Verify-PD - Disaggregated Serving for Speculative Decoding

2. Team Members

- Naveenraj Kamalakannan (nk3940)
- Megh Panandikar (mp6545)

3. Goal

The primary goal of this project is to improve the tail latency (p95/p99) of large language model (LLM) inference under real-world, mixed-traffic conditions. We will achieve this by isolating the "verification" step of speculative decoding into its own dedicated, medium-priority execution queue. This disaggregation, which we term "Verify-PD," aims to prevent the computationally intensive verification pass from interfering with latency-critical, single-token "decode" steps, thereby preserving speculative throughput gains without sacrificing interactive performance.

4. Challenges

- **Resource Contention:** Verification in speculative decoding, running on the same hardware as high-priority decode steps, can bottleneck operations and increase latency.
- **Dynamic Workloads:** LLM serving requires adaptable scheduling for varied requests (prefill, decode, output lengths) due to fluctuating acceptance rates and queue backlogs.
- **System Overhead:** New queues, streams, and a feedback controller add complexity. Performance gains from disaggregation must significantly outweigh new scheduling latencies.

5. Approach

Our approach is to re-architect the speculative decoding scheduler into a three-lane system with strict, preemptive priorities.

- **Three-Lane System Architecture:**
  1. **Prefill Lane:** Executes prefill step for the entire prefix for both models. This is expected to be compute bound but not memory bound as all KV values are calculated and stored in parallely.
  2. **Decode Lane:** Generates the next L tokens using the small model and a single token in decode of the large model. This is not very compute heavy as we are decoding tokens one by one however since we keep accessing memory it is memory bound. We will leverage CUDA Graphs to minimize kernel launch overhead..
  3. **Verify Lane:** In this step the initial prefix plus the L generated tokens by the draft model are input into the large verifier model and we do a forward pass where each of the L generated tokens is verified. This lane will use its own CUDA stream to run in parallel with other operations and is expected to be mostly compute bound but also requiring some memory speed to access prefill cache.

- **Acceptance-Aware Feedback Controller:** This is the core intelligence of our system. It will monitor key performance indicators in real-time:
  - Per-request token acceptance ratio.
  - The current depth of the Verify Lane queue.
  - The p95 latency of the Decode Lane.

  If the verify queue grows or decode latency rises, the controller will dynamically shrink the draft length L or increase the drafter's temperature to reduce the verification workload. Conversely, when there is headroom, it will grow L to maximize throughput.

6. Implementation Details

- **Hardware:** We'll use modal.com or vast.ai for NVIDIA A100 or H100 GPU, with ablation studies on a dual-GPU setup for isolation testing of the Verify Lane.
- **Software:** Our system will build on vLLM, using its PagedAttention and modifying its scheduler and CUDA stream management.
- **Dataset:** The ShareGPT dataset

7. Demo Planned

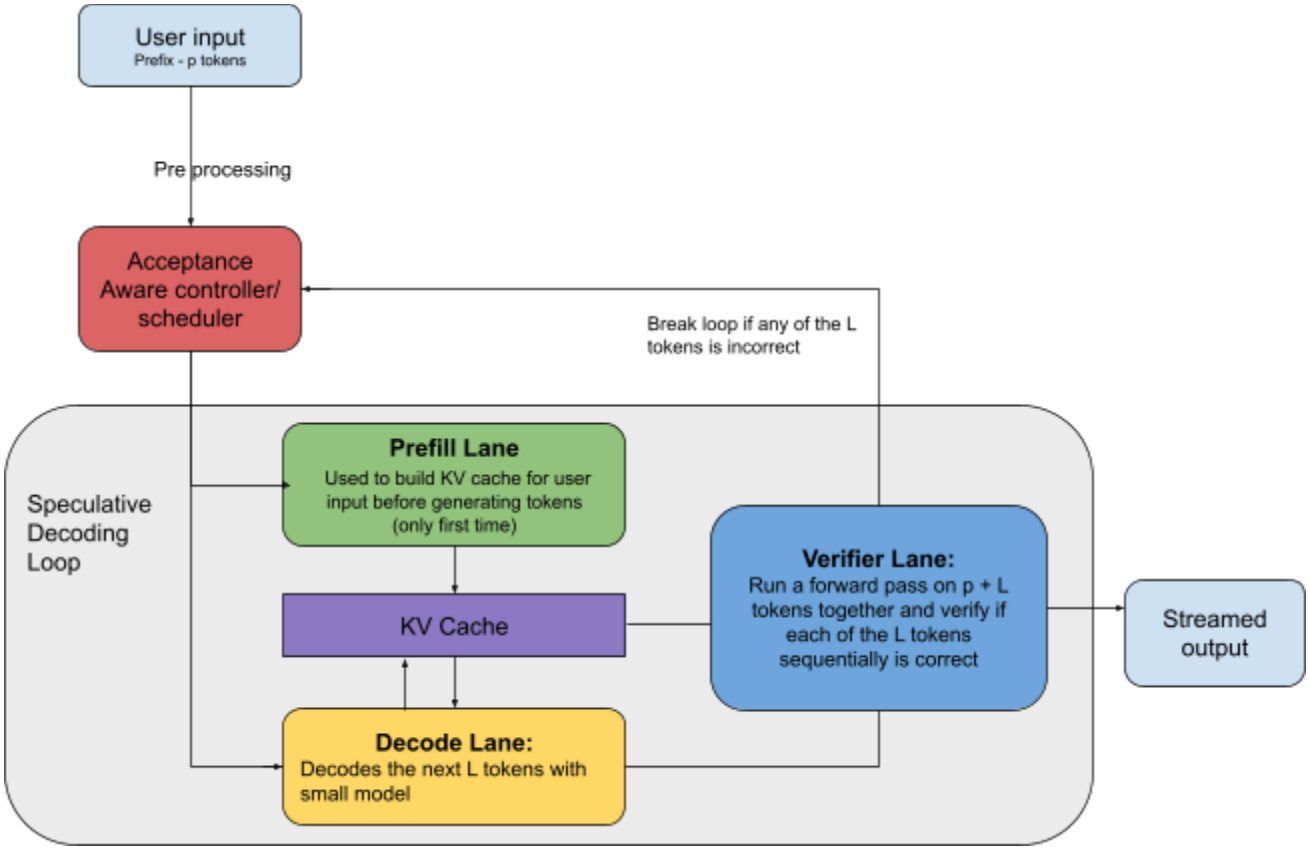
The final demo will feature a comparative dashboard of Verify-PD and vLLM's speculative decoding performance. The dashboard will display live throughput, latency (with p50/p95/p99 and histograms), scheduler health (Decode and Verify lane queue depth), and GPU utilization (SM occupancy and memory bandwidth). We expect Verify-PD to show stable, low p95 decode latency under mixed workloads and moderate acceptance rates, ensuring quality of service, unlike the baseline's p95 latency spikes.

8. References

1. **Leviathan, Y., Kalman, M., & Matias, Y. (2023).** ***Fast Inference from Transformers via Speculative Decoding***
   This paper introduces Speculative Decoding, which speeds up LLM inference using a smaller model to generate tokens and a larger model to verify them.
   Our project optimizes its deployment by addressing resource contention between verification and decoding, crucial for low tail latency in production, without altering the core algorithm.

2. **Zhong, Y., Liu, S., Chen, J., Hu, J., Zhu, Y., Liu, X., Jin, X., & Zhang, H. (2024).** ***DistServe: Disaggregating Prefill and Decoding for Goodput-Optimized Large Language Model Serving.***
   This paper improves LLM efficiency by separating the compute-bound prefill and memory-bound decode steps, allocating appropriate GPU resources.
   We will apply this to Speculative Decoding by creating three lanes for prefill, decode, and token verification, boosting speed through scheduling and hardware optimization.

Flow Chart


User input
Prefix - p tokens
Pre processing
Acceptance Aware controller/ scheduler
Break loop if any of the L tokens is incorrect
Speculative Decoding Loop
Prefill Lane
Used to build KV cache for user input before generating tokens (only first time)
KV Cache
Verifier Lane:
Run a forward pass on p + L tokens together and verify if each of the L tokens sequentially is correct
Streamed output
Decode Lane:
Decodes the next L tokens with small model