JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 12, pp. 121-129, December 2022
https://doi.org/10.9708/jksci.2022.27.12.121

A study of submarine combat management system docker-based server virtualization design and performance analysis

Sang-Gil Son*

*Engineer, Naval R&D Center, Hanwha Systems, Gumi, Korea

[Abstract]

the Naval Combat Management System(CMS) has been installed and used in various ships since its localization, and has been developed by continuously introducing the latest technology. Recently, surface ship CMS have applied server virtualization and desktop virtualization(Virtual Desktop Infra, VDI) technologies among virtualization technologies to increase system stability and limitations on the limited space and weight of ships. On the other hand, submarine CMS do not have virtualization technology applied, so there are limitations in space and weight limitations and CMS efficiency improvement. To this end, this paper proposes a next-generation submarine CMS using Docker-based server virtualization. Through performance analysis between the processor of the existing CMS and the processor to which Docker-based server virtualization was applied, it was confirmed that the method proposed in this paper is applicable to the next-generation submarine CMS.

▸ **Key words**: Submarine Combat Management System, Virtualization, Performance analysis, Docker, Efficiency

[요 약]

함정전투체계는 국산화 이후 다양한 함정에 탑재되어 사용되고 있으며 최신 기술을 지속적으로 도입하며 발전을 이루어왔다. 최근 수상함 전투체계는 함정의 한정된 공간과 중량에 대한 제약사항 그리고 체계 안정성을 높이기 위해 가상화 분야 기술 중 서버 가상화와 데스크탑 가상화(Virtual Desktop Infra, VDI) 기술을 적용하였다. 반면, 잠수함 전투체계는 가상화 기술이 적용되지 않아 공간과 중량 제약 그리고 전투체계 효율성 향상에 한계가 있다. 이를 위해 본 논문에서는 도커 기반의 서버 가상화를 적용한 차세대 잠수함 전투체계를 제안한다. 기존 전투체계의 처리장치와 도커 기반 서버 가상화를 적용한 처리장치간 성능분석을 통하여 본 논문에서 제안하는 방법이 차세대 잠수함 전투체계에 적용 가능함을 확인하였다.

▸ **주제어**: 잠수함 전투체계, 가상화, 성능 분석, 도커, 효율성

• First Author: Sang-Gil Son, Corresponding Author: Sang-Gil Son
*Sang-Gil Son (sanggil12.son@hanwha.com), Naval R&D Center, Hanwha Systems
• Received: 2022. 10. 28, Revised: 2022. 12. 02, Accepted: 2022. 12. 14.

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

I. Introduction

함정 전투체계(Naval Combat Management System)란 함정에 탑재된 모든 탐지 장비, 무장, 항해 지원 장비 등을 네트워크로 연결하여 통합된 전술 상황 정보를 만들어 공유하고 표적의 탐지, 추적에서부터 위협분석, 무장할당, 교전 및 명중 여부 평가 분석에 이르기까지 지휘 및 무장통제를 자동화함으로써 위협에 대한 전투 효과를 극대화시키기 위한 통합체계로 지휘통제, 무장통제, 전술 자료 교환 및 전시 등을 수행하는 체계이다[1].

초기 함정 전투체계는 기능적인 요구사항을 만족시키는 것에 중점을 두고 설계 및 개발이 되었으나 최근 함정전투체계는 기존의 요구사항을 포함하여 재사용성, 제한적 공간 등을 고려한 설계가 필요하다. 이를 해결하기 위하여 클라우드 컴퓨팅 기술의 적용이 주목되고 있으며 그 중 가상화 기술을 적용을 하고 있다.

수상함 전투체계에서는 서버 가상화(Server Virtualization)를 통하여 유지 보수 및 확장성을 높였으며, 데스크탑 가상화(Virtual Desktop Infra, VDI) 기술을 적용하여 GUI 응용SW의 가상화를 적용하여 사용자가 데스크톱 구성 및 설정을 자신에게 맞게 바꾸어 사용할 수 있도록 하였다[2].

수상함에 비해 비교적 최근에 국산화 개발이 완료된 잠수함 전투체계는 아직 가상화 기술이 적용되어 있지 않아 최근 활발히 연구되고 있는 추세이다. 특히 잠수함의 경우 공간적인 제약 및 체계 중량 감소, 전투체계의 효율적인 운용 등이 중요한 요소이며 이러한 요소를 개선하기 위해 Vmware 기반의 랙 서버 타입과 VPX 서버타입의 적용성을 연구가 진행되었다[3]. Vmware와 같은 가상화 솔루션을 사용함으로써 차세대 잠수함 전투체계에 가상화 기술 적용을 검토하고 있으나 도커를 이용한 잠수함 전투체계의 가상화 기술 적용은 검토되지 않았다.

본 논문에서는 잠수함 전투체계에 적합한 도커 기반의 서버 가상화 설계 방법을 제시하였다. 또한, 도커 기반의 서버 가상화 성능분석을 통하여 제안한 방법에 대한 적용 가능성을 입증하였다. 이를 통해 기존 잠수함 전투체계보다 높은 확장성과 경량화를 가진 설계를 할 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 함정 전투체계의 구성, 가상화 그리고 도커에 대한 설명을 하였고, 3절에서는 잠수함 전투체계의 도커 기반의 서버 가상화 설계 적용 방안에 대하여 설명하였다. 4절에서는 기존 잠수함 전투체계의 환경과 가상화 설계가 적용된 환경에서의 성능 분석을 통하여 제안한 방법에 대한 적용 가능성을 입증하였다. 마지막 5절에서는 결론 및 추후 연구과제로 이 논문을 마무리하였다.

II. Preliminaries

1. Naval Combat Management System

함정 전투체계는 함정에서 인간의 두뇌와 같은 역할을 수행하는 핵심 장비이다[4]. 레이더, 전자광학체계 등 함정에 탑재된 센서 장비를 통해 표적을 분석하고 함포 등의 함정에 탑재된 무장체계에 교전 명령을 내림으로써 대함, 대잠전에서 최적의 공격 및 방어 수단을 실시간으로 제공한다. 잠수함 전투체계는 Fig. 1의 함정 전투체계와 같이 공통적으로 센서체계, 항해체계, 무장체계 그리고 통신체계로 구성되며, 실시간으로 처리된 전술정보를 관리하고 전술상황 감시, 센서통제, 무장할당 및 발사통제 등을 포함한 각종 전술대응을 용이하도록 지원한다.[5,6]

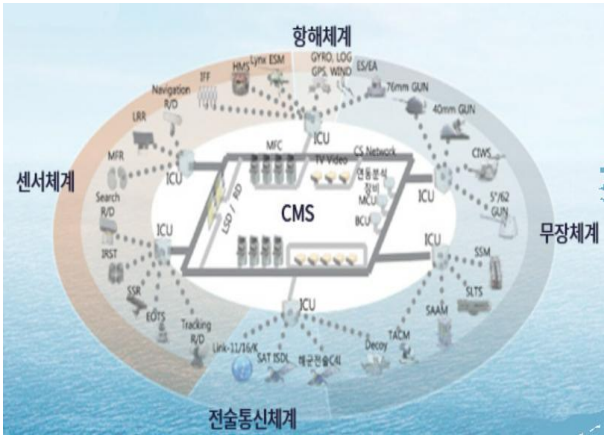

Fig. 1. Naval Combat Management System[7]

2. International trends

미 해군은 다양한 구성 요소에 가상화 기술을 채택하였으며 대표적으로 CANES(Consolidated Afloat Networks and Enterprise Services) 가상화 솔루션을 적용하였다[8].

개발이 진행함에 따라 모듈식 설계, 개방형 표준으로 새로운 이지스 소프트웨어 아키텍쳐를 구현하는데 진전이 있었지만 빠르게 변화하는 위협 환경에 하드웨어 및 소프트웨어의 업그레이드는 신속하게 진행되지 못하였다. 전투 시스템 컴퓨팅 인프라는 개별 용량의 일부에서만 작동하여 자원 활용에 대한 비효율성과 수명주기 간 유지비용을 증가시킨다. AEGIS VT(Virtual Twin)를 개발하여 요구되는 공간 감소, 하드웨어에서 소프트웨어를 분리하여 유연성 및 적응성을 향상, 소프트웨어 업데이트 속도 및 빈도

를 미래 해군의 요구사항을 충족하였다[9]. 미 해군에서도 가상화 환경이 자원을 효율적으로 활용한다고 판단하였다.

3. Virtualization

가상화는 하나의 물리적인 서버에 가상머신을 활용하여 여러 가지의 운영체제와 응용을 실행할 수 있도록 하는 소프트웨어 기술이다. 각각의 가상머신은 독립적으로 운용되며 하나의 장치에 여러 운영체제를 실행시켜 자원 활용의 효율을 높일 수 있다[10].

가상화 분야는 크게 3가지로 분류할 수 있으며 서버 가상화, 데스크탑 가상화, 어플리케이션 가상화가 있다[11]. 서버 가상화는 Fig. 2와 같이 하나의 물리적인 하드웨어를 다수의 가상 하드웨어로 분리하여 가상머신 상에서 다수의 운영체제 및 어플리케이션을 독립된 환경에서 실행할 수 있다.

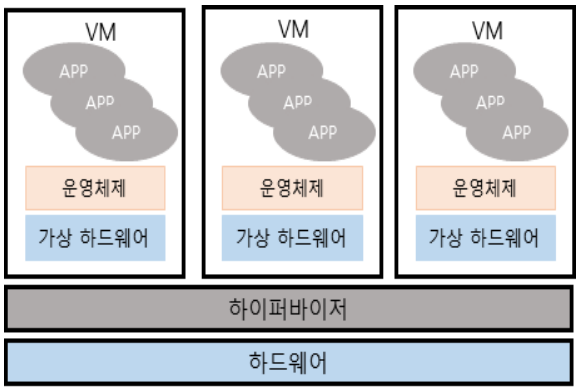

Fig. 2. Server Virtualization Architecture.

데스크탑 가상화는 VDI(Virtual Desktop Infra)이라고도 하며 서버 기반의 가상화 방법 중 하나로써, 가상화를 통한 서버 통합개념을 확장하여 클라이언트 데스크탑까지 중앙에서 관리하도록 하는 것이다[12]. VDI에서는 하이퍼바이저가 서버를 가상머신으로 세분화하고, 가상머신은 가상 데스크탑을 호스팅하며, 사용자는 각자의 기기를 통해 가상 데스크탑 원격에 접속을 한다[13]. VDI의 경우 각 데스크탑을 클라이언트 장치에 원격으로 서비스를 제공하여 사용자가 데스크탑 구성 및 설정을 자신에게 맞게 바꾸어 사용할 수 있도록 한다. Fig. 3과 같이 데스크탑 가상화의 구성 요소는 사용자가 중앙의 가상머신에 원격 접속하기 위한 클라이언트, 사용자 인증 등 정책을 적용하고 접속 요청에 따라 어떠한 가상머신을 사용자에게 전달할지에 대한 정보는 저장하는 세션 브로커 서버, 사용자 계정을 통합적으로 관리하고 인증을 담당하는 인증서버, 실제 가상머신이 실행되는 요소로 하이퍼바이저가 있다.

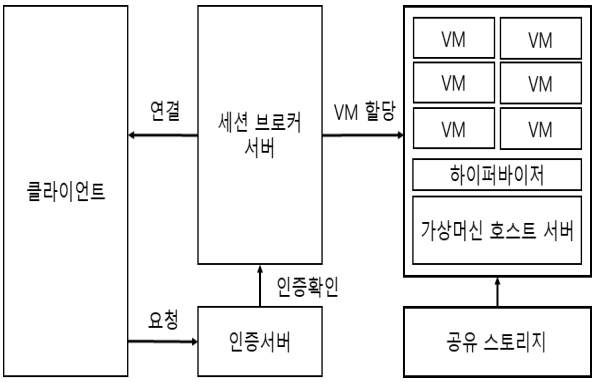

Fig. 3. VDI(Virtual Desktop Infra) Architecture

4. Docker

가상화 기술은 하나의 물리적인 하드웨어에 가상의 하드웨어를 분리하여 다수의 운영체제를 사용하도록 하며 게스트 OS의 자원은 호스트 OS에서 관리하게 된다. 그러나 하나의 운영체제 위에 또 다른 운영체제를 운용하기 때문에 성능적으로 제한이 있을 수 밖에 없다[14].

도커는 컨테이너 기반의 오픈소스 가상화 플랫폼이다. 다양한 어플리케이션과 실행환경을 컨테이너로 패키징하여 배포 및 관리가 가능하다. 도커에는 이미지와 컨테이너라는 중요한 요소가 있으며 도커 이미지는 서비스 운영에 필요한 프로그램, 실행 파일, 라이브러리, 실행 환경의 설정 정보 등을 묶은 템플릿이다. 컨테이너는 도커 이미지를 독립적인 공간에서 실행하는 환경이다. 도커 컨테이너가 격리된 공간으로 어떠한 내부 작업을 하여도 Host OS에 영향을 주지 않기 때문에 독립된 개발 환경을 보장한다. 그리고 도커 이미지는 커널을 포함하고 있지 않아 크기가 작으며 배포가 용이하다. 도커는 이미지만 있다면 컨테이너를 실행할 수 있기 때문에 확장성과 이식성이 좋다. 하드웨어 수준에서 가상화가 되는 가상머신과 달리 애플리케이션 계층에서 가상화가 된다. 하나의 머신 위에서 Host OS커널을 공유하기 때문에 가상머신에 비하여 리소스 사용이 적다[15,16]. Fig. 4와 같이 가상머신(Vitrual Machine, VM)과 도커 간 구조 차이가 있는 것을 확인할 수 있다. 도커에서는 Hypervisor와 Guest OS가 없으며 Docker Engine을 이용하며 컨테이너를 실행하여 가상화 환경을 제공하게 된다. 컨테이너 자체에는 OS가 설치되지 않기 때문에 가상머신에 비해 경량화된다.

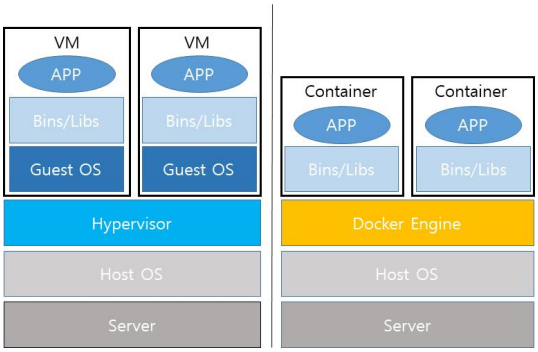

Fig. 4. Comparison of VM and Docker structures

5. Data Distribution Service(DDS)

DDS는 분산 네트워크에서 발간(Publish)/구독(Subscribe)을 기반으로 실시간 데이터를 배포하는 기능을 제공하는 통신프로토콜이다. 데이터 중심 미들웨어로 물리적인 위치에 관계없이 통신이 가능하며 Application 간의 독립성이 보장이 된다. 확장성이 보장되며 고장에 대한 영향성이 서버-클라이언트 구조에 비해 최소화된다. 실시간 시스템을 위한 다수의 통신품질제어(Qos) 기능을 제공하며 자동 경로 탐색 기능을 이용한 유연한 다자간 통신 능력을 제공한다. 데이터 통신을 하는 노드의 개수가 많고 데이터 흐름이 복잡한 경우 서버-클라이언트 통신 기술 기반의 시스템보다는 신뢰성, 확장성 등의 향상된 성능을 기대할 수 있다[17,18].

III. The Proposed Scheme

1. Virtualization based Submarine CMS architecture

본 논문에서는 기존 전투체계의 안정성을 유지하면서 전투체계의 중요한 제약사항인 공간 및 중량을 줄이기 위하여 도커 기반의 서버 가상화를 제안한다. 도커 기반의 서버 가상화는 경량화 된 가상머신처럼 사용이 가능하며 도커 컨테이너의 생성 및 관리가 용이하다. 도커 기반의 서버 가상화를 적용할 경우 연동장치 및 처리장치의 역할을 하던 SBC(Single Board Computer)를 대신하여 서버 PC에 연동장치 및 처리장치를 가상화하면 전투체계의 공간 및 중량을 줄일 수 있다. 또한 도커가 설치되어 있다면 운영체제와 같은 환경적인 요소에 관계없이 컨테이너를 실행할 수 있어 확장성과 이식성이 매우 좋아지며 도커는 격리된 공간에 필요한 파일만을 가지고 실행하기 때문에 비교적으로 서버의 리소스 사용이 적으며 속도도 향상되어 효율적인 전투체계 운용이 가능하다.

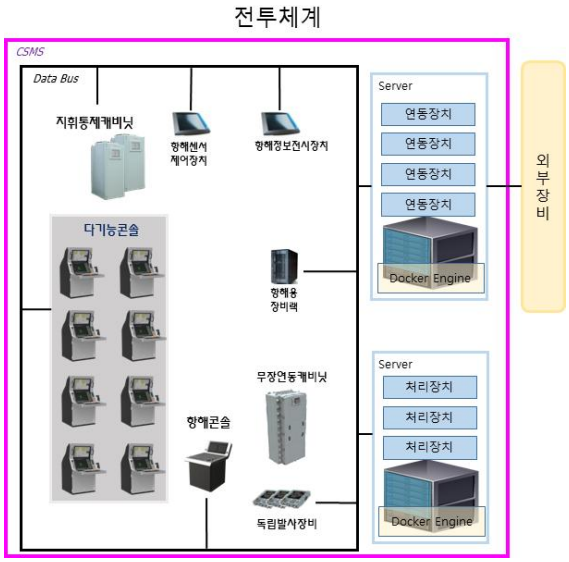

Fig. 5. Design CMS with Virtualization

Fig. 5와 같이 연동장치와 처리장치를 도커 기반의 서버 가상화를 적용한 잠수함 전투체계 환경을 설계하였다. 처리장치 및 연동장치의 가상화 설계를 하였으며 1대의 서버 PC에 다수의 처리장치 및 연동장치의 가상화를 통하여 운용하도록 한다. 이더넷 기반의 통신을 위해 기존의 시리얼 통신은 Serial To Ethernet Converter를 활용하여 가상화된 장치가 이더넷 통신을 하도록 한다. 그 외의 하드와이어 제어나 이산신호는 기존 구성을 유지하는 방안으로 제안한다. 잠수함 전투체계는 이더넷 기반의 통신을 하는 장치가 많아 다수의 연동장치와 처리장치를 가상화할 수 있다. 잠수함 전투체계에서 사용하는 연동장치 및 처리장치는 100대 이상의 SBC를 사용하고 있는데 일반적인 상황에서 10% 미만의 자원(CPU/메모리)을 사용하여 정보를 처리한다. 자원 활용성을 높이는 측면에서 가상화를 적용할 수 있으며 이를 통해 SBC의 개수를 줄일 수 있으므로 효율적인 공간 활용에도 이점을 갖는다.

잠수함 전투체계는 노드 간의 데이터 통신을 위하여 DDS를 사용하고 있기 때문에 노드의 변경이 발생하여도 SW수정 없이 실행이 가능하다. 이러한 점은 기존의 연동장치와 처리장치를 가상화로 설계 가능토록 한다. 연동장치나 처리장치의 추가 또는 삭제 등의 HW구조적인 변경이 발생할 경우 공간 확보, 구성 변경에 따른 노력과 시간, 비용이 발생하지만 Fig. 6과 같이 도커 기반의 서버 가상화를 적용한 경우 서버에서 가상화한 장치를 삭제 또는 추가함으로써 HW구조 변경이 없이 사용할 수 있어 높은 확장성을 보여준다.

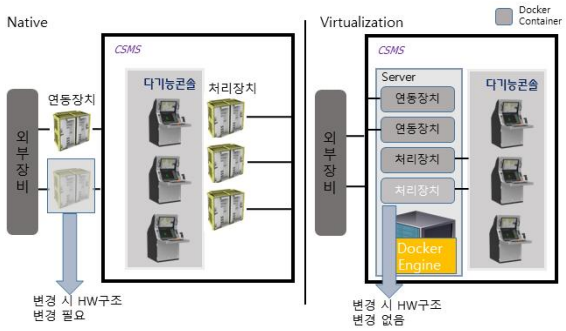

Fig. 6. Compare Extensibility Native and Virtualization

2. Redundancy of processing device

전투체계에서는 체계의 안정성을 높이기 위하여 주요장비의 이중화 또는 다중화를 적용하고 있다. 기존 전투체계는 이중화/다중화는 Active/Standby로 운용하며 Active의 오류 발생 시 Standby가 Active로 전환되면서 주요 기능의 연속성을 보장하게 된다. 이중화/다중화를 적용함에 있어 Active/Standby로 운용이 되며 이 외에도 유휴장비가 Standby로 전환되기 위하여 여유 상태로 운용되고 있다. 예를 들어 10대의 SBC 중 4대는 Atcive, 4대는 Standby로 운용중이며 남은 2대는 Standby로 전환을 대기한다. 정상적으로 운용되는 상태에서는 유휴장비는 어떠한 기능도 하지 않는다. 도커 기반 서버 가상화에서는 Fig. 7과 같이 이중화/다중화를 관리하는 별도로 관리하는 ReduncdancyManager SW를 두어 도커 컨테이너의 상태를 감시 및 실행하는 방식으로 유휴장비에 대한 효율적인 운용 방안을 제시한다. 유휴장비 없이 Acitve/Standby만 실행된 상태에서 Active 장비에 오류가 발생 시 Active장비는 종료가 되며 Standby장비는 Active로 전환이 된다. RedundancyManager SW는 Actvie장비의 종료를 확인 후 해당 SW 간의 Active장비의 종료 상태를 공유하여 리소스의 여유가 있는 Server에서 Active-Standby(Standby로 전환하기 위한 Active 컨테이너)로 도커 컨테이너를 실행한다. 이러한 방식으로 효율적으로 리소스를 활용하며 유휴장비 없이 이중화/다중화를 운용할 수 있다.

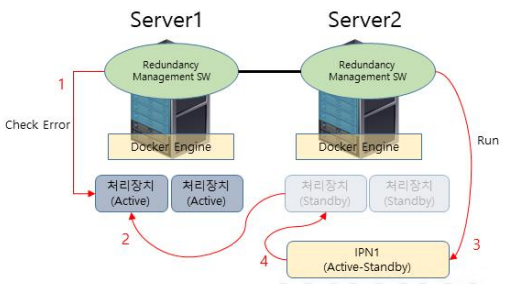

Fig. 7. Redundancy/multiplex flow chart

RedundancyManager SW를 구현하기 위하여 디자인 패턴을 적용하였다. 디자인 패턴은 GOF(Gang of Four)패턴이라고도 불려지며 소프트웨어 개발 과정에서 축적된 설계 노하우를 재사용하기 좋은 형태로 정리한 형태이며, 생성패턴, 구조패턴, 행위패턴으로 분류되어 진다[19]. RedundancyManager SW의 클래스 다이어그램은 Fig. 8과 같다. 디자인 패턴 중 상태 패턴(State Pattern)을 적용하여 메인함수 CMain 클래스, 도커 컨테이너의 상태를 송신하는 CSendDockerContainerStatus 클래스, 도커 컨테이너의 상태를 감시하는 CCheckDockerContainerStatus 클래스, 도커 컨테이너 상태에 따라 시작, 중지, 재시작을 처리하는 CDockerContainerStart, CDockerContainerStop, CDockerContainerRestart 클래스로 구성된다. 클래스 상세설명은 Table 1와 같다.

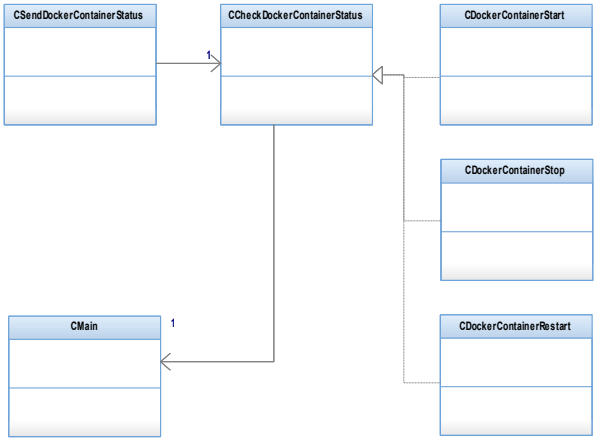

Fig. 8. RedundancyManager SW class diagram

Table 1. RedundancyManager SW class description

Class Name	Description
CMain	Object main class
CSendDockerContainerStatus	Periodically Send running docker container state to other server
CCheckDockerContainerStatus	Periodically check the status of a running docker container
CDockerContainerStart	Start docker container
CDockerContainerStop	Stop docker container
CDockerContainerRestart	Restart docker container

IV. Test and Analysis

1. Test Environment Configuration

본 논문에서는 제안하는 내용을 확인하기 위하여 Fig. 9와 같이 환경을 구성하여 성능분석을 하였다. Native환경은 도커 기반의 서버 가상화를 적용하지 않은 환경을 구성하였다. 전시처리장치 PC, 시뮬레이터 PC 그리고 처리장치 PC를 랜스위치를 통하여 연결되도록 하였다. Virtualization환경은 Server PC에 도커 기반의 서버 가상화를 적용하여 처리장치를 수행하도록 구성하였다. 처리장치 PC에서의 전투체계 SW실행과 도커 기반의 가상화에서의 전투체계 SW실행을 하였을 경우의 성능을 비교하여 도커 기반 서버 가상화를 적용하여도 전투체계 운용이 가능함을 확인하려고 한다.

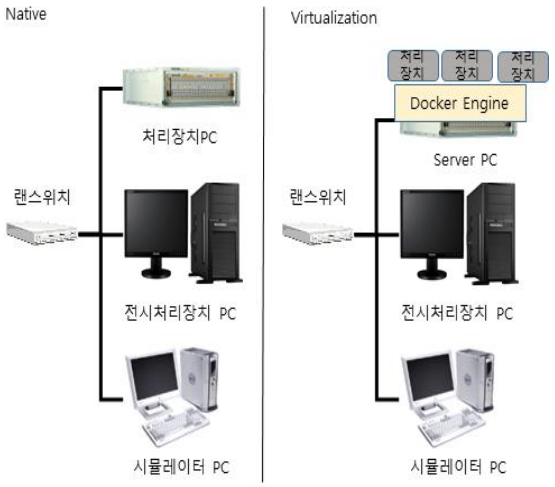

Fig. 9. Test Environment

전시처리장치 PC에는 전투체계 운용자화면을 전시하기 위한 전투체계 SW를 실행하도록 하였으며 시뮬레이터 PC에서는 전투체계 데이터를 모의하기 위한 시뮬레이터SW를 실행하였다. 처리장치PC와 가상화된 처리장치는 표적정보에 대한 처리, 장비 상태 감시, 장비 연동, 무장 운용 등의 기능을 수행하는 전투체계 SW를 실행하도록 하였다.

Native환경에 처리장치PC를 추가하여 Native환경과 Virtualization환경의 Active/Standby 전환하는 시간을 측정하고 RedundancyManager SW를 활용하여 Virtualization환경에서 유휴장비 없이 전투체계가 운용될 수 있음을 확인한다. 환경 구성 장비의 세부정보는 Table 2와 같다.

Table 2. Specification of Test Environment

	CPU	Mem	OS
전시처리장치 PC	Intel Core i7-4770 @3.40Ghz	16GB	Windows7
처리장치PC/ Server PC	Intel Xeon CPU E3-1280 3.60GHz	24GB	Ubuntu 20.04.4 LTS
시뮬레이터 PC	Intel Core i7-4770 @3.40Ghz	16GB	Windows7

2. Resource performance Test Result And Analysis

시험은 가상화를 적용하지 않은 처리장치PC에서 표적정보 처리, 장비 상태 감시, 장비 연동, 무장 운용을 각각 수행하여 CPU/Memory사용률을 측정하였고 도커 기반의 서버 가상화를 적용한 처리장치에는 도커 컨테이너 4개를 동시에 실행하여 각각의 기능을 수행하도록 하여 CPU/Memory사용률을 측정하였다. Test Case 1~4는 Native환경, Test Cast 5는 Virtualization환경에서 수행을 하였다. 성능시험을 위한 테스트 케이스의 상세정보는 Table 3와 같다.

Table 3. Test Case Description

No	Test Case	Description
1	Track Info 1000 in native	Generate 1000 Random Track Information
2	Device status monitoring in native	100 device status simulations (absent, available, unavailable)
3	Equipment interface status in native	10 Sensor status simulation and data interface
4	Weapon engagement in native	Weapon engagement and operation
5	Case 1~4 apply in virtualization	Execute Cases 1-4 in 4 docker containers

시험은 Test Case별로 CPU/Memory사용률을 측정하였다. CPU/Memory의 사용률을 확인하기 위해서 리눅스 OS의 리소스 정보를 전시해주는 top 명령어를 통하여 주기적으로 확인을 하였다. Fig. 10은 Case 1~4의 각 CPU사용률을 나타내며 0.5 ~ 5.12%로 나오는 것을 확인할 수 있다. 각 Test Case별로 운용되는 데이터 량이 다르기 때문에 CPU사용률이 다르게 나왔다. Fig. 11은 Test Case 1~4까지의 CPU사용률 합과 Server PC에 도커 컨테이너 4개를 동시에 실행하여 측정한 CPU사용률 합의 비교한 것으로 각각 8.52%, 11.3% 으로 가상화를 적용한 경우 2.8%정도 더 사용하는 것을 알 수 있다. Fig.12는 Test Case 1~4의 각 Memory사용률을 나타내며 3.28~3.6%로 나오는 것

을 확인할 수 있다. Test Case 1~4까지의 전체 사용량의 합과 가상화를 적용한 경우 각각 13.46%, 4.18%가 나왔으며 가상화를 적용한 경우 더 낮은 수치가 나오는 것을 확인할 수 있다. CPU사용률의 전체 합이 크게 나오는 이유는 각 장비별로 통신을 위해 실행되는 SW가 각각의 도커 컨테이너에서 실행이 되며 통신을 시작하게 되면서 CPU사용률이 크게 나온다고 할 수 있다. Memory사용률은 가상화를 적용한 경우가 적게 나오는 것을 확인할 수 있으나 해당 수치는 OS를 운영하는데 사용되는 Memory가 포함되어 있기 때문에 Total에는 4개의 OS Memory가 포함되어 있어 높게 나온다고 할 수 있다. 시험결과의 CPU/Memory사용률은 전체 용량에 비해 낮은 수치인 것을 확인할 수 있으며 Table 4와 같이 전투체계 성능요구사항에도 충분히 만족하는 것을 확인할 수 있다.

Table 4. CPU/Memory Requirements of CMS

	Requirements
CPU	50% or less
Memory	50% or less of the total amount

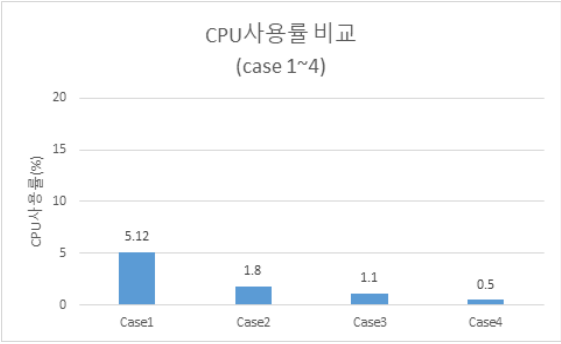

Fig. 10. Compare cpu utilization(case 1~4)

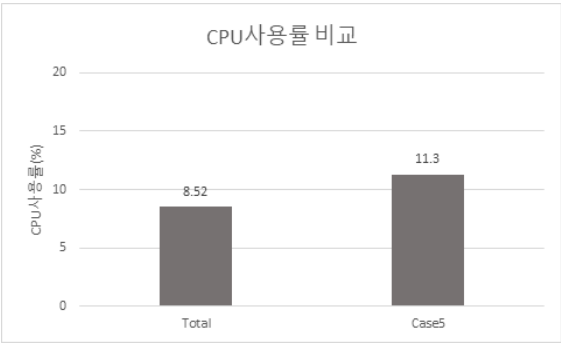

Fig. 11. Compare cpu utilization

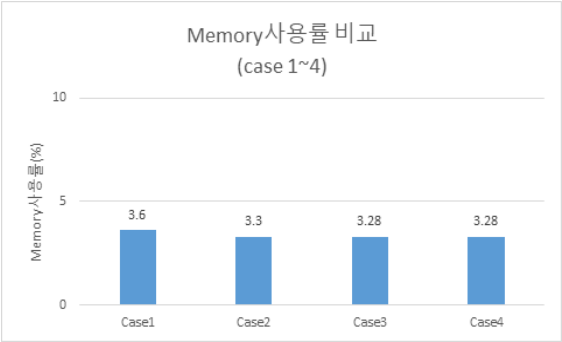

Fig. 12. Compare memory utilization(case 1~4)

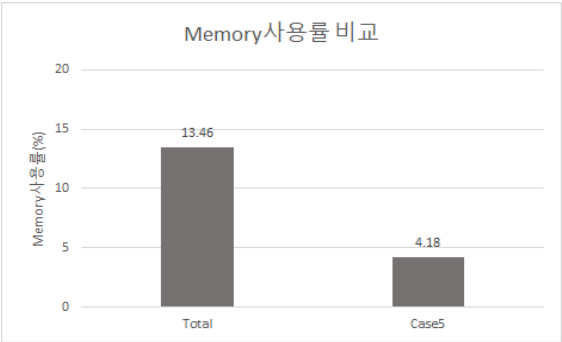

Fig. 13. Compare memory utilization

DDS 메시지 응답 시간을 측정하기 위하여 Topic의 크기가 1024 byte인 DDS 메시지를 1회 전송 기준으로 100개의 Topic을 반복적으로 전송하도록 하였다. 전시처리장치에서 송신, 처리장치에서 수신하여 DDS 메시지 응답 시간을 측정하였다.

Fig. 14 에서는 DDS 메시지의 응답 시간 측정 결과를 보여주고 있다. 그래프의 X축은 실험 횟수이며 총 10번을 반복하여 평균값을 계산하였다. 기존 전투체계 환경과 가상화 환경 간의 비교를 하였을 경우 거의 차이가 없음을 확인할 수가 있다. 가상화로 인한 DDS 메시지 지연 또는 누락이 발생하지 않음을 확인할 수 있다.

	1	2	3	4	5	6	7	8	9	10	평균
Native	861	872	785	758	1011	1037	937	1107	845	1003	922
Virtualization	700	964	669	980	1022	943	846	930	1153	867	907

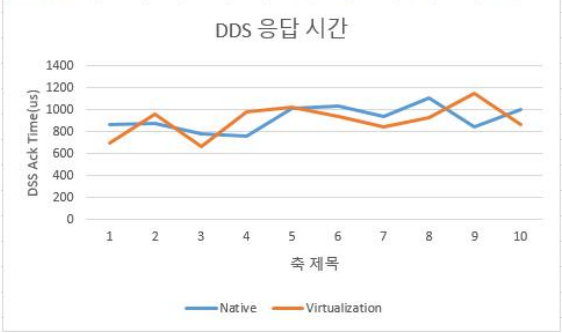

Fig. 14. Experimental result of transmission on DDS environment

3. Redundancy Test Result And Analysis

도커 기반 서버 가상화와 기존 잠수함 전투체계에서의 이중화 시험을 진행하였다. Active상태에서 Standby상태로 전환되는 시간과 유휴장비의 Standby상태로 전환되는 시간을 측정하여 도커 기반 서버 가상화의 적용 가능성을 확인하였다.

시험은 Native환경과 Virtualization환경에서 처리장치의 상태가 Standby에서 Active로 전환되는 시간을 측정하였다. 오류 발생 시점부터 Standby상태인 처리장치가 Active상태로 전환되는 시점을 측정을 하였다. Native환경은 처리장치PC 2대를 두고 시험을 하였으며 Virtualization환경은 도커 컨테이너 2개를 실행하여 Active/Standby상태의 처리장치를 구성하였다. 이중화 시험을 위해 전투체계의 동일한 소프트웨어를 실행하였다. Fig. 15의 결과를 보면 오류가 발생하여 Standby상태에서 Active상태로 전환되는 시간은 두 환경에서 1초 내로 전환되며 차이가 거의 나지 않음을 확인할 수 있다. 도커 기반 서버 가상화 환경에서 이중화 적용 가능함을 확인할 수 있다.

단위 : ms

횟수	1	2	3	4	5	6	7	8	9	10	평균
Native	909	820	840	920	930	750	790	833	799	780	837.1
Virtualization	887	920	866	776	845	954	855	844	912	822	868.1

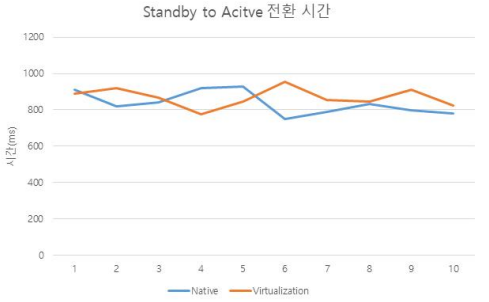

Fig. 15. Experimental result of Standby to Active swtich over

Fig. 16을 보면 유휴장비가 Standby상태로 변경되는 시간을 확인할 수 있다. Standby상태로 변경되는 시간은 유휴장비에 소프트웨어가 정상 실행된 상태를 기준으로 측정하였다. Native환경과 Virtualization환경의 시간을 비교해보면 평균적으로 1초 내외의 차이가 나는 것을 확인할 수 있다. Virtualization환경에서 도커 컨테이너가 실행되는 시간으로 차이가 발생하였다고 볼 수 있다. 시간 차이가 발생하나 도커 기반 서버 가상화를 적용하면 유휴장비를 운용하지 않고 동적으로 처리장치를 운용할 수 있으며 기존 잠수함 전투체계 대비 장비의 리소스 활용을 효율적으로 사용할 수 있다.

단위 : ms

횟수	1	2	3	4	5	6	7	8	9	10	평균
Native	2511	2764	3002	2714	2433	2766	3111	2974	2968	3187	2843
Virtualization	3622	4112	3801	4233	3766	3811	4322	3715	3965	3566	3891.3

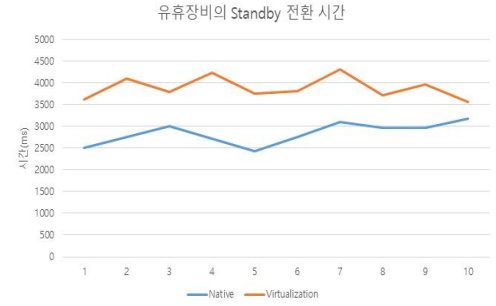

Fig. 16. Experimental result of Standby status swtich over of idle equipment

V. Conclusions

본 논문에서는 도커 기반 서버 가상화를 이용한 잠수함 전투체계를 설계하는 방법을 제시하였고 성능분석 및 이중화 시험을 통하여 적용 가능성을 실험하였다. 연동장치 및 처리장치의 가상화를 통하여 장치가 줄어듦으로써 잠수함 전투체계의 공간적인 제약과 체계 중량 감소를 해결할 수 있다. 그리고 도커가 설치되어 있다면 운영체제와 같은 환경적인 요소에 관계없이 컨테이너를 실행할 수 있어 확장성과 이식성도 높아질 수 있다. 도커 기반 서버 가상화를 적용한 처리장치를 구성하여 시험한 결과 CPU/Memory사용률은 총량에 비해 낮은 수치이며 전투체계 요구사항을 충분히 만족하였으며 DDS 메시지 응답결과 또한 차이가 거의 없었다. 가상화 환경에서 이중화 전환 시간을 측정한 결과에서도 기존 전투체계 환경와 차이가 없어 도커 기반 서버 가상화 적용 가능성을 확인하였다. 각 장치별로 사용하는 리소스가 많지 않기 때문에 리소스 사용에 대한 효율이 낮았으나 도커 기반 서버 가상화를 적용하게 된다면 리소스를 좀 더 효율적으로 사용할 수 있게 되며 기존 잠수함 전투체계에서와 같이 유휴장비를 운용하지 않아도 되기 때문에 최적화된 시스템 운용이 가능하다. 추후 연구로는 다른 가상화 솔루션을 적용 및 성능 분석하여 가상화 솔루션 간의 비교를 통해 어떠한 가상화 솔루션이 잠수함 전투체계 가상화에 적합한 지 연구를 해나갈 예정이다.

REFERENCES

[1] J. G. Lee, “A Study on the Standard Architecture of Weapon Control Software on Naval Combat System,” Journal of The Korea Society of Computer and Information Vol. 26 No. 11, pp. 101-110, November 2021.

[2] S. M. Kwon, and Seumg-Mo Jung, “Virtualization based high efficiency naval combat management system design and performance analysis,” Journal of The Korea Society of Computer and Information Vol. 23 No. 11, pp. 9-15, November 2018.

[3] D. W. Lee, B. K. Bae, and K. S. Chho, “A feasibility study of virtualization for Submarine Combat System,” Journal of The Korea Society of Computer and Information Vol. 27 No. 9, pp. 121-129, September 2022.

[4] C. S. Baek, and J. H. Ahn, “A Study of the Standard Interface Architecture of Naval Combat Management System,” Journal of The Korea Society of Computer and Information Vol. 26 No. 1, pp. 147-154, January 2021.

[5] K. T. Kwon, K. P. Kim, and H. J. Choi, “Design of the Scalable Naval Combat System Software using Abstraction and Design Pattern,” Journal of The Korea Society of Computer and Information Vol. 24 No. 7, pp. 101-108, July 2019.

[6] C. G. Yi, and Y. G. Kim, “A Study on software Security Test of Naval Ship Combat System,” Journal of The Korea Institude of Communications and Information Sciences, Vol. 45, No. 03, pp. 628-637, March 2020. DOI: 10.7840/kics.2020.45.3.628

[7] Naval Combat Management system, https://www.hanwhasystems.com/kr/business/defense/naval/combat_index.do

[8] E. S. Roberts, “Virtualization of Aegis: A Study of the Feasibility of Applying Open Architecture Technology to the Surface Navy’s Most Complex Automated Weapon System,” Monterey, CA: Naval Postgraduate School, pp .11, 2011.

[9] A. M. Biehn, J. R. Clarke, J. J. Juster, and T. E. Voth, “Weapon System Virtualization and Continuous Capability Delivery for US Navy Combat Systems,” Conference Proceedings of EAAW, 2 – 3, pp. 1-9, July 2019, DOI: 10.24868/issn.2515-8171.2019.010

[10] J. L. Yu, and K. W. Cho, “A Dynamic Virtual Machine Allocation Technique using Network Resource Contention,” International Journal of Software Engineering and Its Applications, Vol. 8, No. 9, pp.17-28, September 2014.

[11] K. S. Song, “A Study of Feasibility and Performance Analysis of VDI based on GPU Acceleration for Naval Combat System,” Journal of The Institute of Electronics and Information Engineers, Vol. 58, NO. 11, November 2021. DOI: 10.5573/ieie.2021.58.11.86

[12] S. W. Kang, S. J. Park, and C. I. Park, “Design and Implementation of a Device Virtualization Framework to control Virtual Desktop,” International Journal of Software Engineering and Its Applications, Vol. 16, No. 6, pp.702-706, June 2010.

[13] K. S. Song, S. C. Kwak, H. S. Lee, N. Y. Son, and Y. S. Han, "A Design and Development of On-board Training System for Engineering Control System based on Virtualization," Journal of The Institute of Electronics and Information Engineers, Vol. 59, NO. 8, August 2022. DOI: 10.5573/ieie.2022.59.8.104

[14] M. S. Lee, M. S. Kang, I. H. Kim, and J. H. Kim, "Design and Performance Comparison of Docker Container BasedDeep Learning Model Management System for Real-Time Analysis," The Journal of Korean Institute of Communications and Information Sciences, Vol. 46, No. 2, February 2021. DOI: 10.7840/kics.2021.46.2.390

[15] M. G. Park, and H. Lee, “Design and Implementation of Docker Container-based GPU Virtualization System and Web Services for Multiple Users,” KIISE Transactions on Computing Practices, Vol. 28, No. 1, pp. 42-50, January 2022. DOI: 10.5626/KTCP.2022.28.1.42

[16] D. G. Kim, Y. S. Park, and T. Y. Chung, "Design of Deep Learning Platform Considering Docker Based Big-data Analysis Environments,“ Journal of Korean Institute of Intelligent Systems, Vol. 32, No. 3, pp. 201-208, June 2022. DOI: 10.5391/JKIIS.2022.32.3.201

[17] Y. W. Jeong, "A Study on an Improved DDS Discovery Method for a Large-scale System," Journal of The Korea Society of Computer and Information, Vol. 25, No. 10, pp. 51-58, October 2020.

[18] DDS, https://www.omg.org/omg-dds-portal/

[19] J. W. SUK, and I. T. RYOO, "Data Transmission Processing System Design for Real-Time Distributed Simulation by Using Software Design Patterns," Journal of Digital Contents Society, Vol. 10, No. 4, pp. 649-657, Dec 2009.

Authors

Sang-Gil Son received the B.S. degrees in Computer Engineering from Pusan National University, Korea, in 2014. He is currently working in Hanwha Systems Co. from 2014. He is interested in Weapon Control Software of the naval combat management system, Server Virtualization.