# コンテナ仮想化技術による SDN 演習システムの提案

長谷部 竜司[1]　新村 正明[2]

**概要**：SDN ネットワーク技術の普及に伴い，ネットワーク技術者の需要が高まり，技術者育成の演習が多くの教育機関で行われている．実機を利用する演習では，人数分の演習機器を用意する必要などの問題が生じていたため，信州大学ではこの問題を解決するために，ハードウェア仮想化を用いて，効率的な演習を行っている．しかし，SDN 演習では，学習者ごとに演習環境をハードウェア仮想化を用いる手法で提供していたため，演習環境の提供数に限りがあるという問題が生じていた．そのため本研究では，コンテナ仮想化によって SDN 演習環境を構築することで，学習者が Docker・Open vSwitch・trema といった技術を利用できる環境の軽量化を行った．その結果，ハードウェア仮想化と比較し，2 倍程度のメモリ効率を示した．

**キーワード**：SDN，ネットワーク演習，コンテナ仮想化，BYOD

# Proposal for SDN exercise system using container virtualization technology

Tatsushi Hasebe[1]　Masaaki Niimura[2]

***Abstract:*** With the spread of SDN network technology, the demand for network engineers has been increasing, and many educational institutions have been conducting exercises to train engineers. In order to solve this problem, Shinshu University has developed a hardware virtualization system. In order to solve this problem, Shinshu University is using hardware virtualization for efficient training. However, in the SDN training, since the training environment was provided for each learner using the hardware virtualization method, the number of training environments was limited. However, in SDN training, the number of training environments provided to each learner is limited because of the hardware virtualization method. In this study, we propose to build SDN training environments using container virtualization. In this study, we constructed an SDN training environment using container virtualization to lighten the environment where learners can use technologies such as Docker, Open vSwitch, and trema. As a result, the memory efficiency was about twice as high as that of hardware virtualization.

***Keywords:*** SDN, Network Practice, Container Virtualization, BYOD

## 1. はじめに

近年では人と物をつなぐ ICT や IoT といった技術の発展により，ネットワークに接続される機器の増加や，接続方法の多様化から，ネットワーク仮想化技術の需要が高まっている．このため大学等の教育機関では SDN(Software Defined Networking) 等のネットワーク構築演習が盛んに行われている．

また，ネットワーク以外にも，コンピュータ・ストレージを仮想化する SDI(Software Defined Infrastructure) 技術が普及している．これらは，コンピュータ上の演算装置・記憶装置・ストレージ・ネットワークを分割し，仮想化さ

[1] 信州大学大学院総合理工学研究科 Shinshu University Graduate School of Science and Technology, Nagano, Nagano 380–8553, Japan
[2] 信州大学
shinshu-u, Nagano, Nagano 380–8553, Japan

れたコンピュータ・ネットワーク機器をソフトウェアにより動的に作成・制御を行う技術である．

これら SDN・SDI 技術の実践的な演習を行うためには，サーバ・ルータ・スイッチなどの機材を利用する必要があるが，これらすべての機材を学習者分用意することは金銭・作業量の面から困難であることが考えられる．そのため，多くの学習者が実践的な演習を行うためには，仮想化技術を用いて，利用する機材の削減・効率化を行う必要がある．

本研究では，SDN 演習環境をコンテナ仮想化技術で提供し，演習サーバのリソース資源を低減させる手法を提案する．

## 2. 関連研究・技術

### 2.1 web による演習システム

演習環境を学習者が持ち込んだパソコン上に構築する場合，持ち込まれるパソコンの OS 等が異なるなど，環境の作成方法や操作方法が異なるため，環境の一元化が困難である．田中らは，Bring Your Own Device(BYOD) によって生じる環境統一の課題を，web アプリケーションとして学習者が演習システムを利用できるようにシステムを開発した [1]．このプラットフォームでは Learning Management System との連携を行いユーザ認証を行っている．

井口らは，実機を用いたネットワーク演習が世界的に普及する一方，学習者が手軽に演習に取り組むことが出来ないことを問題点として挙げ，User Mode Linux(UML) を用いたネットワーク構築演習支援システムを開発した [2]．この演習システムでは，学習者が web 上の操作画面を用いて，ルータやスイッチを配置し，ping 等を実行することでネットワーク演習を実施する．この操作画面は演習システム開発者が事前に埋め込んだ機能・プロトコル以外の実行を行うことはできず，最新のプロトコルの実施などは演習システムの再構築が必要である．

以上のように，持ち込まれた端末に対応するために，演習システムを web アプリケーションとして開発することが広く行われている．

### 2.2 仮想化技術

仮想化技術とはコンピュータを分割・隔離することで仮想的なコンピューターを作成する技術のことである．主に利用されている仮想化技術として，後述するハードウェア仮想化とコンテナ仮想化の 2 種類が存在する．

#### 2.2.1 ハードウェア仮想化

ハードウェア仮想化とは，CPU やネットワークデバイスのエミュレーションを行い，ホストと隔離されたオペレーションシステムを実行する技術である．後述するコンテナ仮想化と異なり，オペレーションシステムがホストと独立している，そのため，カーネルを共有する際に生じるセキュリティリスクが低く，仮想環境内の自由度が高い．ハードウェア仮想化を実現するソフトウェアとして，VirtualBox・KVM・VMware などが存在する．

#### 2.2.2 コンテナ仮想化

コンテナ仮想化とは Linux カーネルの機能である Namespace を用いて，プログラムを他のプログラムと隔離・グループ化することで，軽量な仮想環境を実現する技術である．主にコンテナ仮想化では下記の項目が Namespace(NS) により分離されている [3]．

- プロセス (PID NS)
- ファイルシステム (MOUNT NS)
- プロセス間通信 (IPC NS)
- ネットワーク (NETWORK NS)
- その他

コンテナ仮想化を実現するソフトウェアとして，Docker，LXD・LXC などが存在する．

##### 2.2.2.1 Docker と LXD・LXC の比較

Docker 及び Docker コンテナはホスト OS の特権ユーザとして動作している．これに対し，LXD・LXC はホスト OS では非特権ユーザとして動作するが，コンテナ内では特権ユーザとして動作している．特権が利用できる範囲のイメージを図 1 に示す．

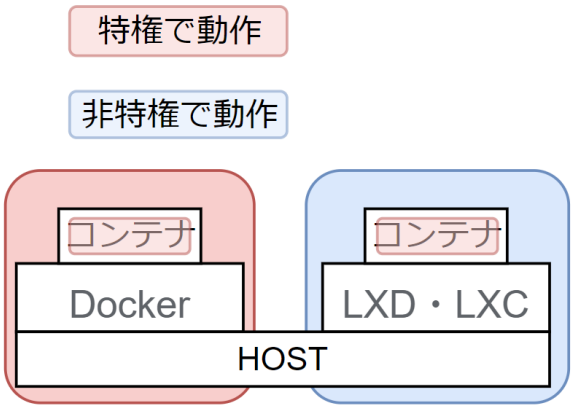


**図 1** 特権で動作するイメージ図
**Fig. 1** State in the exercise environment.

このように Docker ではコンテナにもホスト OS に適用される特権権限が与えられることになるため，コンテナ内の特権権限の行使に厳しい制限がかけられている．一方，LXD では完全な非特権ユーザとして動作するため LXD コンテナ内の特権権限の行使には制限がかけられていない．ソースコード 1，2 に示すように，Docker コンテナ内では Namespace の実行が制限されているが，LXD では制限されていないことがわかる．

## 3. 提案手法

### 3.1 Docker の利用が可能な演習環境の必要性

Docker はアプリケーション開発やプログラミング環境

**ソースコード 1** docker による namespace の実行

```
1 user@host:~$ docker run -it --rm ubuntu
      bash
2 root@18f60303015d:/# unshare -u sh
3 unshare: unshare failed: Operation not
      permitted
4 user@host:~$ docker run --cap-add
      SYS_ADMIN -it --rm ubuntu bash
5 root@7db5e8900808:/# unshare -u sh
6 #
```

**ソースコード 2** lxd による namespace の実行

```
1 user@host:~$ lxc launch ubuntu container
2 Creating container
3 The local image 'ubuntu' couldn\'t be
      found, trying 'ubuntu:' instead.
4 Starting container
5 user@host:~$ lxc exec container bash
6 root@container:~# unshare -u sh
7 #
```

構築などで幅広く活用されており，現在では 700 万のレポジトリが存在している [4]．レポジトリには Python や Java といった主要なプログラミング言語の実行環境のほか，最新のプログラムや技術も Docker により配布される場合が多い．そのため，演習環境内で Docker が利用できることは，実行できるプログラムの選択肢を増やす事につながり，ネットワーク演習では最新の通信プロトコルを利用することができる．以上の点から，学習者の演習環境内で Docker が利用できることは，様々な学習が可能になるというメリットが生じる．

### 3.2 演習環境内での Docker の実行方法

学習者ごとに演習環境を割り当てる場合，学習者分のコンピュータを用意することは予算・準備の面から難しい．そのため，仮想化技術を用いて演習サーバから学習者が利用する演習環境を提供する手法が一般的である．

演習環境は仮想化技術を用いて提供されるため，Docker を演習環境で動作させるためには，仮想化技術内で Docker が動作することが求められる．そのため本研究では，仮想化環境内でも Docker を利用できるような仕組みを提案する．

#### 3.2.1 Docker in 仮想マシン

田中らの先行研究で行われている仮想環境の提供方法である [1]．この方法では仮想マシンの仕組み上，確実に動作させることができるが，後述する演習サーバ (表 1) では，演習環境を 20 台程度起動すると，環境内の動作が遅くなる現象が生じた．この現象の理由として，東らの研究ではホスト Linux のプロセススケジュールのアルゴリズムによって，仮想マシンの数が増加すると仮想マシン内の性能が低下することを指摘している [5]．具体例としては，16 台の仮想マシンと，1 台の仮想マシンの性能比較を行い，16 台のほうが 1 台より 4 割性能が低下していた．

以上から仮想マシンでは台数に比例する性能低下が起こることが考えられ，大規模な演習環境の提供は，困難である可能性がある．

**表 1** SDN 演習サーバの構成
**Table 1** Configure the SDN exercise server.

| 項目 | 詳細 |
|---|---|
| cpu | Intel(R) Xeon(R) CPU E5-2640 v4 |
| memory | 128GB |
| host os | Ubuntu 20.04.2 LTS |
| host kernel | 5.4.0-72-generic |

#### 3.2.2 Docker in Docker

Docker コンテナ内で Docker デーモンを起動しコンテナを作成する方法であり，仮想マシンより軽量に動作する．しかし，Docker コンテナ内で Docker・OpenFlow を動作させるためには，Docker コンテナに「Privileged」もしくは「CAP_NET_ADMIN」などの特権権限が必要であるが，前節で述べたように Docker コンテナには特権権限に制限がかけられており，演習ごとに必要な権限を確認して個別に付与する必要がある．

#### 3.2.3 Docker in LXD・LXC

LXC コンテナ内でコンテナ仮想化技術を動作させる技術（入れ子）[6] を用いて Docker を動作させる方法である．前節で述べたように，LXC コンテナ内では特権権限を行使することができるため，LXC コンテナ内で Docker・OpenFlow を動作させることが可能になる．

以上より，LXD・LXC を用いることで，仮想マシン・Docker と比べ，軽量かつ簡便に仮想化技術内で仮想化技術が動作する．したがって本研究では LXD・LXC を用いた演習環境の構築を提案する．

## 4. 演習環境の構築と評価

### 4.1 対象となる演習

一般的な仮想化ネットワーク技術演習では，下記のような手順で演習が実施される．

(1) 通信課題の提示
(2) 仮想スイッチを作成
(3) ネットワークコマンド実行用の仮想 PC（コンテナ）を作成
(4) 仮想スイッチと PC を接続
(5) 仮想スイッチの経路を制御するコントローラの作成・実行
(6) 仮想 PC を用いて通信パケットを送信・状態の検証
(7) 課題の通信が可能であれば演習終了・出来なければ

（5）に戻る

演習の具体例として，「PC1 からスイッチ 1・ソフトウェアルータ (PC3)・スイッチ 2 を経由して PC2 に ping を行う」が提示された場合，学習者は演習環境内でネットワークの作成を行う．演習環境内では「仮想 PC」「仮想スイッチ」を作成できるシステムとして「Docker」「Open vSwitch」を用いた場合，演習環境内の状態は図 2 のような状態になる．

ここで，動的ルーティングなどの，BGP プロトコルを利用する演習の場合は，Quagga・FRRouting などのルーティングソフトウェアのコンテナイメージを Docker HUB から取得する．

また，SDN 技術の一つである OpenFlow の演習を行う場合には Open vSwitch を OpenFlow 対応に設定する．

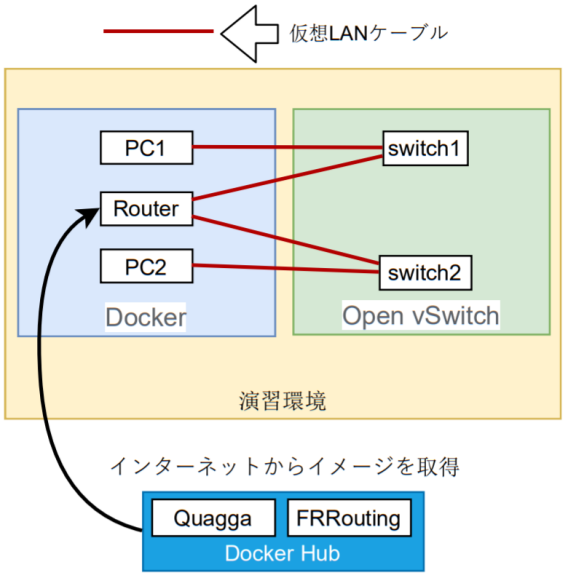


**図 2** 演習環境内の状態
**Fig. 2** State in the exercise environment.

### 4.2 演習環境の構築

関連研究で述べたように，学習者が自身のコンピュータ (端末) を持ち込むことが一般的になっている．そのため本研究では，持ち込まれた端末に対応するため，web アプリケーションとして利用できる演習環境を LXD を用いて構築した．この演習環境では Docker・OpenFlow の実行が可能である．

また，ブラウザのみでの動作が可能な，codeserver[7] をコンソール・エディタとして用いた．図 3 に学習者が操作する演習環境の操作画面を示す．

### 4.3 演習環境の評価

今回作成した演習環境と，従来の仮想マシン (VM) の消費する計算資源量を比較し評価を行う．また，実際の授業で利用し，運用情報を収集する．

#### 4.3.1 計算資源量の比較

LXD のコンテナモードと VM モードを用い，コンテナと仮想マシンで使用する計算資源量の違いを調査した．調査方法としては，表 2 の環境において，SDN 環境を構築するセットアップスクリプトを作成し，SDN 演習環境の機能を持つ，同一コンテナ・仮想マシンを構築した．その後，コンテナ・仮想マシンの両方で下記の作業をスクリプトを用いて実行した．

(1) 演習環境の起動
(2) 60 秒後，起動時の情報を取得
(3) OpenFlow ネットワークの作成
(4) OpenFlow ネットワークにコンピュータを追加
(5) OpenFlow コントローラーの起動
(6) 追加したコンピュータから ping を実行
(7) OpenFlow のテーブルを表示
(8) 追加したコンピュータを削除
(9) OpenFlow ネットワークの削除
(10)演習時の情報を取得

大学の授業で利用する演習環境は，授業期間中は動作させておく必要があり，演習環境は待機状態が多いと考えられる．そのため，待機状態での計算資源の消費量傾向を確認するため，演習スクリプト実行後に CPU・メモリの集計を行った．授業は 1 週間ごとに実施されることを想定し，7 日後の集計となっている．

**表 2** SDN 演習環境開発環境の構成
**Table 2** Configuration of the SDN Training Environment Development Environment.

| 項目 | 詳細 |
|---|---|
| cpu | Intel(R) Xeon(R) CPU E3-1225 v3 |
| memory | 32GB |
| host os | Debian GNU/Linux 10 (buster) |
| host kernel | Debian 4.19.181-1 |
| vm os | 20.04.2 LTS (Focal Fossa) |
| vm kernel | 5.4.0-1040-kvm |
| container os | 20.04.2 LTS (Focal Fossa) |
| container kernel | Debian 4.19.181-1 |

**表 3** SDN 演習環境の CPU 使用時間
**Table 3** CPU usage time for SDN exercise environment.

| | コンテナ | 仮想マシン |
|---|---|---|
| 停止時 | 0 sec | 0 sec |
| 起動時 | 11 sec | 17 sec |
| 演習時 | 16 sec | 23 sec |
| 7 日後 | 999 sec | 1456 sec |

その時得られた CPU・メモリの使用量を，**表 3** および

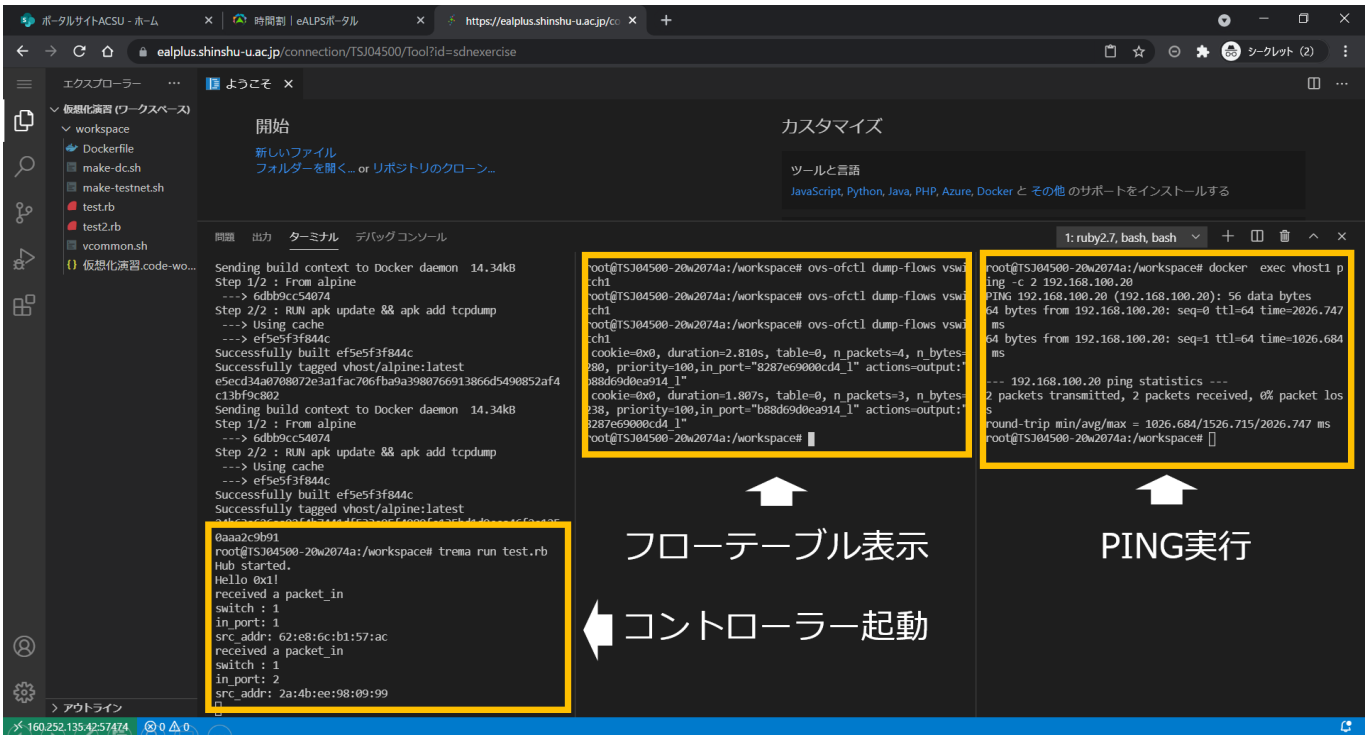


**図 3** 実装した環境画面
**Fig. 3** Screenshot of the implemented SDN training environment.

**表 4** SDN 演習環境のメモリ使用量
**Table 4** Memory usage of SDN exercise environment.

| | コンテナ | 仮想マシン |
|---|---|---|
| 起動時 | 0MB | 0MB |
| 起動時 | 395.51MB | 670.83MB |
| 演習時 | 471.77MB | 809.96MB |
| 7 日後 | 257.59MB | 959.06MB |

**表 4** に示す．CPU 使用量とは，停止時から計測時点の範囲における演習環境が利用した CPU の実行累積時間である．また，メモリ使用量は計測時点で演習環境が利用した実メモリ使用量である．

まず，起動時には仮想マシンの方が CPU・メモリの使用量が多く，複数の演習環境の起動には大きな負荷がサーバに発生する．一方，演習時のプログラム実行には秒単位で見た場合，CPU の処理量に変化は見られない．さらに，7 日後の SDN 演習環境の状態に着目すると，仮想マシンの方が待機状態で 1.4 倍程度の CPU を使用していることがわかる．コンテナのメモリ使用量が減少しているのは，メモリ上のファイルキャッシュが破棄された結果だと考えられる．

以上から，ハードウェア仮想化よりコンテナ仮想化技術の方が，CPU・メモリが効率的であることを確認した．

一方，仮想マシンでは徐々にメモリ使用量が増加することが予想され，この現象はサーバのメモリ不足を引き起こすと考えられる．また，演習時の CPU 利用状況から，今回構築した演習コンテナは今までの演習環境と同様のパフォーマンスを提供できると考えられる．

### 4.3.2 授業実践による評価

信州大学の 2021 年前期・仮想化技術特論 [8] においては毎週 1 回 90 分間，SDN 演習を行った．この授業では，学習者自身が持ち込んだ端末で演習環境に接続し，図 4 のような複雑なネットワーク演習を行う．本研究で作成したコンテナ演習環境を利用し，53 人の学習者が演習を行った際の負荷状況を取得した．演習サーバとして利用したシステムの構成は表 1 に示す．

負荷状況を図 5，6，7，8 に示す．

CPU の負荷状況に着目すると，演習環境は CPU の 1 スレッドの処理内で収まっており，負荷はかなり低いことがわかる．またメモリに着目すると，利用当初は 36GB 程度使用していたが，2 週間後は 28GB 程度とメモリ使用量の減少が見られる．これは前述したファイルキャッシュが破棄によるものだと考えられる．

演習サーバのメモリ量から計算すると，200 人程度の演習環境の収容ができる．また，台数増加に伴う演習環境の速度低下の現象は見られなかった．前年度に行った演習では 20 人程度が限界であったため，仮想マシンと比較して 10 倍程度効率が向上する事が考えられる．

## 5. おわりに

SDN・SDI 演習環境をコンテナ仮想化により実行できる環境を作成し，演習サーバに効率的に収容可能な演習環境を提案した．また実際の SDN 演習に適用することで，本

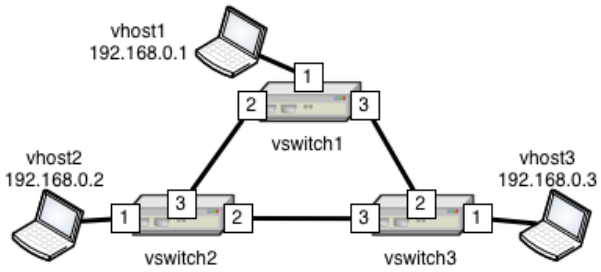


**図 4** 複雑なネットワークの例
**Fig. 4** Example of a complex network.

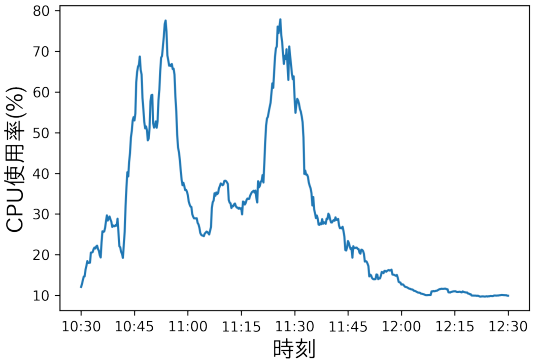


**図 5** 5 月 7 日 CPU 使用量
**Fig. 5** May 7 CPU usage.

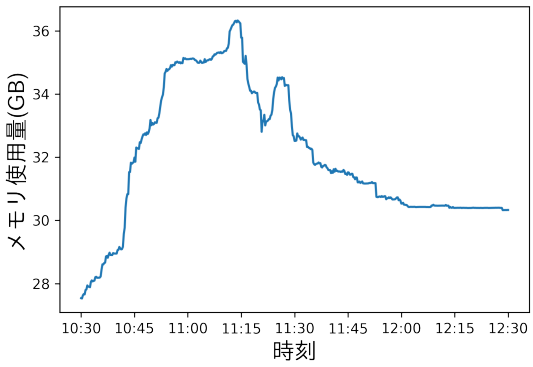


**図 6** 5 月 7 日 メモリ使用量
**Fig. 6** May 7 Memory usage.

研究で提案したシステムが演習環境を提供する機能を有することがわかった.

今後は，今回提案したコンテナによる SDN・SDI 演習環境が，より多くの仮想化技術演習の演習が可能であるか検証を行う．さらに，実用的な演習システムを構築するために Learning Management System などとの連携などの検討も行う.

**謝辞** 本研究は JSPS 科研費 18K02897 の助成を受けたものです..

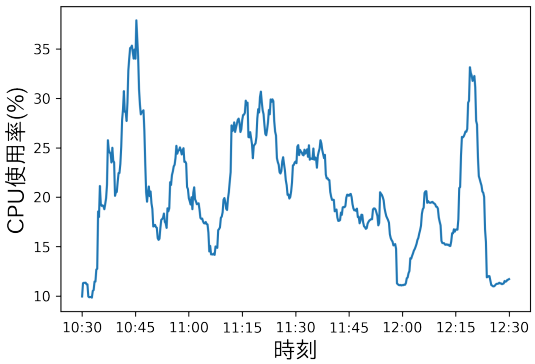


**図 7** 5 月 21 日 CPU 使用量
**Fig. 7** May 21 CPU usage.

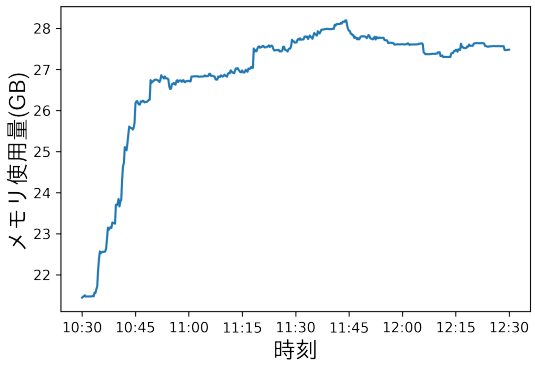


**図 8** 5 月 21 日 メモリ使用量
**Fig. 8** May 21 Memory usage.

## 参考文献

[1] 田中篤志ほか：LTI とリバースプロクシの連携による演習サーバ接続システム，研究報告教育学習支援情報システム（CLE）(2018).
[2] 井口信和：仮想ルータを活用したネットワーク構築演習支援システムの開発，情報処理学会論文誌，Vol. 52, No. 3, pp. 1412–1423 (2011).
[3] Bui, T.: Analysis of Docker Security, *CoRR*, Vol. abs/1501.02967 (online), available from ⟨http://arxiv.org/abs/1501.02967⟩ (2015).
[4] Docker, I.: why-docker, Docker, Inc. (online), available from ⟨https://www.docker.com/why-docker⟩ (accessed 2021-06-18).
[5] 東賢一朗：プロセススケジューラが仮想マシンへ与える影響に関する考察，第 73 回全国大会講演論文集，Vol. 2011, No. 1, pp. 25–26 (2011).
[6] linuxcontainers.org: LXD News, Canonical Ltd (online), available from ⟨https://linuxcontainers.org/ja/lxd/news/⟩ (accessed 2021-06-08).
[7] CODER: code-server, CODER TECHNOLOGIES INC. (online), available from ⟨https://github.com/cdr/code-server⟩ (accessed 2021-06-13).
[8] 新村正明：仮想化技術特論，Shinshu University（オンライン），入手先 ⟨https://campus-3.shinshu-u.ac.jp/syllabusj/Display?NENDO=2021&BUKYOKU=TS&CODE=⟩（参照 2021-06-18）.