

Leveraging Hardware-Aware Computation in Mixed-Precision Matrix Multiply: A Tile-Centric Approach

Qiao Zhang¹, Rabab Alomairy^{2,3}, Dali Wang⁴, Zhuowei Gu¹, and Qinglei Cao¹

¹ Saint Louis University, USA

² Massachusetts Institute of Technology, USA

³ King Abdullah University of Science and Technology, KSA

⁴ Oak Ridge National Laboratory, USA

Abstract. General Matrix Multiplication (GEMM) is a critical operation underpinning a wide range of applications in high-performance computing (HPC) and artificial intelligence (AI). The emergence of hardware optimized for low-precision arithmetic necessitates a reevaluation of numerical algorithms to leverage mixed-precision computations, achieving improved performance and energy efficiency. This research introduces an adaptive mixed-precision GEMM framework that supports different precision formats at fine-grained tile/block levels. We utilize the PaRSEC runtime system to balance workloads across various architectures. The performance scales well on ARM CPU-based Fugaku supercomputer, Nvidia GPU-based A100 DGX, and AMD GPU-based Frontier supercomputer. This research aims to enhance computational efficiency and accuracy by bridging algorithmic advancements and hardware innovations, driving transformative progress in various applications.

Keywords: High-performance computing · Task-based runtime · General matrix multiply · Mixed precision.

1 Introduction

General Matrix Multiplication (GEMM) lies at the core of linear algebra and acts as a critical kernel in numerous High-Performance Computing (HPC) applications—such as earthquake simulation [1] and weather and climate prediction [2]—and in Artificial Intelligence (AI) applications, including fully connected layers in traditional neural networks and natural language processing [3]. GEMM is a computationally expensive operation due to its high demands on both memory and computation, whose performance directly affects the efficiency of these applications, making it a focal point for optimization on diverse architectures, ranging from homogeneous CPU clusters to heterogeneous GPU systems.

Over the last ten years, hardware designers have prioritized crafting processors that excel in high-speed, energy-efficient, and reduced-precision computations [4]. A notable example is the widespread use of Nvidia GPUs in both HPC and AI tasks, driven by their exceptional processing capabilities and efficiency.

The latest Nvidia GPUs, including the H100 and the upcoming B100, incorporate a range of precision formats, each influencing peak theoretical throughput differently. The lowest precision levels (FP8/INT8 on H100 and FP4 on B100) achieve speedups of $58\times$ and $233\times$, respectively, relative to FP64 computations, underscoring the dramatic acceleration possible when shifting from conventional double-precision arithmetic to Tensor Core-accelerated lower-precision execution. A similar evolution is observed in CPUs, where FP64 performance is typically double that of FP32 and quadruple that of FP16. This transition has motivated researchers to reassess classical numerical techniques, identifying areas where reduced precision is sufficient without compromising the overall quality of solutions. Additionally, leveraging lower precision not only diminishes memory demands—allowing more data to fit within high-speed memory and thereby improving execution rates [5, 6]—but also cuts down on energy consumption, making it a compelling strategy for boosting energy efficiency in computational workflows [7].

The synergy between hardware advancements and algorithmic refinements plays a pivotal role in tackling complex computational challenges. Mixed-precision computations, which involve utilizing multiple numerical precisions within a single task, have emerged as a powerful strategy [8, 9]. This methodology has been successfully integrated into numerous scientific and engineering disciplines, such as deep learning [10], computational fluid dynamics [11, 12], astronomy [13], climate modeling [14], molecular dynamics [15], and quantum mechanics [16]. The primary advantage of these techniques is their ability to harness lower-precision hardware capabilities to enhance computational speed while preserving necessary accuracy constraints. Nevertheless, most existing implementations rely on coarse-grained precision adjustments and are optimized for single-GPU or single-node setups, which limits their applicability to large-scale distributed environments. Efficiently handling mixed-precision computations in distributed multi-GPU architectures introduces significant hurdles due to disparities in data representation and precision consistency. Moreover, many existing solutions are tailored to specific hardware platforms, reducing their portability across diverse computing architectures.

This paper introduces a mixed-precision GEMM framework (GEMM-MP) at fine-grained tile/block levels and integrates GEMM-MP into the runtime system PaRSEC. The contributions of this paper are as follows:

- Introducing a tile-centric mixed-precision GEMM algorithm that carefully balances accuracy and performance, enabling a robust and trustworthy mixed-precision solution for HPC and AI applications.
- Leveraging a dynamic task-based runtime system to efficiently orchestrate heterogeneous tasks, dynamically adapting to workload variations while minimizing scheduling overhead and maximizing resource utilization.
- Demonstrating strong scalability and portability across a diverse range of homogeneous and heterogeneous computing platforms, ensuring broad applicability and efficiency.

To the best of our knowledge, this is the first work to introduce a tile-centric mixed-precision GEMM algorithm.

The structure of this paper is as follows. Section 2 discusses prior research relevant to our work. In Section 3, we provide an overview of PaRSEC. The tile-centric mixed-precision GEMM framework is described in Section 4. A performance evaluation is then conducted in Section 5, and finally, Section 6 outlines our conclusions and future directions.

2 Related Work

2.1 Mixed-Precision Matrix Computations

Mixed-precision approaches have garnered extensive interest [8,9] and are widely applied across multiple computational fields [10–16]. A well-known example is iterative refinement techniques for linear solvers, where the system is initially solved using lower precision, followed by higher precision corrections applied iteratively to refine the residual. This strategy has demonstrated significant reductions in computational time and energy consumption, making it an effective approach for enhancing efficiency in large-scale computations [7,17]. However, the effectiveness of such methods is highly dependent on the matrix condition number, and the requirement to store multiple precision copies of matrices can introduce substantial memory overhead.

Ongoing research explores performance optimization while ensuring numerical reliability by selectively employing reduced precision in non-critical computations while preserving higher precision where necessary [2,18,19]. These methodologies primarily target linear system solutions. In deep learning, reduced precision is typically applied to forward and backward computations, whereas weight updates often require higher precision accumulation to maintain numerical stability. This trade-off, however, may lead to accuracy degradation due to precision loss in key computations of forward and backward propagation. In this work, we introduce an adaptive tile-centric mixed-precision GEMM framework that dynamically adjusts precision levels within a single computation. This fine-grained approach can provide another solution to ensure that accuracy remains uncompromised by recognizing that not all computational components require the same precision level in HPC and applications.

2.2 Runtime Systems

Recent advancements in task-based programming models have driven the scalability of scientific applications on modern supercomputers. A variety of runtime systems have been developed to facilitate efficient task execution, optimizing resource utilization across diverse architectures. OpenMP [20] has emerged as a widely adopted standard for shared-memory parallelism, incorporating tasking constructs that provide dynamic scheduling and fine-grained control. By leveraging compiler directives and runtime support, OpenMP simplifies concurrency

management and enhances performance in multi-threaded environments. Building upon OpenMP, OmpSs [21] extends capabilities for heterogeneous architectures, introducing advanced dependency tracking and asynchronous execution strategies. Similarly, COMPSs [22] provides a comprehensive programming environment, streamlining task parallelism across distributed computing infrastructures. For heterogeneous, distributed computing platforms, StarPU [23] delivers a flexible runtime system that integrates scheduling mechanisms with explicit kernel annotations, optimizing execution across CPUs and GPUs. Meanwhile, HPX [24] leverages the ParalleX model to support highly scalable applications through lightweight threading and asynchronous execution paradigms in modern C++. Another prominent system, Legion [25], emphasizes task and data partitioning, enabling automatic parallelism detection and improved locality optimizations. By decoupling computation from hardware mapping, Legion achieves high scalability for complex workloads in distributed and heterogeneous environments. This work utilizes PaRSEC, which is described in the following section.

3 The PaRSEC Runtime System

PaRSEC [26,27] serves as a generic framework designed for scheduling and managing micro-tasks on distributed, multi-core, and heterogeneous architectures. It integrates a runtime environment, multiple programming models for defining task graphs—applied in dense linear algebra computations [28] and irregular workloads such as mixed-precision techniques [6], low-rank factorizations [29], and sparse operations [30]—along with auxiliary libraries that ease the transition from legacy implementations. Additionally, it incorporates performance analysis utilities [31] to assist in tuning applications for large-scale heterogeneous platforms. PaRSEC provides users with a collection of Domain-Specific Languages (DSLs), such as the Parameterized Task Graph (PTG) [32], Template Task Graph (TTG) [33], and Dynamic Task Discovery (DTD) [34], to enhance flexibility in algorithmic expression. Among these, PTG encodes the entire DAG in a compact, parameterized format, enabling optimizations such as collective communication reuse and separation of data distribution from execution. This abstraction aligns with the core methodology of our research on adaptive mixed-precision computations, and we employ PTG in this work.

4 Algorithm Descriptions

The General Matrix-Matrix Multiplication (GEMM) operation computes the product of two matrices, optionally scaled and accumulated with a third matrix, and is defined as: $C \leftarrow \alpha AB + \beta C$, where A and B are input matrices, C is the output matrix, and α, β are scalar coefficients. Algorithm 1 describes the sequential process of the adaptive mixed-precision GEMM, where the input matrices $A(i, l)_{m \times k}$, $B(l, j)_{k \times n}$, and the output matrix $C(i, j)_{m \times n}$ are partitioned into tiles/blocks of a fixed size. This algorithm follows the SUMMA (Scalable Universal Matrix Multiplication Algorithm) approach [35]. The outermost loop

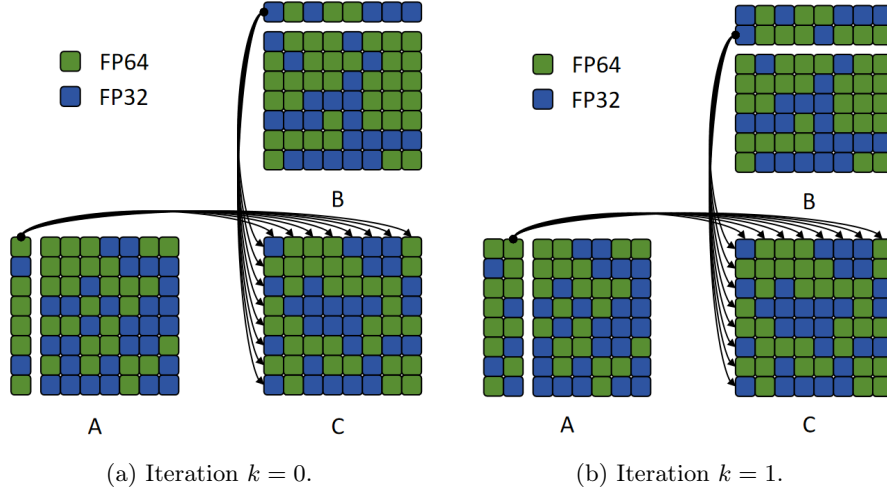


Fig. 1: Demonstration of the first two iterations of the mixed-precision GEMM. Arrows are representative dependencies that introduce communications.

traverses the reduction dimension k , while the two inner loops iterate over the row index i and the column index j . At each iteration, the algorithm updates the value of the matrix C . The notations $\#$, $\$$, and $*$ indicate different data representations or precisions for the operands.

Fig. 1 illustrates task execution across the first two iterations as defined in Algorithm 1. The visualization employs color coding to distinguish numerical precisions: green for double precision (FP64 or DP) and blue for single precision (FP32 or SP). This study focuses on FP64 and FP32, with plans to extend to additional precision formats in future work. Arrows denote representative dependencies arising from data transfer and communication between matrix tiles ($A \rightarrow C$ and $B \rightarrow C$) within a single iteration. Each dependency/communication is associated with a specific datatype. Due to this adaptive tile-centric precision approach, a task may receive data in a precision differing from its operational precision, necessitating potential precision conversion. Various conversion strategies will be explored in subsequent research. This work employs the receiver-side conversion strategy, meaning datatype conversion is handled at the receiving end. That is, the datatype in each data flow is dictated by the precision of the

Algorithm 1: Tile-centric Mixed-Precision GEMM.

```

1 for  $l = 0$  to  $k$ 
2   for  $i = 0$  to  $m$ 
3     for  $j = 0$  to  $n$ 
4        $C^*(i, j) = \alpha A^\#(i, l) \times B^\$(l, j) + \beta C^*(i, j)$ 

```

stored data. For example, if a tile in A or B is stored in FP64, the transmitted data remains in double precision.

This algorithm is integrated into the PaRSEC runtime system, which serves as the underlying framework for execution. By leveraging PaRSEC, we ensure efficient parallel processing of matrix operations while optimizing the trade-offs among computation, communication, and memory usage, due to the imbalanced workload introduced by the adaptive tile-centric mixed-precision algorithm.

5 Performance Results and Analysis

5.1 Experimental Settings

The experiments are conducted on three systems.

- Guyot: It is a GPU-accelerated AMD compute node, which features two EPYC 7742 CPUs, each with 64 cores running at 2.25 GHz, 2,063 GB of main memory, and eight NVIDIA A100-SXM4-80GB GPUs.
- Fugaku: This system is built on ARM architecture and consists of more than 150,000 compute nodes. Each node is powered by a 48-core A64FX CPU, capable of reaching a peak frequency of 2.2 GHz in boost mode and 2.0 GHz in normal mode, and is equipped with 32 GB of HBM2 memory.
- Frontier: This is a GPU-accelerated supercomputer utilizing AMD hardware, comprising 9,408 compute nodes. Each node integrates a 64-core AMD Optimized 3rd Gen EPYC processor alongside four AMD MI250X GPUs, supported by 512 GB of DDR4 memory.

We use the term “a’D:b’S” to represent the percentage of different precision formats, where D/S denotes double/single precision, and a/b represents the percentage of double/single precision, ensuring $a + b = 100$. For data distribution, we employ a 2D block cyclic scheme with a process grid of size $P \times Q$, structured to be as square as possible. On Fugaku, we execute in normal mode due to power restrictions and disable the Sector Cache Optimizations (SCO) due to memory conflicts [18]. When running on Frontier, we use 4 ranks per node. The optimal tile size is empirically determined, set to 1024 on Fugaku, and 2048 on Guyot and Frontier.

5.2 Precision Map of Kernel Execution

Fig. 2 visualizes the precision of numerical kernels executed on each individual tile, as referenced in Fig. 1, by presenting a heatmap under different configurations. Each point represents the precision of a tile in FP64 or FP32, randomly generated in our settings. The three subfigures illustrate different distributions of precision in A , B , and C : (a) 80D:20S, where 80% of the tiles use double precision (FP64) and 20% use single precision (FP32); (b) 50D:50S, where FP64 and FP32 tiles are equally distributed; and (c) 20D:80S, where 20% of the tiles use FP64 and 80% use FP32. These configurations exhibit varying degrees of mixed-precision computation, which will be employed in the subsequent performance measurements.

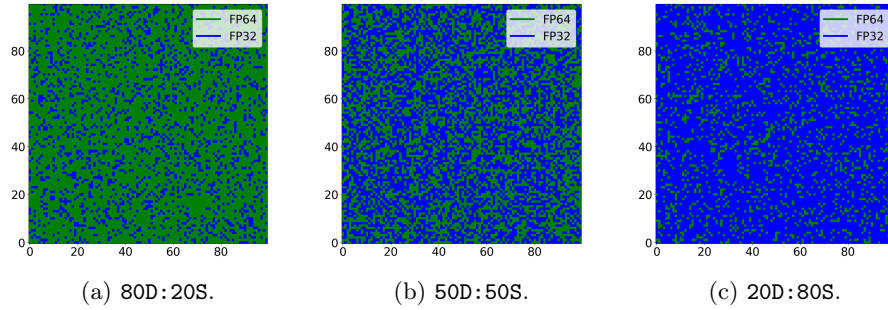


Fig. 2: Kernel precision heatmap for a matrix of size $102,400 \times 102,400$ with a tile size of $1,024 \times 1,024$.

5.3 Shared Memory Performance

Fig. 3 presents the shared-memory performance across different hardware platforms. Each subfigure illustrates performance under various precision settings and the corresponding speedup relative to 100D:0S. Different precision configurations exhibit distinct scaling behaviors, with performance generally improving as the proportion of lower precision increases. (a) Presents performance on a single node of Fugaku. The dashed line represents the practical FP64 peak performance, computed as the GEMM performance per core multiplied by the number of cores. The 100D:0S configuration achieves 84.7% of this practical peak, while 0D:100S doubles it. (b) Depicts results on a single Nvidia A100 GPU on Guyot, where FP64 and FP32 precisions share the same theoretical peak performance. The 100D:0S configuration attains up to 88.9% of the theoretical peak. The decrease in performance with a lower proportion of FP32 may be attributed to increased data movement and/or overheads of datatype conversion. (c) Demonstrates performance on eight Nvidia A100 GPUs on Guyot, exhibiting an almost linear scaling from the performance of a single GPU (see Fig. 3(b)). The 100D:0S configuration reaches 83.9% of the theoretical peak. (d) Evaluates performance on a single node of Frontier. The practical peak performance for 0D:100S on one node is approximately 280.0 Tflop/s, with an achieved performance of about 181.3 Tflop/s, corresponding to 64.8% of the practical peak. The 100D:0S performance lags a little behind, potentially due to the NIC's proximity to the GPU on Frontier and the ongoing development of GPU-direct communication support in PaRSEC.

5.4 Performance Scalability

Fig. 4 presents the distributed-memory performance on 64 nodes of Fugaku and Frontier, further confirming the trends observed in shared-memory performance, as shown in Fig. 3: the achieved performance approaches the theoretical/practical peak, and different precision configurations exhibit distinct scaling behaviors,

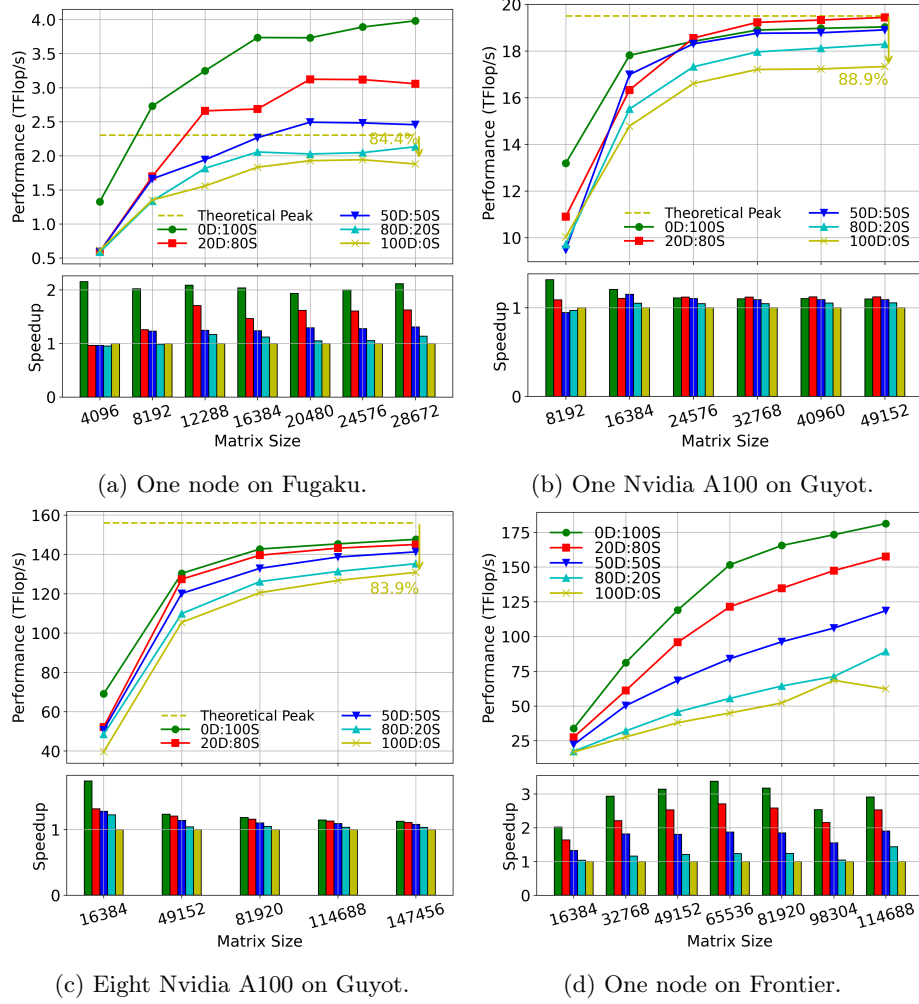


Fig. 3: Shared-memory performance of GEMM-MP.

with performance generally improving as the proportion of lower precision increases. Moreover, GEMM-MP attains nearly linear scaling from shared-memory performance to 64 nodes on both Frontier and Fugaku. For instance, in the 0D:100S configuration, the one-node performance on Fugaku is approximately 4.0 Tflop/s, while the performance on 64 nodes reaches 242.2 Tflop/s with a parallel efficiency of 94.6%. On Frontier, the one-node performance is about 181.3 Tflop/s, increasing to 11,310.3 Tflop/s on 64 nodes with a parallel efficiency of 97.5%. All in all, these results underscore the significance of the proposed tile-centric mixed-precision framework in achieving optimal computational ef-

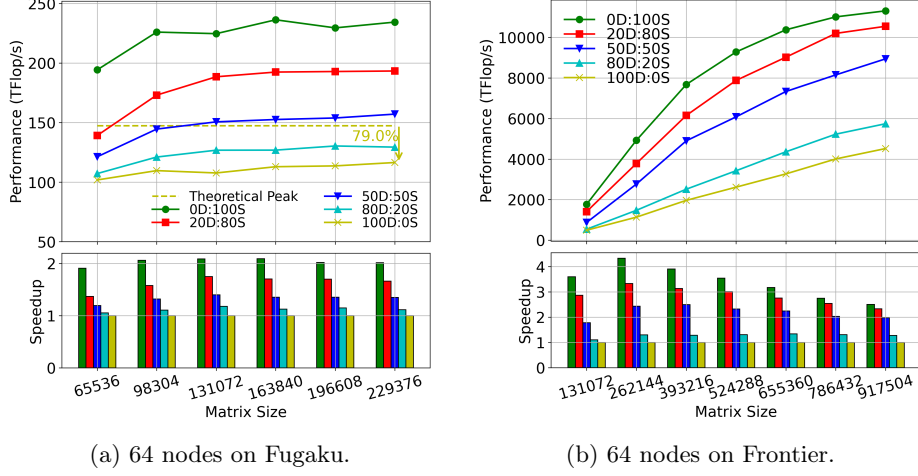


Fig. 4: Distributed-memory performance of GEMM-MP.

efficiency in both shared-memory and distributed-memory environments across various architectures.

6 Conclusion and Future Work

This work introduces GEMM-MP, an adaptive tile-centric mixed-precision GEMM framework. Built on ParSEC, the proposed approach efficiently manages heterogeneous tasks arising from varying precision formats, ensuring high-performance execution across both shared- and distributed-memory environments. GEMM-MP demonstrates strong efficiency and scalability on homogeneous CPU setups and heterogeneous GPU systems, making it highly adaptable to diverse hardware architectures. Going forward, we plan to extend the framework by incorporating additional precision formats and developing trustworthy precision selection strategies for both lossless and lossy approaches. We also aim to investigate different datatype conversion strategies. Furthermore, we seek to integrate this mixed-precision framework with other runtime systems and deploy it in real-world HPC and AI applications.

Acknowledgments

This research was supported by internal awards from Saint Louis University (Grant-0001651 and PROJ-000498) and the U.S. National Science Foundation (Award OAC-2451577). For computer time, this research used the Lonestar6 cluster from Texas Advanced Computing Center, the compute node at Innovative Computing Laboratory of the University of Tennessee, Knoxville, the Fugaku supercomputer at RIKEN, and Frontier supercomputer at Oak Ridge National Laboratory.

References

1. Tsuyoshi Ichimura, Kohei Fujita, Takuma Yamaguchi, Akira Naruse, Jack C Wells, Thomas C Schulthess, Tjerk P Straatsma, Christopher J Zimmer, Maxime Martinasso, Kengo Nakajima, et al. A fast scalable implicit solver for nonlinear time-evolution earthquake city problem on low-ordered unstructured finite elements with artificial intelligence and transprecision computing. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 627–637. IEEE, 2018.
2. Sameh Abdulah, Allison H. Baker, George Bosilca, Qinglei Cao, Stefano Cas-truccio, Marc G. Genton, David E. Keyes, Zubair Khalid, Hatem Ltaief, Georgiy L. Stenchikov Yan Song, and Ying Sun. Boosting earth system model outputs and saving petabytes in their storage using exascale climate emulators. *ACM Gordon Bell Prize for Climate Modelling Finalist*, 2024.
3. A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
4. "Hans Meuer, Erich Strohmaier, Jack Dongarra, and Horst Simon". The Top500 List. June 2024. <http://www.top500.org>.
5. Sameh Abdulah, Qinglei Cao, Yu Pei, George Bosilca, Jack Dongarra, Marc G Genton, David E Keyes, Hatem Ltaief, and Ying Sun. Accelerating geostatistical modeling and prediction with mixed-precision computations: A high-productivity approach with parsec. *IEEE Transactions on Parallel and Distributed Systems*, 33(4):964–976, 2021.
6. Qinglei Cao, Sameh Abdulah, Hatem Ltaief, Marc G Genton, David Keyes, and George Bosilca. Reducing data motion and energy consumption of geospatial modeling applications using automated precision conversion. In *2023 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 330–342. IEEE, 2023.
7. Azzam Haidar, Ahmad Abdelfattah, Mawussi Zounon, Panruo Wu, Srikara Pranesh, Stanimire Tomov, and Jack Dongarra. The design of fast and energy-efficient linear solvers: On the potential of half-precision arithmetic and iterative refinement techniques. In *Computational Science–ICCS 2018: 18th International Conference, Wuxi, China, June 11–13, 2018, Proceedings, Part I*, pages 586–600. Springer, 2018.
8. Azzam Haidar, Harun Bayraktar, Stanimire Tomov, Jack Dongarra, and Nicholas J Higham. Mixed-precision iterative refinement using tensor cores on gpus to accelerate solution of linear systems. *Proceedings of the Royal Society A*, 476(2243):20200110, 2020.
9. Alessio Netti, Yang Peng, Patrik Omland, Michael Paulitsch, Jorge Parra, Gustavo Espinosa, Udit Agarwal, Abraham Chan, and Karthik Pattabiraman. Mixed precision support in hpc applications: What about reliability? *Journal of Parallel and Distributed Computing*, 181:104746, 2023.
10. SR Nandakumar, Manuel Le Gallo, Christophe Piveteau, Vinay Joshi, Giovanni Mariani, Irem Boybat, Geethan Karunaratne, Riduan Khaddam-Aljameh, Urs Egger, Anastasios Petropoulos, et al. Mixed-precision deep learning based on computational memory. *Frontiers in neuroscience*, 14:406, 2020.
11. Aaron Walden, Eric Nielsen, Boris Diskin, and Mohammad Zubair. A mixed precision multicolor point-implicit solver for unstructured grids on gpus. In *2019 IEEE/ACM 9th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*, pages 23–30. IEEE, 2019.

12. Federico Brogi, Simone Bnà, Gabriele Boga, Giorgio Amati, T Esposti Ongaro, and Matteo Cerminara. On floating point precision in computational fluid dynamics using openfoam. *Future Generation Computer Systems*, 152:1–16, 2024.
13. Nicolas Doucet, Hatem Ltaief, Damien Gratadour, and David Keyes. Mixed-precision tomographic reconstructor computations on hardware accelerators. In *2019 IEEE/ACM 9th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*, pages 31–38. IEEE, 2019.
14. Siyuan Chen, Yi Zhang, Yiming Wang, Zhuang Liu, Xiaohan Li, and Wei Xue. Mixed-precision computing in the grist dynamical core for weather and climate modelling. *Geoscientific Model Development Discussions*, 2024:1–28, 2024.
15. Weile Jia, Han Wang, Mohan Chen, Denghui Lu, Lin Lin, Roberto Car, E Weinan, and Linfeng Zhang. Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning. In *SC20: International conference for high performance computing, networking, storage and analysis*, pages 1–14. IEEE, 2020.
16. Taisuke Boku, Ken-Ichi Ishikawa, Yoshinobu Kuramashi, and Lawrence Meadows. Mixed precision solver scalable to 16000 mpi processes for lattice quantum chromodynamics simulations on the oakforest-pacs system. In *2017 Fifth International Symposium on Computing and Networking (CANDAR)*, pages 362–368. IEEE, 2017.
17. Azzam Haidar, Stanimire Tomov, Jack Dongarra, and Nicholas J Higham. Harnessing gpu tensor cores for fast fp16 arithmetic to speed up mixed-precision iterative refinement solvers. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 603–613. IEEE, 2018.
18. Qinglei Cao, Sameh Abdulah, Rabab Alomairy, Yu Pei, Pratik Nag, George Bosilca, Jack Dongarra, Marc G Genton, David E Keyes, Hatem Ltaief, and Ying Sun. Reshaping geostatistical modeling and prediction for extreme-scale environmental applications. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (ACM Gordon Bell Prize Finalist)*, pages 1–12. IEEE, 2022.
19. Hatem Ltaief, Rabab Alomairy, Qinglei Cao, Jie Ren, Lotfi Slim, Thorsten Kurth, Benedikt Dorschner, Salim Bougouffa, Rached Abdelkhalek, and David E. Keyes. Toward capturing genetic epistasis from multivariate genome-wide association studies using mixed-precision kernel ridge regression. *ACM Gordon Bell Prize Finalist*, 2024.
20. OpenMP. OpenMP 4.5 Complete Specifications, 2015.
21. A. Duran, R. Ferrer, E. Ayguade, R. M. Badia, and J. Labarta. A proposal to extend the OpenMP tasking model with dependent tasks. *Intl. Journal of Parallel Programming*, 37(3):292–305, 2009.
22. Francesc Lordan, Enric Tejedor, Jorge Ejarque, Roger Rafanell, Javier Alvarez, Fabrizio Marozzo, Daniele Lezzi, Raúl Sirvent, Domenico Talia, and Rosa M Badia. Servicess: An Interoperable Programming Framework for the Cloud. *Journal of Grid Computing*, 2014.
23. C. Augonnet, S. Thibault, R. Namyst, and P. Wacrenier. StarPU: A unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency Computat. Pract. Exper.*, 23:187–198, 2011.
24. T. Heller, H. Kaiser, and K. Iglberger. Application of the ParalleX execution model to stencil-based problems. *Computer Science - Research and Development*, 28(2-3):253–261, 2013.

25. Michael Bauer, Sean Treichler, Elliott Slaughter, and Alex Aiken. Legion: Expressing locality and independence with logical regions. In *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, pages 1–11. IEEE, 2012.
26. George Bosilca, Aurélien Bouteiller, Anthony Danalis, Thomas Hérault, Pierre Lemarinier, and Jack J. Dongarra. DAGuE: A generic distributed DAG engine for high performance computing. *Parallel Comput.*, 38(1-2):37–51, 2012.
27. Aurelien Bouteiller, Thomas Hérault, Qinglei Cao, Joseph Schuchart, and George Bosilca. PaRSEC: Scalability, flexibility, and hybrid architecture support for task-based applications in ECP. *International Journal of High Performance Computing Applications*, 2024.
28. Qinglei Cao, George Bosilca, Nuria Losada, Wei Wu, Dong Zhong, and Jack Dongarra. Evaluating data redistribution in parsec. *IEEE Transactions on Parallel and Distributed Systems*, 33(8):1856–1872, 2021.
29. Qinglei Cao, Yu Pei, Kadir Akbudak, George Bosilca, Hatem Ltaief, David Keyes, and Jack Dongarra. Leveraging parsec runtime support to tackle challenging 3d data-sparse matrix problems. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 79–89. IEEE, 2021.
30. Qinglei Cao, Rabab Alomairy, Yu Pei, George Bosilca, Hatem Ltaief, David Keyes, and Jack Dongarra. A framework to exploit data sparsity in tile low-rank cholesky factorization. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 414–424. IEEE, 2022.
31. Qinglei Cao, Yu Pei, Thomas Hérault, Kadir Akbudak, Aleksandr Mikhalev, George Bosilca, Hatem Ltaief, David Keyes, and Jack Dongarra. Performance analysis of tile low-rank cholesky factorization using parsec instrumentation tools. In *2019 IEEE/ACM International Workshop on Programming and Performance Visualization Tools (ProTools)*, pages 25–32. IEEE, 2019.
32. Anthony Danalis, George Bosilca, Aurelien Bouteiller, Thomas Hérault, and Jack Dongarra. PTG: an Abstraction for Unhindered Parallelism. In *2014 Fourth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing*, pages 21–30. IEEE, 2014.
33. George Bosilca, Robert J Harrison, Thomas Hérault, Mohammad Mahdi Javanmard, P Nookala, and Edward F Valeev. The Template Task Graph (TTG)-an Emerging Practical Dataflow Programming Paradigm for Scientific Simulation at Extreme Scale. In *IEEE/ACM 5th International Workshop on Extreme Scale Programming Models and Middleware (ESPM2)*. IEEE, 2020.
34. R. Hoque, T. Hérault, G. Bosilca, and J. Dongarra. Dynamic Task Discovery in PaRSEC: A Data-flow Task-based Runtime. In *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, Scala '17*, 2017.
35. Robert A Van De Geijn and Jerrell Watts. Summa: Scalable universal matrix multiplication algorithm. *Concurrency: Practice and Experience*, 9(4):255–274, 1997.