ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 24-02-01
j.inst.Korean.electr.electron.eng.Vol.28,No.2,123~130,June 2024
https://doi.org/10.7471/ikeee.2024.28.2.123

마이크로서비스아키텍처 기반 경량형 모의실험환경

A Ligthtweight Experimental Frame based on Microservice Architecture

함 규 식*, 김 현 기*, 김 진 우*, 장 수 영*, 김 은 경**, 최 창 범*★

Gyu-Sik Ham*, Hyeon-Gi Kim*, Jin-Woo Kim*,
Soo-Young Jang*, Eun-Kyung Kim**, and Chang-beom Choi*★

Abstract

As technology advances swiftly and the lifespan of products becomes increasingly short, there is a demand to fasten the pace of research outcomes, product development, and market introduction. As a result, the researchers and developers need a computational experiment environment that enables rapid verification of the experiment and application of research findings. Such an environment must efficiently harness all available computational resources, manage simulations across diverse test scenarios, and support the experimental data collection. This research introduces the design and implementation of an experimental frame based on a microservice architecture. The experimental frame leverages scripts to utilize computing resources optimally, making it more straightforward for users to conduct simulations. It features an experimental frame capable of automatically deploying scenarios to the computing components. This setup allows for the automatic configuration of both the computing environment and experiments based on user-provided scenarios and experimental software, facilitating effortless execution of simulations.

요 약

기술이 급속도로 발전하고 제품 수명주기가 짧아짐에 따라 연구 성과의 도출과 제품 개발 및 출시 과정을 가속화할 필요성이 점차 증대되고 있다. 이에 따라 개발자의 연구 결과를 빠르게 확인하고 적용하기 위한 모의실험을 위한 컴퓨팅 환경이 필요하게 되었다. 모의실험을 위한 컴퓨팅 환경은 가용한 컴퓨팅 자원을 최대한 활용할 수 있어야 하며 실험하고자 하는 다수의 시나리오에 대해서 모의실험을 관리하고 실험 결과 취합을 용이하게 진행해야 한다. 이와 같은 모의실험 환경을 구축하기 위해서 본 연구는 마이크로서비스 아키텍처 기반의 모의실험 환경을 설계하고 구현하였다. 제안하는 모의실험 환경은 모의실험을 수행하고자 하는 사용자가 손쉽게 실험을 수행할 수 있도록 스크립트 기반으로 가용 컴퓨팅 자원을 활용하여 실험 환경을 구성하고 자동으로 시나리오가 배포될 수 있도록 실험 틀을 설계하였으며 사용자가 제공하는 시나리오와 실험 대상 소프트웨어를 활용하여 컴퓨팅 환경과 실험이 자동으로 구성되어 모의실험을 수행할 수 있도록 하였다.

Key words : Microservice Architecture, Experimental Frame, Simulation, Discrete Event System, Information Management System

* Dept, of Computer Engineering, Hanbat National University(Researcher, Professor)
* Dept, of Artificial Inteligence Software, Hanbat National University(Professor)
★ Corresponding author
E-mail : cbchoi@hanbat.ac.kr, Tel : +82-42-821-1144
※ Acknowledgment
This work was supported by project for 'Customized techonolgy partner' funded Korea Ministry of SMEs and Startups in 2023. (project No. RS-2023-00283612)
Manuscript received Feb. 23, 2024; revised Mar. 17, 2024; accepted May. 23, 2024.

Ⅰ. 서론

현대 사회로 접어들면서 기술 발전이 빠르게 진행되고 제품의 수명주기가 짧아짐에 따라 연구 결과를 빠르게 도출하고 제품의 개발을 앞당기기 위해서 모의실험을 위한 컴퓨팅 환경을 구성하고 해결 대상 문제를 컴퓨터로 모델링하고 이를 실행하여 해결하는 모델링 및 시뮬레이션을 활용한 실험과 실험 환경의 중요성이 증가하고 있다[1][2]. 이에 따라 기술 연구자와 기술을 활용한 제품 개발자는 모의실험을 수행하기 위해서는 컴퓨팅 실험 환경을 구성하고 구성된 컴퓨팅 환경에 실험 목적에 맞는 시나리오와 모의 대상 컴퓨터 시뮬레이터를 배포함과 동시에 시뮬레이션을 실행하여 데이터를 확보하고 수집된 데이터를 바탕으로 분석을 수행하는 작업을 수행한다. 이와 같은 과정에 있어서 컴퓨팅 실험 환경을 구축하고 시나리오와 시뮬레이터의 배포 및 실험 관리에 대한 어려움이 있어 이를 극복하기 위한 다양한 연구가 제시된 바 있다[3-8]. 기존에 제안된 연구는 일반적으로 컴퓨팅 자원을 최대한 활용하기 위한 방법으로 분산된 컴퓨팅 자원을 활용하는 방법이나 실행 환경에 대한 가상화를 통하여 컴퓨팅 자원을 제공하고 가상화 기술을 활용하여 다양한 모의실험을 지원하는 방안으로 제시되었다. 하지만, 모의실험에 특화된 실험 환경을 연구 모의실험을 지원하기 위하여 제안된 방법에 있어서도 연구자와 제품 개발자를 추가적으로 구현하거나 부가적인 작업을 수행해야 한다는 제약사항이 있다.

본 논문에서는 연구자와 제품 개발자의 모의실험을 지원하기 위한 마이크로 아키텍처 기반의 경량형 모의실험 환경을 제안한다. 제안하는 방법은 대표적인 마이크로 아키텍처 기술인 도커(Docker)[9]를 사용하여 시뮬레이터의 실행 환경을 구성하는 방법으로 모의실험을 관리하기 위하여 이산사건시스템명세(DEVS: Discrete Event System Specification) 형식론[10] 기반의 제어기를 활용하고 이를 바탕으로 실험을 관리한다.

Ⅱ. 관련연구

2.1. IEEE1516 기반 DEXSim Framework

분산된 컴퓨팅 자원을 활용하여 실험을 관리하는 방법으로 IEEE1516 기반 DEXSim 프레임워크가 있다[3]. DEXSim 프레임워크는 이기종 시뮬레이션 간 연동의 국제 표준인 IEEE1516를 활용하여 분산된 컴퓨팅 자원을 활용할 수 있는 방법을 제공하였다. 일반적으로 이기종 시뮬레이션은 서로 다른 개발 환경과 실행 환경을 가지고 있으나 국제 표준인 IEEE1516을 통하여 연동 시뮬레이션을 수행하는 경우 분산 미들웨어를 바탕으로 이기종 시뮬레이터 간의 데이터를 주고받을 수 있는 프로토콜과 이를 지원하는 소프트웨어를 지원하여 연동 시뮬레이션 중에 데이터 동기화 및 시뮬레이션 모델 관리가 가능하다. DEXSim 프레임워크는 이기종 시뮬레이션 모델들이 상호 연동하여 하나의 시뮬레이션을 수행하는 형태로 분산 미들웨어를 사용하는 것이 아니라 단일 시뮬레이션 모델을 복제하여 연동 미들웨어에 참여시키고 각 시뮬레이션 모델에 수행해야 하는 시나리오 데이터를 동기화시키고 랜덤 시드 값과 실험 수를 변경하여 실험을 관리하는 형태로 모의실험을 관리한다.

DEXSim 프레임워크의 경우 분산 미들웨어가 실행되는 컴퓨팅 환경이 한번 정해지면 해당 환경에서만 동작하는 시뮬레이션 모델만이 실행된다는 제약사항이 있으며 연구자가 시뮬레이션을 수행하기 위해서는 분산된 컴퓨팅 환경에 대상 시뮬레이터와 시뮬레이터의 실행 환경을 수동으로 구축해야 한다는 제약사항이 있다. 본 논문에서 제안하는 경량형 실험 환경은 마이크로 서비스 아키텍처를 활용하여 시뮬레이터가 요구하는 실행 환경에 대한 설정을 자동화할 수 있으며 실험 환경 외부에서 시뮬레이터의 복제, 결과 수집을 관리할 수 있어 연구자와 개발자가 쉽게 모의실험을 진행할 수 있다.

2.2. 컨테이너 기반 다중 시뮬레이션 관리

컨테이너를 사용하여 다중 시뮬레이션을 관리하기 위한 방법으로 도커를 활용한 시뮬레이션 관리 측면에서 강화학습을 수행하기 위하여 다중 시뮬레이션을 관리하는 방법이 제시된 바 있다[5]. 해당 연구는 단일 컴퓨팅 환경에서 동일 시뮬레이션 인스턴스를 복수개 실행하지 못하기 때문에 이를 효율적으로 관리하기 위하여 각 시뮬레이션 인스턴스를 독립적으로 실행할 수 있도록 인스턴스를 격리시켜 복수 개의 시뮬레이션 인스턴스를 수행할 수 있도록 지원하는 방법이다. 이를 위하여 컨테이너 기반의 실험 관리 시스템으로 actor 컨테이너와 learner 컨테이너로 시뮬레이터를 구분하여 시뮬레이션을 수행할 수 있는 구조를 제안하였다. 제안된 구조는 하나의 컴퓨팅 환경에서 단일 학습에서 하나의 컴퓨팅 환경에서 고립된 복수 개의 학습 인스턴스를 관리하여 기존 학습보다 더 많은 양의 데이터를 축적할 수 있다는 장점이 있다. 하지만, 특정 에이전트 기반의 강화학습 시뮬레이션 이외의 범용 시뮬레이션을 구성하고 실행하는 경우에는

확장이 어려운 문제가 있다. 또한, 범용 모의실험을 위하여 필요로 하는 실험관리에 대한 기능 부재로 인하여 연구자와 제품개발자가 활용하기에는 제약사항이 있다. 본 연구는 실험관리를 위하여 모의실험을 수행하기 전에 외부에 실험환경을 구축하는 단계와 시뮬레이터와 시나리오를 복사하는 단계와 실험과정을 모니터링하는 단계를 비롯하여 최종적으로 시뮬레이션 결과를 수집하는 제어기를 DEVS 형식론을 활용하여 관리한다.

2.3. 가상머신을 활용한 시뮬레이션 관리

모의실험을 수행하는 과정에서 서로 다른 실행환경에 대한 지원을 위하여 가상환경에서 시뮬레이션을 관리하는 프레임워크에 대한 연구가 제안된 바 있다[10]. 제안된 프레임워크는 실시간 시뮬레이션 어플리케이션을 사용하는 과정에서 시나리오 변수 값을 동적으로 변화시킬 수 있는 환경으로 모의 대상시스템 내부의 각 구성요소 간의 종속 관계를 약화시킨 가상머신 기반의 아키텍처 구조를 제시하였고 사용자가 실시간으로 실행되고 있는 시뮬레이터에 시나리오 변수 값을 변경하고 이를 실행할 수 있는 시뮬레이션 구조를 설계하였다. 특히 사용자가 시뮬레이터나 시뮬레이션 모델의 내부를 수정하지 않고 각 구성요소가 요구하고 반환하는 입/출력 인터페이스를 정의하고 해당 인터페이스에 부합되는 조합으로 시뮬레이션을 재구성할 수 있다는 장점이 있다. 하지만, 범용 모의실험을 적용하기 위하여 해당 구조를 도입해야 한다는 제약이 있으며 가상머신을 사용하여 시스템을 구성하는 경우 가상의 컴퓨팅 환경 위에 운영체제를 설치하고 해당 시뮬레이션 소프트웨어가 실행되기 때문에 컨테이너 기반의 가상화 기술에 비하여 성능이 저하되어 모의실험을 운영하고자 하는 사용자에게 있어 최대의 성능을 제공하지 못한다는 문제가 있다.

Ⅲ. 마이크로서비스 아키텍처 기반 모의 실험환경

경량형 컨테이너를 활용한 가상화 기술 기반의 마이크로 서비스 아키텍처는 사용자에게 서비스를 제공하기 위하여 단일 인스턴스로 서비스를 구현하는 것이 아니라 복수 개의 프로세스로 구성요소를 정의하고 해당 구성요소가 정의된 어플리케이션 프로그래밍 인터페이스(API: Application Programming Interface)를 활용하여 서비스를 구축하는 방법이다. 이때 각 프로세스가 독립된 실행 공간인 컨테이너 내에서 동작하고 해당 프로세스 간에 API를 호출하여 데이터를 주고받으며 서비스를 제공하는 형태로 서비스를 구성한다. 본 논문에서는 실험관리를 위하여 외부에서 컨테이너를 생성하고 관리하는 관리 프로세스로 DEVS 형식론 기반의 실험 제어기와 독립적으로 실행되는 모의실험 컨테이너로 구성된 마이크로서비스 아키텍처 기반의 모의실험 환경을 제안한다.

3.1. DEVS 형식론 기반 모의실험 제어기 및 관측기

DEVS형식론은 이산적인 사건을 처리하고 발생시키는 시스템을 정형적으로 표현하기 위한 방법으로 외부 입력 사건과 시간의 경과에 따른 시스템의 동특성을 기술하기 위한 집합 론 기반의 수학적 형식론이다. 본 연구에서는 모의실험을 관리하는 제어기와 모의실험 대상 시스템 간의 관계를 이산사건시스템으로 정의하고 이를 관리하는 제어기를 DEVS 형식론으로 정의한다. DEVS 형식론은 시스템의 동특성을 원자모델의 형태로 관리한다. 다음 그림 1은 DEVS 형식론의 원자모델을 나타낸다.

원자 모델은 3개의 집합(X, Y, S)과 4개의 함수(δ_{ext}, δ_{int}, λ, ta) 로 구성되어 있으며 시스템의 상태를 유지하면서 외부 입력 사건이 유입되었을 때 상태가 어떻게 천이되는 지와 특정 상태가 일정시간 동안 유지되었을 때 외부로 어떤 출력 사건을 발생시킬 지와 출력 사건을 발생시키고 난 이후에 시스템의 상태를 어떻게 변경시킬 것인지에 대한 규칙을 기술한다. 다음 그림 2는 컨테이너 외부에서 모의실험을 관리하는 개념도를 나타낸 것이다.

$$AM = < X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta >$$

X : Input Events Set

Y : Output Events Set

S : States Set

$\delta_{ext} : Q \times X \rightarrow S$; External Transition Function

$Q = \{(s,e) \mid s \in S,\ 0 \le e \le ta(s)\}$; Total state of AM,

e : elapsed time

$\delta_{int} : S \rightarrow S$; Internal Transition Function

$\lambda : S \rightarrow Y$; Output Function

$ta : S \rightarrow R_{\infty}^{+}$; Time Advance Function

Fig. 1. Specification of Atomic Model.
그림 1. 원자모델의 명세

그림 2는 컨테이너를 생성하는 제어기와 시뮬레이터가 독립적으로 실행되는 컨테이너 환경, 그리고 해당 컨테이너 내의 시뮬레이터를 관측하여 컨테이너의 소멸을 관리하는 관측기로 구성되어 있다. 제어기는 Start 상태에서 무한히 대기 중에 있다가 외부에서 start 이벤트를 받게 되면 각 컨테이너를 생성하고 t_1 주기로 관리 메시지

로 net 포트를 사용하여 메시지를 전송한다. 이후 모든 컨테이너의 관리가 종료되면 Start상태로 천이하여 대기한다. 관측기는 대기상태에 무한히 있다 외부에서 start 이벤트를 받게 되었을 시 컨테이너를 t_2 주기로 관측을 수행하여 시뮬레이션이 종료되어 결과를 생성하는지 모니터링한다. 이후 모든 시뮬레이션이 종료되어 결과를 획득하면 외부로 done 이벤트를 발생시키며 대기 상태로 천이한다.

3.2. 모의실험 지원을 위한 경량형 컨테이너

모의실험 지원을 위한 경량형 컨테이너는 운영체제와 CPU와 같이 시뮬레이터의 실행에 대한 제약사항을 벗어나도록 독립된 컴퓨팅 환경을 제공하고 제어기와 관측기로 동작한다.

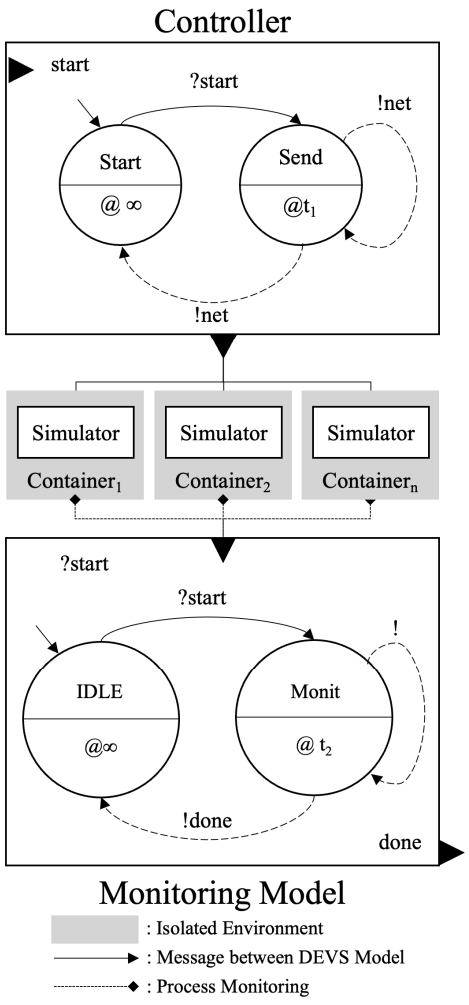

Fig. 2. Lightweight Experimental Frame based on the Microservice Architecture.
그림 2. 마이크로서비스 아키텍처 기반 경량형 모의실험틀

그림 3은 단일 컴퓨팅 환경에서의 개념도로 경량형 컨테이너는 하나의 독립된 컨테이너로 동작하여 하나의 컨테이너가 단일 시뮬레이션 실험을 담당한다.

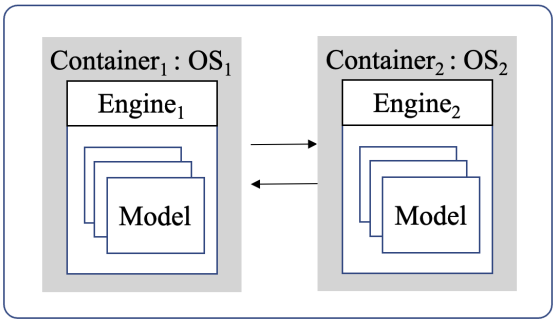

Fig. 3. Proposed Standalone Simulation Environments.
그림 3. 제안하는 단일 시뮬레이션 환경

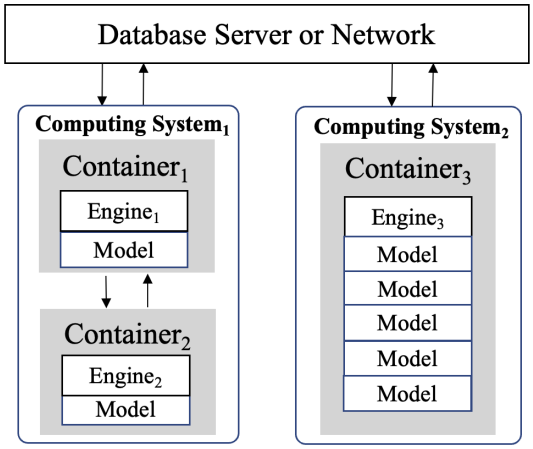

Fig. 4. Proposed Multi-Simulation Environments.
그림 4. 제안하는 다중 시뮬레이션 환경

그림 4는 물리적 혹은 가상의 네트워크로 연결되어 구성된 다중 컴퓨팅 환경에서의 시뮬레이션 실험환경으로 기존 마이크로서비스 아키텍처 기반 서비스와 같이 컨테이너 사이에 정의된 API를 각 컨테이너가 사용하여 상호작용하는 형태로 실험환경을 구성한다. 이때, 사용자가 복잡한 시스템 설정에 대한 지식이 없이 실험을 구축할 수 있도록 외부의 모의실험 제어기로 컨테이너가 시나리오와 환경에 대한 설정값을 JSON(JavaScript Object Notation), XML(eXtendable Markup Lanaguage)와 같은 형식을 이용하여 전달받으며 이를 통하여 사용자가 설정값을 쉽게 변경하거나 수정할 수 있도록 설계되었다. 제어기는 사용자 설정값에 따라 모의실험 컨테이너를 실행하며 컨테이너는 생성과 동시에 컨트롤러부터 데이터를 전달받아 시뮬레이션을 수행한다. 모의실험 컨테이너는 컨테이너 내부에 실험 대상 시뮬레이터에 맞는 운영체제, 라이브러리, 애플리케이션의 환경을 구성하며 사용자와의 상호작용을 최소화하여 사용자 오류 발생 확률

을 줄이고 실험 진행 및 관리의 안정성을 높일 수 있다.

또한, 모의실험 컨테이너는 모의실험 관측기에 시뮬레이션의 실행 상태를 실시간으로 제공하며, 모의실험 관측기에서 시뮬레이터의 실행상태를 관측하면서 시뮬레이션 상태 변화이벤트에 따라 실험을 관리한다.

3.3. 모의실험환경을 활용한 실험방법론

연구자 또는 제품 개발자가 모의 실험을 수행하기 위해서는 실험 목적을 설정하고 해당 실험목적에 맞는 시뮬레이션 환경을 컨테이너로 구성될 수 있도록 설정하는 것이 필요하다. 그림 5는 제안하는 모의실험환경에서 컨테이너를 구성하여 실험을 수행하기 위한 과정을 도식화한 것으로 모의실험 컨테이너는 기존에 구축된 컨테이너를 재사용하거나 새로운 컨테이너를 구축할 시에 고려해야 할 사항과 절차를 표현한 것이다.

먼저 사용자는 실험 목적에 맞는 컨테이너가 존재하는지를 확인한 후 모든 컨테이너가 존재한다면 각 실험을 위한 컨테이너를 생성하여 실험을 수행한다. 만일 사용자가 실험 목적에 부합하는 컨테이너를 찾을 수 없는 경우에는 새로운 컨테이너를 구성하기 위하여 실험을 위한 시뮬레이션 환경에 맞는 컨테이너를 생성하는 과정을 거친다. 해당 컨테이너를 생성하기 위해서는 먼저 컨테이너에서 실행될 운영체제 종류를 선택하고 해당 컨테이너에서 실행될 시뮬레이터와 해당 시뮬레이터를 실행하기 위하여 필요한 환경을 설정하여 컨테이너가 생성될 수 있도록 한다.

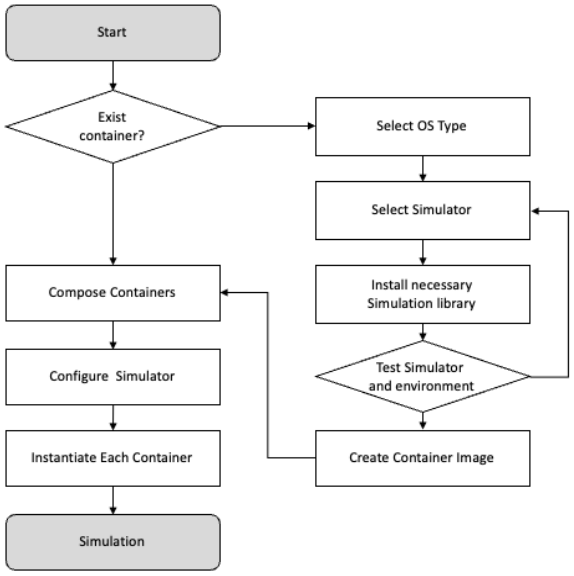

Fig. 5. Flowchart of Implementation of Experiment Container.

그림 5. 모의실험 컨테이너 구축 흐름도

```
$ app.py
Container name : Container_1
[Rx] Simulation Parms :
{'parms' : {'weight' : 75, 'height' : 180, 'disease' :
0, 'pose' : 'standing', 'health' : 100, 'heart rate' :
124, 'site id' : 'vision1'}}
[Sim] Start
[Sim] Done
[Tx] Simulation Results : {'id' : 'container_1',
'result' = 'safe'}
$
```

Fig. 6. Execution screen according to the mock-up experiment container construction flowchart (Container).

그림 6. 모의실험 컨테이너 구축 흐름도에 따른 실행 화면(컨테이너)

다음 그림 6는 시스템이 동작함에 따라 새로운 컨테이너가 생성되고, 이후 컨테이너 내부의 프로그램이 작동되며 반환된 내부 실행 결과다. 그림 6의 실선 부분은 시뮬레이션 제어기로부터 시뮬레이션에 필요한 파라미터를 수신한 로그로 해당 파라미터를 바탕으로 시뮬레이션을 진행한다(Rx). 그림 6의 점선 부분은 시뮬레이션의 결과값으로 해당 결과값을 시뮬레이션 모니터에게 전달한다(Tx).

Ⅳ. 사례 연구

본 연구에서는 제안하는 마이크로 서비스 아키텍처 기반의 시뮬레이션 환경을 구축하기 위하여 간단한 DEVS 시뮬레이션을 활용하였다. 사례 연구를 위해 구축을 위해 본 연구에서는 시뮬레이션 인스턴스 제어기, 시뮬레이션 인스턴스 실행기, 시뮬레이션 인스턴스 관측기를 도커 기반으로 구성하여 실험을 진행하였다. 사례연구로 진행한 간단한 시뮬레이션은 신체정보를 활용하여 근로자의 칼로리 소모량을 기반으로 한 신체적 이상 징후 탐지에 관한 시뮬레이션을 진행하였다.

실험방법은 근로자의 정보를 데이터베이스(Database)에 사전 저장한 후 근로자의 움직임이 감지되면 시뮬레이션을 진행하도록 하였다. 근로자의 움직임 관측을 프로세스는 실행기에서 실행되며 관측한 근로자의 정보를 이용하여 시뮬레이션 인스턴스를 실행한다. 이때, 인스턴스 생성 개수는 사용자의 목적에 따라 사전 정의한 설정파일(txt, yaml, py)을 수정하여 변경할 수 있다. 시뮬레이션 인스턴스가 생성되면 시뮬레이션 인스턴스 관측기가 실행되어 시뮬레이션이 정상 작동 되었는지 와 시뮬

레이션 종료여부를 관측한다. 생성된 시뮬레이션의 인스턴스가 모든 실행을 마치게 되면 시뮬레이션 인스턴스 제어기가 생성된 도커 컨테이너를 모두 삭제하여 컴퓨팅 자원의 관리를 돕는다.

본 실험은 2.3 GHz 8코어 Intel Core i9, 64GB 메모리의 Mac환경에서 진행하였다. 도커 환경을 실험하기 위해서 본 실험에서는 도커 인스턴스를 5개, 10개, 20개로 설정하여 실험하였으며 환경의 비교를 위하여 가상머신으로는 Oracle Virtual Box를 사용하였다. 그러나, 가상머신은 컴퓨팅 자원을 많이 소모하여 도커환경과 다르게 5개, 10개, 20개의 인스턴스가 아닌 1개, 2개. 3개의 인스턴를 생성하여 진행하였다. 실험은 인스턴스 개수별 컨테이너 생성, 소멸, 시뮬레이션 시간과 CPU와 메모리의 할당량을 측정하였다. 다음 그림 7와 8은 실험

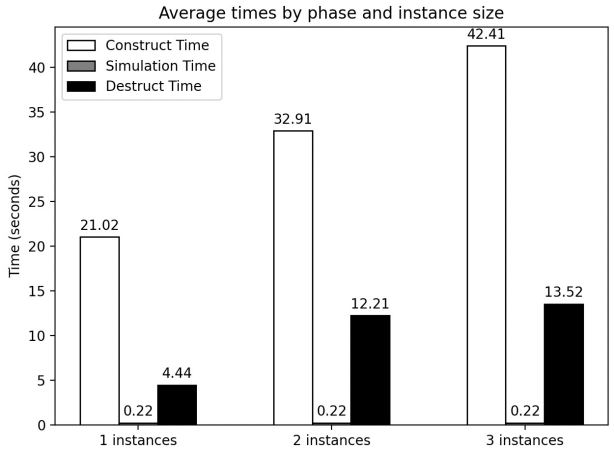

Fig. 7. Graph of number of instance's construction, simulation and destruct time(Virtual Machine).
그림 7. 인스턴스별 생성, 시뮬레이션, 소멸 시간 그래프(가상머신)

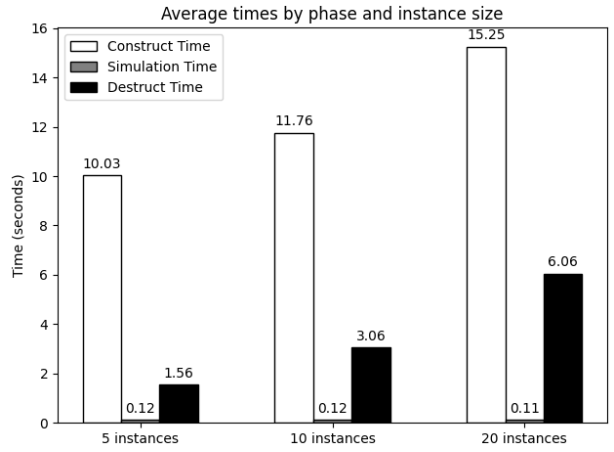

Fig. 8. Graph of number of instance's construction, simulation and destruct time(Docker).
그림 8. 인스턴스별 생성, 시뮬레이션, 소멸 시간 그래프(도커)

을 통해 확인한 각각의 환경에서의 시뮬레이션 인스턴스 생성 및 소멸, 시뮬레이션 시간에 대한 결과 그래프이다.

그림 7과 8의 하얀 바는 전체 인스턴스의 총 생성시간을 내타내며 회색 바는 전체 시뮬레이션 시간을 나타내고, 검은 바는 전체 인스턴스의 총 소멸 시간을 나타낸다.

다음 표 1은 각 환경에서 생성된 인스턴스별 컴퓨팅 자원의 할당량을 나타낸 표이다. 도커 컨테이너는 1개, 5개, 15개, 20개를 측정하였으나 실험 환경에서 컴퓨팅 자원의 한계로 인하여 가상머신의 정밀 측정은 3개까지 측정하였다.

Table 1. Comparison of CPU and Memort usage betwwen Docker and Virtual Machines by number of instances.
표 1. 인스턴스 개수별 도커, 가상머신 간 CPU, 메모리 사용량 비교

구분	Docker		Virtual Machine	
인스턴스 개수	CPU (%)	Memory (mb)	CPU (%)	Memory (mb)
1	0.35	21	23	4096
2	0.71	42	57	8192
3	1.08	63	82	12288
5	1.80	106	-	-
10	3.63	212	-	-
20	7.31	424	-	-

*) 실험환경에서 가상 머신 3개 이상으로 실행 시 정밀 측정 불가

표 1에 따라 도커 컨테이너는 유동적인 자원 할당이 가능하여 가상머신과 비교하여 CPU사용량과 Memory 사용량이 현저하게 적음을 알 수 있다. 그림 7과 8, 표 1을 통해 전체적으로 도커환경을 사용함이 가상머신을 사용하는 것 보다 시간적, 자원적 이득임을 알 수 있다. 또한, 각 환경에서 인스턴스별 생성시간과 소멸시간은 선형적으로 증가하고 CPU와 메모리 사용량 또한 인스턴스가 증가함에 따라 자원 사용이 선형적으로 높아짐을 알 수 있다. 그러나, 인스턴스의 개수증가에 따른 시뮬레이션 시간은 큰 변동이 없어 다중 시뮬레이션을 효과적으로 할 수 있음을 알 수 있다.

사례연구를 통해 병렬 다중 인스턴스를 이용하여 시뮬레이션을 진행함에 있어 본 연구에서 제안하는 프레임워크를 활용하여 시뮬레이션 횟수별 높은 시간적 효율과 효과적으로 컴퓨팅 자원을 관리할 수 있음을 보여준다. 이러한 효과는 사용자로 하여금 다중 시뮬레이션을 진행함에 있어 컴퓨팅 자원을 효과적으로 사용할 수 있다.

V. 결론

본 연구는 마이크로서비스 아키텍처를 구성할 때 잘 알려져 있는 도커 컨테이너 기반의 모의실험 환경을 제안하였다. 제안하는 환경은 DEVS 형식론 기반의 제어기와 관측기, 독립된 컴퓨팅 자원이 할당된 시뮬레이터 컨테이너로 구성되어 있으며, 이들은 각각 설정값 전달, 결과 관리 및 실시간 모니터링 및 시뮬레이션 실행의 역할을 한다. 제안하는 컨테이너 기반의 접근법은 시뮬레이션 간의 상호작용을 최소화하고, 오류 발생 확률을 줄이며, 다양한 운영체제와 라이브러리 환경에서의 이식성을 보장할 수 있다. 또한 비교적 낮은 설정 진입장벽으로 인하여 사용자는 복잡한 설정 없이도 효과적으로 모의 실험을 분산된 환경에서도 실행할 수 있으며, 실시간 모니터링은 문제가 발생했을 때 빠른 대응이 가능하다. 본 논문에서는 사례 연구를 통하여 제안한 환경이 높은 확장성과 재사용성을 가짐을 확인하였으며 컨테이너 기반의 모의 실험환경을 바탕으로 연구자와 제품 개발자가 컴퓨팅 자원을 효율적으로 사용하여 실험을 수행할 수 있음을 보여준다.

References

[1] T. G. Kim, "M&S Engineering," *Korea Information Processing Society Review*, vol.14, pp.3-17, 2007.
[2] B. Kim, and T. G. Kim. "Multifaceted Modeling Methodology for System of Systems using IEEE 1516 HLA/RTI," *Journal of Korea Society for Simulation*, vol.26, no.2, pp.19-29, 2017.
DOI: 10.9709/JKSS.2017.26.2.019
[3] C. Choi, K-M. Seo, T. G. Kim, "DEXSim: an experimental environment for distributed execution of replicated simulators using a concept of single simulation multiple scenarios". *SIMULATION.* vol. 90, no.4, pp.355-376, 2014.
DOI: 10.1177/0037549713520251
[4] W. Byeon, H. Lim, J. Yun. "Performance Analysis of Docker Container Migration Using Secure Copy in Mobile Edge Computing," *Journal of the Korea Institute of Information Security & Cryptology*, vol.31, no.5, pp.901-909, 2021.
DOI: 10.13089/JKIISC.2021.31.5.901
[5] K. V. Knyazkov, S. V. Kovalchuk, "Modeling and Simulation Framework for Development of Interactive Virtual Environments," *Procedia Computer Science*, vol.29, pp.332-342, 2014.
DOI: 10.1016/j.procs.2014.05.030
[6] Bagrodia, R., Meyer, R., Takai, M., Chen, Y. A., Zeng, X., Martin, J., & Song, H. Y. "Parsec: A parallel simulation environment for complex systems," *Computer*, vol.31, no.10, pp.77-85, 1998.
DOI: 10.1109/2.722293
[7] Fujimoto, R. M. "Parallel and distributed simulation systems," *2001 Winter Simulation Conference* Vol. 1, pp.147-157. IEEE. 2001.
DOI: 10.1109/WSC.2001.977259
[8] J. S. Ahn, & Cho T. H. Cho. "A Simulation Execution Time Reduction Method through Parallel Processing of Model Execution in DEVS based IEEE 1516 HLA/RTI," *Korean Institute of communications and Information Sciences*, pp.1220-1221, 2021.
[9] Docker, Docker: Accelerated Container Application Development, https://www.docker.com/
[10] L. Zhang, B. P. Zeigler, Y. Laili, *Model Engineering for Simulation*, Academic Press, 2019.

BIOGRAPHY

Gyu-SiK Ham (Member)

2023 : BS degree in Computer Engineering, Hanbat University.
2023~present : MS course in Computer Engineering, Hanbat National University

Hyeon-Gi Kim (Member)

2024 : BS degree in Computer Engineering, Hanbat University.
2024~present : MS course in Computer Engineering, Hanbat National University

Jin-Woo Kim (Member)

2024 : BS degree in Computer Engineering, Hanbat University.
2024~present : MS course in Computer Engineering, Hanbat National University

Soo-Young Jang (Member)

2006 : BS degree in Industrial Engineering, Korea Advanced Institute of Science and Technology (KAIST).
2008 : MS degree in Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST).
2014 : PhD degree in Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology(KAIST).
2014~2017 : Senior Researcher, Samsung Electronics.
2017~2023 : Senior Researcher, Electronics and Telecommunications Research Institute(ETRI).
2023~present : Professor, Dept, of Computer Engineering, Hanbat National University

Eun-Kyung Kim (Member)

2003 : BS degree in Computer Engineering, Yonsei University.
2005 : MS degree in Electrical Engineering, Pohang University and Science and Technology(POSTECH).
2019 : PhD degree in Electronic and Electrical Engineering, Korea Advanced Institute of Science and Technology(KAIST).
2005~2021 : Principal Researcher/Senior Researcher/ Researcher, Electronics and Telecommunications Research Institute(ETRI).
2021~present : Professor, Dept, of Artificial Intelligence Software, Hanbat National University

Chang-Beom Choi (Member)

2005 : BS degree in Computerl Engineering, Kyunghee University.
2007 : MS degree in Computer Science, Korea Advanced Institute of Science and Technology(KAIST).
2014 : PhD degree in Electronic and Electrical Engineering, Korea Advanced Institute of Science and Technology(KAIST).
2014~2021 : Professor. Dept, of Global Entrepreneurship and Information Communication Technology, Handong University
2021~present : Professor, Dept, of Computer Engineering, Hanbat National University