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], Robert Pavel[2], Sumathi Lakshmiranganatha[2], Jamal Mohd-Yusof[2], Andrew Reisner[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 in optimizing its energy efficiency. Thus, reducing the energy cost of these types of computations is a major challenge.
The study of balancing energy efficiency and execution time at a data center scale could have a positive environmental impact.

Matrix Chain Multiplication

- Problem: Given a sequence of matrices $\{A_1, A_2, \ldots A_n\}$ with sizes $\{P_0, P_1, \ldots P_n\}$, compute $\prod_{k=1}^{n} A_k$
- Multiplication order can significantly impact the performance of the algorithm

Sizes 10x30, 30x5, 5x60

$R = A_1 A_2 A_3$

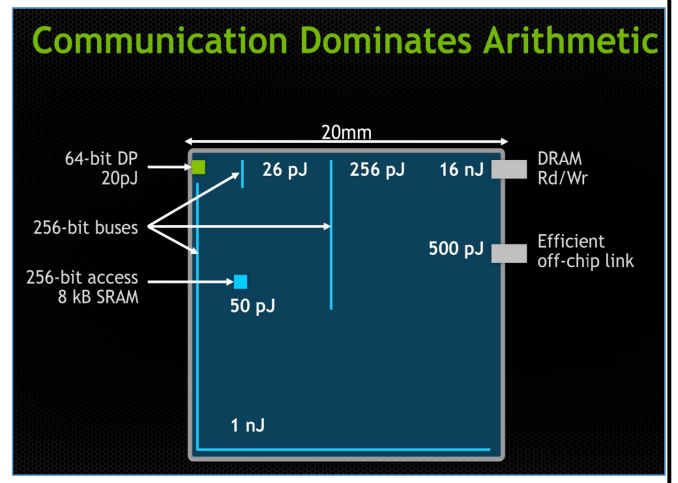
$(A_1 A_2) A_3 = 10{\times}30{\times}5 + 10{\times}5{\times}60 = 4{,}500$ multiplications

$A_1 (A_2 A_3) = 30{\times}5{\times}60 + 10{\times}30{\times}60 = 27{,}000$ multiplications

- 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 GPU itself
  - Energy of the operations executing on the GPU
- Memory operations dominate the executed operations
  (*Bill Dally, Challenges of Future Computing Systems, HiPEAC 2015*)
- Our goal is to optimize the total energy consumption for Matrix Chain Multiplications

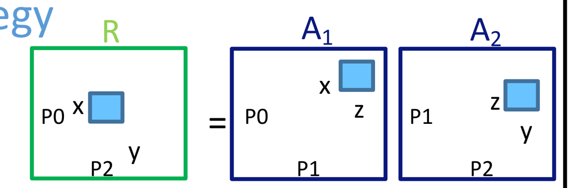


Single Matrix Multiplication[1]

Single Matrix Multiplication : Blocking Strategy

Total number of global data transfers (read + writes) :

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



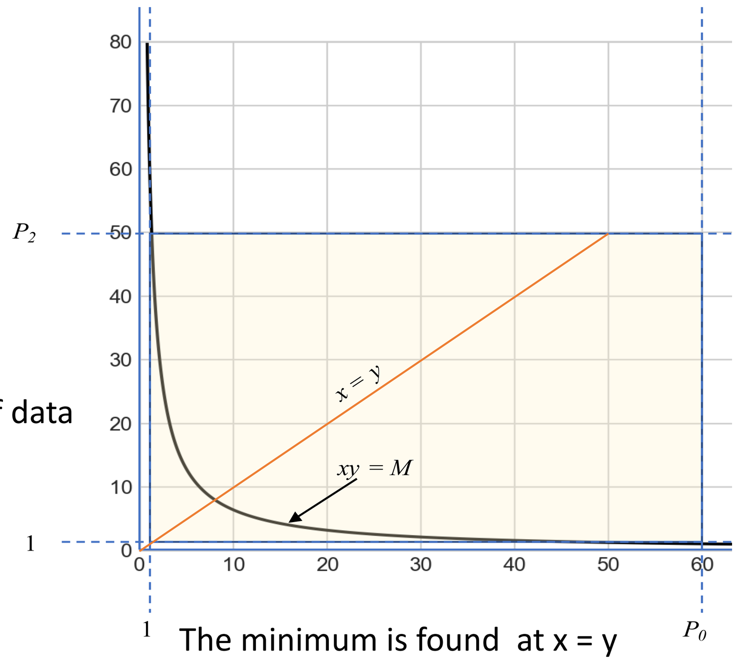
Given the matrix sizes P_0, P_1, P_2 and the on-chip memory capacity M,

solve for x, y, z that *minimize* the number of off-chip data transfers.

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

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

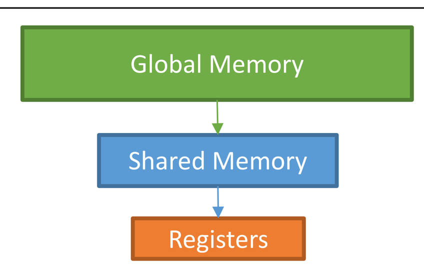
where $x = y = \sqrt{M}$, and $z = [1, P1]$



The minimum is found at x = y

Low level TEE-ACM² optimizations

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

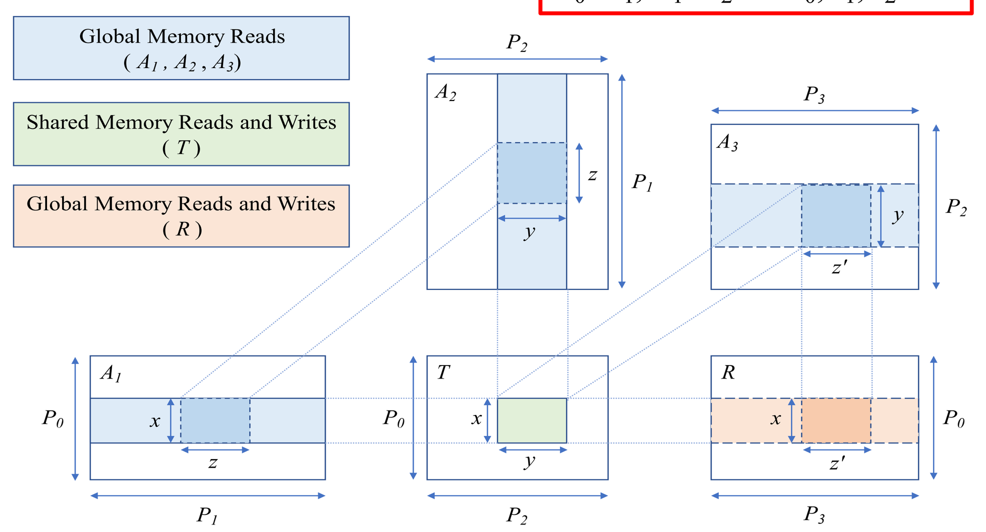


Fused Matrix Multiplication[1]

Blocking Strategy

Notation:
$P_0 x P_1$, $P_1 x P_2$ → P_0,P_1,P_2

- Extending the approach to 3 matrices :
- To compute (A_1A_2) A_3, the intermediate results produced in matrix, $T = A_1 A_2$ are consumed directly from on-chip memory to produce values of the final matrix, $R = T A_3$. This avoids writing of Intermediate matrix T to Off-chip memory.

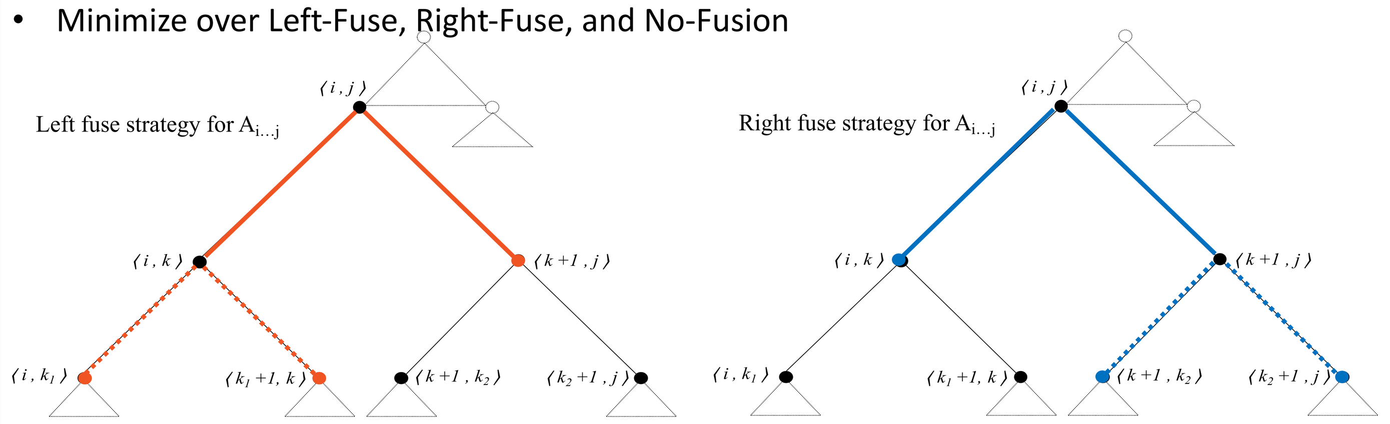


Closed form solutions

	Off-chip Data Transfers	x^*	y^*
Single Matrix Multiplication	$S_{MM}^* = \frac{2P_0P_1P_2}{\sqrt{M}} + P_0P_2$	$\sqrt{M}$	$\sqrt{M}$
Left Fused Multiplication of 3 Matrices	$F_l^* = \frac{2P_0P_1P_2(1+\alpha)\sqrt{\alpha'}}{\sqrt{M}} - P_0P_2$	$\sqrt{M/\alpha'}$	$\sqrt{M\alpha'}$
Right Fused Multiplication of 3 Matrices	$F_r^* = \frac{2P_1P_2P_3\,(1+\beta)\sqrt{\beta'}}{\sqrt{M}} - P_1P_3$	$\sqrt{M\beta'}$	$\sqrt{M/\beta'}$

Optimal Algorithm for Matrix Chain Multiplication

- The OP_Count algorithm produces a tree decomposing a Matrix Chain, which indicates the order to compute the matrix products to minimize operations
- OP_DM_Fuse is created from OP_Count, that reduces the number of operations and then minimizes the off-chip data transfers by using fusion
- Minimize over Left-Fuse, Right-Fuse, and No-Fusion



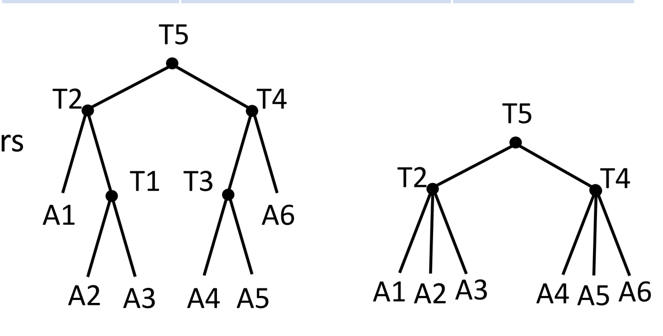
Example for fusion decision algorithm

- 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)

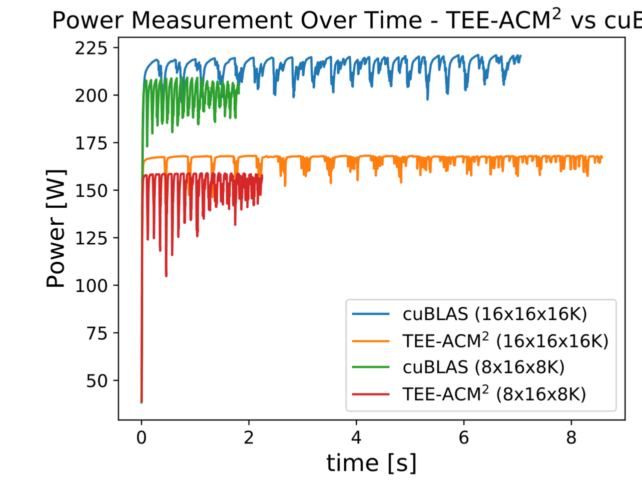
1. Visit each node of the OP_Count tree
2. Consider options: no fuse, left fuse, right fuse
3. Calculate the minimum number of data transfers for each node
4. Use the optimal over all possible options

- The output tree consists of the minimum number of computations and data transfers, using matrix fused multiplication

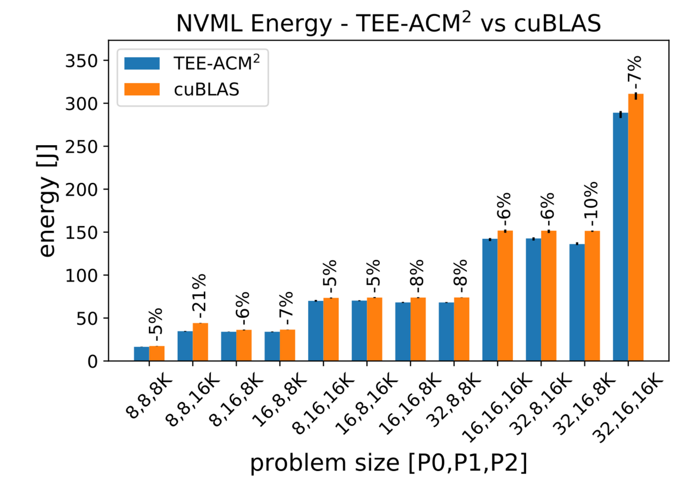


Input Tree   Output Tree

Single Matrix Multiplication Results



Power savings of TEE-ACM² over cuBLAS lower is better

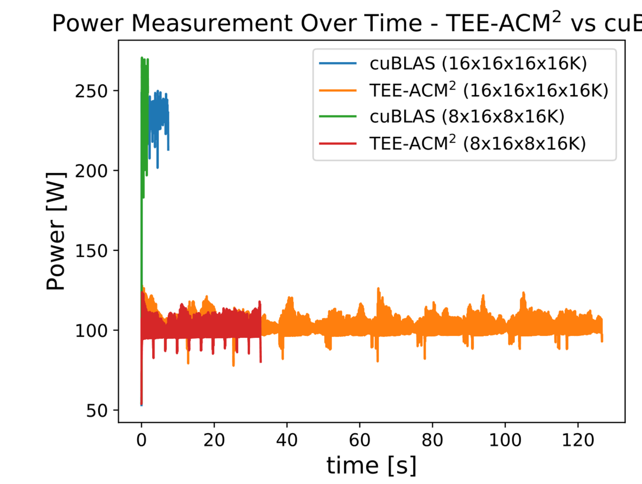


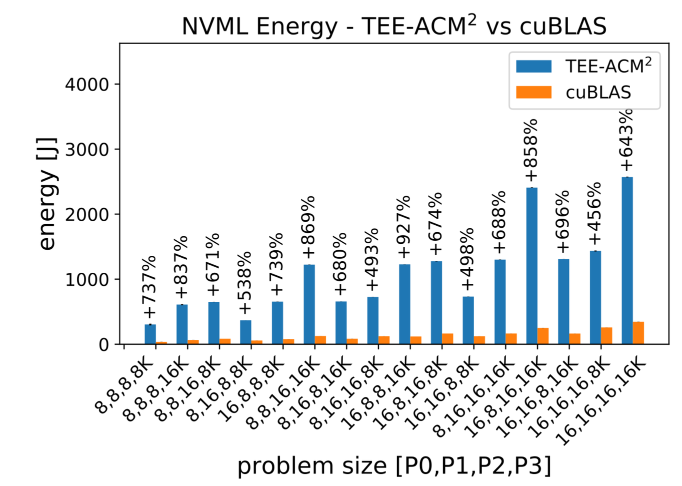
Energy savings of TEE-ACM² over cuBLAS lower is better

- 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%

*Present results were obtained on a Darwin's node equipped with v100 GPUs and Cascade Lake CPUs (average of 10 runs)

Fused-Kernel Results

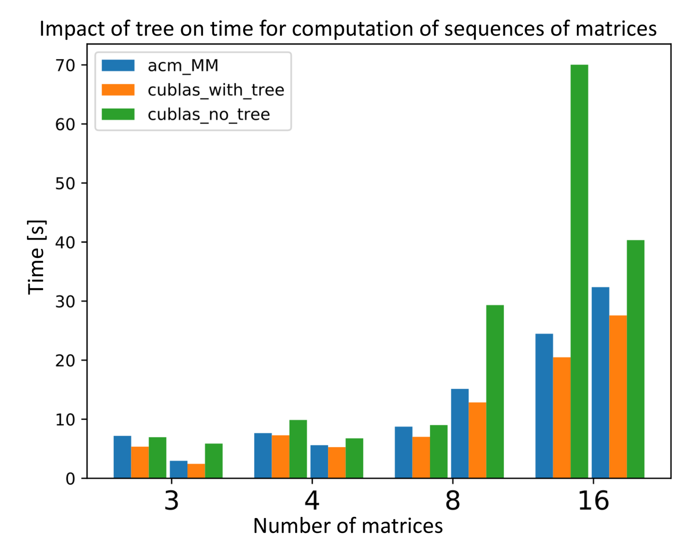


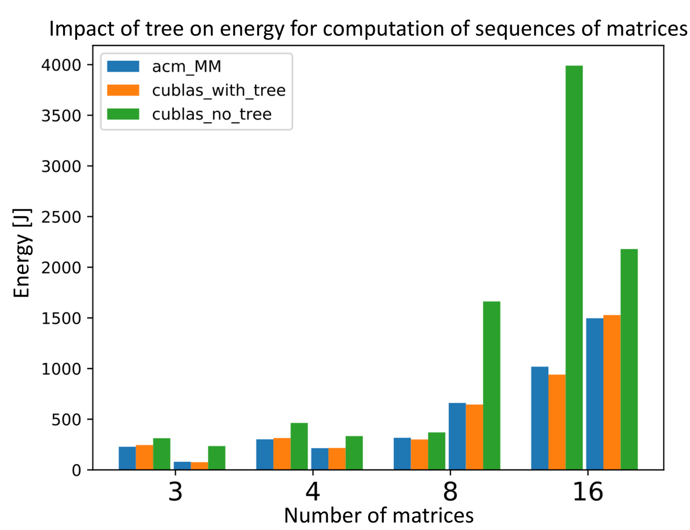


- Our implementation of fused-kernel approach uses atomic adds that drastically impact the execution time. We spend ~80% of the time performing atomic adds in shared memory
- Reducing/Eliminating atomic adds will improve execution time

*Present results were obtained on a Darwin's node equipped with v100 GPUs and Sandy Bridge CPUs (average of 10 runs)

Matrix Chain Multiplication Results





- The OP-Count tree implementation with cuBLAS computes the same sequence 35% faster and consumes 43% less energy on average
- TEE-ACM² vs Cublas: TEE-ACM² takes ***5-25% more time*** with ***3% energy savings on average***

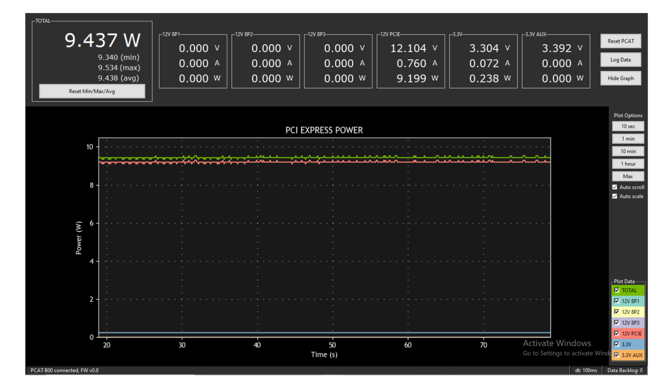
*Present results were obtained on a Darwin's node equipped with v100 GPUs and Cascade Lake CPUs The plots shows the results from 3 kernels usage per matrix multiplication

Energy Measurement Accuracy

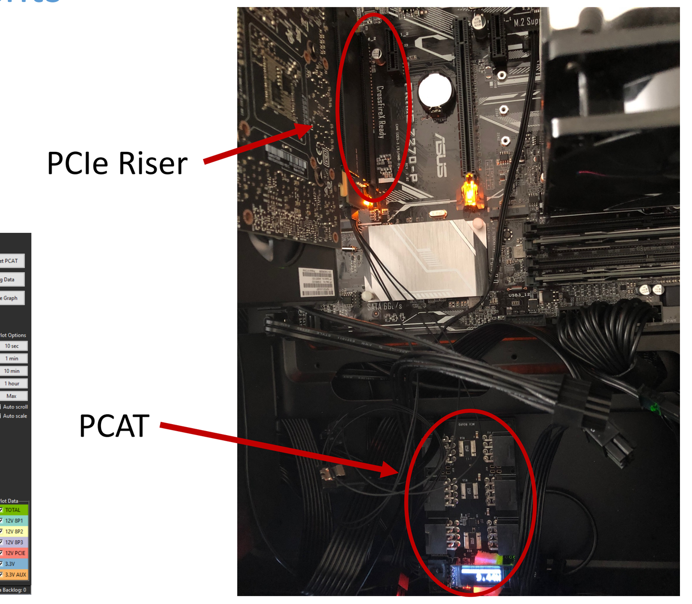
Tools used for power measurements

Software
- Nvidia Management Library (NVML)

Hardware
- Power Capture Analysis Tool (PCAT)

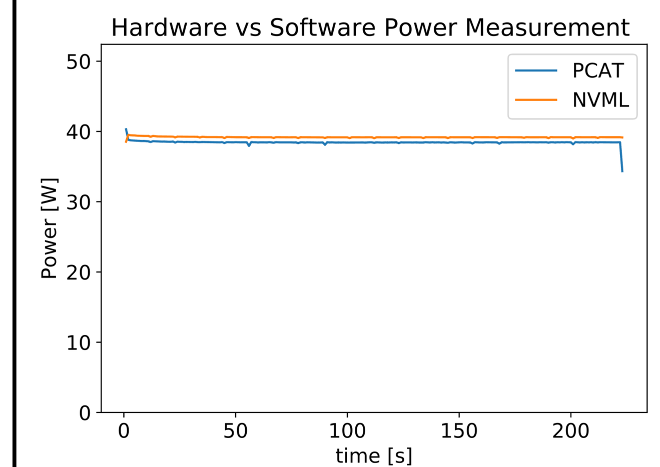


PCAT software interface

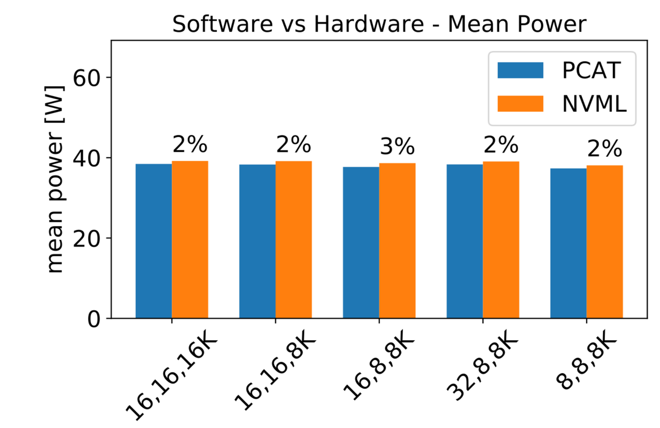


Installation of PCAT device on GPU for hardware power measurement

Comparison of hardware and software power measurements



Power Time Series obtained by PCAT and NVML (16x16x16K problem size)



NVML power precision on an entire run

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

*Present results were obtained on a a Quadro P2000 GPU

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
- 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 three
  - Using TEE-ACM², we save up to 3% energy while taking 25% more time
- 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] Prajapati, Nirmal. *Analytical Cost Metrics: Days of Future Past.* Doctoral Dissertation. Colorado State University. Fort Collins, Colorado. 2019.
[2] Dally, Bill. "Challenges for future computing systems. Keynote speech at The 10th HiPEAC." (2015).

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

LA-UR-22-27515