Towards Real-Time LLM Inference on Heterogeneous Edge Platforms

Rakshith Jayanth[†], Neelesh Gupta[†], Souvik Kundu[‡], Deepak A Mathaikutty[‡], Viktor Prasanna[†]
[†]*University of Southern California,* [‡]*Intel*
[†]{jayanthr, neeleshg, prasanna}@usc.edu, [‡]{souvikk.kundu, deepak.a.mathaikutty}@intel.com

***Abstract*—The deployment of computationally intensive workloads, such as Large Language Models (LLMs), at the edge presents significant challenges due to resource constraints and limited computing power. State-of-the-art edge platforms address this with integrated high-performance accelerators optimized for machine learning workloads. While numerous high-performance edge platforms are now available in the market, there is limited exploration of methods to maximize performance and optimize resource utilization at the edge. Our work aims to address this gap by proposing methodologies for maximizing performance through the distribution of model inference on a heterogeneous edge platform. This poster presents preliminary findings from our ongoing research project funded by NSF IUCRC IDEAS Center. The Center is focused on Edge Intelligence and applications that can be potentially mapped to Edge platforms.**

***Index Terms*—Edge Computing, LLM Inference, SoC**

I. Introduction

The deployment of LLMs on edge platforms is gaining significant importance as it not only reduces communication latencies but also enhances data privacy. Several vendors are developing System-on-Chips (SoCs) to support compute-intensive AI workloads. Examples include Intel's AI PCs (featuring Meteor Lake processors) [1] and AMD's AI PCs (based on Ryzen processors) [2]. These advanced processors typically integrate multi-core CPUs, GPUs, and specialized AI accelerators like Neural Processing Units (NPUs).

However, despite the availability of complex SoCs, edge platforms have challenges due to limited processing power and memory bandwidth compared to data centers. Also, the computational demands of LLM inference often exceed the capabilities of a single computing device on edge platforms. Previous works addressed these limitations by either performing an edge-server collaborative inference [3], [4] or, distributing the inference across homogeneous compute units on multiple edge platforms [5]. In contrast, we propose an efficient mapping of computations across multiple cores of an integrated heterogeneous platform to effectively manage demanding workloads. The main contributions are as follows:

- We propose a framework for efficient mapping of LLM inference across heterogeneous compute units with optimal memory management.
- We analyze the compute-bound and memory-bound characteristics of LLM inference.
- We present an initial analysis of LLM performance on an integrated heterogeneous platform.

II. Background

Edge heterogeneous platforms have multicore CPUs integrated with GPUs and specialized accelerators such as NPUs. Integrated cores in edge platforms have a uniform shared memory architecture and are connected via a common interconnection network. Cores are equipped with Direct Memory Access (DMA) units to boost memory bandwidth.

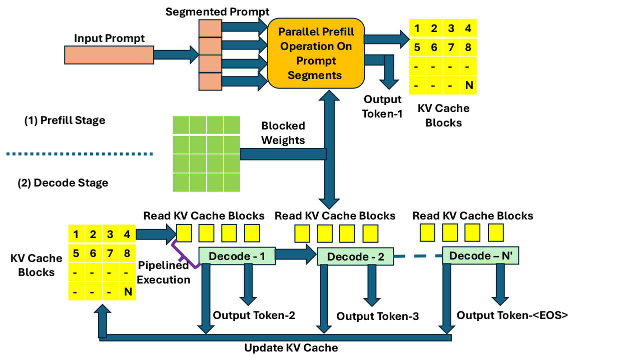

Fig. 1. Proposed Workflow of Prefill and Decode stages

III. Proposed Methodology

We analyze the compute stages, prefill and decode, of generative LLM inference based on their performance on each available core to perform a core-stage mapping. The primary criteria while mapping is the compute-bound and memory-bound characteristics of the stages. This is *static mapping* as the mapping is done only once.

During LLM inference, a Key-Value(KV) cache [6] is generated to optimize memory utilization in self-attention layers across multiple decoding steps. The KV cache size increases linearly with context length, posing challenges for limited on-chip memory and bandwidth on edge platforms. Due to the limited on-chip memory bandwidth, the transfer of the entire KV cache from the prefill unit to the decode unit creates unnecessary wait cycles in the decode unit. To mitigate this overhead, we propose to segment the prefill stage by splitting the prompt into multiple fixed-size segments and processing them in parallel, as shown in Fig. 1, resulting in blocks of KV cache. Blocked KV cache processing will reduce memory fragmentation along with decreasing the latency in accessing past tokens and their values. During the decode stage, KV cache blocks are read sequentially, with the processing of

TABLE I
PROFILING RESULTS OF LLM MODELS [1]
(BATCH SIZE: 1, PRECISION: INT4, PROMPT LENGTH: 128 TOKENS)

Metric	TinyLlama-1B		Llama3-8B	
	CPU	GPU	CPU	GPU
Prefill Time (s)	4.95	0.18	31.00	1.07
Time per Output Token (Sec/Token)	0.20	0.06	1.18	0.22
Throughput (Tokens/Sec)	4.48	15.60	0.47	3.94

previously read blocks executed in parallel with the reading of new blocks, resulting in an asynchronous overlap of computation and communication in the decode unit. At the end of each iteration, the key and value vectors are updated to the KV cache. For determining optimal block size, we perform an offline design-space exploration by profiling the performance of the compute units and the platform interconnect on a range of block sizes to analyze their compute and communication times and find an optimal block size with minimal communication, resulting in zero wait cycles in the decode unit.

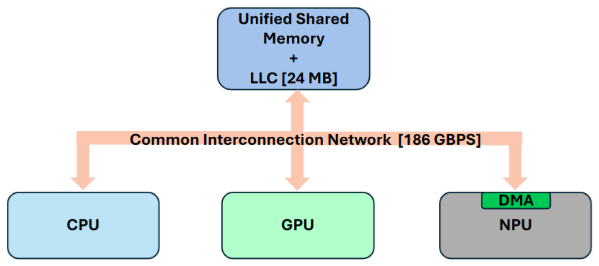
Unified Shared Memory + LLC [24 MB]
Common Interconnection Network [186 GBPS]
CPU
GPU
DMA
NPU

Fig. 2. Device Interaction on Intel AI PC

IV. PRELIMINARY RESULTS

This section presents preliminary results from our experimental evaluation. The target hardware for our experiments is the Intel Core Ultra 7 165H Processor that consists of a CPU integrated with a GPU and NPU.

1) Peak Performance: In order to perform efficient mapping of stages on the heterogeneous platform, we analyzed the GPU and NPU based on their raw computing power. The peak performance of GPU [7]for both FP16 operations (32.8 TFLOPS) and Int8 operations (66 TOPS) supersedes NPU's performance (4 TFLOPS and 8 TOPS, respectively), making it a preferred choice for compute-intensive operations. NPU's architecture [8] supports DMA, improving memory bandwidth. Since all the cores are connected via a common interconnect, the performance of NPU, due to DMA, can potentially see a boost in memory-intensive operations.

2) LLM Computational Characteristics: The major stages in LLM inference are the prefill stage and the decode stage. The prefill stage processes the complete prompt, where the major operation is matrix multiplication, making it a compute-bound stage. In the decode stage, one token is generated at a time autoregressively, and the main operation here is matrix-vector multiplication, making it a memory-bound stage.

[1]Note: Profiling results may vary across different software stack and configurations. The preliminary results presented here are specific to the evaluation framework used in this study for representative academic purpose.

3) Static Mapping of LLM Inference: Considering the mapping of LLM inference on the available cores, we performed an initial analysis on the CPU and the integrated GPU. For our analysis, we considered the TinyLlama model having 1B parameters and the Llama3 model with 8B parameters. From Table I, it is evident that the performance of the GPU is better than the CPU for both the prefill and the decode stages, as indicated by the Time Per Output Token. It should also be noted that the GPU performance degrades in Llama3-8B in comparison to TinyLlama-1B. Since the model is larger, there will be an increase in the number of layers, leading to lower throughput. Along with this, for larger models, the computation becomes memory-bandwidth intensive, potentially affecting GPU's performance. In short, in an environment that has only a CPU and GPU, it is preferred to perform inference on GPU.

V. CONCLUSION

In this poster, we addressed the requirement for mapping LLM inference on the edge heterogeneous platform and presented our ongoing work on developing a framework for efficient LLM inference. We briefly discussed the initial profiling results on two popular LLMs. We target the Llama 3 model family for further analysis and experiments.

ACKNOWLEDGMENT

This work is supported by Intel Labs and the National Science Foundation(NSF) IDEAS Center [9] grants under the award numbers SaTC-2104264 and CNS-2009057.

REFERENCES

[1] W. Gomes, S. Morgan, B. Phelps, T. Wilson, and E. Hallnor, "Meteor lake and arrow lake intel next-gen 3d client architecture platform with foveros," in *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022, pp. 1–40.

[2] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families : Industrial product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 57–70.

[3] F. Cai, D. Yuan, Z. Yang, and L. Cui, "Edge-llm: A collaborative framework for large language model serving in edge computing," in *2024 IEEE International Conference on Web Services (ICWS)*, 2024.

[4] Z. Yang, Y. Yang, C. Zhao, Q. Guo, W. He, and W. Ji, "Perllm: Personalized inference scheduling with edge-cloud collaboration for diverse llm services," 2024. [Online]. Available: https://arxiv.org/abs/2405.14636

[5] M. Zhang, J. Cao, X. Shen, and Z. Cui, "Edgeshard: Efficient llm inference via collaborative edge computing," 2024. [Online]. Available: https://arxiv.org/abs/2405.14371

[6] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean, "Efficiently scaling transformer inference," in *Proceedings of Machine Learning and Systems*, D. Song, M. Carbin, and T. Chen, Eds., vol. 5. Curan, 2023, pp. 606–624.

[7] Intel-gpu. [Online]. Available: https://www.intel.com/content/www/us/en/content-details/819275/unlocking-the-ai-power-of-intel-arc-gpu-for-the-edge-a-deep-dive-into-hardware-and-software-enablement.html

[8] Intel-npu. [Online]. Available: https://intel.github.io/intel-npu-acceleration-library/npu.html

[9] Nsf-iucrc-ideas-center. [Online]. Available: https://iucrc.nsf.gov/centers/center-for-intelligent-distributed-embedded-applications-and-systems-ideas/