# Scaling Singular Values Beyond GPU Memory Limits: Out-of-Core, GPU-Accelerated, and Unified Across Data Precision and Hardware

Evelyne Ringoot
Massachusetts Institute of Technology
Cambridge, MA, USA
eringoot@mit.edu

Rabab Alomairy (advisor)
Massachusetts Institute of Technology
Cambridge, MA, USA
rabab.alomairy@mit.edu

Valentin Churavy (advisor)
University of Mainz & University of Augsburg
Mainz, Germany
vchuravy@uni-mainz.de

Alan Edelman (advisor)
Massachusetts Institute of Technology
Cambridge, MA, USA
edelman@mit.edu

## 1 Introduction

Today's algorithms process terabyte-scale datasets [4, 8]. Among dimensionality-reduction techniques, the *Singular Value Decomposition* (SVD) is particularly important as it is rank-revealing [7]. For example, in *Low-Rank Adaptation* of large language models [5], the SVD reduces trainable parameters, thereby lowering inference time and storage needs while preserving accuracy.

However, the diversity of emerging architectures—including multi-core CPUs, GPUs, and specialized accelerators—combined with the explosion of data precisions (e.g., FP64, FP32, FP16, BF16) poses a fundamental challenge for performance portability. Traditional HPC libraries, such as cuSOLVER, oneMKL, MAGMA, and SLATE [3] are hardware-specific and require separate implementations for different architectures and data precisions. This fragmentation creates significant challenges when scaling workloads across heterogeneous environments where computation must seamlessly transition between devices, exploit mixed-precision arithmetic, and adapt to rapidly evolving hardware. In addition, development time and user complexity increase with every specialized library. With this work, we aim to propose an alternative: a unified composable singular value solver, where new hardware can be seamlessly plugged into the existing algorithms.

In previous work [9], we addressed the divergence in GPU-resident linear algebra libraries by developing an open-source singular value solver that delivers performance comparable to specialized libraries across multiple data precisions and GPU platforms, including NVIDIA, AMD, Intel, and Apple Silicon. demonstrating that portability does not need to come at the cost of performance. In this work, we present an out-of-core GPU-accelerated singular value solver designed to achieve performance portability that:

- Unifies heterogeneous hardware under a single extensible interface.
- Supports multi-precision to exploit the full capabilities of modern accelerators.
- Provides out-of-core scalability for datasets larger than GPU memory.


*Conference'17, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 2 Methods

We build on the QR-based communication algorithm proposed by Kabir et al. [2, 6]: for every diagonal tile of the matrix, a unitary transformation is calculated that renders the first block column upper triangular (RQ sweep) and the first block row lower diagonal (LQ sweep), resulting in a banded matrix (Figure 1). In the out-of-core implementation, every block row (RQ sweep) or block column (LQ sweep) is loaded into GPU memory successively, leveraging the GPU for computation and the CPU for data storage, while overlapping communication and computation. Once the remaining lower left part of the matrix fully fits into GPU memory, a GPU-resident computation without CPU-GPU communication follows.

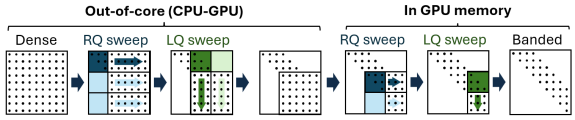


**Figure 1: The classic two-stage QR-based SVD. From [9].**

While the communication strategy from Kabir et al [6] was designed to minimize total communication cost, novel developments in low-latency hardware with higher memory have shifted the focus to balancing communication and computation to achieve maximum total performance. As such, while the previous proposed communication strategy is to keep the top row in GPU memory together with the lower right partition, we propose keeping a larger number of top rows in memory instead, more efficiently making use of the GPU computing power, as shown in Figure 2.

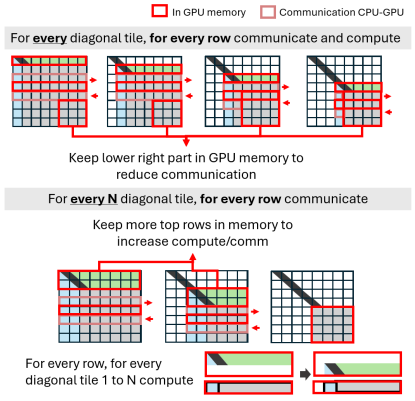


**Figure 2: The optimized data communication strategy (bottom) compared to the method proposed by Kabir et al (top).**

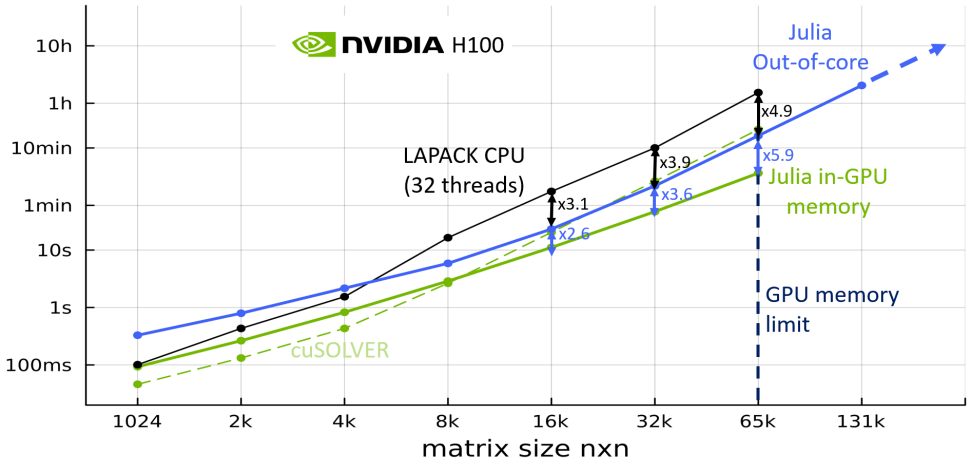


**Figure 3: Runtime of the unified out-of-core singular value solver, compared to the in-GPU unified solver, CUSOLVER, and LAPACK on the CPU using 32 threads: the out-of-core solver outperforms the CPU-based LAPACK solver, while handling matrix sizes larger than the in-GPU solver and CUSOLVER.**

We build this algorithm utilizing the multiple-dispatch capabilities of the Julia language [1]. In particular, we implement a `LargeMatrix` format, and re-use the previous GPU-resident algorithm for the reduction to band form for both `LargeMatrix` and `AbstractGPUMatrix`. Only in the subfunction `GETSMQRT` does the algorithm then split off to include communication for the `LargeMatrix` format, as shown in Algorithm 1.

**Algorithm 1** Julia: stage one of the QR-based SVD, extended from [9]

```
function banddiag!(A::GPUorLargeMatrix{T}, Tau::
    AbstractGPUMatrix{T}, N::Int, backend) where T
    for k in 1:(N-1)
        GETSMQRT!(A,Tau,k, N, backend)
        GETSMQRT!(A',Tau,k, N, backend, "LQ")
    end
    GETSMQRT!(A,Tau,N, N, backend)
end
function GETSMQRT!(A::AbstractGPUMatrix, ...)
    GETSQRT_fused!(A,k)
    UNTSMQR_fused!(A, k)
end
function GETSMQRT!(A::LargeMatrix, ...)
    GETSMQRT!(A.data[k],...)
    ... #communication CPU-GPU
    KernelAbstractions.@synchronize
end
```

## 3 Results

Figure 3 shows that our Julia-based unified *out-of-core* implementation consistently outperforms CPU-based solutions such as `LAPACK`, while also enabling the processing of matrices exceeding the GPU memory capacity—a limitation for both `cuSOLVER` and the unified *in-GPU* Julia implementation, which performs on-par with vendor libraries[9]. Although matrices that fit entirely within GPU memory are typically dispatched to the *in-core* algorithm, for benchmarking purposes, we evaluate the out-of-core implementation under constrained memory conditions: the GPU is assumed to accommodate only the lower-right quarter of the matrix at each size. Consequently, the *out-of-core* algorithm executes on half of the rows and columns, while the remaining quarter is processed in-core, allowing us to study the scaling behavior across a wide range of matrix sizes.

In addition to providing performance across matrix sizes, our implementation is agnostic to data precision and hardware architecture. Figure 4 shows the performance on AMD GPUs and NVIDIA GPUs for various data sizes and types.

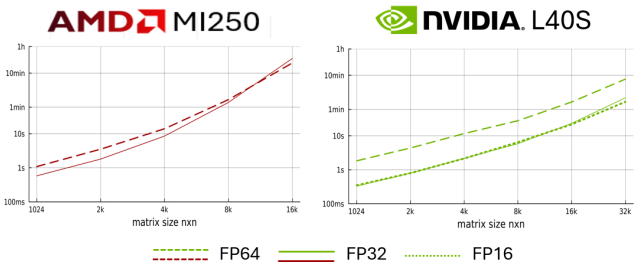


**Figure 4: Runtime of the unified out-of-core singular value solver across hardware and precision.**

## Acknowledgment

We thank our collaborators James Schloss, Julian Samaroo, and Tim Besard. We acknowledge IBEX of KAUST, MIT ORCD, US NSF (CNS-2346520, PHY-2028125, RISE-2425761, DMS-2325184, OAC-2103804), DARPA (HR00112490488), DoE (DE-NA0003965), and USAFR (FA8750-19-2-1000). The U.S. Government, its agencies and employees do not make any warranty, nor endorses, recommend or favors, nor assumes any liability for anything in this report. The views expressed herein are those of the authors alone.

## References


[1] Jeff Bezanson, Jiahao Chen, Benjamin Chung, Stefan Karpinski, Viral B. Shah, Jan Vitek, and Lionel Zoubritzky. 2018. Julia: dynamism and performance reconciled by design. *Proc. ACM Program. Lang.* 2, OOPSLA, Article 120 (Oct. 2018), 23 pages. doi:10.1145/3276490

[2] Jack Dongarra, Mark Gates, Azzam Haidar, Jakub Kurzak, Piotr Luszczek, Stanimire Tomov, and Ichitaro Yamazaki. 2018. The Singular Value Decomposition: Anatomy of Optimizing an Algorithm for Extreme Scale. *SIAM Rev.* 60, 4 (2018), 808–865. arXiv:https://doi.org/10.1137/17M1117732 doi:10.1137/17M1117732

[3] Mark Gates, Ahmad Abdelfattah, Kadir Akbudak, Mohammed Al Farhan, Rabab Alomairy, Daniel Bielich, Treece Burgess, Sébastien Cayrols, Neil Lindquist, Dalal Sukkari, et al. 2025. Evolution of the SLATE Linear Algebra Library. *The International Journal of High Performance Computing Applications* 39, 1 (2025), 3–17.

[4] Google. 2022. The Size and Quality of a Data Set. Google Developers Foundational courses in Machine Learning. https://developers.google.com/machine-learning/data-prep/construct/collect/data-size-quality

[5] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[6] Khairul Kabir, Azzam Haidar, Stanimire Tomov, Aurelien Bouteiller, and Jack Dongarra. 2017. A Framework for Out of Memory SVD Algorithms. In *High Performance Computing*, Julian M. Kunkel, Rio Yokota, Pavan Balaji, and David Keyes (Eds.). Springer International Publishing, Cham, 158–178.

[7] Carla D. Martin and Mason A. Porter. 2012. The Extraordinary SVD. *The American Mathematical Monthly* 119, 10 (2012), 838–851. doi:10.4169/amer.math.monthly.119.10.838

[8] Cameron Musco and Christopher Musco. 2015. Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition. In *Advances in Neural Information Processing Systems* (2015), C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc.

[9] Evelyne Ringoot, Rabab Alomairy, Valentin Churavy, and Alan Edelman. 2025. Performant Unified GPU Kernels for Portable Singular Value Computation Across Hardware and Precision. (2025). arXiv:arXiv:2508.06339 doi:10.1145/3754598.3754667