Challenges of Run-time Auto-tuning for Sparse Iterative Solvers

Takahiro Katagiri[1]

[1] Information Technology Center, The University of Tokyo,
2-11-16 Yayoi, Bunkyo-ku, Tokyo 113-8658, JAPAN.
katagiri@cc.u-tokyo.ac.jp

Abstract. In this position talk, the challenges of run-time auto-tuning (AT) for sparse iterative solvers are discussed. The challenges are summarized at the meta solver level, inter solver level, and inner solver level. In addition, standardization of the AT interface is discussed.

Keywords: Run-time Auto-tuning, Sparse Iterative Solver, Standardization, OpenATLib and Xabclib

1 Introduction

High performance is not easily achievable with current computer architectures. Deep hierarchical and non-uniform accesses of memory hinder execution performance. In addition, the best algorithm and its implementation depend heavily on input data, such as non-zero element distribution of a sparse matrix. At the library level, it is difficult to find the best algorithm without input matrix information. Many projects hence focus on run-time auto-tuning (AT).

2 Challenges of Run-time Auto-tuning on Sparse Iterative Solvers

AT for sparse iterative solvers, especially for routines based on the Krylov subspace method, require the following AT functions at the Meta, Inter and Inner levels:

Meta Solver Level

A) To target optimization, a policy description from end-users, such as (i) execution speed, (ii) memory space, (iii) number of CPUs (or computer resource selection), and (iv) computation accuracy. Conventional AT libraries only focus on (i). To extend the AT library, the function of policy description is needed.

B) Specification for AT timings. For example: one time before calling a routine, frequently performing, and every time when calling a routine.

C) End-user hint interface. For example, the interface of sparse matrix information from the end-user description.

D) Selection of sparse iterative algorithms. For example, the GMRES(*m*) method or IDR(*s*) method for linear equations solvers.

Inter Solver Level

E) High-performance sparse BLAS implementations. Especially, SpMxV (Sparse Matrix-Vector Multiplication) is crucial. In addition, to unroll the depth selection of SpMxV, implementation selection is also important. A selection of dedicated kernels, such as the 7-point stencil, is desirable.

F) Selection of implementation of principal linear algebra computation. For example, multiple implementation of Gram-Schmidt orthogonalization.

Inner Solver Level

G) Tuning for dedicated parameters, such as the restart frequency, of the sparse iterative algorithm.

H) Preconditioner selection, such as ILU or SPAI. Corresponding parameters of the preconditioner, such as the fill-in depth or dropping tolerance, are also tuning parameters.

I) Selection of sparse data format and its transformation. For example, CRS or JDS. The overhead of transformation should be considered in AT time.

Another issue to establish run-time AT is standardization related to the above AT functions. The issue is summarized as follows.

Standardization

J) AT functions should be standardized as an API. This is crucial to establish re-usability of the AT functions.

K) The data format of the AT database, including the AT log, should be standardized. This enables the reduction of total AT time.

L) The solver interface, especially in a parallel solver, should be standardized with respect to real application usage. This is outside the AT function issue. However, it is very important to establish usability of the library. Many applications include physical mesh information to perform preconditioning. The library interface, including AT functions, should be considered with the mesh information.

We are now developing run-time AT functions and an API library for an iterative solver in the Xabclib project [1]. A challenge in this project is to establish a numerical policy function for AT, as described in A) (ii)-(iv). In addition, we are designing standardization of the API for AT via OpenATLib [2], which is an API library. Some interfaces of OpenATLib will be explained in this position talk**.**

Acknowledgments. This work is partially supported by a Grant-in-Aid for Scientific Research (B) “Development of Auto-tuning Specification Language Towards Manycore and Massively Parallel Processing Era,” No. 21300007. I would like to thank all members of the AT library group of “Seamless and Highly-Productive Parallel Programming Environment for High-Performance Computing,” MEXT, Japan, for daily discussion of the AT study.

References

1. Xabclib Project, http://www.abc-lib.org/Xabclib/index.html
2. Katagiri, T., Sakurai, T., Kuroda, H., Naono, K., and Nakajima, K., Xabclib: An Iterative Solver with a General Auto-tuning Interface "OpenATLib" , International Workshop on Peta-Scale Computing Programming Environment, Languages and Tools (WPSE 2010) (2010) In oral presentation.