Numerical Libraries and Software Framework for Application Programs in the Post Peta Era

Takeshi Iwashita
Information Initiative Center, Hokkaido University, Sapporo, Japan
Email: iwashita@iic.hokudai.ac.jp

I. Introduction

Recently, most of supercomputers are based on the cluster type configuration, in which a number of computational nodes are connected with an internal data transfer network. In a next generation high-end system, the computational node is expected to involve accelerators or many core processors. Under this circumstance, how can we support programmers?

First, to efficiently exploit the computational performance of the node, we should develop high performance multi-threaded kernel programs such as linear solvers, eigenvalue solvers, and matrix factorizations. These programs can be utilized as components of large-scale application programs which run on multiple nodes. They are also used in various (lab-level) applications to be conducted on PCs. Moreover, they are expected to be included in commercial software. High performance kernel programs for single computational node are one of keys of high end application program, and they have a significant impact on various numerical simulations including Big Data analyses.

Second, we also have to support programmers to intend to use multiple nodes, especially a huge number of nodes. While there are many research targets, for example, parallel languages, high performance communication libraries, in this context, we here focus on numerical libraries and software frameworks. For the distributed memory environment, the software library usually fixes the distributed data structure. Consequently, programmers have to consider the data structure of the software to be used at the initial stage of the development of their application program. This is not easy in practice. Considering this, we propose two options. One is to provide the complete simulation software package like Open FOAM for users. The other is to provide a framework for application programs for the user. The framework is developed for a specific application domain or a specific simulation method. Usually, it cannot be a single routine, and it includes various functions necessary for multi-process parallel processing, for example, data distribution and communication among the processes. By adding a couple of user subroutines and/or modifying the framework program, users develop their own simulation program.

In the followings, we introduce our research activities based on the above perspective.

II. Software framework for boundary element method and a distributed parallel H-matrices library

We have been developing a software framework for large-scale boundary element analyses in a CREST project (project leader: Prof. Kengo Nakajima, Univ. of Tokyo) promoted by JST (Japan Science and Technology Agency). The software product is named ppOpen-APPL/BEM, and is currently open to the public. The software product consists of BEM-BB framework, templates, and HACApK library shown in Fig. 1. All the components are parallelized for a distributed memory parallel computer.

A. BEM-BB framework: software framework for distributed parallel BEM analyses

The BEM-BB framework is a software framework to support large-scale BEM analyses conducted on a distributed memory parallel computer. A program code of BEM analysis generally consists of following parts: 1) Model data input part, 2) Boundary element integral operation part, 3) Boundary condition setting part, 4) Coefficient matrix and right-hand side vector generation part, 5) Linear solver, and 6) Results output part. Our software framework mainly supports the parts 1), 4), 5), and 6).

In the data input part, the framework program scatters the model data to the processes. Next, to generate the coefficient matrix and right-hand side vector, each thread independently calls the user function which describes the integral operation between boundary elements. The boundary condition settings should be done by users themselves for a general BEM analysis. The linear solver included in the software solves the arising linear system of equations. The linear solver is parallelized in the hybrid parallel processing model (MPI and OpenMP). In the final step, the simulation result is outputted to a file.

Currently, we have two implementations for the framework. One implementation is based on dense matrix computations, which is oriented to a general purpose BEM analyses. The other one uses H-matrix method to reduce the simulation time drastically.

Moreover, for some specific analysis domains, the software provides users with the template (BEM-BB template). The template program includes the user function for the integration and the boundary condition setting. By using the template with the framework, a user can execute a parallel

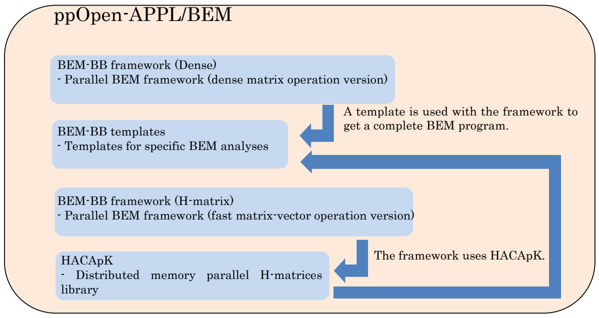

Figure 1. Overview of the ppOpen-APPL/BEM software

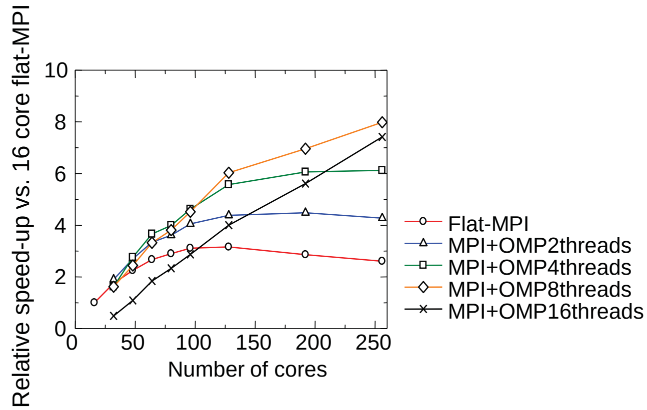

Figure 2. The relative speedup of H-matrix vector multiplication in HACApK library

BEM analysis with a minimum or no programming cost. Currently, the template for a static electric field analysis is available.

B. HACApK: distributed memory parallel H-matrices library

The HACApK is a numerical library for computations on a hierarchical matrix (H-matrix) which is an approximated representation of a dense matrix arising from an integral equation method such as BEM [1]. In the H-matrix technique, the matrix is divided into a set of submatrices. In typical BEM analyses, a significant part of these submatrices can be approximated by low-rank matrices without losing numerical accuracy of the computations with the matrix. While there are several approximation method for the submatrices, the HACApK uses the most basic method: the adaptive cross approximation (ACA). In an ideal case, introducing H-matrix with ACA reduces the complexity from $O(n^2)$ of a dense matrix to $O(n \log n)$, where n denotes the dimension of the matrix.

The HACApK library includes routines for the H-matrix construction and the H-matrix vector multiplication. Both routines are parallelized in the hybrid parallel programming model. Consequently, the HACApk library can be also used as a multi-threaded library for various application programs. The operation of the library have been confirmed in static electric filed analyses and earthquake cycle simulations run on x86 clusters and Fujitsu FX10. In these test operations, the developed library exhibited a better sequential performance than the Hlib which is the most popular H-matrix library, and also showed the effectiveness of the hybrid parallel programming model (Fig. 2).

III. High performance multi-threaded sparse triangular solver

We have studied about multi-tread parallelization of sparse triangular solver for more than 10 years. The sparse triangular solver is an important computational kernel involved in various linear solvers, such as Gauss-Seidel method (smoother), SOR method, ICCG method. In the study, we have mainly investigated parallel ordering techniques in the context of parallelization of the ICCG method. Consequently, we proposed (algebraic) block multi-color ordering to achieve fast convergence, a high cache hit ratio, and good parallel scalability. It is shown that the proposed technique is more effective than the conventional multi-color ordering [2]. Moreover, when compared with the additive Schwarz type parallelization method, the proposed technique is advantageous on a many-core processor, in which a number of threads are used. Park et al. address that our technique is critical for efficient implementation of symmetric Gauss-Seidel smoother on Xeon Phi in their SC14 paper about HPCG benchmark [3]. Currently, we are preparing multi-threaded implementation of ICCG solver based on block multi-color ordering which is open to the public.

References

[1] A. Ida, T. Iwashita, T. Mifune, and Y. Takahashi, “Parallel Hierarchical Matrices with Adaptive Cross Approximation on Symmetric Multiprocessing Clusters,” Journal of Information Processing, Vol. 22, (2014), pp. 642-650.

[2] T. Iwashita, H. Nakashima, and Y. Takahashi, “Algebraic Block Multi-Color Ordering Method for Parallel Multi-Threaded Sparse Triangular Solver in ICCG Method,” Proc. IPDPS2012, (2012).

[3] J. Park, M. Smelyanskiy, K. Vaidyanathan, A. Heinecke, D. D. Kalamkar, X. Liu, Md. M. A. Patwary, Y. Lu, P. Dubey, “Efficient Shared-Memory Implementation of High-Performance Conjugate Gradient Benchmark and Its Application to Unstructured Matrices,” Proc. SC14, (2014).