Chapter 3

Large-Scale Parallel Numerical Computing Technology Research Team

3.1 Members

- Toshiyuki Imamura (Team Leader)
- Yiyu Tan (Researcher)
- Yoshiharu Ohi (PostDoctoral Researcher)
- Yusuke Hirota (PostDoctoral Researcher)
- Daichi Mukanoki (PostDoctoral Researcher)
- Daisuke Takahashi (Senior Visiting Researcher)
- Franz Franchetti (Visiting Researcher)
- Yoshio Okamoto (Visiting Researcher)
- Takeshi Fukaya (Visiting Researcher)
- Shigeo Orii (Research Consult)
- Doru Adrian Thom Popovich (Student Trainee)
- Masaaki Aoki (Student Trainee)
- Masatoshi Takayanagi (Student Trainee)
- Ryota Kayanuma (Student Trainee)
- Koichiro Watanabe (Student Trainee)
- Yukiko Akinaga (Assistant)
- Aya Motohashi (Assistant)

3.2 Research Activities

The Large-scale Parallel Numerical Computing Technology Research Team conducts research and development of large-scale, highly parallel and high-performance numerical software for K computer. Simulation programs require various numerical techniques to solve systems of linear equations, to solve eigenvalue problems, to compute and solve non-linear equations, and to do fast Fourier transforms. In order to take advantage of the full potential of K computer, we must select pertinent algorithms and develop a software package by assembling numerical libraries based on the significant concepts of high parallelism, high performance, high
precision, resiliency, and scalability. Our primary mission is to develop and deploy highly parallelized and scalable numerical software on K computer, namely KMATHLIB. It comprises several components such as for solving

- systems of linear equations,
- eigenvalue problems,
- singular value decomposition,
- fast Fourier transforms, and
- nonlinear equations.

The K-specific topics and technical matters for emerging supercomputer systems are also our challenging works such as

- Tofu interconnect,
- parallel I/O,
- fault detection (soft-error), and
- higher accuracy computing.

We are going to complete this project through a tight collaboration among computational science (simulation), computer science (hardware and software), and numerical mathematics. Our final goal is to establish fundamental techniques to develop numerical software libraries for next generation supercomputer systems based on strong cooperation within AICS.

### 3.3 Research Results and Achievements

Following series of the annual reports from 2012-13 to 2015-16, we summarize the latest results of our running projects, mainly focused on 1) development of KMATHLIB, 2) development of EigenExa, 3) investigation of FDTD related methods, and 4) other fundamental studies to optimize the BLAS kernels through automatic parameter tuning. The plans and the publication list are also presented in the last section.

#### 3.3.1 Extension of Usability and Maintainability for Numerical Libraries

##### 3.3.1.1 KMATHLIB_API

KMATHLIB_API is a software framework to use numerical libraries via uniform interfaces, and includes large-scale, highly parallel and high-performance numerical libraries and related software for peta-scale supercomputer systems such as the K computer. Since the data structures and the usage of the numerical libraries are generally different, we must understand them to use the numerical libraries. If some numerical libraries are used in a program, data structure conversion is often necessary. Therefore, a program becomes complicated, and the cost of application development and the maintenance increase. If KMATHLIB_API is used, the data structure conversion such as the above is automatically treated by internal functions of KMATHLIB_API. KMATHLIB_API provides highly abstracted interfaces, and various numerical libraries are available by the uniform description of the program. The software package of KMATHLIB_API was released in May 2016. Accordingly, we promoted KMATHLIB at the poster session in JSIAM 2016 [29]. Also, we continue to develop about a framework function improvement of KMATHLIB_API. For the partial support from COE program, we made the tutorial materials of KMATHBLI_API, and held the lecture class of AICS public software ‘KMATHLIB’ using such materials on 28th March 2017.
3.3. Research Results and Achievements

3.3.2 Performance Optimization by Communication Avoidance and Communication Reduction

3.3.2.1 Performance Modeling by Overfitting with Non-Negative Constraints

In order to establish a methodology for the performance prediction model for large-scale parallel computing, the LP method, which introduces suppress of overfitting with non-negative constraint, is applied to predictive models of HPL running on an SGI ICE X and EigenExa running on the K computer [9]. In previous works, we examined that the LP method predicted more accurately than the lasso’s method, which is often used in the sparse modeling field. We selected the base functions as \( \{ N^3, N^2, N, N^3/\sqrt{p}, N^2/\sqrt{p}, N/\sqrt{p}, N^3/p, N^2/p, N/p \} \), and constructed a model functions represented by a linear combination of the base functions. The obtained model of EigenExa presents communication cost, especially the sum of MPI_Allreduce and MPI_Bcast, as

\[
\begin{align*}
    c_1 \frac{N^2}{\sqrt{p}} + c_2 N^3 + c_3 N.
\end{align*}
\]

The result contains unexpected factors \( N^3 \), which does not come from theoretical discussions and has no effect of parallelization. Then, it suggests that a large scale eigenvalue problem incurs performance degradation due to communication overhead.

3.3.2.2 Block-CAHTR

Communication avoidance (CA) is considered to be a promising technology to overcome the drawbacks resulting from the communication latency. We proposed CAHTR (CA-Householder Tridiagonalization) method in FY2015-16. The algorithm performs effectively on the expected situations such as computing a small problem with a number of processes. We have extended CAHTR as a block algorithm, which naturally improves parallelism and reduces communication latency.

In Fig 3.1, the statements highlighted in red require three MPI_Allreduce’s per iteration. The proposed algorithm was presented at PMAA2016 conference [19], and overview of our research activities for Communication Avoidance (CA) and Communication Hiding (CH) technique was presented at SIAM CSE2017 [24].
CHAPTER 3. LARGE-SCALE PARALLEL NUMERICAL COMPUTING TECHNOLOGY RESEARCH TEAM

3.3.2 Communication Avoiding QR

We have studied Communication Avoiding QR factorization (CAQR). Needless to say, QR factorization is a significant algorithm in a large scale matrix decomposition, and it is taken advantage in the least square approximation, orthogonalization of vectors, singular value decomposition (SVD), and computing a Moore-Penrose pseudo-inversion, et al. At the moment, ScaLAPACK is thought as a de facto standard numerical package for QR factorization on distributed parallel computers. However, it does not perform on modern multicore cluster systems, efficiently. We introduced a tiling CAQR algorithm, in which a tiling method and CAQR are combined. The algorithm is suitable for massively parallel computing because it reduces communication overhead radically. Our implementation shows better performance improvement than ScaLAPACK when the number of nodes is beyond a couple of hundreds (see Fig. 3.2).

3.3.3 More Advanced Implementation of Eigenvalue Solver

3.3.3.1 Performance Evaluation of a Quadruple Precision Eigenvalue Solver QPEigenK/QPEigenG

We have been studying on quadruple precision eigensolvers in collaboration with Japan Atomic Energy Agency. In many computational science and engineering applications, large and dense eigenvalue problems are frequently solved. As the matrix dimension increases, the accuracy of the computed eigenpairs generally becomes worse due to rounding error. Therefore, higher precision eigensolvers are required in some of the applications. However, at the start of FY2016, only double precision eigensolvers (such as PDSYEVD, PDSYGVD routines in ScaLAPACK) were available for massively parallel distributed memory parallel computers (e.g. the K computer). To overcome the problem, JAEA and we have developed a quadruple precision standard and generalized eigensolver libraries QPEigenK and QPEigenG, respectively. Based on a double precision eigensolver library EigenK, JAEA has implemented the eigensolvers of QPEigenK and QPEigenG, in which all the computations are carried out in double-double arithmetic. We have evaluated and analyzed the performance of the solvers.

We measured execution time for a various number of block sizes on the K computer. We observed that the eigensolvers in QPEigenK and QPEigenG show an excellent scalability on the K computer. The results were presented in HPC in Asia Poster Session held with an international conference ISC’16 [10].

We also carried out an analysis of the performance of the eigensolvers in QPEigenK with a performance improved quadruple precision BLAS on the K computer. We found that the increase of the block size does not accelerate the tridiagonalization (Fig. 3.3a) while it reduces the time for back transformation (Fig. 3.3b). The results were presented at international conferences [15,17] and a domestic conference [25].

3.3.3.2 Divide-and-Conquer Algorithm for Manycore Systems

Recent manycore processors (e.g. Intel Xeon Phi Knights Corner (KNC) and Knights Landing) have significantly high FLOPS. However, to run applications efficiently on the manycore systems, it is required to fully utilize a
3.3. RESEARCH RESULTS AND ACHIEVEMENTS

3.3.4 Development of High-Performance Linear Algebra Libraries on Future Architectures

In recent years, the improvement of single processor performance has reached its technological limits, and the improvement of system performance has to rely on increasing parallelism. In fact, the number of processors in a system, the number of cores in a processor, and the vector length of SIMD processors are increasing. As computer architecture becomes more complex and more parallelized, software development and code optimization will be made more challenging. Also, not only to achieve high performance, but also to support accurate, fault-tolerant, and energy-efficient computations are required toward the emerging exascale computing. Therefore, we are conducting a study for developing linear algebra software with a new design on modern many-core architectures such as GPUs and highly parallel systems.

3.3.4.1 Automatic Thread-Block Size Adjustment for Memory-Bound BLAS Kernels on GPUs

We have been studying the implementation and optimization techniques of linear algebra kernels such as BLAS on GPUs for a few years. In the previous FY, we proposed a method to determine the nearly optimal thread-block size on memory-bound BLAS kernels on GPUs. It is known that the performance of CUDA kernels often
depends on the number of threads per thread-block (thread-block size), and the optimal thread-block size often differs according to the GPU hardware running the kernel and the given data size to the kernel. Therefore, such a method to determine the optimal thread-block size is required to develop high performance CUDA kernels. In this FY, we submitted a paper about the method to an international conference and had a presentation (ATMG 2016 [5]). The results were introduced and discussed in some other conferences (poster presentation at GTC Japan 2016 [28], oral presentation at ATμWS2016 [27] and SIAM CSE17 [22]).

3.3.4.2 Reduced-Precision Floating-Point Formats

In FY2015, we proposed the implementation of reduced-precision floating-point formats that have shorter significand bit-length than IEEE standards on GPU and CPU. By eliminating waste data movement in the computation using such reduced-precision formats, we expect to save memory (cache) usage and improve the computation speed and energy efficiency. In this FY, we continued the study and had a poster presentation at an international conference (poster presentation at Cluster 2016 [11] with extended abstract). In this presentation, we showed the performance of our reduced-precision formats on some linear algebra kernels including sparse matrix-vector multiplication on three GPUs. We introduced the results in a domestic event (poster presentation at GTC Japan 2016 [28]) and a domestic workshop (oral presentation at LSPANC2017 [21]) as well.

3.3.4.3 Verified Computation for Linear Systems on Supercomputers

Although most numerical computations are performed with floating-point operations, the precision is limited, and the computation results may have errors due to rounding. Therefore, the result of floating-point operations is just an approximate solution of the problem, and in general, we are not able to know how accurate the solution is. Verified computation is a technique to give the accuracy of the approximate solution obtained by floating-point operations. However, the performance of verified computations on highly parallel computers has not been well discussed so far. We evaluated the performance of a program of the verified computation for solving linear systems $Ax = b$, developed by Dr. Morikura et al. at Waseda University, on the K computer and the FX100 system installed in RIKEN ACCC. The theoretical cost of the verified computation takes six times of that without verification (approximate computation using general floating-point operations). However, we showed that when the number of nodes is increased for a fixed size problem, the actual cost of the execution time can be about two times of that due to the difference of the performances in terms of strong-scaling. This result was presented at an international conference (oral presentation at EASIAM 2016 [4]) and a domestic conference...
Figure 3.5: FPGA diagram for Matrix-matrix multiplication

3.3.5 Hardware-based Acceleration of Numerical Linear Algebra

Numerical linear algebra has been widely applied in high performance computing to solve scientific and engineering problems. It requires computing systems to have huge computation capacity, and data throughput as problem size is increased. Although many methods have already been proposed to speed up computation through parallel programming in supercomputers or cluster systems, both memory wall and power wall problems prohibit system performance and power efficiency improvements resulting from technology scaling and instruction level parallelism. Therefore, how to reduce power consumption while increasing computation performance is one of the core concerns in current high performance computing (HPC) systems. On the other hand, hardware specialization in the form of GPUs, FPGAs, and ASICs offers a promising path toward major leaps in processing capability while achieving high energy efficiency. Particularly, with the development of semiconductor technology, FPGAs provide high energy efficiency, and have been widely applied in acceleration on computationally intensive applications, such as deep learning, data center. In this research, a low power and high performance FPGA-based accelerator for matrix-matrix multiplication is designed, and will finally extend the architecture to support other linear algebra algorithms.

Fig 3.5 shows the accelerator based on the systolic architecture, in which a processing element (PE) array was used to carry out the computation. In Fig. 3.5, the elements of matrices A and B are input into the computing engine horizontally and vertically from the block RAMs inside FPGA, and they are shifted rightward and downward cycle by cycle along computation, respectively. Each PE is a multiplier-accumulator and performs multiplications by one row and one column of matrices, and all PEs worked in parallel to perform multiplications of multiple rows and columns of matrices. Furthermore, a data reuse scheme and a novel memory hierarchy are proposed to speed up data access. The accelerator is developed through a combination of hardware/software co-design and algorithm/architecture co-design. The acceleration system with 16 × 16 PEs, and matrix multiplication up to the scale 20480 × 20480 in double precision will be finally verified and implemented by the FPGA board VC 709 from Xilinx, DE5 NET from Altera, and Intel’s HARP system. Their performance will be evaluated through comparisons with other solutions, such as many-core PC and GPU. Intel approved us to remote access to a centralized FPGA cluster based on its non-released product, which contained a 12-core Intel microprocessor and an Arria-10 FPGA in a multi-chip package.
3.3.6 Advanced Approaches for Numerical Analysis

3.3.6.1 MTDM (Meshless Time-Domain Method)

In the simulation by using the finite-difference time-domain method (FDTDM), since nodes of electric and magnetic fields have to be arranged based on an orthogonal grid, it is difficult to treat the complex shaped domain including a curved surface. It is expected that the meshless time-domain method (MTDM) which applied a meshless method to the FDTDM is effective for a simulation in the complex shaped domain. Because it is indispensable to cope a large-scale simulation, we developed the basis simulation code using the 3-D MTDM in FY2015. In FY2016, we worked on speed-up of the time-dependent part of the simulation code. Concretely, the identical calculation patterns are identified in part of the MTDM core, and we can unify them into one operation. However, since a computational cost to identify the part which performs the identical calculation is large, it did not lead to dramatic speedup. In order to realize speedup, reduction of the computational cost to identify the part which performs the same calculation is essential. Accordingly, we performed the oral presentation in NAS 2016 [30].

3.3.6.2 Parareal (Parallel in time integration)

Recently, the number of computational core increase drastically due to the improvement of many-core and parallel computation technology. It is expected that the number of computational cores becomes more than 100 million in the next-generation super computer. If recent technology is applied to the next-generation computer system, suitable calculation efficiency may not be performed. It is indispensable to develop the new parallel computing technology based on the next-generation computer system. In the present study, time is receiving the most attention as a new axis of parallelization. The article about time parallel computation was published since 50 years ago. However, it did not develop because computational accuracy was very low. The computational accuracy drastically improved by Parareal method published in 2001, the time parallel computation using Parareal method reaches the practical use level In FY2016, we investigated the basic theory of Parareal method and its application field, and presented at the auto-tuning micro workshop [31].

3.4 Schedule and Future Plan

3.4.1 KMATHLIB2 and novel Numerical Algorithms

For the studies of KMATHLIB, the first milestone, which we design and build a prototype of a generic programming framework for the scientific parallel program, was completed during FY2012-2016. Enhancing KMATHLIB, we have started a new project to develop KMATHLIB2, which is intended to be a future bridge between user API and numerical library for more flexible parallel processing, for example, concurrent of multiple thread teams. In FY2017-19, we review and investigate:

- task-based numerical libraries,
- parallel programming languages, and
- parallel runtime systems.

We are going to design KMATHLIB2 and prototype implementations. Then, we implement the whole system from FY2019 to FY2022.

For the study of each numerical algorithm, we continue to develop an implementation of eigensolver and series of communication avoiding algorithms such as CACG, CAGMRES, CAQR, 2.5D algorithms, multi-axis decomposition FFT, and so on. Also, new numerical principle to compute eigenvalue, such as the spectral divide and conquer method base on the polar-decomposition, is our challenging work to the next step of the EigenExa library.

From FY2017, we receive a new fund, which supports the research for Tensor Network (TN) scheme, and have started a new project to develop a batched SVD solver, and a high performance numerical tensor library, which contains a parallel Decker decomposition and a parallel CP decomposition.

\[ A = \sum_{r=1}^{R} \mu_r u_r^{(1)} \circ u_r^{(2)} \circ \cdots \circ u_r^{(N)} \]

The tensor decomposition has higher complexity of flops and parallelism, and consequently it would apply to usage of accelerators as well as FPGAs described in the next subsection.
3.4.2 Future Devices for Numerical Computation (GPU, Manycore and FPGA)

Recent GPU technologies are truly focused on deep-learning, and half-precision accumulator and tensor calculator are installed on a next generation NVIDIA GPU, so called Volta processor. As we already studied Intel Xeon Phi, Knight Corner, processor, new generation Knight Landing must be evaluated for numerical linear algebra and other numerical libraries. Also, another high performance accelerator of PEZY-SC2 will be coming within FY2017. Therefore, broad-range of new architecture must be investigated systematically. It might be a good theme to build a new benchmark package for GPU and MIC concerning a numerical algebra such as multi-precision BLAS kernels, multi-dimensional FFT, eigen solver, tensor decomposition, and random number generator.

For the FPGA study, the designed accelerator device will be debugged, evaluated, and extended to other linear algebra algorithms. The design tradeoffs between power and performance will be investigated and analytical models for architectural tradeoffs will be researched to provide a deep insight on energy efficient linear algebra accelerators and explore limitations in current and future architectures for energy efficiency. The performance tuning of an FPGA-based acceleration system will be discussed, and potentials of using FPGA in HPC will be examined.

3.4.3 Advanced Numerical Analysis (Parareal)

For the Parareal study, we pursue to develop a preliminary code in order to examine feasibility of the Parareal method to several types of partial differential equation. Also, we evaluate characteristics of the parallel performance on various supercomputer systems in order to promote the Parareal method in practical simulation fields, such as fluid dynamics, particle simulation, and so on.

3.5 Publications

3.5.1 Journal Articles


3.5.2 Conference Papers


3.5.3 Posters and Presentations


3.5. PUBLICATIONS


3.5.4 Patents and Deliverables


