Inference · Technical report

A speedup that only appears in long conversations

Trevor Rosato

A one-line change to which kernel handles a single matrix shape made a ternary 27B model generate faster in every paired trial, by 9.5% on average, once a conversation held 46,000 tokens of history. On short prompts it barely registered. Where you measure decides what you find.

November 12, 2026 Work conducted September 2026 6 minute read

Ternary Bonsai 2 27B stores almost every weight of a 27-billion-parameter model as one of three values. Its publisher reports that it keeps about 98% of the full-precision model's average score across fourteen reasoning benchmarks, in about a ninth of the memory.[1] That makes it one of the most promising candidates in our consumer-GPU research. This report covers two changes we made to serving it on an RTX 4070, and one lesson about how to test changes like them.

Shrinking the draft model

We serve Bonsai with speculative decoding, pairing it with a separate multi-token-prediction draft model.[2] On a 12 GB card, the draft's memory competes directly with conversation context, so we recompressed the draft's largest matrices. Eight

matrices stored at eight bits per weight became six at four bits and two at six. Their combined size fell from 451 MB to 263 MB, freeing **179 MiB** without touching the target model at all. Peak memory for the whole server stayed at 11,106 MiB, just inside the card.

Choosing a better kernel for one shape

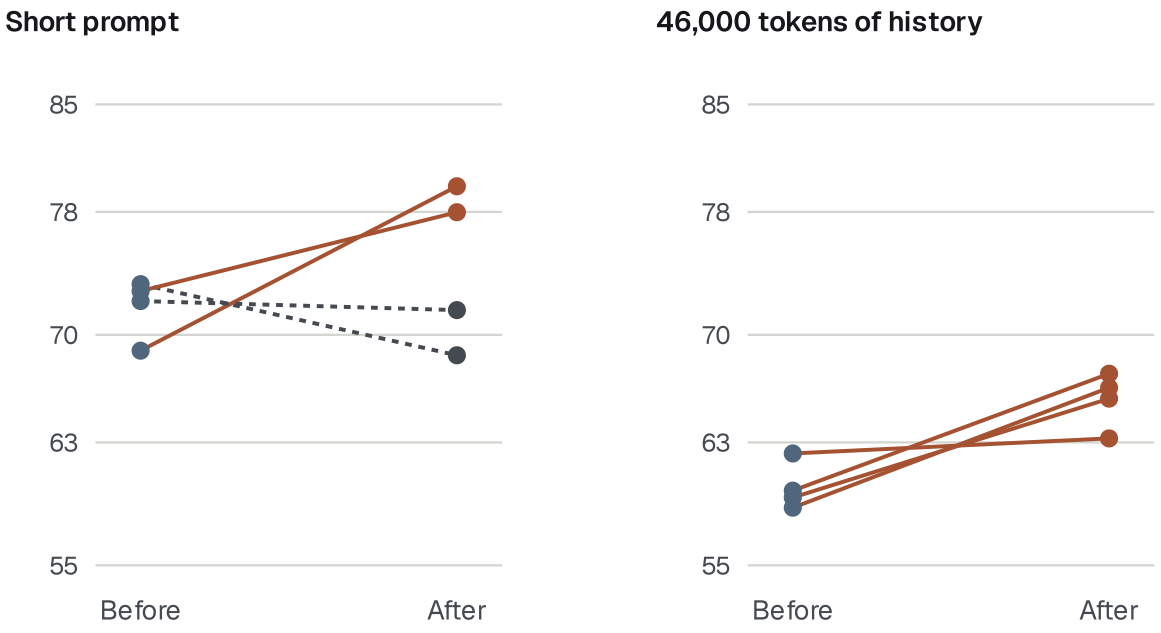
The second change was smaller still. During speculative verification, the model checks two or three proposed tokens at once. For one particular matrix shape on this GPU architecture, the runtime's general dispatch rule did not send those two- and three-row cases to the vector kernel it already had, which suits them better, as the shape of a matrix multiplication determines which implementation performs best.[3] We changed the dispatch rule for that shape and those widths, and nothing else; the patch is public and pinned to a fixed commit.[4] We were not alone in finding it. A contributor to Prism ML's llama.cpp fork made the same change independently in September, measured on an RTX 3060,[5] and a broader rule that sends this shape to the same kernel has since been merged upstream.[6] What this report adds is a paired measurement of what the change does to complete requests deep in a long conversation.

No new kernel, no new algorithm. Changes like this are easy to dismiss and easy to get wrong, because a microbenchmark of the kernel says very little about what happens to a real request. A different kernel can produce very slightly different numbers, because floating-point results depend on the order of operations,[7] which can change which tokens the draft proposes, which changes how many the target accepts, which changes how much work the whole request does.

Measure where the work is

So we measured complete requests, in four matched pairs with the order

counterbalanced, under two conditions: a short prompt, and a follow-up in a conversation that already held **46,015 tokens** of cached history. Both generated 2,048 tokens.



Decoding speed, tokens per second. Each line is one matched pair.

Figure 1
On short prompts the change was a coin flip: two pairs improved and two got slower. With a long conversation behind it, all four pairs improved, by a geometric mean of 9.5%.

With history, the result was unambiguous: every pair improved, and generation went from a median of **59.6 to 66.2 tokens per second**. On short prompts, the change won two pairs and lost two, a geometric mean gain of 3.5%.

Had we tested only short prompts, we would have shelved a change that matters most deep in a long conversation, which is where a tutoring session ends up. Configured context length is not the same thing as occupied context: decoding attention has to read the history a conversation actually holds,[8] and a server that is allowed 49,000 tokens behaves differently from one that is actually holding 46,000.

The gain belongs to the integrated system: the faster kernel and the tokens the drafter proposed and had accepted moved together, and for choosing what to run, the combined speed is what counts.

What this changed

Two habits. We now evaluate inference changes with long, filled conversation histories, not just with short prompts. And we treat narrow dispatch choices on our own hardware as a real source of speed. General-purpose rules have to serve every shape on every card; a deployment that spends most of its time on a handful of shapes can do better by measuring them.

Methods and scope

All measurements come from one RTX 4070 (12 GB) with the llama.cpp runtime as distributed for Bonsai. The comparison used four matched pairs per condition, in counterbalanced order (AB, AB, BA, BA), with the same model files in both arms. Four short-prompt pairs cannot distinguish a 3.5% gain from noise. The long-history gain cannot be split between the dispatch change and the change in drafted and accepted tokens, which moved together; separating them would need the drafting held fixed. Rates are decoding throughput; complete-request times, including prompt processing, moved in the same direction. Both arms scored 7 of 12 on a small quality check, with no observed score regression between them; it is not a broad quality evaluation. Benchmark figures for Bonsai 2 27B are the publisher's.

The target model comes from Prism ML and the draft model from ProCreations, both building on Qwen and llama.cpp. Our contribution was assembling the artifact, recompressing the draft, and the paired evaluation at long, filled histories. The dispatch change was reached independently by others, as noted above, and our numbers apply to the pinned build we measured.

References

1. Prism ML, Ternary Bonsai 2 27B model card https://huggingface.co/prism-ml/Ternary-Bonsai-2-27B-gguf.
2. ProCreations, Ternary-Bonsai-2-27B-MTP https://huggingface.co/ProCreations/Ternary-Bonsai-2-27B-MTP.
3. NVIDIA, "Matrix Multiplication Background User's Guide" https://docs.nvidia.com/deeplearning/performance/dl-performance-matrix-multiplication/index.html, Deep Learning Performance documentation, 2023.

4. bonsai_bf16_small_mmvf.patch https://github.com/trevor050/escha-rtx4070-lab/blob/ed8d6788915367961beebcd24b60e2c2bb2cadec/bonsai/work/bonsai-graft/bonsai_bf16_small_mmvf.patch, escha-rtx4070-lab, commit ed8d678.
5. sudoingX, “CUDA: fast, batch-invariant small-batch PTQ1_0 mat-vec (1.5x decode on Ampere)” https://github.com/PrismML-Eng/llama.cpp/pull/218, PrismML-Eng/llama.cpp pull request #218, opened September 19, 2026. One of its commits sends bf16 matrices under 64 rows at 2 to 8 columns to the mat-vec kernel, naming the 5120 × 48 gate projection. It reached Prism’s branch through pull request #221 https://github.com/PrismML-Eng/llama.cpp/pull/221, merged September 29, 2026.
6. ynankani, “CUDA: use MMVF for thin f16/bf16 mul_mat at small batch size” https://github.com/ggml-org/llama.cpp/pull/29633, ggml-org/llama.cpp pull request #29633, merged October 5, 2026.
7. PyTorch, “Numerical accuracy” https://docs.pytorch.org/docs/2.14/notes/numerical_accuracy.html, PyTorch 2.14 documentation.
8. Tri Dao, Daniel Haziza, Francisco Massa and Grigory Sizov, “Flash-Decoding for long-context inference” https://pytorch.org/blog/flash-decoding/, PyTorch Blog, 2023.

Cite this

Trevor Rosato. "A speedup that only appears in long conversations." research.honorlyai.com, November 12, 2026. https://research.honorlyai.com/a-speedup-that-only-appears-in-long-conversations/