When a Warm Cache Changes the Answer in Reduced-Depth Inference

A correctness note on prefix caching with document-only depth plans on a hybrid model

Benny Goldstein and Eytan Yair Goldstein

bennylabs.org

November 21, 2026

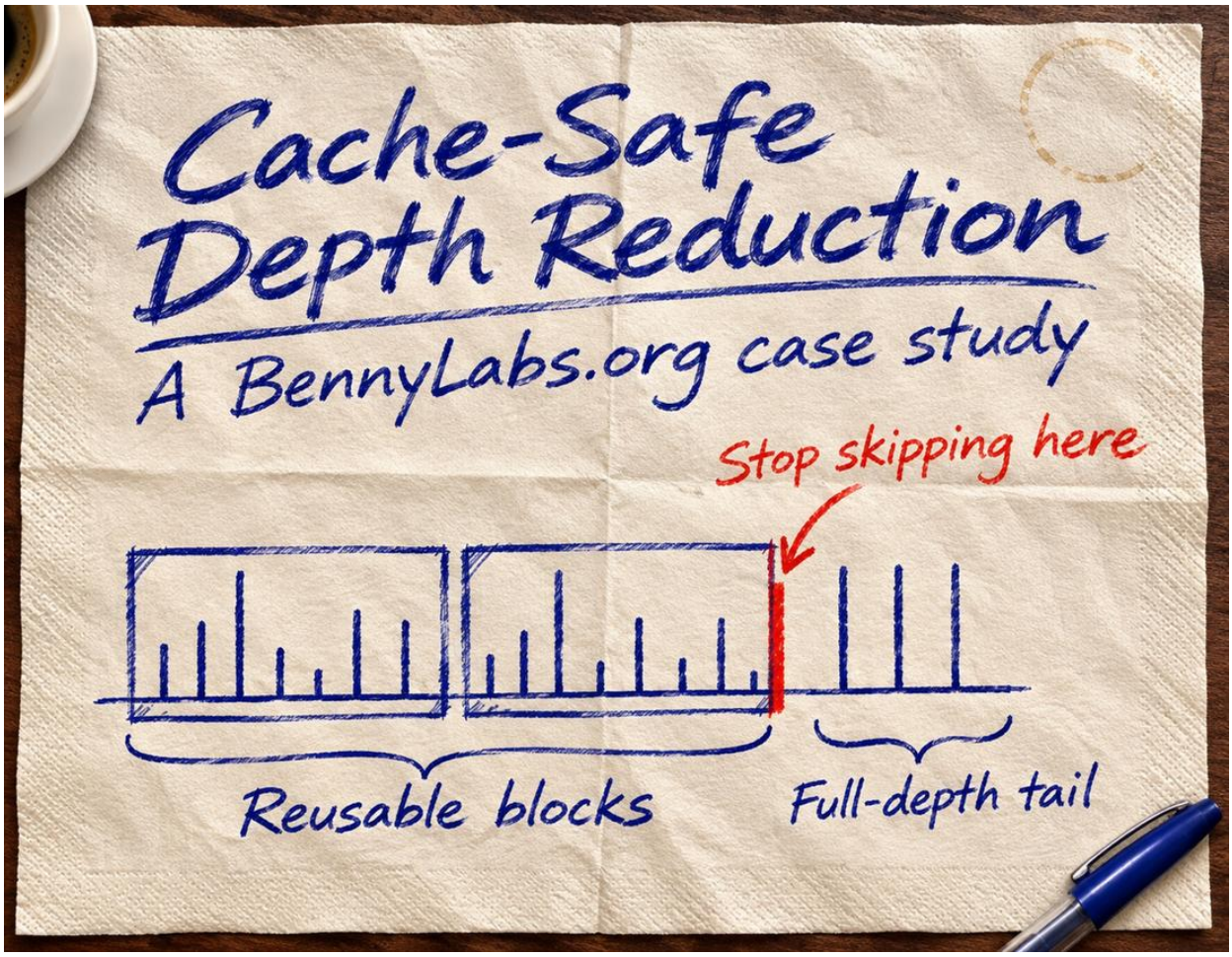


Figure 1. The idea on a napkin: reduced depth stops at the last reusable block, and the tail runs at full depth.

Abstract

Two ways of making long-document question answering cheaper are often wanted together. Depth reduction lets most document tokens run through fewer layers. Prefix caching stores the state of a document once and reuses it for every later question. We show that one way of

combining them gives different outputs depending on whether the cache is warm, and we show two ways to remove the difference.

The setting is a depth plan that applies to the document only, with the question at full depth, served by vLLM with its prefix cache on, on the hybrid model Qwen3.8-27B. The engine can resume this model only at multiples of 784 tokens, because its linear-attention layers keep a recurrent state that is checkpointed at that spacing. This value is documented and is printed by the engine at startup; it is not our finding. Its consequence for depth reduction is. On a follow-up question the engine recomputes up to 783 document tokens after the last checkpoint, and it recomputes them at full depth, while the first request ran them at reduced depth. Over 240 follow-up questions, none reproduced the output log-probabilities of a cold request exactly, and 21 produced different tokens. Ending the reduced-depth region on a multiple of 16 tokens, the engine's nominal block for standard models, did not help: 18 of 240 still produced different tokens, and only 16 matched a cold request exactly.

Two fixes each gave exact agreement on all 240 follow-ups. The first ends the reduced-depth region on the engine's real reuse boundary. The second re-applies the document's plan on every request. The first needs no plan at follow-up time but gives up reduced-depth work on short documents. The second gives up no reduced-depth work but needs the plan to be stored with the cache entry and applied again.

We did not measure a speed saving, and say so. This is a note about correctness. It gives a test that anyone combining the two methods can run on their own engine and model.

1. Introduction

A large language model spends the same number of layers on every token. For a long document followed by a short question, nearly all of that work goes into the document. Depth reduction attacks this by letting some tokens run fewer layers. In the variant studied here, a token that stops at layer k keeps its layer-k state, and later tokens and later layers read that state as if it were final. We call this the exit state.

Prefix caching attacks the same cost from another side. The serving engine stores the state of a prompt prefix, and a later request that starts with the same prefix reuses it. For document question answering, the document is read once and reused for every follow-up.

A deployment wants both, and a depth plan for cached documents should be made from the document alone, so that one stored state serves every question. The question then runs at full depth. This is the design we study. Its weak point is the stretch of document that the cache cannot serve. An engine resumes a prefix only at certain boundaries. Document tokens after the last boundary are recomputed with each new question. If the first request ran some of them at reduced depth and the follow-up recomputes them at full depth, the two requests are different computations, and the same question can get a different output depending on whether the cache was warm.

What is and is not new here. That hybrid models can be resumed only at checkpoint boundaries is known: it is described by Liu et al. (Tail-Replay), the 784-token value for this model appears in public cache documentation, and vLLM prints it at startup. That layer skipping and prefix reuse must be kept consistent is also known: vSkipper (Da et al.) serves

per-token layer skipping in SGLang with the prefix cache on, and keeps separate cache namespaces for dense and skipped execution. Our contribution is narrower. We measure the mismatch for a document-only depth plan with exit states on a hybrid model in a real engine, we show that aligning to the nominal block does not remove it, and we compare two fixes on the same requests.

2. Setting

Model. Qwen3.8-27B (revision 1d4bf0f2) has 64 layers. Three of every four (48) are linear-attention layers that carry a recurrent state; every fourth (16) is a full-attention layer with a key and value per token. Weights are 16-bit, about 54 GB.

Engine. vLLM 0.32.0, PyTorch 2.13, one NVIDIA H100 80 GB, prefix caching enabled, seed 0, at most 4 sequences, 8,192 batched tokens, context 33,920. The engine reported a unified block size of 784 tokens for this model, used both for hashing prefixes and for checkpointing recurrent state, with recurrent state saved only at multiples of that size. We call this spacing S. Cache hits in our runs came only in multiples of 784 tokens.

Depth plan. A planner assigns each document token a depth of 26, 38, 51 or 64 layers from a fixed table, and keeps numbers and negations at full depth. The planner is taken as given in this note. The planned documents ran at about 64% of the layer-steps of full depth. Reduced depth was applied inside the engine by a plug-in that runs the affected layers one at a time.

Controls. Each test carries three controls and is discarded if one fails. The plain model run twice must agree with itself. Our plug-in with an all-full-depth plan registered, which should leave the engine's own path untouched, must equal the plain model on every request. A planted fault must be caught by the same pass line. Pass lines were written before the runs and not changed. Where a result missed its line we say so.

3. The failure

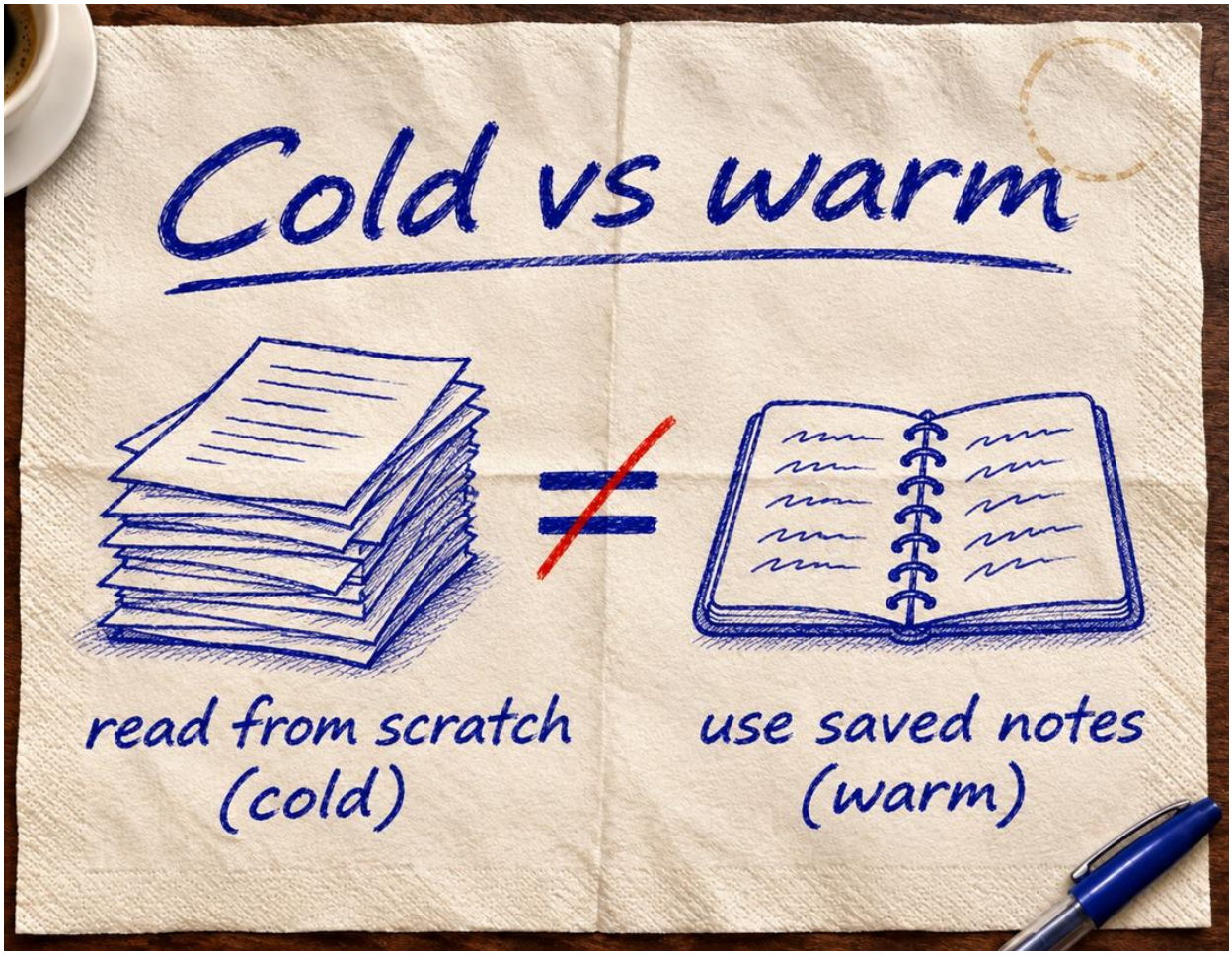


Figure 2. Cold versus warm: reading from scratch and using saved notes should give the same result.

Let a document have n tokens. The engine can resume at S × floor(n / S). A cold request runs all n document tokens at their planned depths and stores the state. A follow-up reuses the stored state up to the last boundary and recomputes the remaining n mod S document tokens together with the new question. With a document-only plan, the follow-up has no plan for those tokens and runs them at full depth. Every later token then reads a slightly different state than it did on the cold request.

On a standard transformer served with a 16-token block, the exposed stretch would be at most 15 tokens. On this model S is 784, the stretch is at most 783 tokens, and in our documents every follow-up was exposed.

4. Two fixes

A. End the reduced-depth region on a reuse boundary. Run tokens up to S × floor(n / S) at planned depth and every later token at full depth. Cold and warm requests then perform the same computation. On a cache hit, nothing has to be known at follow-up time; after a cache miss or an eviction the plan is needed again (Section 8). S must be the engine's real reuse

spacing for the model at hand; Section 6 says how to find it. The cost is the reduced-depth work given up on at most S − 1 tokens per document.

B. Re-apply the plan on every request. Keep the document's plan with the cache entry and apply it again to whatever document tokens a follow-up recomputes. No reduced-depth work is given up. The engine must be able to find the plan from the cached prefix. This is the same principle as keying the cache by execution mode, as vSkipper does with a request's prefill mode.

5. Experiment

Thirty documents, ten each at about 2,048, 8,192 and 32,768 tokens, carry planted facts in real web text. Each document is read cold with one question, and then asked eight follow-up questions, giving 240 follow-ups per arm and 4,950 requests in all. For each follow-up we compare the eight generated token ids and their log-probabilities with those of a cold request for the same full prompt. "Exact" below means all sixteen values are identical. We did not dump internal state tensors.

Arm	Same tokens as cold	Exact (ids and log-probs)
Plain, full depth	240 of 240	240 of 240
Reduced depth, no fix	219 of 240	0 of 240
Reduced depth, ends on a multiple of 16	222 of 240	16 of 240
Fix A: ends on a multiple of 784	240 of 240	240 of 240
Fix B: plan re-applied, no rounding	240 of 240	240 of 240
Fix B on top of rounding to 16	240 of 240	240 of 240
Planted fault	98 of 240	0 of 240

The number of cached tokens on each follow-up was the same as for the plain model in every arm, so neither fix costs cache hits.

All three controls passed. The plain model agreed with itself exactly on 240 of 240 requests, cold and warm. The plug-in with an all-full-depth plan registered took no plug-in steps and equalled the plain model on 240 of 240. The planted fault was caught. Every follow-up is compared with a cold request from the same arm, so both sides of each comparison run through the same code path. Exact agreement on all 240 pairs under each fix also shows that the reduced-depth path itself was repeatable in these runs.

Rounding to 16 was the rule we first proposed, from a model of a 16-token cache, and it was the pass line set before this run. It failed. Fix A at the engine's own spacing was added during the build, after the 784-token spacing was read from the engine and before any scored request was run. The rows with the plan re-applied were collected in the same run as a secondary measurement. We recognised them as a fix only when analysing the results, so fix B is a finding of that analysis and not a hypothesis set before the run. The last fix row shows that re-applying the plan also repairs a plan rounded to the wrong boundary.

How much harm does the mismatch do? We scored the changed answers against the known facts. With no fix, 21 follow-ups changed, on 11 of the 30 documents: 20 kept the same score with different continuation tokens, one turned a wrong answer right, and none turned a right answer wrong. With rounding to 16, 18 produced different tokens, on 11

documents: 17 at the same score, one wrong to right, none right to wrong. Accuracy was 234 or 235 of 240 in every reduced-depth arm, cold or warm, against 237 for the plain model. So in this sample the mismatch did not cost accuracy. What it costs is repeatability: the same question on the same document returns different text depending on cache state, which breaks regression tests, answer caching and audit.

Cost of fix A. At 2,048 tokens, fix A left 1,517 tokens at reduced depth against 1,852 with no rounding, and planned work rose from 64% to 71% of full depth. The loss is bounded by 783 tokens per document, under 2.4% of a 32,768-token document. Fix B has no such cost.

Time and energy. We did not measure a saving.

Cold read, time relative to plain	2,048	8,192	32,768 tokens
Plain repeated	0.96	0.94	1.06
Reduced depth, ends on 16	0.94	0.94	1.00
Reduced depth, fix A	0.94	0.97	1.02

The reduced-depth arms are inside the variation of the plain model against itself. The card ran at 86 to 88 °C and was throttled for nearly the whole run, so these timings are weak evidence in either direction. Follow-ups took 0.97 to 1.02 of the plain warm time under either fix. Separately from the controls, we forced full depth through the plug-in's layer-by-layer path. Against the engine's compiled path this changed the cold output on 8 of 240 requests, and it cost time. This is the numerical floor of the plug-in path, reported for completeness. It is not one of the controls and does not enter the cold and warm comparisons, which are made within one arm. A uniform depth for all document tokens at the same planned work as fix A scored 197 of 240, against 235 for the plan.

6. A test anyone can run

1. **Find S.** Read the engine's startup log for the block size it sets for the model. Confirm it: send a long prompt, then a second prompt sharing its first m tokens, and read the number of cached tokens the engine reports. Vary m one token at a time around a suspected boundary; the cached count steps up only at multiples of S. Do not assume the configured block size. For the model here the configured default for standard models is 16 and the real spacing is 784.
2. **Run the pair.** For a set of documents, issue each full prompt once on an empty cache. Then empty the cache, ask a different question on the same document, and issue the full prompt again as a follow-up. Compare the generated token ids and log-probabilities exactly.
3. **Check the harness.** The plain model must give exact agreement on every pair, and a reduced-depth cold request issued twice on an empty cache must agree with itself. If either fails, something is not deterministic in your configuration and the test cannot be read. We did not run the second check as a separate arm; in our runs it is implied by the exact agreement of all 240 pairs under each fix.
4. **Read the result.** Any pair that differs under depth reduction is a cache-consistency fault. Either fix of Section 4 should bring the count to zero.

7. Supporting results from a reference runner

Before the engine test we studied the same design in a reference runner that compares hidden states byte for byte and models a prefix cache of a chosen block size. Scoring is by teacher forcing on planted-fact tasks. Tables are in Appendix A.

- **Rounding at 16, 32 and 64 tokens.** With a modelled cache of each size, rounding to that size gave identical cold and warm state on 270 of 270 prompts, where about half of exposed prompts differed without it. Rounding gave up 0.67%, 1.36% and 2.86% of the skipped layer-steps.
- **A plan made without the question.** Over 480 follow-up questions on 60 documents, a plan made from the document alone scored 99.4%, a plan rebuilt for each question 100.0%, and a uniform depth at equal work 84.8%.
- **A plan that arrives late.** Starting the first layers before the plan is ready, with delays up to 400 ms, left the stored state byte-identical on 96 of 96 prompts.
- **How much can be skipped.** With half of all tokens stopped early, a two-hop lookup at 32,768 tokens scored 98.0% against 100.0% for full depth on 200 prompts (4 lost, none gained), and a uniform depth at equal work scored 80.5%.

Three results were weaker, and we report them.

- **A second model family.** On Mistral-Nemo-Base-2407, a standard transformer, rounding gave identical state on 269 of 270 prompts. The pass line was all of them. We have not found the cause of the one row. The document-only plan was no worse there than a per-question plan, but the plan itself cost about 11 points against the plain model, so our planner did not transfer well.
- **A public question set.** On 300 QuALITY questions with generated answers (documents of 4,100 to 8,700 tokens) the plan scored 86.7%, the plain model 87.7%, and a uniform depth at equal work 87.7%. The plan was safe but no better than a uniform cut.
- **The longest prompts.** At 131,072 tokens the half-stopped arm ran out of memory on 9 of 60 prompts, so that length has no verdict. On the 51 that ran it lost no answer.

8. Limitations

No speed or cost saving is shown. Reduced depth was not faster than full depth in the engine, on a throttled card and through a plug-in outside the compiled path. vSkipper reports latency gains for per-token skipping inside SGLang; we have not reproduced that for exit states.

One engine, one model, one spacing. The engine result is for vLLM 0.32.0 and Qwen3.8-27B at S = 784. We did not run a standard transformer in the engine, where S should be the configured block, nor a second engine.

Outputs were compared, not state tensors. Exact agreement of eight tokens and their log-probabilities is strong evidence that the states agree, but it is not a comparison of the cached state itself.

Cache misses and eviction are not covered. If a follow-up misses the cache entirely, the engine recomputes the whole document at full depth under fix A, and requests after it may

read a state that differs from the first. Fix B covers this case in principle. Neither was measured under eviction or under load.

The accuracy advantage over a uniform cut is shown on planted-fact tasks only. On the public set there was none.

The planner is taken as given. A different planner would change the accuracy numbers, though not the consistency argument.

Checkpoint-free caches are untested. Tail-Replay removes the fixed boundaries by rebuilding recurrent state from a short suffix. Under such a cache fix A has no boundary to align to, and fix B would be the one to try.

9. Related work

Early exit and layer skipping for language models include EE-LLM, LayerSkip, Mixture-of-Depths and FlexiDepth. SwiftKV lets later layers reuse an earlier layer's state for prompt tokens, uniformly for all tokens; our uniform-depth control is closest to it.

vSkipper (Da, Ferhatosmanoglu and Kalyvianaki, arXiv 2609.37062) makes per-token layer skippers run inside SGLang with continuous batching, paged caches and the radix prefix cache, and reports latency and throughput gains. It keeps prefix reuse consistent by having every layer write its own keys and values for skipped tokens and by keeping separate cache namespaces for dense and skipped prefill. Our fix B applies the same principle to a per-document plan, and our setting differs in using exit states, a plan that covers only part of a request, and a hybrid model with coarse reuse boundaries.

Tail-Replay (Liu, Qi, Wu, Liu and Chen, arXiv 2608.30310) describes why hybrid models can be resumed only at recurrent-state checkpoints and proposes rebuilding an approximation of that state by replaying a short suffix instead. The block size of 784 tokens for Qwen3.8-27B is listed in the LMCache documentation (the recipe for the Qwen3.5, Qwen3.6 and Qwen3.8 series, github.com/LMCache/LMCache, docs/source/recipes/qwen3_5.rst) and is logged by vLLM at startup. Prefix caching in vLLM builds on PagedAttention (Kwon et al.).

Run-to-run differences in model output under 16-bit arithmetic were measured by Yuan et al. (NeurIPS 2025); we use deterministic settings and a plain-against-plain control for that reason.

We are not aware of a published measurement of the cold and warm mismatch for document-only depth plans on a hybrid model, or of the observation that aligning to the nominal block does not remove it. We may have missed one.

10. Conclusion

A depth plan that covers only the document, served through a prefix cache on a hybrid model, gave outputs that depended on whether the cache was warm: none of 240 follow-ups matched a cold request exactly. Aligning the reduced-depth region to 16 tokens did not fix it, because the engine resumes this model only every 784 tokens. Aligning to 784, or re-applying the

plan on every request, fixed it on all 240. In our sample no correct answer became wrong, but outputs depended on cache state, which callers generally do not expect.

The practical advice is short. Read the engine's real reuse spacing for your model, run the cold and warm pair test of Section 6, and choose one of the two fixes. Showing that reduced depth also saves time in this setting remains open.

Appendix A. Reference-runner tables

All on Qwen3.8-27B, one H100, teacher forcing, modelled prefix cache.

```
A.1 Rounding, modelled cache of block size b
b      Identical after rounding    Unrounded, exposed rows identical
Skipped layer-steps given up
16     270 of 270                  125 of 249
0.67%
32     270 of 270                  113 of 261
1.36%
64     270 of 270                  112 of 267
2.86%
Accuracy with rounding against without: -0.4, -0.4, -0.7 points, each
inside its 95% interval.
Rounding up instead of down (planted fault) broke 54% to 58% of exposed
rows.

A.2 Plan made without the question, 480 follow-ups on 60 documents of 2,048
tokens
Plain, full depth                                  480     100.0%
Plan rebuilt for each question                     480     100.0%
Plan made once, without the question               477      99.4%
One uniform depth at the same work                 407      84.8%
Planted fault (every document token at 2 layers)            0.4%

A.3 Plan arrives after the first layers have started, 96 prompts of 2,048
tokens
Delay 0, 1, 10, 100, 400 ms: 0 of 96 prompts with any byte of difference at
each delay.
Planted fault (give up after 50 ms): 96 of 96 changed.

A.4 Half of all tokens stopped early, two-hop lookup
                                  32,768 tokens      131,072 tokens
                                  (200 prompts)      (51 of 60 prompts
ran)
Plain, full depth                     100.0%             98.0%
50% of tokens stopped                  98.0%            100.0%
Uniform depth, same work               80.5%             80.4%
Planted fault                           0.0%              0.0%
At 32,768: paired difference -2.0 points, 95% interval -3.9 to -0.1, one-
sided sign test p = 0.0625.
The 131,072 column is incomplete (9 prompts ran out of memory) and carries
no verdict.

A.5 Retrieval by position, one needle at ten positions, 200 prompts
Lengths 16,384 / 32,768 / 65,536. Plain 100.0%, 98.8%, 100.0%.
The planned run lost no length-by-position cell against plain or against a
uniform cut at equal work.
```

Appendix B. Reproducibility

- Engine: vLLM 0.32.0, PyTorch 2.13.0 (CUDA 13.0), 16-bit weights, prefix caching on, seed 0, max sequences 4, max batched tokens 8,192, max model length 33,920, GPU memory fraction 0.9.
- Model: Qwen/Qwen3.8-27B at revision 1d4bf0f2ff6012fd82039f2fa52739d0dd7c60c0.
- Reported cache configuration: block size 784, hash block size 784, recurrent-state block size 784, recurrent-state cache mode "align".
- Card: NVIDIA H100 PCIe, 80 GB. Temperature 86 to 88 °C during the engine run.
- We will release the frozen prompts, the scoring scripts with their pass lines as written before the runs, and the per-request rows, including the runs that missed their pass line.

Methods described in this note are the subject of pending patent applications.