jh220009

Hierarchical Low-Rank Approximation Methods on Distributed Memory and GPUs

Rio Yokota (Tokyo Institute of Technology)

Abstract

The purpose of this research is to develop a scalable and highly optimized open source library for hierarchical low-rank approximation of dense matrices. During the previous JHPCN project we have extended the $\mathcal{H}$-matrix code to perform not only matrix-vector multiplications, but also matrix-matrix multiplication, LU factorization, and QR factorization.We have also extended the parallelization to support not only OpenMP and MPI, but also batched GPU kernels and task-based parallelization. The four main goals for the fiscal year 2022 are: 1) Application of $\mathcal{H}$-matrix algorithm to the tridiagonalization during the eigenvalue solvers of dense matrices, 2) Extending the GPU implementation of $\mathcal{H}$-matrices to make use of TensorCores, 3) Extending the $\mathcal{O}(N)$ $\mathcal{H}$-matrix LU factorization to distributed memory, 4) Extending the $\mathcal{O}(N)$ $\mathcal{H}$-matrix LU factorization to LDL factorization. We were able to achieve our research goal for all 4 objectives. This year's results were published in top journals like ACM TOMS and top conferences such as SC22.

1 Basic information

1.1 Collaborating JHPCN centers

- Hokkaido University
- The University of Tokyo
- Tokyo-tech
- Nagoya University
- Kyoto University
- Osaka University

1.2 Theme area

- Large-scale computational science

1.3 Research area

- Very large-scale numerical computation

1.4 Project members and their roles

Rio Yokota (Overall coordination)
Ichitaro Yamazaki (Overall advise)
Akihiro Ida (Lattice $\mathcal{H}$-matrix)
Takeshi Iwashita (Electromagnetics App.)
Takeshi Fukaya (QR)
Satoshi Ohshima (GPU optimization)
Kengo Nakajima (Preconditioning)
Toshihiro Hanawa (GPU optimization)
Tetsuya Hoshino (GPU optimization)
Tasuku Hiraishi (MPI)
Hiroyuki Ootomo (TensorCore)
Sameer Deshmukh (PaRSEC)
Muhammad Ridwan Apriansyah (QR)
Qianxiang Ma (Algorithm)

Thomas Spendlhofer (Iterative Refinement)
Sora Takashima (Generalization)
Xinyu Zhang (Modeling)
Tomoya Takahashi (I/O)
Kai Okawa (Visualization)
Sixue Wang (Fisher matrix)
Shukai Nakamura (CPU optimization)
Hiro Ishii (Hessian matrix)
Tomokazu Saito (FMM)
Ishikawa Satoki (Auto-tuning)
Shota Nakamura (MPI)
Wang Zhaoqing (GPU optimization)
Toshiki Omi (GPU optimization)

2 Purpose and Significance of the Research

2.1 Purpose of Research

The purpose of this research is to develop a scalable and highly optimized open source library for structured low-rank approximation of dense matrices, *e.g.* $\mathcal{H}$-matrix, $\mathcal{H}^2$-matrix, HSS, HODLR, BLR. In this project report we will simply call these various types of structured low-rank approximations as $\mathcal{H}$-matrices. Such large dense matrices naturally appear in electromagnetic, seismic, quantum, and fluid simulations, in scientific computing. Unlike their dense counterparts which require $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory, $\mathcal{H}$-matrices can perform matrix multiplication and factorization in $\mathcal{O}(N)$ time and $\mathcal{O}(N)$ memory.

During the previous JHPCN project we have extended the $\mathcal{H}$-matrix code from an $\mathcal{O}(N \log^2 N)$ method to a new $\mathcal{O}(N)$ method for LU factorization of dense matrices. We also benchmarked our implementation against other open source libraries for $\mathcal{H}$-matrices. Furthermore, we extended the BLR QR factorization [1] to a full $\mathcal{H}$-matrix QR factorization. The present JHPCN project applies the $\mathcal{H}$-matrix algorithm to the tridiagonalization during the eigenvalue solvers of dense matrices, extends the GPU implementation to make use of TensorCores, extends the $\mathcal{O}(N)$ LU factorization to distributed memory, and extends the $\mathcal{O}(N)$ LU factorization to LDL factorization.

2.2 Significance of Research

Hardware architecture is now moving towards low-precision arithmetic, backed by the increasing demand from the machine learning field. When such low-accuracy can be tolerated, exact dense linear algebra operations become unnecessary, and libraries such as BLAS and LAPACK, which are at the heart of HPC applications, can be replaced by hierarchical low-rank ($\mathcal{H}$-matrix) libraries that effectively do the same work in linear time. There is still ample room for investigation regarding the use of such low-precision in scientific computing applications, where methods such as iterative refinement have recently gained interest. $\mathcal{H}$-matrices can be used as a scalable preconditioner for such problems, and we aim to quantify the advantage over existing state-of-the-art methods in this JHPCN project.

Furthermore, batched operations on GPUs are becoming popular and libraries such as MAGMA and cuBLAS are providing low-level functions that can process many small dense matrix operations in large batches. $\mathcal{H}$-matrices can benefit greatly from such batched dense linear algebra libraries, and in doing so will be able to extract a large portion of the performance of the latest GPU and many-core architectures including Tensor Cores. Since libraries like MAGMA and CUBLAS are optimized to use Tensor Cores, we do not have to do the implementation ourselves.

3 Significance as JHPCN Joint Research Project

Each member of this project has different expertise, all of which are essential for the development and verification of a high performance $\mathcal{H}$-matrix library.

- R. Yokota' s group is currently developing a C++-based $\mathcal{H}$-matrix code Hatrix that uses advanced C++ features to provide a collection of primitives for performing $\mathcal{H}$-matrix computation with hybrid parallelism for MPI, OpenMP, and CUDA over half of the project members are students in his group.
- A. Ida and T. Iwashita are developers of HACApK – a hybrid MPI-OpenMP-CUDA implementation of the $\mathcal{H}$-matrix.
- T. Hiraishi has experience in load-balancing for distributed memory $\mathcal{H}$-matrix codes.
- I. Yamazaki is the developer of dense linear algebra libraries such as MAGMA and PLASMA.
- S. Oshima, T. Hanawa and T. Hoshino have expertise in tuning solvers for GPUs and Xeon Phi.
- K. Nakajima has expertise in parallel preconditioned iterative solvers.

The combination of these expertise is necessary for achieving the goals mentioned above. There are a few existing $\mathcal{H}$-matrix implementations, but they are limited to shared memory and have not been ported to GPUs. To our knowledge, HACApK and HiCMA are the only multi-GPU $\mathcal{H}$-matrix codes available at the moment. This could only have been done through a JHPCN international collaboration between the experts in each area.

4 Outline of Research Achievements up to FY2021 (Only for continuous projects)

Up to FY2021 we have tackled various problems regarding hierarchical low-rank approximation and its parallel implementation. There are various derivatives of hierarchical low-rank approximation methods such as; BLR, HODLR, HSS, $\mathcal{H}$-matrix, and $\mathcal{H}^2$-matrix. We started from the most basic variant – BLR, which uses low-rank off-diagonal blocks, but not a hierarchical matrix. We started with the most basic operations such as matrix-vector and matrix-matrix multiplication. This was extended during FY2016 to LU factorization and implemented in OpenMP and MPI.

- In FY2017, we extended the matrix format to more complex HSS and $\mathcal{H}$-matrix structures, and extended the implementation to GPUs for the matrix-vector multiplication. We utilized batched MAGMA operations to process the matrix-vector multiplication efficiently on GPUs.

- In FY2018, we further extended the implementation of the LU factorization to multiple-GPUs using a hybrid MPI + OpenMP + CUDA code.
- In FY2019 we extended the $\mathcal{H}$-matrix code to $\mathcal{H}^2$-matrix by using a nested basis. We also used a runtime for $\mathcal{H}$-LU on GPU, but found that such runtimes like StarPU and OmpSs incur too much overhead. For the inner kernels, we ported the QR decomposition to run on TensorCores, and implemented the QR decomposition using the BLR matrix.
- In FY2020 we implemented the uniform basis BLR, and QR factorization on TensorCores with error correction. We also developed a Eigenvalue computation based on BLR-QR, and developed a GPU implementation of the lattice $\mathcal{H}$-matrix.
- In FY2021 we improved the complexity of the $\mathcal{H}^2$-matrix LU decomposition from $\mathcal{O}(N \log^2 N)$ to $\mathcal{O}(N)$, compared various runtime systems, benchmarked against other libraries such as STRUMPACK and LORAPO, and extended to LDL decomposition.

5 Details of FY2022 Research Achievements

The four main goals for the fiscal year 2022 are

1. Application of $\mathcal{H}$-matrix algorithm to the tridiagonalization during the eigenvalue solvers of dense matrices
2. Extending the GPU implementation of $\mathcal{H}$-matrices to make use of TensorCores
3. Extending the $\mathcal{O}(N)$ $\mathcal{H}$-matrix LU factorization to distributed memory
4. Extending the $\mathcal{O}(N)$ $\mathcal{H}$-matrix LU factorization to LDL factorization

We were able to complete all tasks. Task 1 was published in [4], task 2 has been submitted to IJHPCA, task 3 has been published in [7,8], task 4 has been submitted to ICPP2023 and its distributed memory implementation to EuroMPI2023. Other publications in Section 7 are results from previous years that were published in FY2022.

5.1 Application of $\mathcal{H}$-matrix algorithm to the tridiagonalization during the eigenvalue solvers of dense matrices

5.1.1 Research plan

Eigenvalue decomposition of dense matrices are performed by libraries such as ELPA or EigenExa. However, the dense matrix must first be tridiagonalized before the eigendecomposition is performed. This tridiagonalization can be done in $\mathcal{O}(N \log N)$ time if $\mathcal{H}$-matrices are applied. During FY2022 1Q, we will develop a novel algorithm for applying $\mathcal{H}$-matrices during the tridiagonalization phase of a dense eigenvalue decomposition.

5.1.2 Progress

We developed a fast tridiagonalization method based on the block low rank (BLR) structure, that reduces the complexity of the tridiagonalization from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^{7/3})$ [3]. Block Householder vectors are also formed using BLR-matrices. The procedure

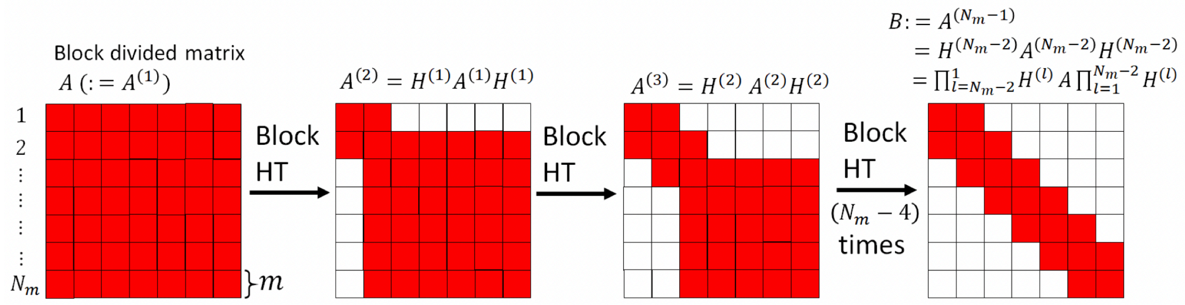
Block divided matrix
$A\ (:= A^{(1)})$
$A^{(2)} = H^{(1)}A^{(1)}H^{(1)}$
$A^{(3)} = H^{(2)}A^{(2)}H^{(2)}$
$B := A^{(N_m-1)} = H^{(N_m-2)}A^{(N_m-2)}H^{(N_m-2)} = \prod_{l=N_m-2}^{1} H^{(l)} A \prod_{l=1}^{N_m-2} H^{(l)}$
Block HT
Block HT
Block HT (N_m-4) times
1
2
N_m
m

Fig. 1 A block-divided dense matrix A is transformed into a block tridiagonal matrix B using the block Householder transformation.

for forming a block tridiagonal structure is shown in Fig. 1. In numerical experiments of a string free vibration problem with known analytical solutions, for large eigenvalues, the calculated eigenvalues using the proposed method converge toward the analytical ones in accordance with the theoretical convergence curves. Owing to the reduced complexity, an eigenvalue decomposition of a matrix was solved with about N = 300,000, which is significantly larger than the limit of conventional methods for dense matrices, within a reasonable amount of time on CPU cores. For the calculation time, the proposed method was faster than the conventional method when the matrix size N was larger than a few tens of thousands.

5.2 Extending the GPU implementation of $\mathcal{H}$-matrices to make use of TensorCores

5.2.1 Research plan

In a separate JHPCN project we have developed an error correction algorithm that recovers full single precision accuracy while using TensorCores. Our results showed an exact match between the accuracy of a pure single precision computation and a computation on TensorCores with our error correction scheme[2]. Half precision is not usable in most $\mathcal{H}$-matrix computations, but with our novel error correction scheme, we should be able to make use of TensorCores during $\mathcal{H}$-matrix computations without loss of accuracy.

5.2.2 Progress

Until the first half of FY2022 we were planning to use Kokkos for implementing our $\mathcal{H}$-matrix code on GPUs. However, we had a algorithmic breakthrough which allowed us to make the LU/Cholesky factorization inherently parallel, which we presented at SC22. This new algorithm allows us to simply call batched GPU kernels on Tensor Cores for the three basic functions for LU factorization GETRF, TRSM, and GEMM. This is much more efficient than calling many GPU kernels through Kokkos. Our code can scale up to 128 GPUs and is about 10,000x faster than the 2021 Gordon Bell finalist code LORAPO as shown in Fig. 2.

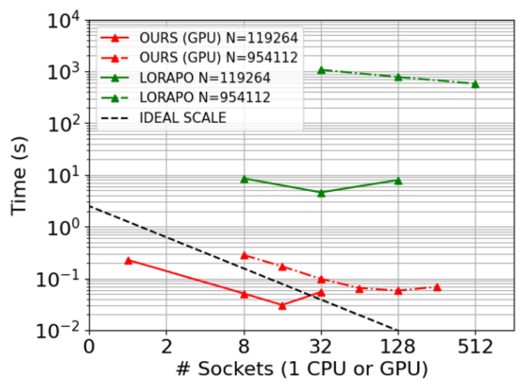

Fig. 2 Strong scaling experiments on GPUs comparing our code with LORAPO.

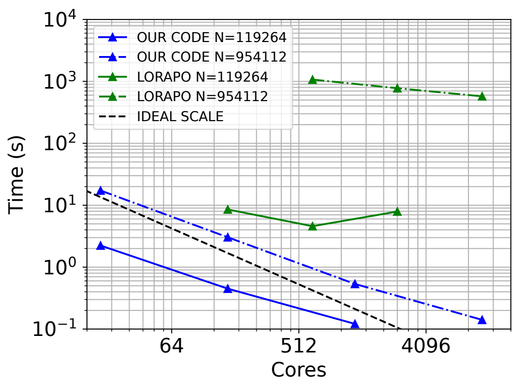

Fig. 3 Strong scaling experiments on CPUs comparing our code with LORAPO.

5.3 Extending the $\mathcal{O}(N)$ $\mathcal{H}$-matrix LU factorization to distributed memory

5.3.1 Research plan

During FY2021 we have developed a novel algorithm that can reduce the complexity of $\mathcal{H}$-matrix LU factorization to $\mathcal{O}(N)$. This algorithm is also able to remove the data dependency on the trailing matrices, which is a huge problem when parallelizing LU factorization of dense matrices. We will make use of this feature to construct an embarrassingly parallel dense LU factorization in distributed memory.

5.3.2 Progress

Out of the various hierarchical low-rank matrix formats such as $\mathcal{H}$-matrix, $\mathcal{H}^2$-matrix, HSS, HODLR, BLR, only HSS and $\mathcal{H}^2$-matrices have $\mathcal{O}(N)$ complexity for LU factorization. For HSS matrices, it is possible to remove the dependency on the trailing matrices during LU factorization, which results in a highly parallel algorithm. This allows the LU factorization to be performed in an embarrassingly parallel fashion. However, the weak admissibility of HSS causes the rank of off-diagonal blocks to grow for 3-D problems, and the method is no longer $\mathcal{O}(N)$. On the other hand, the strong admissibility of $\mathcal{H}^2$-matrices allows it to handle 3-D problems in $\mathcal{O}(N)$, but introduces a dependency on the trailing matrices. In the present work, we pre-compute the fill-ins and integrate them into the shared basis, which allows us to remove the dependency on trailing-matrices even for $\mathcal{H}^2$-matrices. Comparisons with a block low-rank factorization code LORAPO showed a maximum speed up of 4,700x for a 3-D problem with complex geometry [7].

5.4 Extending the $\mathcal{O}(N)$ $\mathcal{H}$-matrix LU factorization to LDL factorization

5.4.1 Research plan

Applications in electronic structure calculation require the computation of the k-th eigenvalue, where the value or range of the eigenvalue is not known a priori. A common method for solving such a problem is to perform a binary search though successive LDL factorizations. Electronic structure calculations result in dense matrices, so a method which can compute an LDL factorization of a dense matrix in $\mathcal{O}(N)$ will greatly reduce the computation time of such calculations. In FY2022 4Q , we will develop such an algorithm. The overall complexity of the binary search will be $\mathcal{O}(N \log N)$, since we need to perform the LDL factorization $\mathcal{O}(\log N)$ times.

5.4.2 Progress

We have extended the $\mathcal{O}(N)$ $\mathcal{H}$-matrix LU factorization in task 3 to LDL factorization, which was quite trivial due to the similarity of the matrix operations in LU and LDL factorizations. We applied this to a electronic structure calculation that requires the computation of the k-th eigenvalue. The LDL factorization is performed recursively during the binary search to find the k-th eigenvalue. We confirmed that the accuracy of the resulting eigenvalue matched that of a dense $\mathcal{O}(N^3)$ eigenvalue solver, while the complexity was reduced to $\mathcal{O}(N \log N)$. This drastic improvement in computational complexity will allow us to scale such dense eigenvalue computations in electronic structure calculations to orders of millions or even billions.

6 Self-review of Current Progress and Future Prospects

The four main goals for the fiscal year 2022 are

1. Application of $\mathcal{H}$-matrix algorithm to the tridiagonalization during the eigenvalue solvers of dense matrices
2. Extending the GPU implementation of $\mathcal{H}$-matrices to make use of TensorCores
3. Extending the $\mathcal{O}(N)$ $\mathcal{H}$-matrix LU factorization to distributed memory
4. Extending the $\mathcal{O}(N)$ $\mathcal{H}$-matrix LU factorization to LDL factorization

For (1), owing to the reduced complexity, an eigenvalue decomposition of a matrix was solved with about N = 300,000, which is significantly larger than the limit of conventional methods for dense matrices [4].

For (2), our novel $\mathcal{H}^2$-ULV factorization algorithm and batched GPU implementation can scale up to 128 GPUs and is about 10,000x faster than the 2021 Gordon Bell finalist code LORAPO. This work has been submitted to IJHPCA.

For (3), the $\mathcal{H}^2$-ULV factorization algorithm on distributed memory is about 4,700x faster than the 2021 Gordon Bell finalist code LORAPO. This work was accepted to SC22 [7].

For (4), we have successfully developed a $\mathcal{H}$-matrix LDL decomposition, and used it while slicing the spectrum using a binary search to obtain the nth eigenvalue of a dense matrix in $\mathcal{O}(N \log N)$ time.

7 List of publications and presentations

Journal Papers (Refereed)

1. S. Deshmukh, Rio Yokota, George Bosilca, “Cache Optimization and Performance Modeling of Batched, Small, and Rectangular Matrix Multiplication on Intel, AMD, and Fujitsu Processors”, ACM Transactions on Mathematical

Software, 2023.

2. M. R. Apriansyah, R. Yokota, “QR Decomposition of Block Low-Rank Matrices”, ACM Transactions on Mathematical Software, https://doi.org/10.1145/3538647 (2022).
3. H. Ootomo, R. Yokota, “Recovering Single Precision Accuracy from Tensor Cores While Surpassing the FP32 Theoretical Peak Performance”, The International Journal of High Performance Computing Application, Vol. 26, No. 4, https://doi.org/10.1177/10943420221090256 (2022).
4. A. Ida, “Solving Block Low-Rank Matrix Eigenvalue Problems”, Journal of Information Processing,Vol. 30, pp.538-551 (2022).

Proceedings of International Conference Papers (Refereed)

5. H. Ootomo, Rio Yokota, Mixed-Precision Random Projection for RandNLA on Tensor Cores, Platform for Advanced Scientific Computing (PASC), Jun. 2023.
6. H. Ootomo, H. Manabe, K. Harada, R. Yokota, Quantum Circuit Simulation by SGEMM Emulation on Tensor Cores and Automatic Precision Selection, ISC High Performance, May 2023.
7. Q. Ma, S. Deshmukh, R. Yokota, Scalable Linear Time Dense Direct Solver for 3-D Problems Without Trailing Sub-Matrix Dependencies, The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC22), Nov. 2022.

Presentations at International conference (Non-refereed)

8. Q. Ma, R Yokota, O(N) Factorization of Dense Matrices on GPUs Without Trailing Submatrix Dependencies, SIAM CSE, Feb. 2023.
9. M. R. Apriansyah, R. Yokota, Parallel QR Factorization of Block Low-Rank Matrices, SIAM CSE, Feb. 2023.
10. H. Ootomo, R. Yokota, Reducing Shared Memory Footprint to Leverage High Throughput on Tensor Cores and its Flexible API Extension Library, HPC Asia, Feb. 2023.
11. S. Ohshima, A. Ida, R. Yokota and I. Yamazaki(+), QR Factorization of Block Low-Rank Matrices on Multi-Instance GPU, The 23rd International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT’ 22), Dec. 2022.