GPU Programming Final Project
Optimization of the Llama2 ONNX model

Lucas Degeorge, Tristan Kirscher, Corentin Pla

May 27, 2024

Abstract

This report describes the optimization of Llama2[1] LLM for inference. The optimization process involved exporting the model to the ONNX[2] format and applying optimizations using ONNX optimizer[3]. We detail the methodology employed, the chosen optimizations (e.g., graph optimizations), and the performance improvements achieved through our optimization process.

Project Code

The code for our project is publicly available on GitHub : `https://github.com/lucasdegeorge/Prog_GPU`.

1 Introduction

LLMs like Llama2 hold immense potential for NLP tasks. However, their demanding computational needs can limit their deployment on resource-constrained devices. Optimizing these models for efficient execution is crucial for broader adoption and real-world applications.

This report explores optimizing the Llama2 model for faster inference. We utilize the ONNX format, a vendor-neutral standard for representing deep learning models, to facilitate optimization across different hardware platforms. The ONNX optimizer offers various optimization passes that can reduce model size and accelerate inference.

Our goal is twofold:

- **Faster inference**: Achieve quicker processing times for tasks like text generation and question answering, leading to improved responsiveness and suitability for real-time applications.
- **Enhanced resource efficiency**: Optimize the model size, enabling deployment on devices with limited memory capabilities.

2 Methodology

Our optimization approach leverages the ONNX format and its associated optimizer. The methodology can be summarized in the following steps:

1. **Exporting Llama2 to ONNX** : This process is achieved through the Python script that traverses the Llama2 model architecture and translates its components into the ONNX intermediate representation. The resulting ONNX model can then be utilized by the ONNX optimizer for further optimizations.
2. **Optimizing the ONNX Model**: The ONNX optimizer offers various optimization passes designed to improve model efficiency. We will focus on relevant passes, such as potential fusion of operations (e.g., convolution, bias addition, ReLU) into a single step. This fusion can potentially reduce computation and memory access overhead, leading to performance improvements.

3. **Benchmarking**: To evaluate the effectiveness of optimization, we will benchmark both the original and optimized models. This will involve running representative inference tasks (e.g., text generation) and measuring execution times. The comparison of execution times will reveal the speedup achieved through optimization.

3 Computation Graph and Optimizations

Figure 1: Original Llama2 Model

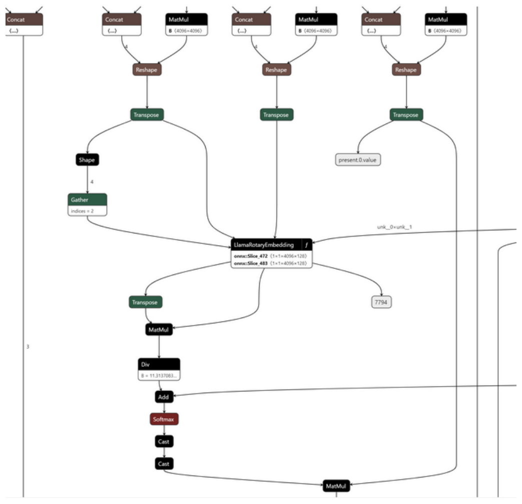

Figure 2: Optimized Llama2 Model

Figure 1 shows the computation graph of the original Llama2 model. In contrast, Figure 2 depicts the optimized model's graph after applying optimizations. We can observe several fused operations and eliminations in the optimized graph. These simplifications are expected to reduce computation overhead and potentially improve inference speed.

The following figures show specific optimizations applied to the ONNX graph, such as Identity elimination in Figure 3.

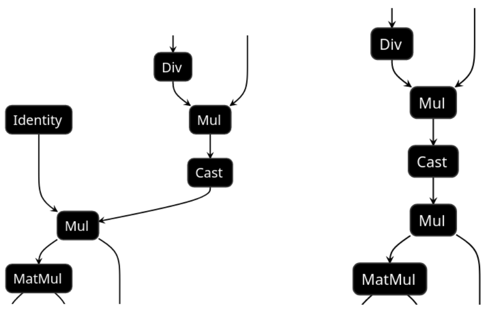

Figure 3: Eliminate Identity Pass

4 Benchmarking

Setup

- **Operation fusions**:

- 'Eliminate identity'
- 'Fuse transpose into GEMM': This pass aims to fuse the transpose operations directly into the GEMM operation.
- 'Fuse matmul add bias into GEMM': fuses matrix multiplication and bias addition into a single GEMM operation.

- **Hardware of Configuration A**:
  - GPU: Nvidia RTX 3090 24Go
  - CPU: Intel Xeon W2235 3.8GHz
  - Memory: 64Go
- **Hardware of Configuration B**:
  - GPU: None
  - CPU: Intel Core i7-1165G7 2.8GHz
  - Memory: 16Go
- **Size of the models**:
  - Hidden dimension = 512 and Number of attention head = 4
  - Hidden dimension = 1024 and Number of attention head = 8
  - Hidden dimension = 2048 and Number of attention head = 16
- **Benchmarking Task**: We compute the inference times of Llama 2 with and without the application of operation fusions (passes). The size of the models varies too.
- **Repetition**: Benchmarks were repeated 100 times to account for potential variations.

Results

The following tables summarize the execution times measured for both the original and optimized Llama2 models during the benchmarking process. The size of the model is Hidden dimension = 2048 and Number of attention heads = 16:

Model	Average Execution Time (s)	Speedup
Original	0.0454	-
'Eliminate identity'	0.0447	1.4%
'Fuse transpose into GEMM'	0.0434	4.4%
'Fuse matmul add bias into GEMM'	0.0444	2.2%

Table 1: Benchmarking Results with Configuration A: Execution Time and Speedup

Model	Average Execution Time (s)	Speedup
Original	5.1001	-
'Eliminate identity'	5.1148	-0.2%
'Fuse transpose into GEMM'	3.9862	21%
'Fuse matmul add bias into GEMM'	3.9630	22%

Table 2: Benchmarking Results with Configuration B: Execution Time and Speedup

These results indicate that the effectiveness of the optimizations varies significantly between passes and hardware configurations. While 'Eliminate identity' shows minimal or negative impact, the 'Fuse transpose into GEMM' and 'Fuse matmul add bias into GEMM' optimizations demonstrate substantial performance improvements, especially under Configuration B (CPU only).

The following figures show the evolution of the execution time of the original and optimized models against the size (hidden size and number of attention heads) of the models in the case of Configuration A

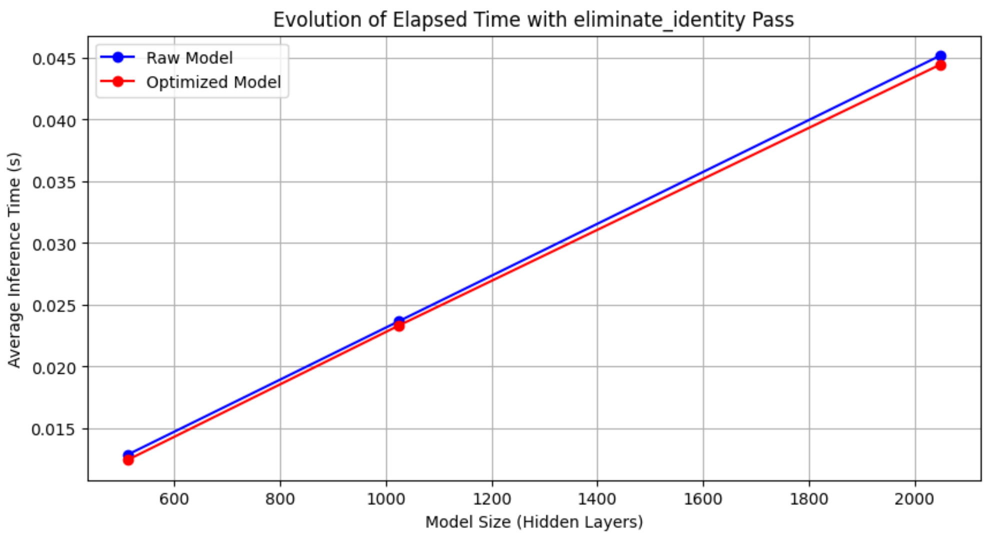

Figure 4: Evolution of the execution time against the size with pass 'eliminate identity' and Configuration A

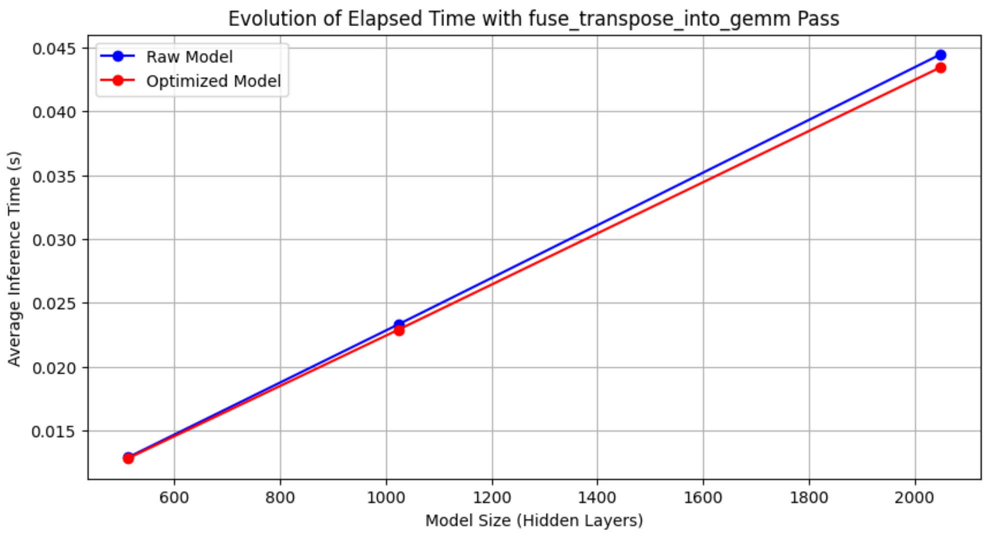

Figure 5: Evolution of the execution time against the size with pass 'fuse transpose into gemm' and Configuration A

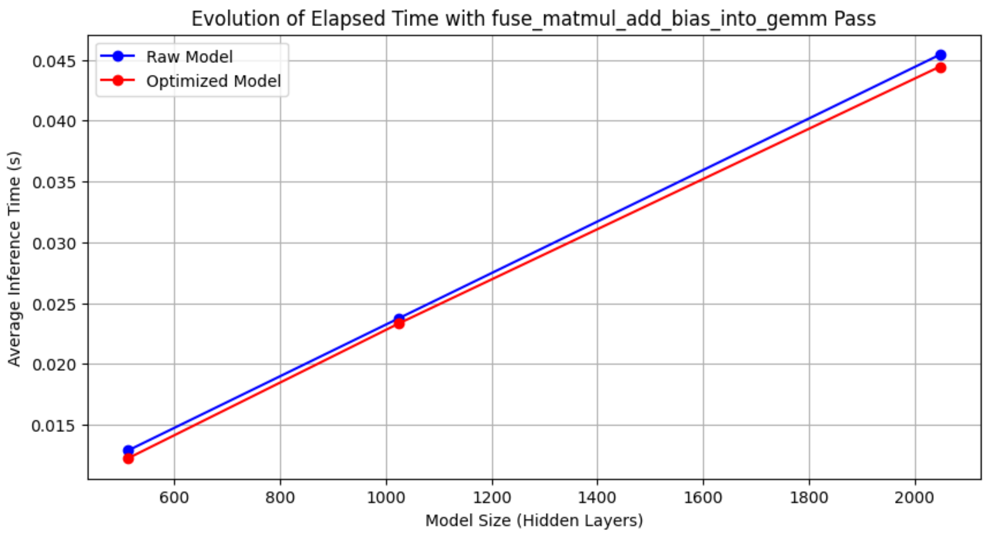

Figure 6: Evolution of the execution time against the size with pass 'fuse matmul add bias into gemm' and Configuration A

The following figures show the evolution of the execution time of the original and optimized models against the size (hidden size and number of attention heads) of the models in the case of Configuration B

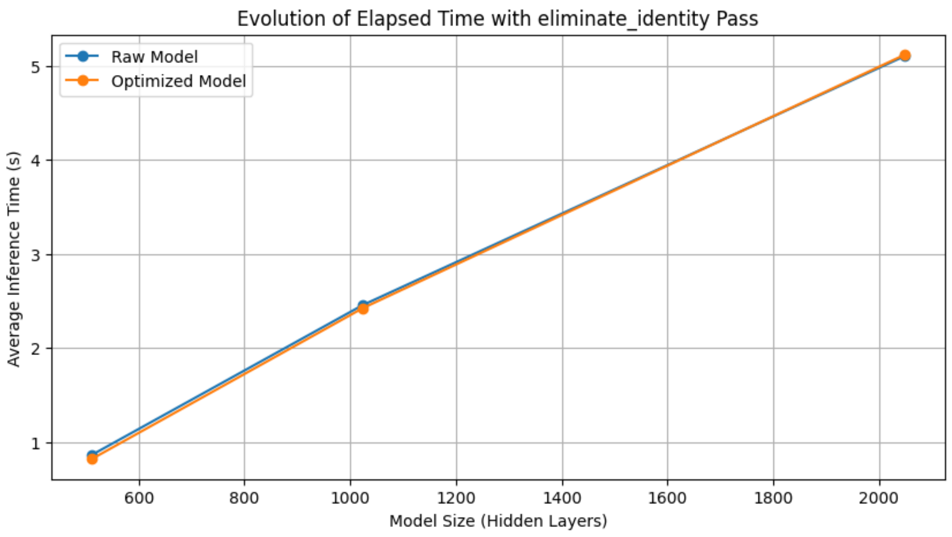

Figure 7: Evolution of the execution time against the size with pass 'eliminate identity' and Configuration B

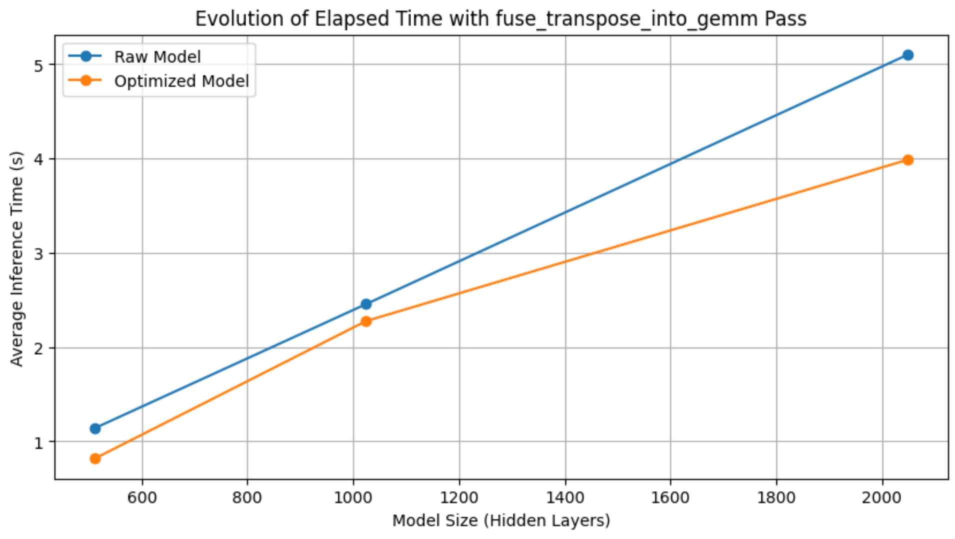

Figure 8: Evolution of the execution time against the size with pass 'fuse transpose into gemm' and Configuration B

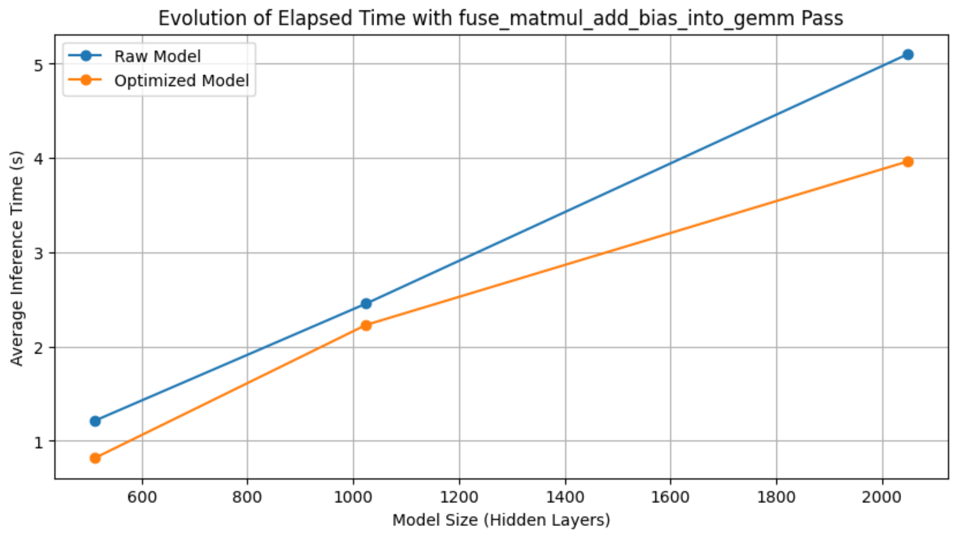

Figure 9: Evolution of the execution time against the size with pass 'fuse matmul add bias into gemm' and Configuration B

5 Conclusion and Discussion

This project successfully explored the optimization of a Llama-2 model. The key achievements include:

- **Exporting the model to ONNX:** This standardized format allows for broader platform compatibility and optimization opportunities.
- **Optimizing the computation graph:** Fused operations and eliminations have improved the model efficiency.
- **Benchmarking the results:** The results show the effectiveness of the operation fusions studies in this work, with the most substantial gains observed for the 'Fuse transpose into GEMM' and 'Fuse matmul add bias into GEMM' passes.

While these steps demonstrate progress, further exploration could be done to fully optimize the model. Here are key areas for future investigation:

- **Quantization:** Implementing quantization techniques can significantly reduce model size and potentially improve inference speed. However, a thorough evaluation of the accuracy-performance trade-off is crucial.
- **Additional optimization techniques:** Exploring other optimization strategies like pruning or knowledge distillation could lead to further size and performance improvements.
- **Performance trade-offs:** Optimization techniques introduce slight accuracy reductions. It is crucial to evaluate the accuracy-performance trade-off for specific use cases and find the optimal balance.

References

[1] "Llama 2: Open foundation and fine-tuned chat models." `https://llama.meta.com/llama2`, 2023.

[2] J. Jaegle, A. Brockman, A. Church, and et al., "Onnx: Open neural network exchange." `https://onnx.ai`, 2021.

[3] "Onnx optimizer: Actively maintained onnx optimizer." `https://github.com/onnx/optimizer`, 2024.