Master's Thesis Proposal

A Heterogeneous Acceleration System For Efficient Long-Context LLM Inference Using KV Cache Vector Retrieval

Timon Fercho* Hongshi Tan† Yu Feng† Yao Chen†
Bingsheng He† Gustavo Alonso*

September 2024

1 Introduction

Long-context LLM inference Large Language Models (LLMs) have gained significant attention in recent years due to their exceptional performance in solving various natural language processing (NLP) and general-purpose tasks [50]. Increasingly demanding applications like chain-of-thought [38] reasoning, code analysis [9], information retrieval [26] and synthesizing multi-modal data such as images, audio, or video require LLMs to support longer context lengths [42, 52, 16, 34].

With input sizes approaching millions of tokens [16, 34] and the memory required for caching activations of previous tokens growing linearly with the sequence length, the key-value (KV) cache [31] becomes the main bottleneck for long context inference [42, 43]. In addition, larger batch sizes and the implications of the memory wall [11] phenomenon only worsen the issue [16, 20].

Addressing the KV cache memory bottleneck Because model compression techniques such as weight quantization are not sufficient for reducing memory consumption for large input sequences [43], extensive research efforts have recently been devoted to mitigating the KV cache memory bottleneck, such as KV cache quantization [16, 30, 44], compression [49, 41, 27, 7], and offloading [18, 14, 51, 25].

In this context, recent work on dynamic sparse attention has revealed that the attention score can be effectively used to estimate the relevance of previously generated tokens [52, 10, 29, 35]. In contrast to some KV cache compression

*ETH Zurich
†National University of Singapore

methods that prune tokens based on attention scores, PQCache [46] and RetrievalAttention [28] adopt a different strategy by offloading the KV cache to CPU memory and framing the retrieval of tokens from the KV cache as a maximum inner product search (MIPS) problem. This innovative approach allows for applying approximate nearest neighbor search (ANNS) techniques, enabling high-recall retrieval from the KV cache.

Opportunities for heterogeneous and disaggregated inference While their approach significantly improves accuracy over previous KV compression techniques on long-context tasks, it faces increased inference latency during the decoding phase, where token retrieval occurs on the critical path of each generation step. Both PQCache [46] and RetrievalAttention [28] utilize the CPU for product quantization (PQ) [21] or inverted file index (IVF) [2] construction during the prefilling phase, as well as for the retrieval of relevant tokens during the decoding phase. For the latency-critical decoding phase, we see significant potential for system-level optimizations by leveraging FPGAs to accelerate ANNS, as they have demonstrated considerable success in this domain [23, 22, 47].

These observations lead us to the key question we aim to address: *Can we build a heterogenous acceleration system for efficient long-context LLM inference using KV cache offloading and dynamic sparse attention, which leverages ANNS to achieve both high accuracy and low latency generation during the decoding phase?* In this work, we want to primarily focus on the decoding stage, as previous studies [4, 3, 15] have shown that FPGAs are especially effective at accelerating the memory-bound generation.

As a natural extension of this question, one could imagine also quantizing the KV cache or offloading the sparse attention computation, which directly follows the token retrieval. While quantization of the KV cache has shown much promise as an orthogonal optimization to compression techniques [16, 30, 44], the latter is motivated by the observation that FPGAs have been efficiently used to perform sparse attention computation [45, 37, 19, 15, 8, 33]. Given recent works on designing inference serving systems, which demonstrate the performance benefits from disaggregating the prefilling and decoding phases [14, 51, 18], we conclude that exploring this research direction holds great potential.

2 Work plan

The work could consist of the following units:

1. Conduct a literature review on KV cache offloading, dynamic sparse attention, and existing FPGA-based ANNS acceleration systems.

2. Experiment with and evaluate existing KV cache offloading and GPU-based sparse attention implementations. Report accuracy, latency, throughput, and resource usage of existing systems.

3. Design and implement an efficient system for long-context inference: During the prefilling phase, it offloads the KV cache for clustering/index construction to CPU memory. During the decoding phase, it retrieves tokens from the KV cache using FPGA-accelerated ANNS [1] and generates tokens using dynamic sparse attention on the GPU.

4. Evaluate the system in terms of accuracy, latency, throughput, and resource utilization, emphasizing performance during the decoding phase for long-context tasks, and compare these results against existing studies.

5. Optional: Explore sparse attention acceleration on FPGAs and extend the design to enable also offloading the sparse attention computation.

3 Prerequisites

With a strong background in computer systems, information retrieval, machine learning, and computer architecture—further strengthened by hands-on experience with Approximate Nearest Neighbor Search (ANNS) systems during my bachelor's thesis under Prof. Alonso, as well as my role as a teaching assistant for "VLSI 1: HDL-based Design for FPGAs"—, I believe I am well-equipped to undertake this research project. My recent work, where I contributed to the design of an integer transformer accelerator as part of a research project with Prof. Benini [39], has further strengthened my expertise at the intersection of hardware design and machine learning. These experiences have prepared me to bridge the gap between the relevant domains and address the challenges inherent in this project. Additionally, having conducted an initial review of recent literature and built a strong foundation during my studies, I am confident in acquiring the advanced skills required and diving deeper into the related research areas.

Finally, a successful research project thrives on collaboration and proper guidance within a suitable research environment. Under the supervision of Prof. Alonso, Prof. He, and their students - whose labs have made notable contributions in related areas - I believe we have an excellent foundation with the expertise and resources available to pursue this exciting research endeavor.

4 Appendix

In this section, we examine the design choices and potential overheads introduced by PQCache and RetrievalAttention during the prefilling and decoding phases, evaluate their overall performance in terms of latency and accuracy, and present key findings from existing works on FPGA-based acceleration for efficient LLM inference and approximate nearest neighbor (ANN) search.

[1] Implementation could be done using high-level synthesis (HLS) [5] and may be supported by open-sourced kernel libraries or accelerator design languages such as Allo [3, 4, 13, 12].

Prefilling overheads During the prefilling phase, PQCache and RetrievalAttention offload token keys and values for each attention head in a pipelined manner. Both approaches mitigate communication overhead by overlapping it with GPU-based attention computation.

For index construction, PQCache opts for lightweight product quantization (PQ) [21] over graph-based indices [32, 36] due to the lower computational cost of PQ. RetrievalAttention, in contrast, employs a more complex indexing technique that reduces the number of vectors to scan by mapping queries to key vectors. Importantly, neither approach rebuilds the index during the decoding phase; instead, newly generated tokens are added to the key-value (KV) cache without modifying the index.

Constructing the PQ codes involves running the k-means algorithm on the key vectors, which, by default, has a computational complexity of $\mathcal{O}(s \cdot d \cdot T)$ [46]. This complexity limits the ability to fully overlap the k-means execution with attention computation, particularly for shorter input sequences. To address this, PQCache introduces an adaptive k-means strategy that reduces the number of iterations T, trading off a slight reduction in accuracy for better latency. The performance implications of these design choices will be discussed in the following sections.

Quantitative analysis of existing approaches Both PQCache [46] and RetrievalAttention [28] utilize ANN retrieval to efficiently select key tokens during decoding. Across various benchmarks, including LongBench [1], InfinityBench [48], RULER [17], and Needle-in-a-Haystack [24], these methods consistently outperform their respective baselines, often approaching the performance of full-attention computation.

In terms of prefilling performance, PQCache achieves the lowest time-to-2nd-token latency compared to its baselines by effectively hiding communication and index construction overheads. Unlike approaches like H2O [49], PQCache leverages the FlashAttention optimization [6], as it does not require materializing intermediate attention scores. Moreover, it avoids additional overheads, such as the block-level KV cache management used in InfLLM [40]. In contrast, RetrievalAttention does not report prefilling latency, which may suggest higher costs associated with its more complex index construction.

However, both PQCache and RetrievalAttention exhibit performance drawbacks during the decoding phase. Unlike earlier approaches that perform attention over a fixed number of tokens [28], these ANN-based methods require scanning and attending to a significant proportion of KV cache vectors to maintain high recall and accuracy. Scanning between 3% and 20% of the KV cache vectors [46, 28] introduces a vector search bottleneck, with overheads increasing as the context length grows. For example, RetrievalAttention reports that vector search accounts for 34%-67% of decoding latency for a 128K context, depending on the index used [28]. Additionally, sparse attention computation consumes 20%-40% of the decoding latency. Overall, PQCache and RetrievalAttention are outperformed by their baselines by 2x to 7x during the decoding phase.

To address these challenges, heterogeneous acceleration emerges as a promising solution for enhancing ANN-based sparse attention computation. Notably, FPGAs have shown 2x-6x speedups over CPUs in ANN tasks [22, 23] and can outperform GPUs in terms of token generation latency by 20x-100x for memory-bound workloads like the decoding phase [4]. This suggests significant promise for FPGA-based acceleration in KV cache retrieval and sparse attention computation, particularly for long-context LLM inference.

References

[1] Yushi Bai et al. "LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding". In: *ArXiv* abs/2308.14508 (2023). URL: `https://api.semanticscholar.org/CorpusID:261245264`.

[2] Dmitry Baranchuk, Artem Babenko, and Yury Malkov. "Revisiting the Inverted Indices for Billion-Scale Approximate Nearest Neighbors". In: *ArXiv* abs/1802.02422 (2018).

[3] Hongzheng Chen et al. "Allo: A Programming Model for Composable Accelerator Design". In: *Proceedings of the ACM on Programming Languages* 8 (2024), pp. 593–620.

[4] Hongzheng Chen et al. "Understanding the Potential of FPGA-Based Spatial Acceleration for Large Language Model Inference". In: *ACM Transactions on Reconfigurable Technology and Systems* (2023).

[5] Jason Cong et al. "FPGA HLS Today: Successes, Challenges, and Opportunities". In: *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 15 (2022), pp. 1–42.

[6] Tri Dao et al. "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness". In: *ArXiv* abs/2205.14135 (2022).

[7] Harry Dong et al. "Get More with LESS: Synthesizing Recurrence with KV Cache Compression for Efficient LLM Inference". In: *ArXiv* abs/2402.09398 (2024).

[8] Hongxiang Fan et al. "Adaptable Butterfly Accelerator for Attention-based NNs via Hardware and Algorithm Co-design". In: *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)* (2022), pp. 599–615.

[9] Chongzhou Fang et al. "Large Language Models for Code Analysis: Do LLMs Really Do Their Job?" In: *ArXiv* abs/2310.12357 (2023).

[10] Suyu Ge et al. "Model Tells You What to Discard: Adaptive KV Cache Compression for LLMs". In: *ArXiv* abs/2310.01801 (2023).

[11] Amir Gholami et al. "AI and Memory Wall". In: *IEEE Micro* 44 (2024), pp. 33–39.

[12] Jude Haris et al. "Designing Efficient LLM Accelerators for Edge Devices". In: *ArXiv* abs/2408.00462 (2024).

[13] Andy He et al. “HLSTransform: Energy-Efficient Llama 2 Inference on FPGAs Via High Level Synthesis”. In: *ArXiv* abs/2405.00738 (2024).

[14] Jiaao He and Jidong Zhai. “FastDecode: High-Throughput GPU-Efficient LLM Serving using Heterogeneous Pipelines”. In: *ArXiv* abs/2403.11421 (2024).

[15] Seongmin Hong et al. “DFX: A Low-latency Multi-FPGA Appliance for Accelerating Transformer-based Text Generation”. In: *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)* (2022), pp. 616–630.

[16] Coleman Hooper et al. “KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization”. In: *ArXiv* abs/2401.18079 (2024).

[17] Cheng-Ping Hsieh et al. “RULER: What’s the Real Context Size of Your Long-Context Language Models?” In: *ArXiv* abs/2404.06654 (2024). URL: `https://api.semanticscholar.org/CorpusID:269032933`.

[18] Cunchen Hu et al. “Inference without Interference: Disaggregate LLM Inference for Mixed Downstream Workloads”. In: *ArXiv* abs/2401.11181 (2024).

[19] Mingqiang Huang et al. “EdgeLLM: A Highly Efficient CPU-FPGA Heterogeneous Edge Accelerator for Large Language Models”. In: *ArXiv* abs/2407.21325 (2024).

[20] Yingbing Huang et al. “New Solutions on LLM Acceleration, Optimization, and Application”. In: *ArXiv* abs/2406.10903 (2024).

[21] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. “Product Quantization for Nearest Neighbor Search”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (2011), pp. 117–128.

[22] Wenqi Jiang et al. “Chameleon: a Heterogeneous and Disaggregated Accelerator System for Retrieval-Augmented Language Models”. In: *ArXiv* abs/2310.09949 (2023).

[23] Wenqi Jiang et al. “Co-design Hardware and Algorithm for Vector Search”. In: *SC23: International Conference for High Performance Computing, Networking, Storage and Analysis* (2023), pp. 1–16.

[24] Greg Kamradt. *Needle-in-a-Haystack*. GitHub repository. 2024. URL: `https://github.com/gkamradt/LLMTest_NeedleInAHaystack`.

[25] Wonbeom Lee et al. “InfiniGen: Efficient Generative Inference of Large Language Models with Dynamic KV Cache Management”. In: *USENIX Symposium on Operating Systems Design and Implementation.* 2024.

[26] Patrick Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”. In: *ArXiv* abs/2005.11401 (2020).

[27] Yuhong Li et al. “SnapKV: LLM Knows What You are Looking for Before Generation”. In: *ArXiv* abs/2404.14469 (2024).

[28] Di Liu et al. “RetrievalAttention: Accelerating Long-Context LLM Inference via Vector Retrieval”. In: 2024.

[29] Zichang Liu et al. “Scissorhands: Exploiting the Persistence of Importance Hypothesis for LLM KV Cache Compression at Test Time”. In: *ArXiv* abs/2305.17118 (2023).

[30] Zirui Liu et al. “KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache”. In: *ArXiv* abs/2402.02750 (2024).

[31] Shi Luohe et al. “Keep the Cost Down: A Review on Methods to Optimize LLM’ s KV-Cache Consumption”. In: *ArXiv* abs/2407.18003 (2024).

[32] Yury Malkov and Dmitry A. Yashunin. “Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2016), pp. 824–836. URL: https://api.semanticscholar.org/CorpusID:8915893.

[33] Hongwu Peng et al. “A length adaptive algorithm-hardware co-design of transformer on FPGA through sparse attention and dynamic pipelining”. In: *Proceedings of the 59th ACM/IEEE Design Automation Conference* (2022).

[34] Machel Reid et al. “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context”. In: *ArXiv* abs/2403.05530 (2024).

[35] Luka Ribar et al. “SparQ Attention: Bandwidth-Efficient LLM Inference”. In: *ArXiv* abs/2312.04985 (2023).

[36] Suhas Jayaram Subramanya et al. “DiskANN : Fast Accurate Billion-point Nearest Neighbor Search on a Single Node”. In: 2019. URL: https://api.semanticscholar.org/CorpusID:209392043.

[37] Hanrui Wang, Zhekai Zhang, and Song Han. “SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning”. In: *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (2020), pp. 97–110.

[38] Jason Wei et al. “Chain of Thought Prompting Elicits Reasoning in Large Language Models”. In: *ArXiv* abs/2201.11903 (2022).

[39] Philip Wiese et al. “Toward Attention-based TinyML: A Heterogeneous Accelerated Architecture and Automated Deployment Flow”. In: *ArXiv* abs/2408.02473 (2024).

[40] Chaojun Xiao et al. “InfLLM: Training-Free Long-Context Extrapolation for LLMs with an Efficient Context Memory”. In: 2024.

[41] Guangxuan Xiao et al. “Efficient Streaming Language Models with Attention Sinks”. In: *ArXiv* abs/2309.17453 (2023).

[42] Jiayi Yuan et al. “KV Cache Compression, But What Must We Give in Return? A Comprehensive Benchmark of Long Context Capable Approaches”. In: *ArXiv* abs/2407.01527 (2024).

[43] Zhihang Yuan et al. “LLM Inference Unveiled: Survey and Roofline Model Insights”. In: *ArXiv* abs/2402.16363 (2024).

[44] Yuxuan Yue et al. “WKVQuant: Quantizing Weight and Key/Value Cache for Large Language Models Gains More”. In: *ArXiv* abs/2402.12065 (2024).

[45] Shulin Zeng et al. “FlightLLM: Efficient Large Language Model Inference with a Complete Mapping Flow on FPGAs”. In: *Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (2024).

[46] Hailin Zhang et al. “PQCache: Product Quantization-based KVCache for Long Context LLM Inference”. In: *ArXiv* abs/2407.12820 (2024).

[47] Jialiang Zhang, Soroosh Khoram, and Jing Jane Li. “Efficient Large-Scale Approximate Nearest Neighbor Search on OpenCL FPGA”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 4924–4932.

[48] Xinrong Zhang et al. “Bench: Extending Long Context Evaluation Beyond 100K Tokens”. In: *ArXiv* abs/2402.13718 (2024). URL: `https://api.semanticscholar.org/CorpusID:267770255`.

[49] Zhenyu (Allen) Zhang et al. “H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models”. In: *ArXiv* abs/2306.14048 (2023).

[50] Wayne Xin Zhao et al. “A Survey of Large Language Models”. In: *ArXiv* abs/2303.18223 (2023).

[51] Yinmin Zhong et al. “DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving”. In: *USENIX Symposium on Operating Systems Design and Implementation.* 2024.

[52] Zixuan Zhou et al. “A Survey on Efficient Inference for Large Language Models”. In: *ArXiv* abs/2404.14294 (2024).