An Automatic Performance Tracking System for Large-Scale Numerical Applications

Shoichi Hirasawa, Hiroyuki Takizawa and Hiroaki Kobayashi

Abstract In this work, we propose an Automatic Performance Tracking System for analyzing the changes in execution performance and finding the source code modifications that cause the degradation of performance portability. The proposed system works in order to support evolving a large-scale numerical application while maintaining its performance portability across multiple target computing systems. By evaluating the performance of an application on every computing system, the proposed system helps application developers find the source code modifications that degrade the execution performance on a computing system. The proposed system also retrieves multiple versions of an application from its code repository, and automatically executes them on a newly added computing system. As a result, application developers are able to analyze how the source code modifications in the past affect the performance on the new computing system. Based on the evaluation results, the application developers can review the source code changes to improve the performance portability of the HPC application through the system.

1 Introduction

Multiple types of computing systems and tool chains are widely used these days. High-performance computing (HPC) applications sometimes need to migrate to new target computing systems because of their long software life cycles. The burden of migrating HPC applications to new target computing systems is usually heavy because of the large code sizes of such applications.

To alleviate the heavy cost of the migration, the code of an application should be maintained in such a way as to be able to execute in high performance on multiple

S. Hirasawa (✉) · H. Takizawa · H. Kobayashi
Graduate School of Information Sciences, Tohoku University, Sendai, Japan
e-mail: hirasawa@sc.cc.tohoku.ac.jp

H. Takizawa
e-mail: takizawa@tohoku.ac.jp

H. Kobayashi
e-mail: koba@tohoku.ac.jp

M.M. Resch et al. (eds.), *Sustained Simulation Performance 2016*,
DOI 10.1007/978-3-319-46735-1_10

computing systems. In this work, the capability of an HPC application to achieve high performance on different types of computing systems is defined as *performance portability*. If an application code has high performance portability, it is expected to easily migrate the application to a new target computing system.

HPC applications are usually optimized only for a small number of computing systems to increase their execution performances. When optimizations are applied to an application with consideration only for specific computing systems, the execution performance on other types of computing systems may degrade. As a result, optimization efforts taking a long time for a small number of computing systems may lead to degrading performance portability of the application.

Generally, code optimizations for a specific computing system may degrade the performance of the application on another system. Thus, it is necessary to prevent applying such optimizations so as to keep the performance portability high. The degradation of performance portability can be detected by finding out performance degradation of the application on a computing system. To find out the performance degradation, execution performance on every computing system needs to be obtained, tracked, and compared. Although unit testing frameworks [1] and automatic bug detection methods [2, 3] have been proposed, to the best of our knowledge, there is no performance tracking system to maintain high performance portability of HPC applications.

In this work, an Automatic Performance Tracking System (APTS) that supports maintaining high performance portability of HPC applications is designed and developed. The APTS finds the changes of a source code that decrease the performance portability of the application. Because of the complexity of current computing systems, it is difficult to model and predict the execution performances of HPC applications on their target computing systems. With the APTS, execution performances of applications are obtained by actually executing them on every target computing system.

This paper is organized as follows. Section 2 discusses the performance portability of HPC applications. Section 3 proposes the APTS. Section 4 evaluates the APTS and Sect. 5 provides conclusions and future work.

2 Performance Portability of HPC Applications

In this work, computer systems used for developing and executing HPC applications are categorized into the following three types. Figure 1 shows the computer systems that are considered to be used in developing, building, and executing the applications.

1. Development systems: computer systems that are used to edit application source codes.
2. Building systems: computer systems that are used to compile source codes and build execution binaries of applications.
3. Execution systems: computer systems that are used to execute applications.

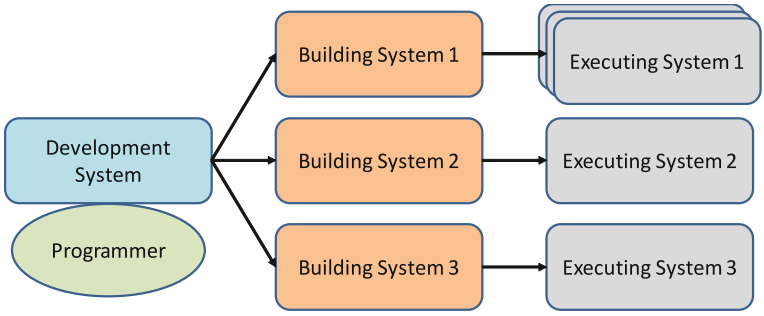

Fig. 1 Target computers for application development

When a programmer builds an application on the same system as the development system, the building environment such as compiling tool chains needs to be installed on the development system. Multiple building environments need to be installed on the development system when the application is developed considering multiple execution systems. However, installing all building environments on the development system is not always possible especially when software licences for production compilers are not available to programmers. In this work, therefore, an environment of multiple building systems and one development system is supposed to be used for developing applications. In the following subsections, programmer's burdens that can potentially be reduced by using tools are discussed.

2.1 Finding the Cause of Performance Degradation on Execution Systems

The source code of an HPC application tends to be large because many algorithms and optimizations are sometimes added during its long software life cycles. When a programmer finds a performance degradation of such an application on an execution system, the programmer needs to find the cause of degrading the performance of the application from the source code. The burden is usually heavy because of the large size of the source code of the application. To alleviate the burden, a system that notifies the cause of performance degradation on an execution system is useful.

Additionally, new execution systems are sometimes added to the target execution systems of an application because the life cycles of HPC applications tend to be longer than those of current execution systems. The programmer needs to know code modifications in the past to find the cause of performance degradation when a new execution system is added to the target systems.

2.2 Application Performance on Execution Systems

It is usually difficult to statically predict execution performance of an application on an execution system. It is because current execution systems have become too complex to create accurate performance models. As a result, an application needs to be executed on the target execution systems to evaluate the execution performance.

To evaluate execution performance, the source code needs to be built on the corresponding building systems of the target execution systems because execution performance of an HPC application depends on compilers and their optimization flags. With multiple development systems, programmers need to build and execute applications while editing one source code multiple times on the development systems.

2.3 Source Codes Synchronization Among Multiple Building Systems

When multiple building systems are used in addition to the development system as in Fig. 1, source codes need to be synchronized among them. Currently, programmers need to synchronize them manually with multiple tools while editing the source code of the application. This task is tedious and error-prone. Therefore, a tool that automatically synchronizes source codes among the development system and building systems can potentially ease the burden.

3 An Automatic Performance Tracking System

In this work, an Automatic Performance Tracking System (APTS) is proposed. The APTS is executed on the development system. In this work, it is assumed that a target application already has a build script and also an execution script with input data. Therefore, using the scripts, the APTS can build and execute the application on every execution system for performance evaluation and result verification.

3.1 Overview of the Automatic Performance Tracking System

The APTS tracks the changes of execution performance along with the modifications on an application code. Execution performances are automatically profiled by actually executing the application on its target execution systems. With the execution performances, the APTS finds the source code modifications that are causes of degrading the execution performance on an execution system. A programmer is able to use the found source code modifications to improve the performance portability

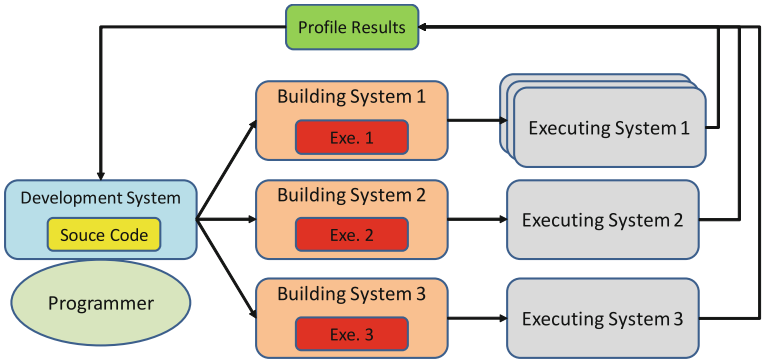

Fig. 2 Development framework of the APTS

of the application. For a programmer, it is easy to migrate an application to a new execution system if the application has high performance portability. The APTS has the following functions to help a programmer develop an application while keeping high performance portability (Fig. 2).

1. A function to track the changes of execution performance along with source code modifications.
2. A function to automatically build and execute applications on their target execution systems.
3. A function to automatically synchronize source codes among all building systems.

3.2 Performance Tracking Function Along with Source Code Modifications

While developing and optimizing an application, the modification to improve the execution performance on one execution system may degrade the execution performance on another execution system. When a new system is added to the execution systems of an application, the execution performance on the new system might be too low compared to its peak performance. In such a case, the low execution performance might be due to a certain source code modification for performance optimization in the past. Note that the new execution system was not available at the time, and the programmer could not check if the modification degrades the performance until the new system becomes available. Therefore, the programmer is required to check if every source code modification in the past degrades the performances on the available execution systems by tracking the past source code modifications whenever a new system becomes available.

The APTS uses version controlling systems such as CVS [4] or Git [5] to automatically find the performance changes with the past source code modifications on

the newly added execution system. When a new system is added to the execution systems, the APTS automatically retrieves the past source codes of the application from the version controlling system. Then, the APTS automatically profiles the execution performance on the new execution system for every past version of the source code. By comparing the execution performances of two neighboring versions, performance degradation on the execution system can be detected. With the neighboring version numbers, the programmer can inspect an actual cause of the performance degradation on the new execution system.

4 Evaluation of the Automatic Performance Tracking System

4.1 Evaluation Setups of Finding the Cause of Degrading Performance portability

In this evaluation, the APTS is implemented as a plug-in program of the Eclipse integrated development environment (Eclipse IDE). It is implemented with the Plug-in Development Environment (PDE), which is the standard development framework of plug-in programs for the Eclipse. `Eclipse 4.2.1 Build id:20121004-1855` is used for developing and executing the APTS.

The source codes synchronization function is implemented with the Secure Copy (scp) command. `OpenSSH_6.1p1` is used in the APTS. To build the source codes of the target application, `Makefile` and `make` command are used. The executable file of the application is launched using the Secure Shell (ssh) command on the target execution systems. The `time` command is used to obtain the execution performances.

The APTS is evaluated to check if it is able to find the source code modifications that are the causes to degrade the performance portability of an application. A real HPC application of the entire growth process of binary alloy nanopowders in thermal plasma synthesis [6] is used in this evaluation. Three building systems are used from one development system. All building systems are installed in Cyberscience Center of Tohoku University. The development system, which is a desktop PC (Intel Core i7-3930K 3.2 GHz, 16 GB Memory, SSD), is installed in another building of mechanical engineering in the same campus of Tohoku University. The specifications of building systems are shown in Table 1. Server 1, 2 and 3 are also used as execution systems corresponding to the building systems.

4.2 *Results of Analysing the Degradation of the Performance Portability*

The evaluation results are shown in Fig. 3. The horizontal axis indicates version numbers of the target application. The vertical axis on the left-hand side shows the speedup ratio from the execution time of the application code Version 1 running on Server 1. The vertical axis on the right-hand side shows the number of modified source code lines between a neighboring two versions.

The application has been optimized for Server 1 along with the version numbers. Hence, the performance of Server 1 increases with the version number. On the other hand, the performance degrades by changing from Version 5 to 6 on Server 2. The Tesla C2070 GPU of Server 2 is newer than the Tesla C1060 GPU of Server 1. From these results, it is observed that the change from Version 5 to 6 degrades the performance portability of the application.

As the execution performance does not degrade on Server 3, which has a newer K20 GPU than C2070, the modification between Versions 5 and 6 only degrades the execution performance on Server 2 among the three. The number of different source code lines between Versions 5 and 6 is 37.

In the evaluation results, it is shown that the APTS is able to limit the number of source code lines that cause the performance degradation on execution systems. In this particular evaluation, the APTS can successfully reduce the number of source

Table 1 Specifications of building systems and execution systems

System name	Linux ver.	CPU	GPU	CUDA
Server 1	2.6.18	Core i7 920 2.67GHz	Tesla C1060	5.0
Server 2	2.6.32	Core i7 930 2.8GHz	Tesla C2070	5.0
Server 3	2.6.18	Core i7 920 2.67GHz	Tesla K20c	5.0

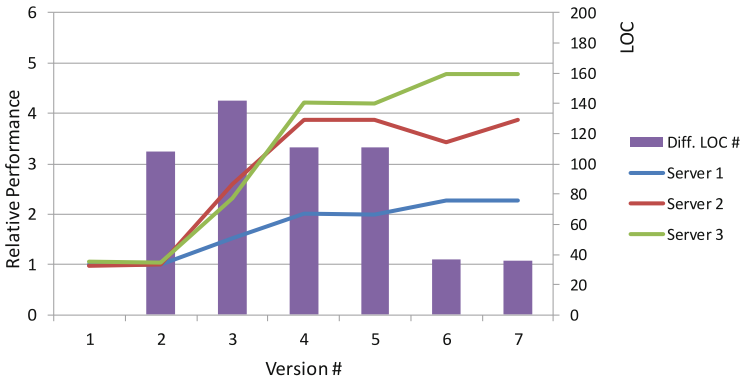

Fig. 3 Execution performances of application versions and line numbers of corresponding code difference

code lines that cause the performance degradation to 37. As a result, the APTS is able to support programmers to develop a large-scale application with high performance portability.

5 Conclusions and Future Work

In this paper, we have designed and implemented an Automatic Performance Tracking System (APTS). It automatically finds the version of an application, from which the performance is degraded on an execution system. It is implemented as a plug-in program of the Eclipse IDE. It has a function of transferring necessary files of an application to build machines. It then builds and executes the application to collect its execution performance on each execution system.

The APTS supports the development work of programmers by alleviating the burden of building and executing the application on multiple systems. It finds the version of an application code that degrades the execution performance on an execution system. As a result, the APTS helps a programmer maintain high performance portability of an application by keeping the execution performance high on multiple execution systems.

With the evaluation results, it has been shown that the APTS can successfully find the version of the real application that degrades the execution performance on an execution system. It has also been shown that the APTS can obtain the execution performances of the application on multiple execution systems by transferring and building the application on multiple building systems. With these functions, the manual work of performance evaluation necessary for programmers is automated and, as a result, the APTS is able to support the development work on maintaining high performance portability of HPC applications.

Realizing functions such as automatically evaluating the performance with profilers such as gprof and nvprof, obtaining the performance profile results, and reasoning the codes that degrade performance portability by providing the profiling results to the programmers are parts of our future work. Supporting batch queuing systems for executing applications on HPC computing systems is also important. Migrating the implementation for the code base of PTP [7] is also considered to provide the information of execution performance in the editor.

Acknowledgements The authors would like to thank Prof. Shigeta of Osaka University for allowing us to use the application. This work is partially supported by JST CREST “An Evolutionary Approach to Construction of a Software Development Environment for Massively-Parallel Heterogeneous Systems.”

References

1. Zhu, H., Hall, P.A.V., May, J.H.R.: Software unit test coverage and adequacy. ACM Comput. Surv. **29**(4), 366–427 (1997)
2. Kim, S., Zimmermann, T., Pan, K., Whitehead, E.J.: Automatic identification of bug-introducing changes. In: 21st IEEE/ACM International Conference on Automated Software Engineering, 2006. ASE '06, pp. 81–90 (2006)
3. Williams, C.C., Hollingsworth, J.K.: Automatic mining of source code repositories to improve bug finding techniques. IEEE Trans. Soft. Eng. **31**(6), 466–480 (2005)
4. http://cvs.nongnu.org/. Cvs - concurrent versions system
5. http://gitscm.com/. Git - the fast version control system
6. Shigetam, M., Watanabe, T.: Growth model of binary alloy nanopowders for thermal plasma synthesis. J. Appl. Phys. **108**(4), 043306–043306–15 (2010)
7. Watson, G.R., Rasmussen, C.E., Tibbitts, B.R.: An integrated approach to improving the parallel application development process. In: IEEE International Symposium on Parallel Distributed Processing, 2009. IPDPS 2009, pp. 1–8, May 2009