EXACT WARM-STATE REUSE FOR HYBRID DIFFUSION LANGUAGE MODELS: CONTRACTS, SPEEDUPS, AND FAILURE BOUNDARIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Hybrid diffusion language models repeatedly revise a small set of tokens while most context is already stable, yet conventional inference can replay the entire sequence at every denoising step. Reuse is harder than causal prefix caching because a hybrid model combines paged attention state with recurrent Gated Delta Network (GDN) state, noncausal masks, absolute positions, and edit-dependent validity. We formulate a fail closed incremental execution contract over regions. A cache entry binds physical key-value pages and layer-local recurrent frontiers to token, position, model, adapter, mask-contract, and parent version identities. Our accepted GPU fast path is deliberately narrower than the general contract: a stable contiguous prefix and an active contiguous suffix, for which the attention/MLP active rows and the ordered GDN continuation are the same suffix. On one frozen HybridDiffusion-2B checkpoint and one NVIDIA A30, warm reuse reduces measured serving-route CUDA latency from 1,055.27 to 137.39 ms (7.68×) for a 2,048-token prefix, 64-token suffix, and four diffusion steps, with bit-identical output hashes and 18.2% lower route-local peak memory. Across 36 controlled shapes, median warm speedup is 3.03×; 100 randomized region-contract cases pass their recorded correctness gates. We also report contrary evidence: natural generation is 11.99% slower, a KV-only route fails physical-page validation, and a four-region GPU route faults. We therefore claim an exact contiguous warm-state mechanism and a measured architecture envelope, not general fragmented execution, universal serving speedup, or cross-model transfer.

1 INTRODUCTION

Discrete diffusion language models replace left-to-right generation with iterative denoising, enabling parallel token revision and flexible conditioning (Austin et al., 2021; Li et al., 2022; Sahoo et al., 2024; Han et al., 2023). Hybrid models add a further systems challenge: attention layers expose a paged key–value (KV) cache, while recurrent layers carry hidden state across token order. In editing and structured generation, a denoising step may change only a suffix or a few regions, but the serving stack typically executes all token rows again.

This is not solved by ordinary prefix caching. PagedAttention (Kwon et al., 2023) and RadixAttention (Zheng et al., 2024) reuse causal KV pages under a single textual history. A hybrid diffusion request additionally needs an exact recurrent frontier, the mask under which stable state was constructed, absolute rotary positions, and dependency versions. Reusing stale recurrent state can silently change outputs; rebuilding stable query, attention, and MLP rows gives up most of the prospective saving.

We study the following question: *when a diffusion step edits only part of a hybrid sequence, which model state can be reused exactly, and what speedup does that exact contract actually deliver?* Our answer, EXACTREGION, treats inference as demand-driven incremental computation (Hammer et al., 2014) over a region DAG. Stable regions are read-only state; edits invalidate their declared descendants; and recurrent computation replays conservatively from the first invalidated position. The serving-accepted result in this paper is the contiguous prefix–suffix specialization. General Region-DAG logic is evaluated as a contract and controlled characterization, not claimed as a reliable

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

fragmented GPU serving path. Unlike approximate diffusion caches (Ma et al., 2025; Nguyen-Tri et al., 2026a;b), the acceptance oracle is exact output identity to a declared reference contract.

Our contributions are:

- an explicit region-state identity that joins physical KV locations, per-layer GDN frontiers, absolute positions, attention contracts, and parent versions;
- an exact invalidation and conservative replay rule for declared region dependencies, with 100 randomized contract cases and no learned state merging;
- a serving-stack implementation of the contiguous prefix–suffix specialization, with fail-closed lifecycle transitions and allocator-authoritative physical page-table evidence; and
- a measurement-based characterization: up to 14.92× in the recorded controlled context sweep and 7.68× in a sealed serving-route case, together with negative results that define the supported envelope.

2 Related Work

Diffusion and hybrid language models. D3PM (Austin et al., 2021), Diffusion-LM (Li et al., 2022), SSD-LM (Han et al., 2023), masked diffusion language modeling (Sahoo et al., 2024), and LLaDA (Nie et al., 2025) establish iterative text generation as a credible alternative to autoregression. FLARE (Zhu et al., 2026) converts a hybrid-attention autoregressive model into a diffusion model that interleaves full attention and Gated Delta Networks (Yang et al., 2025). We do not change its weights, training objective, or denoising rule; we change how a valid repeated computation is scheduled and restored.

LLM serving and prefix reuse. PagedAttention virtualizes KV storage (Kwon et al., 2023); SGLang's RadixAttention shares prefixes across structured programs (Zheng et al., 2024); FlashAttention and FlashInfer optimize attention execution and serving layouts (Dao et al., 2022; Ye et al., 2025). These systems motivate physical-page reuse, but a KV-only identity does not describe recurrent hybrid state or a diffusion mask. Speculative decoding (Leviathan et al., 2023) avoids some target-model work through a draft-and-verify protocol; our mechanism instead reuses exact state from the same model after localized edits.

Hybrid and recurrent prefix caching. Sparse Prefix Caching stores exact recurrent checkpoints and recomputes from the deepest retained boundary (Shirokikh & Nikolenko, 2026). Tail-Replay instead keeps exact KV but approximately reconstructs recurrent state from a short recent suffix (Liu et al., 2026b). HYPIC composes segment transition operators and repairs attention at segment seams for position-independent hybrid-model caching (Liu et al., 2026a). Fixed State, Long Reach trains block-diffusion models around a constant-size state interface (Singh et al., 2026). These systems establish that hybrid caching is a broader field. Our narrower contribution is an exact, allocator-validated joint KV/GDN lifecycle for repeated execution in an existing hybrid diffusion stack; we do not claim the first hybrid cache or superiority without direct experiments.

Diffusion caching and incremental computation. dKV-Cache delays or conditions KV refresh across denoising steps (Ma et al., 2025); Elastic-Cache adapts refresh by attention and layer dynamics (Nguyen-Tri et al., 2026a); and Flash-dLLM combines an I/O-aware cache with draft-and-verify decoding (Nguyen-Tri et al., 2026b). These methods target accuracy–efficiency trade-offs across public diffusion models, whereas our accepted path requires exact identity to its declared full-replay oracle. DeepCache, Learning-to-Cache, and TeaCache reuse temporally similar features in image or video diffusion (Ma et al., 2024b;a; Liu et al., 2025). The dependency closure resembles self-adjusting computation (Hammer et al., 2014), but GPU serving adds physical allocators, paged masks, recurrent frontiers, and asynchronous timing to the correctness boundary. Appendix I gives a claim-level comparison.

3 Mathematical Architecture and Method

Region contract. Let a sequence of length n be partitioned into ordered regions $\mathcal{R} = \{R_i\}_{i=1}^{m}$. Each region carries interval $[s_i, e_i)$, status $a_i \in \{\text{stable}, \text{active}\}$, version v_i, and declared parents P_i.

The directed graph $G = (\mathcal{R}, E)$ is acyclic. An attention edge is legal only when permitted by the contract:

$$M(q,k) = 1 \iff R(k) = R(q) \ \vee \ R(k) \in \mathrm{Anc}(R(q)), \tag{1}$$

with the additional rule that stable queries cannot observe active keys. Stable state must have been constructed under the same M.

For logical active positions A, we gather hidden rows but preserve original indices p_A (rather than renumbering from zero), which keeps rotary embeddings (Su et al., 2024) invariant. In an attention layer,

$$Q_A = H_A W_Q, \quad K = [K_S; K_A], \quad V = [V_S; V_A], \quad O_A = \mathrm{Attn}(Q_A, K, V; M_A). \tag{2}$$

No Q_S, stable attention output, or stable MLP output is computed.

Invalidation and recurrence. If regions E_0 are edited, the exact logical invalidation set is

$$\mathcal{I}(E_0) = E_0 \cup \bigcup_{R \in E_0} \mathrm{Desc}_G(R). \tag{3}$$

The GDN layer is recurrent in textual order. Until independent state forking is available, we therefore choose the conservative boundary $b = \min_{R_i \in \mathcal{I}} s_i$ and define the recurrent replay set $G_b = [b, n)$. Exact recurrent equivalence requires the same ordered hidden-row continuation over G_b, not merely the sparse set A. In the accepted contiguous path, $A = G_b = [b, n)$, so active attention/MLP rows and GDN replay rows coincide. For internal or fragmented edits, generally $A \neq G_b$; such cases characterize the intended contract and replay cost, but the paper does not claim accepted arbitrary-fragment GPU execution. This distinction reconciles Eq. 2 with the recurrent proof in Appendix A.

Exact cache identity. For request r, layer ℓ, and boundary b, a reusable state is keyed by

$$\begin{aligned}\kappa = (&r, \mathrm{slot}, \mathrm{generation}, \ell, b, \mathrm{modelRev}, \mathrm{adapterRev},\\ &\mathrm{contract}, \{(i, v_i, P_i)\}, H(x_{<b}), H(p_{<b})).\end{aligned} \tag{4}$$

The value contains canonical physical KV locations and cloned convolutional/recurrent GDN tensors. A hit is accepted only if parent versions match, the int64 prefix page table is contiguous and in range, and it equals the authoritative allocator view. Thus a token match alone is insufficient.

Lifecycle. The first exact request does prefix $[0, b)$ in its canonical row shape then snapshots KV and GDN state and publishes a result scoped proof. This publication validates before a suffix enters decode. The next steps are to restore the frontier, attach the physical prefix, submit active rows and commit a new version. Zero boundary requests are replayed in full. We find that the canonical prefix construction is important. BF16 kernels can vary with row shape, so a frontier created within a monolithic n-row forward is not necessarily the exact frontier of the segmented reference.

4 Repository-Based Implementation

The contract was built as an evaluation-serving extension on top of a Qwen3.5 hybrid diffusion stack. The implementation alters execution metadata and cache lifecycle without modifying checkpoint, training loss or precision rule.

Position and dependency managers. `RegionPositionMap` constructs one immutable mapping from regions to absolute token positions. `gather_active_positions` and `scatter_active_states` validate shape, uniqueness, and bounds. `DependencyGraph` topologically computes parents, ancestors, descendants, and Eq. 3; version checks require every declared parent pair to match.

Scheduler and attention. The request object owns an immutable execution specification and runtime plan. The batch scheduler submits only gathered rows and resolves their prefix locations from the real request-to-token pool. `ForwardBatch` carries original query positions and creates an active-query by total-KV mask. FlashInfer selects a custom paged backend, so active queries read stable physical pages without materializing stable queries. Full-attention projections and MLPs consequently see only submitted rows.

Table 1: Headline timing boundary. All post-route checks remain mandatory acceptance gates, but are not included in the route-latency claim.

Stage	Headline timer	Recorded evidence
Prepare, restore, forward, publication	included	CUDA events and component phases
Terminal synchronization	included once	synchronization-count gate
Page-table/output equality transfer	after timer	mandatory post-route acceptance
Checkpoint/provenance pre-flight	outside request	SHA-256 and architecture gate

Table 2: Accepted evidence collections. Collections overlap and are not pooled. "Exact" denotes output-hash identity to the corresponding full reference.

Collection	Cases	Timed reps.	Exact	Fallback/recovery
Sealed serving-route case	1	10	yes	0/0
Headline workload set	9	10	yes	0/0
Broad workload grid	36	5	yes	0/0
Long-context extension	4	20	yes	0/0
Random Region-DAG validation	100	n/a	yes	0/0

GDN frontier cache. Each linear-attention layer restores a snapshot indexed by Eq. 4. Snapshot values are cloned to prevent aliasing. Publication occurs only after the real KV/GDN copy succeeds; all normal and output-processing paths consume the same proof. A restore fails closed on slot generation, token hash, position hash, contract, model revision, adapter revision, parent version, tensor layout, or physical-location mismatch.

Listing 1: Accepted contiguous warm execution (simplified).

```
plan = dependency_graph.plan(edits)
if not cache.matches(plan.identity): return full_replay(plan)
if not plan.is_contiguous_suffix(): return UNACCEPTED_GENERAL_PATH
assert canonical_pages(plan.prefix) == cache.pages
restore_all_gdn_layers(cache.frontier)
rows = gather(hidden, range(plan.boundary, n))
# For the accepted path: attention/MLP active rows == ordered GDN replay rows.
out = model.forward(rows, custom_paged_mask(plan))
assert no_fallback and no_recovery
return scatter(out, range(plan.boundary, n))
```

The implementation deliberately keeps three restrictions. First, multi-parent recurrent merging is replayed rather than learned. Second, batch size one is the accepted latency path. Third, unsupported routes are registry entries that fail before inference rather than aliases mislabeled as baselines.

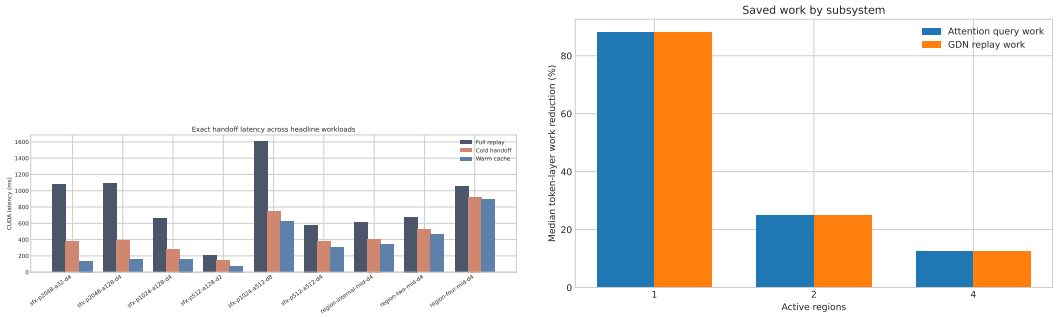
5 Experimental Results

Protocol. We freeze one HybridDiffusion-2B checkpoint (24 layers, hidden size 2,048, BF16 tensors) and run on one NVIDIA A30 with CUDA 12.8, PyTorch 2.9.1, BF16, tensor parallelism 1, batch size 1. Serving routes timing uses CUDA events, three warmups and ten measured repetitions without debug synchronization. Full replay reconstructs the canonical prefix and live suffix at each diffusion step . Warm execution restores a previously established frontier . Every accepted row requires exact output hashes, preserved positions, no fallback and recovery, and physical-page validity. The headline figure is route latency, not concurrent-server request latency: device-to-host acceptance checks that do not influence execution are performed after the single timed terminal synchronization and are reported as post-route verification. Peak memory is route-local. Additional randomized contract testing covers 100 region graphs; all pass their recorded gates. Table 1 and Appendix E make the measurement boundary explicit.

Table 3: Serving-route CUDA latency on A30 (BF16, TP=1, batch 1). Each row is exact-output equivalent and fallback/recovery free. "Rows" counts submitted query rows across four diffusion steps.

Prefix/active	Full (ms)	Ours (ms)	Speedup	Reduction	Rows (full/ours)
1,024/128	648.84	156.43	4.15×	75.89%	4,608/512
2,048/32	1,080.58	141.84	7.62×	86.87%	8,320/128
2,048/64[†]	1,055.27	137.39	7.68×	86.98%	–

[†]Independently sealed serving-route benchmark; 8,192 reused prefix positions were proven by canonical page-table identity. It is not pooled with the two cache-system comparison rows.



(a) Full, cold, and warm latency.

(b) Attention and GDN work reduction.

Figure 1: Serving-route latency and mechanism-level accounting for the nine accepted headline workloads. Cold construction is separated from steady-state warm reuse; this is not a concurrent-server throughput experiment.

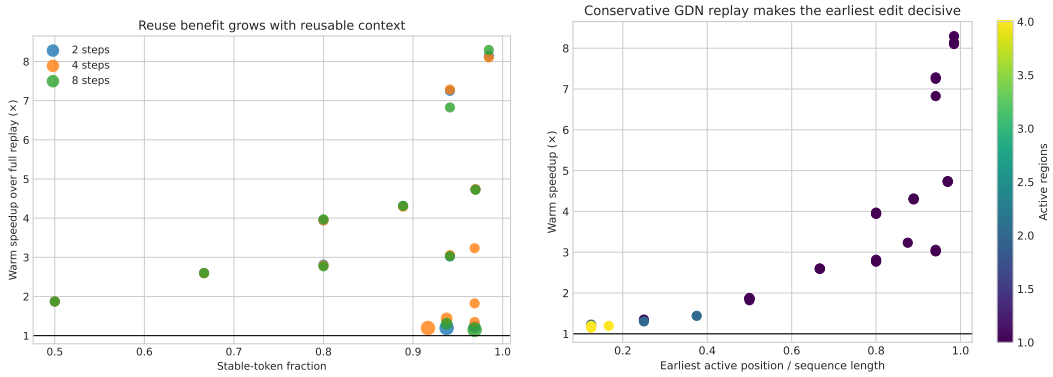
Headline route latency and memory. Table 3 shows that the benefit is driven by eliminated token-layer work. For the independently sealed 2,048/64/four-step case, warm execution is 7.68× faster and lowers peak memory from 15.06 to 12.31 GiB (18.2%). The measured 95% interval for the full-minus-warm latency improvement is 917.61–918.64 ms. In the two accepted cache-system comparison cases, query rows fall from 4,608 to 512 and from 8,320 to 128, respectively.

Where the architecture helps. Across the broad 36-case grid, median warm speedup is 3.03×, median latency reduction is 66.94%, and the best case is 8.30×. The median memory reduction is 5.09%. Table 4 and Figure 2 reveal that stable-token fraction is necessary but not sufficient: context scaling and late edits expose more stable work, whereas dense or early edits force longer recurrent replay. Fragmentation additionally pays mask, gather, and conservative-replay overhead. Thus the execution frontier, not token count alone, is the architectural unit that predicts latency.

Long-context scaling. Table 5 shows that the reuse advantage increases when the stable prefix dominates execution. The four-case extension reaches a 13.20× median warm speedup, 92.40% median latency reduction, and approximately 30.9% median peak-memory reduction. These are stress-test results under the supported contiguous prefix–suffix contract, not a claim that all 4K region layouts achieve the same gain. The complete scaling plot is provided in Appendix 3.

Table 6: Mechanism ablation (median over the recorded cases). Cold handoff builds state; warm execution reuses both attention and recurrent state.

Route	Latency (ms)	Speedup	Work saved
Full replay	1,073.03	1.00×	0%
Cold handoff	521.05	2.35×	66.67%
Warm region state	289.13	4.06×	88.89%



(a) Speedup grows with stable fraction.

(b) Late edits reduce recurrent replay.

Figure 2: Two predictors of reuse value. Stable fraction controls removable token-layer work, while the earliest edited position controls the conservative GDN replay frontier.

Table 4: Architecture sweeps. Medians summarize distinct workload cases, not repeated timings.

Sweep	Cases	Med. speedup	Best	Med. work saved
Prefix boundary	12	2.09×	4.86×	69.81%
Edit density	12	1.29×	4.83×	21.39%
Fragmentation	5	1.22×	1.82×	17.19%
Context scaling	16	4.09×	14.92×	92.48%
Long-context extension	4	13.20×	14.01×	–

Cold versus warm state. Table 6 separates frontier creation from steady-state reuse. Cold handoff is 2.35× faster than replay in the recorded ablation, while warm region state is 4.06×. This distinction prevents one-time snapshot cost from being hidden in a steady-state claim. A small deterministic router selected the fastest measured valid route on its held-out grid, but always-warm tied the oracle there; we therefore do not claim routing as a learning contribution.

6 Limitations and Negative Results

The strongest limitation is external validity. All accepted method latency results use one 2B checkpoint, one A30, TP=1, and batch size one. Public Qwen3.5-2B autoregressive and SDAR-1.7B diffusion runs provide hardware context only: neither executes ExactRegion. Consequently, the paper demonstrates a mechanism and scaling envelope, not model-agnostic transfer.

The natural generation comparison is also important: exact handoff is 11.99% slower than normal self-speculative generation in that workload (bootstrap interval $[-15.29, -7.06]$ percentage points under the artifact's sign convention). Across its 15 prompts, answer token IDs and decoded text are identical between routes; this is a small paired identity check, not a downstream quality estimate. The architecture pays off when stable state is reused across edits or diffusion steps and can lose when reuse is scarce or set-up dominates. The safe KV-only route did not validate because of invalid live physical prefix locations. No KV-only performance number is reported. Similarly, the four-fragment cached route threw an illegal CUDA memory access prior to its first accepted record. These failures motivate a conservative route policy and disallow any claim that arbitrary fragmented GPU execution is complete.

Finally, exact output hashes enforce equivalence to the selected reference, not independent task quality. The controlled full-replay oracle uses the same declared Region-DAG mask as the cached path, which isolates execution reuse but does not itself prove equivalence to the checkpoint's native generation mask. The separate 15-prompt normal vs exact identity check is just supporting evidence. No downstream accuracy improvement is expected because numerically the computation is supposed

Table 5: Long-context stress test with a 4,096-token reusable prefix (20 repeated timings per deterministic case). All rows pass exact-output and runtime-safety gates.

Active	Steps	Full (ms)	Warm (ms)	Speedup	Warm MAD (%)	Memory reduction
32	4	2,036.73	145.38	14.01×	7.07	31.31%
32	8	4,045.01	289.13	13.99×	3.47	31.30%
128	4	2,043.72	166.66	12.26×	4.26	30.57%
128	8	4,103.12	330.67	12.41×	2.95	30.57%

Table 7: Negative and incomplete results retained as scope constraints.

Test	Outcome	Consequence
Natural generation check	exact handoff 11.99% slower	Benefits are edit/reuse conditioned; do not claim universal request speedup.
Safe KV-only diffusion	failed page-table validation	No accepted KV-only latency baseline; ordinary KV reuse is not shown sufficient or insufficient.
Four fragmented regions	CUDA illegal memory access	Region-DAG semantics pass synthetic tests, but fragmented GPU execution is not accepted.
Public Qwen/SDAR	contextual runs only	No cross-model transfer claim.
Router	small offline grid; always-warm tied oracle	Router is engineering evidence, not a learned-routing contribution.

to be The 100-case graph test supports contract coverage, but is not a proof over all interleavings of allocators, eviction events or concurrent serving schedules.

7 Future Work and Conclusion

The first step is to fix and independently validate the fragmented custom-paged execution and then build a proper KV-only recurrent-replay baseline. Broader evidence should include multiple checkpoint scales and GPU families, concurrent requests, batching, eviction pressure, and a public hybrid diffusion model. Region level GDN state forking could remove conservative replay for independent siblings. Multi-parent regions should start with explicit replay rather than learned state merging. Only once the measurement grid contains cases in which different routes actually win, a cost model should be deployed.

In short, repeated diffusion inference is an incremental program whose reusable unit is not just a token prefix. Exact reuse [18] requires a joint identity over dependencies, positions, masks, physical attention pages and recurrent frontiers. EXACTREGION converts high stable-token ratio to substantial measured latency and memory savings without retraining or changing outputs within that supported envelope. The negative results are equally architectural: they identify page-table validity, recurrent provenance, and fragmented GPU execution as the boundaries that future hybrid diffusion systems must make explicit.

Reproducibility Statement

The submission includes a frozen artifact manifest, case definitions, route-level raw records, exact-hash gates, hardware/software provenance, and scripts that separate correctness, cold construction, and warm timing. Appendix D maps claims to artifacts. The manuscript does not report results from failed or unsupported routes as measurements.

AI Assistance Disclosure

Generative AI tools were used for language editing, code assistance, experiment orchestration, and manuscript formatting. The authors executed the experiments, checked the source artifacts, verified the numerical claims and citations, and take full responsibility for the submission.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/958c530554f78bcd8e97125b70e6973d-Abstract.html.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Matthew A. Hammer, Yit Phang Khoo, Michael Hicks, and Jeffrey S. Foster. Adapton: Composable, demand-driven incremental computation. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 156–166, 2014. doi: 10.1145/2594291.2594324.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 11575–11596, 2023. doi: 10.18653/v1/2023.acl-long.647.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-0313.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7353–7363, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Liu_Timestep_Embedding_Tells_Its_Time_to_Cache_for_Video_Diffusion_Model_CVPR_2025_paper.html.

Yifei Liu, Juntong Wu, Yang Liu, Junhao Hu, Minghao Li, Xiaoxu Chen, and Weihang Chen. HYPIC: Accelerating hybrid-attention llm serving with position-independent caching. *arXiv preprint arXiv:2607.01299*, 2026a. URL https://arxiv.org/abs/2607.01299.

Yirui Liu, Ruoling Qi, Xuaner Wu, Penghang Liu, and Jian Chen. Tail-replay: Escaping the curse of linear attention in prefix caching for hybrid llms. *arXiv preprint arXiv:2608.30310*, 2026b. URL https://arxiv.org/abs/2608.30310.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. doi: 10.52202/079017-4236.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15762–15772, 2024b. URL https://openaccess.thecvf.com/content/CVPR2024/html/Ma_DeepCache_Accelerating_Diffusion_Models_for_Free_CVPR_2024_paper.html.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dKV-Cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*, 2025. URL https://arxiv.org/abs/2505.15781.

Quan Nguyen-Tri, Mukul Ranjan, and Zhiqiang Shen. Attention is all you need for kv cache in diffusion llms. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/3afaa2102fb8ea44cbadc13e45bba718-Abstract-Conference.html.

Quan Nguyen-Tri, Mukul Ranjan, and Zhiqiang Shen. Flash-dLLM: Io-aware kv caching and parallel decoding for fast, memory-efficient diffusion llms. *arXiv preprint arXiv:2609.26796*, 2026b. URL https://arxiv.org/abs/2609.26796.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *DeLTa Workshop at the International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=wzl61tIUj6.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4135.

Mikhail Shirokikh and Sergey Nikolenko. Sparse prefix caching for hybrid and recurrent llm serving. *arXiv preprint arXiv:2605.05219*, 2026. URL https://arxiv.org/abs/2605.05219.

Vaibhav Singh, Pierre-André Noël, Torsten Scholak, Eugene Belilovsky, and Oleksiy Ostapenko. Fixed state, long reach: What a constant-size cache buys block diffusion at scale. *arXiv preprint arXiv:2609.11998*, 2026. URL https://arxiv.org/abs/2609.11998.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: *Enhanced transformer with rotary position embedding. Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. Flashinfer: Efficient and customizable attention engine for llm inference serving. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/dbf02b21d77409a2db30e56866a8ab3a-Abstract-Conference.html.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2000.

Yuchen Zhu, Jing Shi, Chongjian Ge, Hao Tan, Yiran Xu, Wanrong Zhu, Jason Kuen, Koustava Goswami, Rajiv Jain, Yongxin Chen, Molei Tao, and Jiuxiang Gu. Flare: Diffusion for hybrid language model. *arXiv preprint arXiv:2606.01774*, 2026. URL https://arxiv.org/abs/2606.01774.