学生群の学習状況を把握できる 遠隔プログラミング演習支援システムに関する研究

松本拓也 [1]　碓崎賢一 [2]　荒木俊輔 [3]

概要：近年，コロナ禍の影響により遠隔授業が広く普及した．しかしながら，遠隔授業には，教員から学生達の挙動が見えなくなり，その学習状況を把握することが困難になるという問題がある．
プログラミング授業・演習において，学生の活動はコーディングやコンパイル，実行といった PC 上の操作として現れる．このため，これらの操作情報を取得し，その情報を用いて学生個人やクラス全体の演習状況を分かりやすい形で可視化できれば，教員が遠隔地の学生の学習状況を把握することができる．
この仕組みを実現するために，遠隔プログラミング授業を対象に，学生の学習状況を把握できる演習支援システムの提案・構築を行う．

キーワード：遠隔学習，教育支援，情報可視化

A Research on Remote Programming Exercise Support System that can monitor the learning status of groups of students

Takuya MATSUMOTO[†1]　Ken'ichi KAKIZAKI[†2]　Shunsuke ARAKI[†3]

Abstract: In recent years, remote learning has become widely used due to the Corona disaster. However, remote learning has the problem that the teacher cannot see the students' behavior, making it difficult to grasp their learning status.
In programming classes and exercises, students' activities appear as operations on the PC, such as coding, compilation, and execution. Therefore, if the information on these operations can be obtained and used to visualize the progress of individual students and the class as a whole in an easy-to-understand form, instructors can grasp the learning status of students in remote locations.
In order to realize this mechanism, we propose and construct an exercise support system for remote programming classes that can grasp the learning status of students.

Keywords: Distance learning, Education support, Information visualization

1. はじめに

近年では，新型コロナウイルス感染症の流行により，教室のような密な環境で人々が集まることは困難となり，全国の大学では，遠隔授業の導入が余儀なくされた．急な授業形態の変更による準備不足に加え，コミュニケーションの取りづらさや学生が感じる孤独感といった遠隔授業が持つ問題点により，従来通りの授業の質を保つことは大変困難であった[1].

しかしながら，遠隔授業には，教員や学生が一つの空間に集まる手間を省いたり，動画を利用した授業の提供により学生が自由な時間で受講できたり多くの利点が存在している．With コロナを目指す現代社会において，遠隔授業をより効果的に活用することは必要不可欠である[2].

遠隔プログラミング授業においても，学生間で相談がしづらいことにより，学生の中でも演習の進度状況に大きく差が出てしまう問題や，学生の演習状況を確認することが出来ないため，教員が問題を抱えた生徒を把握することが難しいという問題が存在する．

遠隔プログラミング演習において，対面授業と同様の質の授業を提供できることを目指した，遠隔プログラミング演習を支援する仕組みが求められる．

本研究では，遠隔プログラミング演習において，学生の活動情報を収集し，収集した情報を整理することで，クラス全体の全体像や学生個人の詳細な学習状況の可視化を行う．

2. 背景

2.1 授業形態

大学を始めとした様々な教育の場では，様々な授業形態が取られる．この授業形態は，対面授業と遠隔授業の 2 つに大きく分けられる．対面授業とは，学生や教員が教室な

1 九州工業大学情報工学府情報創成工学専攻
Graduate School of Computer Science and Systems Engineering, Kyushu Institute Of Technology
2 九州工業大学大学院情報工学研究院　教授・博士（工学）
Prof., Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Dr. Eng.
3 九州工業大学大学院情報工学研究院　准教授・博士（情報工学）
Assoc. Prof., Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Dr. Info. Eng.

どの一つの実空間に集まって授業を行う事であり，遠隔授業とは，学生や教員は実際に集まらず，離れた場所からインターネットを介して授業を行う事である．

2.2 遠隔授業

遠隔授業では，授業に参加する人間が一つの空間に集まる必要性が無くなることにより，授業を行うための場所の用意や移動時間の削減が可能である．また，授業の録画を学生が視聴できるようにすることで，自由な時間に学習を行う事が可能となる．

このように遠隔授業には，教員側にも学生側にも大きなメリットがあり，様々な場面での活用が期待される．加えて，近年コロナ禍により，一つの空間に大人数が集まることが困難となった事で，遠隔授業は大きく普及した[2]．

2.3 問題点

しかし，遠隔授業には解決すべき問題点が多数存在する．例として以下のようなものが挙げられる．

・教員が学生の学習状況を把握しづらい
・学生が他の学生と交流を図りづらい
　孤独や不安を感じてしまう
・インターネット環境を整える必要がある

今回はその内、教員が学生の学習状況を把握することが困難であり，悩みを抱えた学生への対応が遅れてしまうという問題の解決に取り組む．

この問題を解決するためには，教員が学生の学習状況を自動的に把握可能にする仕組みが求められる．そこで，学生から活動情報を取得し，その活動情報を整理することで，教員に学習状況を可視化する仕組みを提案する．

2.4 遠隔プログラミング授業

学生から活動情報の取得を目指す上で，プログラミング授業について考える．

プログラムング授業における学生の活動を考える．学生の活動の流れを図 1 に示す．

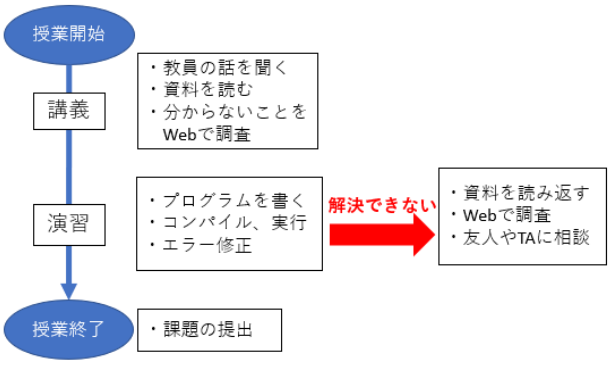


図 1　学生の活動の流れ

まず，学生は授業が開始すると，教員による講義を受ける．教員の話を聞き，資料を読み，その中で分からないことがあった際には，資料を読み返すだけでなく，インターネットを利用した調査も行う．

その後，講義が終わると，教えられた知識を身に着けるために演習を始める．演習では，プログラムを作成し，コンパイルや実行を行う．また，作成したプログラムに不備があった場合コンパイルや実行時にエラーが発生する．このエラーの原因をプログラムの中から探し出し，改善方法を考えるために，講義資料を読み返したり，インターネットで調査を行ったり，友人や TA(ティーチング・アシスタント)，教員に相談したりする．こうして作成し直したプログラムに対して，再度コンパイルや実行を行っていく．

このように作成したプログラムが想定通りの動作をするように試行錯誤する中で，学生は知識を自分のものにしていく．

授業終了後では，授業中あるいは授業後に課された課題に取り組み，その課題を提出する．

以上が学生の活動の流れであるが，演習で行う活動の大半がコーディングやコンパイル，実行などの PC 上の操作として行われるものであり，PC を通して学生の活動情報を取得することが容易である．

また，ある学生がプログラムを編集→コンパイル→エラー→再度プログラムを編集→…という流れを繰り返していることが確認できたならば，その学生は問題を抱えており，教員が支援する必要があると分かる．また，その学生の演習過程を時系列に沿って確認できれば，学生が何に悩んでいて，その問題にどのように取り組んだのかを確認が出来る．

このように，コーディングやコンパイル，実行といった活動情報を整理することで，クラス全体の学習状況と学生個人の詳細な学習状況を把握することが出来る．

そこで，遠隔プログラミング授業を対象に，遠隔授業の問題を解決する仕組みを考えていく．

3. 関連研究

遠隔授業における学生の学習状況を把握することを目指した取り組みは様々なアプローチによって行われている．

遠隔授業において，zoom や Microsoft Teams 等のオンライン会議用ソフトウェアを利用して実施されることがほとんどである．これらのツールでは基本的にチャットや音声を利用してコミュニケーションを図るが，教員と学生が 1 対多である授業に対しては，学生側からの質問等のアクションに心理的な障壁が生まれてしまう．そこで中川[3]は「CommentScreen」[4]という視聴者がコメントを打つと授業画面に匿名でかつリアルタイムに表示されるアプリケーションを遠隔授業に採用した．結果として，状業への理解の促進や孤独感が感じなくなるといった学生の心理面に大きな効果を示した．

また，遠隔プログラミング授業において，プログラミング演習環境として VPL や BitArrow[5]などが利用されている．

北島ら[6]はオンラインプログラミング環境 BitArrow を用いたプログラミング演習を提供し，BitArrow のログから学生の学習状況の把握を行った．

このように，近年，遠隔授業が持つ問題点を解決するための試みが多く為されている．しかしながら，これらの手法では，学生は自主的に教員に対して意思表示を行う必要性があったり，教員がリアルタイムで学生の学習状況を把握することが困難であったりする．

4. 提案する授業支援の仕組み

教員が学生の学習状況を把握することが困難であるという問題を解決するためには，

学生から活動情報を取得し，その活動情報を整理した形で教員に学習状況を可視化する仕組みを提案し，これにより，教員が学生の学習状況を把握することを目指す．

4.1 学生からの情報取得

まず, 学生から活動情報を取得する方法について考える．

学生が通常通りプログラミング演習を行いながら，その中で特定の活動を行った際に，その活動情報を取得する必要がある．

また，学生が各々コンパイラを用意した場合，授業資料に沿ったプログラミングを行ったとしても OS やコンパイラのバージョンによって処理結果が異なってしまう．

そこで，Web ブラウザ上で動作する IDE(投稿開発環境)を提案する．学生は Web ページにアクセスすることで IDE を利用する．

コンパイルや実行といった処理は学生の PC 上では行わず, サーバにプログラムを送信し, サーバ側で処理を行う．その後，処理結果を IDE に反映することで学生はコンパイルや実行を実現する．また，学生が処理を行った際には，活動情報をサーバに送信し，データベースに保存する．

これによって，学生の演習環境や活動情報を取得する仕組みを統一的に提供することが出来る．

IDE には以下の機能を実装する．

- プログラムを作成するエディタ
- 作成したプログラムに対する処理
  - 整形　・コンパイル　・実行
- 処理結果の表示

またサーバには，コンパイルや実行といったプログラムへの処理を行った際の処理情報に加えて，一定間隔でプログラムの編集情報として，プログラムの文字数の増減情報を送信する．

4.2 IDE(統合開発環境)

実装した IDE を図 2 に示す．学生はこの IDE を利用して演習を行う．

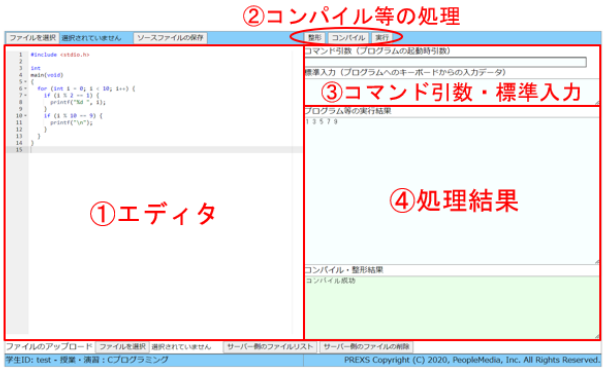


図 2　IDE の構造

まず初めに，学生は Google Chrome, Microsoft Edge 等を Web ブラウザから URL を利用して，IDE にアクセスする．その後、学生は①番のエディタでプログラムを作成・編集する。作成・編集したプログラムに対して、整形・コンパイル・実行処理を行うには②番のボタンを押すことで利用できる。また、実行の際に、コマンド引数や標準入力を入力したい場合には③番の各種領域に適宜入力することで利用できる。整形処理の結果はエラーが生じなかった場合には①番のエディタに反映され、エラーが生じた場合には④番の領域に表示される。コンパイル・実行処理の結果はエラーの有無に関わらず④番の領域に表示される。

このように、学生はこの IDE を利用して、通常通り演習を行う事が出来る．

4.3 情報取得間隔

加えて，この IDE を通して活動情報をサーバに送信し，そのタイミングはコンパイル・実行処理情報の場合はその処理を行う度に，プログラムの編集情報の場合は特定のタイミングが存在しないので一定間隔で送信する．

クラスは百人単位で行われているため，この間隔が 1 秒間である場合，1 秒間に百回程度データベースに保存する必要があり，サーバへの負荷は大きい．また教員が学生の演習過程を確認する際に確認する量が多くなる．対してプログラムは数文字程度しか変化が起こらず，非効率であると言える．逆に，10 分間である場合，1 秒間に 1 回に満たず，サーバへの負荷は大幅に小さい．しかしプログラムは数百文字の変化が起きてしまい，学生の思考を追跡することは困難となってしまう．

そこで，サーバへの負荷を抑えつつ，プログラムの変化が数十文字程度で，学生の思考を追跡することが出来る 1 分間隔とした．

4.4 学生個人の詳細な演習状況の可視化

取得した活動情報から学生個人の詳細な状況を教員に可視化することで，問題を抱えてしまった学生が，何を悩んでいるのか，どのような思考を経て，問題を解決するために何に取り組んだのかなどを把握できるようにする．

情報の可視化には，表やグラフといった様々な表現法がある．今回はプログラミング授業という百人単位での実用

される想定される授業を対象とするため，一つ一つの通信量と可視化における表示領域はなるべく最小にする必要があるため，それぞれの情報をテキストで表現でき，また教員が学生個人の詳細な状況を把握しやすいように，演習情報を時系列順に並べ，その活動に対して複数の情報を表示できる表形式による可視化を行った．

また，ある時刻における行った活動に対して，整形を「F」，コンパイルを「C」，実行を「E」としてテキストによって簡潔に示し，その活動の際のソースコードや標準出力，エラー出力の文字数を表示することでソースコードや標準出力の文字数からどの程度目標通りに演習を行えているか，またエラー出力からソースコードにどの程度エラーが含まれていたのかを簡易的に表示可能にした．

加えて，エラーが発生し得る処理がエラー無く動作したかを一目で確認できるようにするため，上手くいった場合は青，エラーが発生した場合は赤に背景色を設定した．

このようにして，学生個人の演習状況を可視化した例を表 1 に示す．

表 1　可視化した学生の活動履歴

日時	活動	ソース	標準出力	エラー出力
2022-12-06 03:12:33	E	0	5	0
2022-12-06 03:12:31	C	71	0	0
2022-12-06 03:12:29	F	71	0	0
2022-12-06 03:12:20	C	71	0	311

表 1 では，ある学生の演習の様子を可視化している．学生は 12:20 に作成したプログラムに対してコンパイルを行った所失敗した．しかし，その後 9 分かけて，プログラムを編集し，その後，再度整形し，コンパイルを実行した所，成功し，その後の実行にも成功したことが確認できる．

このように，時系列に沿って学生の活動を確認していくことで学生の詳細な活動情報が確認できる．

また、表 1 では，任意の活動に対してソースコードや標準出力，エラー出力の文字数を示していたが，それだけでは，具体的な内容を把握することが出来ない．加えて，実行の際には，標準入力やコマンド引数の情報も必要となる．このように，問題を抱えた学生の対処を行う上で，表 1 のみでは情報が不足する．

そこで，表内の任意の時刻の行を選択することで，その時刻における学生の IDE を復元できるようにした．(図 3)

図 3　特定時刻の IDE 復元

この 2 つの表を組み合わせることによって，学生が演習に取り組んだ過程を追いながら，要所でどのように間違ってしまったのかを理解することができる．

4.5 クラス全体の演習状況の可視化

前項では，学生個人の詳細な演習状況の可視化について述べたが，クラス全体の演習状況の可視化について提案する．

百人単位の授業において，教員は授業が想定通りに進んでいるかを判断するため，クラス全体の進捗を確認し，一方で，学生の中に問題を抱えてしまった学生が居ないか把握する必要がある．このように，学生個人の演習状況とクラス全体の演習状況を同時に可視化する必要がある．そのため，4.4 節と同様に，表形式を採用する．

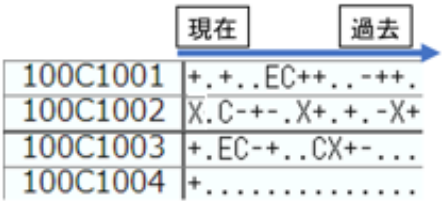
ある学生がエラーを繰り返す，作業が何分間も止まっているなどの情報があれば学生が問題を抱えていることを判別できるため，学生の状態を以下のようにテキストとして表す．

・活動無し：「.」
・プログラムの文字数の増加：「+」，減少「-」
・コンパイル：「C」　・実行：「E」　・エラー：「X」

各学生のある時刻の状態をテキストで簡潔に表現し，それらを用いて文字列として演習過程を簡潔に表現する．そして，この文字列を並べることで，クラス全体の演習状況を可視化できる．

このようにして，クラス全体の演習状況を可視化した例を表 2 に示す．

表 2　可視化したクラス全体の演習状況



現在時刻が最も左になるように，1 分間隔で活動状況を配置しており，1 分の間に複数の状態が重複している場合は，実行が出来たならば演習が完了した可能性があるため，実行，エラー，コンパイル，プログラムの文字数の増加の順で優先しており，これらの活動が行われなかった場合には，活動無しとして表現される．

表 2 では，学生 100C1002 はエラーを短期間に繰り返しており，学生 100C1004 は何もしていない状態が続いていることが確認でき，問題を抱えている可能性が確認できる．

このように，学生個人の状況を少ない情報で，素早く把握できつつ，クラス全体の状況を把握することが出来る．

この 2 つの表と復元した IDE を用いることで，クラス全体から学生個人まで学習状況を把握することができ，問題を抱えてしまった学生の演習の経緯を時系列順に把握し，効率的に学生の問題解決への支援を行う事が可能となる．

5. 実現方式

5.1 概要

4 節で提案した仕組みを実現するために，学生の学習状況を把握できる演習支援システムの提案・構築を行う．

学生は Web ブラウザ上で動作する IDE を利用して演習を行う．コンパイルや実行を行う際には，プログラムといった必要な情報と合わせて，サーバにリクエストを行い，サーバ側で処理を行う．その後，処理結果を IDE に反映させる．また，このコンパイルや実行といった活動時や 1 分間隔で活動情報をサーバに送信し，データベースに保存する．教員がサーバに学生の演習状況の可視化を求めると，データベースに保存されている情報を表形式にまとめた上で，教員に可視化する．

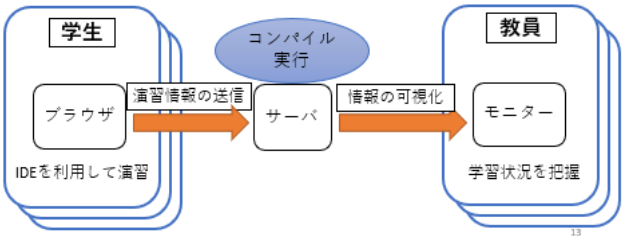


図 4　システムの構造

構築するサーバには，Web サーバとして Nginx[7]，アプリケーションサーバとして Node.js[8]，データベースサーバとして MySQL[9]を採用している．

対象とするプログラミング言語は C や C++といったコンパイラ言語を対象とする．コンパイル言語では，ファイルを実行する際には，コンパイルした実行可能ファイルを実行するだけで良い．本システムでは，マルチスレッドで処理することで，百人隊の学生からのリクエストを処理することを考えている．そのため，Python のようなインタプリタ言語ではなく，この特性を持ったコンパイル言語が適している．

また，提案システムを構築する上で，必要な要件は以下とである．

・学生からの活動情報を取得する仕組み
・教員へ学習状況を可視化する仕組み
・データベースの構築
・アカウントと権限の管理

5.2 学生の活動と活動情報取得

学生は図 2 の IDE を利用し、演習を行う．

この際，作成したプログラムに対してコンパイルや実行といった処理を行うが，この処理は学生側の PC 上では行わず，サーバ上で処理を行い，その処理結果を IDE に反映する．

この仕組みを実装するため，学生の IDE 内のコンパイルや実行ボタンには，サーバに対して，該当する処理に必要な情報と共にリクエストを送信する．コンパイルの際にはプログラムを，実行の際にはコマンド引数や標準入力を送信する．サーバはリクエストを受け取ると，サーバ側を動作させている LinuxPC 内の，学生のプログラム管理用ディレクトリ内にて，リクエストされた処理を行う．その後処理結果として，標準出力やエラー出力をレスポンスとして返信する．

この処理と共に，この活動における情報をデータベースに記録する．記録する情報は，どの活動か，その活動における必要な情報，活動の処理結果，活動が行われた日時である．

また，学生から，活動とは別に 1 分間隔で情報を取得する．学生が IDE にアクセスした時間から 1 分間隔で，ソースコード情報と共にサーバにリクエストを送信する．

5.3 教員への学習状況可視化

5.2 節で取得した学生の活動情報を元に，教員に学習状況を可視化する．

教員は，教員用の Web ページにアクセス後，4 章で述べた 3 種の教員へ示す図表(図 2，表 1，表 2)をそれぞれリクエストしていく．サーバ側では，リクエストされた図表を構築するために必要な情報をデータベースから取得し，情報を整理して教員に可視化する．

まず，学生の詳細な学習状況の可視化には，どの学生かや，活動が行われた日時，どの活動を行ったかに加えて，ソースコードや標準出力，エラー出力などの文字数を表示している．これらの情報は全て学生の活動時に得られる情報であるため，表として整理して可視化を行う．

次に，学生の特定時刻の IDE の復元には，どの学生かや，学生の詳細な学習状況可視化の表の内から選択した時刻，そのソースコードに加えて，情報があるなら，標準入力やコマンド引数，標準出力，エラー出力を表示している．これらの情報も全て学生の活動時に得られる情報であるため，表として整理して可視化を行う．

最後に，クラス全体の学習状況の可視化には，各学生の活動時に取得した情報と 1 分間隔で取得した情報が必要であり，時刻と，どの活動かまたはソースコードの文字数がどう変化したかである．

5.4 アカウントと権限の管理

クラス単位で演習支援システムを運用する上で，各学生から活動情報やプログラムをサーバへ送信されるが，これらが誰のものであるかを判別する仕組みは当然必要である．

また，この演習支援システムでは，学生が作成したプログラムをサーバ側で実行する．そのため，他の人の成果物を覗いたり，削除されたりする可能性が発生するため，各学生が他の人に影響を与えない仕組みを提供する必要がある．

そこで，ログインページによるアカウントの管理に加えて，Linux によるファイルと権限の設定を行う．

5.4.1 システムへのログイン

学生は IDE を利用する前にログインページにて，ユーザ情報としてユーザ ID とパスワードを入力する．これによって，部外者がシステムにアクセスすることを防ぐと共に，学生が送信する活動情報やプログラムにユーザ情報を紐づけることで管理できる．

また，アカウント作成時には，アカウント管理用テーブルにアカウントを追加するだけでなく，Linux のアカウントと学生のプログラム管理用ディレクトリ内に，学生専用のディレクトリを作成する．

5.4.2 Linux によるファイル管理

学生がコンパイルの処理のリクエストを送信した際には，この学生専用のディレクトリ内に送信した作成したプログラムが保存され，コンパイルが行われる．コンパイルにより作成された実行可能ファイルも同ディレクトリに保存され，実行の際に利用される．このようにして，学生側の PC 上にコンパイラ等や演習環境が無くても，我々が提供する IDE によって演習を行う仕組みが作成できた．

また，この仕組みでは，プログラムをサーバ側の PC 上で，学生が作成したプログラムを実行するため，プログラムを実行する際のみ，実行権限を学生用のレベルに落とす必要がある．そこで，5.5.1 で作成した Linux アカウントを用いる．各アカウント専用のディレクトリに対して，自身以外には何も権限を与えないように設定することで，学生は自身のディレクトリのみにアクセス可能であり，他者のファイルを覗き見る・削除するなどが行えないように制限した．

6. 実験・評価

6.1 負荷実験

提案システムが想定する遠隔プログラミング授業では，百人単位の学生が参加し，システムが提供する IDE を通して，同時に演習を行う．そこで，システムが耐えることができる通信量を計測し，百人単位で運用を行うことが可能であること確認するために，負荷実験を実施する．

6.2 実験環境

本実験において，サーバとして 1 台の PC を利用した．この PC の性能を以下に示す．

- OS：CentOS-7
- CPU：Interl(R) Core(TM) i5-2000, 4-Core, 3.30GHz
- 記憶装置：SSD
- メモリ：24GB

6.3 実験方法

本実験では，学生が行う活動の内，複数処理において負荷の大きいコンパイルを対象とし，1 つのクライアントからサーバにコンパイル処理のリクエスト送信を高速に行う．コンパイルのリクエストにおいては，時間がかかると考えられる処理はコンパイル処理とデータベースへの記録の 2 点であるため，以下のように段階的にシステムにかける負荷を大きくしていく．

① サーバ側で処理を行わない通信
② サーバ側でコンパイルのみを行う通信
③ サーバ側でコンパイルとデータベースへ記録する通信

また，学生が百人単位のプログラミング授業を想定しているが，クライアント数を 100 として，負荷実験を行う事は困難である．そこで，リクエストを送信する総量が変わらなければ，リクエストを送信するクライアントの数を変化させても，サーバへの負荷が変わらないことを確認する．そこで，リクエストを送信するクライアント数が 1 の場合とクライアント数が 4 の場合についても比較し，学生の人数に影響せずに運用が可能であることを確認する．

また，学生は活動時の他に，一定時間ごとにサーバに編集情報を送信する．そのため，一定時間ごとに行われる処理に対しても負荷実験を行う．コンパイルとは異なり，編集情報の保存時には，時間がかかると考えられる処理はデータベースの保存のみのため，段階的には負荷をかけることはせず，処理全体に対して負荷実験を行う．

コンパイルに利用したソースファイルは 104 行で，サーバ PC 上におけるコンパイル処理時間は 0.08 秒程度であった．スレッド数は CPU のコア数の倍の数である 8 とした．負荷の指標としてリクエストの処理速度(1 秒間に処理できるリクエスト数)や Load Average と CPU 使用率を利用した．

6.4 結果

負荷実験の結果を表 3~5 に示す．

まず，①②③それぞれにおける処理可能速度とその際の最高 Load Average，CPU 使用率は表 3 となった．①②③と段階的に，処理可能速度が低下すると共に，最高 LoadAverage や CPU 使用率が向上していることが確認できる．②では，処理可能速度は秒間 50 回であったが，これは 1 リクエストに対し，1 コアあたり 0.08 秒で処理が可能であることを示している．この値は本実験で利用したソースファイルのコンパイル時間と一致するため，処理時間の大部分がコンパイルであり，期待通りの結果が得られていることが確認できる．

③では，②と同様に考えると，1 リクエストに対し，1 コアあたり 0.1 秒で全体の処理が可能であることを示しているため，データベースへ記録にかかる処理時間は 0.02 秒程度である．従って，良好な結果が確認できた．

表 3　段階的なコンパイル通信の負荷実験

	処理可能速度	最高 LA	CPU 使用率
①	1000 回/s	1.32	10~14
②	50 回/s	1.59	18~21
③	40 回/s	3.51	38~40

③における処理可能速度である秒間 40 回のリクエスト

数を用いて，クライアント数を 1 として，秒間 40 回のリクエストを行う場合とクライアント数を 4 として，それぞれ秒間 10 回のリクエストを行う場合を比較した結果を表 4 に示す．

結果として，クライアント数が 4 の場合の方が Load Average や CPU 使用率が若干高くなったが，リクエストの総数が変わらなければ，利用者の人数はサーバへの負荷に影響がないことが確認できた．

表 4　クライアント数の変化による負荷への影響

クライアント数	処理可能速度	最高 LA	CPU 使用率
1	40 回/s	3.51	38~40
4	10 回/s	3.69	39~42

コンパイル処理と編集情報の保存処理の比較結果を表５に示す．編集情報の保存処理では，コンパイルの時とは異なり，データベースへの記録にのみ時間が掛かる．そのため，コンパイル処理より高速に情報を処理することを可能となっている．

表 5　コンパイルと編集情報保存の比較

	処理可能速度	最高 LA	CPU 使用率
コンパイル	40 回/s	3.51	38~40
編集情報保存	100 回/s	2.35	9~11

6.5 性能評価

プログラミング授業において，一人の学生がコンパイルや実行といった活動を行う頻度は分間 1~2 回程度だと考えられ，これに 1 分間ごとに行う編集情報保存処理も加えると，百人規模の授業では，合計で秒間 3~5 回程度の頻度となる．

実験結果からコンパイル処理に対して毎秒 40 回程度まで対処でき，編集情報に対して毎秒 100 回程度まで対処可能であることが確認できている．加えて，学生の人数はシステムへの負荷に影響しないことが確認できている．

よって，本システムの学生とサーバ間の通信が，百人規模の授業への運用が可能であることが確認できた．

7. まとめ

本稿では，学生群の学習状況の把握を可能にする遠隔プログラミング演習支援システムの提案を行った．プログラミング演習において学生が行う活動情報を収集し，その情報を整理することによって，クラス全体や学生個人の学習状況の可視化を行った．これにより，教員はクラス全体の進捗状況を把握しつつ，問題を抱えた学生の特定やその学生への支援が円滑に行うことが可能となった．

また，本システムの運用に向けて，百人単位の学生が同時にシステムを利用した際に問題なく運用できることを確認した．

参考文献

[1] 文部科学省，“コロナ対応の現状，課題，今後の方向性について”, p3, 2020.
[2] 服部 辰広，松田 康宏，伊藤 譲，久保山 和彦，“対面授業と比較した遠隔授業の学習効果に関する研究－保健医療学部生整復医療学科学生に対するアンケート調査より－”，2022, 日本体育大学紀要 51，p. 1001-1009.
[3] 中川 晃“遠隔授業における「時刻同時コメント」の教育効果に関して”, 2202，コンテンツ教育学会誌 4 巻, p25-39.
[4] “CommentScreen”, https://www.commentscreen.com/, (参照 2023-02-10).
[5] “オンラインプログラミング環境　ビットアロー(Bit Arrow)”, https://bitarrow.eplang.jp/, (参照 2023-02-13).
[6] 北島 茂樹，山中脩也，長 慎也，今野 貴之，“オンラインプログラミング演習環境における対話の実践と評価”, 2021.
[7] “NGINX”, https://www.nginx.co.jp/, (参照 2023-02-10)
[8] “Node.js”, https://nodejs.org/, (参照 2023-02-10)
[9] “MySQL”, https://www.mysql.com/jp/, (参照 2023-02-10)