while remaining at or above baseline accuracy. The per-request variant (Algorithm 4) Pareto-dominates the global-only variants in ∼80% of cells. Per-c panels for $c \in \{2, 8\}$ are in Appendix D.

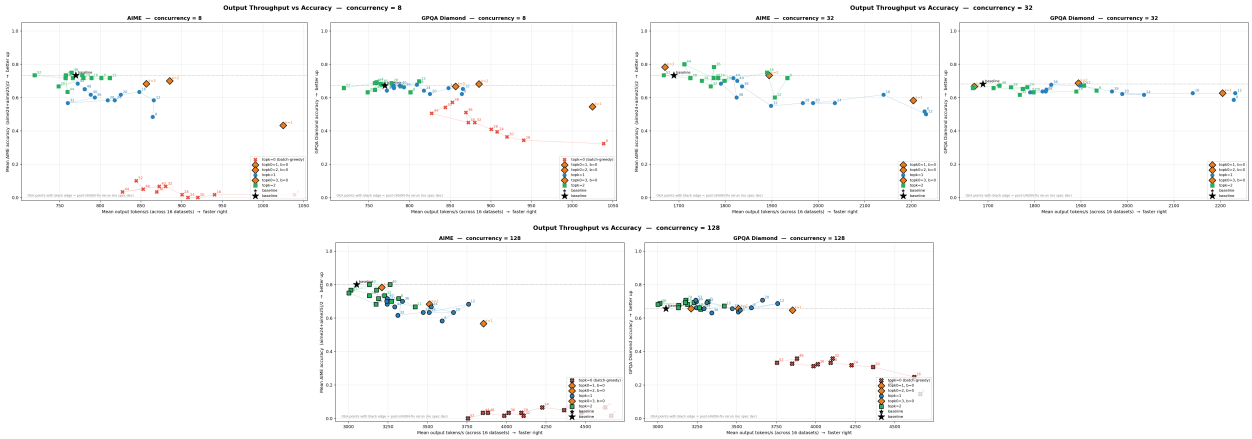


Figure 6. **Scenario 1 (no spec. dec., single-load).** OTPS vs. accuracy on AIME and GPQA at c=8, c=32, and c=128 (left to right) for GPT-OSS 120B on a single H200. Each point is one expert-pruning configuration; the red star is the unpruned baseline. The accuracy-preserving Pareto edge widens at larger c: at c=128 several pruning configurations gain $\geq 30\%$ OTPS over baseline while staying within 1% accuracy. The full per-c panels (including c=4, 16, 64) are in Appendix D.

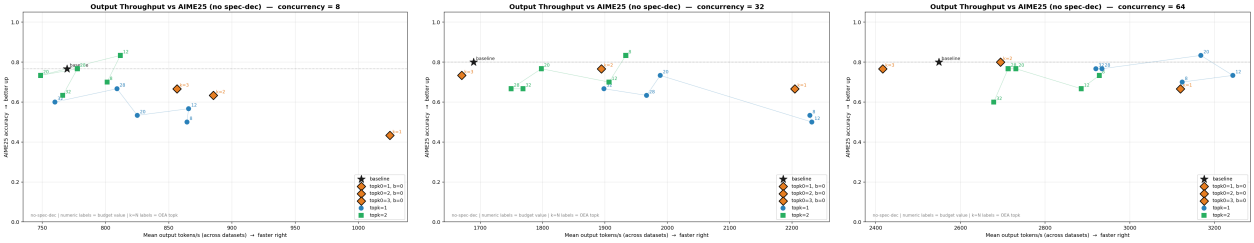


Figure 7. **Scenario 3 (no spec. dec., mixed-load).** OTPS vs. AIME-25 accuracy at c=8, c=32, and c=64 (left to right). The mixed-load regime exposes the failure mode of OEA-style aggressive pruning: filler tokens dominate the active set and reasoning accuracy collapses, whereas fixed-budget pruning clamps the active set and preserves accuracy. The advantage grows with c — at c=64 our (b=12, k_0=2) retains AIME-25 accuracy while OEA k_0=1 falls below 0.40. Per-c panels for $c \in \{4, 16\}$ are in Appendix D.

References

Abnar, S., Shah, H., Busbridge, D., Ali, A. M. E., Susskind, J., and Thilak, V. Parameters vs flops: Scaling laws for optimal sparsity for mixture-of-experts language models. *arXiv preprint arXiv:2501.12370*, 2025.

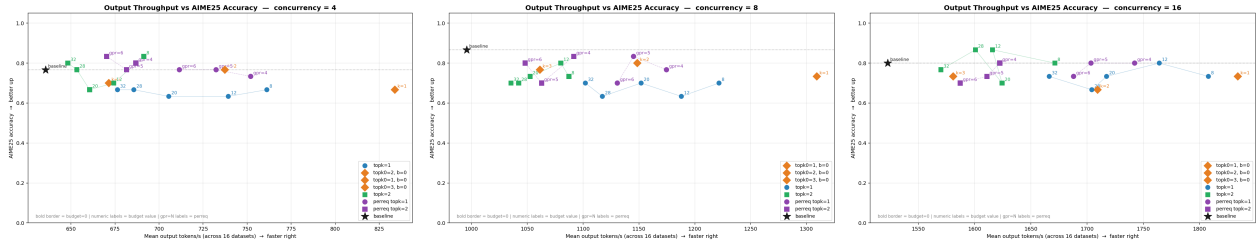


Figure 8. **Scenario 4 (Eagle3 spec. dec., mixed-load).** OTPS vs. AIME-25 accuracy at c=4, c=8, and c=16 (left to right). Combining speculative decoding with cross-request workload mixing reproduces the Scenario 3 ordering, with our fixed-budget greedy retaining its Pareto edge. The effective batch is $6c$ tokens per step so c=16 already corresponds to a 96-token MoE call — the regime where pruning's memory-IO savings dominate. Per-c panel for c=2 is in Appendix D.

Bai, S., Li, H., Zhang, J., Hong, Z., and Guo, S. Diep: Adaptive mixture-of-experts compression through differentiable expert pruning. *arXiv preprint arXiv:2509.16105*, 2025.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Gupta, V., Sinha, K., Gavrilovska, A., and Iyer, A. P. Lynx: Enabling efficient moe inference through dynamic batch-aware expert selection. *arXiv preprint arXiv:2411.08982*, 2024.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Li, Y., Wei, F., Zhang, C., and Zhang, H. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Annual Conference on Neural Information Processing Systems*, 2025.

Liu, E., Zhu, J., Lin, Z., Ning, X., Blaschko, M. B., Yan, S., Dai, G., Yang, H., and Wang, Y. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*, 2024.

Lu, X., Liu, Q., Xu, Y., Zhou, A., Huang, S., Zhang, B., Yan, J., and Li, H. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. *arXiv preprint arXiv:2402.14800*, 2024.

Oncescu, C.-A., Wu, Q., Chung, W. T., Wu, R., Gopal, B., Wang, J., Dao, T., and Athiwaratkun, B. Opportunistic expert activation: Batch-aware expert routing for faster decode without retraining, 2025. URL https://arxiv.org/abs/2511.02237.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

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

A. Experts Selection for Multi-GPU Inference

Expert parallelism (EP) is another promising technique to scale MoE model training across multiple GPUs. Recently, LLM serving systems such as vLLM and SGLang integrated EP into their platform to accelerate inference on multiple GPUs. In this approach, experts are partitioned across G GPU groups, with each group g hosting a subset of experts $E_g \subseteq E$. While expert parallelism enables serving large MoE models that exceed single-GPU memory, it introduces a load balancing challenge: the inference latency for each layer is determined by the GPU with the largest number of activated experts, as all GPUs must synchronize after processing.

However, expert parallelism also presents an optimization opportunity: by considering the GPU placement of experts during selection, we can balance the load across GPUs while maintaining model quality. This is particularly important for production deployments where consistent latency is critical.

A.1. GPU-Aware Proxy Function

Consider a system with G GPU groups, where GPU group g hosts experts E_g. The experts form a partition: $E = \bigcup_{g=1}^{G} E_g$ with $E_g \cap E_{g'} = \emptyset$ for $g \neq g'$. For a batch B with router scores $G^{(l)}$ at layer l, we define the per-GPU load for subset S: $\text{Load}_g(S) = |S \cap E_g|$, which counts the number of selected experts on GPU g. The bottleneck load is: $\text{MaxLoad}(S) = \max_{g \in \{1,\dots,G\}} \text{Load}_g(S)$. We maximize our proxy $f(S; G^{(l)})$ while minimizing the bottleneck load, thus GPU-aware optimization problem is:

$$\max_{S \subseteq E} f(S; G^{(l)}) \quad \text{s.t.} \quad \text{MaxLoad}(S) \leq m_{\text{gpu}}, \tag{3}$$

where m_{gpu} is the per-GPU budget. Standard greedy selection (Algorithm 1) can overload a few GPUs with high-scoring experts, while GPU-aware selection trades off gating score to achieve balanced load and lower latency. We modify our greedy approach to select experts per GPU. At each iteration, instead of selecting the globally best expert, we select the best expert for each GPU group in turn. This ensures that no single GPU accumulates disproportionately many experts. Based on this insight, we propose Algorithm 5 which performs GPU-balanced greedy selection and Algorithm 6 which combines warm-up initialization with GPU-aware optimization.

Algorithm 5 enforces balanced load by design: after G iterations, each GPU has at most one more expert than any other GPU, ensuring $\text{MaxLoad}(S) \leq \lceil |S|/G \rceil$. As previously, it still prioritizes high-scoring experts within each GPU, thus maintaining quality.

A.2. Experiments for Expert Parallelism Scenario

In Table 1 our EP-aware algorithm (Algorithm 6 with $(k_0 = 1, m_g = 5)$) for DeepSeek-R1 achieves a 73% drop in total activate experts while maintaining accuracy within 1% of baseline. It effectively reduces the peak GPU load by 3× (25.6 → 8.6 experts), directly translating to improved memory efficiency and reduced communication overhead.

B. Experimental Setup

Hardware and serving. All single-GPU experiments serve the model on one NVIDIA H200 (141 GB HBM3e). The full run is parallelised across 8 H200s, with one independent vLLM container per GPU (one server per device, no tensor or pipeline parallelism), so reported throughput numbers are per-server and unaffected by cross-device communication. The expert-parallelism experiments (Section A) use 8 H200s with EP=8.

Models. Our primary model is GPT-OSS-120B (128 experts, 4 active per token). For the expert-parallelism setting we additionally evaluate DeepSeek-R1. Each model is served with vLLM v1, `-max-model-len=16384`, and the OpenAI-Harmony reasoning parser; the speculative-decoding configurations use the public Eagle3 draft head (`nvidia/gpt-oss-120b-Eagle3-short-context`) with 5 draft tokens per step.

Efficient kernel. Our pruning logic is implemented as a drop-in patch to vLLM's fused triton MoE kernel (`gpt_oss_triton_kernels_moe.py`). Construction of the keep-mask, both for the global-greedy (Algorithm 2) and the per-request speculative variant (Algorithm 4), is fused into the routing path: a single $[E]$ boolean buffer is built from the union of warm-up and greedy contributions, then masked into the gating logits before `matmul_ogs`. The buffer has fixed shape, so the kernel remains CUDA-graph compatible and incurs no extra synchronisation. We measure all throughput numbers with this fused implementation served from inside vLLM.

Algorithm 5 GPU-Aware Greedy Expert Selection

Require: Experts per GPU $E_1, \ldots, E_G$, router scores $G^{(l)}$,
per-GPU budget m_g, initial subset S_0
$S \leftarrow S_0$
while $|S| < m_g \cdot G$ **do**
for $g \in 1, \ldots, G$ **do**
Best expert on GPU g
$e^* \leftarrow \arg\max_{E_g \setminus S}[f(S \cup \{e\}; G^{(l)}) - f(S; G^{(l)})]$
$S \leftarrow S \cup \{e^*\}$
end for
end while
return S

Algorithm 6 Expert Parallelism Aware Selection

Require: Experts per GPU $E_1, \ldots, E_G$, router scores $G^{(l)}$,
batch B, per-GPU budget m_g, warm-up k_0
Warm-up: top-k_0 per token
$S_0 \leftarrow \bigcup_{x \in B} \text{Top}_{k_0}(G_x^{(l)})$
Calling Algorithm 5
$S \leftarrow \text{GPUAWAREGREEDY}(E_1, \ldots, E_G, G^{(l)}, m_g, S_0)$
for each token $x_i \in B$ **do**
Refinement: top-k within selected set
Route x_i to $\text{Top}_k(G_i^{(l)}, S)$
end for
return S

Throughput measurement. Speed is measured via `vllm bench serve` against an OpenAI-compatible chat-completions endpoint at concurrencies $c \in \{2, 4, 8, 16, 32, 64, 128\}$ (the spec-dec run tops out at c=32 since the effective batch is $c \cdot (n{+}1)$ tokens per step). Each cell sends a fixed input pool (e.g. ShareGPT-1k, GSM8K-1k, HumanEval-Python-1k) and reports the steady-state OTPS (*output tokens per second*) returned by the bench harness. Reported figures are means over the entire pool.

Accuracy measurement. Accuracy is collected with the `lm-evaluation-harness` in OpenAI-API mode, against the same vLLM endpoint. We report the canonical metric per task: *exact-match flexible-extract* for GSM8K and AIME (we never use strict-match, which is a format artifact under reasoning-style outputs), and *exact-match* for GPQA after a regex-based answer extraction pass. AIME numbers use the union of AIME-2024 and AIME-2025 (n=30 each) so deltas below ∼5 pt are within the per-set standard error.

Workload regimes. We evaluate each method along two orthogonal axes: *speculative decoding* (off vs. Eagle3 with n=5) and *batch composition* (single-load: every request comes from the same dataset; mixed-load: $c/4$ reasoning slots interleaved with $3c/4$ ShareGPT and HumanEval filler slots that compete for the same expert budget). The four combinations give scenarios S1–S4 used in the next subsections.

C. Activated Experts Under Pruning

The accuracy/throughput trade-off ultimately follows from how aggressively each method shrinks the active expert set. We instrument the patched MoE kernel to log, on every layer call, both the number of *kept* experts (the size of the keep-mask) and the number of *activated* experts (unique experts hit by the post-mask top-k routing across all tokens in the step). The mean over decode steps at full batch (M=c for non-spec, M=6c for spec. dec.) is reported in Tables 2–7 and plotted against measured OTPS in Figure 9. Observed pattern:

- Baseline routes ∼24–36 experts per layer call at c=8 (non-spec) and ∼40–56 at c=8 (spec. dec.); pruned configurations cut this 2–6×.

Method	Accuracy	# Experts	Max/GPU
GSM-8K (Batch Size 8)			
Original	0.956	50.7	11.3
Algorithm 6 $(1, 5)$	0.946	33.2	5.94
IFeval (Batch Size 16)			
Original	0.697	160.4	25.6
Algorithm 6 $(1, 5)$	0.696	43.4	8.64

Table 1. DS-R1 accuracy and per-GPU load (batch size 8 and 16).

- For OEA (warm-up only, b=0), activated$\approx$kept since the mask is constructed exactly from the per-row top-k_0.
- For our fixed-budget greedy with budget $b \geq 16$, activated$<$kept saturates around $\sim$18–28: even when the mask permits 50+ experts, the model's natural top-4 routing only fires $\sim$20.
- Per-request variants (Algorithm 4) activate slightly more experts than global-greedy at matched budget, consistent with the diversity benefit they exploit in Scenario 2.

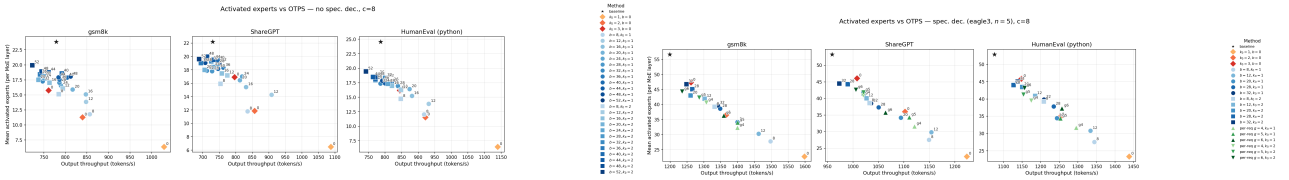


Figure 9. **Mean activated experts vs. OTPS** at c=8 on three single-load datasets, without (left) and with (right) Eagle3 speculative decoding. Each dot is one method labelled by its budget b; marker shape encodes k_0 (circle/square) and per-request variant (triangle). Lower-left is better activation efficiency, lower-right is the Pareto frontier of throughput. Full per-concurrency plots and the LaTeX tables appear in Appendix D.

D. Additional experiments artifacts

D.1. Per-scenario OTPS-accuracy plots and tables

The plots and tables below report per-task OTPS (output tokens/s) and accuracy across all methods that appear in the corresponding scenario figure, at every concurrency we measured. Plots replicate the per-c OTPS-vs-accuracy figures (one per task or task-pair) so the appendix exposes the data behind the main-text figures. Columns are per-task OTPS and the avg. Δ % OTPS over baseline, right columns are per-task accuracy plus the avg. accuracy and avg. Δ acc over baseline. Method labels in **bold** mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). $k_0 = 0$ batch-greedy configs that collapse on at least one dataset are excluded from the tables.

Table 2. **Mean activated experts** (per MoE layer call, full-batch decode) on gsm8k, no spec. dec. , varying batch concurrency c.

Method	c=4	c=8	c=16
baseline	14.07	23.86	36.53
k_0=1, b=0	4.00	6.36	10.29
k_0=2, b=0	6.62	11.27	18.46
k_0=3, b=0	9.59	15.77	25.41
b=8, k_0=1	9.14	11.78	15.72
b=12, k_0=1	10.50	13.85	18.67
b=16, k_0=1	11.02	15.11	20.43
b=20, k_0=1	11.06	15.90	21.89
b=24, k_0=1	11.32	16.55	22.67
b=28, k_0=1	11.87	17.03	23.85
b=32, k_0=1	12.07	17.26	24.01
b=36, k_0=1	12.13	17.52	24.89
b=40, k_0=1	12.33	17.91	25.35
b=44, k_0=1	12.17	17.94	26.26
b=48, k_0=1	12.24	18.06	25.81
b=52, k_0=1	12.13	18.39	26.35
b=8, k_0=2	10.59	15.15	21.84
b=12, k_0=2	11.30	16.09	23.89
b=16, k_0=2	11.60	17.04	24.34
b=20, k_0=2	11.87	17.53	25.52
b=24, k_0=2	11.94	17.77	26.26
b=28, k_0=2	11.93	18.11	27.12
b=32, k_0=2	12.27	18.33	27.20
b=36, k_0=2	12.09	18.45	27.08
b=40, k_0=2	12.26	18.59	27.50
b=44, k_0=2	12.29	18.78	27.94
b=48, k_0=2	12.65	18.91	27.89
b=52, k_0=2	12.51	19.96	28.11

D.1.1. SCENARIO 1 — NO SPEC. DEC., SINGLE-LOAD

Table 3. **Mean activated experts** (per MoE layer call, full-batch decode) on gsm8k, spec. dec. (eagle3, $n = 5$), varying batch concurrency c.

Method	c=4	c=8
baseline	41.36	56.70
k_0=1, b=0	15.25	22.77
k_0=2, b=0	26.25	36.59
k_0=3, b=0	34.47	47.35
b=8, k_0=1	20.03	27.81
b=12, k_0=1	22.86	30.32
b=20, k_0=1	26.76	34.18
b=28, k_0=1	28.80	38.63
b=32, k_0=1	30.74	39.72
b=8, k_0=2	28.23	39.27
b=12, k_0=2	30.43	41.97
b=20, k_0=2	31.16	43.05
b=28, k_0=2	33.00	45.16
b=32, k_0=2	34.59	46.81
per-req g=4, k_0=1	21.33	32.29
per-req g=5, k_0=1	23.12	34.03
per-req g=6, k_0=1	24.93	36.36
per-req g=4, k_0=2	28.07	40.69
per-req g=5, k_0=2	29.66	42.32
per-req g=6, k_0=2	30.59	44.33

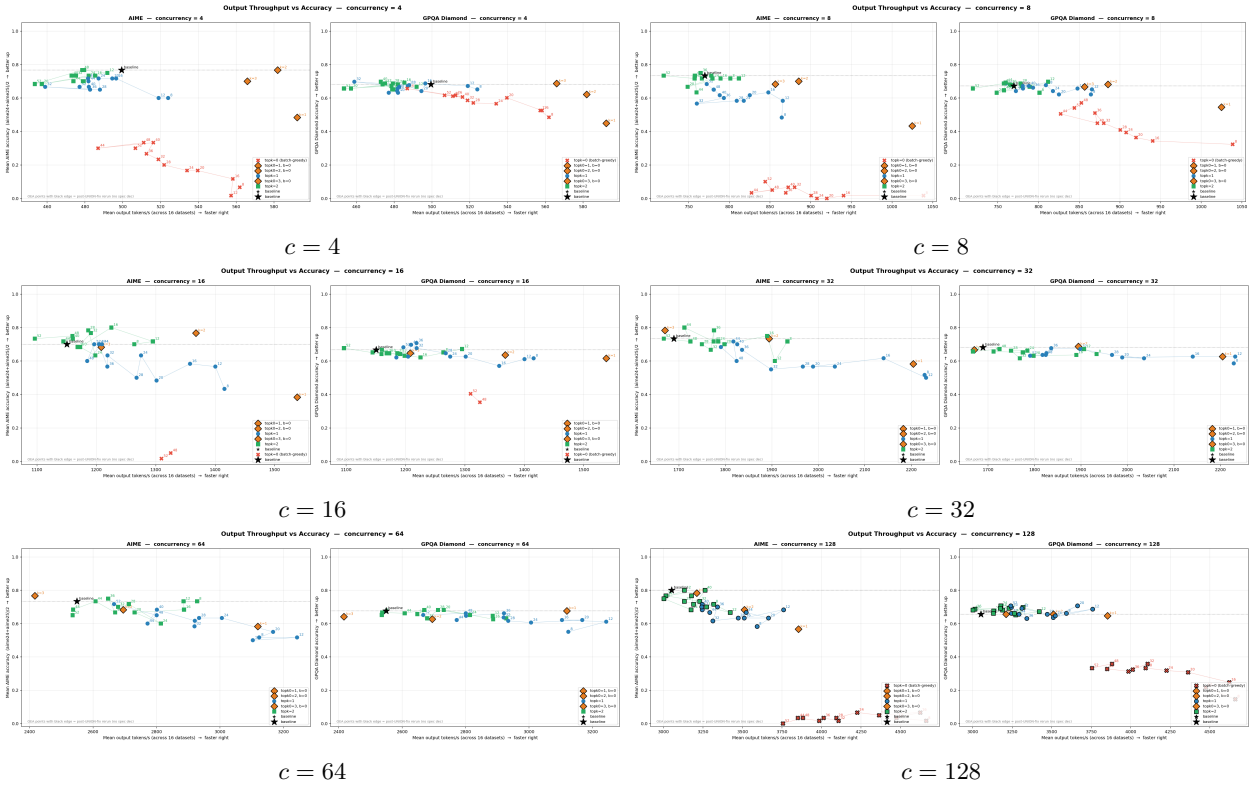


Figure 10. **Scenario S1 (no spec. dec., single-load) — OTPS vs. accuracy on AIME∪GPQA, all concurrencies.** Each panel is one concurrency c. Markers: orange diamonds = OEA (b=0); blue/green = global pruning at k_0=1/2; red X = k_0=0 batch-greedy; purple = per-request (when present); black star = unpruned baseline. Numeric labels next to each point are the budget b (or k_0 for OEA, g for per-request).

Table 4. **Mean activated experts** (per MoE layer call, full-batch decode) on ShareGPT, no spec. dec. , varying batch concurrency c.

Method	c=4	c=8	c=16
baseline	13.32	22.20	33.63
k_0=1, b=0	4.00	6.50	11.16
k_0=2, b=0	6.82	11.86	19.28
k_0=3, b=0	9.95	16.90	26.12
b=8, k_0=1	9.12	11.81	16.00
b=12, k_0=1	10.44	14.27	18.62
b=16, k_0=1	11.05	15.43	20.52
b=20, k_0=1	11.41	16.44	22.01
b=24, k_0=1	11.37	17.00	23.14
b=28, k_0=1	11.96	17.80	24.10
b=32, k_0=1	12.21	17.86	24.54
b=36, k_0=1	12.16	18.32	25.46
b=40, k_0=1	12.23	18.16	25.79
b=44, k_0=1	12.29	18.78	26.15
b=48, k_0=1	12.48	19.06	26.64
b=52, k_0=1	12.39	19.09	27.21
b=8, k_0=2	10.83	15.93	22.70
b=12, k_0=2	11.51	17.19	23.70
b=16, k_0=2	11.95	17.47	25.40
b=20, k_0=2	12.27	18.01	25.11
b=24, k_0=2	12.21	18.44	26.57
b=28, k_0=2	12.46	18.89	26.18
b=32, k_0=2	12.36	19.30	26.70
b=36, k_0=2	12.64	19.02	27.96
b=40, k_0=2	12.63	19.40	28.21
b=44, k_0=2	12.63	19.52	28.44
b=48, k_0=2	12.62	19.99	28.86
b=52, k_0=2	12.76	19.60	29.42

Table 5. **Mean activated experts** (per MoE layer call, full-batch decode) on ShareGPT, spec. dec. (eagle3, $n = 5$), varying batch concurrency c.

Method	**c=4**	**c=8**
baseline	38.36	53.30
k_0=1, b=0	14.11	22.54
k_0=2, b=0	23.83	36.00
k_0=3, b=0	31.13	46.12
b=8, k_0=1	19.44	27.53
b=12, k_0=1	22.20	29.82
b=20, k_0=1	25.39	34.19
b=28, k_0=1	27.47	37.33
b=32, k_0=1	29.51	38.53
b=8, k_0=2	27.56	38.62
b=12, k_0=2	28.02	40.09
b=20, k_0=2	29.36	41.55
b=28, k_0=2	32.15	44.31
b=32, k_0=2	31.45	44.52
per-req g=4, k_0=1	20.78	31.61
per-req g=5, k_0=1	22.75	34.34
per-req g=6, k_0=1	23.89	35.75
per-req g=4, k_0=2	26.98	40.92
per-req g=5, k_0=2	28.41	41.86
per-req g=6, k_0=2	29.49	42.65

Table 6. **Mean activated experts** (per MoE layer call, full-batch decode) on HumanEval-Py, no spec. dec. , varying batch concurrency c.

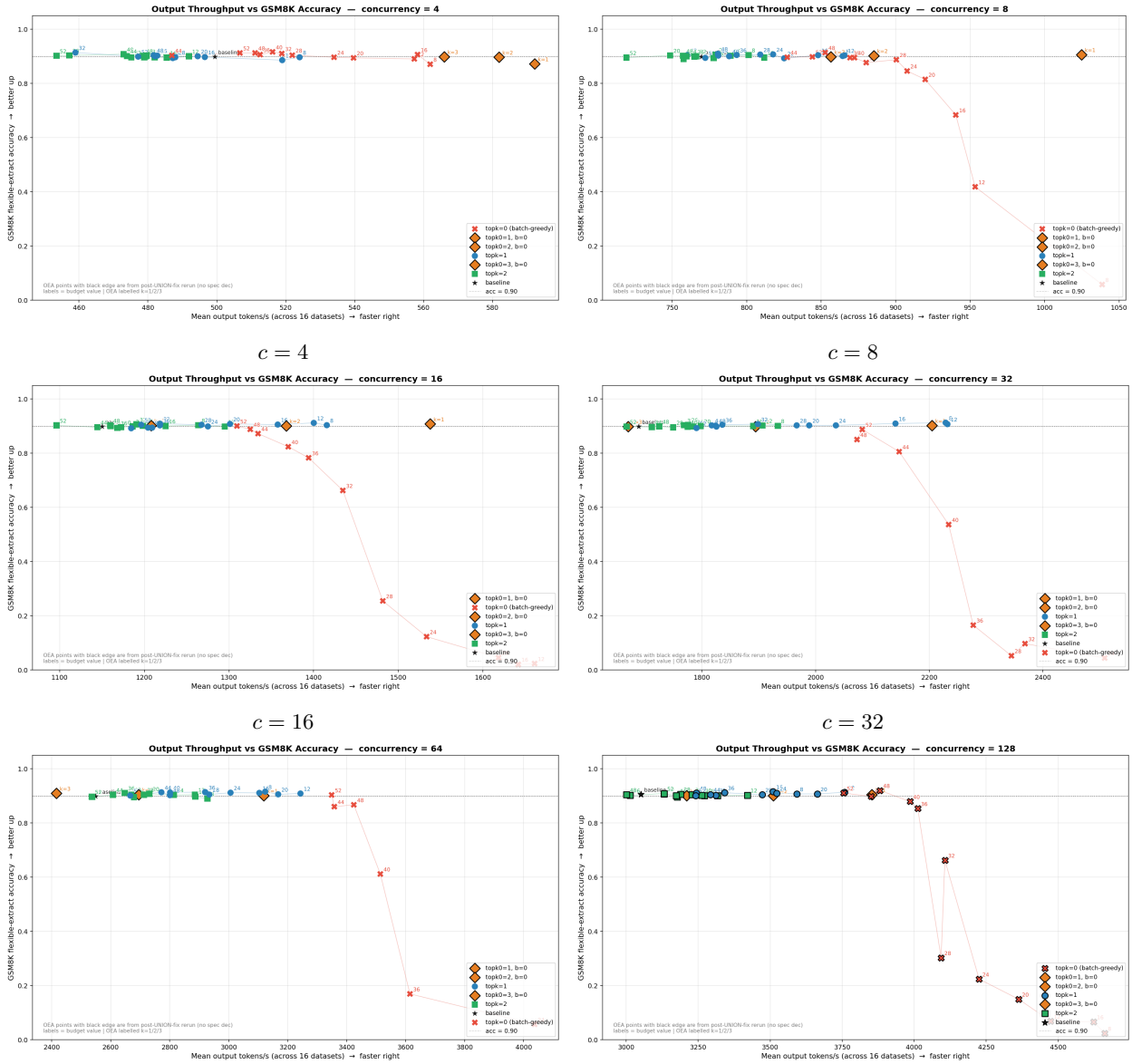
Method	c=4	c=8	c=16
baseline	14.02	24.58	39.83
k_0=1, b=0	4.00	6.45	11.04
k_0=2, b=0	6.88	11.56	18.66
k_0=3, b=0	9.61	16.20	25.05
b=8, k_0=1	9.24	12.05	16.25
b=12, k_0=1	10.34	13.87	18.43
b=16, k_0=1	11.06	15.23	20.12
b=20, k_0=1	11.33	16.43	20.99
b=24, k_0=1	11.41	16.40	22.50
b=28, k_0=1	11.83	16.96	23.64
b=32, k_0=1	11.51	17.38	23.77
b=36, k_0=1	11.52	17.58	24.18
b=40, k_0=1	11.75	17.43	24.68
b=44, k_0=1	11.72	17.40	25.44
b=48, k_0=1	11.72	17.87	26.04
b=52, k_0=1	12.18	18.22	26.23
b=8, k_0=2	10.63	14.73	21.88
b=12, k_0=2	11.04	16.25	23.34
b=16, k_0=2	11.60	16.98	23.60
b=20, k_0=2	11.76	17.95	25.01
b=24, k_0=2	11.39	17.35	25.53
b=28, k_0=2	11.74	18.25	25.74
b=32, k_0=2	12.10	18.36	26.82
b=36, k_0=2	11.81	18.82	26.54
b=40, k_0=2	12.25	18.09	26.47
b=44, k_0=2	12.11	18.52	26.88
b=48, k_0=2	12.15	18.55	26.80
b=52, k_0=2	12.14	19.47	27.20

Table 7. **Mean activated experts** (per MoE layer call, full-batch decode) on HumanEval-Py, spec. dec. (eagle3, $n = 5$), varying batch concurrency c.

Method	**c=4**	**c=8**
baseline	38.93	52.94
k_0=1, b=0	14.55	23.37
k_0=2, b=0	24.82	34.72
k_0=3, b=0	32.80	45.88
b=8, k_0=1	20.01	27.55
b=12, k_0=1	22.39	30.78
b=20, k_0=1	25.89	34.44
b=28, k_0=1	28.38	37.84
b=32, k_0=1	30.00	39.83
b=8, k_0=2	27.84	39.31
b=12, k_0=2	29.36	40.91
b=20, k_0=2	30.88	43.48
b=28, k_0=2	32.75	44.08
b=32, k_0=2	34.10	45.06
per-req g=4, k_0=1	20.66	31.69
per-req g=5, k_0=1	22.80	34.38
per-req g=6, k_0=1	24.51	37.31
per-req g=4, k_0=2	27.35	39.55
per-req g=5, k_0=2	29.25	41.28
per-req g=6, k_0=2	30.19	43.21

Method	**OTPS (tokens/s)** AIME	GPQA	GSM8K	**avg.** Δ % OTPS	**Accuracy (%)** AIME	GPQA	GSM8K	**avg.** acc. (%)	**avg.** Δ acc
baseline	**520**	**501**	**504**		**76.7**	**68.2**	**89.8**	**78.2**	
k_0=1, b=0	**647**	**617**	**581**	**+21.1%**	**48.3**	**44.9**	**87.3**	**60.2**	**-18.0**
k_0=2, b=0	**639**	**605**	**577**	**+19.5%**	**76.7**	**62.1**	**89.7**	**76.2**	**-2.1**
k_0=3, b=0	**596**	**566**	**574**	**+13.9%**	**70.0**	**68.7**	**89.9**	**76.2**	**-2.0**
b=8, k_0=1	545	520	526	+4.4%	60.0	65.2	89.7	71.6	-6.6
b=12, k_0=1	530	513	525	+2.9%	60.0	67.2	88.6	71.9	-6.3
b=16, k_0=1	515	498	495	-1.0%	71.7	68.7	89.8	76.7	-1.5
b=20, k_0=1	510	487	500	-1.8%	71.7	64.1	90.1	75.3	-2.9
b=24, k_0=1	508	490	498	-1.8%	71.7	67.2	89.4	76.1	-2.2
b=28, k_0=1	505	489	486	-3.0%	65.0	67.7	89.7	74.1	-4.1
b=32, k_0=1	474	453	462	-8.8%	66.7	69.7	91.4	75.9	-2.3
b=36, k_0=1	501	483	485	-3.6%	65.0	66.7	90.3	74.0	-4.2
b=40, k_0=1	501	482	485	-3.7%	66.7	63.1	90.1	73.3	-4.9
b=44, k_0=1	499	479	481	-4.3%	70.0	65.7	90.3	75.3	-2.9
b=48, k_0=1	502	480	485	-3.7%	71.7	64.1	90.4	75.4	-2.8
b=52, k_0=1	498	483	479	-4.2%	66.7	63.1	89.9	73.2	-5.0
b=8, k_0=2	494	477	481	-4.8%	76.7	65.2	90.3	77.4	-0.9
b=12, k_0=2	505	490	490	-2.5%	75.0	66.7	90.1	77.2	-1.0
b=16, k_0=2	503	482	490	-3.2%	73.3	69.2	89.5	77.4	-0.9
b=20, k_0=2	472	450	464	-9.1%	68.3	65.7	90.4	74.8	-3.4
b=24, k_0=2	499	477	485	-4.2%	73.3	68.2	89.6	77.0	-1.2
b=28, k_0=2	498	483	481	-4.1%	70.0	68.7	90.1	76.3	-2.0
b=32, k_0=2	495	472	489	-4.5%	73.3	68.7	89.6	77.2	-1.0
b=36, k_0=2	496	483	484	-4.0%	70.0	65.2	89.7	74.9	-3.3
b=40, k_0=2	496	482	481	-4.3%	73.3	69.7	90.8	78.0	-0.3
b=44, k_0=2	497	482	484	-4.0%	70.0	68.2	90.1	76.1	-2.1
b=48, k_0=2	495	479	476	-4.9%	76.7	67.2	90.2	78.0	-0.2
b=52, k_0=2	467	454	456	-9.6%	68.3	65.7	90.2	74.7	-3.5

Table 8. **Scenario S1 (no spec. dec., single-load) at concurrency c=4.** All methods that appear in Figure 6 and the corresponding per-c figures in `figs/full_post_oea_rerun/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).



$c = 4$ $c = 8$ $c = 16$ $c = 32$ $c = 64$ $c = 128$

Figure 11. **Scenario S1 (no spec. dec., single-load) — OTPS vs. GSM8K accuracy, all concurrencies.** Each panel is one concurrency c. Markers: orange diamonds = OEA (b=0); blue/green = global pruning at k_0=1/2; red X = k_0=0 batch-greedy; purple = per-request (when present); black star = unpruned baseline. Numeric labels next to each point are the budget b (or k_0 for OEA, g for per-request).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	800	722	779		73.3	67.2	89.8	76.8	
k_0=1, b=0	**1109**	**969**	**993**	**+33.5%**	**43.3**	**54.5**	**90.7**	**62.9**	**-13.9**
k_0=2, b=0	**894**	**842**	**883**	**+13.9%**	**70.0**	**68.2**	**90.3**	**76.2**	**-0.6**
k_0=3, b=0	918	804	846	+11.7%	68.3	66.7	89.9	75.0	-1.8
b=8, k_0=1	888	806	856	+10.8%	48.3	62.1	90.1	66.9	-9.9
b=12, k_0=1	899	807	848	+11.0%	58.3	65.2	90.4	71.3	-5.5
b=16, k_0=1	872	852	847	+11.8%	63.3	65.7	90.4	73.1	-3.6
b=20, k_0=1	810	729	816	+2.4%	61.7	62.1	89.3	71.0	-5.7
b=24, k_0=1	838	817	789	+6.3%	58.3	64.1	90.8	71.1	-5.7
b=28, k_0=1	833	801	786	+5.2%	58.3	67.7	90.7	72.2	-4.6
b=32, k_0=1	777	674	747	-4.4%	56.7	68.7	90.4	71.9	-4.8
b=36, k_0=1	812	789	786	+3.8%	60.0	66.2	90.5	72.2	-4.6
b=40, k_0=1	804	734	804	+1.8%	61.7	67.2	90.1	73.0	-3.8
b=44, k_0=1	799	774	784	+2.5%	65.0	65.7	90.3	73.7	-3.1
b=48, k_0=1	802	694	812	+0.4%	65.0	67.2	91.0	74.4	-2.4
b=52, k_0=1	790	685	741	-3.6%	68.3	64.1	89.5	74.0	-2.8
b=8, k_0=2	787	758	784	+1.3%	71.7	63.1	90.5	75.1	-1.7
b=12, k_0=2	**793**	**813**	**794**	**+4.3%**	**71.7**	**69.7**	**89.5**	**77.0**	**+0.2**
b=16, k_0=2	775	701	763	-2.6%	71.7	66.7	90.2	76.2	-0.6
b=20, k_0=2	730	704	737	-5.6%	66.7	63.1	90.4	73.4	-3.4
b=24, k_0=2	757	778	751	-0.6%	71.7	67.2	90.6	76.5	-0.3
b=28, k_0=2	**792**	**774**	**749**	**+0.6%**	**73.3**	**68.7**	**89.4**	**77.1**	**+0.4**
b=32, k_0=2	786	761	741	-0.5%	71.7	67.7	90.1	76.5	-0.3
b=36, k_0=2	**780**	**756**	**741**	**-1.0%**	**75.0**	**68.2**	**89.9**	**77.7**	**+0.9**
b=40, k_0=2	778	717	791	-0.6%	73.3	64.6	89.1	75.7	-1.1
b=44, k_0=2	738	712	761	-3.9%	63.3	69.2	90.1	74.2	-2.6
b=48, k_0=2	777	706	744	-3.1%	71.7	68.7	90.2	76.9	+0.1
b=52, k_0=2	705	720	723	-6.6%	73.3	65.7	89.5	76.2	-0.6

Table 9. **Scenario S1 (no spec. dec., single-load) at concurrency c=8.** All methods that appear in Figure 6 and the corresponding per-c figures in `figs/full_post_oea_rerun/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	1098	1012	1222		70.0	66.7	89.8	75.5	
k_0=1, b=0	**1468**	**1162**	**1788**	**+32.6%**	**38.3**	**61.6**	**90.8**	**63.6**	**-11.9**
k_0=2, b=0	**1437**	**1151**	**1515**	**+23.1%**	**76.7**	**63.6**	**90.1**	**76.8**	**+1.3**
k_0=3, b=0	1243	960	1242	+3.4%	68.3	64.6	90.2	74.4	-1.1
b=8, k_0=0	2097	1543	2010	+69.6%	–	–	–	–	–
b=8, k_0=1	1296	1143	1498	+18.2%	43.3	61.6	90.4	65.1	-10.4
b=12, k_0=1	1379	1131	1529	+21.2%	56.7	61.1	91.2	69.7	-5.8
b=16, k_0=1	1311	1197	1510	+20.6%	58.3	57.1	90.6	68.7	-6.8
b=20, k_0=1	1275	1073	1404	+12.6%	48.3	62.6	90.8	67.3	-8.2
b=24, k_0=1	1222	1145	1416	+13.6%	63.3	62.6	89.9	72.0	-3.5
b=28, k_0=1	1206	1123	1330	+9.8%	50.0	64.6	90.5	68.4	-7.1
b=32, k_0=1	1106	1217	1365	+10.7%	63.3	67.7	90.9	74.0	-1.5
b=36, k_0=1	1085	1080	1304	+4.1%	56.7	70.7	90.3	72.6	-2.9
b=40, k_0=1	1252	913	1282	+3.5%	70.0	62.6	89.5	74.0	-1.5
b=44, k_0=1	1063	998	1304	+1.0%	70.0	69.7	89.5	76.4	+0.9
b=48, k_0=1	1136	988	1287	+2.4%	60.0	62.1	89.2	70.5	-5.1
b=52, k_0=1	1219	1157	1200	+7.3%	70.0	68.2	90.5	76.2	+0.7
b=8, k_0=2	1332	1024	1312	+10.1%	70.0	65.2	90.4	75.2	-0.3
b=12, k_0=2	1328	1236	1358	+17.7%	71.7	67.2	89.8	76.2	+0.7
b=16, k_0=2	**1283**	**1013**	**1305**	**+8.1%**	**80.0**	**62.1**	**90.1**	**77.4**	**+1.9**
b=20, k_0=2	1210	1037	1221	+4.1%	68.3	64.6	89.6	74.2	-1.3
b=24, k_0=2	1140	1066	1271	+4.4%	63.3	63.1	90.1	72.2	-3.3
b=28, k_0=2	**1051**	**1046**	**1249**	**+0.4%**	**78.3**	**64.6**	**89.9**	**77.6**	**+2.1**
b=32, k_0=2	1117	1147	1241	+5.2%	76.7	64.1	90.7	77.2	+1.7
b=36, k_0=2	1101	960	1199	-2.2%	68.3	64.6	89.4	74.1	-1.4
b=40, k_0=2	1189	971	1195	+0.7%	73.3	66.7	90.0	76.7	+1.2
b=44, k_0=2	1084	949	1203	-2.9%	71.7	65.2	89.6	75.5	-0.0
b=48, k_0=2	1181	1023	1237	+3.3%	75.0	64.1	90.3	76.5	+1.0
b=52, k_0=2	988	989	1166	-5.7%	73.3	67.7	90.3	77.1	+1.6

Table 10. **Scenario S1 (no spec. dec., single-load) at concurrency c=16.** All methods that appear in Figure 6 and the corresponding per-c figures in `figs/full_post_oea_rerun/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	1624	1341	1784		73.3	68.2	89.8	77.1	
k_0=1, b=0	2091	1787	1930	+22.3%	58.3	62.6	90.2	70.4	-6.7
k_0=2, b=0	**2093**	**1525**	**2105**	**+20.5%**	**73.3**	**68.7**	**89.8**	**77.3**	**+0.2**
k_0=3, b=0	1475	1340	1654	-5.9%	78.3	66.7	89.8	78.3	+1.2
b=8, k_0=0	2854	2449	3209	+79.2%	–	–	–	–	–
b=12, k_0=0	3319	2738	3174	+94.3%	–	–	–	–	–
b=16, k_0=0	3342	2786	2867	+89.4%	–	–	–	–	–
b=20, k_0=0	2632	2168	2973	+63.6%	–	–	–	–	–
b=8, k_0=1	**2558**	**1798**	**2667**	**+47.9%**	**51.7**	**58.6**	**91.3**	**67.2**	**-9.9**
b=12, k_0=1	**2171**	**2127**	**2652**	**+46.3%**	**50.0**	**62.6**	**90.8**	**67.8**	**-9.3**
b=16, k_0=1	**2448**	**1537**	**2316**	**+32.7%**	**61.7**	**62.6**	**91.0**	**71.8**	**-5.4**
b=20, k_0=1	2021	1682	2140	+23.0%	56.7	62.1	90.3	69.7	-7.4
b=24, k_0=1	2242	1631	2343	+30.9%	56.7	61.6	90.3	69.5	-7.6
b=28, k_0=1	2183	1405	2184	+21.5%	56.7	63.6	90.3	70.2	-6.9
b=32, k_0=1	1852	1389	2127	+13.0%	55.0	67.2	90.8	71.0	-6.1
b=36, k_0=1	2068	1516	1990	+17.4%	66.7	67.7	90.5	75.0	-2.2
b=40, k_0=1	1770	1489	1984	+10.4%	70.0	64.6	89.8	74.8	-2.3
b=44, k_0=1	1766	1661	1916	+12.5%	71.7	63.6	90.3	75.2	-1.9
b=48, k_0=1	1977	1450	1973	+13.7%	60.0	63.6	90.2	71.3	-5.8
b=52, k_0=1	1705	1242	1968	+3.5%	68.3	63.1	89.3	73.6	-3.5
b=8, k_0=2	1863	1515	2221	+17.9%	71.7	64.1	90.1	75.3	-1.8
b=12, k_0=2	2145	1390	2135	+19.4%	60.0	67.2	90.2	72.5	-4.7
b=16, k_0=2	2100	1508	2096	+20.1%	75.0	63.6	90.1	76.2	-0.9
b=20, k_0=2	1441	1465	2009	+3.5%	70.0	63.1	90.1	74.4	-2.7
b=24, k_0=2	1752	1303	2000	+6.4%	71.7	66.2	89.8	75.9	-1.2
b=28, k_0=2	1719	1420	1852	+5.1%	70.0	66.2	89.5	75.2	-1.9
b=32, k_0=2	1928	1401	1858	+9.2%	66.7	61.6	90.4	72.9	-4.2
b=36, k_0=2	**1876**	**1568**	**1831**	**+11.0%**	**78.3**	**65.2**	**90.5**	**78.0**	**+0.9**
b=40, k_0=2	1867	1565	1898	+12.2%	71.7	65.2	89.8	75.5	-1.6
b=44, k_0=2	**1646**	**1384**	**1804**	**+1.7%**	**80.0**	**65.7**	**89.7**	**78.4**	**+1.3**
b=48, k_0=2	1834	1502	1942	+11.1%	71.7	67.2	89.9	76.3	-0.9
b=52, k_0=2	1785	1483	1788	+6.4%	73.3	65.7	89.8	76.3	-0.8

Table 11. **Scenario S1 (no spec. dec., single-load) at concurrency c=32.** All methods that appear in Figure 6 and the corresponding per-c figures in `figs/full_post_oea_rerun/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

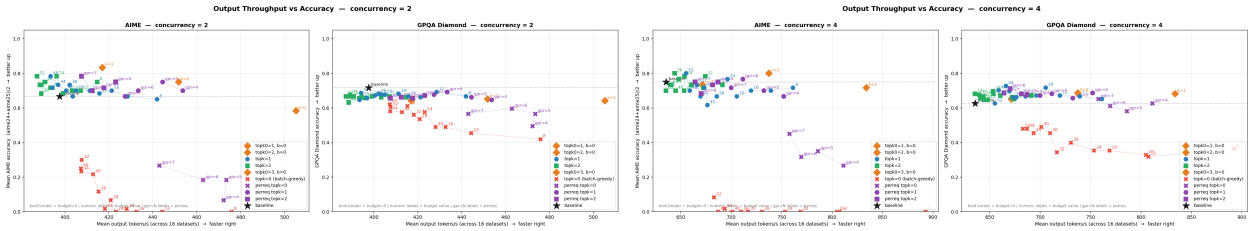
Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	2481	2011	2746		73.3	67.7	89.8	77.0	
k_0=1, b=0	3436	2340	2997	+21.2%	58.3	67.7	90.1	72.1	-4.9
k_0=2, b=0	3708	2416	2696	+21.8%	68.3	62.6	90.5	73.8	-3.1
k_0=3, b=0	2574	2103	2620	+0.8%	76.7	64.1	91.0	77.3	+0.3
b=8, k_0=0	5667	4621	4954	+110.6%	–	–	–	–	–
b=12, k_0=0	5389	3346	4413	+81.6%	–	–	–	–	–
b=16, k_0=0	5140	4338	4218	+89.2%	–	–	–	–	–
b=20, k_0=0	5086	4199	3847	+81.4%	–	–	–	–	–
b=28, k_0=0	2974	3975	4380	+56.5%	–	–	–	–	–
b=32, k_0=0	3540	3836	3967	+56.7%	–	–	–	–	–
b=52, k_0=0	3922	2620	3646	+40.8%	–	–	90.4	90.4	+13.4
b=8, k_0=1	3804	2523	3499	+35.7%	51.7	55.1	91.6	66.1	-10.8
b=12, k_0=1	**3098**	**3214**	**3755**	**+39.1%**	**51.7**	**61.1**	**90.9**	**67.9**	**-9.1**
b=16, k_0=1	3053	3105	3202	+29.3%	50.0	62.1	91.1	67.8	-9.2
b=20, k_0=1	**2969**	**3048**	**3554**	**+32.2%**	**55.0**	**62.1**	**90.6**	**69.2**	**-7.7**
b=24, k_0=1	2915	2994	3197	+25.8%	63.3	60.6	91.2	71.7	-5.2
b=28, k_0=1	2840	2002	2981	+8.1%	63.3	61.6	90.5	71.8	-5.1
b=32, k_0=1	2890	2348	3310	+18.1%	58.3	63.6	91.4	71.1	-5.8
b=36, k_0=1	3489	2314	3033	+22.1%	61.7	66.2	91.4	73.1	-3.9
b=40, k_0=1	2787	2711	2808	+14.7%	68.3	64.6	91.3	74.8	-2.2
b=44, k_0=1	2708	2191	2683	+4.7%	60.0	62.1	91.4	71.2	-5.8
b=48, k_0=1	2667	2648	2910	+13.6%	65.0	66.2	90.3	73.8	-3.1
b=52, k_0=1	2667	2595	2855	+12.1%	71.7	65.2	90.1	75.7	-1.3
b=8, k_0=2	**3431**	**2732**	**3054**	**+27.3%**	**73.3**	**63.1**	**89.0**	**75.2**	**-1.8**
b=12, k_0=2	2771	2216	3073	+11.3%	73.3	64.6	90.4	76.1	-0.8
b=16, k_0=2	3332	2675	3208	+27.3%	68.3	62.6	89.8	73.6	-3.4
b=20, k_0=2	2732	2174	2784	+6.2%	66.7	68.2	90.8	75.2	-1.7
b=24, k_0=2	3287	2575	2757	+19.1%	60.0	65.2	90.4	71.8	-5.1
b=28, k_0=2	**3214**	**2143**	**2885**	**+13.9%**	**71.7**	**68.2**	**90.4**	**76.8**	**-0.2**
b=32, k_0=2	2235	2093	2856	-0.8%	70.0	63.1	89.5	74.2	-2.7
b=36, k_0=2	**2615**	**2068**	**2825**	**+3.7%**	**75.0**	**65.7**	**91.1**	**77.3**	**+0.3**
b=40, k_0=2	3109	2456	2907	+17.0%	66.7	68.2	90.5	75.1	-1.8
b=44, k_0=2	3078	2014	2355	+2.9%	73.3	66.2	90.4	76.6	-0.3
b=48, k_0=2	2188	2050	2595	-5.6%	68.3	66.7	89.6	74.9	-2.1
b=52, k_0=2	2564	2405	2507	+3.3%	65.0	65.2	89.8	73.3	-3.6

Table 12. **Scenario S1 (no spec. dec., single-load) at concurrency c=64.** All methods that appear in Figure 6 and the corresponding per-c figures in `figs/full_post_oea_rerun/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	2028	2000	3783		80.0	65.7	90.6	78.8	
k_0=1, b=0	**3391**	**2797**	**5298**	**+47.1%**	**56.7**	**64.6**	**90.4**	**70.6**	**-8.2**
k_0=2, b=0	2476	1999	3598	+3.4%	68.3	65.7	90.1	74.7	-4.1
k_0=3, b=0	**2586**	**2551**	**3526**	**+10.9%**	**78.3**	**65.7**	**90.1**	**78.0**	**-0.7**
b=8, k_0=1	2653	2220	4123	+15.2%	58.3	66.2	90.8	71.7	-7.0
b=12, k_0=1	**3232**	**2201**	**4952**	**+33.0%**	**68.3**	**68.7**	**91.4**	**76.1**	**-2.6**
b=16, k_0=1	3060	2090	4700	+26.1%	63.3	63.6	91.6	72.9	-5.9
b=20, k_0=1	3048	2530	4804	+32.9%	63.3	70.7	90.8	74.9	-3.8
b=24, k_0=1	2945	2059	4565	+22.5%	66.7	64.6	91.0	74.1	-4.7
b=28, k_0=1	2945	2441	4187	+22.6%	63.3	65.7	90.4	73.1	-5.6
b=32, k_0=1	2892	2386	3271	+9.5%	61.7	69.2	90.3	73.7	-5.0
b=36, k_0=1	2391	2909	4062	+19.9%	70.0	63.1	91.2	74.8	-4.0
b=40, k_0=1	2357	2298	3913	+9.7%	71.7	66.2	90.9	76.2	-2.5
b=44, k_0=1	2787	1950	4360	+16.5%	66.7	65.7	90.5	74.3	-4.5
b=48, k_0=1	2731	1914	3018	-1.9%	68.3	69.7	91.1	76.4	-2.4
b=52, k_0=1	2303	1911	3970	+4.8%	70.0	70.7	90.1	77.0	-1.8
b=8, k_0=2	2317	1936	4028	+6.0%	71.7	69.7	90.1	77.1	-1.6
b=12, k_0=2	2725	2228	3772	+11.7%	66.7	67.2	90.3	74.7	-4.0
b=16, k_0=2	2703	2206	4143	+15.9%	70.0	65.2	90.1	75.1	-3.7
b=20, k_0=2	2305	1939	3633	+0.9%	73.3	67.7	90.8	77.3	-1.5
b=24, k_0=2	2278	2196	3789	+5.8%	76.7	70.7	89.6	79.0	+0.2
b=28, k_0=2	2669	2198	3560	+7.9%	71.7	68.2	90.7	76.8	-1.9
b=32, k_0=2	2647	1815	3801	+5.8%	73.3	69.2	90.4	77.7	-1.1
b=36, k_0=2	2225	1556	3344	-8.8%	76.7	68.7	90.2	78.5	-0.2
b=40, k_0=2	2208	1826	3810	+0.4%	80.0	66.2	90.3	78.8	+0.1
b=44, k_0=2	2589	2158	3747	+8.7%	68.3	69.2	90.1	75.9	-2.9
b=48, k_0=2	2501	1765	2917	-8.0%	75.0	68.2	90.4	77.9	-0.9
b=52, k_0=2	**2177**	**1820**	**4356**	**+6.9%**	**80.0**	**66.2**	**90.9**	**79.0**	**+0.3**

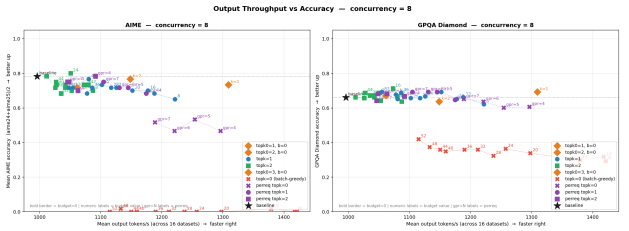
Table 13. **Scenario S1 (no spec. dec., single-load) at concurrency c=128.** All methods that appear in Figure 6 and the corresponding per-c figures in `figs/full_post_oea_rerun/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

D.1.2. SCENARIO 2 — EAGLE3 SPEC. DEC., SINGLE-LOAD

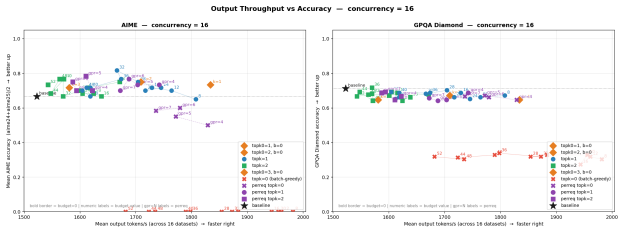


$c = 2$

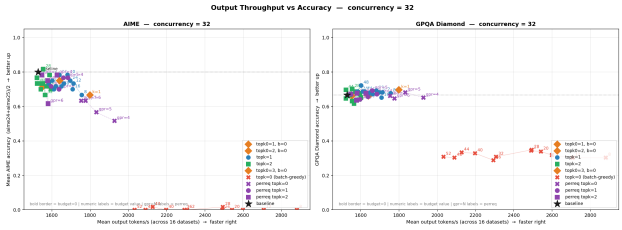
$c = 4$



$c = 8$



$c = 16$

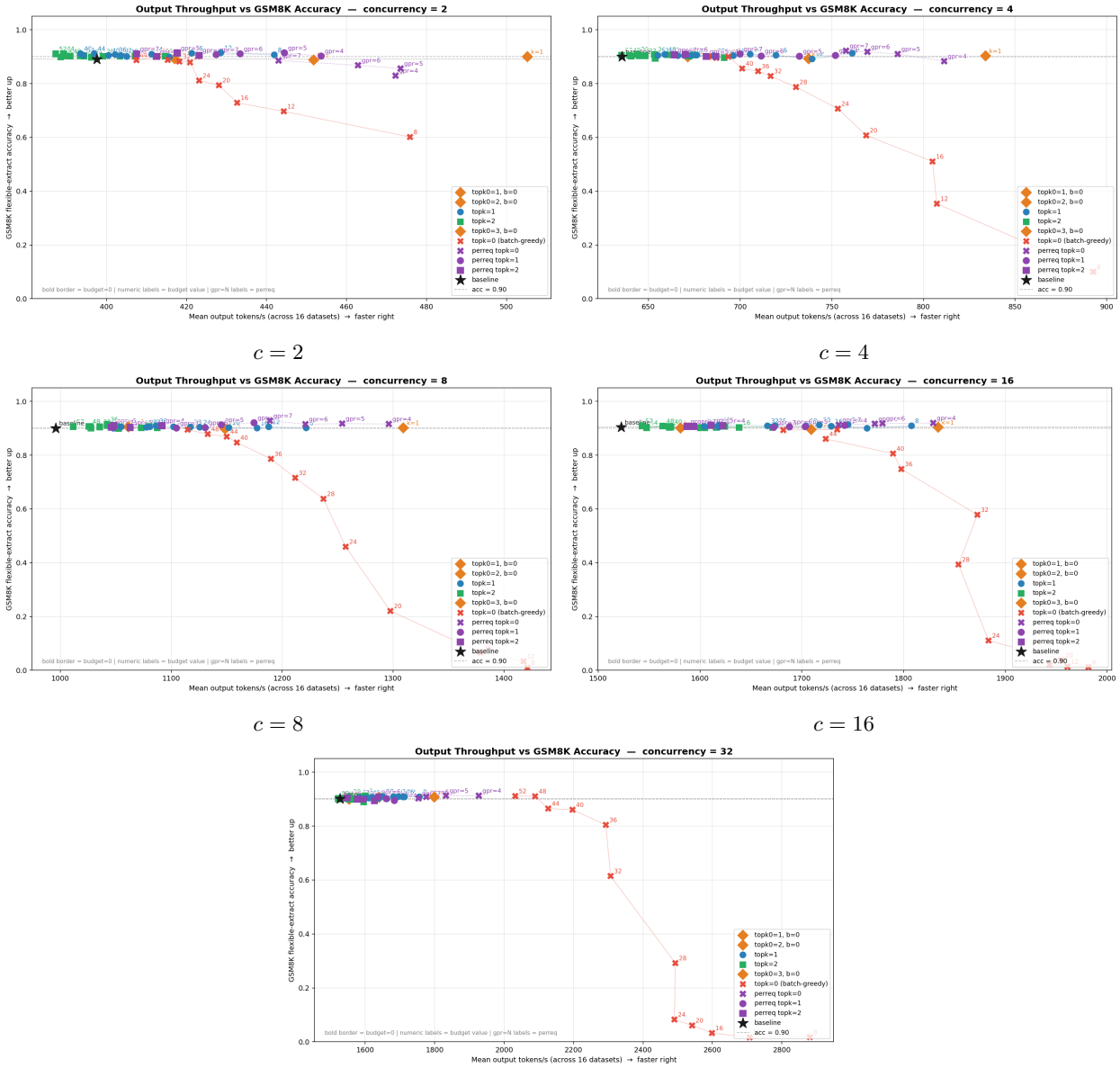


$c = 32$

Figure 12. **Scenario S2 (Eagle3 spec. dec., single-load) — OTPS vs. accuracy on AIME∪GPQA, all concurrencies.** Each panel is one concurrency c. Markers: orange diamonds = OEA (b=0); blue/green = global pruning at k_0=1/2; red X = k_0=0 batch-greedy; purple = per-request (when present); black star = unpruned baseline. Numeric labels next to each point are the budget b (or k_0 for OEA, g for per-request).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	442	390	474		66.7	71.7	89.0	75.8	
k_0=1, b=0	**572**	**493**	**634**	**+30.0%**	**58.3**	**64.1**	**90.0**	**70.8**	**-5.0**
k_0=2, b=0	**506**	**448**	**549**	**+15.0%**	**75.0**	**65.2**	**88.8**	**76.3**	**+0.5**
k_0=3, b=0	**464**	**413**	**506**	**+5.8%**	**83.3**	**64.1**	**89.2**	**78.9**	**+3.1**
b=8, k_0=1	493	431	543	+12.3%	65.0	66.7	90.7	74.1	-1.7
b=12, k_0=1	481	425	525	+9.6%	66.7	69.2	91.4	75.8	-0.0
b=16, k_0=1	471	418	513	+7.3%	70.0	66.2	91.3	75.8	+0.0
b=20, k_0=1	460	410	500	+4.8%	68.3	66.7	89.8	74.9	-0.8
b=24, k_0=1	459	411	494	+4.4%	70.0	68.2	90.9	76.4	+0.6
b=28, k_0=1	447	398	484	+1.6%	70.0	67.7	90.1	75.9	+0.1
b=32, k_0=1	445	399	481	+1.4%	66.7	67.7	90.4	74.9	-0.9
b=36, k_0=1	440	401	486	+1.5%	73.3	68.2	90.8	77.4	+1.6
b=40, k_0=1	447	394	481	+1.2%	70.0	67.2	90.4	75.9	+0.1
b=44, k_0=1	437	391	478	-0.1%	73.3	65.7	91.1	76.7	+0.9
b=48, k_0=1	434	391	475	-0.5%	78.3	64.6	91.1	78.0	+2.2
b=52, k_0=1	433	397	475	-0.1%	71.7	68.7	90.5	77.0	+1.2
b=8, k_0=2	465	410	501	+5.3%	75.0	66.2	90.1	77.1	+1.3
b=12, k_0=2	460	405	494	+4.0%	70.0	66.7	90.2	75.6	-0.2
b=16, k_0=2	441	405	492	+2.4%	70.0	66.2	90.1	75.4	-0.4
b=20, k_0=2	444	394	486	+1.3%	68.3	66.7	90.1	75.0	-0.7
b=24, k_0=2	438	395	476	+0.2%	78.3	65.7	89.8	77.9	+2.1
b=28, k_0=2	437	385	479	-0.4%	71.7	66.2	90.2	76.0	+0.2
b=32, k_0=2	442	391	474	-0.0%	71.7	65.7	90.4	75.9	+0.1
b=36, k_0=2	428	385	467	-2.1%	68.3	66.7	90.4	75.1	-0.7
b=40, k_0=2	440	388	471	-0.5%	75.0	66.7	90.1	77.3	+1.5
b=44, k_0=2	433	381	466	-2.1%	73.3	67.7	91.1	77.4	+1.6
b=48, k_0=2	432	384	469	-1.7%	73.3	63.1	89.9	75.5	-0.3
b=52, k_0=2	430	391	465	-1.6%	78.3	66.7	91.1	78.7	+2.9
perreq g=4, k_0=1	**510**	**448**	**553**	**+15.6%**	**70.0**	**64.6**	**90.2**	**75.0**	**-0.8**
perreq g=5, k_0=1	**494**	**439**	**539**	**+12.6%**	**75.0**	**66.2**	**91.4**	**77.5**	**+1.7**
perreq g=6, k_0=1	486	426	533	+10.6%	70.0	69.2	91.1	76.7	+1.0
perreq g=7, k_0=1	480	424	521	+9.1%	66.7	67.7	90.8	75.0	-0.8
perreq g=4, k_0=2	**478**	**420**	**516**	**+8.2%**	**75.0**	**67.2**	**90.5**	**77.6**	**+1.8**
perreq g=5, k_0=2	465	416	505	+6.1%	71.7	65.7	91.4	76.2	+0.4
perreq g=6, k_0=2	461	410	497	+4.7%	70.0	66.2	90.1	75.4	-0.4
perreq g=7, k_0=2	457	405	491	+3.5%	78.3	65.7	91.1	78.3	+2.6

Table 14. **Scenario S2 (Eagle3 spec. dec., single-load) at concurrency c=2.** All methods that appear in the corresponding figure in `figs/eagle3_acc_speed_v4/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy) over all rows shown in the table. $k_0 = 0$ (batch-greedy) configurations are excluded if their per-task accuracy is below baseline on at least one dataset. "–" indicates the configuration was not collected at this (c).



$c = 2$ $c = 4$ $c = 8$ $c = 16$ $c = 32$

Figure 13. **Scenario S2 (Eagle3 spec. dec., single-load) — OTPS vs. GSM8K accuracy, all concurrencies.** Each panel is one concurrency c. Markers: orange diamonds = OEA (b=0); blue/green = global pruning at k_0=1/2; red X = k_0=0 batch-greedy; purple = per-request (when present); black star = unpruned baseline. Numeric labels next to each point are the budget b (or k_0 for OEA, g for per-request).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	711	619	751		75.0	62.6	90.0	75.9	
k_0=1, b=0	**955**	**824**	**1006**	**+33.8%**	**71.7**	**68.2**	**90.3**	**76.7**	**+0.8**
k_0=2, b=0	**838**	**726**	**885**	**+17.7%**	**80.0**	**68.7**	**89.2**	**79.3**	**+3.4**
k_0=3, b=0	759	665	805	+7.1%	73.3	65.2	89.9	76.1	+0.3
b=8, k_0=1	875	741	903	+21.0%	71.7	65.2	91.3	76.0	+0.2
b=12, k_0=1	852	719	884	+18.0%	70.0	66.7	89.2	75.3	-0.6
b=16, k_0=1	826	700	867	+15.0%	70.0	67.2	90.5	75.9	+0.0
b=20, k_0=1	778	684	847	+11.0%	66.7	71.7	90.9	76.4	+0.6
b=24, k_0=1	779	681	832	+10.2%	76.7	67.2	90.4	78.1	+2.2
b=28, k_0=1	771	685	824	+9.5%	66.7	66.7	90.4	74.6	-1.3
b=32, k_0=1	765	667	805	+7.5%	61.7	66.2	90.6	72.8	-3.1
b=36, k_0=1	759	662	801	+6.8%	71.7	66.2	90.5	76.1	+0.2
b=40, k_0=1	752	657	801	+6.2%	66.7	69.7	90.4	75.6	-0.3
b=44, k_0=1	748	664	790	+5.8%	73.3	72.7	90.4	78.8	+2.9
b=48, k_0=1	736	640	785	+3.8%	70.0	70.2	90.8	77.0	+1.1
b=52, k_0=1	720	644	769	+2.4%	80.0	62.6	90.4	77.7	+1.8
b=8, k_0=2	775	683	829	+9.9%	73.3	67.2	89.6	76.7	+0.8
b=12, k_0=2	760	661	813	+7.4%	78.3	65.2	90.6	78.0	+2.2
b=16, k_0=2	763	671	809	+7.7%	71.7	66.2	90.2	76.0	+0.1
b=20, k_0=2	744	646	792	+4.9%	73.3	67.2	90.7	77.1	+1.2
b=24, k_0=2	730	648	785	+3.9%	78.3	68.2	89.3	78.6	+2.7
b=28, k_0=2	726	643	781	+3.3%	70.0	69.7	90.8	76.8	+1.0
b=32, k_0=2	727	636	750	+1.5%	76.7	64.6	90.4	77.2	+1.4
b=36, k_0=2	727	637	767	+2.4%	80.0	66.7	90.9	79.2	+3.3
b=40, k_0=2	729	631	772	+2.4%	70.0	64.6	90.4	75.0	-0.9
b=44, k_0=2	719	639	774	+2.4%	73.3	65.7	90.1	76.4	+0.5
b=48, k_0=2	720	628	762	+1.4%	73.3	67.7	90.5	77.2	+1.3
b=52, k_0=2	716	625	748	+0.4%	70.0	68.2	90.5	76.2	+0.4
perreq g=4, k_0=1	850	741	915	+20.4%	66.7	68.7	90.4	75.3	-0.6
perreq g=5, k_0=1	834	709	885	+16.6%	70.0	65.7	90.1	75.3	-0.6
perreq g=6, k_0=1	800	703	846	+12.9%	76.7	68.2	90.2	78.4	+2.5
perreq g=7, k_0=1	805	682	840	+11.8%	71.7	67.2	91.0	76.6	+0.7
perreq g=4, k_0=2	774	674	817	+8.8%	75.0	67.7	90.1	77.6	+1.7
perreq g=5, k_0=2	777	673	814	+8.8%	73.3	68.2	90.1	77.2	+1.3
perreq g=6, k_0=2	759	660	796	+6.4%	68.3	71.2	90.8	76.8	+0.9
perreq g=7, k_0=2	746	648	795	+5.2%	75.0	68.7	90.7	78.1	+2.2

Table 15. **Scenario S2 (Eagle3 spec. dec., single-load) at concurrency c=4.** All methods that appear in the corresponding figure in `figs/eagle3_acc_speed_v4/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy) over all rows shown in the table. $k_0 = 0$ (batch-greedy) configurations are excluded if their per-task accuracy is below baseline on at least one dataset. "–" indicates the configuration was not collected at this (c).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	1135	895	1199		78.3	66.2	90.1	78.2	
k_0=1, b=0	**1528**	**1290**	**1598**	**+36.8%**	**73.3**	**69.2**	**90.1**	**77.6**	**-0.6**
k_0=2, b=0	1375	1032	1366	+16.8%	76.7	63.6	89.8	76.7	-1.5
k_0=3, b=0	1214	1031	1262	+8.6%	71.7	67.2	90.5	76.5	-1.7
b=8, k_0=1	1415	1209	1497	+27.6%	65.0	62.1	90.1	72.4	-5.8
b=12, k_0=1	1398	1070	1462	+21.7%	68.3	66.2	90.5	75.0	-3.2
b=16, k_0=1	1330	1138	1436	+20.9%	70.0	65.2	90.1	75.1	-3.1
b=20, k_0=1	1336	1047	1399	+17.1%	70.0	69.2	90.2	76.5	-1.7
b=24, k_0=1	1316	1009	1376	+14.6%	71.7	66.7	90.4	76.2	-2.0
b=28, k_0=1	1289	1104	1348	+15.9%	71.7	65.7	90.4	75.9	-2.3
b=32, k_0=1	1166	1082	1341	+11.2%	73.3	65.7	90.5	76.5	-1.7
b=36, k_0=1	1249	987	1322	+10.2%	71.7	68.7	90.9	77.1	-1.1
b=40, k_0=1	1146	1084	1297	+9.2%	68.3	67.7	90.4	75.5	-2.7
b=44, k_0=1	1230	1066	1303	+11.5%	76.7	65.2	90.5	77.4	-0.7
b=48, k_0=1	1129	968	1277	+4.5%	71.7	68.2	90.0	76.6	-1.6
b=52, k_0=1	1215	964	1277	+7.0%	71.7	69.2	90.5	77.1	-1.1
b=8, k_0=2	1179	1001	1332	+8.8%	70.0	63.6	90.2	74.6	-3.6
b=12, k_0=2	1152	1001	1302	+7.0%	73.3	66.7	90.7	76.9	-1.3
b=16, k_0=2	1251	971	1300	+9.1%	73.3	71.2	90.1	78.2	+0.0
b=20, k_0=2	1120	1045	1261	+6.1%	70.0	64.1	89.9	74.7	-3.5
b=24, k_0=2	**1099**	**962**	**1295**	**+3.9%**	**80.0**	**68.7**	**90.7**	**79.8**	**+1.6**
b=28, k_0=2	1211	1029	1265	+8.6%	73.3	66.2	91.1	76.8	-1.3
b=32, k_0=2	1189	939	1248	+4.5%	68.3	67.2	90.5	75.3	-2.8
b=36, k_0=2	1195	1014	1250	+7.1%	71.7	67.2	91.4	76.8	-1.4
b=40, k_0=2	1168	1024	1238	+6.2%	73.3	68.7	90.1	77.4	-0.8
b=44, k_0=2	1100	1014	1238	+3.8%	75.0	68.7	90.6	78.1	-0.1
b=48, k_0=2	1182	994	1231	+5.5%	71.7	65.7	90.7	76.0	-2.2
b=52, k_0=2	1058	989	1228	+1.4%	78.3	66.2	90.6	78.4	+0.2
perreq g=4, k_0=1	1351	1150	1399	+20.8%	68.3	64.6	92.0	75.0	-3.2
perreq g=5, k_0=1	1225	1020	1400	+12.9%	71.7	69.2	91.3	77.4	-0.8
perreq g=6, k_0=1	1303	1020	1359	+14.0%	71.7	69.2	90.2	77.0	-1.2
perreq g=7, k_0=1	1275	993	1305	+10.6%	75.0	69.2	90.1	78.1	-0.1
perreq g=4, k_0=2	**1228**	**1052**	**1307**	**+11.1%**	**78.3**	**66.7**	**91.0**	**78.7**	**+0.5**
perreq g=5, k_0=2	1232	991	1286	+8.7%	70.0	68.2	90.3	76.2	-2.0
perreq g=6, k_0=2	1119	970	1236	+3.0%	75.0	68.2	91.0	78.1	-0.1
perreq g=7, k_0=2	1199	1028	1266	+8.2%	75.0	64.1	90.4	76.5	-1.7

Table 16. **Scenario S2 (Eagle3 spec. dec., single-load) at concurrency c=8.** All methods that appear in the corresponding figure in `figs/eagle3_acc_speed_v4/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy) over all rows shown in the table. $k_0 = 0$ (batch-greedy) configurations are excluded if their per-task accuracy is below baseline on at least one dataset. "–" indicates the configuration was not collected at this (c).

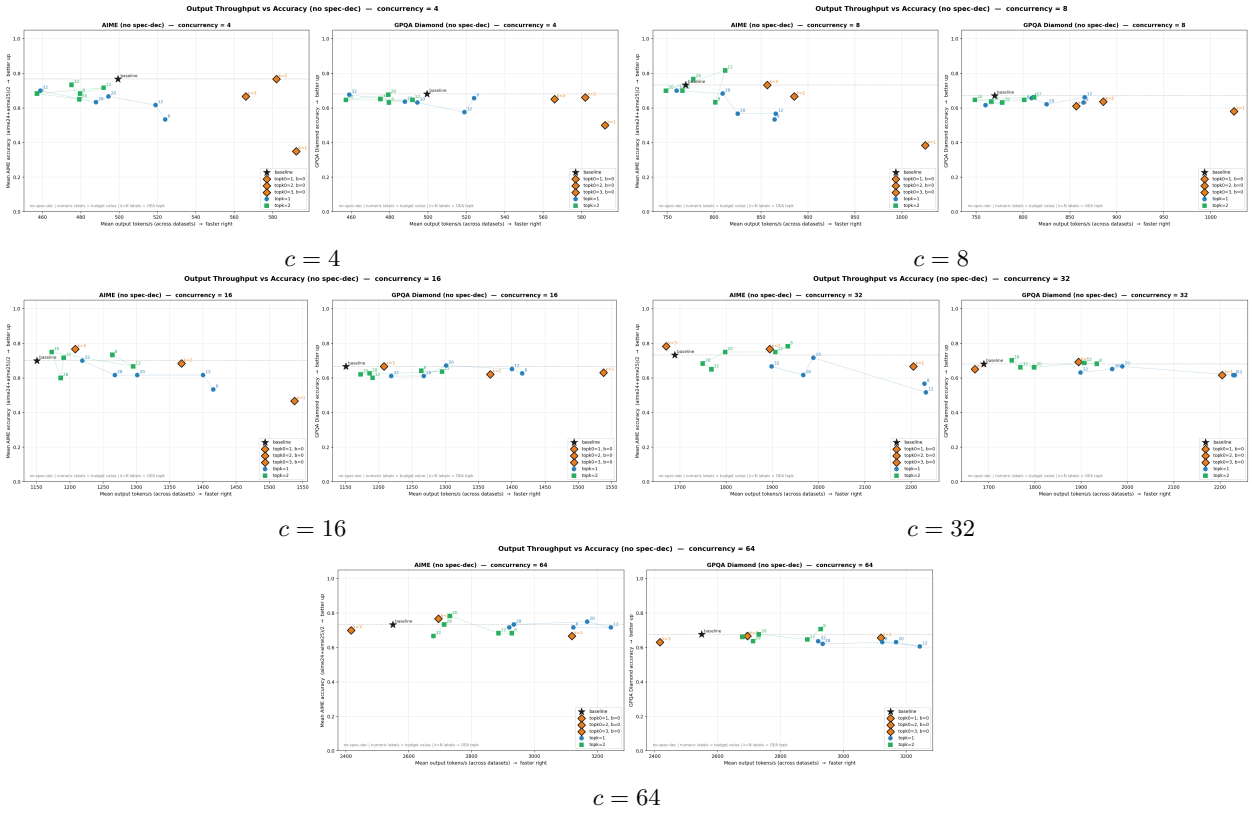
Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	1584	1332	1858		66.7	71.2	90.4	76.1	
k_0=1, b=0	**1954**	**1650**	**2365**	**+25.0%**	**73.3**	**64.6**	**90.4**	**76.1**	**+0.0**
k_0=2, b=0	**2034**	**1734**	**1947**	**+19.7%**	**75.0**	**67.2**	**89.5**	**77.2**	**+1.2**
k_0=3, b=0	1662	1393	1941	+4.6%	71.7	64.6	90.1	75.5	-0.6
b=8, k_0=1	1884	1641	2268	+21.4%	65.0	67.2	90.9	74.4	-1.7
b=12, k_0=1	2084	1530	2332	+24.5%	70.0	66.2	90.0	75.4	-0.7
b=16, k_0=1	1874	1552	2081	+15.4%	71.7	68.7	90.8	77.0	+1.0
b=20, k_0=1	1815	1509	2155	+14.8%	70.0	66.2	91.2	75.8	-0.3
b=24, k_0=1	2056	1707	2130	+23.5%	71.7	65.2	91.4	76.1	-0.0
b=28, k_0=1	**2006**	**1496**	**2090**	**+17.1%**	**75.0**	**70.2**	**90.9**	**78.7**	**+2.6**
b=32, k_0=1	**1746**	**1480**	**2075**	**+11.0%**	**81.7**	**68.2**	**90.9**	**80.3**	**+4.2**
b=36, k_0=1	**1959**	**1457**	**2054**	**+14.6%**	**76.7**	**68.7**	**90.9**	**78.8**	**+2.7**
b=40, k_0=1	1715	1458	1937	+7.0%	71.7	68.7	91.2	77.2	+1.1
b=44, k_0=1	1686	1397	1934	+5.1%	71.7	65.2	91.1	76.0	-0.1
b=48, k_0=1	1673	1396	1994	+6.1%	66.7	66.2	90.4	74.4	-1.7
b=52, k_0=1	1655	1427	1981	+6.0%	70.0	69.2	90.7	76.6	+0.5
b=8, k_0=2	1759	1689	2143	+17.1%	75.0	67.7	90.4	77.7	+1.6
b=12, k_0=2	1688	1450	2043	+8.5%	71.7	68.7	90.4	76.9	+0.8
b=16, k_0=2	1915	1449	2001	+12.4%	66.7	66.2	90.3	74.4	-1.7
b=20, k_0=2	1893	1427	1995	+11.3%	68.3	64.1	90.7	74.4	-1.7
b=24, k_0=2	1680	1581	1970	+9.6%	68.3	69.2	90.2	75.9	-0.2
b=28, k_0=2	1662	1379	1962	+4.8%	70.0	67.2	90.2	75.8	-0.3
b=32, k_0=2	1648	1378	1946	+4.1%	66.7	68.2	90.7	75.2	-0.9
b=36, k_0=2	1634	1371	1927	+3.3%	76.7	71.7	90.4	79.6	+3.5
b=40, k_0=2	1630	1403	1857	+2.4%	76.7	64.1	90.5	77.1	+1.0
b=44, k_0=2	1585	1365	1924	+2.1%	68.3	69.2	90.2	75.9	-0.2
b=48, k_0=2	1610	1546	1909	+6.1%	76.7	67.7	90.8	78.4	+2.3
b=52, k_0=2	1643	1222	1901	-0.2%	73.3	66.7	90.9	77.0	+0.9
perreq g=4, k_0=1	1794	1534	2150	+14.7%	73.3	68.7	91.1	77.7	+1.6
perreq g=5, k_0=1	1774	1465	2106	+12.0%	73.3	64.6	90.6	76.2	+0.1
perreq g=6, k_0=1	1983	1664	2064	+19.6%	76.7	64.1	90.5	77.1	+1.0
perreq g=7, k_0=1	1948	1628	2024	+17.3%	70.0	65.7	90.3	75.3	-0.8
perreq g=4, k_0=2	1904	1448	2002	+12.1%	70.0	66.7	91.0	75.9	-0.2
perreq g=5, k_0=2	1683	1412	1982	+6.3%	78.3	64.6	91.1	78.0	+2.0
perreq g=6, k_0=2	1679	1390	1879	+3.6%	75.0	68.7	90.7	78.1	+2.0
perreq g=7, k_0=2	1864	1381	1868	+7.1%	70.0	69.2	90.7	76.6	+0.5

Table 17. **Scenario S2 (Eagle3 spec. dec., single-load) at concurrency c=16.** All methods that appear in the corresponding figure in `figs/eagle3_acc_speed_v4/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy) over all rows shown in the table. $k_0 = 0$ (batch-greedy) configurations are excluded if their per-task accuracy is below baseline on at least one dataset. "–" indicates the configuration was not collected at this (c).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	1628	1756	1774		80.0	66.7	90.1	78.9	
k_0=1, b=0	**1758**	**1645**	**2203**	**+8.7%**	**66.7**	**69.7**	**90.8**	**75.7**	**-3.2**
k_0=2, b=0	1624	1468	1882	-3.6%	75.0	66.2	90.7	77.3	-1.7
k_0=3, b=0	1524	1392	1848	-7.6%	71.7	66.2	89.9	75.9	-3.0
b=8, k_0=1	1692	1562	2100	+3.8%	66.7	67.7	90.8	75.1	-3.9
b=12, k_0=1	1705	1578	2068	+3.8%	73.3	68.2	90.8	77.4	-1.5
b=16, k_0=1	1663	1568	2053	+2.5%	70.0	65.2	90.8	75.3	-3.6
b=20, k_0=1	**1645**	**1545**	**2069**	**+2.0%**	**78.3**	**69.2**	**90.9**	**79.5**	**+0.5**
b=24, k_0=1	1650	1510	1999	+0.0%	75.0	68.2	90.9	78.0	-0.9
b=28, k_0=1	1466	1487	1983	-4.3%	71.7	68.2	91.1	77.0	-2.0
b=32, k_0=1	1590	1437	1998	-2.6%	73.3	67.2	90.1	76.9	-2.1
b=36, k_0=1	1588	1431	1861	-5.4%	78.3	66.2	90.3	78.3	-0.7
b=40, k_0=1	1432	1462	1961	-5.9%	70.0	68.7	90.4	76.4	-2.6
b=44, k_0=1	1553	1425	2001	-3.5%	71.7	68.7	90.7	77.0	-1.9
b=48, k_0=1	1546	1396	1935	-5.4%	75.0	72.2	90.4	79.2	+0.3
b=52, k_0=1	1542	1374	1873	-7.2%	73.3	68.7	90.6	77.5	-1.4
b=8, k_0=2	1569	1418	1934	-4.6%	70.0	65.2	91.2	75.5	-3.5
b=12, k_0=2	1592	1395	1795	-7.3%	70.0	67.2	89.0	75.4	-3.5
b=16, k_0=2	1548	1432	1880	-5.8%	78.3	67.7	90.8	78.9	-0.0
b=20, k_0=2	1532	1357	1932	-6.5%	73.3	67.7	90.5	77.2	-1.8
b=24, k_0=2	1402	1400	1866	-9.5%	71.7	68.7	89.9	76.8	-2.2
b=28, k_0=2	**1524**	**1390**	**1804**	**-8.5%**	**81.7**	**70.2**	**90.9**	**80.9**	**+2.0**
b=32, k_0=2	1542	1399	1852	-7.1%	66.7	61.6	90.1	72.8	-6.1
b=36, k_0=2	1517	1385	1857	-7.7%	73.3	63.1	89.9	75.5	-3.5
b=40, k_0=2	1502	1334	1841	-9.3%	70.0	66.7	90.4	75.7	-3.3
b=44, k_0=2	1489	1216	1828	-12.1%	73.3	69.7	90.0	77.7	-1.3
b=48, k_0=2	1494	1335	1832	-9.6%	73.3	66.2	90.1	76.5	-2.4
b=52, k_0=2	1482	1334	1824	-10.0%	76.7	64.6	90.3	77.2	-1.7
perreq g=4, k_0=1	1484	1522	2045	-2.1%	76.7	67.2	89.5	77.8	-1.2
perreq g=5, k_0=1	1605	1491	1934	-2.5%	76.7	69.2	90.1	78.7	-0.3
perreq g=6, k_0=1	1575	1403	1891	-5.6%	70.0	65.7	90.9	75.5	-3.4
perreq g=7, k_0=1	1570	1247	1857	-9.4%	71.7	64.1	89.9	75.2	-3.7
perreq g=4, k_0=2	**1779**	**1855**	**1904**	**+7.4%**	**78.3**	**67.7**	**89.5**	**78.5**	**-0.4**
perreq g=5, k_0=2	1543	1410	1879	-6.3%	75.0	63.6	89.9	76.2	-2.8
perreq g=6, k_0=2	1717	1381	1773	-5.6%	61.7	67.7	89.9	73.1	-5.9
perreq g=7, k_0=2	1504	1362	1754	-10.4%	78.3	67.2	90.5	78.7	-0.3

Table 18. **Scenario S2 (Eagle3 spec. dec., single-load) at concurrency c=32.** All methods that appear in the corresponding figure in `figs/eagle3_acc_speed_v4/`. OTPS is per-task output throughput in tokens/s; the avg. Δ % OTPS column is the mean per-task OTPS gain vs. baseline. Accuracy: GSM8K flexible-extract, AIME mean of '24+'25 (`exact_match,none`), GPQA Diamond. The avg. acc. column averages the three; avg. Δ acc is the absolute change vs. baseline. **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy) over all rows shown in the table. $k_0 = 0$ (batch-greedy) configurations are excluded if their per-task accuracy is below baseline on at least one dataset. "–" indicates the configuration was not collected at this (c).

D.1.3. SCENARIO 3 — NO SPEC. DEC., MIXED-LOAD



$c = 4$ $c = 8$ $c = 16$ $c = 32$ $c = 64$

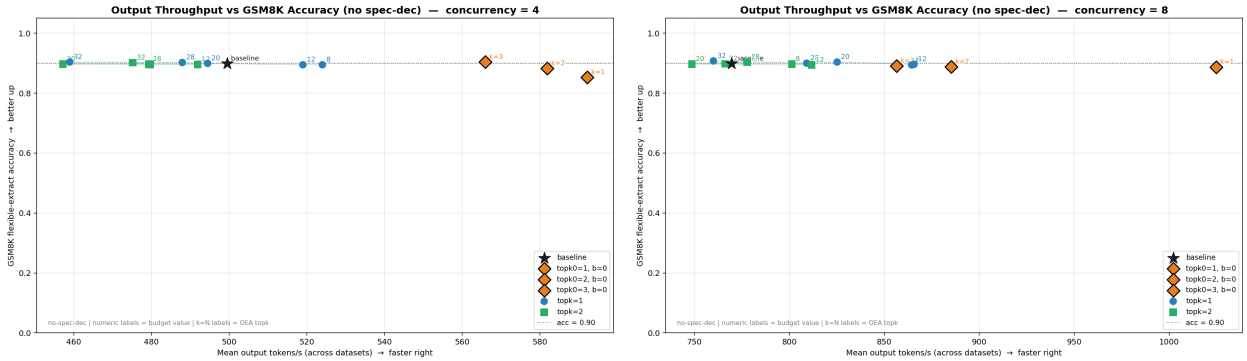
Figure 14. **Scenario S3 (no spec. dec., mixed-load) — OTPS vs. accuracy on AIME∪GPQA, all concurrencies.** Each panel is one concurrency c. Markers: orange diamonds = OEA (b=0); blue/green = global pruning at k_0=1/2; red X = k_0=0 batch-greedy; purple = per-request (when present); black star = unpruned baseline. Numeric labels next to each point are the budget b (or k_0 for OEA, g for per-request).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	**520**	**501**	**504**		**76.7**	**68.2**	**89.8**	**78.2**	
k_0=1, b=0	**647**	**617**	**581**	**+21.1%**	**35.0**	**50.0**	**85.3**	**56.8**	**-21.5**
k_0=2, b=0	**639**	**605**	**577**	**+19.5%**	**76.7**	**66.2**	**88.2**	**77.0**	**-1.2**
k_0=3, b=0	596	566	574	+13.9%	66.7	65.2	90.4	74.1	-4.2
b=8, k_0=1	545	520	526	+4.4%	53.3	65.7	89.5	69.5	-8.7
b=12, k_0=1	530	513	525	+2.9%	61.7	57.6	89.5	69.6	-8.7
b=20, k_0=1	510	487	500	-1.8%	66.7	63.1	89.9	73.2	-5.0
b=28, k_0=1	505	489	486	-3.0%	63.3	63.6	90.2	72.4	-5.8
b=32, k_0=1	474	453	462	-8.8%	70.0	67.7	90.4	76.0	-2.2
b=8, k_0=2	494	477	481	-4.8%	68.3	63.1	89.5	73.6	-4.6
b=12, k_0=2	505	490	490	-2.5%	71.7	64.6	89.5	75.3	-2.9
b=20, k_0=2	472	450	464	-9.1%	68.3	64.6	89.7	74.2	-4.0
b=28, k_0=2	498	483	481	-4.1%	65.0	67.7	89.7	74.1	-4.1
b=32, k_0=2	495	472	489	-4.5%	73.3	65.2	90.1	76.2	-2.0

Table 19. **Scenario S3 (no spec. dec., mixed-load) at concurrency c=4.** All methods that appear in the corresponding figure in `figs/lmeval_acc_no_spec_speed_mixed/`. Accuracy is from the mixed-load lmeval sweep ($c/4$ graded reasoning slots interleaved with $3c/4$ ShareGPT/HumanEval filler slots). OTPS is per-task single-load throughput in tokens/s (the mixed-load sweep was –no-baseline; baseline accuracy is borrowed from S1 single-load at the same c — accuracy is invariant to batch composition for the unpruned baseline). **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

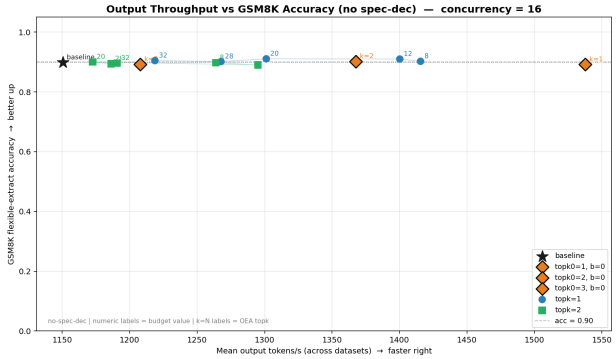
Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	800	722	779		73.3	67.2	89.8	76.8	
k_0=1, b=0	**1109**	**969**	**993**	**+33.5%**	**38.3**	**58.1**	**88.7**	**61.7**	**-15.1**
k_0=2, b=0	**894**	**842**	**883**	**+13.9%**	**66.7**	**63.6**	**88.8**	**73.0**	**-3.8**
k_0=3, b=0	**918**	**804**	**846**	**+11.7%**	**73.3**	**61.1**	**89.1**	**74.5**	**-2.3**
b=8, k_0=1	888	806	856	+10.8%	53.3	63.1	89.4	68.6	-8.2
b=12, k_0=1	899	807	848	+11.0%	56.7	66.2	89.6	70.8	-6.0
b=20, k_0=1	810	729	816	+2.4%	56.7	62.1	90.4	69.7	-7.1
b=28, k_0=1	**833**	**801**	**786**	**+5.2%**	**68.3**	**65.7**	**90.0**	**74.7**	**-2.1**
b=32, k_0=1	777	674	747	-4.4%	70.0	61.6	90.8	74.1	-2.7
b=8, k_0=2	787	758	784	+1.3%	63.3	64.6	89.7	72.6	-4.2
b=12, k_0=2	**793**	**813**	**794**	**+4.3%**	**81.7**	**66.2**	**89.4**	**79.1**	**+2.3**
b=20, k_0=2	730	704	737	-5.6%	70.0	64.6	89.7	74.8	-2.0
b=28, k_0=2	792	774	749	+0.6%	76.7	63.1	90.4	76.7	-0.1
b=32, k_0=2	786	761	741	-0.5%	70.0	63.6	89.8	74.5	-2.3

Table 20. **Scenario S3 (no spec. dec., mixed-load) at concurrency c=8.** All methods that appear in the corresponding figure in `figs/lmeval_acc_no_spec_speed_mixed/`. Accuracy is from the mixed-load lmeval sweep ($c/4$ graded reasoning slots interleaved with $3c/4$ ShareGPT/HumanEval filler slots). OTPS is per-task single-load throughput in tokens/s (the mixed-load sweep was –no-baseline; baseline accuracy is borrowed from S1 single-load at the same c — accuracy is invariant to batch composition for the unpruned baseline). **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

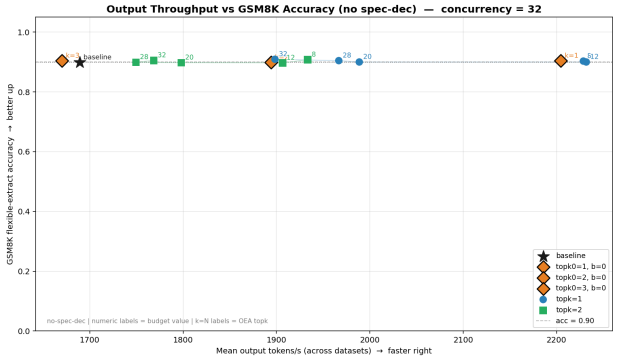


$c = 4$

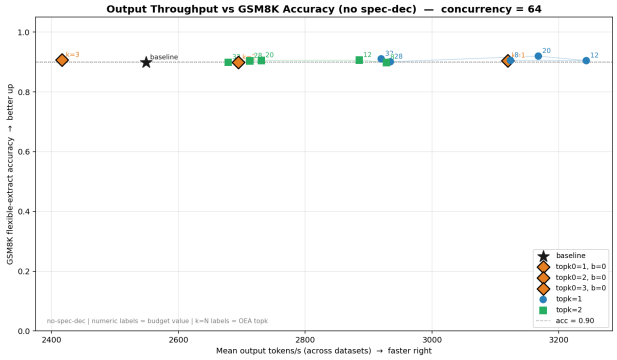
$c = 8$



$c = 16$



$c = 32$



$c = 64$

Figure 15. **Scenario S3 (no spec. dec., mixed-load) — OTPS vs. GSM8K accuracy, all concurrencies.** Each panel is one concurrency c. Markers: orange diamonds = OEA (b=0); blue/green = global pruning at k_0=1/2; red X = k_0=0 batch-greedy; purple = per-request (when present); black star = unpruned baseline. Numeric labels next to each point are the budget b (or k_0 for OEA, g for per-request).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	1098	1012	1222		70.0	66.7	89.8	75.5	
k_0=1, b=0	**1468**	**1162**	**1788**	**+32.6%**	**46.7**	**63.1**	**89.2**	**66.3**	**-9.2**
k_0=2, b=0	**1437**	**1151**	**1515**	**+23.1%**	**68.3**	**62.1**	**90.2**	**73.6**	**-1.9**
k_0=3, b=0	**1243**	**960**	**1242**	**+3.4%**	**76.7**	**66.7**	**89.2**	**77.5**	**+2.0**
b=8, k_0=1	1296	1143	1498	+18.2%	53.3	62.6	90.2	68.7	-6.8
b=12, k_0=1	1379	1131	1529	+21.2%	61.7	65.2	91.0	72.6	-2.9
b=20, k_0=1	1275	1073	1404	+12.6%	61.7	67.2	91.1	73.3	-2.2
b=28, k_0=1	1206	1123	1330	+9.8%	61.7	61.1	90.1	71.0	-4.5
b=32, k_0=1	**1106**	**1217**	**1365**	**+10.7%**	**70.0**	**61.1**	**90.4**	**73.9**	**-1.6**
b=8, k_0=2	**1332**	**1024**	**1312**	**+10.1%**	**73.3**	**64.1**	**89.8**	**75.7**	**+0.2**
b=12, k_0=2	1328	1236	1358	+17.7%	66.7	63.6	89.0	73.1	-2.4
b=20, k_0=2	1210	1037	1221	+4.1%	75.0	62.1	90.0	75.7	+0.2
b=28, k_0=2	1051	1046	1249	+0.4%	60.0	62.6	89.4	70.7	-4.8
b=32, k_0=2	1117	1147	1241	+5.2%	71.7	60.1	89.6	73.8	-1.7

Table 21. **Scenario S3 (no spec. dec., mixed-load) at concurrency c=16.** All methods that appear in the corresponding figure in `figs/lmeval_acc_no_spec_speed_mixed/`. Accuracy is from the mixed-load lmeval sweep ($c/4$ graded reasoning slots interleaved with $3c/4$ ShareGPT/HumanEval filler slots). OTPS is per-task single-load throughput in tokens/s (the mixed-load sweep was –no-baseline; baseline accuracy is borrowed from S1 single-load at the same c — accuracy is invariant to batch composition for the unpruned baseline). **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

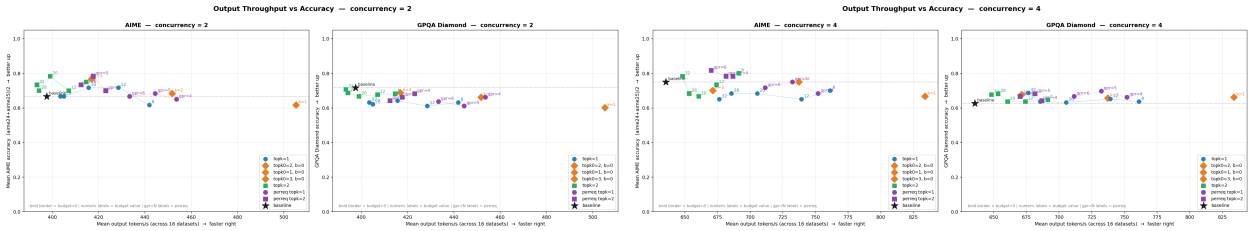
Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	1624	1341	1784		73.3	68.2	89.8	77.1	
k_0=1, b=0	2091	1787	1930	+22.3%	66.7	61.6	90.4	72.9	-4.2
k_0=2, b=0	**2093**	**1525**	**2105**	**+20.5%**	**76.7**	**69.2**	**89.8**	**78.6**	**+1.4**
k_0=3, b=0	1475	1340	1654	-5.9%	78.3	65.2	90.4	78.0	+0.8
b=8, k_0=1	**2558**	**1798**	**2667**	**+47.9%**	**56.7**	**61.6**	**90.3**	**69.5**	**-7.6**
b=12, k_0=1	2171	2127	2652	+46.3%	51.7	61.6	90.0	67.8	-9.4
b=20, k_0=1	**2021**	**1682**	**2140**	**+23.0%**	**71.7**	**66.7**	**90.0**	**76.1**	**-1.0**
b=28, k_0=1	2183	1405	2184	+21.5%	61.7	65.2	90.4	72.4	-4.7
b=32, k_0=1	1852	1389	2127	+13.0%	66.7	63.1	90.9	73.6	-3.6
b=8, k_0=2	**1863**	**1515**	**2221**	**+17.9%**	**78.3**	**68.2**	**90.8**	**79.1**	**+2.0**
b=12, k_0=2	2145	1390	2135	+19.4%	75.0	68.7	89.7	77.8	+0.7
b=20, k_0=2	1441	1465	2009	+3.5%	75.0	66.2	89.8	77.0	-0.1
b=28, k_0=2	1719	1420	1852	+5.1%	68.3	70.2	89.9	76.2	-1.0
b=32, k_0=2	1928	1401	1858	+9.2%	65.0	66.2	90.5	73.9	-3.2

Table 22. **Scenario S3 (no spec. dec., mixed-load) at concurrency c=32.** All methods that appear in the corresponding figure in `figs/lmeval_acc_no_spec_speed_mixed/`. Accuracy is from the mixed-load lmeval sweep ($c/4$ graded reasoning slots interleaved with $3c/4$ ShareGPT/HumanEval filler slots). OTPS is per-task single-load throughput in tokens/s (the mixed-load sweep was –no-baseline; baseline accuracy is borrowed from S1 single-load at the same c — accuracy is invariant to batch composition for the unpruned baseline). **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

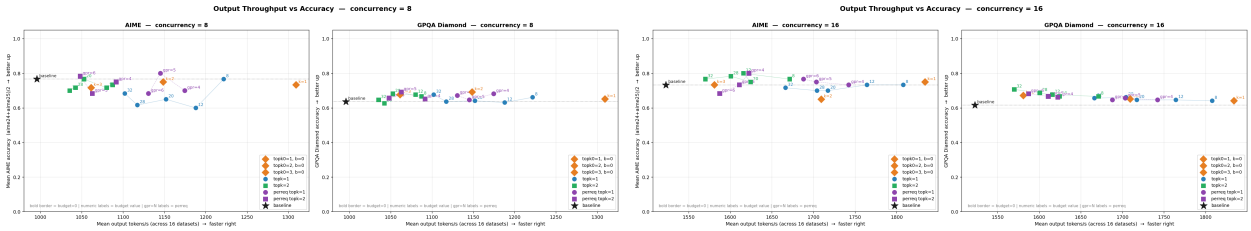
Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	2481	2011	2746		73.3	67.7	89.8	77.0	
k_0=1, b=0	3436	2340	2997	+21.2%	66.7	65.7	90.4	74.3	-2.7
k_0=2, b=0	**3708**	**2416**	**2696**	**+21.8%**	**76.7**	**66.7**	**89.8**	**77.7**	**+0.8**
k_0=3, b=0	2574	2103	2620	+0.8%	70.0	63.1	90.7	74.6	-2.3
b=8, k_0=1	**3804**	**2523**	**3499**	**+35.7%**	**71.7**	**63.1**	**90.5**	**75.1**	**-1.8**
b=12, k_0=1	**3098**	**3214**	**3755**	**+39.1%**	**71.7**	**60.6**	**90.4**	**74.2**	**-2.7**
b=20, k_0=1	**2969**	**3048**	**3554**	**+32.2%**	**75.0**	**63.1**	**92.0**	**76.7**	**-0.3**
b=28, k_0=1	2840	2002	2981	+8.1%	73.3	62.1	90.1	75.2	-1.8
b=32, k_0=1	2890	2348	3310	+18.1%	71.7	63.6	91.1	75.5	-1.5
b=8, k_0=2	3431	2732	3054	+27.3%	68.3	70.7	89.8	76.3	-0.7
b=12, k_0=2	2771	2216	3073	+11.3%	68.3	64.6	90.6	74.5	-2.4
b=20, k_0=2	**2732**	**2174**	**2784**	**+6.2%**	**78.3**	**67.7**	**90.5**	**78.8**	**+1.9**
b=28, k_0=2	3214	2143	2885	+13.9%	73.3	63.6	90.4	75.8	-1.2
b=32, k_0=2	2235	2093	2856	-0.8%	66.7	66.2	89.9	74.2	-2.7

Table 23. **Scenario S3 (no spec. dec., mixed-load) at concurrency c=64.** All methods that appear in the corresponding figure in `figs/lmeval_acc_no_spec_speed_mixed/`. Accuracy is from the mixed-load lmeval sweep ($c/4$ graded reasoning slots interleaved with $3c/4$ ShareGPT/HumanEval filler slots). OTPS is per-task single-load throughput in tokens/s (the mixed-load sweep was –no-baseline; baseline accuracy is borrowed from S1 single-load at the same c — accuracy is invariant to batch composition for the unpruned baseline). **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

D.1.4. SCENARIO 4 — EAGLE3 SPEC. DEC., MIXED-LOAD

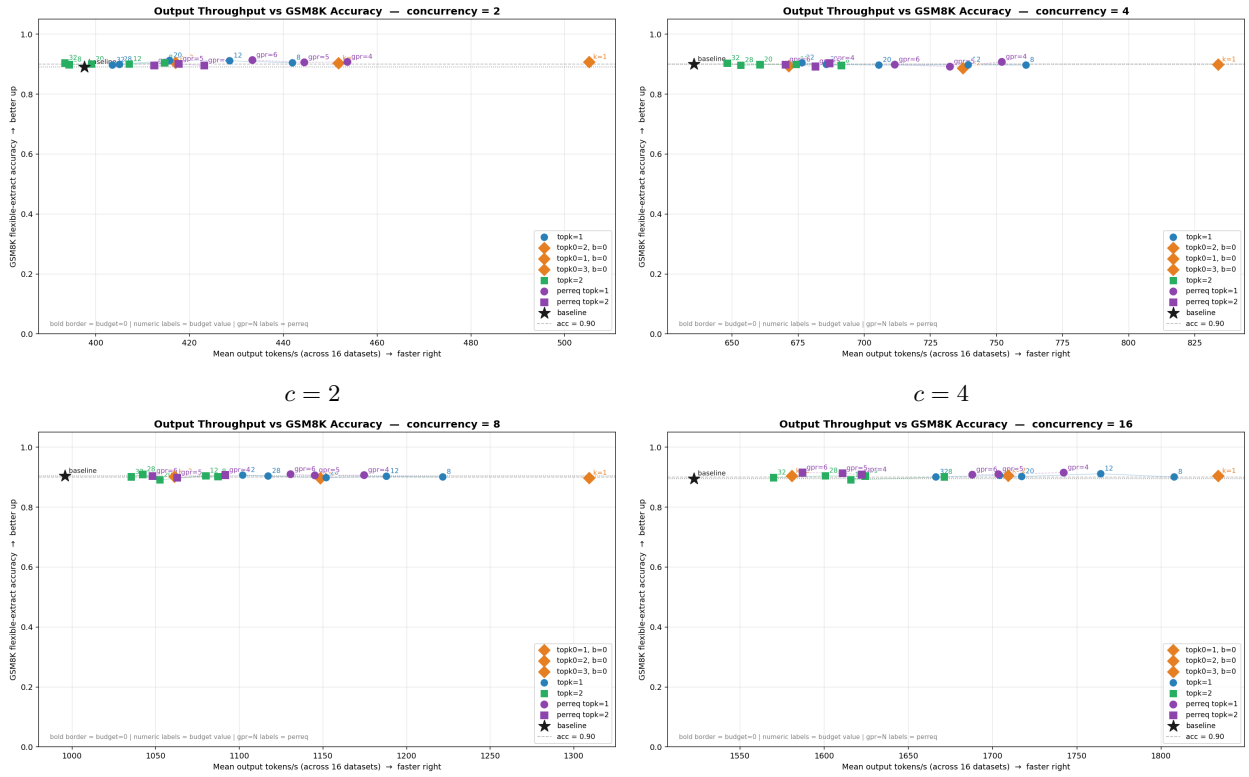


$c = 2$ $c = 4$



$c = 8$ $c = 16$

Figure 16. **Scenario S4 (Eagle3 spec. dec., mixed-load) — OTPS vs. accuracy on AIME∪GPQA, all concurrencies.** Each panel is one concurrency c. Markers: orange diamonds = OEA (b=0); blue/green = global pruning at k_0=1/2; red X = k_0=0 batch-greedy; purple = per-request (when present); black star = unpruned baseline. Numeric labels next to each point are the budget b (or k_0 for OEA, g for per-request).



$c = 2$ $c = 4$

$c = 8$ $c = 16$

Figure 17. **Scenario S4 (Eagle3 spec. dec., mixed-load) — OTPS vs. GSM8K accuracy, all concurrencies.** Each panel is one concurrency c. Markers: orange diamonds = OEA (b=0); blue/green = global pruning at k_0=1/2; red X = k_0=0 batch-greedy; purple = per-request (when present); black star = unpruned baseline. Numeric labels next to each point are the budget b (or k_0 for OEA, g for per-request).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	442	390	474		66.7	71.7	89.0	75.8	
k_0=1, b=0	**572**	**493**	**634**	**+30.0%**	**61.7**	**60.1**	**90.7**	**70.8**	**-5.0**
k_0=2, b=0	**506**	**448**	**549**	**+15.0%**	**68.3**	**66.2**	**90.3**	**74.9**	**-0.9**
k_0=3, b=0	**464**	**413**	**506**	**+5.8%**	**76.7**	**68.7**	**90.4**	**78.6**	**+2.8**
b=8, k_0=1	493	431	543	+12.3%	61.7	63.1	90.4	71.7	-4.0
b=12, k_0=1	481	425	525	+9.6%	71.7	61.1	91.1	74.6	-1.2
b=20, k_0=1	460	410	500	+4.8%	71.7	64.1	91.2	75.7	-0.1
b=28, k_0=1	447	398	484	+1.6%	66.7	62.1	89.9	72.9	-2.9
b=32, k_0=1	445	399	481	+1.4%	66.7	63.1	89.7	73.2	-2.6
b=8, k_0=2	465	410	501	+5.3%	75.0	68.2	90.4	77.9	+2.1
b=12, k_0=2	460	405	494	+4.0%	70.0	67.7	90.0	75.9	+0.1
b=20, k_0=2	444	394	486	+1.3%	78.3	66.7	90.0	78.3	+2.5
b=28, k_0=2	437	385	479	-0.4%	70.0	68.7	89.8	76.2	+0.4
b=32, k_0=2	442	391	474	-0.0%	73.3	70.7	90.4	78.1	+2.3
perreq g=4, k_0=1	**510**	**448**	**553**	**+15.6%**	**65.0**	**66.2**	**90.8**	**74.0**	**-1.8**
perreq g=5, k_0=1	494	439	539	+12.6%	68.3	61.1	90.6	73.3	-2.4
perreq g=6, k_0=1	486	426	533	+10.6%	66.7	63.6	91.4	73.9	-1.9
perreq g=4, k_0=2	**478**	**420**	**516**	**+8.2%**	**70.0**	**68.2**	**89.5**	**75.9**	**+0.1**
perreq g=5, k_0=2	**465**	**416**	**505**	**+6.1%**	**78.3**	**66.2**	**90.1**	**78.2**	**+2.4**
perreq g=6, k_0=2	461	410	497	+4.7%	73.3	64.1	89.5	75.7	-0.1

Table 24. **Scenario S4 (Eagle3 spec. dec., mixed-load) at concurrency c=2.** All methods that appear in the corresponding figure in `figs/lmeval_acc_eagle3_speed_mixed_dataset/`. Accuracy is from the spec-dec mixed-load lmeval sweeps; OTPS is per-task single-load throughput from the Eagle3 speed run. Baseline accuracy at $c \in \{2, 4\}$ is borrowed from the S2 eagle3 single-load sweep (unpruned baseline accuracy is invariant to batch composition). **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	711	619	751		75.0	62.6	90.0	75.9	
k_0=1, b=0	**955**	**824**	**1006**	**+33.8%**	**66.7**	**66.2**	**89.8**	**74.2**	**-1.6**
k_0=2, b=0	**838**	**726**	**885**	**+17.7%**	**75.0**	**65.7**	**88.6**	**76.4**	**+0.6**
k_0=3, b=0	759	665	805	+7.1%	70.0	67.7	89.3	75.7	-0.2
b=8, k_0=1	**875**	**741**	**903**	**+21.0%**	**70.0**	**63.6**	**89.7**	**74.4**	**-1.4**
b=12, k_0=1	852	719	884	+18.0%	65.0	65.2	89.8	73.3	-2.6
b=20, k_0=1	778	684	847	+11.0%	68.3	63.1	89.7	73.7	-2.2
b=28, k_0=1	771	685	824	+9.5%	68.3	63.6	89.9	74.0	-1.9
b=32, k_0=1	765	667	805	+7.5%	65.0	68.7	90.4	74.7	-1.2
b=8, k_0=2	**775**	**683**	**829**	**+9.9%**	**80.0**	**64.6**	**89.5**	**78.1**	**+2.2**
b=12, k_0=2	760	661	813	+7.4%	73.3	63.6	89.9	75.6	-0.2
b=20, k_0=2	744	646	792	+4.9%	66.7	63.6	89.8	73.4	-2.5
b=28, k_0=2	726	643	781	+3.3%	68.3	68.2	89.6	75.4	-0.5
b=32, k_0=2	727	636	750	+1.5%	78.3	67.7	90.3	78.8	+2.9
perreq g=4, k_0=1	**850**	**741**	**915**	**+20.4%**	**68.3**	**66.2**	**90.8**	**75.1**	**-0.8**
perreq g=5, k_0=1	**834**	**709**	**885**	**+16.6%**	**75.0**	**69.7**	**89.2**	**78.0**	**+2.1**
perreq g=6, k_0=1	800	703	846	+12.9%	71.7	66.7	89.8	76.1	+0.2
perreq g=4, k_0=2	774	674	817	+8.8%	78.3	64.1	90.3	77.6	+1.7
perreq g=5, k_0=2	**777**	**673**	**814**	**+8.8%**	**78.3**	**68.2**	**89.2**	**78.6**	**+2.7**
perreq g=6, k_0=2	**759**	**660**	**796**	**+6.4%**	**81.7**	**66.7**	**89.8**	**79.4**	**+3.5**

Table 25. **Scenario S4 (Eagle3 spec. dec., mixed-load) at concurrency c=4.** All methods that appear in the corresponding figure in `figs/lmeval_acc_eagle3_speed_mixed_dataset/`. Accuracy is from the spec-dec mixed-load lmeval sweeps; OTPS is per-task single-load throughput from the Eagle3 speed run. Baseline accuracy at $c \in \{2, 4\}$ is borrowed from the S2 eagle3 single-load sweep (unpruned baseline accuracy is invariant to batch composition). **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	1135	895	1199		76.7	63.6	90.4	76.9	
k_0=1, b=0	**1528**	**1290**	**1598**	**+36.8%**	**73.3**	**65.2**	**89.7**	**76.1**	**-0.8**
k_0=2, b=0	**1375**	**1032**	**1366**	**+16.8%**	**75.0**	**69.2**	**89.6**	**77.9**	**+1.0**
k_0=3, b=0	1214	1031	1262	+8.6%	71.7	67.7	90.1	76.5	-0.4
b=8, k_0=1	**1415**	**1209**	**1497**	**+27.6%**	**76.7**	**66.2**	**90.1**	**77.6**	**+0.7**
b=12, k_0=1	1398	1070	1462	+21.7%	60.0	63.1	90.3	71.1	-5.7
b=20, k_0=1	1336	1047	1399	+17.1%	65.0	64.1	89.8	73.0	-3.9
b=28, k_0=1	1289	1104	1348	+15.9%	61.7	63.6	90.4	71.9	-5.0
b=32, k_0=1	1166	1082	1341	+11.2%	68.3	67.7	90.7	75.6	-1.3
b=8, k_0=2	1179	1001	1332	+8.8%	73.3	66.7	90.1	76.7	-0.2
b=12, k_0=2	1152	1001	1302	+7.0%	71.7	67.7	90.4	76.6	-0.3
b=20, k_0=2	1120	1045	1261	+6.1%	76.7	68.2	89.1	78.0	+1.1
b=28, k_0=2	1211	1029	1265	+8.6%	71.7	62.6	90.9	75.1	-1.8
b=32, k_0=2	1189	939	1248	+4.5%	70.0	64.6	90.1	74.9	-2.0
perreq g=4, k_0=1	1351	1150	1399	+20.8%	70.0	68.2	90.7	76.3	-0.6
perreq g=5, k_0=1	**1225**	**1020**	**1400**	**+12.9%**	**80.0**	**64.6**	**90.6**	**78.4**	**+1.5**
perreq g=6, k_0=1	1303	1020	1359	+14.0%	68.3	67.2	91.0	75.5	-1.4
perreq g=4, k_0=2	1228	1052	1307	+11.1%	75.0	65.2	90.7	76.9	+0.1
perreq g=5, k_0=2	1232	991	1286	+8.7%	68.3	69.2	89.8	75.8	-1.1
perreq g=6, k_0=2	1119	970	1236	+3.0%	78.3	65.7	90.4	78.1	+1.2

Table 26. **Scenario S4 (Eagle3 spec. dec., mixed-load) at concurrency c=8.** All methods that appear in the corresponding figure in `figs/lmeval_acc_eagle3_speed_mixed_dataset/`. Accuracy is from the spec-dec mixed-load lmeval sweeps; OTPS is per-task single-load throughput from the Eagle3 speed run. Baseline accuracy at $c \in \{2, 4\}$ is borrowed from the S2 eagle3 single-load sweep (unpruned baseline accuracy is invariant to batch composition). **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).

Method	OTPS (tokens/s) AIME	GPQA	GSM8K	avg. Δ % OTPS	Accuracy (%) AIME	GPQA	GSM8K	avg. acc. (%)	avg. Δ acc
baseline	1584	1332	1858		73.3	61.6	89.5	74.8	
k_0=1, b=0	**1954**	**1650**	**2365**	**+25.0%**	**75.0**	**64.1**	**90.4**	**76.5**	**+1.7**
k_0=2, b=0	2034	1734	1947	+19.7%	65.0	65.2	90.4	73.5	-1.3
k_0=3, b=0	1662	1393	1941	+4.6%	73.3	67.2	90.4	77.0	+2.1
b=8, k_0=1	1884	1641	2268	+21.4%	73.3	64.1	90.1	75.8	+1.0
b=12, k_0=1	2084	1530	2332	+24.5%	73.3	64.6	91.1	76.4	+1.5
b=20, k_0=1	1815	1509	2155	+14.8%	70.0	64.6	90.2	75.0	+0.1
b=28, k_0=1	2006	1496	2090	+17.1%	70.0	66.2	90.6	75.6	+0.8
b=32, k_0=1	1746	1480	2075	+11.0%	71.7	65.7	90.1	75.8	+1.0
b=8, k_0=2	**1759**	**1689**	**2143**	**+17.1%**	**76.7**	**66.7**	**90.1**	**77.8**	**+3.0**
b=12, k_0=2	1688	1450	2043	+8.5%	80.0	67.7	89.1	78.9	+4.1
b=20, k_0=2	1893	1427	1995	+11.3%	75.0	66.7	90.4	77.3	+2.5
b=28, k_0=2	**1662**	**1379**	**1962**	**+4.8%**	**78.3**	**68.7**	**90.4**	**79.2**	**+4.3**
b=32, k_0=2	1648	1378	1946	+4.1%	76.7	70.7	89.8	79.1	+4.2
perreq g=4, k_0=1	1794	1534	2150	+14.7%	73.3	64.6	91.5	76.5	+1.7
perreq g=5, k_0=1	1774	1465	2106	+12.0%	75.0	65.7	91.0	77.2	+2.4
perreq g=6, k_0=1	**1983**	**1664**	**2064**	**+19.6%**	**76.7**	**64.6**	**90.8**	**77.4**	**+2.6**
perreq g=4, k_0=2	**1904**	**1448**	**2002**	**+12.1%**	**80.0**	**66.2**	**90.8**	**79.0**	**+4.2**
perreq g=5, k_0=2	1683	1412	1982	+6.3%	73.3	66.7	91.3	77.1	+2.3
perreq g=6, k_0=2	1679	1390	1879	+3.6%	68.3	68.2	91.5	76.0	+1.2

Table 27. **Scenario S4 (Eagle3 spec. dec., mixed-load) at concurrency c=16.** All methods that appear in the corresponding figure in `figs/lmeval_acc_eagle3_speed_mixed_dataset/`. Accuracy is from the spec-dec mixed-load lmeval sweeps; OTPS is per-task single-load throughput from the Eagle3 speed run. Baseline accuracy at $c \in \{2, 4\}$ is borrowed from the S2 eagle3 single-load sweep (unpruned baseline accuracy is invariant to batch composition). **Bold** method labels mark the Pareto-optimal frontier on (avg. Δ % OTPS, avg. accuracy). "–" indicates the configuration was not collected at this (c).