Batched Speculative Decoding Verification Kernels: Milestone Report

Steven Kolawole

URL: https://stevenkolawole.github.io/15618/

Progress So Far

I have completed the CPU-side foundation for the project. The PyTorch reference implementation is in place for batched verification, including accepted-length computation, mismatch handling, and KV write-offset bookkeeping. I also built a synthetic workload generator so I can control batch size and acceptance rate while still knowing the correct answer for each sequence.

I then added the CUDA path scaffold: a lazy extension loader, a naive one-thread-per-sequence kernel source, a GPU smoke script, a benchmark script, and a gated GPU parity test. The current CPU test suite passes (6 passed, 1 skipped), and the benchmark harness runs in CPU mode. I have validated the compiled CUDA extension, and the next steps are to begin profiling and optimization.

Goals and Deliverables

Status

I still believe the core deliverables in the proposal are achievable: a naive CUDA verification kernel, Nsight profiling, Opt A ballot-based acceptance scanning, Opt B prefix-sum KV compaction, and ablation plots across batch sizes and acceptance rates. The stretch goals remain nice-to-haves rather than requirements.

The main constraint now is the remaining implementation complexity, since some of the later steps are more subtle than they first appear, especially the KV cache optimization. That part will require careful handling of ragged accepted lengths and memory layout if I want the optimization to be genuinely useful rather than just technically correct. In practice, I would assign write positions with an exclusive prefix sum over the accepted lengths so that each sequence gets a contiguous output segment. The basic idea is simple, but making the actual scatter coalesced and cheap is the hard part; otherwise, the extra complexity and bookkeeping may not be worthwhile.

Beyond that, the remaining work is mostly engineering and measurement, not uncertainty about the overall direction of the project.

Poster Session

For the poster session, I plan to show: (1) a comparison of naive versus optimized kernel behavior from Nsight Compute, (2) latency or speedup curves across batch sizes and acceptance rates, and

(3) a short explanation of why ragged verification causes divergence and why the optimizations help.

Revised Schedule

Dates	Person	Tasks
Apr 15–16	Steven Kolawole	Compile the CUDA extension, run the GPU smoke test, and confirm parity against the CPU baseline.
Apr 17–18	Steven Kolawole	Run timing sweeps across $b \in \{1, 4, 16, 32\}$ and $\alpha \in \{0.3, 0.6, 0.9\}$; record baseline and CUDA timings.
Apr 19–20	Steven Kolawole	Collect Nsight Compute metrics for the naive kernel and summarize warp efficiency, occupancy, and memory behavior.
Apr 21–22	Steven Kolawole	Implement Opt A ballot-based acceptance scanning and verify correctness against the baseline.
Apr 23	Steven Kolawole	Start Opt B prefix-sum KV write compaction and check whether the memory layout actually improves coalescing.

Next Deliverables

Since my last day to work on the project is Apr 23, I am prioritizing the CUDA baseline, Opt A, and the first pass at Opt B before that deadline.