Transformations for Energy Efficient Accelerated Chain Matrix Multiplication (TEE-ACM 2)

Maxim Moraru, Mina Warnet, Julien Loiseau, Vinay Ramakrishnaiah, Nirmal Prajapati, Hyun Lim, Sumathi Lakshmiranganatha, Jamal Mohd-Yusof, Karen Tsai, Richard Berger, et al.

Transformations for Energy Efficient Accelerated Chain Matrix Multiplication (TEE-ACM²)

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, [2]Los Alamos National Laboratory

Introduction

Matrix Chain Multiplication plays a key role in the training of deep learning models. They also appear in physics, computer graphics, image processing, etc. Matrix Multiplications often cause a bottleneck in terms of performance and energy because of the heavy costs in computations and memory operations. While the runtime performance has been studied for years, significantly less effort has been expended into optimizing energy efficiency.

Most processors weren't built with energy in mind [3], only performance. For a supercomputer, the cost of power is the most significant expense, and even with the most affordable energy, the cost of a MW ends up at $1M. Moreover, the Exascale Computing Project [4] defines a 'capable' exascale system as one that operates in a power envelope of 20–30 MW.

Thus, reducing the energy cost of these types of computations is a major challenge.

Matrix Chain Multiplication

Problem: Given a sequence of matrices $\{A_1, A_2, \dots A_n\}$ with sizes $\{P_0, P_1, \dots P_n\}$, compute $\prod_{k=1}^{n} A_k$

Sizes 10×30, 30×5, 5×60

R = $A_1 A_2 A_3$

$(A_1 A_2) A_3 = 10\times30\times5 + 10\times5\times60$ = 4,500 multiplications

$A_1 (A_2 A_3) = 30\times5\times60 + 10\times30\times60$ = 27,000 multiplications

- Multiplication order can significantly impact the performance of the algorithm
- The Optimal Parenthesization (OP_Count) algorithm in "*Cormen et al. Introduction to Algorithms.*" outputs an order of matrix multiplications that minimizes the total number of operations.

Energy Efficient GPU implementation

- Energy consumption of a GPU can be broken down into 2 major parts:
  - Energy of the operations executing on the GPU
  - Energy of the GPU itself
- Memory operations dominate the executed operations

Our goal : optimize the total energy consumption for Matrix Chain Multiplications

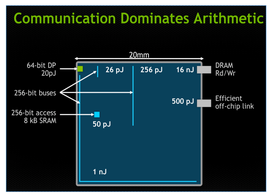

Figure 1: Data transfers vs arithmetic operations (Bill Dally, Challenges of Future Computing Systems, HiPEAC 2015)

Single Matrix Multiplication[1]

Single Matrix Multiplication : Blocking Strategy

Total number of global data transfers (reads + writes) : Minimize !

$$P_1P_2 + P_0P_1P_2 * \frac{(x+y)}{xy}$$

Theorem:
Given the matrix sizes - P_0, P_1, P_2 and shared memory capacity - M, the minimum number of data transfers is given by:

$$\frac{2.P_0.P_1.P_2}{\sqrt{M}} + P_0.P_2$$

Achieved when x = y = $\sqrt{M}$, and z = [1, P1] (*Figure 3*)

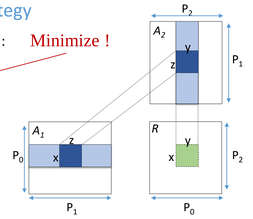

Figure 2: Blocking strategy for single matrix multiplication

Low level TEE-ACM² optimizations

- Double level buffering : use of registers
- Loop unrolling
- Large shared memory tiles

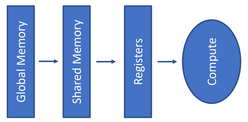

Figure 4 : Double level buffering

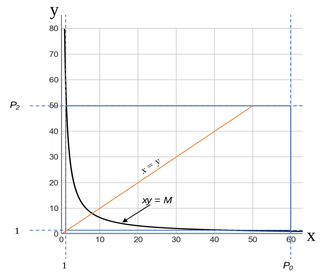

Figure 3 : Minimum number of global loads achieved when x=y

Fused Matrix Multiplication[1]

Extending the approach to three matrices

To compute $(A_1A_2)A_3$, the intermediate result produced, T =$A_1 A_2$ is consumed directly from the on-chip memory to produce values of the final matrix, R = T A_3. This avoids writing of the intermediate matrix, T to off-chip memory.

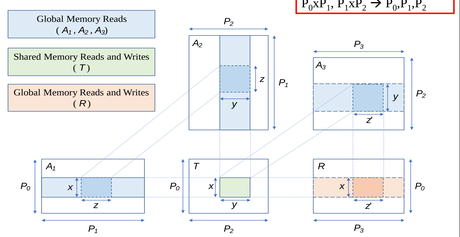

Figure 5: Fused matrix multiplication approach, where three matrices are multiplied in a single GPU kernel.

Optimal Algorithm for Matrix Chain Multiplication

- The OP_Count algorithm produces a tree decomposing a matrix chain, which indicates the order of matrix products to minimize operations
- OP_DM_Fuse is created from OP_Count, which reduces the number of operations and then minimizes the off-chip data transfers by using fusion
- Minimize over the options: left-fuse, right-fuse, and no-fusion

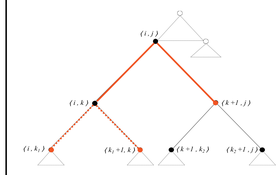

Figure 6: Left fuse strategy

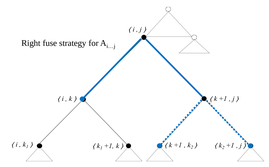

Figure 7: Left fuse strategy

Example for fusion decision algorithm

- Input: 5 matrices with sizes 936, 1008, 552, 368, 1016, 616, 544 and M = 65,536

T-matrix	No Fuse (data transfers)	Fuse left (data transfers)	Fuse right (data transfers)	Final Data Transfers	Fusion decision	Tile sizes (x, y)
T1 : (2,3)	1599696	-	-	1599696	No fuse (A2,A3)	(256,256)
T2 : (1,3)	4683168	-	3072693	3072693	Right fuse (A1 (A2,A3))	(304,208)
T3 : (4,5)	1799336	-	-	1799336	No fuse (A4,A5)	(256,256)
T4 : (4,6)	3116960	2942313	-	2942313	Left fuse ((A4,A5), A6)	(216,296)
T5 : (1,6)	8243798	9364761	9136171	8243798	No fuse (T2,T4)	(256,256)

1. Visit each node of the OP_Count tree
2. Consider options: no fuse, left fuse, right fuse
3. Calculate the minimum number of data transfers for each node
4. Use the optimal over all possible options

Input Tree

Output Tree

- The output tree consists of the minimum number of computations and data transfers for multiplying the given matrix chain

Energy Measurement Accuracy

Tools used for power measurements

Software
- Nvidia Management Library (NVML)

Hardware
- Power Capture Analysis Tool (PCAT)

Hardware vs software power measurements

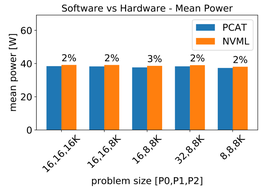

Figure 9 : NVML power precision on an entire run

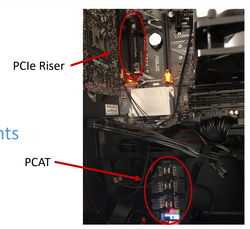

Figure 8 : Installation of PCAT device on GPU for hardware power measurement

- We observed less than 3% difference in power between PCAT and NVML measurements on average

Single Matrix Multiplication Results

Energy efficiency

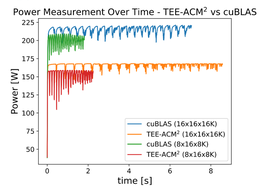

Figure 10 : Power savings of TEE-ACM² over cuBLAS (lower is better)

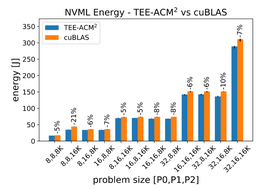

Figure 11 : Energy savings of TEE-ACM² over cuBLAS (lower is better)

- The number of global loads are reduced by 66% compared to cuBLAS. This results in 30% power saving and 8% energy saving on average, and up to 21% savings
- The average execution time overhead of TEE-ACM² is 20%

GPU resources usage

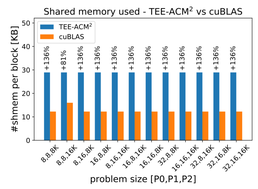

Figure 12 : The amount of shared memory used by TEE-ACM² compared to cuBLAS

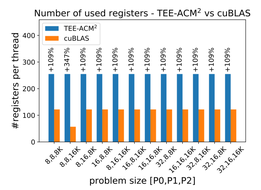

Figure 13 : The number of registers used by TEE-ACM² compared to cuBLAS

- TEE-ACM² uses more than two times on-chip memory than cuBLAS

The impact of varying the tile size

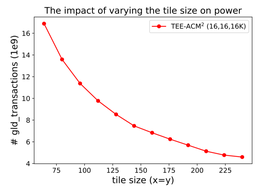

Figure 14 : The number of global loads decreases when the tile size increases

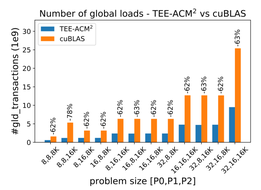

Figure 15 : The number of global loads performed by TEE-ACM² compared to cuBLAS

- By using larger tile sizes, the number of global loads decrease significantly
- cuBLAS performs 3 times more global loads on average than TEE-ACM²

*Present results were obtained on a Darwin node equipped with v100 GPUs and Cascade Lake CPUs (averaged over 10 samples of 10 runs each)

Fused-Kernel Results

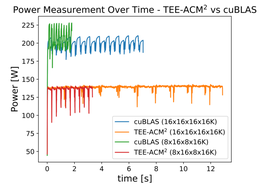

Figure 16 : Power savings of TEE-ACM² over cuBLAS for a fused strategy (lower is better)

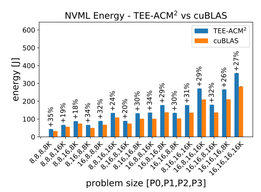

Figure 17 : TEE-ACM² consume more energy than cuBLAS for a fused strategy (lower is better)

- Our implementation of fused-kernel approach uses atomic adds that drastically impact the execution time
- Also, when multiplying three matrices, the GPU resources are distributed across 4 tiles compared to 3 in the case of a single matrix multiplication

*Present results were obtained on a Darwin node equipped with v100 GPUs and Cascade Lake CPUs (averaged over 10 samples of 10 runs each)

Matrix Chain Multiplication Results

OP_Count Tree effect on MCM execution time

Best case : the matrix sizes are increasing → in order computation is most efficient
Worst case : the matrix sizes are decreasing → in order computation is least efficient
Random : the matrix sizes are generated at random between 8k and 24 k

Problem sizes for best and worst:
8,10,12,14,16,18,20k
Problem sizes for random:
10, 20 ,15, 9, 9, 22, 21k

Problem sizes for best and worst:
8 ,9, 10, 11,12, 13, 14, 15, 16, 17, 18, 19, 20k
Problem sizes for random:
20, 16, 17, 10, 16, 19, 9, 24, 16, 16, 12, 8, 10k

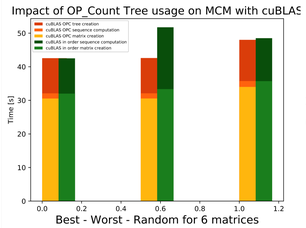

Figure 18 : Execution time of creation and computation for a sequence of 6 matrices

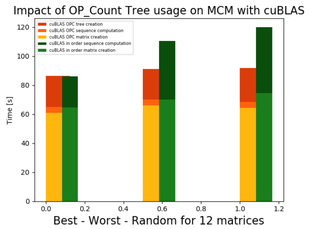

Figure 19 : Execution time of creation and computation for a sequence of 12 matrices

- When computing with the best case, cuBLAS without the tree is 2% faster because of the overhead of building and traversing the tree
- In the worst case, the creation of the matrices is longer without the tree because the parenthesizing isn't optimal, the temporary matrices are bigger, thus the longer execution time

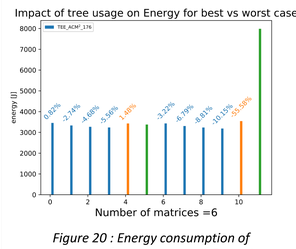

Figure 20 : Energy consumption of computation for a sequence of 6 matrices

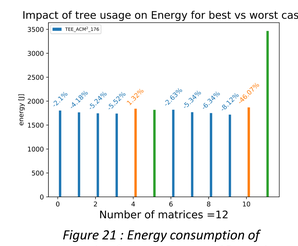

Figure 21 : Energy consumption of computation for a sequence of 12 matrices

- The blue percentages indicate the energy consumption ratio between TEE-ACM² and cuBLAS using OP_Count. By using our optimized kernel, we end up with 5 to 10% energy savings with the optimal tile sizes of 224
- The orange percentages indicate the energy consumption ratio between cuBLAS with and without the tree. In the best case, we consume 1.5% more energy with the tree, but in the worst case, using the tree allows us to save up to 50% energy on average

*Present results were obtained on a Darwin's node equipped with v100 GPUs and Cascade Lake CPUs The plots shows the results from 3 kernels usage per matrix multiplication

Conclusions and Future Work

- Less power consumption may be preferred over shorter execution times for applications such as edge computing. This work focuses on such use cases
- For our fused kernel implementation, the energy consumption was primarily dictated by the execution time (static power)
- Performing fewer global loads and stores significantly decreases the GPU power consumption at the cost of increased execution time
- By using the OP_Count tree we optimize the computation order in a sequence of matrices.
  - Using cuBLAS decreases execution time by a factor of two
  - Using TEE-ACM², we save up to 10% energy
- The hardware counters and APIs supplied by Nvidia provide accurate results with precision comparable to the measurements made using the hardware power capture analysis tool (PCAT)
- Future work: we plan to optimize the fused kernels by removing atomic operations, thus reducing computation time. Once the fused kernels are implemented, we will generate the fusion tree and compare the time and energy between TEE-ACM² with and without fusion

References

[1] Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.
[2] Dally, Bill. "Challenges for future computing systems. Keynote speech at The 10th HiPEAC." (2015).
[3] Kathy Yelick, Exascale computing: opportunities and challenges, NERSC, 2017
[4] Exascale Computing Project, https://insidehpc.com/ecp/
[5] Prajapati, Nirmal. *Analytical Cost Metrics: Days of Future Past.* Doctoral Dissertation. Colorado State University. Fort Collins, Colorado. 2019.

LA-UR-22-28472

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 supplying the Power Capture Analysis Tool*