Transformations for Energy Efficient Accelerated Chain Matrix Multiplication

Maxim Moraru[1], Mina Warnet[1], Julien Loiseau[2], Vinay Ramakrishnaiah[2], Nirmal Prajapati[2], Hyun Lim[2], Sumathi Lakshmiranganatha[2], Jamal Mohd-Yusof[2], Karen Tsai[2], Richard Berger[2], Patrick McCormick[2]
[1]*University of Reims Champagne-Ardenne*
Reims, France
[2]*Los Alamos National Laboratory*
Los Alamos, USA, France

***Abstract*—GPU matrix chain multiplication serves as a basis for a wide range of scientific domains like computer graphics, physics, and machine learning. While its time performance was studied for years, there has been significantly less effort in optimizing its energy efficiency.**

GPU power consumption is heavily impacted by the number of data transfers performed. In fact, a data transfer from global memory needs a thousand times more energy than a double precision arithmetic operation. Thus, minimizing data transfers is key for reducing the energy consumption.

We present an energy efficient solution for Matrix Chain Multiplication on GPUs that minimizes computation as well as off-chip data transfers. For this, optimizations at three different levels are provided.

For a single matrix multiplication, we use a blocking strategy that allows us to achieve the minimum number of global memory loads for a given amount of shared memory. We extend our approach to three matrices to decrease the data transfers even further. Finally, we use a parenthesizing algorithm that minimizes the number of computations as well as memory transfers for a whole sequence of matrices.

***Index Terms*—Matrix Chain Multiplication, Energy Efficiency, GPU computing**

I. Introduction

Many existing GPU libraries, like cuBLAS or other hand-tuned implementations [2], [6], [8], focus on single matrix multiplication. A few also allow Matrix Chain multiplication but mostly focus on runtime performance. In comparison, significantly less effort has been dedicated to optimizing for energy efficiency. Reducing the energetic cost of these types of computation is an important challenge. The information technology sector accounts for 2% of greenhouse gas emissions, with data centers accounting for the majority of these [5]. A single data-center of 800 ranks can pay up to 4 million dollars per year for power consumption [1] . In this context, even a small amount of energy saving can have a significant impact.

GPU power consumption is heavily impacted by the number of data transfers performed [4]. In fact, a data transfer from global memory needs one thousand times more energy than a double precision arithmetic operation. Thus, minimizing data transfers is key to reducing energy consumption. Our work aims to solve this problem by making the following contributions:

- Study and adapt the theoretical part described in [7] on how to minimize the number of data transfers for a GPU matrix multiplication.
- Propose an energy efficient implementation of a single matrix multiplication GPU kernel.
- Propose an implementation for a GPU kernel that multiplies three matrices at a time.
- Extend our approach to a sequence of matrices.
- Study the impact of varying the tile size on the number of data transfers and power consumption on a GPU.

This work was sponsered by the US DOE Advanced Simulation and Computing program (ASC) and Center for Non-Linear Studies (CNLS) and the Los Alamos National Laboratory.

II. Assumptions

We make the following assumptions while solving these problems:

- The matrices are dense and rectangular.
- There are two-levels of memory: on-chip/fast and off-chip/slow.
- Only the matrices necessary for one matrix multiplication can fit into the GPU global memory.
- The matrix sizes are larger than the on-chip memory capacity.

III. Single Matrix Multiplication

Given a matrix A_1 of size $P_0 * P_1$ and a matrix A_2 of size $P_1 * P_2$, we implement a blocking strategy for single matrix multiplication. Here, the tiles from matrices A_1 and A_2 are loaded into shared memory and the resulting tile is stored in registers.

Using the notations depicted in Figure 1, we can derive the total number of global loads, h_R, needed to compute the result matrix, R.

$$h_R = P_0 P_1 P_2 \Big(\frac{x+y}{xy}\Big) \tag{1}$$

In order to reduce off-chip data transfers the term $(x+y)/xy$ should be minimized. For a fixed amount of on-chip memory M, we can prove than the minimum number of global loads is attained when $x = y = \sqrt{M}$. Therefore, the minimum number

LA-UR-22-28471

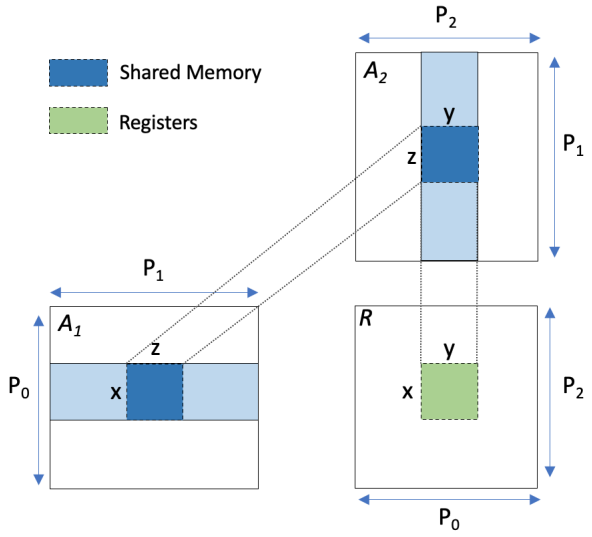

Fig. 1: Single Matrix Multiplication - blocking strategy

of global loads required for multiplying two matrices is given by:

$$h_R^* = \frac{2P_0P_1P_2}{\sqrt{M}} \tag{2}$$

IV. Fused Matrix Multiplication

We extend our approach tothe product of three matrices $R = A_1 * A_2 * A_3$ in order to further decrease the number of data transfers and implement a fused kernel that multiplies 3 matrices at a time. As presented in Figure 2 the tile T is directly multiplied by an entire row of A_3, thereby saving the cost of writing the matrix T into global memory. Consequently, the number of data transfers necessary to compute $A_1 * A_2 * A_3 = R$ is given by:

$$h_{lc} = P_0P_1P_2\frac{x+y}{xy} + P_0P_2P_3\frac{2x+y}{xy} \tag{3}$$

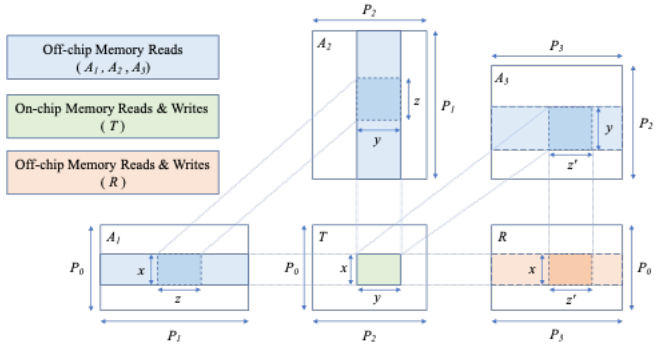

Fig. 2: Fused Matrix Multiplication approach

V. Matrix Chain Multiplication

Finally, we use a parenthesizing algorithm that minimizes the number of memory transfers for a sequence of matrices. While computing a chain of several matrices, there exists an optimal parenthesizing of the chain that gives us the minimal number of operations to compute, depending on the size of the matrices. The algorithm Optimal_Count (OP_Count) [3] calculates the optimal parenthesizing of a sequence of matrices. Achieving the minimal number of computations allows us to reduce data transfers because for each computation, we need 2 reads and 1 write to the global memory. Thus, fewer computations lead to fewer data transfers.

OP_Count gives the optimal splitter, k, for a sequence of matrices, $A_{i\ldots j}$, so that $A_{i\ldots k}A_{k+1\ldots j}$ has the minimal number of operations required to compute $A_{i\ldots j}$. From this decomposition of sequences, we can build the OP_Count Tree that represent the optimal computation order for the sequence $A_{1\ldots N}$.

Once we have the optimal parenthesizing to compute our single matrix multiplication, we can extend OP_Count to OP_DM_Fuse Tree, that uses the fused kernels. Starting from the leaves of the OP_Count Tree, each node of OP_DM_Fuse will have to choose a computation strategy: `Left_fuse, Right_fuse, or No_fuse`. To choose the optimal strategy, the data transfer cost is calculated for each strategy with the equation 3 and the one producing the least data transfers is selected.

VI. Conclusion and Future Work

The goal of this work is to reduce energy consumption in matrix chain multiplication. The strategy was to minimize data transfers since this operation has the greatest impact in terms of time and energy consumption. However, we need to balance static and dynamic power consumption on a GPU, so the execution time cannot be neglected when reducing data transfers.

When implementing the Single Matrix Multiplication, the blocking strategy uses tile sizes as large as possible to minimize the number of data transfers to/from the global memory. We can extend this strategy to three matrices and save the cost of writing back the temporary matrix to the global memory and create two kernels minimizing data transfers for three matrix multiplication.

To multiply the entire sequence of matrices, the OP_Count Tree allows us to order the Single Matrix Multiplication to achieve the minimal number of computations, thus reducing the number of data transfers. Later, the OP_DM_Fuse Tree is build using the 3 previously created kernels to minimize data transfers for the entire sequence.

Acknowledgment

Special thanks to David Rich for his assistance with Darwin Cluster. We also would like to thank Scot Halverson, Nyle Usmani, and Roland Tarrazo from Nvidia for providing the Power Capture Analysis Tool.

References

[1] Compression and decompression is bottlenecking datacenters: Computational storage eases the pain. https://www.marketsandmarkets.com/Market-Reports/data-center-rack-market-210971325.html. Accessed: 2022-08-10.

[2] Lung-Sheng Chien. Hand tuned sgemm on gt200 gpu. *Technical Report, Tsing Hua University*, 2010.

[3] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[4] Bill Dally. Challenges for future computing systems. keynote speech at the 10th hipeac, 2015.

[5] Rajendra Kumar, Sunil K. Khatri, and Mario J. Diván. Optimization of power consumption in data centers using machine learning based approaches: a review. *International Journal of Electrical and Computer Engineering*, 12(3):3192–3203, 06 2022. Copyright - Copyright IAES Institute of Advanced Engineering and Science Jun 2022; Last updated - 2022-05-02.

[6] Jakub Kurzak, Stanimire Tomov, and Jack Dongarra. Autotuning gemm kernels for the fermi gpu. *IEEE Transactions on Parallel and Distributed Systems*, 23(11):2045–2057, 2012.

[7] Prajapati Nirmal. Analytical cost metrics: Days of future past. doctoral dissertation. Fort Collins, Colorado, 2019.

[8] Da Yan, Wei Wang, and Xiaowen Chu. Demystifying tensor cores to optimize half-precision matrix multiply. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 634–643. IEEE, 2020.