Numerical Behavior of GPU Matrix Multiply-Accumulate Hardware

Mantas Mikaitis
Department of Mathematics
University of Manchester

`mantas.mikaitis@manchester.ac.uk`

2022 SIAM Conference on Parallel Processing for Scientific Computing

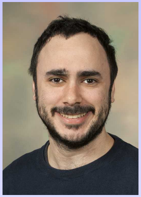
MS29 Understanding and Exploiting Mixed-Precision Accelerators for High-Performance Computing

24th of February, 2022. Virtual.

Collaborators

Max Fasi

Nick Higham



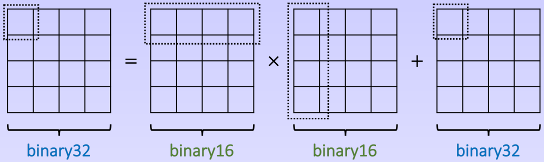
Florent Lopez

Theo Mary

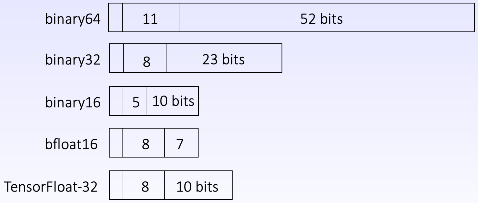
Srikara Pranesh

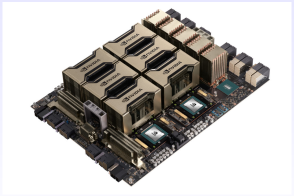
Matrix Multiply-Accumulate Hardware

Matrix multiply-accumulate (**MMA**) on GPUs (**NVIDIA Tensor Cores, AMD Matrix Engines**):



Variety of formats with high throughputs, which accelerated research in **mixed-precision**:





A100 image source: `https://developer.nvidia.com/`.

Low- and Mixed-Precision on GPUs

Many devices now include **MMA** units:

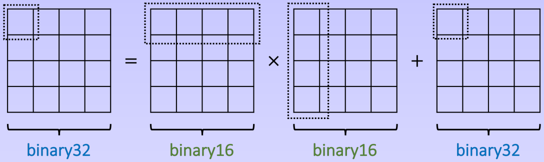
Year	Device	Input formats	Output formats	Throughput (max)
2016	Google TPUv2	bfloat16	binary32	46 Tflop/s
2017	Google TPUv3	bfloat16	binary32	123 Tflop/s
2018	**NVIDIA V100**	binary16	binary32	125 Tflops/s
2018	Graphcore IPU1	binary16	binary32	125 Tflop/s
2020	Graphcore IPU2	binary16	binary32	250 Tflop/s
2020	**NVIDIA A100**	bfloat16, binary16/64, TensorFloat-32	binary32/64	312 Tflop/s
2021	**AMD MI250X**	bfloat16, binary16/32/64	-	383 Tflop/s

127 machines in the November 2021 **TOP500 list** contain NVIDIA devices with **tensor cores**.

How is the table going to look in 2022

Intel's Ponte Vecchio, NVIDIA H100, ... ?

Multiword Arithmetic with Tensor Cores



- What if we can't round inputs to binary16?
- Are we doomed to wasting energy in hundreds of tensor cores?
- Future GPUs may have low binary32/64 performance.
- Use **multiword arithmetic**.
- **Markidis et. al., 2018**, proposed **precision refinement** (thereafter binary16/32 is fp16/32):
  $A_{fp32}B_{fp32} = A_{fp16}B_{fp16} + A_{fp16}R_B + R_A B_{fp16} + R_A R_B$,
  where $R_A = A_{fp32} - A_{fp16}$ and $R_B = B_{fp32} - B_{fp16}$.

Error Analysis

We generalised precision refinement (**Fasi, Higham, Lopez, Mary, Mikaitis, 2022**): given A and B, convert them to **multiword representation** in low precision as

$$A_i = \mathrm{fl}_{\mathrm{low}}\left(A - \sum_{k=1}^{i-1} A_k\right), \quad B_j = \mathrm{fl}_{\mathrm{low}}\left(B - \sum_{k=1}^{j-1} B_k\right).$$

If no exceptions occur in the conversion, we obtain

$$\begin{aligned} A &= \sum_{i=1}^{p} A_i + \Delta A, \quad & |\Delta A| &\le u_{\mathrm{low}}^{p} |A|, \\ B &= \sum_{j=1}^{p} B_j + \Delta B, \quad & |\Delta B| &\le u_{\mathrm{low}}^{p} |B|. \end{aligned}$$

Data represented in multiple words (**unevaluated sum**).

Error Analysis

Then the product $C = AB$ is given by

$$C = \sum_{i=1}^{p} \sum_{j=1}^{p} A_i B_j + A\Delta B + \Delta A B - \Delta A \Delta B.$$

Skipping through to the final error bound, we have

$$\widehat{C} = AB + E, \qquad |E| \leq \left(2u_{\mathrm{low}}^{p} + u_{\mathrm{low}}^{2p}\right)|A||B| + \gamma_{n+p^2-1}^{\mathrm{high}} \sum_{i=1}^{p} \sum_{j=1}^{p} |A_i||B_j|, \tag{1}$$

with $\gamma_{n+p^2-1}^{\mathrm{high}} = \frac{(n+p^2-1)u_{\mathrm{high}}}{1-(n+p^2-1)u_{\mathrm{high}}}$ and $u_{\mathrm{low/high}}$ unit roundoffs.

Error bound

First term in the bound (1) may dominate.

Error Analysis

Choice of split (p)

For fp16, $u_{\text{low}} = 2^{-11}$ and fp32 $u_{\text{high}} = 2^{-24}$. With these $p = 2$ is sufficient for the first term not to dominate the bound since $u_{\text{low}}^2 = 4u_{\text{high}}$.

Next we make a further approximation—**do not compute all products** (p^2): ignore any product $A_i B_j$ such that $i + j > p + 1$.

The modified bound in $\widehat{C} = AB + E$ is

$$|E| \le \left((p+1)u_{\text{low}}^p + \gamma_{n+p^2-1}^{\text{high}}\right)|A||B| + O(u_{\text{high}} u_{\text{low}} + u_{\text{low}}^{p+1}).$$

Due to this, instead of p^2 products, we have $p(p+1)/2$.

Error Analysis

Dominant terms in the error bound for u_{high} corresponding to fp32 and various choices of u_{low} and p:

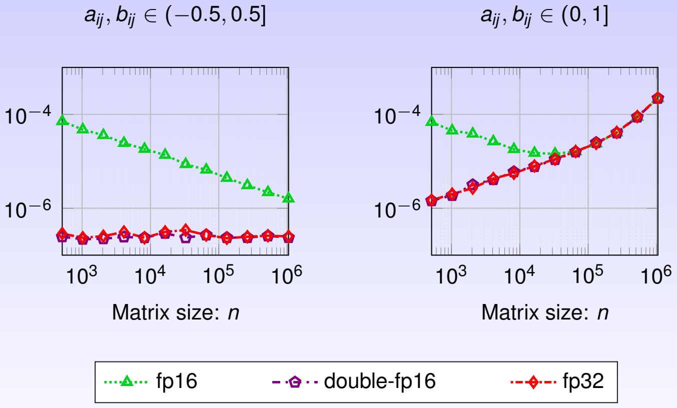
u_{high}	u_{low}	Split	Name	Bound
2^{-24} (fp32)	2^{-11} (fp16)	$p = 1$	fp16	$2 \times 2^{-11} + n \times 2^{-24}$
		$p = 2$	double-fp16	$n \times 2^{-24}$
	2^{-8} (bf16)	$p = 1$	bf16	$2 \times 2^{-8} + n \times 2^{-24}$
		$p = 2$	double-bf16	$3 \times 2^{-16} + n \times 2^{-24}$
		$p = 3$	triple-bf16	$n \times 2^{-24}$

Double-fp16 multiword arithmetic

$AB \approx A_1B_1 + A_1B_2 + A_2B_1$ performed with three tensor core GEMM invocations (note “double” refers to 2×fp16).

Double-fp16 in Emulated Arithmetics

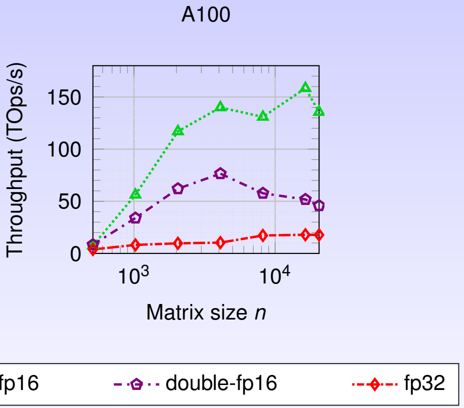
Componentwise relative error of algorithms for computing the product AB:



The matrices are $A \in \mathbb{R}^{512\times n}$ and $B \in \mathbb{R}^{n\times 512}$ (uniform dist.).

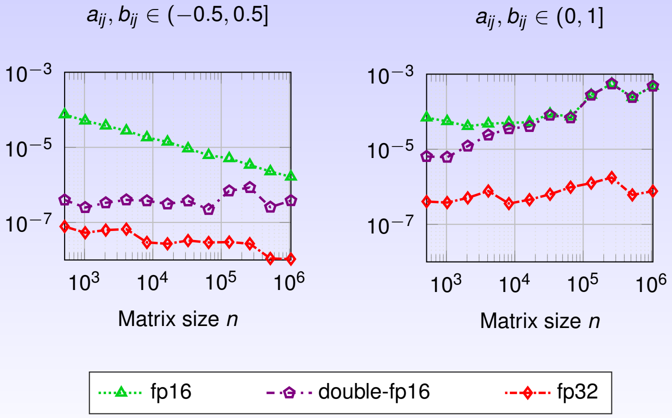
Performance Benchmarking on A100

Throughput of GPU implementations of algorithms for computing the product AB, where $A, B \in \mathbb{R}^{n\times n}$:



Double-fp16 with A100 Tensor Cores

Componentwise relative error of GPU (A100) for computing the product *AB*:



Previous Results from Testing Tensor Cores

In **Fasi, Higham, Mikaitis, Pranesh, 2021** we showed that tensor cores:

- Used **round to zero** instead of **round to nearest**.
- Do not normalize after each addition.
- Compute inner products that can be **nonmonotonic**.

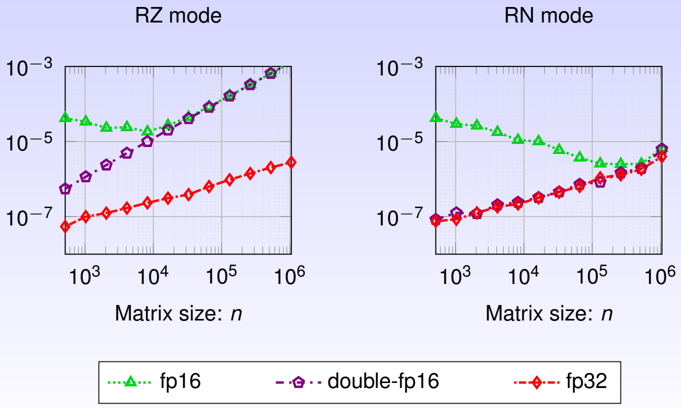
Tensor Core Error Accumulation

Our prediction is that the first property causes errors to accumulate with a factor n rather than the probabilistic bound $\sqrt{n}$ that is common with round to nearest.

Next we **simulated tensor cores** using `CPFloat` (**Fasi and Mikaitis, 2020**) to test this prediction.

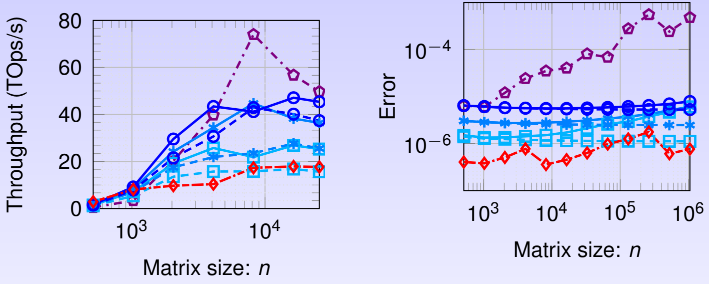
Change of Rounding Mode in Simulated TCs

Componentwise relative error of algorithms for computing the product AB using simulated block FMAs (two rounding modes are used; data interval $(0, 1]$):



Application of FABSum to A100 Tensor Cores

FABSum (**Blanchard, Higham, Mary, 2020**): perform some work in tensor cores and some in standard fp32/64.



double-fp16, cuBLAS
double-fp16, FABsum-v2 ($b = 128$)
double-fp16, FABsum-v2 ($b = 256$)
double-fp16, FABsum-v2 ($b = 512$)
double-fp16, FABsum-v1 ($b = 128$)
double-fp16, FABsum-v1 ($b = 256$)
double-fp16, FABsum-v1 ($b = 512$)
fp32, cublasGemmEx

Summary

The main goal of this work is to **enable high-precision computations with low precision tensor cores**.

- We have presented a generalization of **Markidis et al, 2018** precision refinement.
- Error analysis of **multiword arithmetic** through number of splits p.
- Identified good p and that not all p^2 needed.
- Identified **rounding mode effects in tensor cores**.
- Applied `FABSum` to tensor cores.

Next steps: apply analysis to **AMD MI250X GPUs** (upcoming **Frontier exascale supercomputer**).

References I

S. Markidis, S. W. Chien, E. Laure, I. B. Peng, J. S. Vetter.
NVIDIA Tensor Core Programmability, Performance & Precision.
IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). 2018.

M. Fasi, N. J. Higham, M. Mikaitis, and S. Pranesh.
Numerical Behavior of NVIDIA Tensor Cores.
PeerJ Comput. Sci. 7:e330 (2021).

P. Blanchard, N. J. Higham and T. Mary.
A Class of Fast and Accurate Summation Algorithms.
SIAM J. Sci. Comput. 42. 2020.

References II

M. Fasi, N. J. Higham, F. Lopez, T. Mary, and M. Mikaitis.
Matrix Multiplication in Multiword Arithmetic: Error Analysis and Application to GPU Tensor Cores.
MIMS EPrint 2022.3. 2022.

N. J. Higham and S. Pranesh.
Simulating Low Precision Floating-Point Arithmetic.
SIAM J. Sci. Comput. 41. 2019.

M. Fasi and M. Mikaitis.
CPFloat: A C Library for Emulating Low-Precision Arithmetic.
MIMS EPrint 2020.22. 2020.