利用硬件感知计算在混合精度矩阵乘法中的应用：一种以块为中心的方法

Qiao Zhang[1], Rabab Alomairy[2,3], Dali Wang[4], Zhuowei Gu[1], and Qinglei Cao[1]

[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

摘要 通用矩阵乘法（GEMM）是高性能计算（HPC）和人工智能（AI）广泛应用中的关键操作。针对低精度算术优化的硬件出现，需要重新评估数值算法以利用混合精度计算，从而实现性能和能效的提升。本研究引入了一个支持在细粒度块/单元级别上使用不同精度格式的自适应混合精度 GEMM 框架。我们利用 PaRSEC 运行时系统来平衡各种架构上的工作负载。该性能在基于 ARM CPU 的富岳 超级计算机、基于 Nvidia GPU 的 A100 DGX 以及基于 AMD GPU 的前沿 超级计算机上表现良好。本研究旨在通过结合算法进步和硬件创新，提高计算效率和精度，推动各种应用领域的变革性进展。

Keywords: 高性能计算 · 基于任务的运行时系统 · 通用矩阵乘法 · 混合精度。

1 介绍

通用矩阵乘法（GEMM）处于线性代数的核心，并作为众多高性能计算（HPC）应用中的关键内核—例如地震模拟 [1] 和天气及气候预测 [2]—以及在人工智能（AI）应用中，包括传统神经网络的全连接层和自然语言处理 [3]。GEMM 是一种计算密集型操作，因其对内存和计算都有很高的要求，其性能直接影响这些应用程序的效率，因此成为各种架构优化的重点，从同构 CPU 集群到异构 GPU 系统。

在过去十年中，硬件设计师优先设计在高速、节能和低精度计算方面表现出色的处理器 ~ [4]。一个显著的例子是 Nvidia GPU 在 HPC 和 AI 任务中的广泛应用，这得益于其卓越的处理能力和效率。最新的 Nvidia GPU，

包括 H100 和即将推出的 B100，集成了各种精度格式，每种都对峰值理论吞吐量产生不同的影响。最低精度级别（H100 上的 FP8/INT8 和 B100 上的 FP4）分别比 FP64 计算快 58× 倍和 233× 倍，突显了从传统的双精度算术转向 Tensor Core 加速的低精度执行所能带来的显著加速。在 CPU 中也观察到了类似的发展趋势，在那里，FP64 性能通常是 FP32 的两倍，是 FP16 的四倍。这种转变促使研究人员重新评估经典数值技术，识别出可以在不影响整体解决方案质量的情况下使用降低精度的领域。此外，利用低精度不仅减少了内存需求——允许更多的数据适应高速内存并因此提高执行率 ˜ [5, 6]——还降低了能源消耗，使其成为提高计算工作流能效的一种有吸引力的战略 ˜ [7]。

硬件进步与算法优化之间的协同作用在解决复杂计算挑战中扮演着关键角色。混合精度计算，即在一个任务中利用多种数值精度的方法，已经成为一种强大的策略 [8,9]。这种方法已成功融入众多科学和工程学科领域，如深度学习 [10]，计算流体力学 [11,12]，天文学 [13]，气候建模 [14]，分子动力学 [15] 和量子力学 [16]。这些技术的主要优势在于能够利用较低精度的硬件能力提高计算速度，同时保持必要的准确性约束。然而，大多数现有的实现依赖于粗粒度的精度调整，并且优化单 GPU 或单一节点设置，这限制了它们在大规模分布式环境中的适用性。在分布式多 GPU 架构中高效处理混合精度计算引入了许多挑战，由于数据表示和精度一致性方面的差异。此外，许多现有解决方案针对特定硬件平台进行定制，降低了其跨多样计算架构的可移植性。

本文介绍了一个细粒度的 tile/block 级别的混合精度 GEMM 框架（GEMM-MP），并将 GEMM-MP 集成到运行时系统 PaRSEC 中。本文的贡献如下：

- 引入了一种以瓦片为中心的混合精度 GEMM 算法，该算法仔细平衡了准确性和性能，为 HPC 和 AI 应用提供了一个强大且值得信赖的混合精度解决方案。
- 利用动态任务基于的运行时系统高效地编排异构任务，动态适应工作负载变化，同时最小化调度开销并最大化资源利用率。
- 表现出强大的可扩展性和移植性，适用于各种同质和异构计算平台，确保了广泛的应用性和效率。

据我们所知，这是首次引入以 tile 为中心的混合精度 GEMM 算法的工作。

本文的结构如下。第 2 节讨论了与我们工作相关的先前研究。在第 3 节中，我们提供了 PaRSEC 的概述。瓷砖中心混合精度 GEMM 框架在第 4 节

中进行描述。然后在第 5 节中进行了性能评估，最后，第 6 节概述了我们的结论和未来方向。

2 相关工作

2.1 混合精度矩阵计算

混合精度方法引起了广泛关注 [8, 9] 并在多个计算领域得到了广泛应用 [10–16] 。一个著名的例子是用于线性求解器的迭代细化技术，其中系统最初使用较低精度进行求解，然后通过反复应用较高精度的修正来细化残差。该策略显著减少了计算时间和能耗，使其成为提高大规模计算效率的有效方法 [7,17] 。然而，此类方法的效果高度依赖于矩阵条件数，并且需要存储多个精度版本的矩阵可能会引入较大的内存开销。

正在进行的研究探索在确保数值可靠性的同时通过有选择地在非关键计算中使用降低精度而在必要时保留较高精度来优化性能 [2, 18, 19] 。这些方法主要针对线性系统解决方案。在深度学习中，减少精度通常应用于前向和后向计算，而权重更新往往需要更高的精度累加以保持数值稳定性。然而，这种折衷可能会由于前向和后向传播中的关键计算的精度损失而导致准确性下降。在这项工作中，我们介绍了一个自适应的基于块的混合精度 GEMM 框架，在单个计算过程中动态调整精度水平。这一细粒度的方法可以提供另一种解决方案，以确保在 HPC 和应用程序中认识到并非所有计算组件都需要相同的精度级别，从而保持不妥协的准确性。

2.2 运行时系统

最近基于任务的编程模型的进步推动了现代超级计算机上科学应用程序的可扩展性。开发了一系列运行时系统以促进高效的任务执行，优化跨不同架构的资源利用。开放 MP [20] 已成为共享内存并行性的广泛采用的标准，其中包含了提供动态调度和细粒度控制的任务构造。通过利用编译器指令和运行时支持，OpenMP 简化了并发管理，并提高了多线程环境下的性能。基于 OpenMP，OmpSs [21] 扩展了异构架构的能力，引入了高级依赖跟踪和异步执行策略。类似地，COMPSs [22] 提供了一个全面的编程环境，在分布式计算基础设施中简化任务并行性。对于异构、分布式计算平台，星 PU [23] 提供了一个灵活的运行时系统，该系统将调度机制与显式内核注释集成在一起，优化了 CPU 和 GPU 上的执行。同时，HPX [24] 利用 ParalleX

模型通过现代 C++ 中的轻量级线程和异步执行范例来支持高度可扩展的应用程序。另一个显著的系统，军团 [25]，强调任务和数据分区，实现自动并行性检测和改进的局部优化。通过将计算与硬件映射解耦，Legion 实现了在分布式和异构环境中对复杂工作负载的高度可扩展性。本工作利用了在下一节中描述的 PaRSEC 。

3 PaRSEC 运行时系统

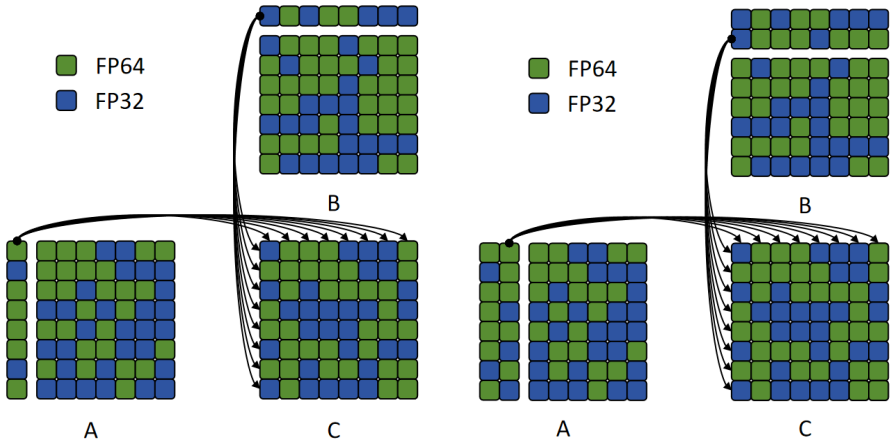
PaRSEC [26, 27] 作为为分布式、多核和异构架构调度和管理微任务设计的通用框架。它集成了运行时环境，以及用于定义任务图的多种编程模型——应用于稠密线性代数计算 [28] 和如混合精度技术 [6]、低秩分解 [29] 和稀疏操作 [30] 等不规则工作负载——以及辅助库，以简化从旧有实现的过渡。此外，它还集成了性能分析工具 [31] 以协助在大规模异构平台上调整应用程序。PaRSEC 为用户提供了一系列领域特定语言（DSL），例如参数化任务图（PTG）[32] 、模板任务图（TTG）[33] 和动态任务发现（DTD）[34] ，以增强算法表达的灵活性。在这些语言中，PTG 以紧凑且参数化的格式编码整个 DAG，允许优化集体通信重用并分离数据分布与执行。这种抽象方法符合我们对自适应混合精度计算核心研究的方法论，并且我们在本工作中使用了 PTG。

4 算法描述

通用矩阵-矩阵乘法（GEMM）操作计算两个矩阵的乘积，可选地缩放并与第三个矩阵累加，定义为：$C \leftarrow \alpha AB + \beta C$，其中 A 和 B 是输入矩阵，C 是输出矩阵，α, β 是标量系数。算法 1 描述了自适应混合精度 GEMM 的顺序过程，其中输入矩阵 $A(i,l)_{m\times k}$、$B(l,j)_{k\times n}$ 和输出矩阵 $C(i,j)_{m\times n}$ 被划分成固定大小的块/瓷砖。该算法遵循 SUMMA（可扩展通用矩阵乘法算法）

Algorithm 1: 以磁瓦为中心的混合精度 GEMM。

```
for l = 0 to k
    for i = 0 to m
        for j = 0 to n
            C*(i,j) = αA#(i,l) × B$(l,j) + βC*(i,j)
```



(a) 迭代 $k = 0$。 (b) 迭代 $k = 1$。

图 1: 混合精度 GEMM 的前两次迭代示例。箭头代表引入通信的依赖关系。

方法 [35]。最外层循环遍历约简维度 k，而两个内层循环则迭代行索引 i 和列索引 j。在每次迭代中，算法更新矩阵 C 的值。符号 #、$ 和 $*$ 表示操作数的不同数据表示或精度。

图 1 说明了如算法 1 中定义的前两个迭代中的任务执行。该可视化采用颜色编码来区分数值精度：绿色表示双精度（FP64 或 DP），蓝色表示单精度（FP32 或 SP）。本研究专注于 FP64 和 FP32，未来工作计划扩展到其他精度格式。箭头表示由于矩阵块之间的数据传输和通信而产生的代表性依赖关系（$A \to C$ 和 $B \to C$）在单一迭代中。每个依赖/通信都与特定的数据类型相关联。由于这种自适应的以块为中心的精度方法，任务可能接收到与其操作精度不同的精度的数据，从而需要潜在的精度转换。各种转换策略将在后续研究中进行探索。本工作采用了接收方转换策略，这意味着数据类型转换在接收端处理。也就是说，每个数据流中的数据类型由存储数据的精度决定。例如，如果 A 或 B 中的块以 FP64 存储，则传输的数据保持为双精度。

该算法被集成到 PaRSEC 运行时系统中，作为执行的基础框架。通过利用 PaRSEC ，我们确保了矩阵操作的高效并行处理，并优化了计算、通信和内存使用之间的权衡，这是由于自适应块中心混合精度算法引入的工作负载不平衡所导致的。

5 性能结果与分析

5.1 实验设置

实验在三个系统上进行。

- 盖约特: 它是一个 GPU 加速的 AMD 计算节点，配备有两个 EPYC 7742 CPU，每个 CPU 有 64 个核心，运行频率为 2.25 GHz，主内存为 2,063 GB，并且配有八块 NVIDIA A100-SXM4-80GB GPU。
- 富岳: 该系统基于 ARM 架构，由超过 150,000 个计算节点组成。每个节点采用 48 核 A64FX CPU，可在加速模式下达到 2.2 GHz 的峰值频率，在正常模式下为 2.0 GHz，并配备 32 GB 的 HBM2 内存。
- 前沿: 这是一台采用 AMD 硬件的 GPU 加速超级计算机，由 9,408 个计算节点组成。每个节点集成了一个 64 核心的 AMD 优化的第三代 EPYC 处理器以及四块 AMD MI250X GPU，并配备了 512 GB 的 DDR4 内存。

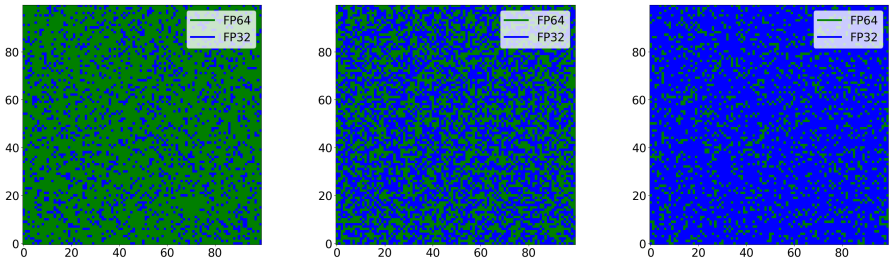
我们使用术语 "'a'D:'b'S" 表示不同精度格式的百分比，其中 D/S 表示双/单精度，a/b 表示双/单精度的百分比，确保 $a + b = 100$。对于数据分布，我们采用大小为 $P \times Q$ 的 2D 块循环方案，并尽量将其结构化为正方形。在富岳 上，由于电力限制，我们在正常模式下执行并禁用扇区缓存优化（SCO）以避免内存冲突 [18]。当在 Frontier 上运行时，我们每节点使用 4 个 rank。最优瓦片大小是通过实验确定的，在富岳 上设置为 1024，在盖约特和前沿 上设置为 2048。

5.2 内核执行精度图

图 2 通过热力图展示了在不同配置下每个独立瓦片上执行的数值内核的精度，如图 1 所引用。每个点代表我们在设置中随机生成的 FP64 或 FP32 格式下的一个瓦片的精度。三个子图展示了在 A、B 和 C 中的精度分布不同：(a)`80D:20S`，其中 80%的瓦片使用双精度（FP64），而 20%使用单精度（FP32）；(b)`50D:50S`，其中 FP64 和 FP32 的瓦片分布相等；(c)`20D:80S`，其中 20%的瓦片使用 FP64 而 80%使用 FP32。这些配置展示了不同程度的混合精度计算，将在后续的性能测量中采用。

5.3 共享内存性能

图 3 展示了不同硬件平台上的共享内存性能。每个子图说明了在各种精度设置下的性能以及相对于 `100D:0S` 的速度提升。不同的精度配置表现



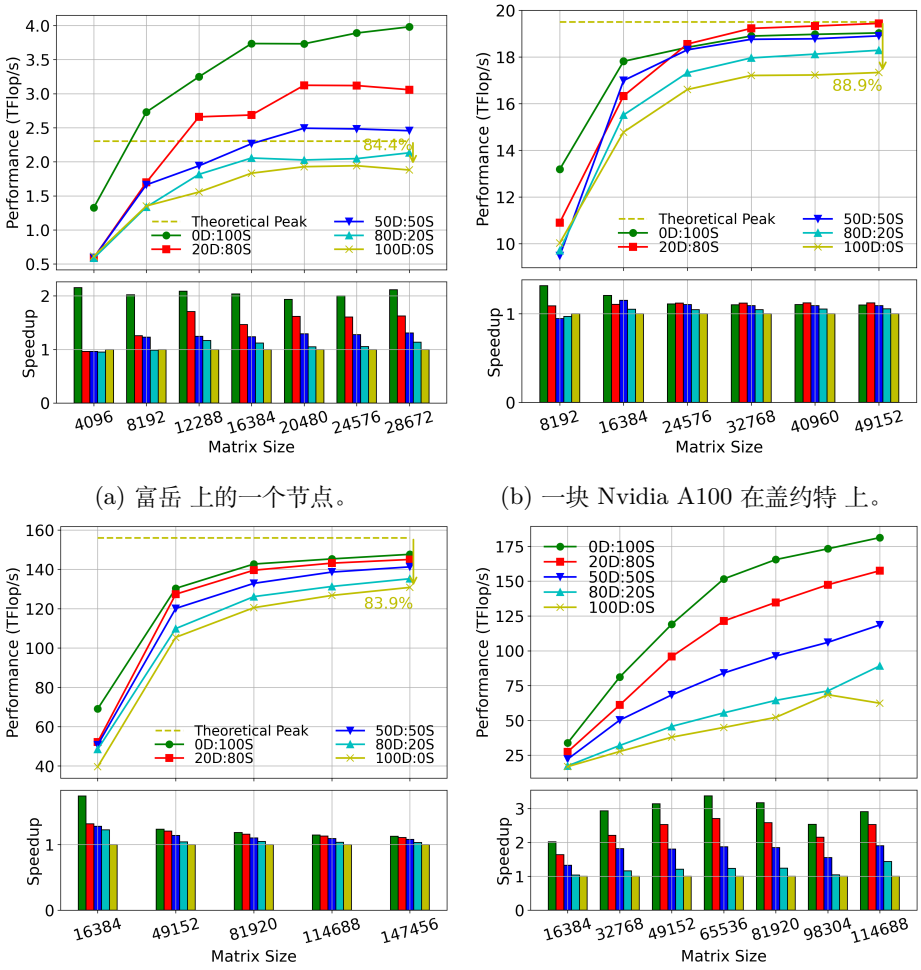
(a) 80D:20S. (b) 50D:50S. (c) 20D:80S.

图 2: 大小为 $102,400 \times 102,400$ 的矩阵的内核精度热图，瓦片大小为 $1,024 \times 1,024$。

出不同的扩展行为，总体上随着低精度的比例增加，性能得到改善。(a) 展示了在富岳 单个节点上的性能。虚线表示实际的 FP64 峰值性能，计算方法为每个核心的 GEMM 性能乘以核心数量。100D:0S 配置实现了该实际峰值的 84.7%，而 0D:100S 则将其翻倍。(b) 展示了在单个 Nvidia A100 GPU 上对盖约特 的结果，其中 FP64 和 FP32 精度具有相同的理论峰值性能。100D:0S 配置达到了理论峰值的 88.9%。随着 FP32 比例降低而导致性能下降可能归因于数据移动增加和/或数据类型转换开销增加。(c) 展示了在八个 Nvidia A100 GPU 上对盖约特 的性能，展示了从单个 GPU 性能的几乎线性扩展（见图 3(b)）。100D:0S 配置达到了理论峰值的 83.9%。(d) 在单个节点上评估性能为前沿 。0D:100S 在一个节点上的实际峰值性能约为 280.0 T 浮点运算/秒，实现的性能约为 181.3 T 浮点运算/秒，相当于实际峰值的 64.8%。100D:0S 的性能稍微落后，可能是由于在前沿 中 NIC 靠近 GPU，以及在 PaRSEC 中正在开发的 GPU 直接通信支持。

5.4 性能可扩展性

图 4 展示了在 64 个节点上富岳 和前沿 的分布式内存性能，进一步证实了共享内存性能中观察到的趋势，如图 3 所示：实现的性能接近理论/实际峰值，不同的精度配置表现出不同的扩展行为，随着较低精度比例的增加，性能通常会提高。此外，GEMM-MP 在共享内存性能到 64 个节点上分别在前沿 和富岳 上几乎达到了线性扩展。例如，在 0D:100S 配置中，富岳 的单节点性能约为 4.0 Tflop/s，而在 64 个节点上的性能达到 242.2 Tflop/s，具有 94.6%的并行效率。在前沿 上，单节点性能约为 181.3 Tflop/s，在 64 个



(a) 富岳 上的一个节点。

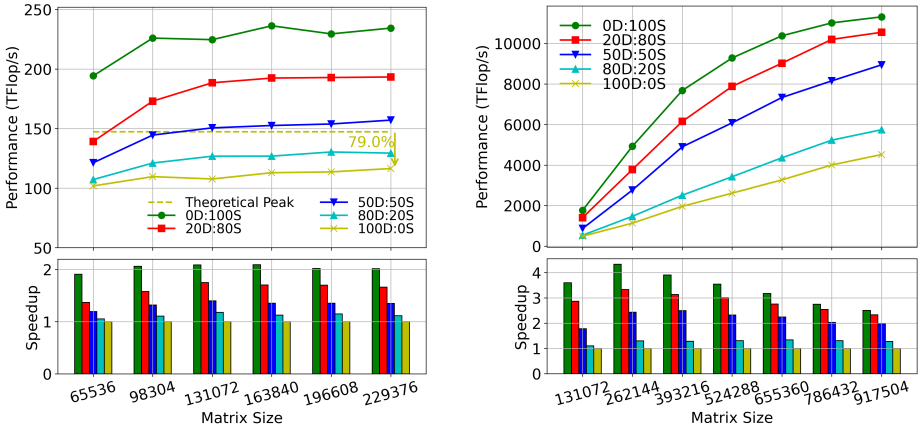
(b) 一块 Nvidia A100 在盖约特 上。

(c) 八个 Nvidia A100 在盖约特 上。

(d) 前沿 上的一个节点。

图 3: 共享内存性能的 GEMM-MP 。

节点上增加到 11,310.3 Tflop/s，并行效率为 97.5%。总而言之，这些结果强调了所提出的以块为中心的混合精度框架在实现共享内存和分布式内存环境中各种架构下的最优计算效率方面的重要性。



(a) 富岳 上有 64 个节点。 (b) 前沿 上有 64 个节点。

图 4: 分布式内存性能的 GEMM-MP 。

6 结论与未来工作

本工作介绍了 GEMM-MP，一个自适应的基于瓦片的混合精度 GEMM 框架。该方法建立在 PaRSEC 之上，有效地管理由于不同精度格式而产生的异构任务，在共享内存和分布式内存环境中均能保证高性能执行。GEMM-MP 在同质 CPU 配置和异质 GPU 系统上表现出强大的效率和可扩展性，使其能够高度适应各种硬件架构。未来，我们计划通过引入额外的精度格式并开发适用于无损和有损方法的信任度精度选择策略来扩展该框架。我们还旨在研究不同的数据类型转换策略。此外，我们寻求将此混合精度框架与其它运行时系统集成，并在实际的高性能计算和 AI 应用中部署它。

致谢

该项研究得到了圣路易斯大学内部奖项(Grant-0001651 和 PROJ-000498)和美国国家科学基金会 (Award OAC-2451577) 的支持。在计算机时间方面，本研究使用了德克萨斯高级计算中心的 Lonestar6 集群、田纳西大学诺克斯维尔分校创新计算实验室的计算节点、理化学研究所的富岳超级计算机以及橡树岭国家实验室的 Frontier 超级计算机。

参考文献

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 Castruccio, 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, Raül 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 Herault, 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 Herault, 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 Herault, 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 Herault, 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. Herault, 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.