Mixed-Precision S/DGEMM Using the TF32 and TF64 Frameworks on Low-Precision AI Tensor Cores

Pedro Valero-Lara, Frank Liu, and Jeffrey S. Vetter
Oak Ridge National Laboratory
Oak Ridge, USA
{valerolarap},{liufy},{vetter},@ornl.gov

Ian Jorquera
Colorado State University
Fort Collins, USA
jorquera@colostate.edu

ABSTRACT

Using NVIDIA graphics processing units (GPUs) equipped with Tensor Cores has enabled the significant acceleration of general matrix multiplication (GEMM) for applications in machine learning (ML) and artificial intelligence (AI) and in high-performance computing (HPC) generally. The use of such power-efficient, specialized accelerators can provide a performance increase between 8× and 20×, albeit with a loss in precision. However, a high level of precision is required in many large scientific and HPC applications, and computing in single or double precision is still necessary for many of these applications to maintain accuracy. Fortunately, mixed-precision methods can be employed to maintain a higher level of numerical precision while also taking advantage of the performance increases from computing with lower-precision AI cores. With this in mind, we extend the state of the art by using NVIDIA's new TF32 framework. This new framework not only burdens some constraints of the previous frameworks, such as costly 32 16-bit castings but also provides an equivalent precision and performance by using a much simpler approach. We also propose a new framework called TF64 that attempts double-precision arithmetic with low-precision Tensor Cores. Although this framework does not exist yet, we validated the correctness of this idea and achieved an equivalent of 64-bit precision on 32-bit hardware.

CCS CONCEPTS

• **Hardware** → *Hardware test*; **Analysis and design of emerging devices and systems**; • **Mathematics of computing** → *Numerical analysis.*

KEYWORDS

Mixed Precision, Tensor Core, GEMM, GPUs

ACM Reference Format:
Pedro Valero-Lara, Frank Liu, and Jeffrey S. Vetter and Ian Jorquera. 2023. Mixed-Precision S/DGEMM Using the TF32 and TF64 Frameworks on Low-Precision AI Tensor Cores. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3624062.3624084

SC-W 2023, November 12–17, 2023, Denver, CO, USA

ACM ISBN 979-8-4007-0785-8/23/11…$15.00
https://doi.org/10.1145/3624062.3624084

1 INTRODUCTION

Recently, graphics processing unit (GPU) and central processing unit (CPU) vendors have become focused on accelerating general matrix multiplication (GEMM) with low-precision AI cores that are now available on newer GPUs, including NVIDIA's Tensor Cores [19], AMD's Matrix Corescite [2], and ARM's SME [4], among other specialized architectures [3, 31]. Accelerators equipped with these lower-precision cores outperform traditional CPUs/GPUs in artificial intelligence (AI) or machine learning (ML) workloads (e.g., low- or mixed-precision arithmetic) and are more power efficient [6].

These specialized hardware components essentially compute a matrix-matrix multiplication using low-precision operands, which are the principal components of multiple AI and ML applications [15, 16]. GEMM is defined as the operation $C \leftarrow \alpha AB+\beta C$, where A is an $m\times k$ matrix, B is a $k\times n$ matrix, and C is an $m\times n$ matrix [18, 40]. The matrices A, B, and C and the constants α and β have entries stored as floating point values generally following the IEEE framework for half-precision (FP16-HGEMM), single precision (FP32-SGEMM), or double precision (FP64-DGEMM).

Although these components are designed to accelerate AI/ML applications, they can also be used to accelerate other kinds of operations, including fast Fourier transform (FFT) [35], linear algebra operations [5, 12, 25], or comparative genomics (the first application to obtain an exaflop) [26], among many others [14, 22, 24, 37]. Given the high performance achieved by this specialized hardware and the importance of GEMM for multiple math libraries and applications [8, 9, 28, 29, 38, 39], the High-Performance Linpack (HPL) benchmark [36], the standard high-performance computing (HPC) benchmark used for the TOP500 list [17], was adapted to make use of these accelerators (i.e., HPL-AI) [21].

This work extends the analysis conducted by M. Fasi et al. [10] by using the new TF32 format. In that work, the authors studied the use of multiword arithmetic and the FP16 formats on NVIDIA Tensor Cores for single-precision operations. The contribution of this work is twofold: (1) the evaluation of the correctness and performance of the TF32 format, which is a new format that enables fast and low-precision matrix-matrix computation with no expensive 32-bit to 16-bit transformations. Also, the use of this new format does not require additional and costly operations to guarantee correctness/precision for single-precision operations, and (2) the proposal and study of a new format called TF64 for high-performance and double-precision Tensor Core–accelerated applications.

The rest of the paper is organized as follows: Section 2 introduces the main characteristics of the Tensor Core architecture and reviews both the methodologies used to obtain high-precision on

low-precision hardware (multiword arithmetic) and the different novel and well-known formats that we can use on these AI accelerators. We evaluate the correctness of the techniques and formats used in this study in Section 3. Related work is presented in Section 4. Finally, conclusions and future directions are outlined in Section 5.

2 MIXED-PRECISION S/DGEMM USING TF32 AND TF64 TENSOR CORE FRAMEWORKS

2.1 Tensor Cores

Tensor Cores [19] are specialized accelerator cores that compute tensors, which are mathematical objects that describe the relationships between other mathematical objects that are linked together. This allows 4×4 matrices to be multiplied and added to a 4×4 matrix. Novel methodologies have been introduced to improve performance by increasing the size of such accelerators. As Tensor Cores have been developed further, they have extended their capabilities to operate on different formats (Figure 1), including low- and mixed-precision arithmetic, which can accelerate AI operations. Table 1 lists the performance of Tensor Cores for different NVIDIA GPU generations.

Table 1: Tensor-Core Performance (TFLOPS) on different generations and formats

	FP64	TF32	BF16	FP16	INT8
Volta V100	-	-	-	128	-
Ampere A100	19.5	156	312	312	624
Hopper H100	67	989	1,979	1,979	3,958

Note that V100, A100, and H100 GPUs provide a wide range of different arithmetics with varying levels of performance. The four-generation tensor cores that equip the Ampere A100 architecture provide more levels of precision than the tensor cores available on its predecessors. Also, different arithmetics have different numbers of computational units on the different NVIDIA GPU architectures, which influences the final throughput.

It is important to note that all these new low-precision formats (FP16, TF32, etc.) are not equivalent to IEEE standards floating-point arithmetic. Also, operations on those formats may not satisfy the many IEEE requirements for correct and optimal rounding modes.

2.2 Multiword Arithmetic

Throughout this study, we will utilize and apply the multiword arithmetic [10], which stores high-precision floating points as the sum of lower-precision floating points. For example, for TF32, let A and B be $n \times n$ matrices with entries stored as FP32. Define $A_1 := (TF32)A$ as the matrix A, the elements of which have been formatted to TF32, and define $A_2 := (TF32)(A - A_1)$ as the matrix that stores the values lost due to the conversion. This gives us $A \approx A_1 + A_2$. Similarly, we define $B_1 := (TF32)B$ and $B_2 := (TF32)(B - B_1)$ in the same way. With the approximations for $A \approx A_1 + A_2$ and $B \approx B_1 + B_2$, we can compute the GEMM using the following approximation:

$$A \cdot B \approx (A_1 + A_2) \cdot (B_1 + B_2) = A_1B_1 + A_1B_2 + A_2B_1 + A_2B_2$$

Here, we can approximate a single-precision general matrix multiplication (SGEMM) by using four independent matrix multiplications, each computed using the TF32 compute mode. Notably, the final product plays a relatively insignificant role in improving precision, and this allows us to accelerate this method by removing the final A_2B_2 GEMM.

To better understand the operations conducted for multiword arithmetic, we include a simple pseudocode in Figure 2 to illustrate the operations required for this analysis.

2.3 TF32

The TF32 format (Figure 3) adopts 8 exponent bits, 10 bits of mantissa, and 1 sign bit. This new format covers the same range of values as FP32 and maintains more precision than BF16 and the same amount as FP16. The precision for TF32 has enough margin for AI applications.

The TF32 mixed-precision framework (Figure 1) for GEMM takes as input two matrices with entries in single precision (FP32). It then converts them to TF32 and computes the multiplications in full precision. Finally, the accumulation or addition is computed in FP32 to limit any accumulation error [7]. The TF32 mixed-precision framework on the NVIDIA A100 can achieve 156 TFLOPS of peak performance, whereas standard FP32 (non-Tensor Core) achieves 19.5 TFLOPS of peak performance. The result is an 8× performance uplift when using TF32.

2.4 TF64

We also extend this work into double precision. First, we propose a new framework called TF64 (also illustrated in Figure 1) that extends the TF32 mixed-precision framework to double precision; that is, the FP64 inputs are converted to FP32 (or a potential future TF64). Multiplication is then computed in full precision, and accumulation is computed in FP64. With adequate hardware support, we assume that many of the same performance increases seen in the TF32 compute mode would also manifest in these potential double-precision frameworks.

Although these frameworks do not exist on Tensor Cores, we find value in understanding the potential benefits of mixed-precision methods on these higher-precision frameworks for scientific applications.

The method presented by M. Fasi et al. [11] naturally extends to using the new TF64 Tensor Core framework. For A and B $n \times n$ matrices with entries stored as FP64, we define $A_1 := (FP32)A$, $A_2 := (FP32)(A - A_1)$, $B_1 := (FP32)B$, and $B_2 := (FP32)(B - B_1)$. This gives us the approximations $A \approx A_1 + A_2$ and $B \approx B_1 + B_2$, meaning we can approximate a double-precision GEMM, $A \cdot B$, with the sum of four FP32, $A_1B_1 + A_1B_2 + A_2B_1 + A_2B_2$.

3 ANALYSIS

For the analysis, cuBLAS mixed-precision kernels (cublasGemmEx) were computed on NVIDIA's A100 GPU, and kernels labeled as TF64 were computed through software on an Intel Xeon E5-2698 v4 CPU to mimic possible future computing frameworks.

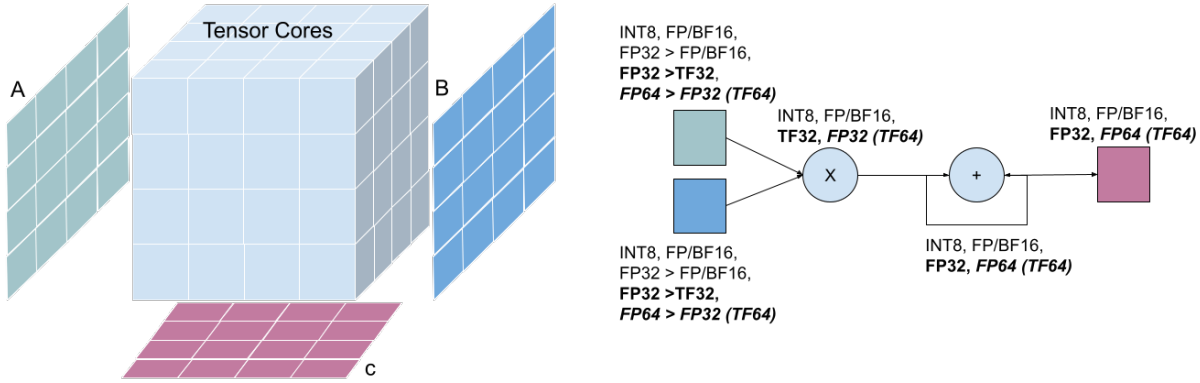

Figure 1: Current and proposed (TF64) Tensor Core frameworks. In bold are the frameworks evaluated in this study.

```
float A, A1, A2, B, B1, B2, C_TF32, C;

//Note that for TF32 and TF64, costly casting are not necessary
A1 = A;
A2 = (A - A1);

B1 = B;
B2 = (B - B1);

C_TF32 = C;

cublasGemmEx (A1, B1, C_TF32);
cublasGemmEx (A1, B2, C_TF32);
cublasGemmEx (A2, B2, C_TF32);
// Note that this line is only added for the analysis of correctness
cublasGemmEx (A2, B2, C_TF32);

cublasSgemm (A, B, C);

errorComputaion (C_TF32, C);
```

Figure 2: Example of multiword arithmetic code for TF32.

3.1 Error Computation

As others have done in related work (Section 4), we compute the ℓ_2 forward error[1] on the matrix multiplication $C = AB$ for random matrices A, B, and C. The ℓ_2 forward error is defined as

$$\ell_2 error = \frac{||C - \hat{C}||_2}{||A||_2||B||_2},$$

where $\hat{C}$ is the matrix product computed in double or quad precision (FP128 uses the quad math GNU library [1]) for comparison with the TF32 and TF64 mixed-precision frameworks, respectively, and $||\cdot||_2$ is the ℓ_2 norm.

For the sake of completeness [34], we also considered other errors, such as ℓ_1:

$$\ell_1 error = \frac{||C - \hat{C}||_1}{||A||_1||B||_1}$$

and ℓ_∞:

$$\ell_\infty error = \frac{||C - \hat{C}||_\infty}{||A||_\infty||B||_\infty}.$$

[1]https://netlib.org/lapack/lug/node75.html

3.2 SGEMM and TF32

First, we look at the TF32 mixed-precision framework with existing hardware support on the A100 (Figure 4). The 1×TF32 mixed-precision framework for GEMM can provide an improvement of about 4× over the 1×FP16 GEMM. However, this improvement is not enough and provides an error of magnitude of $1e^{-5} - 1e^{-6}$.

Using the multiword arithmetic and 4×TF32 GEMMs, we achieve an error close to the error of the SGEMM, with magnitudes of error much lower than that of 1×TF32 GEMM or 1×FP16 GEMM. Furthermore, the precision lost by omitting the fourth matrix multiplication has no noticeable effect on the overall error. Additionally, we have included the |SGEMM-3×TF32| error, as shown on the right side of the graphs. This shows that we can provide a close approximation to that of an SGEMM when using 3×TF32 GEMMs. Based on the performance of the NVIDIA A100 GPUs, we obtain a peak performance of about 53 Tflop/s and a performance uplift of 2.6× over SGEMM when using 3×TF32 GEMMs.

This analysis extends the work of M. Fasi et al. [10] by using the new TF32 Tensor Core framework. By using this framework, no costly 32-bit to 16-bit casting or extra memory is necessary to benefit from using Tensor Core accelerators. The casting is computed at the hardware level (Figure 1) during the execution of the operations instead of at the software level (Figure 2). We can also provide an equivalent or even better precision by using the simplest multiword arithmetic algorithm variant, and no complex variants or tuning is necessary to improve precision. In terms of performance, the simplicity of the algorithm and code used provides equivalent or even faster performance despite not using the higher-performing FP16 framework (Table 1).

3.3 DGEMM and TF64

We propose another new Tensor Core framework to pursue high performance for double-precision HPC applications on low-precision AI accelerators. Although this framework does not exist today, we believe these methods will be effective in accelerating matrix multiplication for scientific applications while maintaining a high level of precision. Also, given the current trend in this architecture with an important addition of new and more precise data formats in the

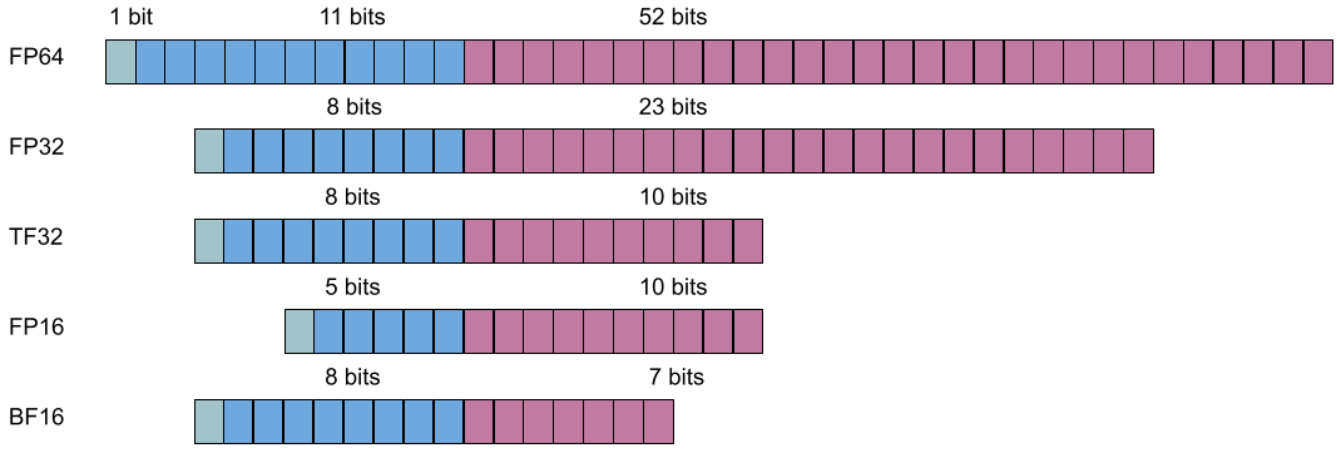

Figure 3: Comparison of bit layout (bit sign, exponent, and fraction) for FP64, FP32, FP16, BF16, and TF32 [19].

last few years, we can expect to have something similar to what we are proposing in the near future.

As expected (Figure 5), we see that this new framework can provide more precision than SGEMM. However, like in our previous comparison of 1×TF32 GEMM vs. HGEMM, this improvement does not match the requirements for double-precision results. However, when using 3×TF64 GEMM, we achieve an error equivalent in magnitude to that of DGEMM. In this case, there is a noticeable effect from omitting the fourth multiplication, particularly in relatively small matrices, but it remains insignificant when compared with the precision gained from the first three matrix multiplications. Overall, we see that approximating DGEMM with 3×TF64 GEMM provides an error reduction on the order of 10^7 for 1×TF64. As in the previous analysis, we included the |DGEMM-3×TF64| error, as illustrated on the right side of the graphs (Figure 5).

4 RELATED WORK

Besides NVIDIA Tensor Cores discussed in this paper, several companies are also employing and developing specialized hardware for high-performance inference, such as AMD [2], ARM [4], Intel, and Cerebras [3, 31]. But not only hardware vendors are designing AI accelerators. Movidius developed the Myriad 2 Vision Processing Unit [23]. Google designed and developed a Tensor Processing Unit (TPU) specifically for inference workloads [33].

We can find some examples of using multiword arithmetic on low-precision hardware, including the work of Markidis et al. [27], who call this technique *precision refinement*. Similar approaches can be found for FFT [32, 35], although most of the state-of-the-art references focus on matrix-matrix multiplication [27, 30]. Also, as in our work, these techniques were used to propose new ideas for more efficient and faster hardware [13] (e.g., block fused multiply–add units on future Intel hardware).

Iterative refinement solvers [12] is another successful example of using AI accelerators in HPC. This kind of algorithm can effectively use the low-precision AI tensor core and reach the necessary high precision required by HPC applications. Indeed, these were the techniques implemented for HPL-AI [20, 21] to reach the exascale by using AI accelerators. However, these algorithms require computing the same kind of operations repeatedly because of the iterative nature of these algorithms, which may reduce the potential benefit in terms of the performance of using tensor cores-like processors. In general, the number of iterations depends on the coefficient of the matrix to be computed, in other words, the condition number [12]. In fact, in certain cases, it can be difficult to reach a good precision [12]. Unlike, iterative solvers, the use of direct solvers can alleviate such limitations in terms of both: performance and precision. Indeed, the work presented in this paper can be used in such direct solvers [38].

As mentioned, this work extends the previously published work by M. Fasi et al. [10] by using the new TF32 format. Also, as far as we know, ours is the first work to propose and analyze a new framework (TF64) for double-precision operation on Tensor Cores.

5 CONCLUSIONS AND FUTURE DIRECTIONS

We extended the state of the art by using the new TF32 Tensor Core format to provide accurate solutions by using relatively simple techniques based on multiword arithmetic, all without the costly 32-bit to 16-bit software-level castings. Also, the achieved performance is equivalent to or higher than previous solutions based on the FP16 Tensor Core framework. This work also proposed a novel Tensor Core framework called TF64 for double-precision and demonstrated the potential effectiveness of mixed-precision Tensor Cores to accelerate double-precision GEMM for scientific and HPC applications.

More analyses are required to study new TF64 formats by using even lower-precision formats with lower exponent and fractions bits. Further work is also needed to expand this analysis to other AI core hardware, such as AMD Matrix Cores, ARM SME, and others.

ACKNOWLEDGMENTS

This research used resources of the Oak Ridge Leadership Computing Facility and the Experimental Computing Laboratory at the Oak Ridge National Laboratory, which is supported by DOE's Office of

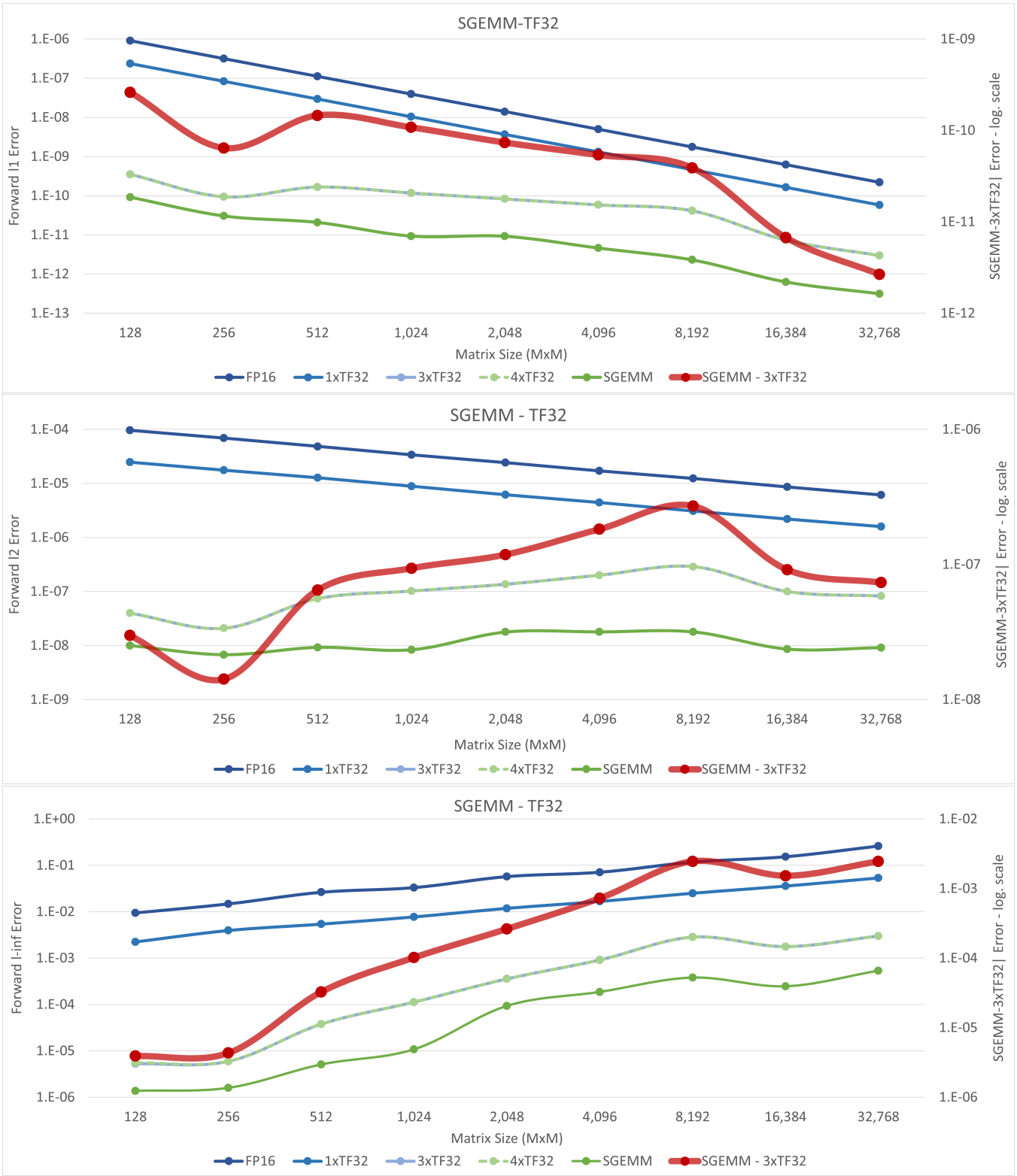

Figure 4: SGEMM and TF32 mixed-precision error analysis.

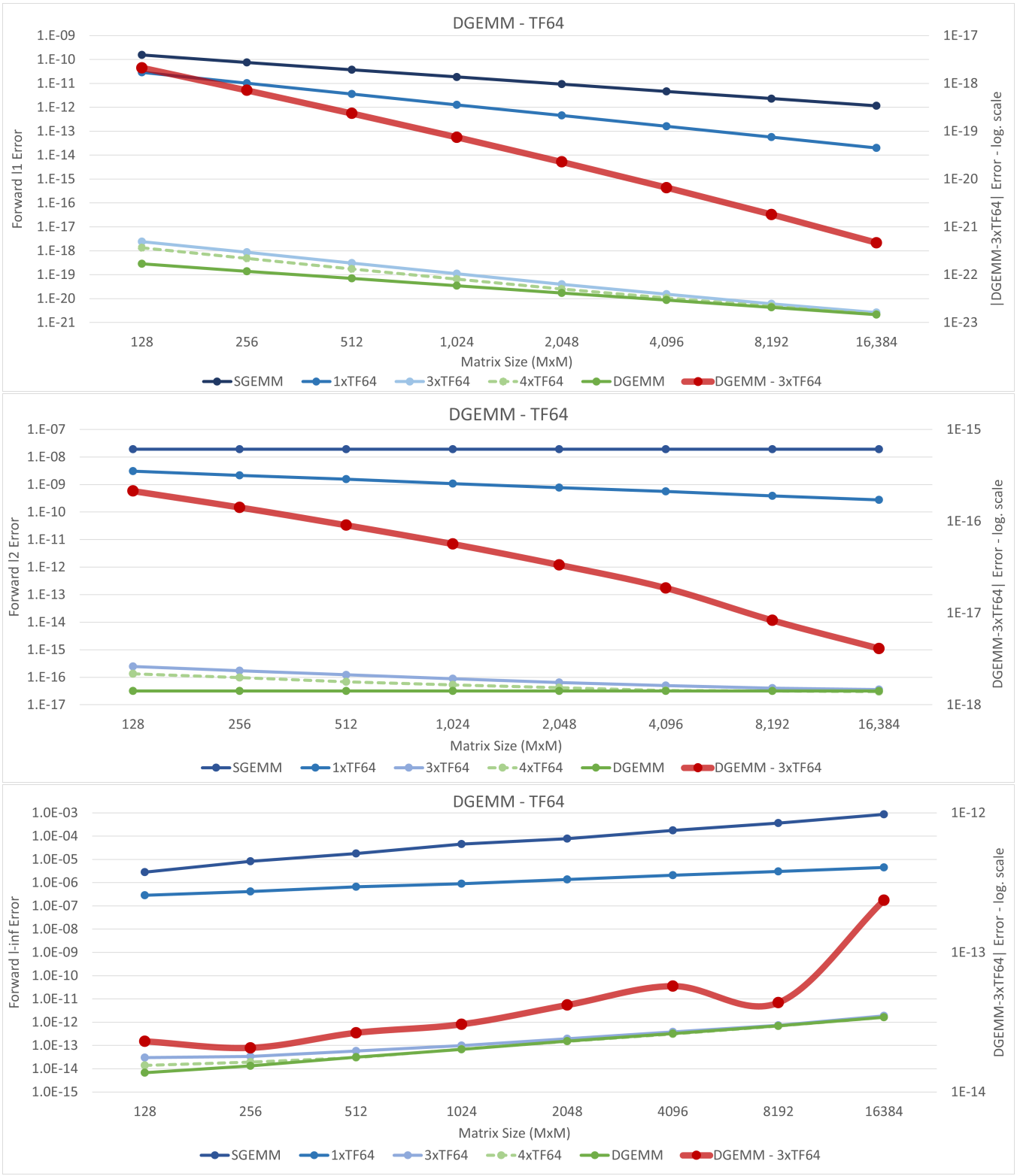

Figure 5: DGEMM and TF64 mixed-precision error analyis.

Science under Contract No. DE-AC05-00OR22725. This research was supported in part by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the DOE's Office of Science and the National Nuclear Security Administration. This manuscript has been authored by UT-Battelle LLC under Contract No. DE-AC05-00OR22725 with the DOE. The publisher, by accepting the article for publication, acknowledges that the US Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of the manuscript or allow others to do so, for US Government purposes. The DOE will provide public access to these results in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

REFERENCES

[1] 2018. The GCC Quad-Precision Math Library. https://gcc.gnu.org/onlinedocs/gcc-8.2.0/libquadmath.pdf [Online; accessed 30-May-2023].

[2] 2021. AMD Instinct MI250X Accelerator. https://www.amd.com/en/products/server-accelerators/instinct-mi250x. [Online; accessed 30-May-2023].

[3] 2022. Cerebras. https://www.cerebras.net/. [Online; accessed 30-May-2023].

[4] 2022. The Scalable Matrix Extension (SME), for Armv9-A. https://developer.arm.com/documentation/ddi0616/latest. [Online; accessed 30-May-2023].

[5] Ahmad Abdelfattah, Hartwig Anzt, Erik G. Boman, Erin C. Carson, Terry Cojean, Jack J. Dongarra, Alyson Fox, Mark Gates, Nicholas J. Higham, Xiaoye S. Li, Jennifer A. Loe, Piotr Luszczek, Srikara Pranesh, Siva Rajamanickam, Tobias Ribizel, Barry F. Smith, Kasia Swirydowicz, Stephen J. Thomas, Stanimire Tomov, Yaohung M. Tsai, and Ulrike Meier Yang. 2021. A survey of numerical linear algebra methods utilizing mixed-precision arithmetic. *Int. J. High Perform. Comput. Appl.* 35, 4 (2021). https://doi.org/10.1177/10943420211003313

[6] Ehsan Atoofian. 2023. PTTS: Power-aware tensor cores using two-sided sparsity. *J. Parallel Distributed Comput.* 173 (2023), 70–82. https://doi.org/10.1016/j.jpdc.2022.11.004

[7] Pierre Blanchard, Nicholas J. Higham, Florent Lopez, Theo Mary, and Srikara Pranesh. 2020. Mixed Precision Block Fused Multiply-Add: Error Analysis and Application to GPU Tensor Cores. *SIAM Journal on Scientific Computing* 42 (2020). https://doi.org/10.1137/19M1289546

[8] Jack J. Dongarra, Mark Gates, Azzam Haidar, Jakub Kurzak, Piotr Luszczek, Panruo Wu, Ichitaro Yamazaki, Asim YarKhan, Maksims Abalenkovs, Negin Bagherpour, Sven Hammarling, Jakub Šístek, David Stevens, Mawussi Zounon, and Samuel D. Relton. 2019. PLASMA: Parallel Linear Algebra Software for Multicore Using OpenMP. *ACM Trans. Math. Softw.* 45, 2 (2019), 16:1–16:35. https://doi.org/10.1145/3264491

[9] Mohammed A. Al Farhan, Ahmad Abdelfattah, Stanimire Tomov, Mark Gates, Dalal Sukkari, Azzam Haidar, Robert Rosenberg, and Jack J. Dongarra. 2020. MAGMA templates for scalable linear algebra on emerging architectures. *Int. J. High Perform. Comput. Appl.* 34, 6 (2020). https://doi.org/10.1177/1094342020938421

[10] Massimiliano Fasi, Nicholas J. Higham, Florent Lopez, Théo Mary, and Mantas Mikaitis. 2023. Matrix Multiplication in Multiword Arithmetic: Error Analysis and Application to GPU Tensor Cores. *SIAM J. Sci. Comput.* 45, 1 (2023), 1. https://doi.org/10.1137/21m1465032

[11] Massimiliano Fasi, Nicholas J. Higham, Florent Lopez, Theo Mary, and Mantas Mikaitis. 2023. Matrix Multiplication in Multiword Arithmetic: Error Analysis and Application to GPU Tensor Cores. (2023). http://eprints.maths.manchester.ac.uk/id/eprint/286 MIMS Preprint(submitted).

[12] Azzam Haidar, Stanimire Tomov, Jack J. Dongarra, and Nicholas J. Higham. 2018. Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2018, Dallas, TX, USA, November 11-16, 2018*. IEEE / ACM, 47:1–47:11. http://dl.acm.org/citation.cfm?id=3291719

[13] Greg Henry, Ping Tak Peter Tang, and Alexander Heinecke. 2019. Leveraging the bfloat16 Artificial Intelligence Datatype For Higher-Precision Computations. In *26th IEEE Symposium on Computer Arithmetic, ARITH 2019, Kyoto, Japan, June 10-12, 2019*, Naofumi Takagi, Sylvie Boldo, and Martin Langhammer (Eds.). IEEE, 69–76. https://doi.org/10.1109/ARITH.2019.00019

[14] Zhuoran Ji and Cho-Li Wang. 2022. Efficient exact K-nearest neighbor graph construction for billion-scale datasets using GPUs with tensor cores. In *ICS '22: 2022 International Conference on Supercomputing, Virtual Event, June 28 - 30, 2022*, Lawrence Rauchwerger, Kirk W. Cameron, Dimitrios S. Nikolopoulos, and Dionisios N. Pnevmatikatos (Eds.). ACM, 10:1–10:12. https://doi.org/10.1145/3524059.3532368

[15] Marc Jordà, Pedro Valero-Lara, and Antonio J. Peña. 2019. Performance Evaluation of cuDNN Convolution Algorithms on NVIDIA Volta GPUs. *IEEE Access* 7 (2019), 70461–70473. https://doi.org/10.1109/ACCESS.2019.2918851

[16] Marc Jordà, Pedro Valero-Lara, and Antonio J. Peña. 2022. cuConv: CUDA implementation of convolution for CNN inference. *Clust. Comput.* 25, 2 (2022), 1459–1473. https://doi.org/10.1007/s10586-021-03494-y

[17] Awais Khan, Hyogi Sim, Sudharshan S. Vazhkudai, Ali Raza Butt, and Youngjae Kim. 2021. An Analysis of System Balance and Architectural Trends Based on Top500 Supercomputers. In *HPC Asia 2021: The International Conference on High Performance Computing in Asia-Pacific Region, Virtual Event, Republic of Korea, January 20-21, 2021*, Soonwook Hwang and Heon Young Yeom (Eds.). ACM, 11–22. https://doi.org/10.1145/3432261.3432263

[18] Hyeonjin Kim and William J. Song. 2023. LAS: Locality-Aware Scheduling for GEMM-Accelerated Convolutions in GPUs. *IEEE Trans. Parallel Distributed Syst.* 34, 5 (2023), 1479–1494. https://doi.org/10.1109/TPDS.2023.3247808

[19] Ronny Krashinsky, Olivier Giroux, Stephen Jones, Nick Stam, and Sridhar Ramaswamy. 2023. NVIDIA Ampere Architecture In-Depth. https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/ [Online; accessed 30-May-2023].

[20] Shuhei Kudo, Keigo Nitadori, Takuya Ina, and Toshiyuki Imamura. 2020. Implementation and Numerical Techniques for One EFlop/s HPL-AI Benchmark on Fugaku. In *11th IEEE/ACM Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, ScalA@SC 2020, Atlanta, GA, USA, November 13, 2020*. IEEE, 69–76. https://doi.org/10.1109/ScalA51936.2020.00014

[21] Shuhei Kudo, Keigo Nitadori, Takuya Ina, and Toshiyuki Imamura. 2020. Prompt Report on Exa-Scale HPL-AI Benchmark. In *IEEE International Conference on Cluster Computing, CLUSTER 2020, Kobe, Japan, September 14-17, 2020*. IEEE, 418–419. https://doi.org/10.1109/CLUSTER49012.2020.00058

[22] Wai-Kong Lee, Hwajeong Seo, Zhenfei Zhang, and Seong Oun Hwang. 2022. TensorCrypto: High Throughput Acceleration of Lattice-Based Cryptography Using Tensor Core on GPU. *IEEE Access* 10 (2022), 20616–20632. https://doi.org/10.1109/ACCESS.2022.3152217

[23] Vasileios Leon, Kiamal Z. Pekmestzi, and Dimitrios Soudris. 2022. Systematic Embedded Development and Implementation Techniques on Intel Myriad VPUs. In *30th IFIP/IEEE 30th International Conference on Very Large Scale Integration, VLSI-SoC 2022, Patras, Greece, October 3-5, 2022*. IEEE, 1–2. https://doi.org/10.1109/VLSI-SoC54400.2022.9939592

[24] Shigang Li, Kazuki Osawa, and Torsten Hoefler. 2022. Efficient Quantized Sparse Matrix Operations on Tensor Cores. *CoRR* abs/2209.06979 (2022). https://doi.org/10.48550/arXiv.2209.06979 arXiv:2209.06979

[25] Neil Lindquist, Piotr Luszczek, and Jack J. Dongarra. 2022. Accelerating Restarted GMRES With Mixed Precision Arithmetic. *IEEE Trans. Parallel Distributed Syst.* 33, 4 (2022), 1027–1037. https://doi.org/10.1109/TPDS.2021.3090757

[26] Lixiang Luo, Tjerk P. Straatsma, Luis Enrique Aguilar-Suárez, Ria Broer, Dmytro Bykov, Eduardo F. D'Azevedo, Shirin S. Faraji, Kalyana C. Gottiparthi, Coen de Graaf, James Austin Harris, Remco W. A. Havenith, Hans Jørgen Aagard Jensen, Wayne Joubert, R. K. Kathir, Jeff Larkin, Ying Wai Li, Dmitry I. Lyakh, O. E. Bronson Messer, Matthew R. Norman, Joseph C. Oefelein, Ramanan Sankaran, Andreas F. Tillack, Ashleigh L. Barnes, Lucas Visscher, Jack C. Wells, and Meilani Wibowo. 2020. Pre-exascale accelerated application development: The ORNL Summit experience. *IBM J. Res. Dev.* 64, 3/4 (2020), 11:1–11:21. https://doi.org/10.1147/JRD.2020.2965881

[27] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S. Vetter. 2018. NVIDIA Tensor Core Programmability, Performance & Precision. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2018, Vancouver, BC, Canada, May 21-25, 2018*. IEEE Computer Society, 522–531. https://doi.org/10.1109/IPDPSW.2018.00091

[28] Narasinga Rao Miniskar, Mohammad Alaul Haque Monil, Pedro Valero-Lara, Frankie Y. Liu, and Jeffrey S. Vetter. 2022. IRIS-BLAS: Towards a Performance Portable and Heterogeneous BLAS Library. In *29th IEEE International Conference on High Performance Computing, Data, and Analytics, HiPC 2022, Bengaluru, India, December 18-21, 2022*. IEEE, 256–261. https://doi.org/10.1109/HiPC56025.2022.00042

[29] Mohammad Alaul Haque Monil, Narasinga Rao Miniskar, Frank Y. Liu, Jeffrey S. Vetter, and Pedro Valero-Lara. 2022. LaRIS: Targeting Portability and Productivity for LAPACK Codes on Extreme Heterogeneous Systems by Using IRIS. In *IEEE/ACM Redefining Scalability for Diversely Heterogeneous Architectures Workshop, RSDHA@SC 2022, Dallas, TX, USA, November 13-18, 2022*. IEEE, 12–21. https://doi.org/10.1109/RSDHA56811.2022.00007

[30] Daichi Mukunoki and Takeshi Ogita. 2020. Performance and energy consumption of accurate and mixed-precision linear algebra kernels on GPUs. *J. Comput. Appl. Math.* 372 (2020), 112701. https://doi.org/10.1016/j.cam.2019.112701

[31] Thomas Norrie, Nishant Patil, Doe Hyun Yoon, George Kurian, Sheng Li, James Laudon, Cliff Young, Norman P. Jouppi, and David A. Patterson. 2021. The Design Process for Google's Training Chips: TPUv2 and TPUv3. *IEEE Micro* 41, 2 (2021), 56–63. https://doi.org/10.1109/MM.2021.3058217

[32] Louis Pisha and Lukasz Ligowski. 2021. Accelerating non-power-of-2 size Fourier transforms with GPU Tensor Cores. In *35th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2021, Portland, OR, USA, May 17-21, 2021*. IEEE, 507–516. https://doi.org/10.1109/IPDPS49936.2021.00059

[33] Sami Salamin, Georgios Zervakis, Florian Klemme, Hammam Kattan, Yogesh Singh Chauhan, Jörg Henkel, and Hussam Amrouch. 2022. Impact of NCFET Technology on Eliminating the Cooling Cost and Boosting the Efficiency of Google TPU. *IEEE Trans. Computers* 71, 4 (2022), 906–918. https://doi.org/10.1109/TC.2021.3065454

[34] Jorge Sastre and Jacinto Javier Ibáñez. 2023. On the backward and forward error of approximations of analytic functions and applications to the computation of matrix functions. *J. Comput. Appl. Math.* 419 (2023), 114706. https://doi.org/10.1016/j.cam.2022.114706

[35] Anumeena Sorna, Xiaohe Cheng, Eduardo F. D'Azevedo, Kwai Wong, and Stanimire Tomov. 2018. Optimizing the Fast Fourier Transform Using Mixed Precision on Tensor Core Hardware. In *25th IEEE International Conference on High Performance Computing Workshops, HiPCW 2018, Bengaluru, India, December 17-20, 2018*. IEEE, 3–7. https://doi.org/10.1109/HiPCW.2018.8634417

[36] Qiao Sun, Wenjing Ma, Jiachang Sun, and Huiyuan Li. 2023. Evolving the HPL benchmark towards multi-GPGPU clusters. *CCF Trans. High Perform. Comput.* 5, 1 (2023), 84–96. https://doi.org/10.1007/s42514-022-00128-6

[37] Yufei Sun, Long Zheng, Qinggang Wang, Xiangyu Ye, Yu Huang, Pengcheng Yao, Xiaofei Liao, and Hai Jin. 2022. Accelerating Sparse Deep Neural Network Inference Using GPU Tensor Cores. In *IEEE High Performance Extreme Computing Conference, HPEC 2022, Waltham, MA, USA, September 19-23, 2022*. IEEE, 1–7. https://doi.org/10.1109/HPEC55821.2022.9926300

[38] Pedro Valero-Lara, Sandra Catalán, Xavier Martorell, Tetsuzo Usui, and Jesús Labarta. 2020. sLASs: A fully automatic auto-tuned linear algebra library based on OpenMP extensions implemented in OmpSs (LASs Library). *J. Parallel Distributed Comput.* 138 (2020), 153–171. https://doi.org/10.1016/j.jpdc.2019.12.002

[39] Pedro Valero-Lara, Jungwon Kim, and Jeffrey S. Vetter. 2022. A Portable and Heterogeneous LU Factorization on IRIS. In *Euro-Par 2022: Parallel Processing Workshops - Euro-Par 2022 International Workshops, Glasgow, UK, August 22-26, 2022, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 13835)*, Jeremy Singer, Yehia Elkhatib, Dora Blanco Heras, Patrick Diehl, Nick Brown, and Aleksandar Ilic (Eds.). Springer, 17–31. https://doi.org/10.1007/978-3-031-31209-0_2

[40] Pedro Valero-Lara, Ivan Martínez-Pérez, Sergi Mateo, Raül Sirvent, Vicenç Beltran, Xavier Martorell, and Jesús Labarta. 2018. Variable Batched DGEMM. In *26th Euromicro International Conference on Parallel, Distributed and Network-based Processing, PDP 2018, Cambridge, United Kingdom, March 21-23, 2018*, Ivan Merelli, Pietro Liò, and Igor V. Kotenko (Eds.). IEEE Computer Society, 363–367. https://doi.org/10.1109/PDP2018.2018.00065

A ARTIFACT DESCRIPTION FOR REPRODUCIBILITY

The code used for this work and analysis is accessible via a public GitHub repository [2]. We used one NVIDIA A100 GPU for the TF32 error and performance analysis, while the TF64 analysis was computed through software using the Intel Xeon E5-2698 v4 CPU to mimic possible future computing frameworks. The code and the corresponding Makefile, which contains the details of the software stack used, can be found in the /src folder. Also, we provide the script (found in the /script folder) that uses the data (found in the /data folder) collected in our experiments to generate the plots (found in the /plot folder) presented in this work.

[2] https://github.com/pedrovalerolara/TF32-TF64.git