Mapping Large Language Model Layers on a Multicore Architecture to Reduce Energy-Delay Product

Kush Rohra
(13-19-02-19-52-23-1-23641)
Department: DSBA
M.TECH 1ST YEAR, IISC
Bangalore, India

Padam Singh
(13-19-02-19-52-23-1-23807)
Department: DSBA
M.TECH 1ST YEAR, IISC
Bangalore, India

***Abstract*—The objective of this project is to map different computation blocks of Large Language Models (LLMs) onto a multicore architecture with the aim of minimizing the Energy-Delay Product (EDP). This involves a comprehensive understanding of both the software (LLM architectures) and hardware (multicore platforms), as well as implementing effective strategies for parallelism and optimization.**

I. Introduction

Large Language Models (LLMs) are a major part of Natural Language Processing (NLP) problems because they support a lot of tasks like text generation, completion, and comprehension. However, they are computationally very expensive in terms of both energy consumption and processing speed requirements. This project will try to optimize the execution of LLMs on multicore architecture to reduce Energy Delay Product (EDP). By mapping different computation blocks of LLM to specific cores or threads, we aim to achieve an optimal balance between energy consumption and processing speed. This study will observe at least 5 different LLM models and demonstrate the efficacy of our approach through experiments and drawn analysis.

II. Related Work

The optimization of Neural Networks on multicore architectures is an active research area. There are many studies that have explored methods to enhance performance and reduce energy consumption for neural networks. Despite these advancements, research specifically targeting the mapping of LLM layers to minimize EDP is limited.

Some of the papers on this topic are as follows:

- *Efficient LLM Inference on CPUs*: This paper explores methods to optimize the inference performance of large language models (LLMs) when running on CPU-based systems. It covers techniques such as quantization, pruning, and software optimizations to reduce the computational load and improve efficiency without compromising accuracy.
- *REDUCT: Keep it Close, Keep it Cool!*: This paper focus on strategies to minimize data movement and manage thermal dissipation in neural networks, particularly for large models. The title suggests an emphasis on keeping data close to the processing units to reduce latency and power consumption, thereby maintaining cooler operating temperatures.
- *Optimizing Inference Performance of Transformers on CPUs*: This paper discusses various methods to enhance the performance of transformer models during inference on CPU architectures. It includes algorithmic improvements, hardware-specific optimizations, and resource management techniques to make transformers more efficient and faster when running on CPUs.

III. Methodology

1) **Model and Data Preparation**
   a) We loaded a pre-trained large language model (LLM) along with its corresponding tokenizer.
   b) For this study, we selected five pre-trained LLMs with varying architectures and use cases, such as BERT, GPT2, Electra, T5, and RoBERTa.
   c) We encoded the input text data using the loaded tokenizer to format it appropriately for the selected LLM.
2) **Selecting a Suitable Hardware Platform**
   a) We chose a multicore platform with a sufficient number of cores (e.g., 8-16 cores) and the capability to measure energy consumption.
   b) For this study, we utilized a virtual machine (VM) with the following specifications: 32 vCPUs and 64 GB of memory.
3) **Layer-wise Profiling**
   a) We iterated through each transformer layer in the encoder of the selected LLM.
   b) We measured the time taken by each layer to process individual input tokens.
   c) We calculated the average processing time for each layer across all input tokens.
4) **Energy-Delay Product (EDP) Calculation**

a) We calculated the EDP for each layer by multiplying the average layer duration by the assumed power consumption, set to 1.0 in this study.

5) **EDP Improvement Compared to Baseline Model**

a) We profiled each model layer while executing on different cores or threads.
b) We recalculated the average processing time for each layer across all input tokens.
c) We compared the new average processing times to those of the baseline model to determine the improvements.

6) **Documentation**

a) As a final step, we utilized built-in Python libraries to visualize the results of each LLM across different datasets and plotted the EDP improvements for each LLM layer.

IV. Results

For evaluating the model's performance across different layers, three distinct datasets were utilized:

- ag_news: This dataset comprises news articles categorized into four classes: World, Sports, Business, and Sci/Tech
- yelp_reviews: This dataset includes customer reviews from Yelp, focusing on sentiment analysis tasks.
- amazon_polarity: This dataset involves reviews from Amazon products, used for sentiment analysis where reviews are labeled as either positive or negative.

Each dataset served to assess how the model's performance varied across its layers, providing insights into the effectiveness and improvement of the model's capabilities for different types of textual data.

A. BERT Model

For a Pretrained BERT Model, we got the following results:

- We saw an overall improvement in execution times from 3.5% to 26%

		BERT Model		
		Baseline EDP	Improved EDP	Percentage Improvement
ag_news	Mean	0.036756	0.03539	3.59
	Total	0.441075	0.424678	
amazon_polarity	Mean	0.04321	0.031833	26.32
	Total	0.518523	0.381994	
yelp_review_full	Mean	0.054982	0.052939	3.66
	Total	0.6597868	0.635276	

(a) BERT Model Results against different datasets

(b) Performance improvements on various datasets using the BERT model. Subfigures show specific aspects of the results.

Fig. 1: BERT Model Comparisons

B. Electra Model

For a Pretrained Electra Model, we got the following results:

- We saw an overall improvement in execution times from 3.5% to 15%

		Electra Model		
		Baseline EDP	Improved EDP	Percentage Improvement
ag_news	Mean	0.06998791	0.05892462	15.45
	Total	1.67971	1.414191	
amazon_polarity	Mean	0.08051995	0.07764873	3.56
	Total	1.9324788	1.86356952	
yelp_review_full	Mean	0.130615	0.11874	9.09
	Total	3.134762	2.85056	

(a) Electra Model Results against different datasets

(b) Performance improvements on various datasets using the Electra model. Subfigures show specific aspects of the results.

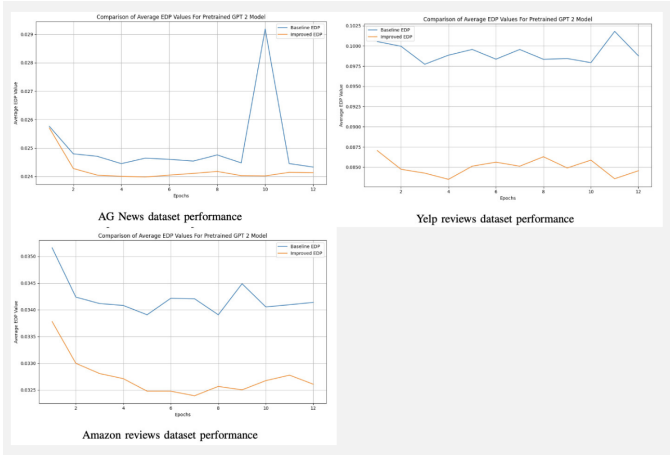
Fig. 2: Electra Model Comparisons

C. GPT-2 Model

For a Pretrained GPT-2 Model, we got the following results. We saw an overall improvement in execution times from 3.29% to 13%

		GPT2 Model		
		Baseline EDP	Improved EDP	Percentage Improvement
ag_news	Mean	0.02509885	0.0242151	3.19
	Total	0.30118622	0.29058145	
amazon_polarity	Mean	0.0341834	0.0326189	4.57
	Total	0.4102008	0.3913741	
yelp_review_full	Mean	0.09944	0.085786	13.72
	Total	1.193283	1.029438	

(a) GPT-2 Model Results against different datasets



(b) Performance improvements on various datasets using the GPT-2 model. Subfigures show specific aspects of the results.

Fig. 3: GPT-2 Model Comparisons

D. RoBERTa Model

For a Pretrained RoBERTa Model, we got the following results. We saw an overall improvement in execution times upto 13

		RoBERTa Model		
		Baseline EDP	Improved EDP	Percentage Improvement
ag_news	Mean	0.02387747	0.02948048	-23.43
	Total	0.286529	0.353765	
amazon_polarity	Mean	0.0303839	0.0349944	-15.16
	Total	0.3646072	0.4199324	
yelp_review_full	Mean	0.21993	0.189623	13.76
	Total	2.639161	2.275477	

(a) RoBERTa Model Results against different datasets

(b) Performance improvements on various datasets using the RoBERTa model

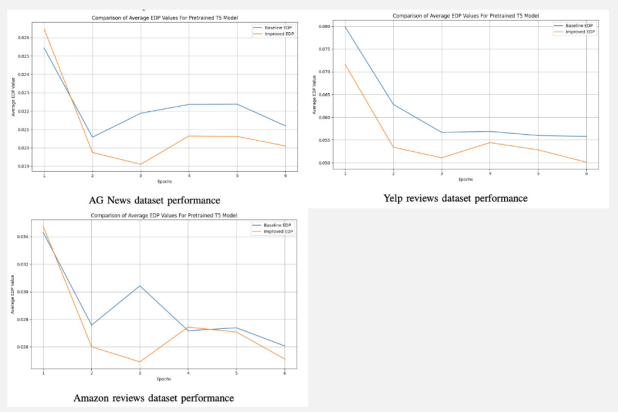
Fig. 4: RoBERTa Model Comparisons

E. T5 Model

For a Pretrained T5 Model, we got the following results. We saw an overall improvement in execution times from 5.5% to 8%.

		T5 Model		
		Baseline EDP	Improved EDP	Percentage Improvement
ag_news	Mean	0.0229694	0.021113	8.04
	Total	0.1378184	0.1266782	
amazon_polarity	Mean	0.0288231	0.0272174	5.57
	Total	0.1729407	0.1653044	
yelp_review_full	Mean	0.0606906	0.055309	8.9
	Total	0.364143	0.335066	

(a) T5 Model Results against different datasets



(b) Performance improvements on various datasets using the T5 model

Fig. 5: T5 Model Comparisons

F. EDP Improvement comparisons between different models

We have evaluated the performance of five pretrained language models (BERT, Electra, GPT-2, RoBERTa, and T5) across three datasets (ag_news, amazon_polarity, and yelp_review_full). The primary goal was to reduce the energy consumption using model mapping and optimization strategies. Below are the brief summary about the analysis performed for these LLM models:

1) BERT Model:

- The BERT model demonstrated notable overall improvements in energy consumption, ranging from 3.5 to 26 percent across the datasets.
- The improved method reduces energy consumption by percentages ranging from approximately 0.96% to 7.11%. Specifically, layers 6 and 7 show the highest improvements at around 5.53% and 7.11%, respectively
- This suggests that the optimization techniques employed were effective in enhancing the model's efficiency without impacting performance.
- Further improvements could potentially be achieved through quantization, reducing the precision of model parameters to decrease memory usage and computational requirements.
- Pruning less important connections within the model could further streamline its execution.

2) Electra Model:

- Electra model also exhibited overall energy improvements in the range of 3.5 to 15 percent.
- The improved method consistently reduces energy improvements, with percentage improvements ranging from approximately 9.27% to 20.54%. Specifically, layers 1, 6, and 7 exhibit improvements of around 19.78%, 20.54%, and 18.50%, respectively.
- This consistency across models indicates that the optimization strategies might be generalizable to different transformer-based architectures.
- Given the model's size, quantization could be a promising avenue for further optimization, potentially leading to even faster inference and reduced energy consumption.
- Pruning techniques could be explored to eliminate redundant parameters and further enhance efficiency.

3) GPT-2 Model:

- The GPT-2 model showed overall energy improvements between 3.29 to 13 percent.
- The most significant improvement is observed in layer 10, with a percentage decrease of approximately 17.70%. Other layers show more modest improvements, ranging from 0.23% to 2.69%
- This further reinforces the effectiveness of the optimization techniques in accelerating the inference process for various language models.
- Similar to BERT and Electra, quantization and pruning need to be investigated to potentially unlock additional performance gains and energy savings.

4) RoBERTa Model:

- The RoBERTa model exhibited the most significant variation in results.
- The most significant improvement is seen in layer 8, with a reduction of approximately 19.29%. Other layers also exhibit notable enhancements, with improvements ranging from 7.16% to 17.43%
- While the RoBERTa model did not achieve the same level of improvement as the other models, it still managed to reduce execution times by up to 13 percent against one of the dataset.
- It also showed negative improvements (increased execution times) compared to other models.
- This suggests that there might be model-specific factors influencing the degree of optimization achievable.
- Quantization and pruning need to be explored further as potential ways to improve it further.

5) T5 Model:

- The T5 model showed overall improvements ranging from 5.5 to 9 percent.
- Similar to BERT and Electra, quantization and pruning need to be investigated further to potentially unlock additional performance gains and energy savings.
- Layer 3 showed the highest improvement at approximately 12.70%, followed by layers 4 and 5 with improvements of around 7.69% and 7.83%, respectively. However, layer 1 experienced a slight performance decline of about 4.08%.

V. Conclusion and future work

A. Conclusion

This project demonstrates the ways to optimize large language models to achieve faster execution times and reduced energy. The results indicate that the effectiveness of optimization strategies can vary depending on the model architecture and the dataset used. Further investigation is needed to understand the factors influencing these variations and to develop more tailored optimization techniques. Additionally, exploring quantization and pruning techniques could provide further performance improvements and energy savings for these models, making them more efficient and useful for various applications.

B. Future work

- **Mapping Strategies**: Explore different mapping strategies (core affinity, thread-level parallelism, data parallelism) to identify the impact on the EDP.
- **Optimization Techniques**: Investigate techniques like quantization or model pruning to further reduce energy consumption and latency.
- **Different LLM models**: Investigate about more LLM models and leverage them to improve efficiency of the solution. Dataset:
- **Large datasets**: Perform the traning against large and different variety of datasets.
- **Hardware Platform**: Select GPU based system to compare the performance of the model.

References

[1] Lava Bhargava, Manjari Gupta, and S Indu. Mapping techniques in multicore processors: current and future trends. 2021.

[2] Erika Hernández-Rubio, Isidoro Gitler, Amilcar Meneses Viveros, and Mireya Paredes-López. Energy consumption model in multicore architectures with variable frequency. 2021.

[3] Erika Hernández-Rubio, Isidoro Gitler, Amilcar Meneses Viveros, and Mireya Paredes-López. Energy consumption model in multicore architectures with variable frequency. 2021.

[4] Shi Qiub Humza Naveeda, Asad Ullah Khana. A comprehensive overview of large language models. 2024.

[5] Selvakumar Jayakumar and Gomatheeshwari Balasekaran. Appropriate allocation of workloads on performance asymmetric multicore architectures via deep learning algorithms. 2020.

[6] Wolfgang J. Paul, Petro Lutsyk, and Jonas Oberhauser. Multi core processors. 2020.

[7] Yuhang Yao Shanshan Han, Qifan Zhang. Llm multi-agent systems: Challenges and open problems. 2024.

[8] Aniket Shivam, Vijayalakshmi Saravanan, and Sudeep Chauhan. Reducing power dissipation in multi-core processors using effective core switching. *International Journal of Computer and Information Technology (IJCIT)*, 3(6):1234–1240, November 2014.

[9] Young Kim Yonghee Yun, Eun Ju Hwang. Energy-efficient design time task mapping algorithm for mpsoc. 2015.

[10] Zhimin Gu Jizan Zhang Zhihua Gan, Mingquan Zhang. Minimizing energy consumption for embedded multicore systems using cache configuration and task mapping. 2016.

[1]–[10].