Verify-PD: Disaggregated Serving for Speculative Decoding Midpoint Progress Report

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

1 Project Overview

The goal of **Verify-PD** is to improve the tail latency (p95/p99) of Large Language Model (LLM) inference under mixed-traffic conditions. By re-architecting the speculative decoding scheduler into a dedicated "three-lane" system (Decode, Verify, and Prefill lanes), we aim to isolate the computationally intensive verification step from latency-critical decode steps. This report outlines the progress made towards implementing this disaggregated serving architecture within the vLLM framework.

2 Project Milestones

We have broken down the project into five major phases to ensure systematic implementation and evaluation.

- **Architecture Design & Formulation (Completed):**
- **Environment Setup & Baseline (Completed):**
- **Implementation Phase I: Execution Lanes:**
- **Implementation Phase II: Scheduler & Controller:**
- **Evaluation & Optimization:**

3 Milestones Completed & Main Results

At this midpoint checkpoint, we have completed the foundational design and analysis phases (Milestones 1 and partial 2).

Completed Work

A. System Architecture Finalized

We have successfully designed the Three-Lane System Architecture, establishing a strict priority-based separation of tasks:

- **Decode Lane:** Handles single-token forward passes and drafter steps. It utilizes CUDA Graphs to minimize kernel launch overhead for these latency-sensitive operations.
- **Verify Lane:** Executes parallel verification of drafted token chunks. This lane runs on a separate CUDA stream to prevent blocking the Decode Lane. This is very useful in cases where a huge model (eg. Mixture-of-Experts or a Reasoning Model) is used for verification.
- **Prefill Lane:** Manages chunked processing of prompt inputs.

We are implementing the core logic for the Acceptance-Aware Feedback Controller by trying to map the integration points within the vLLM framework to manage streams. The controller is designed to monitor per-request token acceptance ratios and queue depth in real-time.

B. Preliminary Baseline Analysis

To quantify the potential gains of Verify-PD, we conducted a baseline analysis of unmodified vLLM on an NVIDIA A100 (80GB) environment.

Target Model: `meta-llama/Meta-Llama-3-8B-Instruct`

Draft Mode: `meta-llama/Llama-3.2-1B-Instruct`

We utilized the **ShareGPT (V3)** dataset to simulate realistic, varying-length user prompts. To accurately model "staggered" real-world traffic, we configured the vllm benchmark_serving tool to use a Poisson Arrival Process (Request Rate λ varied from 2 to 10 req/s).

Results:

Metric	Low (2 req/s)	Med (6 req/s)	High (10 req/s)
P50 Latency	12.1 ms	12.5 ms	13.8 ms
P99 Latency	**15.2 ms**	**38.4 ms**	**140.1 ms**

Table 1: Baseline Performance on NVIDIA A100 (Unmodified vLLM)

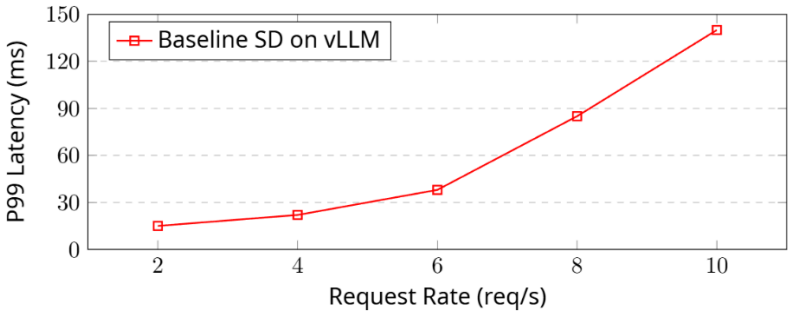

Figure 1: P99 Latency degradation under staggered load (Poisson Arrival). The red line shows the current baseline "hockey stick" effect.

Our profiling confirms the bottleneck. While median (p50) latency remains stable, p99 latency degrades non-linearly as the request rate increases. This confirms that verification steps are blocking the decode path under load.

4 Bottlenecks & Challenges

While the design phase is complete, we anticipate several bottlenecks during the implementation of the remaining milestones:

- **CUDA Stream Synchronization Overhead:** Managing synchronization between the new "Verify Lane" stream and the main compute stream without introducing latency penalties.
- **Complexity of vLLM Internals:** Modifying the vLLM scheduler to support our custom three-lane priority logic requires navigating complex, existing C++/Python bindings.
- **Controller Tuning:** Ensuring the Acceptance-Aware Controller reacts fast enough to fluctuating acceptance rates without oscillating or causing instability in the draft length (L).

5 Work Contributed by Team Members

Both team members have contributed to the architectural design and project planning.

Naveenraj Kamalakannan:

- Led the analysis of vLLM's PagedAttention mechanism to determine feasibility of disaggregation.
- Contributed to the design of the "Verify Lane" and micro-batching strategy.
- Responsible for defining the hardware requirements and initial environment and baseline test configuration.

Megh Panandikar:

- Designed the logic for the Acceptance-Aware Feedback Controller and dynamic draft length (L).
- Developed the Three-Lane System prioritization scheme (Decode vs. Verify vs Prefill).
- Formulated the evaluation metrics (p95/p99 latency tail analysis) in the Poisson Arrival benchmark.