Optimising Mixture-of-Experts Serving through Semantic Caching

Part III Project Proposal

Yavuz Ferhatosmanoglu (yff23), Churchill College

Supervisors: Dr Eva Kalyvianaki and Wei Da

Abstract

With the rise of large language models, mixture-of-experts (MoE) architectures have emerged as a scalable way to increase model capacity without proportional increase in computation (DeepSeek-AI, 2025; Jiang et al., 2024; OpenAI, 2025). Yet, serving MoE models remains challenging due to routing overhead, expert parallelism, and limited opportunities for computation reuse (Yun et al., 2024; Zhu et al., 2025; Angelopoulos et al., 2025; Liu et al., 2025). This project proposes a novel direction in MoE serving optimisation: semantic caching of intermediate expert-layer outputs. Semantically similar inputs may produce reusable expert representations, and identifying this similarity through embedding comparison can enable skipping redundant expert computations. We aim to design, build and evaluate such a system, investigating the trade-offs between latency, throughput, and output quality.

1 Introduction

Mixture-of-expert (MoE) architectures introduce serving challenges: expert routing must be efficient and balanced, expert parallelism and disaggregation across GPUs introduce coordination overhead, and traditional caching is difficult to adapt to dynamic expert selection (Angelopoulos et al., 2025; Liu et al., 2025; Yun et al., 2024; Zhu et al., 2025). Prior work has explored optimisations such as quantisation with dynamic pruning, expert prefetching, and cache scheduling (Huang et al., 2025; Zhang et al., 2025).

Recent studies examine semantic caching at the prompt or response level (Bang, 2023; Gim et al., 2024; Jin et al., 2024), but internal caching within MoE layers remains unexplored. This project investigates *semantic caching of intermediate expert-layer outputs*, where similar inputs may allow reuse of prior expert computations and reduce inference cost.

Research question. *Can redundant expert computation in MoE serving be avoided through similarity-based caching of intermediate layer outputs, and how does this trade off latency and throughput against output quality?*

2 Approach

2.1 System Design

We will build a MoE serving system using open-source backends such as vLLM (Kwon et al., 2023) and SGlang (Zheng et al., 2024). The system will include:

1. **Vector stores** (e.g., Faiss (Douze et al., 2025)) containing embeddings of layerwise intermediate representations. Variants include:
   - Prompt-level stores that bypass all model computation (Bang, 2023).
   - Per-expert stores caching feed-forward outputs for selected MoE layers, based on per-token hidden state (e.g., 4096 dimensions).
   - Lower-dimensional projections (e.g., 4096 to 512) to trade accuracy for search speed.
2. **Dynamic similarity queries** during inference to retrieve nearest neighbours above a *similarity threshold*, enabling reuse of cached expert outputs with layer- and expert-specific metadata.
3. **Cache replacement policies** (e.g., FIFO or LRU) to manage capacity when inserting new embedding–output pairs.
4. **Tunable parameters** (e.g., similarity threshold, cache size, eviction policy, embedding dimension) to adjust based on models and application needs.

2.2 Evaluation

We will compare the cached MoE serving system with *uncached* and *prompt-level caching* baselines (e.g., Bang, 2023) along the following dimensions:

- **Latency & Throughput**: Time to first token, full-sequence latency, and requests per second.
- **Quality/Accuracy**: Output quality metrics such as perplexity, QA accuracy, or BLEU.
- **Cache hit rate**: Share of requests reusing cached outputs as a function of similarity thresholds, cache size, and workload characteristics.
- **Trade-off curves**: How caching parameters affect skipped computation, latency gains, and potential quality loss.

Models. We will evaluate models of varying scale to examine how caching efficiency changes with depth, expert count, and parameter size. Potential models include:

- **Small:** `Mixtral-2x0.3B` or `LLaMA-3-1B` for fast prototyping.
- **Medium:** `Mixtral-8x7B` or `DeepSeek-MoE-16B` on 2-3 GPUs for primary experiments, capturing realistic expert-parallel behaviour.
- **Large (optional):** Resources permitting, larger checkpoints (e.g., `Qwen2-MoE (72B)`) for scaling analysis.
- **Dense baselines:** Dense models (e.g., `LLaMA-3-7B`) to isolate MoE-specific effects.

Datasets. Workloads will combine repeated semantically-related queries (to induce cache hits) with diverse prompts (to force misses). Generation tasks will use public prompt or QA datasets (Kwiatkowski et al., 2019); structured tasks (e.g., translation, summarisation) would use standard benchmarks.

Experiments can be evaluated on both GPUs and CPUs.

3 Potential Outcomes

- An MoE serving system demonstrating semantic caching and expert-computation skipping.
- Quantitative results showing latency/throughput improvements, and trade-offs between hit rate, performance, and quality.
- Insights into when and how semantic caching is beneficial in MoE serving pipelines, focusing on practical factors such as similarity thresholds, embedding stability, and model-scale considerations.

4 Workplan

This section presents the expected timeline for completing the project. The plan accounts for other commitments and contingencies.

8 Dec - 21 Dec

- Conduct literature review on MoE serving systems, caching, and semantic reuse.
- Identify candidate datasets for evaluation tasks (e.g., QA, summarisation, translation).
- Set up access to departmental GPU resources.
- *Other commitments: Projects for R244, L48 and L101.*

22 Dec - 4 Jan

- Set up the development environment (vLLM, SGlang) and run initial tests.
- Milestone: Serve a small model locally (04/01/25).
- *Other commitments: Away for 1 week on holiday; projects for R244, L48 and L101.*

5 Jan - 18 Jan

- Set-up vector indexing libraries like Faiss or Milvus.
- Design the baseline MoE serving pipeline (no caching).
- Milestone: Test simple vector indexing (18/01/25).
- *Other commitments: Projects for R244, L48 and L101.*

19 Jan - 1 Feb

- Design the caching architecture: define embedding checkpoints, design cache replacement policies, similarity metric and thresholds.
- Implement baseline MoE serving pipeline (no caching).
- Milestone: Baseline serving pipeline operational (01/02/26).

2 Feb - 15 Feb

- Implement vector stores within the serving pipeline: embedding extraction, indexing, lookup, and reuse.
- Integrate prompt-level caching to the pipeline.
- Testing and optimisations as seen necessary.
- Milestone: Prototype MoE serving system with vector stores implemented (15/02/26).

16 Feb - 1 Mar

- Implement caching policies and threshold tuning within the pipeline.
- Design the experimental framework, covering the dimensions aforementioned.
- Prepare small workloads with both repeated and diverse queries to simulate cache hits/misses.
- Milestone: Initial workloads prepared (01/03/25).

2 Mar - 15 Mar

- Conduct initial experiments with small/medium-scale models and workloads, tuning similarity thresholds and cache parameters and collecting data.
- Identify larger workloads, with potentially different tasks.
- Testing and optimisations as seen necessary.
- Prepare project progress review report.
- Milestone: Initial experiments run and additional workloads prepared (15/03/26).
- *Other commitments: Projects for L193 and L48.*

16 Mar - 29 Mar

- Expand experiments to varied workloads and cache configurations.
- Scale experiments to medium-sized MoE models with multi-GPU serving, incorporating expert parallelism/pipeline, to investigate scaling behaviour.
- Milestone: Project progress review completed (20/03/26).
- *Other commitments: Projects for L193 and L48.*

30 Mar - 12 Apr

- Conduct ablation studies: cache replacement strategies, embedding dimensions, similarity thresholds.
- Analyse failure cases (diverse inputs, low hit rates).
- Testing and optimisations as seen necessary.
- *Other commitments: away for 1 week on holiday.*

13 Apr - 26 Apr

- Slack time/work on extensions: conduct large-scale experiments/additional optimisations/-final evaluation.
- Draft introduction and background sections of dissertation.
- *Other commitments: Multi-day UK sports tournament.*

27 Apr - 10 May

- Slack time/work on extensions: conduct large-scale experiments/additional optimisations/ final evaluation.
- Draft implementation sections of dissertation.
- Milestone: First draft of the initial sections submitted (10/05/26).

11 May - 24 May

- Draft evaluation and conclusion sections of dissertation.
- Revise initial dissertation sections based on feedback.
- Milestone: Full dissertation draft completed (24/05/26).

25 May - 7 June

- Finalise and revise dissertation based on feedback.
- Milestone: Final dissertation and source code submitted (01/06/26).

8 June - 18 June

- Prepare the final presentation.
- Milestone: Final presentation delivered (18/06/26).

References

Angelopoulos, S., Marchal, L., Obrecht, A., & Simon, B. (2025, August). Cache management for mixture-of-experts llms. In *Euro-par 2025: Parallel processing* (pp. 18–32). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-99872-0_2

Bang, F. (2023, December). GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In L. Tan, D. Milajevs, G. Chauhan, J. Gwinnup, & E. Rippeth (Eds.), *Proceedings of the 3rd workshop for natural language processing open source software (nlp-oss 2023)* (pp. 212–218). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.nlposs-1.24

DeepSeek-AI. (2025). Deepseek-v3 technical report. https://arxiv.org/abs/2412.19437

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., & Jégou, H. (2025). The faiss library. https://arxiv.org/abs/2401.08281

Gim, I., Chen, G., Lee, S.-s., Sarda, N., Khandelwal, A., & Zhong, L. (2024). Prompt cache: Modular attention reuse for low-latency inference. https://arxiv.org/abs/2311.04934

Huang, W., Liao, Y., Liu, J., He, R., Tan, H., Zhang, S., Li, H., Liu, S., & QI, X. (2025). Mixture compressor for mixture-of-experts LLMs gains more. *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=hheFYjOsWO

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., . . . Sayed, W. E. (2024). Mixtral of experts. https://arxiv.org/abs/2401.04088

Jin, C., Zhang, Z., Jiang, X., Liu, F., Liu, X., Liu, X., & Jin, X. (2024). Ragcache: Efficient knowledge caching for retrieval-augmented generation. https://arxiv.org/abs/2404.12457

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., & Petrov, S. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, *7*, 453–466. https://doi.org/10.1162/tacl_a_00276

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient memory management for large language model serving with pagedattention. https://arxiv.org/abs/2309.06180

Liu, D., Yu, Y., Lengerich, B., Wu, Y. N., & Wang, X. (2025). Pikv: Kv cache management system for mixture of experts. https://arxiv.org/abs/2508.06526

OpenAI. (2025). Gpt-5 system card. https://cdn.openai.com/gpt-5-system-card.pdf

Yun, L., Zhuang, Y., Fu, Y., Xing, E. P., & Zhang, H. (2024). Toward inference-optimal mixture-of-expert large language models. https://arxiv.org/abs/2404.02852

Zhang, Y., Pinkert, G., Yang, N., Li, Y., & Yuan, D. (2025). Duoserve-moe: Dual-phase expert prefetch and cache scheduling for efficient moe llm inference. https://arxiv.org/abs/2509.07379

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Barrett, C., & Sheng, Y. (2024). Sglang: Efficient execution of structured language model programs. https://arxiv.org/abs/2312.07104

Zhu, R., Jiang, Z., Jin, C., Wu, P., Stuardo, C. A., Wang, D., Zhang, X., Zhou, H., Wei, H., Cheng, Y., Xiao, J., Zhang, X., Liu, L., Lin, H., Chang, L.-W., Ye, J., Yu, X., Liu, X., Jin, X., & Liu, X. (2025). Megascale-infer: Serving mixture-of-experts at scale with disaggregated expert parallelism. https://arxiv.org/abs/2504.02263