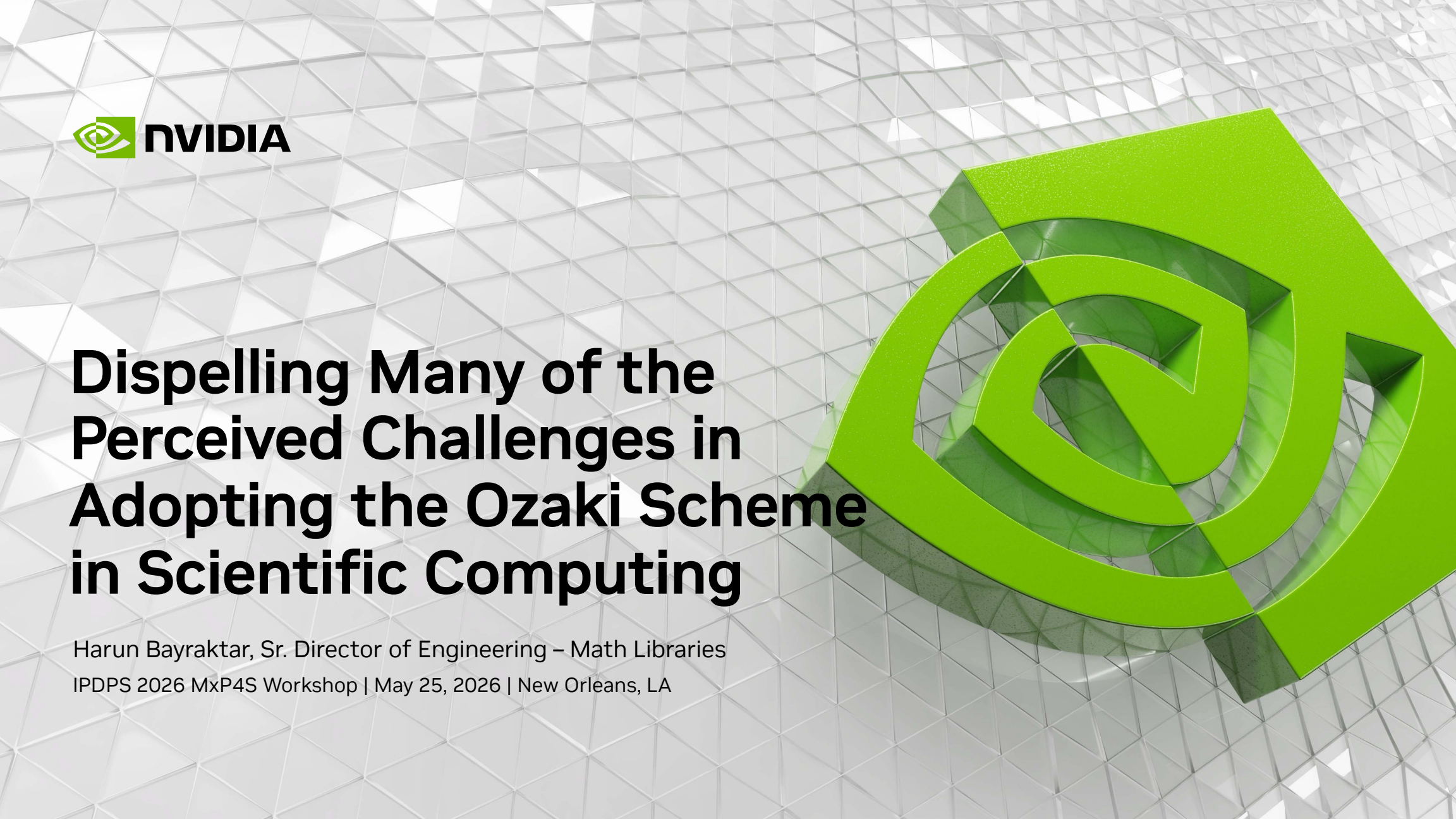


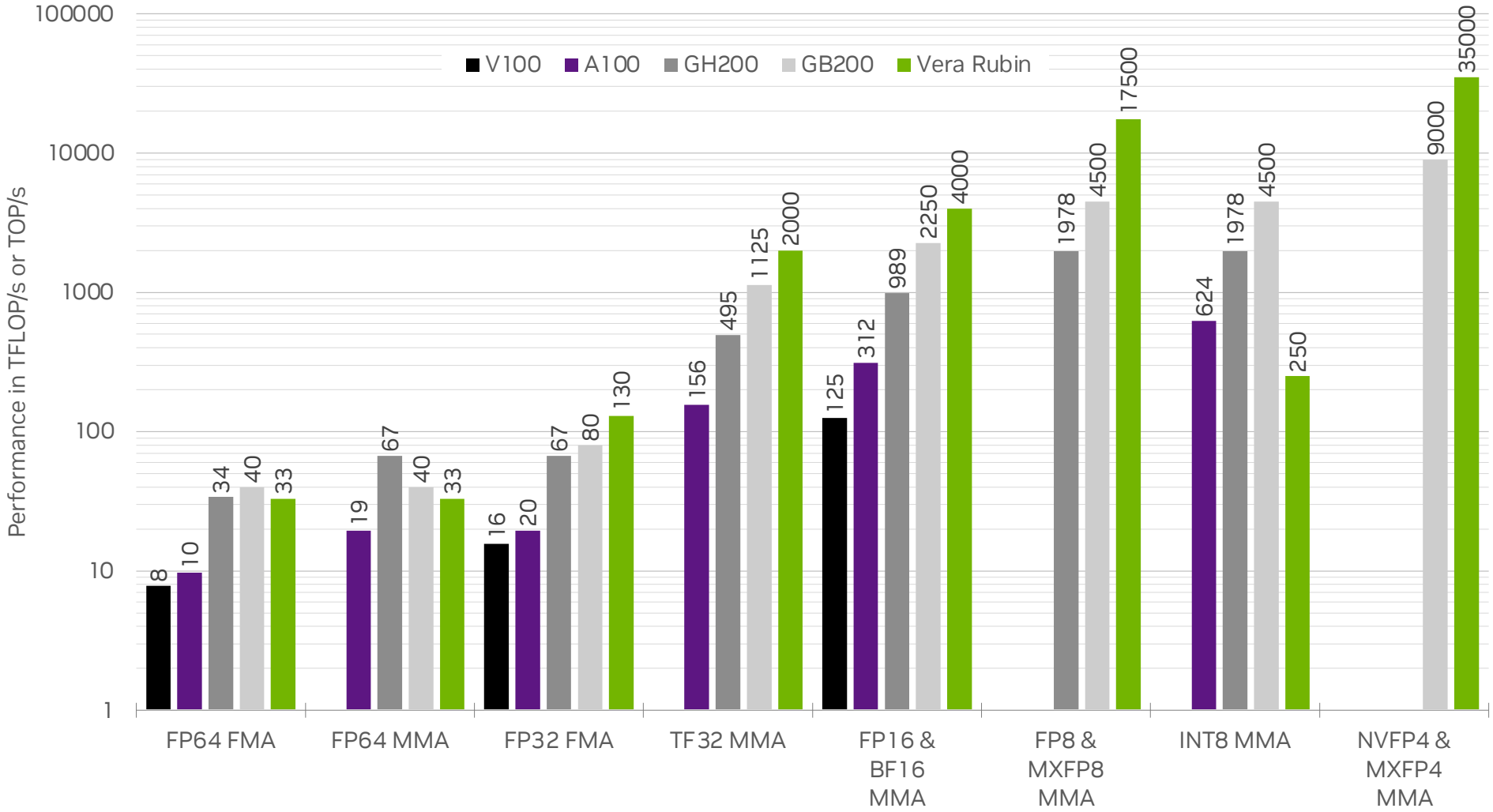
Dispelling Many of the Perceived Challenges in Adopting the Ozaki Scheme in Scientific Computing

Harun Bayraktar, Sr. Director of Engineering – Math Libraries

IPDPS 2026 MxP4S Workshop | May 25, 2026 | New Orleans, LA

Evolution of Peak Performance

Hardware Specs



NVIDIA

Ozaki Scheme for Matrix Multiplication

Enables utilization of Tensor Cores for matrix multiply operations using the desired level of precision

DGEMM on Integer Matrix Multiplication Unit

Hiroyuki Ootomo
ootomo.h@rio.gsic.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

Katsuhisa Ozaki
ozaki@sic.shibaura-it.ac.jp
Shibaura Institute of Technology
Saitama, Japan

Rio Yokota
rioyokota@gsic.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

ABSTRACT

Deep learning hardware achieves high throughput and low power consumption by reducing computing precision and specializing in matrix multiplication. For machine learning inference, fixed-point value computation is commonplace, where the input and output values and the model parameters are quantized. Thus, many processors are now equipped with fast integer matrix multiplication units (IMMU). It is of significant interest to find a way to harness these IMMUs to improve the performance of HPC applications while maintaining accuracy. We focus on the Ozaki scheme, which computes a high-precision matrix multiplication by using lower-precision computing units, and show the advantages and disadvantages of using IMMU. The experiment using integer Tensor Cores shows that we can compute double-precision matrix multiplication faster than cuBLAS and an existing Ozaki scheme implementation on FP16 Tensor Cores on NVIDIA consumer GPUs. Furthermore, we demonstrate accelerating a quantum circuit simulation by up to 4.33 while maintaining the FP64 accuracy.

CCS CONCEPTS

• **Computing methodologies** → **Parallel algorithms**; • **Theory of computation** → **Numeric approximation algorithms**.

KEYWORDS

matrix multiplication, fixed-point arithmetic, floating-point arithmetic, Tensor Cores

ACM Reference Format:

Hiroyuki Ootomo, Katsuhisa Ozaki, and Rio Yokota. 2018. DGEMM on Integer Matrix Multiplication Unit. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

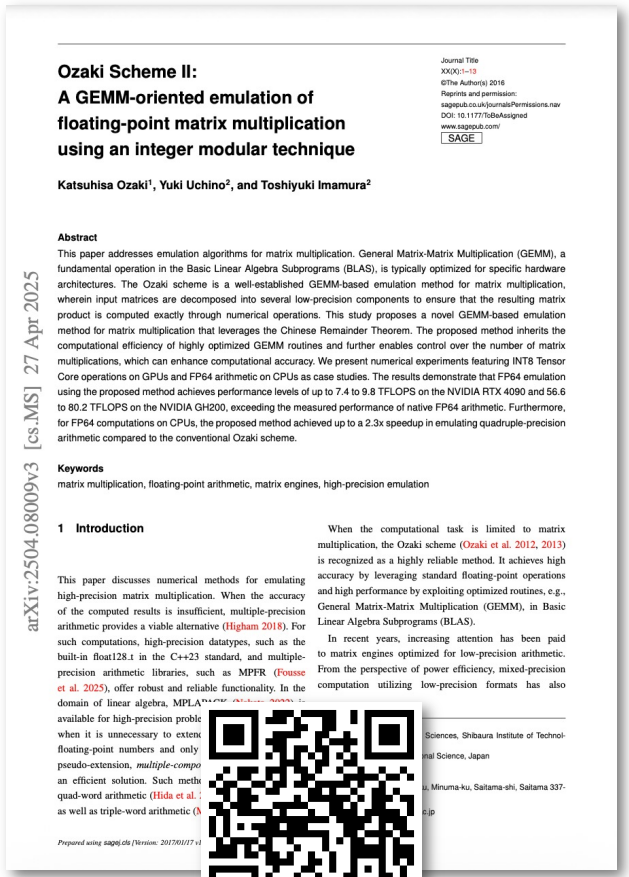
1 INTRODUCTION

The development of processors for machine learning is progressing rapidly, and researchers are exploring their potential for use in other HPC applications [1]. These processors often use low/mixed-precision floating-point, or low-bit-length integer computing units, such as NVIDIA Tensor Cores [26] and Google TPUs [16–18], to achieve high throughput. While the training of deep learning models typically requires floating-point types like FP16, Bfloat16, TF32, and FP32, the inference involves quantizing model parameters and input/output values to use lower-bit-length integer formats. This reduces data size, processor circuit area, and power consumption [14, 16]. As a result, high-performance processors like NVIDIA GPUs and edge devices like Google Coral Edge TPUs [33] are equipped with low-bit-length integer matrix multiplication units (IMMU). For instance, NVIDIA GPUs provide a DP4A instruction that can efficiently compute the inner product of two length-4 8-bit integer (INT8) vectors and accumulate the result in a 32-bit integer (INT32). In addition, NVIDIA Tensor Cores support the multiplication of INT8 matrices with INT32 accumulation from the Turing architecture and the multiplication of 4-bit integer (INT4) matrices from the Ampere architecture. These integer Tensor Cores can achieve a theoretical peak performance that is 2 ∼ 4 times faster than floating-point Tensor Cores. Other processors, such as Google TPU v1 [18], Intel AMX-INT8 [5], and Groq TSP [2], also support the multiplication of INT8 matrices with INT32 accumulation. In light of these advancements, our research aims to investigate whether these processors can be leveraged for other high-performance computing (HPC) applications.

Recent research has explored the use of machine learning processors for HPC applications, in particular low- and mixed-precision floating-point matrix multiplication units (FMMU). For instance, Haidar *et al.* used the FP16 Tensor Core to solve FP64 linear equations with an iterative refinement method [10]. Finkelstein *et al.* have employed FP16 Tensor Cores to perform time-independent quantum response calculations with sufficient accuracy [7]. Our previous study has utilized FP16 Tensor Cores to improve the throughput of quantum circuit simulation by emulating single-precision matrix multiplication using an error correction method and avoiding rounding inside Tensor Cores without loss of accuracy [27, 28]. However, despite the promising results obtained by these studies, few have explored the potential of integer matrix multiplication units for HPC applications. This paper investigates the feasibility of leveraging integer matrix multiplication units, for instance, INT8 Tensor Cores in NVIDIA GPUs, for HPC applications.

Even before the advent of deep learning and specialized hardware for it, researchers had been exploring ways to perform high-precision matrix multiplication on lower-precision computing units. One such approach is the double-double and quad-double precision, where a single value is represented as the sum of two and four FP64 values, respectively, and arithmetic operations are performed using a sequence of FP64 operations [13]. GEMM and other ouble-double precision have been evaluated l AMD Cypress GPUs [22, 24]. Another ap- heme, which also splits a value into multiple s [29, 30]. In this scheme, input matrices are trix slices, and the resulting matrix is ob- g the multiplication of these slices. The key

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX



Ozaki Scheme II: A GEMM-oriented emulation of floating-point matrix multiplication using an integer modular technique

Journal Title
XX(X):1–13
©The Author(s) 2016
Reprints and permission:
sagepub.co.uk/JournalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Katsuhisa Ozaki[1], Yuki Uchino[2], and Toshiyuki Imamura[2]

Abstract

This paper addresses emulation algorithms for matrix multiplication. General Matrix-Matrix Multiplication (GEMM), a fundamental operation in the Basic Linear Algebra Subprograms (BLAS), is typically optimized for specific hardware architectures. The Ozaki scheme is a well-established GEMM-based emulation method for matrix multiplication, wherein input matrices are decomposed into several low-precision components to ensure that the resulting matrix product is computed exactly through numerical operations. This study proposes a novel GEMM-based emulation method for matrix multiplication that leverages the Chinese Remainder Theorem. The proposed method inherits the computational efficiency of highly optimized GEMM routines and further enables control over the number of matrix multiplications, which can enhance computational accuracy. We present numerical experiments featuring INT8 Tensor Core operations on GPUs and FP64 arithmetic on CPUs as case studies. The results demonstrate that FP64 emulation using the proposed method achieves performance levels of up to 7.4 to 9.8 TFLOPS on the NVIDIA RTX 4090 and 56.6 to 80.2 TFLOPS on the NVIDIA GH200, exceeding the measured performance of native FP64 arithmetic. Furthermore, for FP64 computations on CPUs, the proposed method achieved up to a 2.3x speedup in emulating quadruple-precision arithmetic compared to the conventional Ozaki scheme.

Keywords

matrix multiplication, floating-point arithmetic, matrix engines, high-precision emulation

1 Introduction

This paper discusses numerical methods for emulating high-precision matrix multiplication. When the accuracy of the computed results is insufficient, multiple-precision arithmetic provides a viable alternative (Higham 2018). For such computations, high-precision datatypes, such as the built-in float128_t in the C++23 standard, and multiple-precision arithmetic libraries, such as MPFR (Fousse et al. 2025), offer robust and reliable functionality. In the domain of linear algebra, MPLA [illegible] available for high-precision proble when it is unnecessary to extend floating-point numbers and only pseudo-extension, *multiple-compo* an efficient solution. Such meth quad-word arithmetic (Hida et al. as well as triple-word arithmetic (

When the computational task is limited to matrix multiplication, the Ozaki scheme (Ozaki et al. 2012, 2013) is recognized as a highly reliable method. It achieves high accuracy by leveraging standard floating-point operations and high performance by exploiting optimized routines, e.g., General Matrix-Matrix Multiplication (GEMM), in Basic Linear Algebra Subprograms (BLAS).

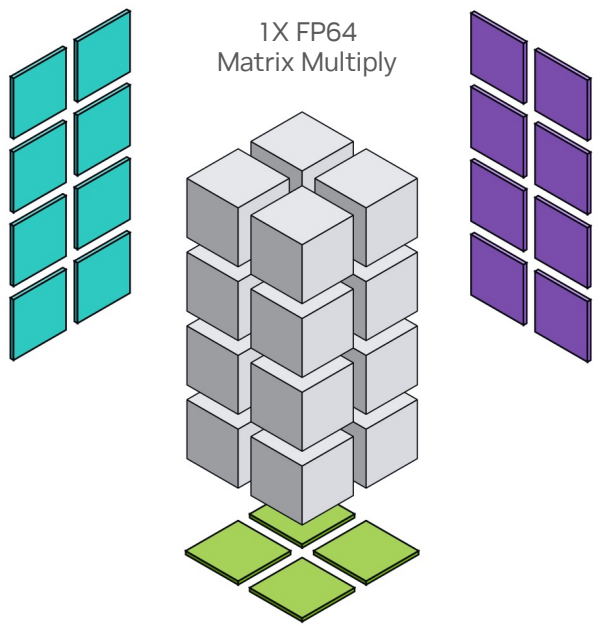
In recent years, increasing attention has been paid to matrix engines optimized for low-precision arithmetic. From the perspective of power efficiency, mixed-precision computation utilizing low-precision formats has also

Sciences, Shibaura Institute of Technol-
nal Science, Japan
, Minuma-ku, Saitama-shi, Saitama 337-
c.jp

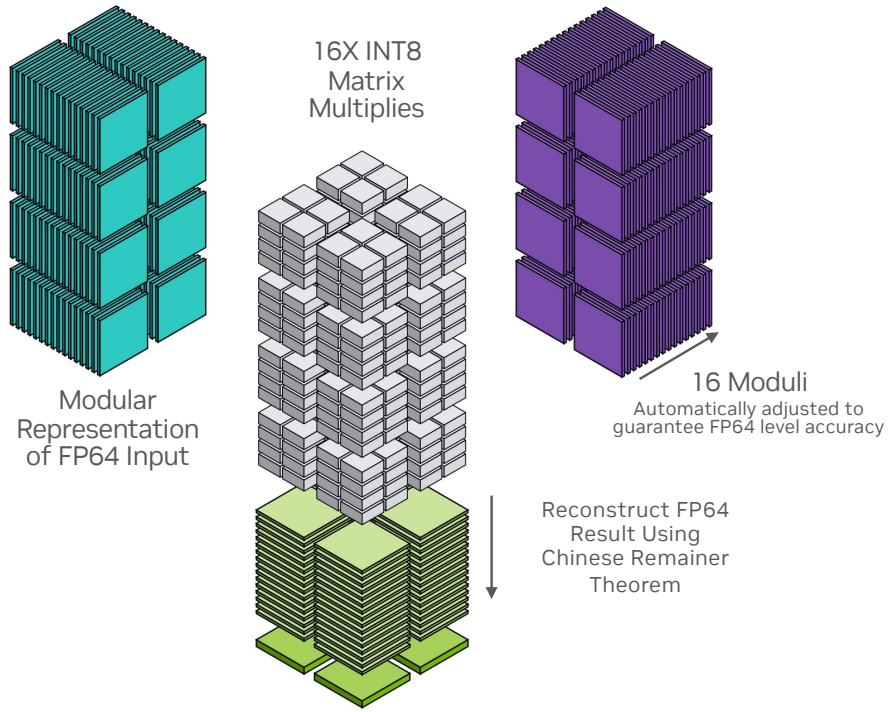
Prepared using sagej.cls *[Version: 2017/01/17 v1*

Matrix Multiplication Emulation With High Throughput Tensor Cores

Ozaki-II method delivers 3.5X speed-up on GB200 for FP64 matrix multiplies



DGEMM 40 TFLOP/s
FP64 Tensor Cores (DMMA)



DGEMM 150 TFLOP/s
Ozaki-II using INT8 Tensor Cores (IMMA)

Augmenting for a Broad User Base

Complementing Existing Research & Innovations to Productize Great Technology

DGEMM on Integer Matrix Multiplication Unit

Hiroyuki Ootomo
ootomo.h@rio.gsic.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

Katsuhisa Ozaki
ozaki@sic.shibaura-it.ac.jp
Shibaura Institute of Technology
Saitama, Japan

Rio Yokota
rioyokota@gsic.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

ABSTRACT

Deep learning hardware achieves high throughput and low power consumption by reducing computing precision and specializing in matrix multiplication. For machine learning inference, fixed-point value computation is commonplace, where the input and output values and the model parameters are quantized. Thus, many processors are now equipped with fast integer matrix multiplication units (IMMU). It is of significant interest to find a way to harness these IMMUs to improve the performance of HPC applications while maintaining accuracy. We focus on the Ozaki scheme, which computes a high-precision matrix multiplication by using lower-precision computing units, and show the advantages and disadvantages of using IMMU. The experiment using integer Tensor Cores shows that we can compute double-precision matrix multiplication faster than cuBLAS and an existing Ozaki scheme implementation on FP16 Tensor Cores on NVIDIA consumer GPUs. Furthermore, we demonstrate accelerating a quantum circuit simulation by up to 4.33 while maintaining the FP64 accuracy.

CCS CONCEPTS

• Computing methodologies → Parallel algorithms; • Theory of computation → Numeric approximation algorithms.

KEYWORDS

matrix multiplication, fixed-point arithmetic, floating-point arithmetic, Tensor Cores

ACM Reference Format:

Hiroyuki Ootomo, Katsuhisa Ozaki, and Rio Yokota. 2018. DGEMM on Integer Matrix Multiplication Unit. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

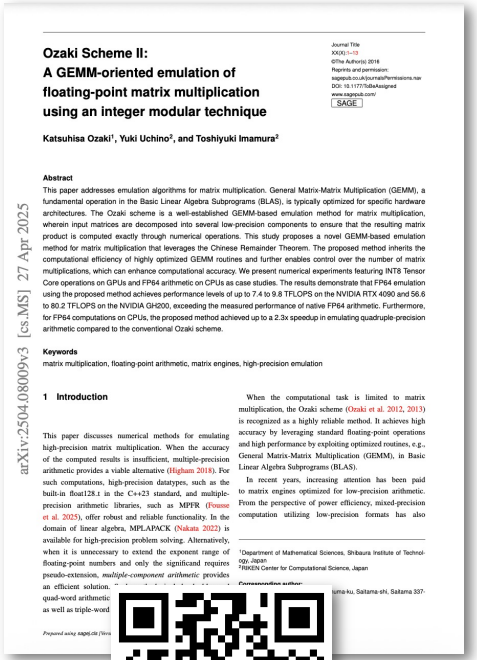
The development of processors for machine learning is progressing rapidly, and researchers are exploring their potential for use in other HPC applications [1]. These processors often use low/mixed-precision floating-point, or low-bit-length integer computing units, such as NVIDIA Tensor Cores [26] and Google TPUs [16–18], to achieve high throughput. While the training of deep learning models typically requires floating-point types like FP16, Bfloat16, TF32, and FP32, the inference involves quantizing model parameters and input/output values to use lower-bit-length integer formats. This reduces data size, processor circuit area, and power consumption [14, 16]. As a result, high-performance processors like NVIDIA GPUs and edge devices like Google Coral Edge TPUs [35] are equipped with low-bit-length integer matrix multiplication units (IMMU). For instance, NVIDIA GPUs provide a DP4A instruction that can efficiently compute the inner product of two length-4 8-bit integer (INT8) vectors and accumulate the result in a 32-bit integer (INT32). In addition, NVIDIA Tensor Cores support the multiplication of INT8 matrices with INT32 accumulation from the Turing architecture and the multiplication of 4-bit integer (INT4) matrices from the Ampere architecture. These integer Tensor Cores can achieve a theoretical peak performance that is $2 \sim 4$ times faster than floating-point Tensor Cores. Other processors, such as Google TPU v1 [18], Intel AMX-INT8 [5], and Graq TSP [2], also support the multiplication of INT8 matrices with INT32 accumulation. In light of these advancements, our research aims to investigate whether these processors can be leveraged for other high-performance computing (HPC) applications.

Recent research has explored the use of machine learning processors for HPC applications, in particular low- and mixed-precision floating-point matrix multiplication units (FMMU). For instance, Haidar et al. used the FP16 Tensor Core to solve FP64 linear equations with an iterative refinement method [10]. Finkelstein et al. have employed FP16 Tensor Cores to perform time-independent quantum response calculations with sufficient accuracy [7]. Our previous study has utilized FP16 Tensor Cores to improve the throughput of quantum circuit simulation by emulating single-precision matrix multiplication using an error correction method and avoiding rounding inside Tensor Cores without loss of accuracy [27, 28]. However, despite the promising results obtained by these studies, few have explored the potential of integer matrix multiplication units for HPC applications. This paper investigates the feasibility of leveraging integer matrix multiplication units, for instance, INT8 Tensor Cores in NVIDIA GPUs, for HPC applications.

Even before the advent of deep learning and specialized hardware for it, researchers had been exploring ways to perform high-precision matrix multiplication on lower-precision computing units. One such approach is the double-double and quad-double precision, where a single value is represented as the sum of two and four FP64 values, respectively, and arithmetic operations are performed using a sequence of FP64 operations [13]. GEMM and other BLAS functions for double-double precision have been evaluated on NVIDIA GPUs and AMD Cypress GPUs [22, 24]. Another approach is the Ozaki scheme, which also splits a value into multiple lower-precision values [29, 30]. In this scheme, input matrices are split into multiple matrix slices, and the resulting matrix is obtained by accumulating the multiplication of these slices. The key

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/6…$15.00
https://doi.org/XXXXXXX.XXXXXXX

arXiv:2306.11975v4 [cs.DC] 30 Mar 2024



Ozaki Scheme II:
A GEMM-oriented emulation of floating-point matrix multiplication using an integer modular technique

Journal Title
XX(X):1–13
©The Author(s) 2016
Reprints and permission:
sagepub.co.uk/JournalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Katsuhisa Ozaki[1], Yuki Uchino[2], and Toshiyuki Imamura[2]

Abstract

This paper addresses emulation algorithms for matrix multiplication. General Matrix-Matrix Multiplication (GEMM), a fundamental operation in the Basic Linear Algebra Subprograms (BLAS), is typically optimized for specific hardware architectures. The Ozaki scheme is a well-established GEMM-based emulation method for matrix multiplication, wherein input matrices are decomposed into several low-precision components to ensure that the resulting matrix product is computed exactly through numerical operations. This study proposes a novel GEMM-based emulation method for matrix multiplication that leverages the Chinese Remainder Theorem. The proposed method inherits the computational efficiency of highly optimized GEMM routines and further enables control over the number of matrix multiplications, which can enhance computational accuracy. We present numerical experiments featuring INT8 Tensor Core operations on GPUs and FP64 arithmetic on CPUs as case studies. The results demonstrate that FP64 emulation using the proposed method achieves performance levels of up to 7.4 to 9.8 TFLOPS on the NVIDIA RTX 4090 and 56.6 to 80.2 TFLOPS on the NVIDIA GH200, exceeding the measured performance of native FP64 arithmetic. Furthermore, for FP64 computations on CPUs, the proposed method achieved up to a 2.3x speedup in emulating quadruple-precision arithmetic compared to the conventional Ozaki scheme.

Keywords

matrix multiplication, floating-point arithmetic, matrix engines, high-precision emulation

1 Introduction

This paper discusses numerical methods for emulating high-precision matrix multiplication. When the accuracy of the computed results is insufficient, multiple-precision arithmetic provides a viable alternative (Higham 2018). For such computations, high-precision datatypes, such as the built-in float128.1 in the C++23 standard, and multiple-precision arithmetic libraries, such as MPFR (Fousse et al. 2025), offer robust and reliable functionality. In the domain of linear algebra, MPLAPACK (Nakata 2022) is available for high-precision problem solving. Alternatively, when it is unnecessary to extend the exponent range of floating-point numbers and only the significand requires pseudo-extension, *multiple-component arithmetic* provides an efficient solution. [illegible] quad-word arithmetic [illegible] as well as triple-word

When the computational task is limited to matrix multiplication, the Ozaki scheme (Ozaki et al. 2012, 2013) is recognized as a highly reliable method. It achieves high accuracy by leveraging standard floating-point operations and high performance by exploiting optimized routines, e.g., General Matrix-Matrix Multiplication (GEMM), in Basic Linear Algebra Subprograms (BLAS).

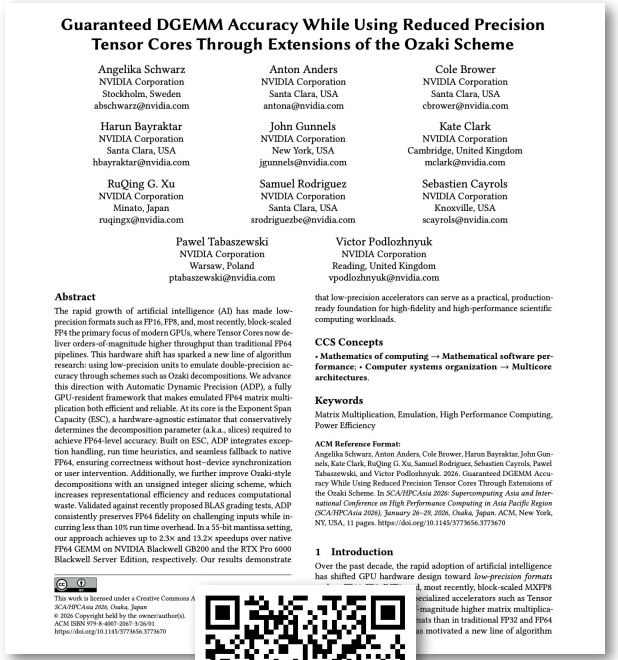
In recent years, increasing attention has been paid to matrix engines optimized for low-precision arithmetic. From the perspective of power efficiency, mixed-precision computation utilizing low-precision formats has also

[1]Department of Mathematical Sciences, Shibaura Institute of Technology, Japan
[2]RIKEN Center for Computational Science, Japan

Corresponding author:
[illegible]uma-ku, Saitama-shi, Saitama 337-

Prepared using sagej.cls [illegible]

arXiv:2504.08009v3 [cs.MS] 27 Apr 2025



Guaranteed DGEMM Accuracy While Using Reduced Precision Tensor Cores Through Extensions of the Ozaki Scheme

Angelika Schwarz
NVIDIA Corporation
Stockholm, Sweden
abschwarz@nvidia.com

Anton Anders
NVIDIA Corporation
Santa Clara, USA
antona@nvidia.com

Cole Brower
NVIDIA Corporation
Santa Clara, USA
cbrower@nvidia.com

Harun Bayraktar
NVIDIA Corporation
Santa Clara, USA
hbayraktar@nvidia.com

John Gunnels
NVIDIA Corporation
New York, USA
jgunnels@nvidia.com

Kate Clark
NVIDIA Corporation
Cambridge, United Kingdom
mclark@nvidia.com

RuQing G. Xu
NVIDIA Corporation
Minato, Japan
ruqingx@nvidia.com

Samuel Rodriguez
NVIDIA Corporation
Santa Clara, USA
srodriguezbe@nvidia.com

Sebastien Cayrols
NVIDIA Corporation
Knoxville, USA
scayrols@nvidia.com

Pawel Tabaszewski
NVIDIA Corporation
Warsaw, Poland
ptabaszewski@nvidia.com

Victor Podlozhnyuk
NVIDIA Corporation
Reading, United Kingdom
vpodlozhnyuk@nvidia.com

Abstract

The rapid growth of artificial intelligence (AI) has made low-precision formats such as FP16, FP8, and, most recently, block-scaled FP4 the primary focus of modern GPUs, where Tensor Cores now deliver orders-of-magnitude higher throughput than traditional FP64 pipelines. This hardware shift has sparked a new line of algorithm research: using low-precision units to emulate double-precision accuracy through schemes such as Ozaki decompositions. We advance this direction with Automatic Dynamic Precision (ADP), a fully GPU-resident framework that makes emulated FP64 matrix multiplication both efficient and reliable. At its core is the Exponent Span Capacity (ESC), a hardware-agnostic estimator that conservatively determines the decomposition parameter (a.k.a., slices) required to achieve FP64-level accuracy. Built on ESC, ADP integrates exception handling, run time heuristics, and seamless fallback to native FP64, ensuring correctness without host–device synchronization or user intervention. Additionally, we further improve Ozaki-style decompositions with an unsigned integer slicing scheme, which increases representational efficiency and reduces computational waste. Validated against recently proposed BLAS grading tests, ADP consistently preserves FP64 fidelity on challenging inputs while incurring less than 10% run time overhead. In a 55-bit mantissa setting, our approach achieves up to 2.3× and 13.2× speedups over native FP64 GEMM on NVIDIA Blackwell GB200 and the RTX Pro 6000 Blackwell Server Edition, respectively. Our results demonstrate that low-precision accelerators can serve as a practical, production-ready foundation for high-fidelity and high-performance scientific computing workloads.

CCS Concepts

• Mathematics of computing → Mathematical software performance; • Computer systems organization → Multicore architectures.

Keywords

Matrix Multiplication, Emulation, High Performance Computing, Power Efficiency

ACM Reference Format:

Angelika Schwarz, Anton Anders, Cole Brower, Harun Bayraktar, John Gunnels, Kate Clark, RuQing G. Xu, Samuel Rodriguez, Sebastien Cayrols, Pawel Tabaszewski, and Victor Podlozhnyuk. 2026. Guaranteed DGEMM Accuracy While Using Reduced Precision Tensor Cores Through Extensions of the Ozaki Scheme. In *SCA/HPCAsia 2026: Supercomputing Asia and International Conference on High Performance Computing in Asia Pacific Region (SCA/HPCAsia 2026), January 26–29, 2026, Osaka, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3773656.3773670

1 Introduction

Over the past decade, the rapid adoption of artificial intelligence has shifted GPU hardware design toward *low-precision formats* [illegible], most recently, block-scaled MXFP8 [illegible]specialized accelerators such as Tensor [illegible]-magnitude higher matrix multiplica[illegible]nats than in traditional FP32 and FP64 [illegible]s motivated a new line of algorithm

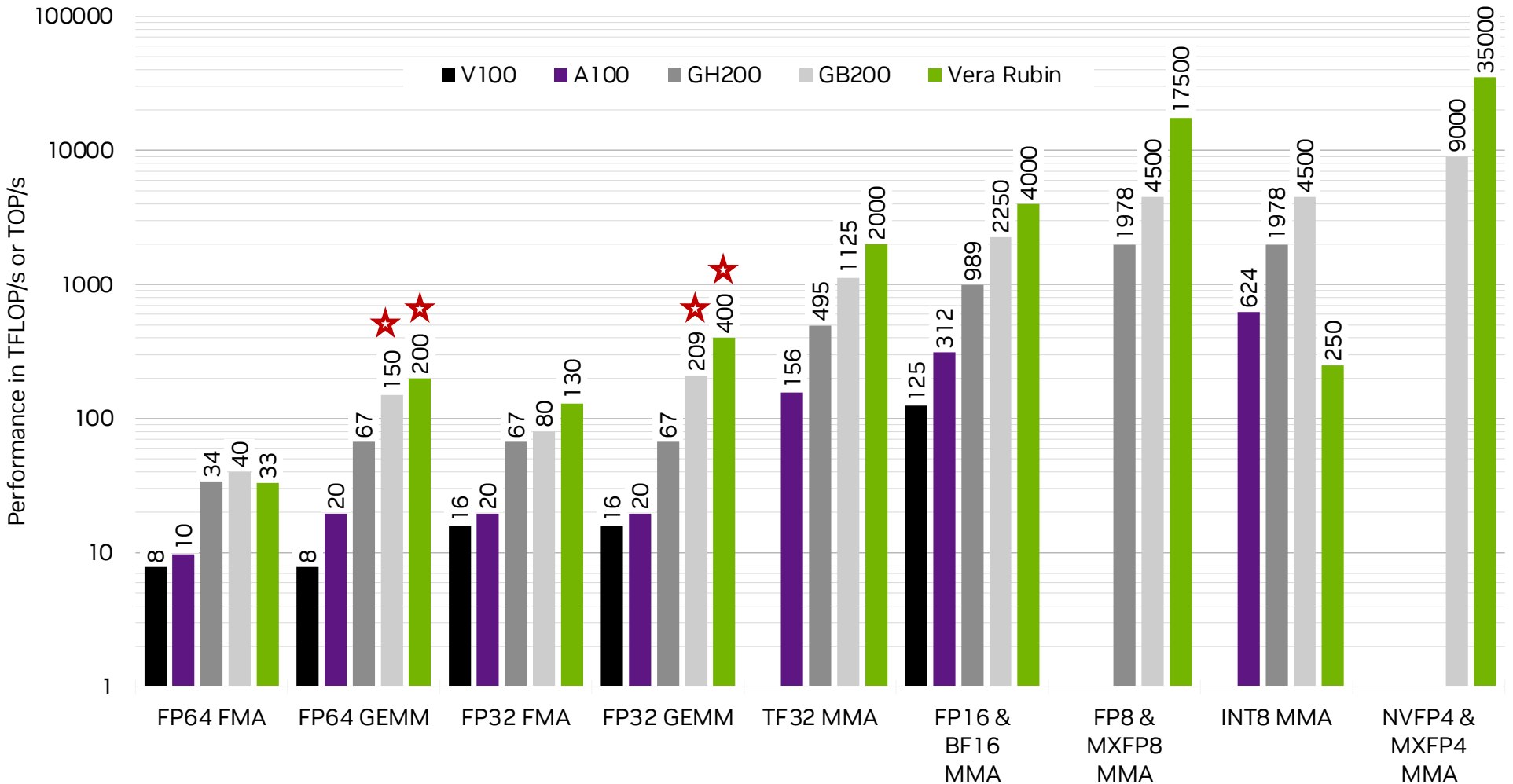
[illegible] This work is licensed under a Creative Commons A[illegible]
SCA/HPCAsia 2026, Osaka, Japan
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2367-3/26/01
https://doi.org/10.1145/3773656.3773670

Productization in Libraries like cuBLAS, cuSOLVER, ...

NVIDIA

Evolution of Peak Performance

Hardware Specs & Emulation



NVIDIA

Five Perceived Challenges With Ozaki Scheme

Hampering adoption by some in Scientific Computing

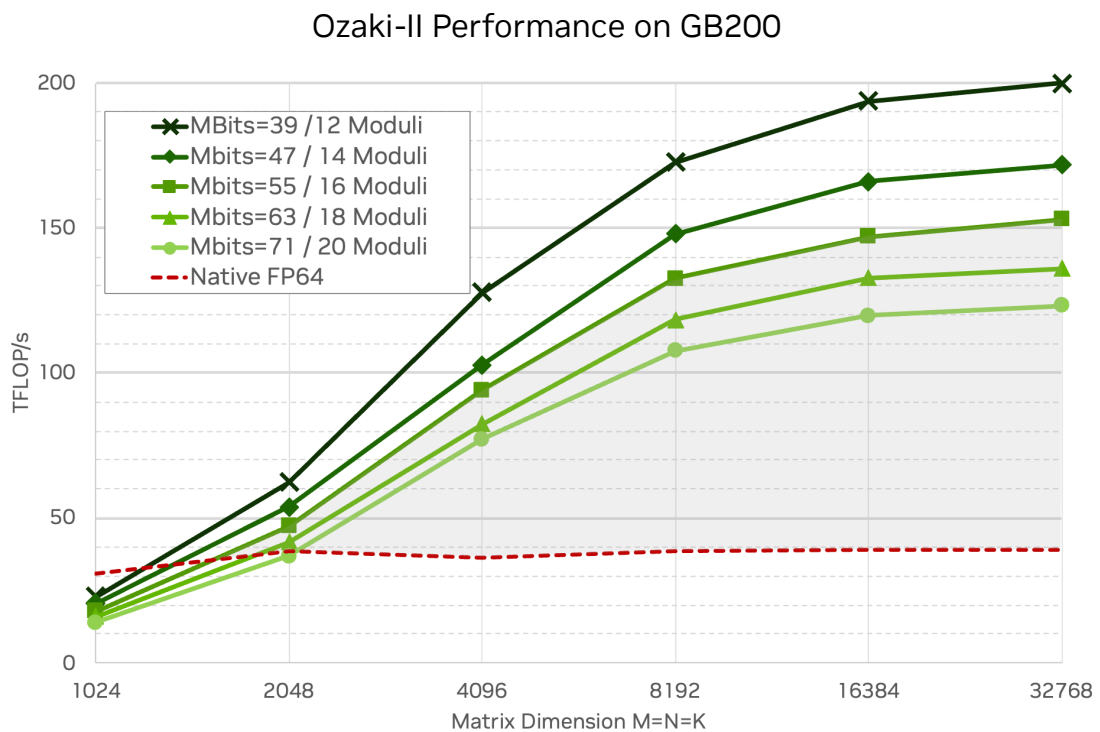
1. Emulation doesn’t work for my problem; too many slices/moduli are needed
2. Results are not as accurate as IEEE FP64
3. Emulation uses too much memory
4. Emulation doesn’t deliver performance for non-square matrices
5. Emulation doesn’t deliver performance for small matrices

“Emulation doesn’t work for my problem; too many slices/moduli are needed.”

- Common sentiment echoed by many computational scientists

Ozaki-II Performance

Does emulated DGEMM behave like FP64 as the problem gets harder? Yes, but at a price

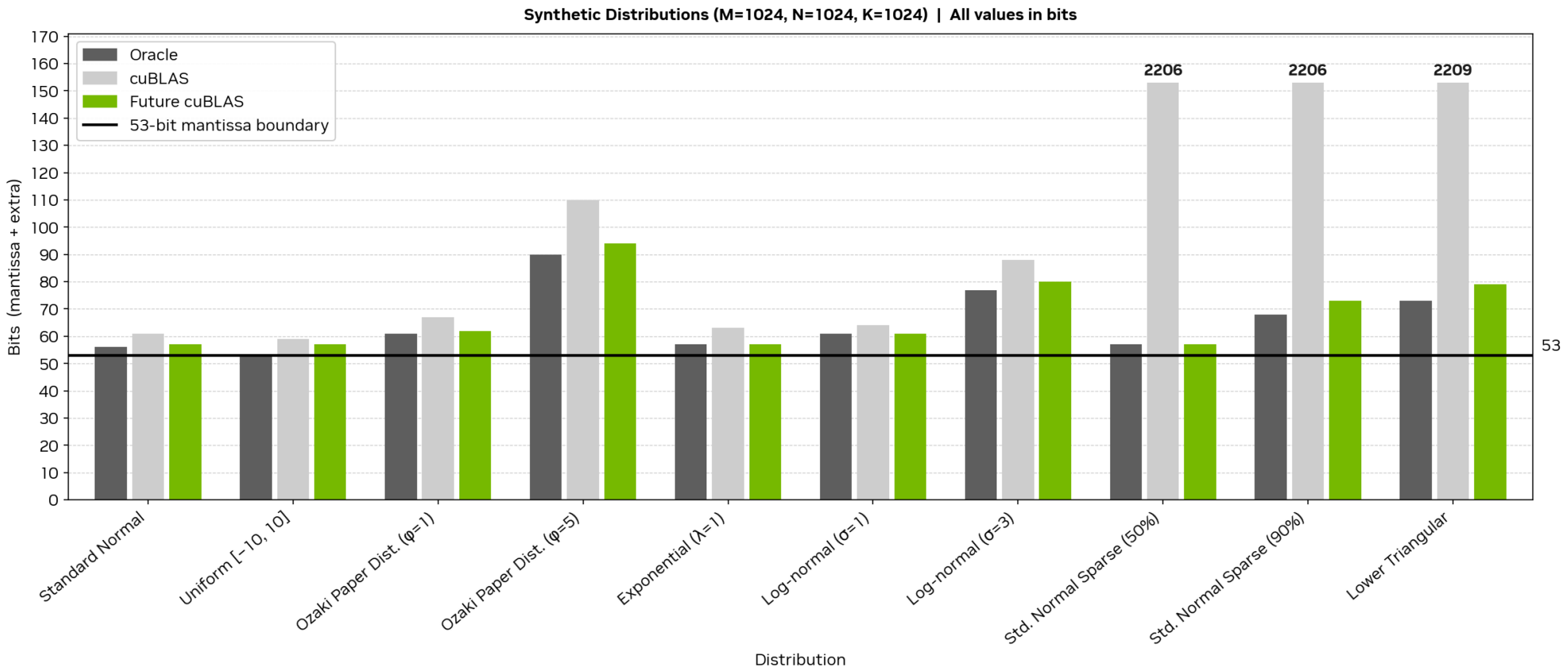


The current ESC algorithm is conservative with a loose bound and for some input data it is overly conservative, leading to poor speed-ups or fallback to native FP64

Improving the Exponent Span Capacity (ESC) Algorithm

The current ESC in cuBLAS has very loose bound (too conservative)

We have a new ESC algorithm we will productize later in 2026 that delivers significant improvements (details of the new algorithm will be published when ready as was done with the current version)



Preliminary data | Data subject to change

“Emulation results are not as accurate as IEEE FP64.”

- Common sentiment echoed by many computational scientists

IEEE 754 Compliance of Matmul Emulation

Current Limitations

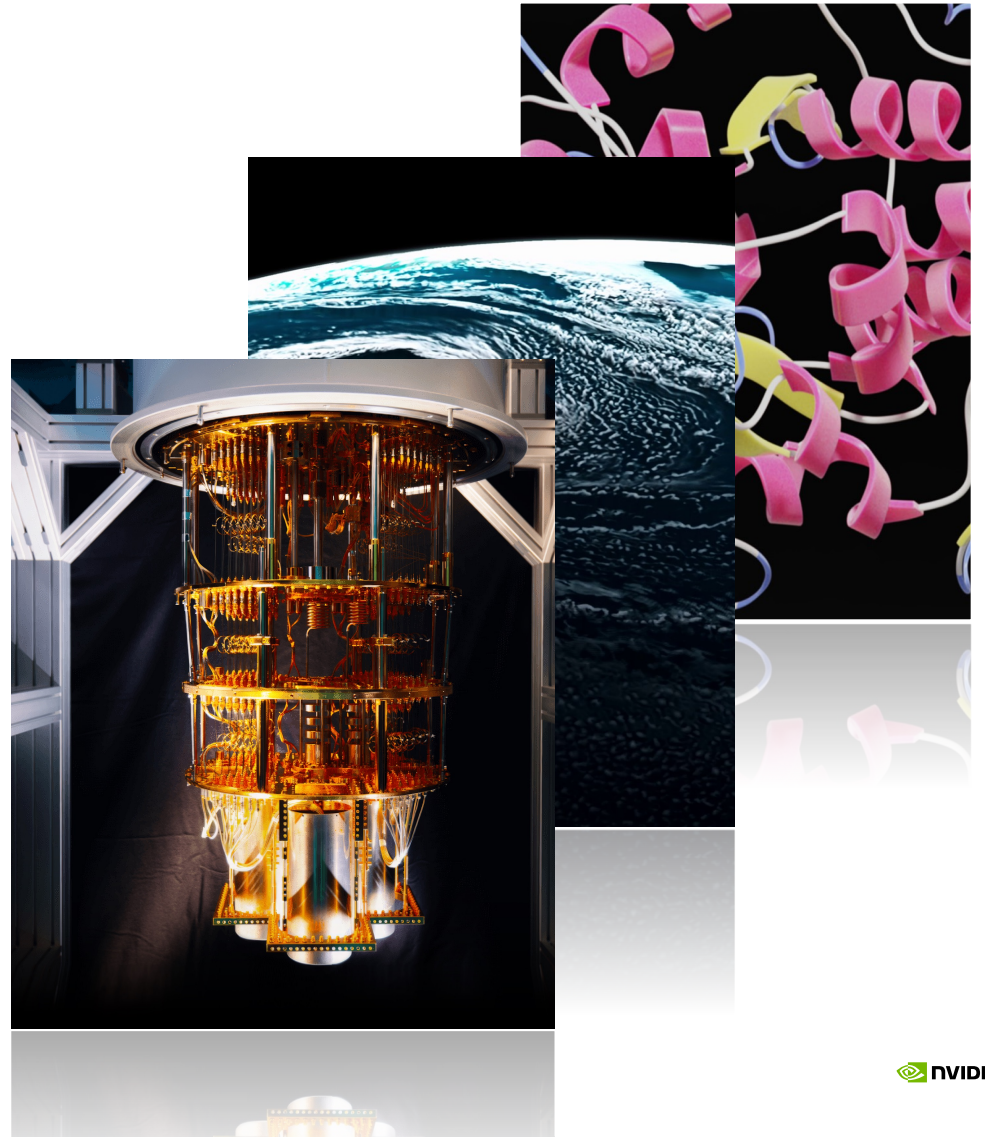
NaN/Inf/+0,-0

- We can detect if these are present on the input matrices and fall back to FP64 native
- We can NOT detect nor propagate these when they happen during the dot product (not necessarily a bad thing)
- We treat +0 and -0 as the same value

We do not observe strict IEEE compliance

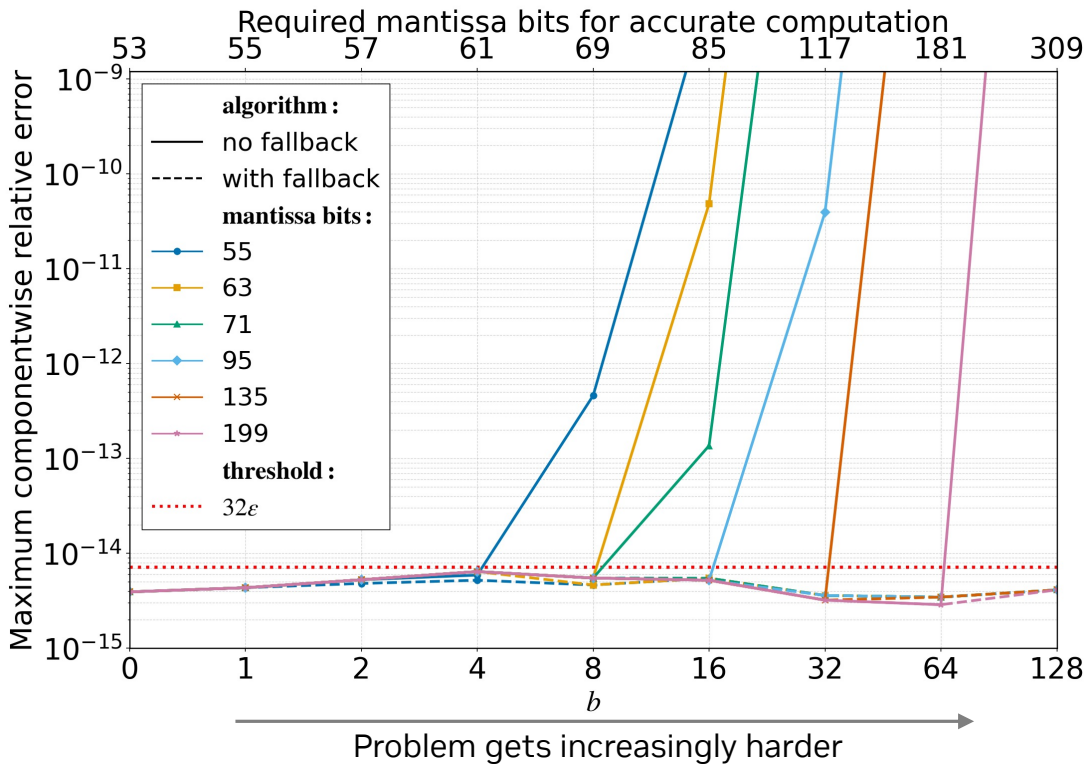
- Since exponents are coupled with results only at certain points (number of times depends on k dimension of matrix)
- Because we store only one exponent per row/column of the matrix and do not check for overflow at a scalar product level, we can generate NaNs and ”sticky” +/-Infs, but not at the IEEE-prescribed granularity

These limitations don’t speak to the accuracy of the results which are FP64 level accurate



Putting ESC to the Test

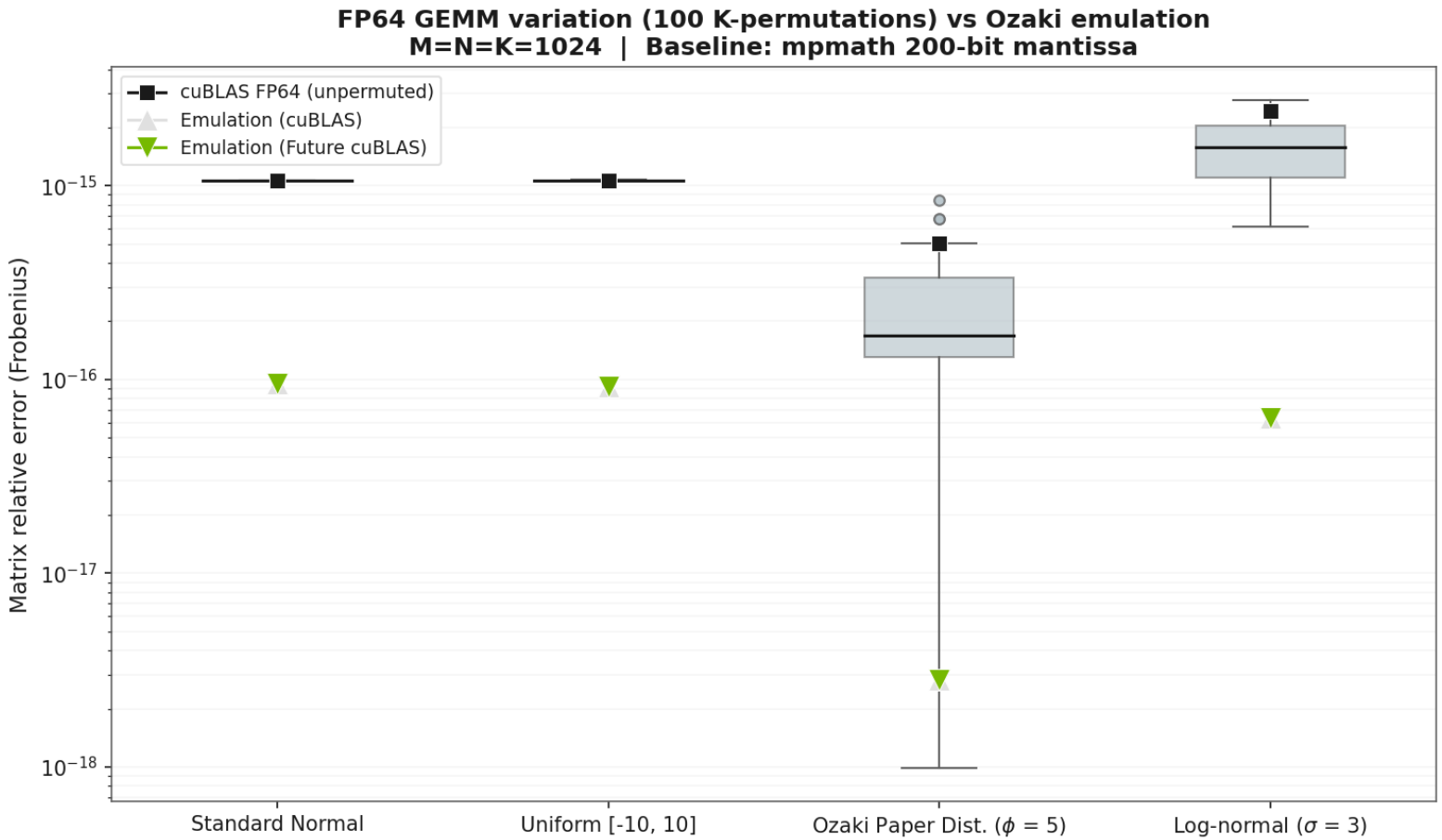
Does emulated DGEMM behave like FP64 as the problem gets harder? Yes, but at a price



The current ESC algorithm is conservative with a loose bound and for some input data it is overly conservative, leading to poor speed-ups or fallback to native FP64

Putting the new ESC to the Test

Does emulated DGEMM behave like FP64 as the problem gets harder? Yes, and will deliver much better performance



Preliminary data | Data subject to change

“Emulation of FP64 matrix multiplications on GPUs consumes up to 2x more memory, which reduces the problems that can be solved.”

- Common sentiment echoed by many computational scientists

Impact of Available Memory on DGEMM Performance

Performance Ratio Using cuBLAS (8GB : Unlimited Memory Tiling) on the B200 GPU

The new blocking strategy using the limited memory policy leads to noticeable performance uplift in some cases

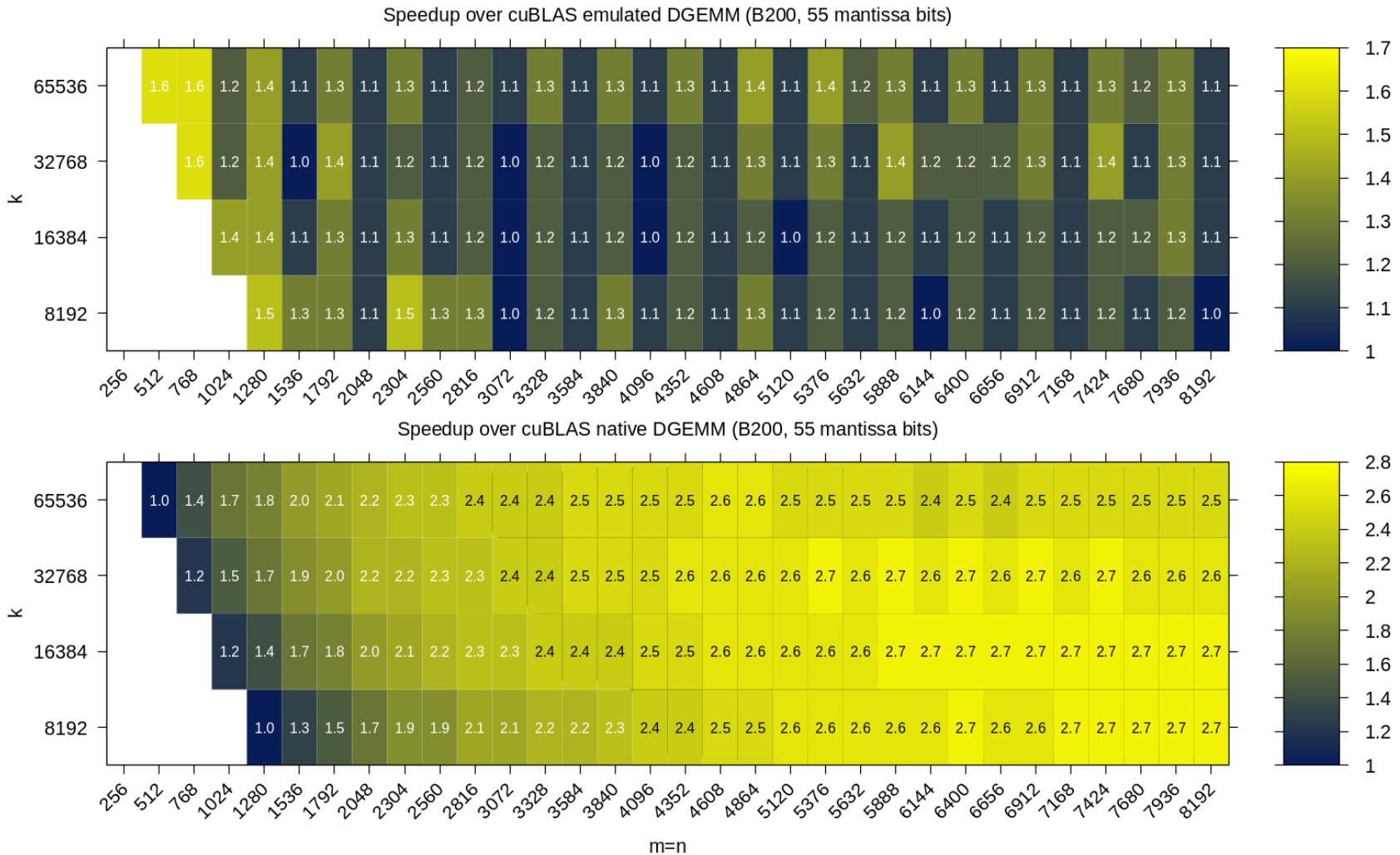
Coming soon in a 2026 cuBLAS release

“Emulation doesn’t deliver performance for non-square matrices.”

- Common sentiment echoed by many computational scientists

Emulation Performance for Non-Square Matrices (Ozaki-I)

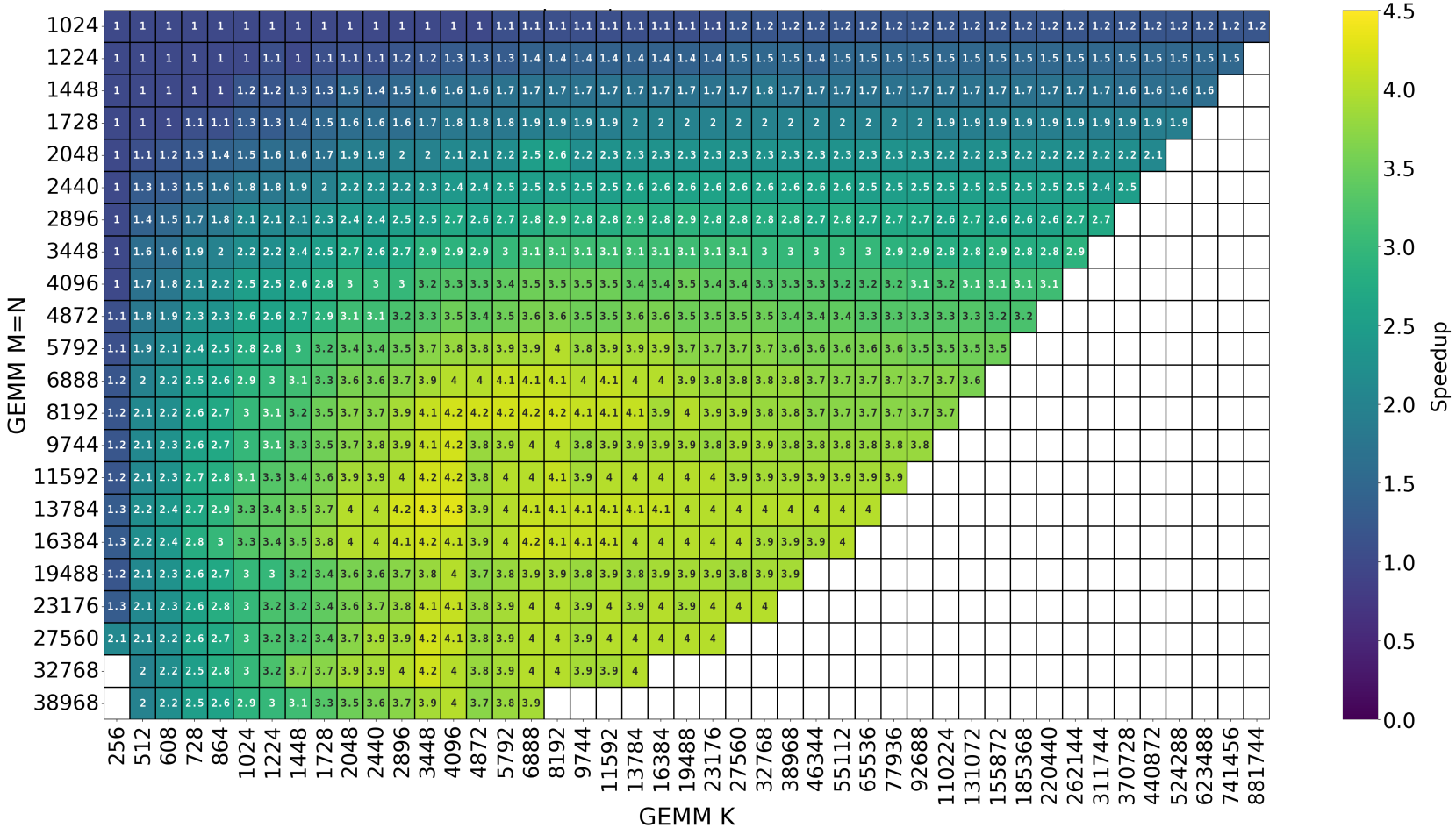
Overheads plus memory vs compute bound regime pose challenges



Upcoming releases will continue to deliver improvements

Performance of DGEMM on GB200 GPUs

Speedup of Ozaki-II Over Native FP64 (Coming to cuBLAS Later in 2026)

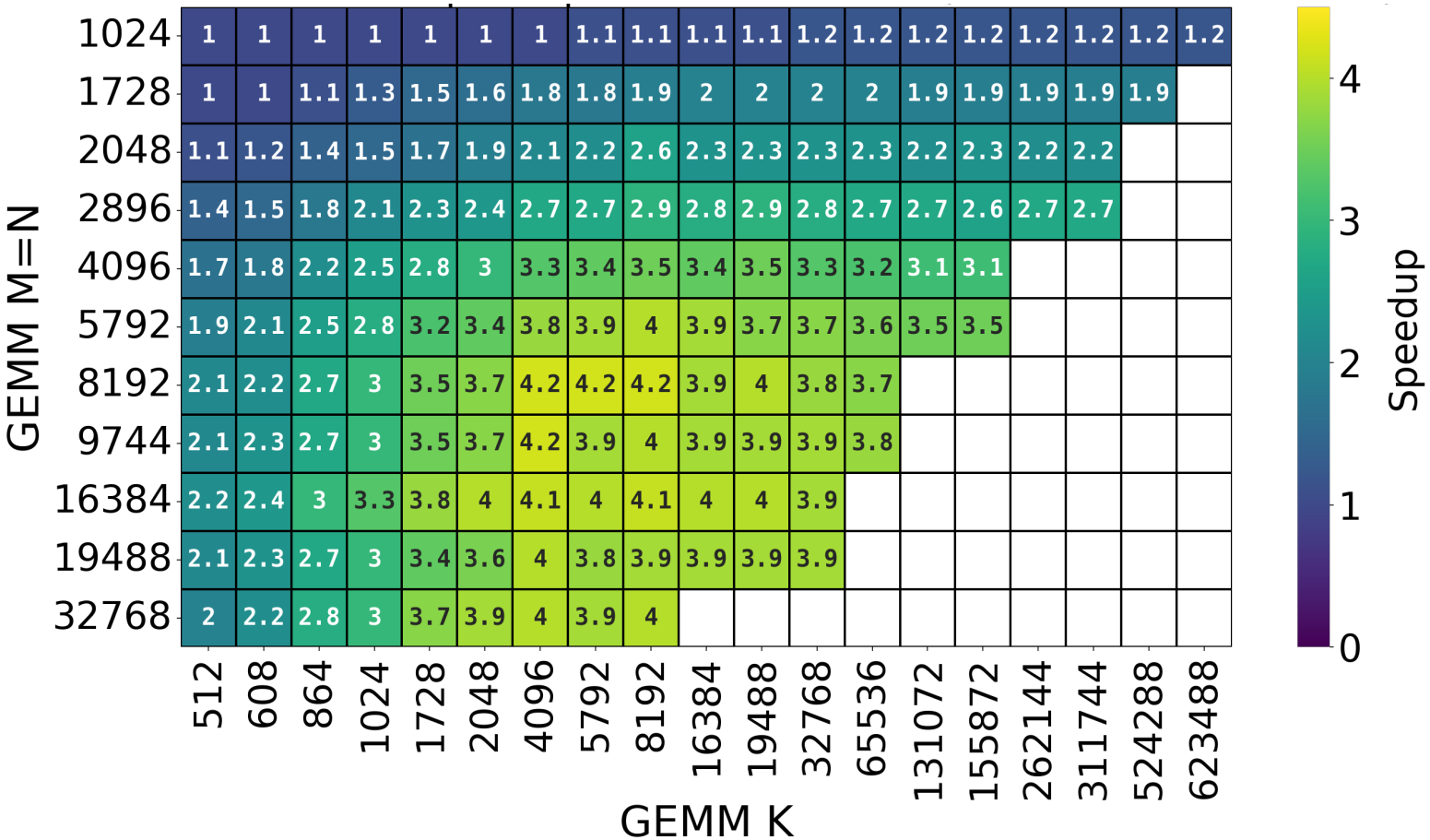


Preliminary data | Data subject to change

NVIDIA

Performance of DGEMM on GB200 GPUs

Speedup of Ozaki-II Over Native FP64 (Coming to cuBLAS Later in 2026)



Preliminary data | Data subject to change

NVIDIA

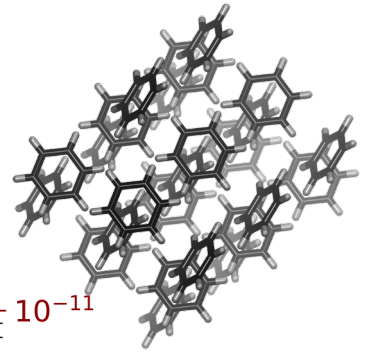
Five Perceived Challenges With Ozaki Schemes

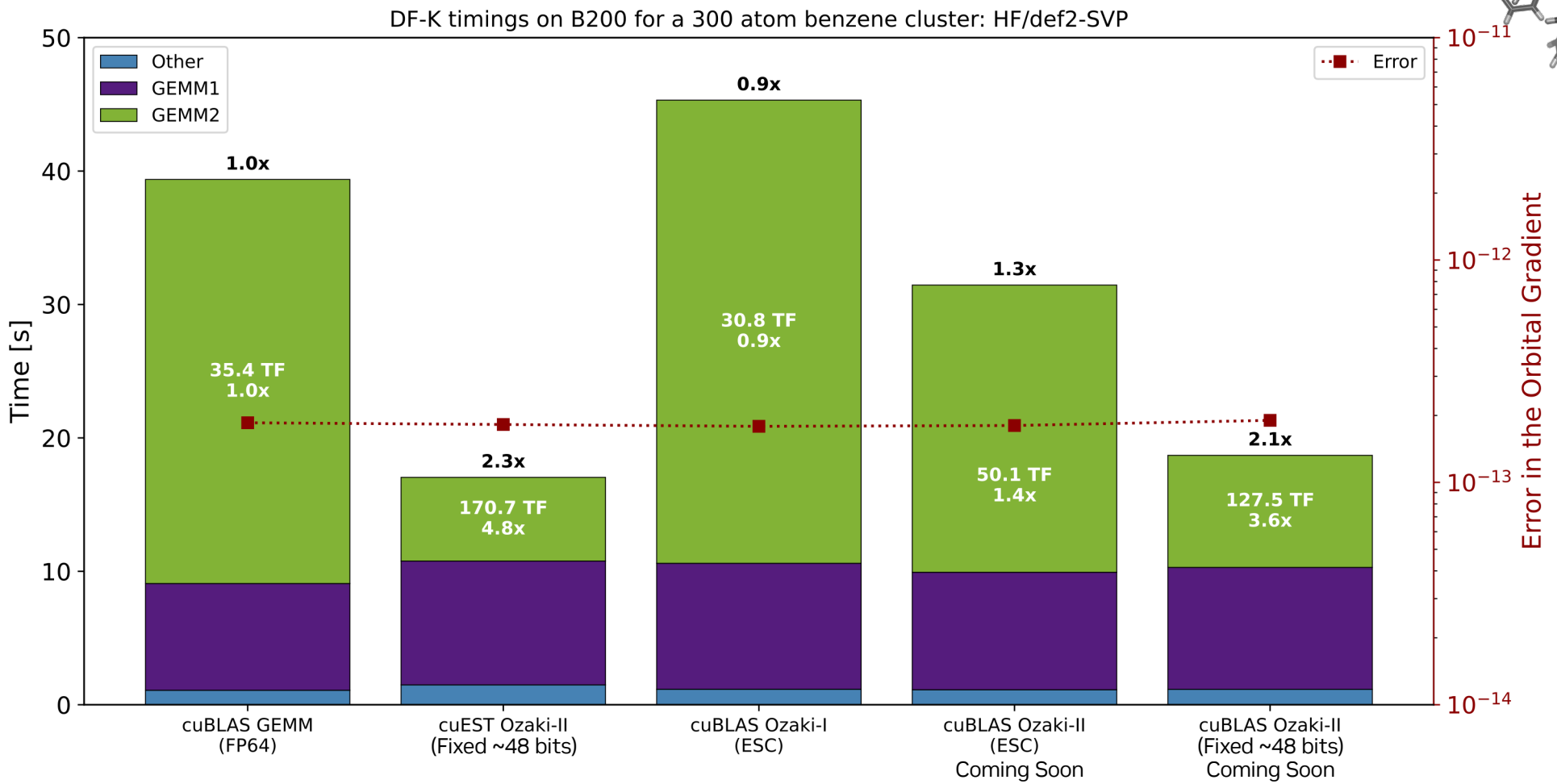
Hampering adoption by some in Scientific Computing

1. Emulation doesn't work for my problem, too many slices/moduli are needed **(coming soon)**
2. Results are not as accurate as IEEE FP64
3. Emulation uses too much memory **(coming very soon)**
4. Emulation doesn't deliver performance for non-square matrices
5. Emulation doesn't deliver performance for small matrices ← **Likely requires new APIs**

Impact of Improvements on Quantum Chemistry

cuEST library using the Ozaki-II New ESC Algorithm in cuBLAS





Preliminary data | Data subject to change

NVIDIA


NVIDIA