Cache-Coherent Resampling for Efficient Test-Time Scaling in LLM Reasoning via Adaptive Sequential Monte Carlo

Ke Wang [1] Zehao Yu [2] Luwei Wang [1] Yongchao Huang [3]

[1]School of Informatics, University of Edinburgh [2]Department of Computer Science, UCL [3]School of Natural and Computing Science, University of Aberdeen

Why ASMC?

Power-shaped trajectory sampling can improve LLM reasoning, but autoregressive Metropolis–Hastings is serial and creates severe p95 tail latency. ASMC makes this test-time scaling path parallel by evolving weighted particles in a batch.

ASMC in 4 Steps

ASMC decodes particles in parallel, reweights them toward $\pi(x) \propto p_\theta(x)^{\alpha^\star}$, and resamples when ESS drops. Cache-coherent resampling applies the ancestor map directly to KV caches, avoiding $O(NL^2)$ prefix replay. Adaptive rules exit early on high top-answer mass and escalate to larger N for hard prompts.

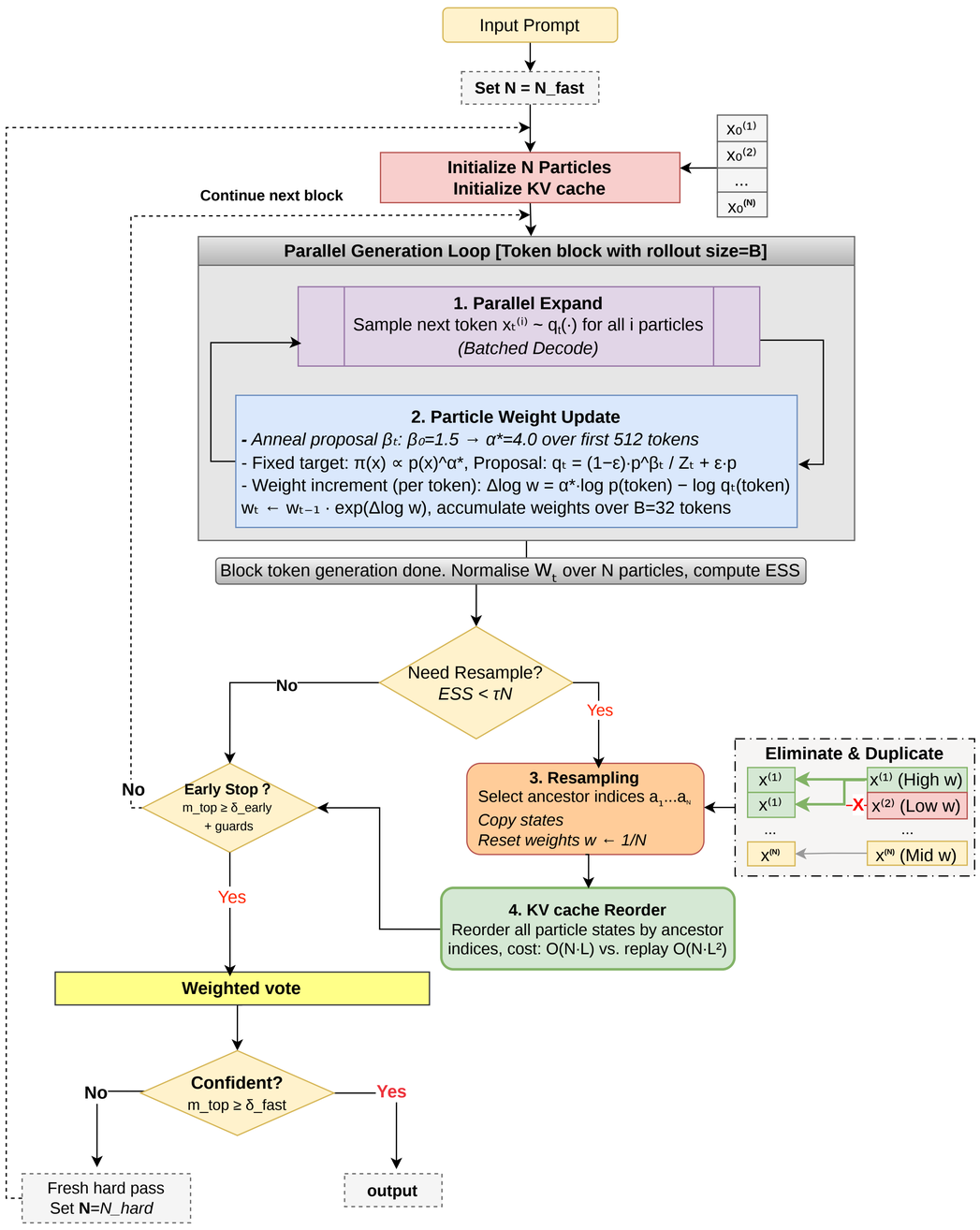

Figure 1. **ASMC pipeline.** Parallel particles are reweighted toward $\pi(x) \propto p(x)^{\alpha^\star}$ and resampled via cache-coherent KV-cache reordering ($O(NL)$). Easy problems exit early; hard problems use larger N.

Adaptive Compute Allocation

ASMC spends compute in proportion to difficulty, via **top-answer mass** $m_{\text{top}} = \max_a \sum_{i:\,\text{ans}(x^{(i)})=a} \tilde{w}_t^{(i)}$:

- **Early exit (easy):** stop when $m_{\text{top}} \geq \delta_{\text{early}}$ and guards hold.
- **Escalation (hard):** if the fast pass ends with $m_{\text{top}} < \delta_{\text{fast}}$, restart with a larger population N_{hard}.
- **Budget cap:** stop when cumulative C_{int} exceeds $C^\star$, enforcing an interaction-budget cap; wall-clock p95 remains the deployment metric.

The fixed-target SMC core targets the power-shaped distribution; adaptive early-exit/escalation rules are empirical latency heuristics.

Latency-Aware Compute Accounting

Prefill/teacher-forcing is highly parallel; autoregressive decode is sequential — so a single FLOP proxy is **miscalibrated across methods**. We log attention interactions, separating cached decode from prefill/replay,

$$C_{\text{int}} = \underbrace{\sum_{k\in\text{decode}} B_k L_k}_{O(L)\ \text{decode}} + \underbrace{\sum_{k\in\text{prefill}} B_k \frac{L_k(L_k+1)}{2}}_{O(L^2)\ \text{replay}},$$

using C_{int} only for compute-matched *selection*, while **wall-clock p95 latency** is the primary deployment metric.

Cache-Coherent Resampling for Transformers

After resampling, particle i must inherit its ancestor's *entire* state (KV caches at all layers, position IDs, masks, buffers). Two naive options both fail at scale:

- **Prefix replay** (compute-bound): recompute inherited prefixes $\Rightarrow O(NL^2)$ attention work.
- **State copy** (memory-bound): overwrite state $\Rightarrow O(NLD_{\text{model}})$ data movement per event.

Our primitive: apply the ancestor map as a single `gather` over the particle axis of every per-layer KV tensor (and all particle-bound state) — regular, bandwidth-friendly, *no token-by-token reconstruction*.

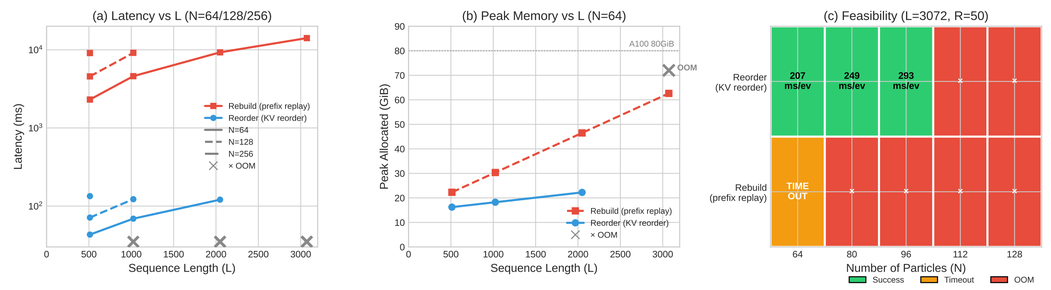

Figure 2. **System cost of one resampling event** (Qwen2.5-Math-7B, bf16). Reorder vs. Rebuild: **(a)** event latency vs. L; **(b)** peak memory at N=64; **(c)** feasibility at L=3072 over 50 events. Reorder is **53–77× faster**; Rebuild times out at N=64 and OOMs for $N \geq 80$, while Reorder remains feasible through N=96.

Results: Accuracy–p95 Frontier on MATH500

On MATH500, ASMC-adapt reaches 80.6% accuracy at p95 = 73.7 s under 128× compute, while sequential MCMC reaches p95 = 1317.5 s. KV reorder is 53–77× faster than prefix replay and remains feasible through N=96 at L=3072.

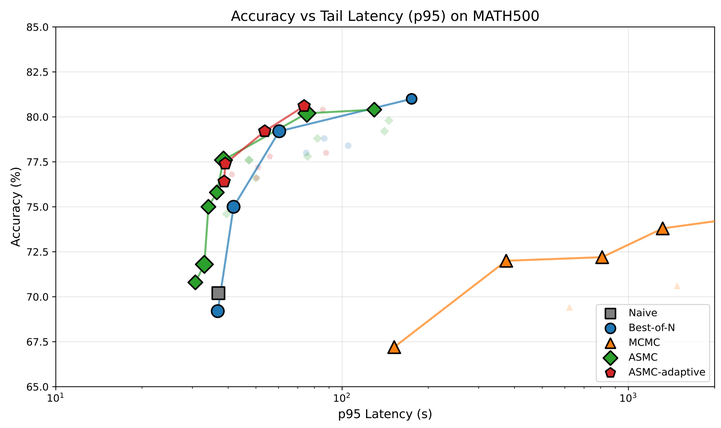

Figure 3. **Accuracy vs. p95 latency.** Best config per method under each compute cap $C^\star$ (C_0=0.38M). ASMC occupies the high-accuracy / low-tail region; MCMC's tail explodes.

Budget	Method	Acc.%	p50 (s)	p95 (s)
8×	Best-of-2	75.0	12.5	41.8
	MCMC	67.2	34.0	152.1
	ASMC(N=4)	71.8	12.5	**33.0**
128×	Best-of-4	79.2	14.5	60.3
	MCMC	73.8	211.6	1317.5
	ASMC(N=64)	80.2	19.8	75.4
	ASMC-adapt (N=128)	**80.6**	20.9	**73.7**

Table 1. **Selected MATH500 results.** ASMC reaches 80.6% at p95 = 73.7 s, vs. MCMC's 1317.5 s tail at the same budget.

Particle Collapse: the Dominant Failure Mode

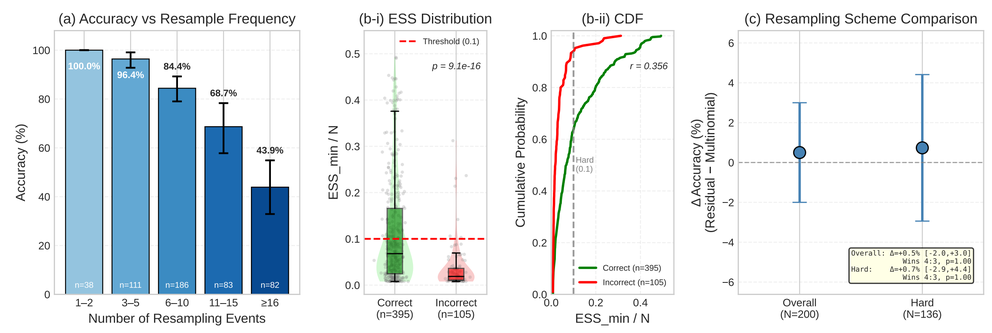

Figure 4. **Diagnosing collapse.** (a) accuracy drops $100\% \to 43.9\%$ as resampling events grow $1\text{–}2 \to \geq 16$; (b) incorrect problems have lower $\text{ESS}_{\min}/N$ ($p < 10^{-15}$); (c) residual vs. multinomial resampling differ by only $\Delta = +0.5\%$ (n.s.).

Takeaway

Tail latency, not just compute, is the bottleneck. Cache-coherent ASMC improves the accuracy–p95 frontier, while particle collapse remains the main failure mode.

ICML 2026

Code: github.com/Vicky-0256/**ASMC**

Contact: K.Wang-72@sms.ed.ac.uk