JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 11, pp. 51-58, November 2020
https://doi.org/10.9708/jksci.2020.25.11.051

Database PasS web service system using Docker

Tai-Sung Hur*

*Professor, Dept. of Computer Science, Inha Technical College, Incheon, Korea

[Abstract]

Most of the students in computer-related departments work on projects, and it is essential to use a database for project execution. To use such a database, it is necessary to install a Database Management System. However, it takes several minutes (hours) to install a DBMS, and some DBMS require a difficult installation process. Therefore, in order to solve this problem, this study proposed a system that can easily install DBMS using Docker.

Docker is an open source project that automates the deployment of Linux applications into software containers. Docker Container is wrapped in a complete file system that includes everything necessary for the execution of software, and includes code, runtime, system tools, system libraries, and anything that is installed on the server. This guarantees that they will always run the same regardless of the environment in which they are running.

After creating a database using this proposed system, you can check the database access information on the web and check the server status in minutes.

As a result of of implementing this proposed system and applying it to the projects of 10 teams, the installation time was reduced by 94.5% for Maria DBMS and 98.3% for Oracle DBMS than individual installation, confirming improved efficiency.

▸ **Key words**: Docker, Database, PasS(Platform as a Service)

[요 약]

컴퓨터 관련학과의 학생들은 대부분 프로젝트를 수행하며, 프로젝트 수행에 있어 데이터베이스 사용은 필수적이다. 이런 데이터베이스를 사용하기 위해서는 데이터베이스 관리 시스템(DataBase Management System)의 설치가 필요하다. 그러나 DBMS를 설치의 경우 수분(시간)이 소요되며, 까다로운 설치 과정을 요구하는 DBMS도 있다. 따라서 이러한 문제를 해결하기 위해 본 연구에서는 Docker를 이용해 DBMS를 쉽게 설치할 수 있는 시스템을 제안하였다.

Docker는 리눅스의 응용 프로그램들을 소프트웨어 Container안에 배치하는 일을 자동화하는 오픈 소스 프로젝트이다. Docker Container는 소프트웨어의 실행에 필요한 모든 것을 포함하는 완전한 파일 시스템 안에 감싸며, 안에는 코드, 런타임, 시스템 도구, 시스템 라이브러리 등 서버에 설치되는 무엇이든 포함된다. 이는 실행 중인 환경과 관계없이 언제나 동일하게 실행될 것을 보증한다.

본 제안 시스템을 이용하여 데이터베이스를 생성한 후 웹에서 데이터베이스 접속 정보를 확인할 수 있으며, 서버 상태를 분 단위로 확인할 수 있도록 하였다.

본 제안 시스템을 구현하여 10팀의 프로젝트에 적용한 결과 개별 설치보다 Maria DBMS의 경우 94.5%, Oracle DBMS의 경우 98.3%의 설치 시간이 감소하여 향상된 효율을 확인하였다.

▸ **주제어**: 도커(Docker), 데이터베이스(Database), PasS(Platform as a Service)

• First Author: Tai-Sung Hur, Corresponding Author: Tai-Sung Hur
*Tai-Sung Hur (tshur@inhatc.ac.kr), Dept. of Computer Science, Inha Technical College
• Received: 2020. 09. 24, Revised: 2020. 10. 30, Accepted: 2020. 11. 01.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

오늘날 컴퓨터 관련 학과 프로젝트의 경우 대부분 데이터베이스를 사용한다. 데이터베이스란 여러 사람에 의해 공유되어 사용될 목적으로 통합하여 관리되는 데이터의 집합을 의미하며, 데이터베이스를 관리하는 시스템을 DBMS(Data Base Management System)라고 한다. 근래에 들어서는 Oracle, Mysql, Microsoft SQL Server 등 여러 DBMS가 등장하였으며, 개발자는 데이터의 특성을 고려하여 적합한 DBMS를 사용한다.

컴퓨터 관련학과 학생들이 개발 프로젝트를 진행할 때 DBMS를 사용하게 되며, 대부분 학생 개개인의 PC에 DBMS를 직접 설치하여 운영하게 된다. 학생들이 개인의 PC에 DBMS를 직접 설치하여 운영한 후, 프로젝트가 끝났을 때 대부분의 DBMS는 백업이 진행되지 않은 채 삭제된다. 또한, DBMS마다 설치 과정이 다르며, 설치 과정이 까다로운 DBMS 또한 존재한다. 이러한 문제를 해결하는 방법의 하나는 경량의 오픈 플랫폼인 Container 기반의 Docker이다[1].

Docker는 리눅스의 응용 프로그램들을 소프트웨어 Container 안에 배치하는 일을 자동화하는 오픈소스 프로젝트이다. Docker Container는 소프트웨어의 실행에 필요한 모든 것을 포함하는 완전한 파일 시스템 안에 감싸며, 안에는 코드, 런타임, 시스템 도구, 시스템 라이브러리 등 서버에 설치되는 무엇이든 포함된다. 이는 실행 중인 환경과 관계없이 언제나 동일하게 실행될 것을 보증한다[2,3].

Docker의 Container를 이미지화하여 배포하는 DockerHub 페이지가 존재하며, 해당 페이지에서 Oracle, MariaDB, MongoDB 등 회사 또는 사용자들이 배포한 이미지들을 받을 수 있다. 해당 이미지들을 받아 사용하게 되면, 수 분(시간) 걸리던 설치 시간을 단축할 수 있으며, Docker Container를 통째로 백업, 복구가 가능하므로 편하게 DBMS를 백업 및 복구를 할 수 있다. 삭제 또한 Docker Container를 삭제하면 시스템에 반영되므로 매우 간단하다.

Docker는 Linux 기반 환경에서 동작하는 프로그램이며, 윈도우 환경을 위한 설치 방법이 별도로 준비되어있다. Docker를 윈도우에서 사용하기 위해서는 VirtualBox를 통해 리눅스 이미지를 생성하여, 생성된 리눅스 이미지 위에 Docker를 동작시키게 된다. 공식으로 배포되는 Docker ToolBox를 사용하면 간단하게 VirtualBox와 Docker를 설치할 수 있다.

본 연구에선 이와 같은 로컬 데이터베이스를 운영하는 혹은 운영하게 될 소규모 팀을 위하여 웹 환경에서 간단한 절차를 통해 원하는 DBMS 이미지의 Docker Container를 서버에 생성할 수 있으며, 생성된 Docker Container DBMS는 사용자가 접근할 수 있도록 IP 주소, 포트번호, 계정 아이디, 비밀번호를 알려준다. 이러한 Docker의 데이터베이스 Container 생성 과정은 웹에서 진행하며, 간단한 회원 가입, 로그인을 통해 이용할 수 있다.

시스템 관리자는 사전에 실제 서버와 웹 시스템을 연동해야 하며, 시스템 연동을 위한 Python 프로그램과 NodeJS 프로그램을 서버에서 동작시켜야 한다.

시스템 연동에 사용되는 Python 프로그램은 서버의 상태를 파악하기 위한 모니터링을 위해 사용되며, NodeJS로 작성된 프로그램은 Docker를 원격에서 제어하기 위해 사용된다. 두 프로그램은 서버 관리자가 미리 세팅해두어야 한다.

따라서 컴퓨터 관련학과 학생들이 프로젝트를 진행할 겻우 제안 시스템을 이용한다면 별도의 DBMS를 설치할 필요가 없으며, 소규모 팀 프로젝트도 편리하게 이용할 수 있다.

II. Related works

1. Docker

Docker는 Linux의 응용 프로그램들을 소프트웨어 Container 안에 배치하는 일을 자동화하는 오픈소스 프로젝트이며, 다양한 실행환경을 컨테이너로 추상화하고 동일한 인터페이스를 제공하여 프로그램의 배포 및 관리를 단순하게 해준다. 백엔드 프로그램, 데이터베이스 서버, 메시지 큐 등 어떠한 프로그램이라도 컨테이너로 추상화할 수 있다[4].

2. Image

이미지는 컨테이너 실행에 필요한 파일과 설정값 등을 포함하고 있는 것으로 상태 값을 가지지 않고 불변(immutable)하다. 컨테이너는 이미지를 실행한 상태라 볼 수 있게 추가되거나 변하는 값은 컨테이너에 저장된다. 같은 이미지에서 여러 개의 컨테이너를 생성할 수 있으며, 컨테이너의 상태가 바뀌거나 삭제되더라도 이미지는 변하지 않는다[4].

Docker의 이미지는 컨테이너를 실행하기 위한 모든 정보를 가지고 있으므로 용량이 꽤 큰 편이며, 이러한 이미지를 조금 수정하였다고 다시 받는 것은 비효율적이다. 해당 문제를 해결하기 위해 Docker는 Layer 개념을 사용하고 Union File System을 이용해 여러 Layer를 하나의 파

일 시스템으로 사용할 수 있게 해준다[5]. 이미지는 Fig. 1에서 보는 바처럼 여러 개의 읽기 전용 Layer로 구성되고, 파일이 추가되거나 수정되면 새로운 Layer가 생성되는 형태로 이미지가 관리된다.

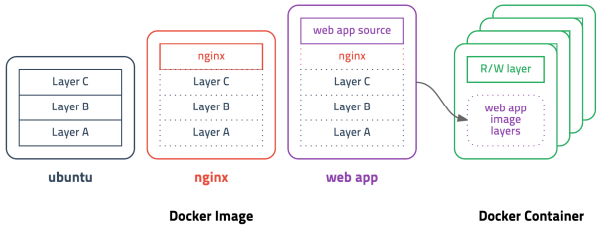

Fig. 1. Image layer storage method

3. Container

Container는 격리된 공간에서 프로세스가 동작하는 기술이다. Container는 가상화 기술의 하나로써 기존의 가상화 방식과는 차이를 둔다. HPC 분야에도 클라우드 컴퓨팅을 적용이 확대되고 있으며, 자원의 유연한 사용과 함께 HPC 워크로드들의 플랫폼 독립성을 위해 Container를 적용하여 다양한 사용자의 요구사항과 인프라 환경에 대응하고 있다[5].

기존의 가상화 방식은 주로 운영체제 가상화였으며, VMWare나 VirtualBox 같은 가상머신은 호스트 운영체제 위에 게스트 운영체제 전체를 가상화하여 사용하는 방식이다. 이 방식은 여러 가지 운영체제를 가상화할 수 있고, 비교적 사용법이 간단하지만 무겁고 느려서 실제 운영환경에선 사용하기 어려웠으며, 이러한 상황을 개선하기 위해 CPU의 가상화 기술(HVM)을 이용한 KVM(Kernel based Virtual Machine)과 반가상화(Paravirtualization) 방식의 Xen이 등장하였다. 이러한 방식은 게스트 운영체제가 필요하긴 하지만 전체 운영체제를 가상화하는 방식이 아니기 때문에 호스트형 가상화 방식에 비해 성능이 향상되었다. 이러한 기술들은 OpenStack 이나 AWS, Rack-space같은 클라우드 서비스에서 가상 컴퓨팅 기술의 기반이 되었다.

Linux는 추가적인 운영체제를 설치하여 가상화하는 방법의 성능 문제를 개선하기 위해 프로세스를 격리하는 방식이 등장한다. 리눅스에서는 이 방식을 리눅스 Container라고 하며 단순히 프로세스를 격리하기 때문에 가볍고 빠르게 동작한다. CPU, 메모리는 프로세스가 필요한 만큼만 추가로 사용하며, 성능적으로 손실이 거의 없다.

Fig. 2에서 보는 바와 같이 하나의 서버에 여러 개의 Container를 실행하면 서로 영향을 미치지 않고 독립적으로 실행되어 마치 가벼운 VM(Virtual Machine)을 사용하는 느낌을 준다. 실행 중인 Container에 접속하여 명령어를 입력할 수 있고 aptget이나 yum으로 패키지를 설치할 수 있으며 사용자 추가 및 여러 개의 프로세스를 백그라운드로 실행할 수도 있다. CPU나 메모리 사용량을 제한할 수 있고, 호스트의 특정 포트와 연결하거나 호스트의 특정 디렉터리를 내부 디렉터리인 것처럼 사용할 수도 있다. 새로운 Container를 만드는 데 걸리는 시간은 겨우 1-2초로 가상머신과 비교도 할 수 없이 빠르다.

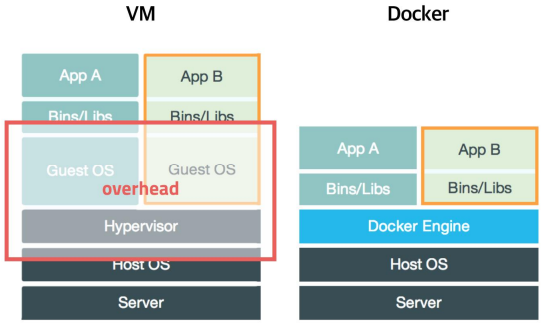

Fig. 2. Virtualization method of VM and Docker

이러한 Container라는 개념은 Docker가 처음 만든 것은 아니며, Docker가 등장하기 이전에 프로세스를 격리하는 방법으로 리눅스에서는 cgroups(control groups)와 name space를 이용한 LXC(Linux container)가 있었고 Free BSD에선 Jail, Solaris에서는 Solaris Zones라는 기술이 있었으며, Docker는 LXC(Linux container)를 기반으로 시작해서 0.9버전에서는 자체적인 libcontainer 기술을 사용하였고, 추후 runC(OCI 사양에 따라 Container를 생성하고 실행하기 위한 CLI 도구) 기술에 합쳐졌다[6].

III. System configuration

1. System Specification

본 개발 시스템의 개발환경은 아래와 같다.

Table 1. System development environment

시스템 개발 환경	
O.S	Window 10, Linux(Docker)
language	Java (jdk1.8)
web framework	SpringBoot
database	MariaDB 10.3.13
web container	Tomcat8

본 개발 시스템의 일반 사용자의 프로세스는 Fig. 3과 같으며 총 7개의 프로세스가 존재한다.

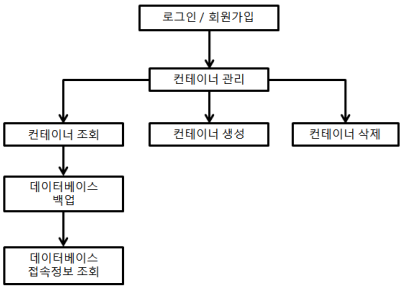

Fig. 3. Web system configuration (User)

Fig. 3에서 보는 바와 같이 사용자는 회원 가입 후 로그인 절차를 거쳐 아주 간단하게 제안 시스템을 이용하여 프로젝트 수행을 위한 DBMS를 설치할 수 있으며, 소규모 팀 프로젝트를 진행할 경우에도 사용자 ID를 공유하여 동일한 데이터베이스를 이용할 수 있다.

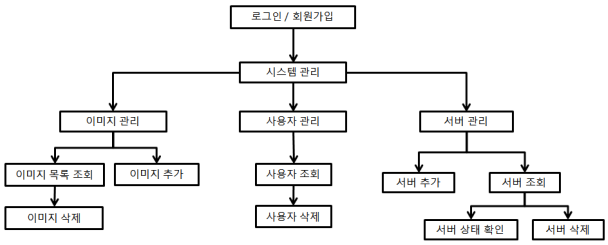

Fig. 4. Web system configuration (Administrator)

Fig. 4는 시스템 관리자 측면에서의 시스템 구성도이다. 14개의 프로세스로 이루어져 있으며, 일반 사용자와 달리 본 시스템을 관리하는 웹 시스템 관리 기능을 사용할 수 있으며, 시스템 관리 기능으로는 이미지 관리, 사용자 관리, 서버 관리가 있다. 이미지 관리에서는 사용자가 생성하게 될 Container의 이미지를 조회, 추가, 삭제하며, Docker Hub에 공개되어있는 Maria DB[7], Oracle[8] 이미지를 사용한다. 사용자 관리에서는 사용자 목록 조회, 사용자들의 Container 목록 조회, 사용자 삭제 기능이 존재한다. 서버 관리 기능에서는 Container가 생성될 리눅스 서버 정보를 추가, 삭제, 조회하게 되는데 웹에서 리눅스 서버 정보를 추가하기 전에 추가하고자 하는 서버에 미리 Docker와 웹 시스템에 연동하는 데 필요한 프로그램을 미리 설치해두어야 한다.

2. System ERD

Fig. 5에서 보는 바와 같이 본 시스템은 총 5개의 테이블로 구성되어있다. 본 시스템을 이용하는 사용자를 관리하는 사용자 테이블과 사용자의 권한을 관리하는 권한 테이블, 서버 정보를 관리하는 서버 테이블, 사용자가 생성하는 Container를 관리하는 Container 테이블, 서버의 모니터링 정보를 조회하는 모니터링 테이블이 존재한다.

Fig. 5. System ERD(logical)

3. Node Program Specification

웹 요청을 통해 원격에서 Container 생성 및 삭제, 관리하기 위하여 NodeJS[9] 환경을 활용하였으며, 프로그램 개발에 사용된 의존 라이브러리는 Table 2와 같다.

Table 2. NodeJS dependent library

Library	Function
express	NodeJS web application frame- work
body-parser	Parsing user request data into Json
node-docker-api	NDocker Remote API support in NodeJS

4. Node Program API List

API 요청을 통해 사용할 수 있는 기능으로는 Table 3에 표현하였다. Table 3에서 보는 바와 같이 웹 시스템에서 사용자가 웹에서 Container 관련 기능을 요청하면 시스템에서 사용자가 요청한 기능과 관련된 API를 Node Program에 요청한다.

Table 3. NodeJS API function list

Request API	Function
/list	Container list inquiry request
/listDetail	Container list and data inquiry request
/create	Container creation request
/delete	Container deletion request
/start	Container start request
/stop	Container stop request

5. Python Program Specification

서버의 Memory, CPU의 자원정보를 조회할 수 있는 기능은 Python으로 구현하였다(Table 4). Table 5의 요청 API를 통해 웹 시스템에서 서버의 자원정보를 조회하며, 관리자가 웹을 통해 서버의 자원정보를 도표로 조회할 수 있다.

Table 4. Python Library

Library	Function
Flask	Python's micro web framework
psutil	System monitoring function

Table 5. Python API function list

Request API	Function
/list	Container list inquiry request

IV. System Implementation

본 연구에서 웹 시스템 구현을 위해 SpringBoot, Node.js[9] 그리고 Maria[10]를 사용하였다.

본 개발 시스템은 사용자들의 데이터베이스 생성, 관리의 편의성을 위한 시스템으로 시스템의 이름은 DDPS (Docker Database Pass System)으로 명명하였다.

Fig. 6은 DDPS 시스템의 로그인 화면이다. 관리자가 생성해준 아이디와 비밀번호를 통해 로그인한다.

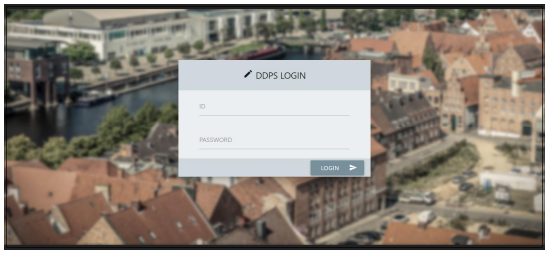

Fig. 6. Login screen

관리자는 로그인 후 서버관리, 사용자관리 화면에 접근할 수 있으며, 서버관리 화면은 Fig 7과 같다.

Fig. 7. Server management screen

서버관리 화면에서는 본 시스템과 연동한 서버들을 조회할 수 있으며, 서버정보 수정 및 삭제, 추가가 가능하다. Fig 8에서 보는 바와 같이 서버를 추가하기 이전에 해당 서버에는 Docker와 NodeJS 개발환경 및 NodeJS로 개발된 시스템 연동 프로그램, Python 개발환경 및 Python으로 개발된 모니터링 프로그램을 설치하여야 한다.

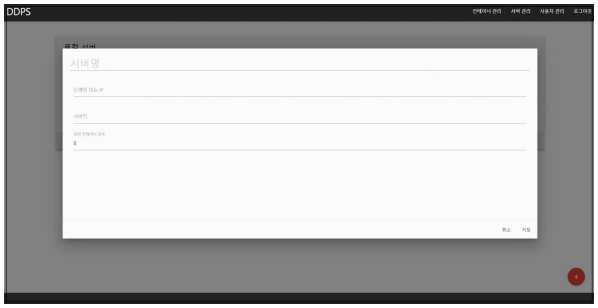

Fig. 8. Screen fo monotiring program

Fig. 9의 사용자 관리 화면은 관리자(ADMIN) 권한의 관리자가 접근할 수 있는 화면으로 사용자 추가, 수정, 삭제가 가능하다. 사용자의 권한을 변경할 수 있으며 본 시스템에서 사용하는 권한으로는 관리자(ADMIN), 사용자(USER) 권한이 있다. 관리자 권한의 사용자는 Container 관리, 서버관리, 사용자 관리가 가능하며, 사용자 권한의 사용자는 Container 관리 메뉴만 사용할 수 있다.

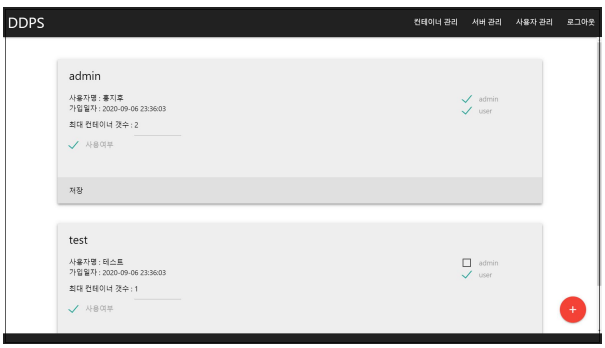

Fig. 9. User screen

Fig. 10은 Container 관리 화면이며, Fig. 11은 Maria DBMS의 Container 추가 화면이며, Fig. 12는 Oracle 11g DBMS의 Container 추가 화면이다.

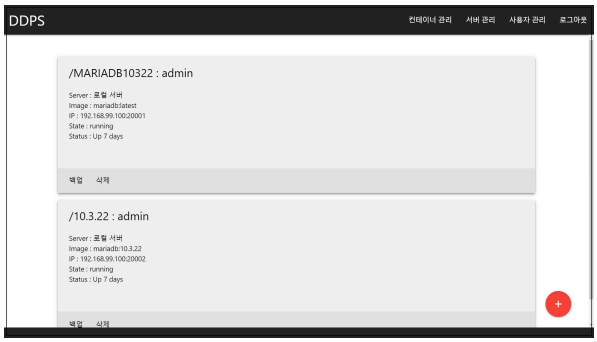

Fig. 10. Container add screen

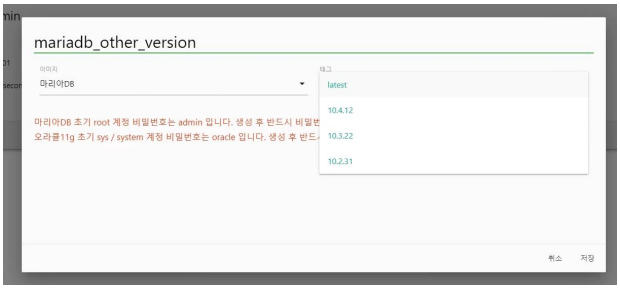

Fig. 11. Maria container add screen

Fig. 10은 Container 관리 화면에서는 사용자의 요청에 따라 데이터베이스 Container를 생성, 삭제할 수 있는 화면이다. 관리자 권한의 경우 모든 사용자의 Container를 확인할 수 있으며, 사용자 권한의 경우 자기 자신의 Container만 조회할 수 있다.

Fig. 12. Oracle 11g container add screen

Container 추가 시, Fig. 11에서 보는 바와 같이 Maria DB의 최근, 10.4.12, 10.3.22, 10.2.31 등 4개의 버전을 생성할 수 있으며, Fig. 12에서 보는 바와 같이 Oracle 11g의 Container를 선택하여 만들 수 있다.

생성된 Container는 생성 직후 바로 사용할 수 있으며, Maria DBMS의 경우 root 계정의 비밀번호는 admin으로 초기화되며, Oracle 11g의 경우 sys 계정의 비밀번호는 oracle로 초기화된다. 사용자는 생성된 데이터베이스에 초기 계정/비밀번호로 접근하여 비밀번호 수정 및 DB 설정을 진행하여야 한다.

V. Experiments

본 시스템을 이용하여 Oracle 11g 및 Maria DB를 설치하였을 경우와 직접 Installer를 받아 PC에 설치 한 경우의 속도를 측정하였다.

PC에 Installer를 다운로드 후, 데이터베이스 설치 소요시간을 측정하면 Table 6에서 보는 바와 같이 Oracle 11g의 경우 690초(12분 30분)가 소요되었으며, Maria의 경우 130초(2분 10초)가 소요되었다.

Table 6. DBMS installation time

DBMS	Installation time(second)
Oracle 11g	690
MariaDB	130

본 시스템으로 Oracle 11g와 Maria DBMS를 설치할 경우 이미지 다운 시간은 Table 7과 같으며, 본 시스템에 로그인 후 Container 관리 페이지에서 Container 다운 생성까지 소요된 사간은 Table 8과 같다.

Table 7. DBMS Installation time using the proposed method

DBMS	IInstallation time(second)
Oracle 11g	58
MariaDB	21

Table 8. DBMS Installation time

DBMS	Installation time(second)
Oracle 11g	6
MariaDB	5

일반적으로 컴퓨터 관련학과의 프로젝트 수업에 10명(팀)이 참여할 때 전체 DBMS 설치에 소용되는 시간은 Table 9과 같으며, Oracle 11g DBMS의 경우 Fig 13, Maria DBMS의 경우 Fig. 14로 표현하였다.

Table 9. Compare of individual installation and proposed method

(sec)

DBMS	installation method	1	5	10
Oracle 11g	individual installation	690	3450	6900
	proposed method	64	88	118
Maria	individual installation	130	650	1300
	proposed method	26	46	71

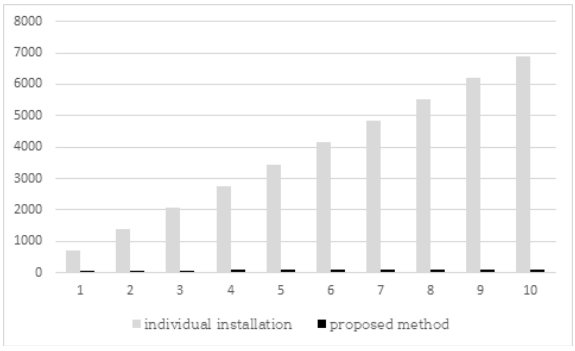

Fig. 13. Compare of individual installation and proposed method of Oracle 11g

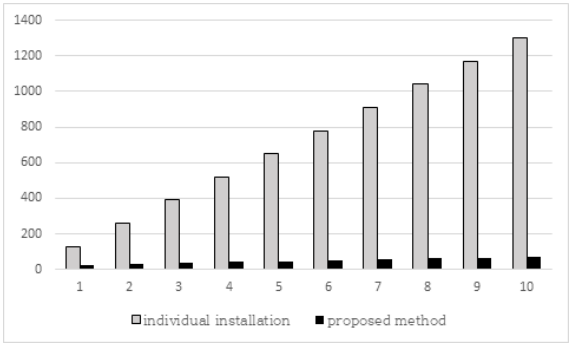

Fig. 14. Compare of individual installation and proposed method of Maria

Table 9에서 보는 바와 같이 10대의 컴퓨터에 DBMS를 설치할 때 Oracle 11g DBMS의 경우 제안 방법이 개별 설치와 비교해 약 1.7%의 시간이 소요되었으며, Maria DBMS의 경우 제안 방법이 개별 설치보다 약 5.5%의 시간이 소요되었다. 따라서 2개의 DBMS의 개별 설치와 본 시스템을 이용한 경우를 비교하면, Oracle 11g DBMS의 경우 98.3%, Maria DBMS의 94.5%의 설치 시간이 감소하였다.

VI. Conclusions

오늘날 컴퓨터 관련 학과에서는 실무 적응력을 향상하기 위해 시스템 개발을 위한 프로젝트 과목을 도입하고 있다. 이러한 프로젝트 수업에는 데이터베이스 사용이 필수적이다. 데이터베이스를 이용하기 위해서는 프로젝트 개발 시스템에 DBMS를 설치해야 한다. 그러나 DBMS를 설치하기 위해서는 DBMS를 다운받아야 하며, 설치에는 수(십) 분의 시간이 소요되며, DBMS마다 설치 과정이 다르며 설치 과정이 까다로운 DBMS도 있다.

따라서 본 연구에서는 Docker를 이용하여 이러한 설치 과정을 간소화한다.

Docker는 리눅스의 응용 프로그램들을 소프트웨어 Container 안에 배치하는 일을 자동화하는 오픈소스 프로젝트이다. Docker Container는 소프트웨어의 실행에 필요한 모든 것을 포함하는 완전한 파일 시스템 안에 감싸며, 안에는 코드, 런타임, 시스템 도구, 시스템 라이브러리 등 서버에 설치되는 무엇이든 포함된다. 이는 실행 중인 환경에 관계없이 언제나 동일하게 실행된다.

본 시스템을 구현하기 위해 Phython, Nodejs와 Maria DBMS를 이용하였다.

본 시스템을 구현 후 우리대학 컴퓨터정보과 2, 3학년 프로젝트 과목에 적용하였다.

학생들이 프로젝트 개발에 많이 사용하고 있는 Oracle 11g, Maria DBMS의 설치 시간을 비교한 결과, Oracle 11g의 경우 학생 개개인이 DBMS를 설치할 경우 690초의 시간이 소요되며, Maria를 설치할 경우 130초의 시간이 소요되나, 본 제안 방법으로 설치할 경우 이미지 다운에 Oracle 11g의 경우 58초가 Maria의 경우 21초가 소요되며, 이를 이용하여 Container를 다운 받아 생성하는데 소요되는 시간은 Oracle 11g가 6초, Maria가 5초이다. 따라서 10대의 컴퓨터에 설치할 경우 제안 방법이 10대의 컴퓨터에 적용할 경우 약 1.7%가 Maria의 경우 약 5.5%의 시간이 소요되어 제안 방법이 효율적임을 확인하였다.

ACKNOWLEDGEMENT

This work was supported by INHA TECNICAL COLLEGE Research Grant.

REFERENCES

[1] Bernstein, Dacid, “Containers and Cloid: FromLXC to Docker to Kubernetes”, Cloud Computing, IEEE, Vol. 1, No. 3, pp. 81-83, Sep. 2014.
[2] Wikipidia https://ko.wikipedia.org/wiki/Docker (software)
[3] Kayounggishere.loc https://velog.io/@hayoungishere/Docker- Kubernetes
[4] Coconut-obsessed https://velog.io/@matisse/도커Docker에-대해
[5] A. Younge, et al., “A Tale of Two Systems: Using Containers to Deploy HPC applications on Supercomputers and Clouds,”, in pric. of the IEEE International Conference on Supercomputers Technology & Science 2017 (CloudCom 2017).
[6] https://subicura.com/2017/01/19/docker-guide-for-beginners-1.html
[7] Docker Hub https://hub.docker.com/_/mariadb
[8] Docker Hub https://hub.docker.com/r/oracleinanutshell/oracle-xe-11g
[9] Node.js Express, https://expressjs.com/ko/
[10] Matia DB, https://mariadb.com/kb/ko/mariadb/

Authors

Tai-Sung Hur received the B.S degree in Dept. of Computer Science from Inha University in 1984, and M.S degree in Dept. of Computer engineering from Soongsil University in 1987, and Ph. D. degree in Dept. of Computer engineering from Inha University in 1992. Dr. Hur has over 30 years of computer education. He is currently a Professor in the Dept. of Computer Science, Inha Technical College. He is interested in Big data, Database and Internet of Things.