HPC Refactoring with Hierarchical Abstractions to Help Software Evolution

Hiroyuki Takizawa, Ryusuke Egawa, Daisuke Takahashi, and Reiji Suda

Abstract This article briefly introduces the concept of our new research project, JST CREST "An Evolutionary Approach to Construction of a Software Development Environment for Massively-Parallel Computing Systems." Since high-performance computing system architectures are going to change drastically, existing application programs will need to evolve for adapting to the new-generation systems. Motivated by this, our project will explore an effective methodology to support the programming for software evolution of valuable existing applications, and also develop a programming framework to bridge the gap between system generations and thereby to encourage migration of existing applications to the new systems. The programming framework will provide abstractions of complicated system configurations at multiple levels, and refactoring tools to help evolving applications to use the abstractions.

H. Takizawa (✉)
Graduate School of Information Sciences, Tohoku University/JST CREST, 6-6-01 Aramaki-aza-aoba, Aoba, Sendai 980-8579, Japan
e-mail: tacky@isc.tohoku.ac.jp

R. Egawa
Cyberscience Center, Tohoku University/JST CREST, 6-3 Aramaki-aza-aoba, Aoba, Sendai 980-8578, Japan
e-mail: egawa@isc.tohoku.ac.jp

D. Takahashi
Faculty of Engineering, Information and Systems, University of Tsukuba/JST CREST, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
e-mail: daisuke@cs.tsukuba.ac.jp

R. Suda
Graduate School of Information Science and Technology, The University of Tokyo/JST CREST, 7-3-1 Hongo, Bunkyo, Tokyo 113-8656, Japan
e-mail: reiji@is.s.u-tokyo.ac.jp

M. Resch et al. (eds.), *Sustained Simulation Performance 2012*,
DOI 10.1007/978-3-642-32454-3_3,

1 Instruction

In conventional HPC software development, the top priority is always given to performance. Lower-level programming may allow an application program to achieve a higher performance by thoroughly specializing the application code for a particular target system. However, low-level programming forces an application programmer to significantly modify the code whenever the target system changes to a new one. As a result, it is difficult to evolve existing computational science applications so as to adapt to future-generation systems, which will be massively-parallel and heterogeneous. Motivated by this, we have started a research project named "An Evolutionary Approach to Construction of a Software Development Environment for Massively-Parallel Computing Systems," which aims to support HPC software evolution adapting to system changes.

The goal of this 5.5-year project supported by JST CREST is to appropriately abstract the increasing hardware complexity of massively parallel heterogeneous computing systems, and thereby to enable computational science applications to adapt easily to new systems. This project emphasizes incremental development of existing applications and continuous software development. Therefore, the project will develop abstraction technologies so as to hide the gap between current and future systems as much as possible.

For supporting software evolution, various abstraction technologies are needed. Since we already have a huge number of valuable applications and it is impossible to completely rewrite their codes, we need to incrementally evolve them based on incremental improvement of existing programming models such as MPI and OpenMP. On the other hand, high-level abstraction is a very powerful tool to facilitate software evolution because it can hide the implementation details that are likely to be system-specific and hence major impediments to software evolution. Therefore, we will develop hierarchical abstraction layers by the following three approaches. One approach is to provide evolutionary programming models and their programming interface for massively parallel heterogeneous systems. Another is to develop numerical libraries as one high-level abstraction layer to achieve a high performance without considering the underlying system hardware. The other is to use domain-specific knowledge to build another high-level abstraction layer in order to ease application development in computational science.

We will also design a new concept of *HPC refactoring* to migrate existing application programs to new ones, which use the above hierarchical abstraction layers. Many research projects have proposed high-level descriptions of computational science applications to realize the automatic/semi-automatic translation from high-level codes to optimized low-level codes. On the other hand, software evolution in this project assumes that low-level codes already exist. The existing codes are optimized usually at a low level for current systems, not for future systems. Therefore, we first need to help migrating the existing codes to high-level ones, and then the high-level ones will be used for future maintenance and evolution while keeping their performances. The migration support, HPC refactoring, is one of the

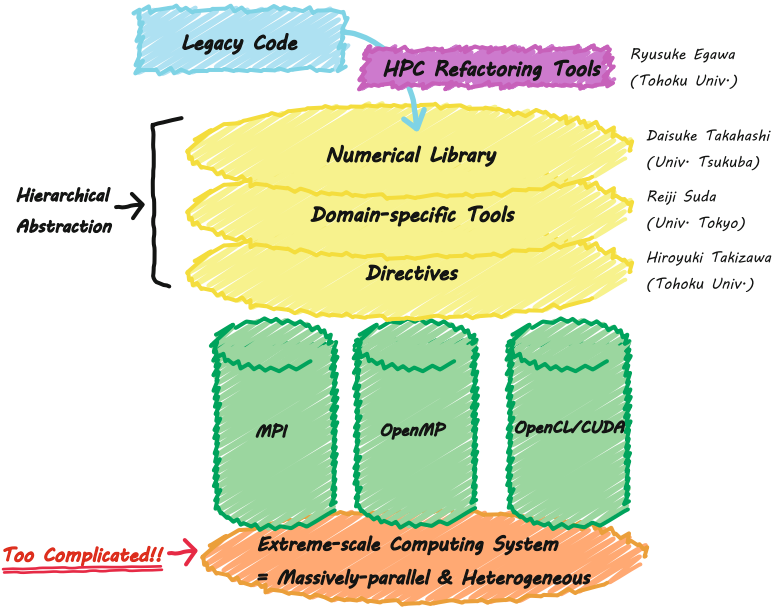

Fig. 1 Overview of the research project. As the hardware configuration of post Petascale computing systems is too complicated, this project will develop hierarchical abstraction layers to facilitate software development and future maintenance. In addition, we will establish a new concept of "HPC refactoring" for smooth migration of existing applications to the abstracted programming environment

most important features characterizing this project. We will integrate the techniques developed in this project into a programming framework, called *Xevoler*.

Since October 2011, we have started developing the above abstraction layers, and also designing the initial version of HPC refactoring catalog, which is the guideline of software evolution under the assumption of using the abstraction layers. In addition, we had a kick-off meeting and created a wiki page for project members as the infrastructure for our collaborative work.

Figure 1 shows the overview of the research project. The project team consists of the following four groups:

- Takizawa group:
  - Programming interface for managing system heterogeneity
  - Customizable refactoring tools and directives
- Takahashi group:
  - Numerical libraries to fully exploit the system performance
  - Fault tolerance and mixed-precision computation

- Suda group:
  - Domain-specific knowledge for extreme parallelization
  - Algorithm/data-structure translation for strong scaling
- Egawa group
  - Cataloging common patterns in software evolution
  - Design methodology for post-Petascale applications

In the followings, we introduce the research topics of each group, and then briefly describe their research progress in Fiscal Year 2011 (FY2011).

2 Programming Models and HPC Refactoring Tools

We discuss the expected difficulties in software development for future computing systems by considering a computing system of CPUs and GPUs as a prototype of future systems. Then, we will develop programming interfaces such as compiler directives to describe effective and efficient collaboration among many different processors. Programming models will also be designed so as to reduce the number of system-dependent parameters decided by programmers, and hence improve the code and performance portabilities of GPU computing applications.

In FY2011, we discussed the concept and future direction of this project with many researchers [1, 2]. We also developed a data dependency analysis tool [3] and a performance analysis tool [4] to help code manipulation by programmers for HPC refactoring. In addition, we developed and evaluated the mechanisms for improving the system dependability [5] and for the cache locality [6]. Those mechanisms will be key technologies for effective use of massively parallel heterogeneous systems.

3 Numerical Libraries for Heterogeneous Computing Systems

In this project, we will develop libraries of Fast Fourier Transform (FFT), Algebraic Multi-Grid (AMG), and mixed-precision basic linear algebra subprograms (BLAS). Although many large-scale applications in computational science internally use numerical libraries, most of the existing libraries are not designed considering future mainstream systems that are massively-parallel and heterogeneous. Thus, it is necessary to develop numerical libraries that can exploit the potential of massively-parallel heterogeneous systems such as large-scale GPU clusters.

In FY2011, we explored an effective implementation scheme of FFT library and prototyped a library for preliminary evaluation on a multi-core cluster system. We also considered the basic design of AMG library for GPU systems [7]. In addition, we prototyped a triple-precision BLAS library and evaluated the performance [8].

4 Use of Domain-Specific Knowledge

We explore software development methodology for parallel applications in computational science from the following two viewpoints. One is focusing on parallelization methods, and the other is on numerical calculation methods.

In FY2011, from the former viewpoint, we have developed a method to reduce collective communications in the conjugate gradient (CG) method [9]. In a standard CG method, collective communications are required twice in one iteration. However, the proposed method called the k-skip CG method needs only one collective communication in $k + 1$ iterations. Although the proposed method needs more computation and is less stable than the standard CG method, its computational complexity is less than the methods in the related work. In addition, we proposed three techniques to reduce the branch divergence, considering the importance of exploiting SIMD parallelism in future systems due to the power efficiency [10]. Those techniques will be applied to the application programs developed by Takahashi group.

From the latter viewpoint, we proposed a method to minimize the number of trials required for a Monte Carlo simulation of optimizing design parameters [11]. We also proposed a new high-order difference formula of fractional calculus [12], which is often used in the field of engineering but whose numerical method is not established yet. Moreover, we analyzed the error of QR update algorithm that can quickly solve linear least-squares problems but accumulates the errors. Then, we proposed a method to restart the update algorithm when the accumulated error exceeds a certain threshold. To explore the application design methodology in the massively-parallel heterogeneous computing era, we started analyzing important application programs in nano-science and bio-science.

In addition, for developing refactoring tools, we surveyed existing technologies in software engineering, programming models, language processing systems, and integrated development environment.

5 Design of HPC Refactoring

We are designing an HPC refactoring catalog by porting the existing applications to various platforms whose successors will potentially become the building blocks of future systems [13]. In FY2011, we have analyzed real application codes used in Tohoku University Cyberscience Center [14], surveyed code maintenance technologies in HPC software development [13], and discussed the format of HPC refactoring catalog. In those activities, we gathered the contents that should be described in the initial version of HPC refactoring catalog.

We also interviewed application programmers to collect opinions that help design of a practical HPC refactoring catalog. Furthermore, by optimizing and parallelizing existing applications, we developed optimization techniques to efficiently use the performance of parallel heterogeneous systems.

6 Conclusions

In this article, we have introduced our new research project for adapting existing applications to new-generation computing systems. In this project, we are developing various abstraction techniques to hide the hardware complexity, and also designing HPC refactoring to help migrating existing application programs to the abstracted programming environment. We will integrate these technologies into a programming framework, named *Xevolver*. Using the framework, we will help evolving various computational science applications in a systematic way.

Acknowledgements The authors would like to thank Prof. Michael Resch of HLRS, Prof. Wenmei W. Hwu of UIUC, and Prof. Chisachi Kato of the University of Tokyo for their valuable comments on this project. The authors would also like to thank Prof. Hiroaki Kobayashi of Tohoku University for constructive discussions.

This work is supported by JST CREST "An Evolutionary Approach to Construction of a Software Development Environment for Massively-Parallel Computing Systems."

References

1. Takizawa, H.: "A new research project for enabling evolution of legacy code into massively-parallel heterogeneous computing applications," The 14th Teraflop Workshop, Stuttgart, Dec 5 (2012).
2. Takizawa, H.: "How can we help software evolution for post-Peta scale computing and beyond?," The 2nd AICS symposium, Kobe, Mar 2 (2012).
3. Sato, K., Komatsu, K., Takizawa, H. and Kobayashi, H.: "A Runtime Dependency Analysis Method for Task Parallelization of OpenCL Programs," IPSJ Transactions on Advanced Computing Systems(ACS), Vol.5 No.1, pp.53–67 (2011).
4. Kanda, H., Okuyama, T., Ino, F. and Hagihara, K.: "An Instrumentation Method for Analyzing Efficiency of Memory Access in CUDA Programs," IPSJ SIG Notes 2012-HPC-133(3), 1–8, Mar 26 (2012).
5. Amrizal, M.A., Sato, K., Komatsu, K., Takizawa, H. and Kobayashi, H.: "Evaluation of a Scalable Checkpointing Mechanism for Heterogeneous Computing Systems," presentation at IPSJ Tohoku Branch Workshop, Mar 2 (2012).
6. Sugimoto, Y., Ino, F. and Hagihara, K.: "Improving Cache Locality for Ray Casting with CUDA," Proc. 25th Int'l Conf. Architecture of Computing Systems Workshops, 339–350, Feb 29 (2012).
7. Takahashi, K., Fujii, A. and Tanaka, T.: "Multiple GPUs-based AMG Method," IPSJ SIG Notes 2012-HPC-133(29), 1–7, Mar 19 (2012).
8. Mukunoki, D. and Takahashi, D.: "Implementation and Evaluation of Triple Precision BLAS Subroutines on GPUs," The 13th Workshop on Parallel and Distributed Scientific and Engineering Computing (PDSEC-12), May 25 (2012).
9. Motoya, T. and Suda, R.: "k-skip Conjugate Gradient Methods: Communication Avoiding Iterative Symmetric Positive Definite Sparse Linear Solver For Large Scale Parallel Computings," IPSJ SIG Tech. Rep. 2012-HPC-133(30), Mar. 27 (2012), in Japanese.
10. Kato, S., Suda, R. and Tamada, Y.: "Optimization Techniques for Reducing Branch Divergence on GPUs," IPSJ SIG Tech. Rep. 2012-HPC-134(5), Jun. 1 (2012), in Japanese.
11. Suda, R. and Nittoor, V.S.: "Efficient Monte Carlo Optimization with ATMathCoreLib," IPSJ SIG Tech. Rep. 2012-HPC-133(21), Mar. 27 (2012).

12. Takeuchi, Y. and Suda, R.: “New numerical computation formula and error analysis of some existing formulae in fractional derivatives and integrals,” The 5th IFAC Symposium on Fractional Differentiation and its Applications (FDA’12), Keynote, May 15 (2012).
13. Egawa, R.: “Designing a Refactoring Catalog for HPC,” The 15th Workshop on Sustained Simulation Performance, Sendai, Mar 23 (2012).
14. Komatsu, K., Soga, T., Egawa, R., Takizawa, H., Kobayashi, H., Takahashi, H., Sasaki, D. and Nakahashi, K.: “Performance Evaluation of BCM on Various Supercomputing Systems,” In 24th International Conference on Parallel Computational Fluid Dynamics, pages 11–12 (2012).