Master thesis project proposal

Speeding Up LRM Token Generation Using FlashHead

Student 1 Fengming Tong (tongf@chalmers.se)
Student 2 Yanyin Tang (yanping@chalmers.se)

Suggested Supervisor at CSE: Matti Karppa

Relevant completed courses student 1:

DAT470, Computational techniques for large-scale
DAT400, High-performance parallel programming
DAT441, Applied machine learning

Relevant completed courses student 2:

DAT470, Computational techniques for large-scale
DAT400, High-performance parallel programming
DAT441, Applied machine learning

February 5, 2026

1 Introduction

Large language models (LLMs) are increasingly used in real-time applications, but their inference latency remains a major bottleneck. In particular, the dense classification head of an LLM—mapping final hidden states to next-token probabilities—can dominate compute time due to huge vocabularies. For example, in models like LLaMA, the output head may comprise ≈50-60% of parameters and FLOPs [1]. From an HPC perspective, this is a massive memory-bound and compute-intensive operation that limits system throughput.

FlashHead is a recently proposed drop-in replacement for this head that treats decoding as a retrieval task. It clusters token embeddings and uses fast approximate search, achieving up to 1.75× speedup in model-level inference while preserving accuracy [1]. A 2025 FlashHead announcement even reports ≈43% latency reduction on LLaMA-3.2 with full accuracy retention [3]. These results suggest that integrating FlashHead into LLaMA-family models could greatly cut inference time. This proposal will explore that integration and evaluate the trade-off between latency and accuracy on standard benchmarks while also considering its performance in parallel environments and multi-GPU scalability.

2 Problem

Inference bottlenecks arise when the output head must score a large vocabulary at each token step. Key aspects of this problem include:

- **Classification-Head Overhead:** Modern LLMs often use vocabularies of 100K+ tokens. A dense ($hidden\_dim \times vocab\_size$) multiplication at each step is costly [1]. As vocabularies grow, the head can consume the majority of inference compute.

- **Real-Time Scenarios:** Applications like interactive chat, multi-turn dialogue, and complex reasoning (chain-of-thought) require low latency generation. Reasoning models that perform deliberate multi-step inference (as in ChatGPT or DeepSeek) incur heavy delays [7]. Any head-speedup directly reduces such end-to-end latency.

- **Accuracy Preservation:** While accelerating inference is crucial, maintaining output quality is mandatory. Misranking tokens can hurt downstream task success. Therefore, any head replacement should aim to match the original softmax output distribution as closely as possible to cut latency without appreciable loss in task accuracy.

- **System Scalability and Parallelism:** In distributed environments, the classification head requires significant synchronization and data movement. Traditional softmax heads often struggle to scale linearly across multiple GPUs due to the massive communication overhead required for

partial result aggregation

3 Context

The vocabulary bottleneck has long motivated specialized output layers. Early approaches like adaptive or hierarchical softmax cluster tokens to reduce compute. For instance, switching to an adaptive softmax can yield 2X–10X speedups in inference with little performance loss [5]. However, such methods typically require retraining or fine-tuning the model to learn the new output structure.

A recent solution is FlashHead, which requires no retraining. It reformulates the classification head as a small-scale retrieval problem. FlashHead clusters token embeddings into equal-sized groups and performs a multi-probe nearest-neighbor search, with probabilistic sampling and low-bit quantization to approximate the dense softmax [1].

Another line of work accelerates inference by parallelizing generation rather than altering the head, such as Speculative Sampling (e.g., EAGLE). The original EAGLE [6] achieved $\approx$2.7–3.5$\times$ speedup on LLaMA2-Chat 70B. EAGLE-3 [7] uses an improved training-time test to fuse multi-layer features, achieving up to 6.5$\times$ speedup. These methods target the autoregressive process; our work is complementary as it focuses on the output head. Other frameworks like vLLM still leave the softmax head in high precision to avoid accuracy drops. FlashHead represents a promising, hardware-friendly way to cut head cost without retraining.

4 Goals and Challenges

The primary objective is to implement FlashHead as a substitute for the dense softmax head in open-source LLaMA-style models and quantify the latency-accuracy-scalability trade-offs. specifically, we aim to:

1. Integrate FlashHead into a LLaMA pre-trained model at inference time (no fine-tuning of model parameters) [8].

2. Evaluate the modified model on reasoning benchmarks, comparing solution accuracy vs. original.

3. Measure inference speed-up relative to the baseline and investigate how parameters (cluster size, probe count, quantization level) affect the trade-off.

4. Evaluate multi-GPU performance: Analyze how FlashHead scales across multiple GPUs using tensor and pipeline parallelism.

Challenges include ensuring the FlashHead output distribution remains close to

the original. FlashHead's multi-probe sampling must capture the most likely tokens; setting the number of probes and sample sizes is critical. Also, clustering must be efficient: we will use equal-sized K-means clusters to enable low-overhead memory access. Finally, reasoning benchmarks often require exact correctness (e.g., solving a math problem), so even small errors in token ranking can translate to failed solutions. We must tune parameters to minimize such errors while maximizing speed gains.

5 Approach

5.1 Methodology

We will base our experiments on open LLaMA-family instruction-tuned models (e.g., LLaMA-3.1). The dense output head will be replaced by the FlashHead mechanism as follows:

- **Parallel Clustering:** Pre-cluster the model's vocabulary embeddings into equal-size groups using K-means(utilizing multi-core CPU/GPU resources). Each cluster has a centroid vector for retrieval.
- **Indexing & Multi-Probe:** Build an ANN index (e.g., using Faiss) over the cluster centroids. At inference time, for each hidden-state vector h, we retrieve the M nearest clusters to focus the search.
- **Token Retrieval:** Within each selected cluster, score tokens by distance/similarity to h (using quantized embeddings) and sample a subset of top candidates.
- **Probability Estimation:** Convert the retrieved candidates' similarity scores into a probabilistic output (e.g., via softmax over sampled tokens).
- **Quantization:** We will quantize cluster embeddings (e.g., 8-bit) to reduce compute and maximize memory bandwidth efficiency and then evaluate the impact on accuracy
- **Parameter Tuning:** We will experiment with the number of clusters (K), number of probes (M), and quantization bitwidth.

This approach is inserted at the final stage of inference. Importantly, no additional model training is needed; the transformer body remains unchanged.

5.2 Evaluation

We will evaluate on a suite of reasoning and dialogue benchmarks:

- **Math Reasoning:** GSM8K [2] and MATH [4]. These require multi-step reasoning, measured by exact-answer accuracy.

- **Dialogue/Chain-of-Thought:** MT-Bench or similar multi-turn evaluation to assess general generation quality.
- **Throughput:**Measure tokens per second (TPS) across different GPU counts.
- **Memory Profiling:**Memory Profiling: Monitor GPU memory bandwidth and cache uti lization to identify system bottlenecks.

Metrics: For each benchmark, we record: (a) Latency (average wall-clock seconds per token or speedup ratio); (b) Top-k Accuracy (probability mass of the correct token); (c) Task Success (e.g., percent of math problems solved correctly). (d) Scaling Efficiency.

We will compare FlashHead-LLaMA against the original LLaMA baseline. Experiments will run on modern GPU hardware (e.g., NVIDIA L40), using a framework like vLLM for consistency.

References

[1] Anonymous. Flashhead: Efficient drop-in replacement for the classification head in language model inference. ICLR 2026 (under review), 2026.

[2] Karl Cobbe et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[3] Embedl. Embedl sets a new standard for on-device llm inference. News Release, December 2025.

[4] Dan Hendrycks et al. Measuring mathematical problem solving with the math dataset. In *NeurIPS*, 2021.

[5] Jonathan Kernes. How to overcome the large vocabulary bottleneck using an adaptive softmax layer. Medium, 2021.

[6] Yuhui Li et al. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

[7] Yuhui Li et al. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.

[8] Hugo Touvron et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.