**Research on multiplexing of remote experiments and training systems
using container virtualization technology**

Aiki Shirahata\*,a, Daiki Koidea, a)Masaaki Sasakia

National Institute of Technology, Sendai College

\*a2211514@sendai-nct.jp

**Abstract**

**While the number of online classes is increasing due to the spread of COVID-19 infection, it is difficult to implement practical classes using special equipment such as electrical circuit practice and experiment remotely. In a remote learning system using actual equipment, the number of students is large and the hardware scale of individual systems (servers) is large, which increases the time cost and hardware resources required to build and multiplex the experimental environment. This research aims to reduce both hardware and management costs by multiplexing experimental training systems on containers. Therefore, this research will minimize server hardware by using container virtualization technology for server construction. Container virtualization is a technology that creates an area called a container on the host OS and runs one container as one virtual machine, allowing the virtual machine to run with fewer resources than other virtualization techniques. In this research, a remote experiment system is constructed to learn digital circuit design with FPGA (Field Programmable Gate Array) board. Learners access the server container from a client PC via a browser to code, compile, and write a circuit configuration files to the practice board. The results of the experiment can be confirmed by immediately displaying the video captured by the FPGA board on the browser. The system is designed to enable this type of training. Build as many training containers as the number of training boards connected to the server using Raspberry Pi 3 to which the training boards and cameras are connected via USB. Docker will be used to build the training containers. It will implement and deploy a web server, web application, and FPGA development environment. After introducing the system, we will ask students to use this remote training system and evaluate usability and comprehension through questionnaires. In addition, the system will be compared with other virtualization methods and evaluated for resource reduction.**

**Keywords:** *Containers, Virtualization, Remote Experiment, Multiplexing*

**Introduction**

Remote teaching has become possible through the development of video calls and web conferencing tools. With the spread of COVID-19 infection, educational institutions have developed remote teaching environments, which will continue to be used in the future. Jacko (2022) found it difficult to remotely conduct practical training using specialized equipment, such as circuit training and electrical laboratory exercises, and most training is conducted by simulation. However, a survey of students conducted at the Technical University of Košice shows that practical training using actual equipment is more desired than simulation. As a leading example of remote experimentation and practical training, Winzker (2020) offers remote classes on digital hardware image processing using FPGA boards at Bonn Rhein-Sieg University of Applied Sciences.





Figure 1 FPGA Remote Lab Overview

Nakata (2019) found that when a remote training system using experiment and practice boards is designed as a one-to-one user and server, the hardware scale becomes too large for a large number of learners to use, and the time cost and hardware resources increase for the development of the experiment environment and multiplexing of the system. In this study, container virtualization technology is utilized for server construction in order to minimize the hardware requirements of servers for remote practice environments. The objective is to multiplex experimental and practical training systems on containers to enable the reduction of both hardware and management costs.

　This paper describes the configuration and implementation of a system that enables the remote operation of experiment and practice boards and real-time video confirmation of the operation results. This study proposes an online training system that utilizes containerized servers. We will implement the system, have users build the system, and collect questionnaires on the usability and multiplexing of the system. Then, we compare and evaluate the proposed system and the conventional system and show the superiority of the proposed system.

**Virtualization of remote experimental systems using Docker**

This study uses a container virtualization method as a virtualization method for individual experimental systems. Figure 2 shows the structure of the container-based virtualization method. Container virtualization creates an area called a container on the host OS and operates one container as one virtual machine. Table 1 shows a comparison with other virtualization methods. Figure 2 shows the configuration of container virtualization.



Figure 2 Container virtualization configuration

Table 1 Characteristics of each virtualization method

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Virtualization method** | **GuestOS** | **Separation Level** | **Overhead** | **Augment ability** |
| Hyper Visor | Required | High | Big | Low |
| Host OS | Not required | High | Big | Low |
| Containers | Not required | Low | Small | High |

Host OS and hypervisor virtualization methods require a guest OS on the virtual machine when building a virtual machine on the host OS. This increases the independence of each virtual machine, but it tends to occupy a large amount of resources and increase the overhead. In addition, the scalability is low because a guest OS must be installed and initially configured each time a virtual machine is built. On the other hand, container-based virtualization does not require a guest OS, so virtual machines are less independent than host- and hypervisor-based virtualization but more scalable. In addition, virtual machines occupy fewer resources, allowing more efficient use of hardware resources such as CPU and memory, and have less overhead.

Docker is an open-source platform for creating, distributing, and running containers. Figure 3 shows an overview of Docker.



Figure 3 Overview of Docker

Docker builds a container by encapsulating an application and its execution environment using a static image on the file system called a Docker image. Users can use a Docker image creation function called Dockerfile to create a container image by themselves, which creates a file named Dockerfile and describes the base Docker image and commands to be executed in code-like format. Dockerfile is created by executing the build command from the Dockerfile in the Docker daemon.

Docker-Compose can be used to manage multiple containers by creating and configuring multiple container images at the same time. When building a container with Docker-Compose, users can create a directory structure as shown in Figure 4, and describe the basic information and configuration of the container to be created in docker-compose.yml. This allows for multiple containers to be multiplexed by specifying directories to be shared by multiple containers and by specifying container settings in a batch.



Figure 4 Directory structure when building containers

Figure 5 shows the flow of the remote practice to be implemented in this study. An FPGA board is used as a practical example to implement a digital circuit operation experiment designed in Verilog HDL. The experimental procedure is as follows:

1. Students write a circuit in Verilog HDL.
2. The local circuit synthesis environment is used to generate circuit connection information for the FPGA.
3. Connect online to the training system server and upload the circuit connection information.
4. Confirm the operation of the FPGA board displayed on the screen in real time by operating switches from the screen on the practical training system.



Figure 5 Assumed remote training flow

This digital circuit is implemented utilizing container virtualization. Figure 6 shows an overview of the training system.



Figure 6　 Overview of the training system

The host machine which serves as the server has multiple FPGA boards and cameras connected via USB. FPGA boards and cameras are the units of the training system and the server deploys the same number of containers as the number of training systems connected to the host machine.

When a student connects to the training system from a client, a screen that is a web application is displayed. This screen has the following functions:

・Upload Verilog HDL source code or circuit connection information

・Streaming display of the board

・Switch operation on the experimental board

The implementation of this system is described below.

**Implementation of a digital circuit remote training system**

The host machine is a Raspberry Pi3 Model B+ and the OS is Ubuntu Server 20.04. The development environment for the FPGA board is Altera's Quartus II Web Edition, which is an FPGA-integrated development environment by former Altera. We chose Raspberry Pi3 as the host machine because it is small and inexpensive, its low performance makes it easy to grasp the resource usage status when multiple students are connected to it, and it is easy to experiment with its operation response. The GPIO pins are used to simulate the operation of switches and buttons on the FPGA board.



Figure 7 DE2-70

Figure 9 shows the configuration of the containers. There is one training system for each FPGA board. The training system consists of a container that plays the role of a web server, a container that stores a web app implemented in Django ("Django container"), and a Quartus container that stores the functions of Quartus, an FPGA integrated development environment. We implemented the web server container, the Quartus container, and the web app container together using Docker-Compose, and built one web server per training board. In the initial configuration of this remote training system, we built one web server container and multiple Quartus containers for writing FPGAs on the host machine as shown in Figure 8, and the user selected the container to use after accessing the web server. However, in this method, the access to one Web server is concentrated and the independence of each experimental system is lost. Therefore, the simplicity of multiplexing, which is an advantage of container virtualization, cannot be satisfied.



Figure 8 Initial container configuration



Figure 9 Container configuration after improvement

In the implementation phase, we implemented a web server container, a Django container, and a Quartus container. All of these containers were built together using Docker-Compose and Dockerfile.

The web server container is based on an Nginx container image, and port forwarding is configured in Docker-Compose. Figure 10 shows the port forwarding configuration.



Figure 10 Port forwarding overview

When accessing the web server in the container from a client PC, we access the IP address of the host machine instead of the IP address of each Docker container. Therefore, when building multiple Nginx servers on a single host machine, we can configure port forwarding between the Nginx container and the host machine and build multiple web servers to be accessed by passing any port on the host machine to port 80 of the container.

The Quartus container was built based on Ubuntu in the Dockerfile, with Quartus II and the libraries necessary to run Quartus II installed. Quartus II executes coding, compilation, etc. on the GUI, and compiles and writes files on the command line, but it also provides an environment for executing these operations with console commands. This container contains the execution environment.

The Django container contains a web app implemented in Django, which is responsible for uploading files, creating project files, coding, writing files to the FPGA board, and other interface functions of the FPGA development environment. Figure 11 shows a mock-up of the Web app interface.



Figure 11 Mock-up of the Web app interface

We implemented a program in Python to write files uploaded from the web app to the FPGA board, and the Linux version of Quartus has a function to compile files by typing commands from the command line. We used Python's "subprocess" library to execute Linux commands from Django in a Quartus container based on Ubuntu Server 20.04. We need the USB port ID of the FPGA board when writing the file (project file) that contains the circuit connection information after compilation to the FPGA board by the Linux command. Therefore, the "lsusb" command includes a process to reference the USB port ID based on the device ID in advance.

We implemented these containers and confirmed that the project files were successfully written to the FPGA on the FPGA board. We built two containers and configured port forwarding between port 8000 and 8001 of the host machine and port 80 of the container, respectively, and connected two FPGA boards to Raspberry Pi and confirmed that writing was executed on each.

**Conclusions**

This study describes a method of implementing a remote practice system using container-based virtualization. a digital circuit design practice system using FPGA boards was constructed.

In the current container configuration, one FPGA board uses three containers internally. The system is built using Docker-Compose, which is easy to manage and maintain, but it has the disadvantage of creating three base Docker images, which has more overhead than if it were consolidated into a single image. In order to determine the optimal host server hardware and container implementation configuration, we would like to implement all functions in an Ubuntu Server-based container, compare the two in terms of processing speed, resource usage, and ease of maintenance, and determine appropriate specifications.

We will further implement the remote experiment system, increase the number of multiplexing units, have students use the system, and conduct a questionnaire on the ease of use and learning effectiveness of the remote experiment system to identify issues.

We plan to make improvements to both the hardware/web application to address the issues.

**References**

Jacko, P. & Beres, M. & …, Kovac, D. (2022). Remote IoT Education Laboratory for Microcontrollers Based on the STM32 Chips. *MDPI Sensors.*

Winzker, M. (2020). *FPGA Vision Remote Lab.* Retrieved from
https://www.h-brs.de/de/fpga-vision-lab

Nakata, R. & Otsuka, R. (2019) Proposal of container-based virtualization Utilization to cyber security training environment “Cyber Range”. *Information Processing Society of Japan.*