Vol.2015-HPC-148 No.25
2015/3/3

ハンズオン講義・デモンストレーションを支援する計算機システム向けフロントエンド環境の提案 -並列処理の講義への適用-

矢崎 俊志[1,a)]　石畑 宏明[2,b)]

概要：本論文では，簡易演習や単発的に行われる講習会での体験活動を支援するため，環境構築が容易でかつ様々な目的に転用できる再利用性が高い計算機向けフロントエンド環境 FEHA (Front-end Environment for Hands-on Activities) を提案する．FEHA は，プログラム等を実行する実行バックエンドとして既存の計算機システムを活用することで，高い再利用性を実現する．FEHA と実行バックエンドとの連携に SSH (Secure SHell) を用いることで，多くの Linux ベースシステムのフロントエンドとして利用可能である．利用者が提出したコマンドやプログラムをシェルスクリプト経由で実行することで，簡易的なセキュリティ機能も実現する．FEHA の初期実装を，大学における並列処理の講義で試用した．講義では 50 名強の学生が FEHA を利用して OpenMP と MPI を体験した．個人用 PC 程度の性能を持つサーバを用いて単一プロセスで実行した FEHA により，10 分間の最大で 500 回，講義時間中の合計で 1,000 回を超えるリクエストを，1 リクエストあたり 1 秒程度で処理できることを示した．

1. はじめに

学習において，学習者自身による体験活動は，技術や知識を効率良く習得，理解するために有効な手段である．計算機分野においては，技術展示，セミナー，チュートリアル等の講習や高等教育における講義などで，デモンストレーションや演習を通じた体験活動を行う．

デモンストレーションや演習を実施するための環境の整備には時間とコストがかかる．多くの場合は，各学習者が自身の環境整備を行わなければならない．既存の計算機システムを利用する場合でも，システム管理者が個々の学習者に対して適切なアクセス権などを設定する必要がある．

この問題に対する 1 つのアプローチとして，ウェブサービスを活用する方法がある．ウェブベースの e-Learning システムは，学習に必要な基本機能を有しており，それらを学習者に対して個別かつ任意のタイミングで提供することができる．e-Learning システムの一部には，プログラミングなどの演習環境を実装したものもある [1]．これを利用することで，時間的な制約を受けにくい演習環境を構築することができる．

実運用されている e-Learning システムは，学習環境を提供するだけでなく，学習課程を管理する Learning Management System (LMS) として提供されているものもある．近年では，LMS を用いて不特定多数の利用者に教育サービスを提供する，Massive Open Online Course (MOOC) [2], [3] と呼ばれるサービスも多数存在する．LMS とは異なり，主にプログラミング言語の習得を目的とした学習サイトも多数存在する [4], [5]．これらのサイトは，特定の言語を対象として，プログラミング手法の解説や仮想プログラミング環境を提供する．

e-Learning システムや学習サイトは，完結した学習環境を利用者が好きなときに利用できる点において有用である．しかし，e-Learning システムの構築自体に手間と時間がかかるため，講義の合間のわずかな時間で行う簡易演習や，イベントなどで単発的に行われる講習会やデモンストレーションに利用することは手間に見合わない場合もある．既に構築されたクラウドサービスを利用する場合でも，利用規模に応じた事前契約や，人数にほぼ比例した費用が必要となる．また，構築したシステムを長期間運用することを考えた場合，利用者支援やコンテンツ開発などを行うために，定常的な運用体制の構築も必要になる．学習サイトは不特定多数向けに固定された学習コンテンツを提供するものであり，学習環境提供者や学習者の多様な志向に対応す

[1] 電気通信大学
1–5–1 Chofugaoka, Chofu, Tokyo 182–8585, Japan
[2] 東京工科大学
1404–1 Katakuramachi, Hachioji, Tokyo 192–0982, Japan
a) yazaki.syunji@uec.ac.jp
b) ishihata@sft.teu.ac.jp

ることは難しい．

以上の背景から，本論文では，簡易演習や単発的に行われる講習会での体験活動を支援するため，環境構築が容易でかつ様々な目的に転用できる再利用性が高い計算機向けフロントエンド環境 FEHA (Front-end Environment for Hands-on Activities) を提案する．

2. 関連研究

e-Learning システムは，古くから研究・開発が進められ，近年では，Intelligent Tutoring System (ITS) や LMS としての利用を志向して，多くの技術が開発されている．Blackboard（旧 WebCT）や Moodle[6] などは，世界中の様々な教育機関で利用されている代表的な LMS である．Moodle は現在もオープンソースとして提供されており，様々な形でカスタマイズされ，利用されている．プログラミングの演習を目的として一般公開されている Moodle 用プラグインとして Virtual Programming Lab (VPL) [1] がある．VPL は Web IDE と，それで作成されたプログラムの安全な実行環境を提供する．Jail と呼ばれる独自のプログラム実行環境により，システムが破壊される恐れが少ないプログラム実行環境を提供している．Glassman らは，MOOC として行われたオンライン講義で提出された大量のコードを元に，講師に類似の解法（コード）を提示することで，学習者に良い解法をフィードバックすることを助けるシステム OverCode を開発している [7].

e-Learning システムとしては実装されていないが，講師が学習者に効率的なフィードバックを与えることを支援する技術の開発も行われている．井垣らは，課題の開始，コードの編集，コンパイル，実行などを記録する Web IDE を開発し，その記録を可視化することで，講師に学習者への適切な支援を促すシステム C3PV を開発した [8]．Hashiura らは，同様の目的で，コードの編集操作を記録する Eclipse Plug-in を開発した [9]．長谷川らは，Java ベースのリアルタイムプログラミング演習環境 IDISS を開発した [10]．IDISS は，講義中にリアルタイムで課題の提示を行う．さらに，提出された Java コードに対してコーディングスタイル，クラス定義の正確性，コンパイルの可不可，ユニットテストなどの評価を行い，その結果をフィードバックする．Wang らは学習者のプログラミング習熟度の応じて，自動的に課題を提示するシステム AutoLEP を開発した [11]．既存の自動課題提示システムが，正解したプログラム数をもとに課題の自動提示を行うのに対して，AutoLEP は文法，構造，意味，テスト結果の全てを考慮することで，より学習者に適した課題を提示する．

3. システム設計

3.1 構想

FEHA は，実際の製品開発や研究で利用されている環境やアプリケーションを簡単に体験できる環境を提供する．FEHA の設計方針を**図 1** にまとめる．専門家 (Specialists) は実システム (Actual system) 上の様々な環境を駆使して製品開発や研究を行う．経験豊富な専門家は，計算機そのものの扱いに長けてる場合が多いため，初学者 (Biginners) に比べ，新しい技術や環境の活用方法を容易に習得することができる．一方で，計算機そのものの扱いに習熟してない初学者は，実際に体験したい技術や環境を試すまでの準備に時間がかかる．一般的なプログラミング言語などは，LMS などが提供する仮想学習環境 (VLE, Virtual Learning Environment) で体験することも可能である．スクリプト言語やウェブ技術には，このような環境が用意されているものが多い．

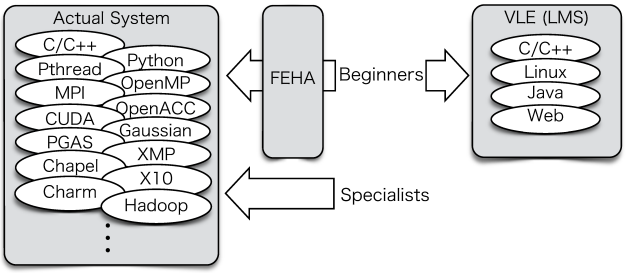

図 1 FEHA の基本構想．専門家 (Specialists) は実システム (Actual system) 上の様々な環境を直接自由に扱うことができる．初学者 (Beginners) は，仮想学習環境 (VLE, Virtual Learning Environment) で基本的な技術を習得する．FEHA を使うことで，初学者であっても，実システム上の環境を簡単に素早く体験することができる．

Fig. 1 The concept of FEHA. Specialists can utilize various environments on an actual system. Beginners can learn technologies in the VLE (Virtual Learning Environment). The beginners also can easily and quickly experience various environments on the actual system by using FEHA.

VLE は簡単な演習環境を提供できる反面，学習環境が実システムと大きく異る場合もある．例えば，演習で作成したプログラムのコードが，実システムでは実行できない場合もある．また，VLE に最新技術や環境を追加実装することが難しい場合もある．

FEHA は，実システムを簡単に利用するためのインタフェースのみを提供し，LMS のように総合的な学習環境を提供しない．これにより，初学者であっても，実システム上の環境を簡単に素早く体験する仕組みを提供する．その用途についても，LMS のように，利用者に対して閉じた VLE を任意の時間に利用できるようにするものではなく，直接指導による，実地での演習やデモンストレーションでの利用を前提とする．

3.2 実装

環境構築が容易でかつ様々な目的に転用できる再利用性

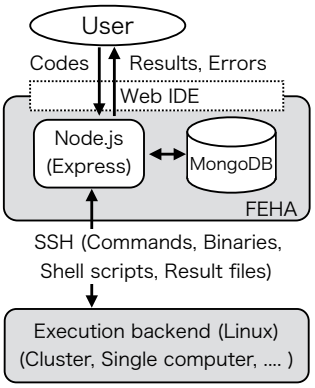

図 2 FEHA の構成．Node.js により Web IDE を提供するデータベースには MongoDB を用いる．学習者は Web IDE でコードを作成し，FEHA はそれをコンパイルし，実行バックエンドで実行する．実行結果は，Web を通じて利用者に返される．実行履歴は，データベースに記録される．

Fig. 2 Configuration of FEHA. Node.js is used to provide web UI. MongoDB is also used as the data base. The learner writes their codes on the web IDE. FEHA compile the codes and run them on the execution backend. Execution results will be returned to the learner through the web. The histories will be recorded in the data base.

が高いシステムを実現するため，次の要件を満たすシステムを実装する．

- 利用者の計算機環境に大きく依存しない学習環境を提供する
- 設置・撤去・再設置が容易である
- わずかなコードの変更で様々な新技術の体験に応用できる

この要件を考慮して，FEHA を**図 2** で示すように構成する．利用者の計算機環境に大きく依存しない利用を実現するため，演習環境として Web IDE を提供する．Web IDE は，Node.js および，Web application framework の Express で実装した．Node.js は軽量なサーバサイドプログラムの開発基盤である．特徴として，多数の HTTP リクエストを単一プロセス・スレッドにより非同期で実行する．既存のマルチプロセス・スレッドを用いたウェブサーバと比べ，比較的少ない計算資源で効率的にリクエストを処理できることが期待できる．実用されている Web application framework の中では，比較的高い性能を達成している [12]．データベースには，Node.js との連携に実績がある MongoDB を用いる．

設置・撤去・再設置を容易にするため，プログラム等を実行する仕組みを FEHA 内部に独自に実装しない．代わりに，外部の実行バックエンド (Execution backend) を利用する．実行バックエンドには，既に運用されている計算機クラスタや PC を想定する．システムの提供者が普段使っている計算機環境をそのまま実行バックエンドとして流用することで，演習のために特別なシステムを準備する手間を省く．OS は Linux のみを対象とする．実行バックエンドがバッチジョブシステムを利用している場合，バッチジョブシステム経由でプログラムを実行することを選択できる．バッチジョブシステムとしては，一般的なスーパーコンピューターやクラスタシステムに広く利用されている SLURM と Torque をサポートする．なお，Node.js，MongoDB，実行バックエンドは全て同じホストに設置することも，全て異なるホストに設置することも可能である．

データコピーおよびコマンドやプログラムの実行など，FEHA と実行バックエンドの連携には，SSH (Secure SHell) を用いる．SSH はほぼすべての Linux で利用可能であり，リモートアクセスのための実質的な標準となっている．この仕組により，FEHA はわずかなコードの変更で，様々な計算機システムのフロントエンドとして機能することができる．

利用者が不正なコードを実行する可能性もある．これに対応するため，既存の LMS では，VPL[1] のように，独自のサンドボックス機能を実装している．一方，FEHA においては，その利用形態として直接指導を想定しており，利用者の行動を講師側がある程度直接監視できる前提で設計する．このことを踏まえ，FEHA では，独自のサンドボックス機能を実装しない．ただし，シェルの機能を用いた簡易的なセキュリティ対策を実装する．FEHA では，利用者が提出したプログラムやコマンドはすべてシェルスクリプト経由で実行される．そこで，シェルに設定可能な limit や ulimit などの計算資源の割り当て制限機能を用いて，破壊的なプログラムやコマンドの実行を制限する．また，どの利用者がどようなプログラムを実行したのかは，すべてログとして記録し，追跡可能にする．

3.3 リクエスト処理手続き

FEHA が利用者から受け取る主なリクエストはプログラム実行リクエストと実行結果取得リクエストである．

FEHA では，プログラムやコマンドの実行がリクエストされた場合は，次の流れで処理を行う．

(1) **ファイル生成**：ユーザの入力から，必要に応じてソースファイルと実行用シェルスクリプトを生成する
(2) **リモートファイルコピー**：ソースファイルと実行用シェルスクリプトを実行バックエンドに転送する
(3) **ビルドとデータベース書き込み**：実行バックエンドでソースファイルをビルドする．ビルド結果とエラーをデータベースに書き込む．ビルドが失敗した場合は，以降の処理を行わない
(4) **リモートファイルコピー**：ビルドしたバイナリファイルを FEHA に転送する
(5) **実行とデータベース書き込み**：実行用シェルスクリプトで，バイナリファイルまたはコマンドを実行する．バッチジョブシステムを利用する場合は，実行用シェ

ルスクリプトをバッチジョブに提出する．ソースファイル，スクリプトファイル，バイナリファイルをデータベースに記録する

ファイル生成とデータベース書き込み以外の一連の処理はSSH により実行される．手続きの中で，リモートファイルコピーが 2 回発生する．一連の処理の中で，これらボトルネックとなることが予想される．FEHA では通信時間を短縮するため，Rsync over SSH による差分コピーを行う．また，SSH を効率的に実行するため，OpenSSH の共有コネクション機能を利用することを前提としている．ビルドにおいて，処理が失敗した場合には，エラー内容がデータベースに保存され，処理はそこで中断される．

FEHA による実行結果取得の流れは次の通りである．

(1) **リモートファイルコピー**：実行バックエンドから FEHA に実行結果ファイルをコピーする
(2) **データベース書き込み**：実行結果ファイルを解析し，実行が終了していればその結果をデータベース書き込む

実行結果の取得について，FEHA では，実行結果を一度ファイルに書き出す．次に，利用者自身が実行結果取得リクエストを発行すると，そのファイルがデータベースに取り込まれる．この仕様は，バッチジョブシステムとの連携を考慮したものである．一般的なバッチジョブシステムは，実行結果として，標準入出力をファイルに書き出して利用者に返す．これに合わせて，FEHA では，バッチジョブシステムを利用しない場合でも，ファイルを介した実行結果の取得を行う．

4. 機能

ここでは，現時点における FEHA のプロトタイプ実装が持つ機能を説明する．機能は主に，一般ユーザ機能，管理者機能，ユーザ管理機能に分けられる．

4.1 一般ユーザ機能

FEHA が提供する Web IDE の一般ユーザ向け画面の例として，C/C++のコーディングを行う画面を**図 3** に示す．Web IDE では，ウェブブラウザ上で動作する JavaScript で実装されたエディタを用い，コマンドやコードを入力する機能を提供する．実行バックエンドが対応していれば，プログラム実行のオプションとして，Pthread，OpenMP，MPI (Message Passing Interface) ライブラリを用いたビルドや実行を指示することができる．現時点では MPI ライブラリとして，MPICH および MVAPICH を切り替えることができ，それに合わせて，ネットワークも Ethernet か InfiniBand のどちらを使うか選択することができる．C/C++言語以外の演習として，現在の実装では，Verilog HDL の実行もサポートしている．

円滑な演習を支援するため，サンプルやテンプレートを

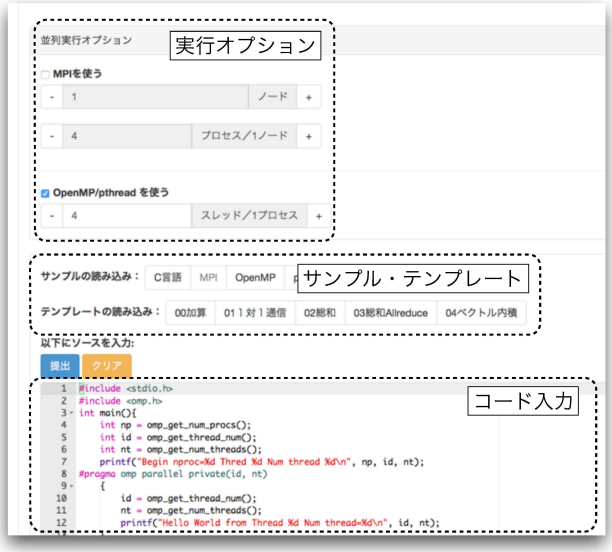

図 3 C/C++コーディング画面例．実行オプションとして，Pthread，OpenMP，MPI の利用と，その時のスレッド・プロセス数を指定することができる（上部）．講師側からサンプルやテンプレートを提供するこもできる（中部）．コード入力には JavaScript で実装されたエディタを用いる（下部）．

Fig. 3 An example screen for C/C++ coding. Users can request use of Pthread, OpenMP, and MPI (Upper). Samples and templates are provided for effective training (Medium). The editor which is implemented by JavaScript is used for coding.

提供する機能を持つ．サンプルとテンプレートの違いは，完動する記述であるかそうでないかにある．サンプルは，演習開始後すぐに利用者にコードを実行してもらうことで，演習の意図や FEHA の使い方を把握してもらうために利用する．テンプレートは記述の一部が欠けたコードで，利用者に簡単な課題を提供するために利用する．どちらも講師側で自由に変更することができる．

図 4 に実行結果の表示画面例を示す．提出された記述は，第 3.2 節で述べた手続きにしたがって処理され，画面中の実行結果更新ボタンの押下により，その結果一覧が利用者に提示される．画面の左側には，これまでの提出履歴が一覧表示されている．確認したい履歴を選択することで，画面右側に，提出された記述およびビルドや実行時の標準入出力が表示される．

4.2 管理者機能

FEHA では，システム管理者と講師は同じであることを想定している．システム管理者は原則として設定ファイルの変更とサーバプロセスの再起動によって FEHA の挙動を変更する．ただし，サンプルとテンプレートは講義中に書き換えられることを想定し，無停止で更新することができるように実装されている．一部の設定は Web IDE 上にその項目が表示され，それを操作することによって変更す

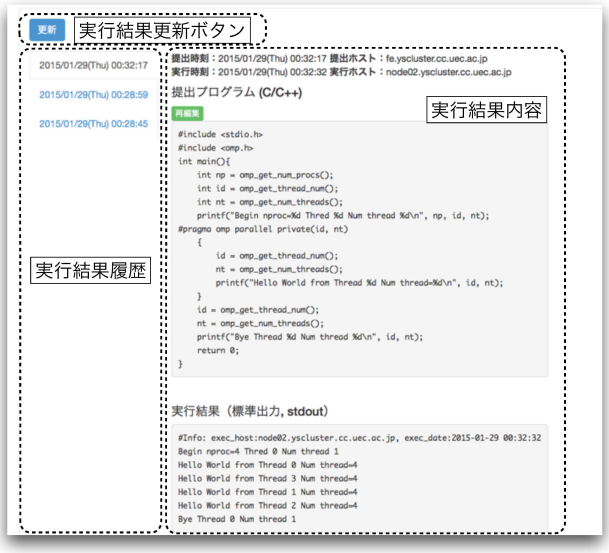

図 4 実行結果確認画面例．左側に実行履歴の一覧，右側にその内容を表示する．

Fig. 4 An example screen for showing execution results. The left part shows a list to select a record to be shown. The right part is showing the detail of the selected result.

ることができる．現時点において，Web 上の管理者機能としては，任意のユーザのパスワードを変更する機能と利用者を一覧する機能が実装されている．

4.3 ユーザ管理機能

管理者側ができるだけユーザ管理を行わないようにユーザ管理機能を設計した．これは，数百人を対象とした講義などでは，事前登録が難しいことを考慮したものである．利用にあたり，一般ユーザは自身でメールアドレスに基づく登録操作を行う．利用登録にあたり，一般ユーザの身元をある程度担保したい場合は，システムの設定により，メールによる簡易的な本人確認を行うこともできる．

5. 試用実験

FEHA を大学における並列処理の講義に適用し，その実現可能性を確認する実験を行った．東京工科大学で 2014 年度後期に開講された「並列処理」の講義において，FEHA のプロトタイプ実装を試用した．この講義は，並列処理に関するハードウェア・ソフトウェア技術を紹介する内容である．2013 年度以前は，座学のみの講義であったが，今回より，FEHA を用いて並列処理のソフトウェア技術として，OpenMP と MPI の体験演習を行った．体験では，サンプルおよびテンプレートを講義の進行状況に応じて随時提供した．並列処理技術は，学生が自分で環境を準備することが難しいものも多く，簡単にプログラムを書いて実行してみる，という体験をすることが難しいものの 1 つである．多くの場合は既存のクラスタシステムや，場合によっては大規模なスーパーコンピューターシステムを利用することがあるが，利用には，システムに対するある程度の熟練が必要であるため，FEHA の適用としては妥当である．

FEHA を動作させるサーバとして，PRIMERGY MX130 S2 (AMD Athlon II X2 220 2.8GHz 2 Cores, 2GB DDR3 800 MHz) を用いた．Node.js は基本的に単一プロセスで動作するが，複数のプロセスでクラスタリングによるロードバランシングなどを行うこともできる．しかし，今回の演習では，単一プロセスによる処理能力を確認するため，1 プロセスのみで FEHA を起動した．

実行バックエンドとして，筆者らが別用途で既に運用しているクラスタシステムを再利用した．このシステムは，4 ノード構成で，各ノードは 8 Cores (2 Sockets) の CPU と，DDR3 32GB のメモリを持つ．また，ネットワークとして InfiniBand QDR と 1 Gbps Ethernet を持つ．FEHA の利用において，実行バックエンド側に必要な設定は FEHA 用のユーザアカウントの作成である．このアカウントによる SSH で，FEHA からのリクエストを実行する．バッチジョブシステムとして Torque を採用しているため，これを利用する．

OpenMP の体験演習では，56 名の学生が FEHA を利用した．この演習において，全ての学生がはじめて FEHA を利用した．実行リクエスト数が最も多かった 10 分間において，163 回の実行リクエストと 352 回の実行結果取得リクエストがあった．それぞれのリクエストにおいて，当該時間における平均リクエスト間隔はそれぞれ 3.7 秒と 1.7 秒であった．1 リクエストの実行に要した時間の平均は，それぞれ 1.34 秒，0.37 秒であった．演習一連では，549 回の実行リクエストと 1,247 回の実行結果取得リクエストを滞り無く処理することができた．

MPI の体験演習では，OpenMP の演習から約 1 ヶ月後に，55 名の学生が FEHA を利用した．OpenMP の体験演習と同様に，実行リクエスト数が最も多かった 10 分間において，76 回の実行リクエストと 217 回の実行結果取得リクエストがあった．それぞれのリクエストにおいて，当該時間における平均リクエスト間隔はそれぞれ 7.89 秒と 2.76 秒であった．1 リクエストの実行に要した時間の平均は，それぞれ 1.31 秒，0.57 秒であった．演習一連では，175 回の実行リクエストと 534 回の実行結果取得リクエストを滞り無く処理することができた．

6. 考察

FEHA と実行バックエンド間の制御やデータ転送を全て SSH で行う点について，遅延によりリクエストの実行時間が長くなる懸念があった．しかし，実際は 10 分間の最大で 500 回，講義時間中の合計で 1,000 回を超えるリクエストを 1 リクエストあたり 1 秒程度の時間で処理することができた．このことから，大学での 50 人規模の講義において，実用性に問題が無いことが確認できた．セキュリティ

に関しては，何人か無限ループするプログラムを実行した学生がいたが，バッチジョブシステムによる実行時間制限と，ulimit によるファイル容量制限により，システム全体の動作に影響は無かった．

今後検討すべき実用上の懸念としては，実行バックエンドの利用に関わる規約の問題が考えられる．一例としてソフトウェアの利用許諾がある．許諾の中には，利用者を限定したものもあるため，規約違反とならないよう，十分な配慮が必要である．また，実行バックエンドとして，計算機センターの共同利用設備を使う場合にも注意が必要である．FEHA と実行バックエンドの連携は，単一のアカウントによる SSH で行われる．このような利用を許可しない設備もあることから，利用には十分な注意が必要である．また，セキュリティに関しても，シェルの機能のみでセキュリティ対策を行っているが，FEHA の利用者や環境によっては十分でない場合もある．その場合は，適切なサンドボックス環境の利用が必要になる．

7. おわりに

本論文では，簡易演習や単発的に行われる講習会での体験活動を支援するため，環境構築が容易でかつ様々な目的に転用できる再利用性が高い計算機向けフロントエンド環境 FEHA を提案し，その構成と初期的な実装を示した．FEHA は，計算機システムを用いた体験演習を支援する Web IDE を提供する．プログラム等を実行する実行バックエンドとして，既存システムを利用することで，高い再利用性を実現する．FEHA と実行バックエンドの連携は，SSH により行う．

FEHA を大学における並列処理の講義に適用し，その実現可能性を確認する実験を行った．実験では，約 55 名の学生が FEHA を使い，並列処理技術の一例である OpenMP と MPI を体験した．10 分間の最大で 500 回，講義時間中の合計で 1,000 回を超える超えるリクエストを 1 秒程度の時間で処理することができたことから，大学での 50 人規模の講義において，実用性に問題が無いことが確認できた．

今後は，最新の並列処理技術をより多くの人に体験してもらうため，PGAS や Hadoop など，話題となっている処理系への対応を行う．プログラミング環境だけでなく，製品開発や研究で多く使われているアプリケーションの体験にも対応を進める．

謝辞

本研究の一部は，JSPS 科研費 24700046 の助成により行われたものである．

参考文献

[1] Rodríguez-del Pino, J. C.: VPL, Viertual Programming lab for Moodle, (online), available from ⟨http://http://vpl.dis.ulpgc.es/⟩ (accessed 2015/1/21).
[2] KhanAcademy(NPO): Khan Academy, (online), available from ⟨https://www.khanacademy.org/⟩ (accessed 2015/1/21).
[3] CourceraInc.: Coursera, (online), available from ⟨https://www.coursera.org/⟩ (accessed 2015/1/21).
[4] 株式会社ドットインストール：ドットインストール，（オンライン），入手先 ⟨http://dotinstall.com/⟩（参照 2015/1/26）．
[5] Codecademy: Codecademy, (online), available from ⟨http://www.codecademy.com/⟩ (accessed 2015/1/21).
[6] MoodleHQ: Moodle, Moodle Pty. Ltd. (online), available from ⟨https://moodle.org/⟩ (accessed 2015/1/21).
[7] Glassman, E. L., Scott, J., Singh, R., Guo, P. and Miller, R.: OverCode: Visualizing Variation in Student Solutions to Programming Problems at Scale, *Proceedings of the Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST'14 Adjunct, New York, NY, USA, ACM, pp. 129–130 (2014).
[8] 井垣　宏，齊藤　俊，井上亮文，中村亮太，楠本真二：プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案，情報処理学会論文誌， Vol. 54, No. 1, pp. 330–339 (2013).
[9] Hiroaki, H., Kazuki., M., Kakafumi, T., Atsuo, H. and Seiichi., K.: An Environment for Collecting Fine-Grained Development Records to Help with Programming Exercise, *Advanced Applied Informatics (IIAI-AAI), 2014 IIAI 3rd International Conference on*, pp. 739–744 (online), DOI: 10.1109/IIAI-AAI.2014.150 (2014).
[10] 長谷川伸，松田承一，高野辰之，宮川　治：プログラミング入門教育を対象としたリアルタイム授業支援システム，情報処理学会論文誌， Vol. 52, No. 12, pp. 3135–3149 (2011).
[11] Wang, T., Su, X., Ma, P., Wang, Y. and Wang, K.: Ability-training-oriented Automated Assessment in Introductory Programming Course, *Computers & Education*, Vol. 56, No. 1, pp. 220–226 (online), DOI: 10.1016/j.compedu.2010.08.003 (2011).
[12] TechEmpower: Web Framework Benchmarks, Round 9, (online), available from ⟨https://www.techempower.com/benchmarks/⟩ (accessed 2014/5/1).