FEHA: an Adaptive Web-Based Front-End Environment to Support Hands-On Training in Parallel Programming

Syunji Yazaki[1], Takeshi Kikuchi[2], Hideaki Tsuchiya[3], Hiroaki Ishihata[4]

Abstract

In this paper, we propose an adaptive web-based programming environment called Front-end Environment for Hands-on Activities (FEHA) for parallel-processing lectures. We designed FEHA to utilize the remaining system resources of existing clusters for education. To meet this objective, we employed an agentless design that does not use cluster-side agent program to control the cluster. FEHA uses only two common UNIX commands: "ssh" and "rsync". Because of this specification FEHA can adaptively utilize computation resources of any cluster systems. We used FEHA in two lectures at a university. Students learned fundamental parallel-programming methodologies in the lectures. In the first lecture, 108 students submitted 1,658 programs through FEHA while 116 students submitted 1,648 programs in the second lecture. We also conducted a questionnaire survey to evaluate the usability of FEHA by using a web usability scale. The results showed that most students have a positive impression for the FEHA design including the web user interface. In particular, the students evaluated the operational stability of FEHA and reported that FEHA is easy to use. From the results, we confirmed that FEHA contributes to making parallel programming lectures more effective and easy to learn.

1. Introduction

Parallel processing technology has become essential in computer science. Many applications and cloud service infrastructures have been parallelized to achieve better performance and quality of service. Thus, training and education for parallel processing technology are required in the information technology industry. However, preparing a parallel programming lecture is a challenge because such a programming environment usually requires a computer cluster with sufficient compute nodes, network, and software. Some academic organizations own cluster systems for research projects. However, utilizing these systems for lectures is difficult because the systems lack a user-friendly interface for learners.

Programming environments for parallel programming beginners have been developed [1]. These systems provide a user-friendly integrated development environment (IDE) for the learners. Some systems also have features that collect learning activities and summarize them to provide appropriate feedback to the learners [2]–[4].

One concern regarding systems is the cost of maintenance. The instructors have to ensure both the parallel programming platform (back end) and user interface application (front end) to conduct the lectures. In particular, a parallel programming environment requires higher hardware and software costs compared to a nonparallel programing environment.

With this background, we developed an adaptive web-based front-end environment for parallel programming training called Front-end Environment for Hands-on Activities (FEHA). FEHA provides a user-friendly programming interface for lectures. It is designed to utilize the existing parallel programming environment without any modifications in the environment. In this paper, we will describe the outline of FEHA. We will also report a case of application of FEHA in a parallel programming class at a university.

2. Related work

Some systems that support programming lectures have been developed. Moodle [5] are typical learning management systems (LMSs) that are widely used in education. Virtual programming lab (VPL) [1] is a web-based programming environment that is implemented as a Moodle plug-in. The VPL has a feature called "Jail" that enables a closed environment to run programs coded by the learners safely. It also provides a web-based integrated development environment (web IDE).

[1] The University of Electro-Communications, Japan
[2] Tokyo University of Technology, Japan
[3] The University of Electro-Communications, Japan
[4] Tokyo University of Technology, Japan

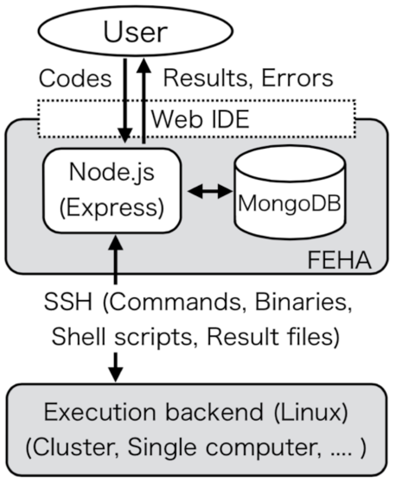

User
Codes
Results, Errors
Web IDE
Node.js (Express)
MongoDB
FEHA
SSH (Commands, Binaries, Shell scripts, Result files)
Execution backend (Linux) (Cluster, Single computer, ...)

Fig. 1. Construction of FEHA.

A few web-based tutoring systems such as Codecademy exist [2]. They are designed for learning typical coding techniques through a step-by-step tutorial. Thus, preventing the learners from writing arbitrary codes.

Paolo et al. developed an incremental hint system for programming exercises [6]. The hint system displays a series of hints according to the requests from a student. The hints are created in advance based on example source codes.

3. Implementation

3.1. Construction and design principle

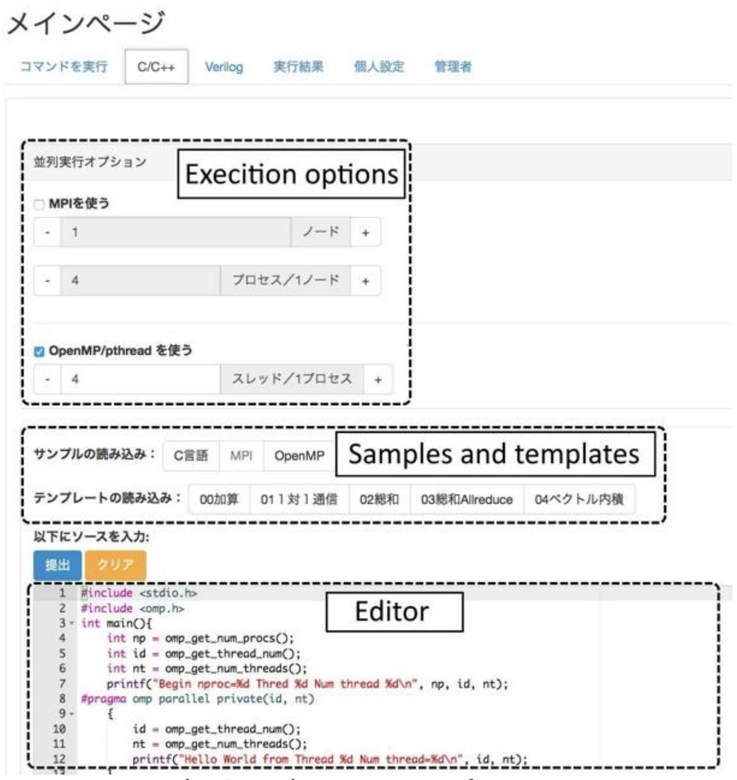
Fig. 1 shows the construction of FEHA. It provides a web IDE especially for parallel programming lectures. The students write their codes in the web IDE. FEHA compiles the codes and runs them on a UNIX-based cluster system (back end). Execution results are stored in the database on FEHA.

As introduced in Section 2, some web-based programming environments have already been proposed. The main difference between FEHA and existing programming environments is that FEHA only works as a web user interface for existing UNIX-based parallel computing clusters. In contrast, existing systems implement their own program execution environment. The motivation for the FEHA is to utilize the remaining system resources of existing clusters for education. Some academic organizations or research groups have their own cluster systems. However, computational resource of a system is not fully utilized in many cases.

For ease of setup and maintenance, we employed MongoDB, Express, AngularJS, and Node.js (MEAN) stack for implementation of FEHA. MEAN recorded a relatively high performance to recent web application frameworks [7]. We can deploy and maintain FEHA efficiently, and can run it even on small server machines using the MEAN stack.

One important principle of the FEHA implementation is that it is agentless. FEHA should work for many types of cluster. Most of them restrict installation of the software and use of TCP/UDP ports. Thus the use of a cluster-side agent program, such as UNIX daemon, should be avoided. To satisfy this requirement, we only use two common UNIX commands, “ssh” and “rsync,” to drive the cluster from FEHA. The commands “ssh” and “rsync” are used to run submitted programs on the cluster and for incremental file syncing between FEHA and the cluster, respectively.

From the point of view of security, the back end may be damaged by unexpected behavior of the submitted program. To avoid this, FEHA builds a shell script, which wraps the submitted program. In the shell script, any runtime restriction performed with the “limit” or “ulimit” UNIX commands can be set for each program exaction. FEHA enables this runtime restriction according to the FEHA administrator settings.



☐ MPIを使う
- 1 ノード +
- 4 プロセス/1ノード +

☑ OpenMP/pthread を使う
- 4 スレッド/1プロセス +

```
#include <stdio.h>
#include <omp.h>
int main(){
    int np = omp_get_num_procs();
    int id = omp_get_thread_num();
    int nt = omp_get_num_threads();
    printf("Begin nproc=%d Thred %d Num thread %d\n", np, id, nt);
#pragma omp parallel private(id, nt)
    {
        id = omp_get_thread_num();
        nt = omp_get_num_threads();
        printf("Hello World from Thread %d Num thread=%d\n", id, nt);
```

Fig. 2. Web IDE screen of FEHA.

3.2. Programming environment

Fig. 2 shows a section of the web IDE screen. It consists of an execution option form, buttons for samples and templates, and an editor form. Users can select the options to run multi-thread (Pthread and OpenMP) and message passing interface (MPI) programs, if the corresponding back end support exists. Sample codes and code templates are provided to support the training. The editor form is implemented using JavaScript, which works on all major web browsers. Further, the user can select one of the MPI libraries, such as MPICH or MVAPICH, which are setup at the back end.

In order to support the lectures, the web IDE screen has buttons that load sample codes or code templates. The difference between the sample codes and code templates is that the samples are codes that are ready to run without modification, whereas the templates are pieces of code that must be modified. A user can load and run the samples to understand the usage of the web IDE. Then, the instructor can utilize the templates to teach the technology.

FEHA has a hint function that shows two types of information as hints to the students: precautions and solutions. Programming beginners frequently make errors. Most of these errors, such as general compilation errors, are not related to learning of the technology. This function helps both students and instructor by reducing these general errors.

The precautions are tips to avoid minor problems. Numerous types of errors occur during the programming. However, based on our preliminary research, five types of error constitute 96% of the total number of errors that occur during a programming class. Hence, FEHA displays these precautions to the student.

The solution is to provide information to avoid general errors. We collect and analyze compilation errors from a lecture that has been conducted earlier. Then, we create a database that contains a list of compilation errors and the corresponding solutions. If the user makes a compilation error, FEHA displays the corresponding solutions in the IDE.

4. Evaluation

4.1. Experimental overview

We used FEHA in two lectures at a university. The students learned fundamental parallel programming methodologies in the lectures. We taught Pthread and OpenMP in the first and second lecture, respectively. Pthread is an implementation of the POSIX thread library. It is a fundamental library for carrying out shared-memory multi-processing on a UNIX system. OpenMP is an application programming interface for using shared-memory multi-processing easily.

In the first and second lectures, 108 and 116 students used FEHA, respectively. The students in both lectures were in the same class. There were a total of 1,658 and 1,648 submissions during the first and second lectures, respectively.
We used a desktop PC (AMD Athlon II X2 220 processor, 1.7 GB DDR3 memory, 1 GigE network) to run FEHA. As the back end, we used our own computer cluster that was already in use for another research project. During the lectures, we did not observe any significant delay in request processing.
We also conducted a questionnaire survey to evaluate the usability of FEHA according to a web usability scale (WUS). The WUS evaluates the usability of web site using seven factors: favorability, usefulness, reliability, layout, operability, visibility, and responsibility. It consists of 21 questions. Each factor is evaluated from three questions. The subjects rate each question on a scale of 1 (poor) to 5 (good). We took averages of scores for each factor. We also asked additional questions for further analysis.

4.2. WUS questionnaire result

A total of 66 and 51 students answered the WUS questionnaires in the first and second lectures, respectively. Table 1 shows a summary of the WUS average scores.
As shown in the table, FEHA achieved greater than three points for all factors. This means that most students have a positive impression of the FEHA design, including the web user interface. In particular, the factors “Reliability” and “Operability” achieved higher grades than the other factors. This indicates that the students rated the operational stability of FEHA and felt that FEHA is easy to use.
From the results, we confirmed that FEHA contributes to making parallel programming lectures more effective and easy to learn.

5. Conclusion

In this paper, we propose an adaptive web-based programming environment called the FEHA for parallel processing lectures. Parallel processing technology has become essential in computer science. However, preparing parallel programming lectures is a challenge because such a programming environment usually requires several expensive hardware and software components. To tackle this problem, we designed FEHA to utilize the remaining system resources of existing clusters for education.
FEHA provides a web IDE for programming lectures. The IDE includes an editor, buttons for selecting options for parallel computing, code snippets, and a hint window for programming beginners.
One important design principle of FEHA implementation is that it is agentless. FEHA works for several different types of clusters. Most of these restrict the installation of software and use of TCP/UDP ports. Thus, FEHA only uses two common UNIX commands, “ssh” and “rsync,” to drive the cluster.
We used FEHA in two lectures at a university. The students learned fundamental parallel programming methodologies in the lectures. In the first lecture, 108 students submitted 1,658 programs through FEFA, while 116 students submitted 1,648 programs in the second lecture.
We also conducted a questionnaire survey to evaluate the usability of FEHA using WUS. The results showed that most students have a positive impression for the FEHA design, including the web user interface. In particular, the students evaluated the operational stability of FEHA and felt that FEHA is easy to use. From the results, we confirmed that FEHA contributes to making parallel programming lectures more effective and easy to learn.

Acknowledgements
A part of this study was supported by KAKENHI grants (26330143 and 25330146).

Table 1. Summary of WUS average scores for FEHA in two lectures.

	Favorability	Usefulness	Reliability	Layout	Operability	Visibility	Responsibility
1st lecture	3.1	3.1	3.3	3.2	3.4	3.1	3.1
2nd lecture	3.3	3.2	3.5	3.3	3.4	3.4	3.1

References

[1] J. C. Rodríguez-del-Pino, “VPL, Viertual Programming lab for Moodle.” [Online]. Available: http://vpl.dis.ulpgc.es/. [Accessed: 14-Apr-2016].
[2] T. Wang, X. Su, P. Ma, Y. Wang, and K. Wang, “Ability-training-oriented Automated Assessment in Introductory Programming Course,” *Comput. Educ.*, vol. 56, no. 1, pp. 220–226, 2011.

[3] H. Hiroaki, M. Kazuki., T. Kakafumi, H. Atsuo, and K. Seiichi., “An Environment for Collecting Fine-Grained Development Records to Help with Programming Exercise,” in *Advanced Applied Informatics (IIAIAAI), 2014 IIAI 3rd International Conference on*, 2014, pp. 739–744.
[4] E. L. Glassman, J. Scott, R. Singh, P. Guo, and R. Miller, “OverCode: Visualizing Variation in Student Solutions to Programming Problems at Scale,” in *Proceedings of the Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology*, 2014, pp. 129–130.
[5] MoodleHQ, “Moodle.” [Online]. Available: https://moodle.org/. [Accessed: 14-Apr-2016].
[6] P. Antonucci, C. Estler, D. Nikolić, M. Piccioni, and B. Meyer, “An Incremental Hint System For Automated Programming Assignments,” in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, 2015, pp. 320–325.
[7] TechEmpower, “Web Framework Benchmarks, Round 9.” .