JH160041-NAHI

Hierarchical Low-Rank Approximation Methods on Distributed Memory and GPUs

Rio Yokota（Tokyo Institute of Technology）

Hierarchical low-rank approximation methods such as H-matrix, H2-matrix, and HSS can compress a dense matrix with O(N2) elements into a hierarchical matrix with O(N) elements. By using such compressed matrices it is possible to perform matrix-matrix multiplication, LU decomposition, and eigenvalue computation in near-linear time. Hierarchical matrices can also be applied to the Schur complements that arise in sparse direct solvers, so their applicability extends to fluid, structure, and electromagnetic simulations. However, these hierarchical algorithms are rather new and highly optimized implementations do not exist at the moment. A highly optimized distributed memory GPU implementation is needed to extract the potential parallelism of these methods.

1. Basic Information

(1) Collaborating JHPCN Centers

The University of Tokyo
Information Technology Center

Tokyo Institute of Technology
Global Scientific Information and Computing Center

Hokkaido University
Information Initiative Center

Kyoto University
Academic Center for Computing and Media Studies

(2) Research Areas

- ☑ Very large-scale numerical computation
- ☐ Very large-scale data processing
- ☐ Very large capacity network technology
- ☐ Very large-scale information systems

(3) Roles of Project Members

Rio Yokota (Tokyo Institute of Technology)
Low-rank approximation using FMM and its GPU-MPI implementation

Ichitaro Yamazaki (University of Tennessee)
Development of distributed memory runtime –ParSEC and blocked BLAS library for GPU –block MAGMA

Akihiro Ida (University of Tokyo)
Feature extension of hybrid MPI/OpenMP H-matrix code –HACApK, and its integration with ParSEC and block MAGMA

Takeshi Iwashita (Hokkaido University)
Application of HACApK to boundary integral solvers for electromagnetics, and optimization of H-matrix-vector product

Takayuki Aoki (Tokyo Institute of Technology)
Application of HACApK to Poisson solvers for multiphase flows

Satoshi Oshima (University of Tokyo)
GPU implementation of HACApK and

integration with MAGMA

Taku Hiraishi (Kyoto University)
Dynamic load-balancing of HACApK

Kengo Nakajima (University of Tokyo)
Extend capability of HACApK within the ppOpen-HPC framework.

Jack Dongarra (University of Tennessee)
Development of distributed memory runtime –ParSEC and blocked BLAS library for GPU –block MAGMA

2. Purpose and Significance of the Research

The present research aims to develop a high performance implementation of hierarchical low-rank approximations on distributed memory and GPU systems. The distributed memory implementation and GPU implementation will be performed in a incremental fashion by starting with matrix-vector multiplication, then matrix factorization, and finally eigenvalue solvers. Even the matrix-vector product has important applications in boundary integral solvers for electromagnetics, so validation using real applications can be performed every step of the way.

2.1 Extension of H-matrices

Distributed memory implementation of LU factorization of H-matrices suffers from large amount of communication and is not currently implemented in HACApK. In the present work, we switch to the non-hierarchical block low-rank (BLR) structure and simplify the data dependency during the LU factorization in order to reduce the amount of communication. In H-matrices, the dense matrix is hierarchically subdivided into smaller and smaller blocks, and each of these blocks must be compressed using low-rank approximation methods. HACApK uses adaptive cross approximation (ACA) for the low-rank approximation. ACA has a small number of arithmetic operations but is known to fail in cases with low local contrast. In the current study, we will use the fast multipole method (FMM) instead of ACA to perform the compression more robustly without loss of performance.

2.2 Implementing H-matrices on next generation supercomputers

Modern supercomputers have CPUs with tens of cores, but future CPUs are anticipated to have hundreds or even thousands of cores. HACApK is optimized for multicore CPU, but requires modifications to extract the full potential of many core processors. To this end, we need to rethink parallel algorithms by considering hardware characteristics.

3. Significance as a JHPCN Joint Research Project

The significance of conducting the current research joint with the team from the University of Tennessee is the integration of the tools that are being developed there. The current work aims to extend the capability of HACApK, which is the only open source hybrid OpenMP/MPI parallel H-matrix library. There are few efforts to use HPC technology in H-matrix codes, let alone efforts to develop highly optimized implementations on many-core processors such as GPUs and Intel Xeon Phis. The current project integrates other popular libraries such as MAGMA, ParSEC, Tascell, and exaFMM. The developer of each of these libraries is a member of this project, so

the integration between these libraries can be done effectively. The focus is on ease of code development, while we also aim to solve problems that could not be previously tackled.

4. Outline of the Research Achievements up to FY 2015

The current project started from FY 2016.

5. Details of FY 2016 Research Achievements

5.1 Research Achievements of First Half of FY 2016

Our goal is to have a GPU implementation of our H-matrix code, and extend it to LU factorizations and use it as a preconditioner [9]. To this end, understanding the relation between H-matrices and FMM is critical, because FMM is known to perform well on GPUs and have recently possessed the ability to be used not only as a mat-vec, but also a preconditioner [1,30]. Therefore, during the first half of FY2016 we focused on understanding the relation between H-matrices and FMM, because this is the shortest path to achieving our objective.

FMM and H-matrices lie at the opposite ends of the spectrum of hierarchical low-rank approximation methods [5], which are shown in Figure 1. ``Compressed operators" in Figure 1 represents a method which recompresses the FMM translation matrix by using SVD. There exists a compute-memory tradeoff between FMM and H-matrices [17].

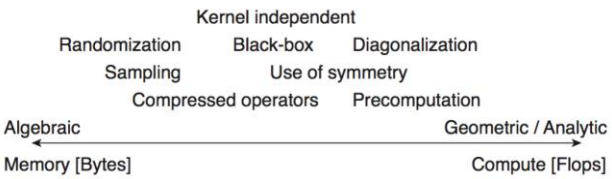


Figure 1. Compute-memory tradeoff in hierarchical low-rank approximation method

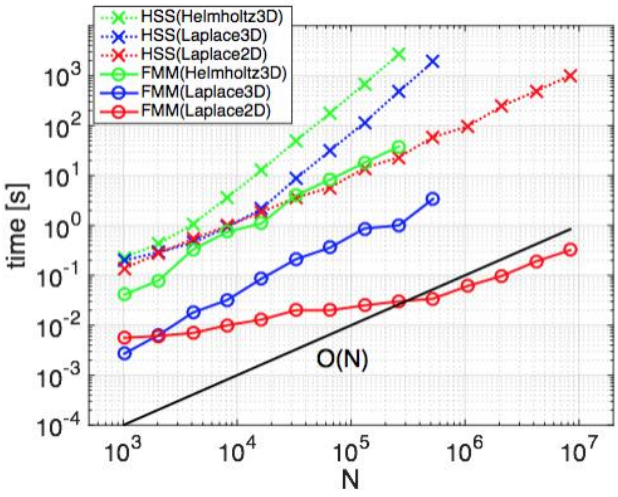


Figure 2 Calculation time for a single matrix-vector multiplication including setup time for the Green's function matrix of a 2-D Laplace, 3-D Laplace, and 3-D Helmholtz equation

This method was originally designed to accelerate the translation of multipole expansion in FMM. When this technique is used the FMM becomes very similar to a H2-matrix or HSS matrix. Therefore, we perform a direct comparison between FMM and HSS.

We use the 2-D Laplace, 3-D Laplace, and 3-D Helmholtz equations as the problem of interest, and generate the matrices from the Green's function. A single core of a 12 core Ivy Bridge (E5-2695v2) is used for the calculations. In Figure 2, we show the calculation time for the FMM and HSS. The x-axis is the size of the matrix, and the y-axis is the calculation time in seconds. FMM has a constant overhead so for small N it does not show O(N) behavior. For sufficiently large N, both FMM and HSS show O(N) behavior. It can be seen that FMM is about 1000 times faster than HSS [17]. This is due to the large calculation cost of the algebraic compression that HSS does, whereas the FMM does not.

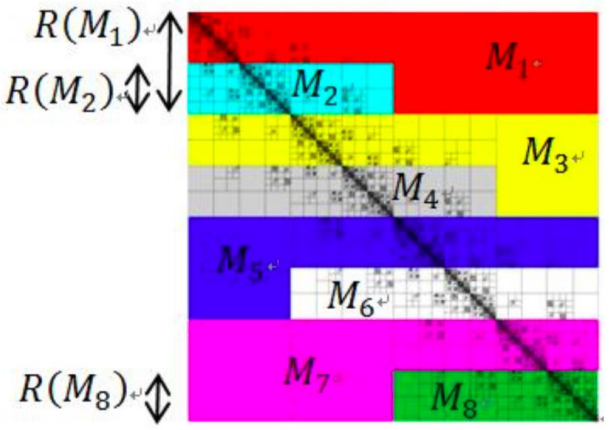


Figure 3. Domain decomposition of H-matrix

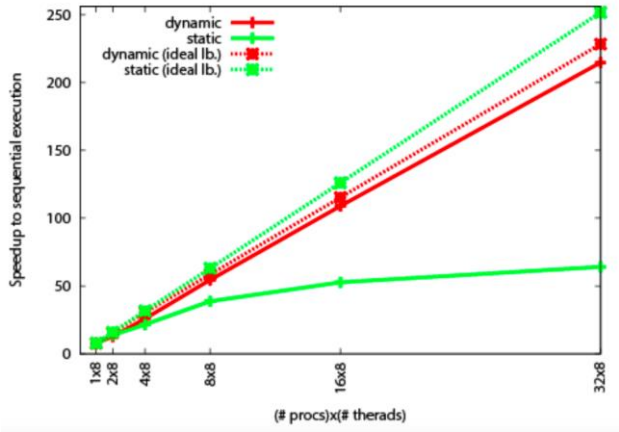


Figure 4. Parallel speedup of H-matrix

Another important aspect of our work in the first half of FY2016 is the load-balancing of the distributed memory implementation [2]. The decomposition of a hierarchical matrix is shown in Figure 4. In the present work, we improve the load-balance by introducing a dynamic load-balancing scheme that predicts the ranks of each block [28]. We use a test problem that gave poor load-balance for a static load-balancing scheme, to validate our new dynamic load-balancing scheme [7]. We use dynamic task scheduling in OpenMP. The solid line in Figure 5 is the result when we use dynamic load-balancing. We can see that all threads are calculating similar number of elements.

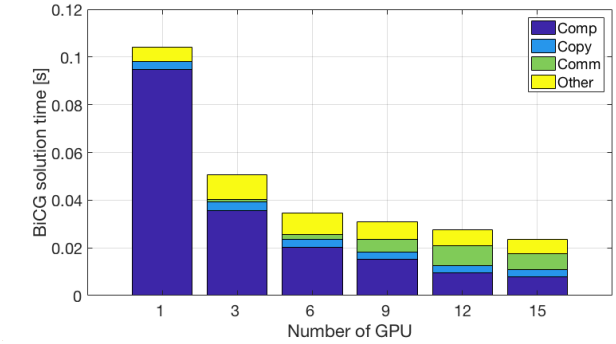


Figure 5. Performance of BiCGSTAB using HACApK for the mat-vec on multiple GPUs.

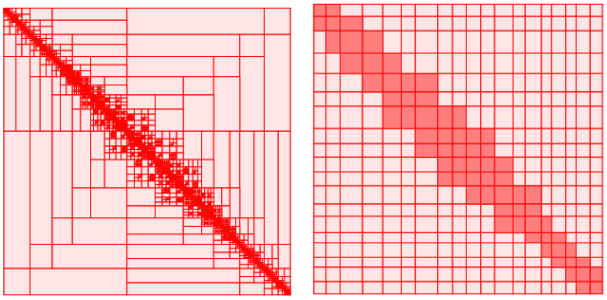
5.2 Research Achievements of Second Half of FY 2016

In the second half of FY2016 we focused on our final objectives of a) implementing HACApK on GPU and b) extending HACApK to LU decomposition.

The GPU implementation of HACApK was achieved through the use of MAGMA. Since one of the developers of MAGMA – Ichitaro Yamazaki was a collaborator in this project, the integration of MAGMA with HACApK was done in a very short amount of time. We were therefore able to have a multi-GPU version by the end of FY2016.

The performance of HACApK on multi-GPU is shown in Figure 3, where ``Comp" is the computational kernels, ``Copy" is the CUDA memory copy time, ``Comm" is the MPI communication. The MPI communication time eventually becomes the bottleneck because the mat-vec computation part takes very little time on the GPU.

We have also extended HACApK to run on the Xeon Phi Knights Landing [23]. This port is more straightforward than that of the GPU, since it is merely changing the BLAS library to the AVX512 optimized version and compiling for the Knights Landing.



(a) H-matrix (b) BLR

Figure 6. Contrast between a H-matrix and BLR matrix structure. The dark red boxes are dense matrices, while the light red boxes are low-rank matrices. BLR is a non-hierarhical block low rank matrix.

We have successfully extended HACApK to perform LU decomposition on H-matrices. We chose to use the BLR format, which is a non-hierarchical version of the block low-rank matrices. This is due to the simpler data dependency of BLR compared to H-matrices when performing the LU decomposition. BLR still uses low-rank approximation for the off-diagonal blocks, and therefore does much less arithmetic operations compared to the dense matrices. Our initial experiments showed that, the data-dependency of the hierarchical matrix [6] greatly decreased the available concurrency during the LU decomposition. This is why we chose to use the non-hierarchical BLR format, which is more parallel.

In order to assess the benefit of using HACApK for LU decomposition, we compared the LU decomposition in HACApK with LAPACK and achieved up to 6 times speed up. This proves that our approach of first compressing the dense matrix into a H-matrix form has a great benefit over performing the LU decomposition on the dense matrix directly.

6. Progress of FY 2016 and Future Prospects

6.1 Progress of First Half of FY 2016

The progress of the first half of FY 2016 can be summarized as follows:

- Analyzed the relation between the analytical (matrix-free) FMM and the algebraic H/H²/HSS-matrix
- Quantified the compute-memory tradeoff between FMM and H-matrices
- Studied the communication properties of the domain decomposition of hierarchical matrix structures
- Developed a dynamic load-balancing scheme for hierarchical matrices

6.2 Progress of Second Half of FY 2016

The progress of the second half of FY 2016 can be summarized as follows:

- Developed the first GPU implementation of a H-matrix code
- Achieved 8x speed up on a K40 GPU over a 20 core Haswell GPU
- Extended the GPU H-matrix code to handle multiple GPUs
- Achieved speed up on up to 15 GPUs
- Extended HACApK to handle LU decomposition of H-matrices
- Achieved 6x speedup over LAPACK

6.3 Future Prospects

For improving the scalability of the distributed memory H-matrix codes, we are considering the possibility of importing various techniques from FMM. We have studies the communication patterns of FMM extensively and have constructed a performance model that predicts the behavior on the largest supercomputers such as Mira, Titan, and Shaheen [10]. This performance

model was used to construct an asymptotically superior communication scheme for FMM [4]. The communication pattern of H-matrices are identical to FMM so these techniques should be directly applicable to these algebraic variants of FMM. We have also investigated the properties of data-locality of FMM [21] and its performance portability [27]. These findings can also be transferred to HACApK in FY 2017 since the project has been renewed.

The lack of straightforward parallelism in the LU decomposition of H-matrices could possibly be circumvented by the use of task-based parallelism. We have been working on efficient use of task-based parallelism [29], and techniques to increase its robustness [13].

Finally, we have been looking into the use of H-matrices in both the convectional applications such as electromagnetic boundary integral problems [3][8], and also non-traditional applications such as long-range force calculation in Monte Carlo methods [16][20][22] and molecular dynamics simulations [19][25].

A more recent effort has been to extend our work on hierarchical low-rank approximation to machine learning [11]. It is well known that machine learning kernels require very low accuracy. This is why NVIDIA is developing 16-bit arithmetic units, while Google goes further and uses 8-bit arithmetic units in their tensor processing unit (TPU). However, the matrix-matrix multiplication is done exactly with the O(N^3) dense matrix kernels in the current implementation of NVIDIA and Nervana (now Intels) machine learning libraries. H-matrices can approximate the matrix-matrix multiplication in $O(N\log^2 N)$ time by sacrificing the accuracy. Therefore, it is a perfect match for this application which requires only 8-bit arithmetic accuracy. Since H-matrices become exponentially faster as the required accuracy decreases, there will be a large benefit if these techniques can be used for machine learning kernels.

7. List of Publications and Presentations

(1) Journal Papers

1. H. Ibeid, R. Yokota, J. Pestana, D. Keyes, ``Fast Multipole Preconditioners for Sparse Matrices Arising from Elliptic Equations", Computing and Visualization in Science, accepted.
2. S. Okuno, T. Hiraishi, H. Nakashima, M. Yasugi, J. Sese: ``Parallelization of Extracting Connected Subgraphs with Common Itemsets in Distributed Memory Environments", Journal of Information Processing, Vol. 25, pp, 256—267, 2017.
3. A. Ida, T. Ataka, T. Mifune, Y. Takahashi, T. Iwashita, A. Furuya,``Application of Improved H-matrices in Micromagnetic Simulations", (in press)
4. R. Yokota, ``Communication Optimization of Distributed Memory FMM for Large Scale Boundary Element Methods", Simulation, Vol. 35, No. 3, pp. 23–29, 2016.
5. R. Yokota, ``Tradeoff between FMM and H^2(HSS)-matrices", Journal of the Japan Society for Computational Engineering and Science, Vol. 21, No. 4, pp. 3498–3501, 2016.
6. A. Ida, ``Numerical Computation and Data Structure of Hierarchical Low-rank Approximation Methods", Simulation, Vol. 35, No. 3, pp. 30—36, 2016.
7. A. Ida, ``Low-rank Approximation Methods for Large Scale Scientific

Computing", Journal of the Japan Society for Computational Engineering and Science, Vol. 21, No. 4, pp. 10-13, 2016.

8. A. Ida, T. Iwashita, T. Mifune, Y. Takahashi, ``Framework for Parallel Boundary Element Analysis Using H-matrices and its Application", Journal of the Japan Society for Computational Engineering and Science, Vol. 21, No. 4, pp. 22-25, 2016.
9. A. Ida, T. Iwashita, T. Mifune, and Y. Takahashi, ``Variable Preconditioning of Krylov Subspace Methods for Hierarchical Matrices With Adaptive Cross Approximation", IEEE Transactions on Magnetics, Vol. 52, Issue 3, Article# 7205104, 2016
10. H. Ibeid, R. Yokota, D. Keyes, ``A Performance Model for the Communication in Fast Multipole Methods on HPC Platforms", International Journal of High Performance Computing Applications, Vol. 30, No. 4, pp. 423–437, 2016.
11. J. Castrillon, R. Yokota, M. Genton, ``Multi-Level Restricted Maximum Likelihood Covariance Estimation and Kriging for Large Non-Gridded Spatial Datasets", Spatial Statistics, Vol. 18, pp. 105–124, 2016.
12. A. Ida, T. Iwashita, M. Ohtani, K. Hirahara, ``Improvement of Hierarchical Matrices with Adaptive Cross Approximation for Large-scale Simulation", Journal of Information Processing, Vol. 23, No. 3, pp.366—372, 2016.
13. T. Hiraishi, S. Okuno, M. Yasugi, ``An Implementation of Exception Handling with Collateral Task Abortion", Journal of Information Processing, Vol. 24, No. 2, pp. 439-449, 2016.

(2) Conference Papers

14. T. Iwashita, A. Ida, T. Mifune and Y. Takahashi, ``Software Framework for Parallel BEM Analyses with H-matrices Using MPI and OpenMP", Tools for Program Development and Analysis in Computational Science, ICCS2017. (in press)
15. T. Iwashita, A. Ida, T. Mifune and Y. Takahashi, ``Software Framework for Parallel BEM Analyses with H-matrices", IEEE 17th Biennial Conference on Electromagnetic Field Computation, Miami USA, Novevmber 13, 2016

(3) Conference Presentations

16. R. Igarashi, A. Ida, ``Monte Carlo Simulation Using Approximate Long-Range Force Calculations", 72nd Annual Conference of the Physical Society of Japan, Osaka, Japan, March 17, 2016.
17. R. Yokota, ``Compute-Memory Tradeoff in Hierarchical Low-Rank Approximation Methods", SIAM Conference on Computational Science and Engineering, Atlanta, USA, February 27, 2017.
18. A. Ida, ``Low Rank Approximation Methods Used in Hierarchical Matrices", ATAT in HPC 2017, Taipei, Taiwan, March 11, 2017.
19. R. Yokota, ``Energy Conservation of Fast Multipole Methods in Classical Molecular Dynamics Simulations", 7th AICS International Symposium, Kobe, Japan, February 24, 2017.
20. R. Igarashi, A. Ida, ``Monte Carlo

Simulation Using Approximation to Long Range Interactions", 1st International Symposium on Research and Education of Computational Science, Tokyo, Japan, November 30, 2016.

21. R. Yokota, ``Improving Data Locality of Fast Multipole Methods", Third Workshop on Programming Abstractions for Data Locality, Kobe, Japan, October 24, 2016.
22. R. Igarashi, A. Ida, ``Monte Carlo Simulation Using Approximate Long-Range Force Calculations", Autumn Conference of the Physical Society of Japan, Kanazawa, Japan, September 14, 2016.
23. S. Ohshima, A. Ida, H. Hanawa, N, Kawai, ``Optimizing Hierarchical Matrix-Vector Multiplication on Many-Core Architectures", Summer United Workshops on Parallel, Distributed and Cooperative Processing, Matsumoto, Japan, August 8, 2016.
24. A. Ida, ``Matrix Operations Using Hierarchical Matrix Forms", Summer United Workshops on Parallel, Distributed and Cooperative Processing, Matsumoto, Japan, August 8, 2016.
25. R. Yokota, ``Fast Multipole Method Library for Multiple Architectures and its Application to Molecular and Fluid Simulations", 8th Symposium of the Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures, Tokyo, Japan, July 14, 2016.
26. A. Ida, ``Application of Hierarchical Matrices with Adaptive Cross Approximation to Large-scale Simulations", ISC High Performance 2016, Frankfurt, Germany, June 23, 2016.
27. R. Yokota, ``Perforamance Portability of FMM, 21st Conference of Japan Computational Engineering Society", Niigata, Japan, May 31, 2016.
28. A. Ida, T. Hiraishi, T. Iwashita, ``Predicting and Balancing Computational Load in Hierarchical Matrix Methods", 21st Conference of Japan Computational Engineering Society", Niigata, Japan, May 31, 2016.
29. S. Okuno, T. Hiraishi, H. Nakashima, M. Yasugi, J. Sese, ``Reducing Redundant Search using Exception Handling in a Task-Parallel Language", 21st International Workshop on High-Level Parallel Programming Models and Supportive Environments, Chicago, USA, May 23, 2016.
30. H. Ibeid, R. Yokota, D. Keyes, ``A Matrix-Free Preconditioner for Elliptic Solvers Based on the Fast Multipole Method", SIAM Conference on Parallel Processing for Scientific Computing, Paris, France, April 12, 2016.

(4) Others

31. A. Ida, ``Development of H-matrices with ACA for Large-scaled BIEM Analyses", 24th Advanced Supercomputing Environment Seminar, Tokyo, Japan, December 1, 2016.