https://doi.org/10.7840/kics.2025.50.8.1246

보안 컨테이너 기반 하이퍼레저 패브릭의 성능 및 자원 소모량 분석

강 윤 경◆, 최 원 미*, 양 경 식°, 유 혁°°

Performance and Resource Consumption Analysis of Hyperledger Fabric Based on Secure Containers

Yunkyung Kang◆, Wonmi Choi*, Gyeongsik Yang°, Chuck Yoo°°

요 약

하이퍼레저 패브릭(Hyperledger Fabric)은 허가형 블록체인을 운용하기 위한 오픈소스 플랫폼으로, 승인된 참가자만 네트워크에 접근하고 체인코드를 검증할 수 있어 높은 무결성과 보안성을 제공한다. 이에, 금융, 의료, 공급망 관리와 같은 다양한 환경에서 활용되며, 아마존, IBM과 같은 클라우드 환경에서도 널리 사용되고 있다. 기존 관련 연구들은 HLF의 성능 및 확장성을 개선하고 컨테이너 기반 노드의 배치에 초점을 맞추고 있으나, 컨테이너 환경에서 보안성을 강화하는 Kata 컨테이너와 그에 따른 성능 차이를 다룬 연구는 부족하다. 본 연구에서는 HLF의 격리 및 보안성을 강화하기 위해 보안 컨테이너인 Kata 컨테이너를 활용한 구동 방식을 제안하고, native 컨테이너와 비교하여 하이퍼레저 패브릭의 성능과 CPU 사용량을 분석한다. 그 결과, Kata 컨테이너는 추가된 가상화 계층으로 인해 CPU 병목 현상과 성능 저하를 보이지만, 상이한 노드별 vCPU 할당을 조정하면 이러한 병목을 완화할 수 있음을 확인하였다. 즉, 본 논문은 HLF의 노드별 CPU 자원 배분을 다각적으로 실험 및 분석함으로써, 보안성을 강화한 블록체인 컨테이너 환경 설계에 실질적인 참고 지표를 제공한다.

키워드 : 하이퍼레저 패브릭, 보안 컨테이너, Kata 컨테이너, 성능
Key Words : Hyperledger Fabric, Secure Container, Kata container, Performance evaluation

ABSTRACT

Hyperledger Fabric(HLF) is an open-source platform for operating permissioned blockchain, allowing only authorized participants to access the network and validate chaincode, thereby providing high integrity and security. Accordingly, it is utilized in various environment such as finance, healthcare, and supply chain management, and is widely used in cloud environments like Amazon and IBM. Existing related work focuses on improving the performance and scalability of HLF and on the placement of container-based nodes, but

※ 이 논문은 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(RS-2021-NR060143), 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(RS-2024-00336564, RS-2023-NR077249), 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 ICT명품인재양성사업(IITP-2025-RS-2020-II201819), 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(RS-2024-00405128)을 받아 수행된 연구임. 또한 본 연구는 Google Cloud Research Credits program의 지원을 받아 수행한 연구임.
◆ First Author : Korea University Department of Computer Science and Engineering, ykkang@os.korea.ac.kr, 학생회원
° Corresponding Author : Korea University Department of Computer Science and Engineering, g_yang@korea.ac.kr, 정회원
°° Corresponding Author : Korea University Department of Computer Science and Engineering, hxy@korea.ac.kr, 종신회원
* Korea University Department of Computer Science and Engineering, ymcui@os.korea.ac.kr
논문번호 : KICS202504-074-C-RU, Received April 1, 2025; Revised May 13, 2025; Accepted May 15, 2025

work addressing the security vulnerabilities when operating in containers is insufficient. In this study, we propose an operating method that utilizes the security container, Kata, to enhance the isolation and security of HLF, and compare it with native containers to analyze Hyperledger Fabric performance and CPU usage. Analysis results show that Kata containers exhibit CPU bottlenecks and performance degradation due to the additional virtualization layer, but these bottlenecks can be alleviated by allocating proper vCPUs per different Hyperledger Fabric nodes. This paper provides a practical reference for designing secure blockchain container environments by experimentally analyzing the CPU resource allocation per HLF node.

Ⅰ. 서 론

하이퍼레저 패브릭(Hyperledger Fabric, HLF)[1]은 Linux foundation에서 제공하는 오픈 소스 허가형 블록체인 플랫폼이다. 허가형 블록체인은 비트코인, 이더리움과 같은 퍼블릭 블록체인과 달리, 사전에 승인된 참여자들만 블록체인 네트워크에 접근하고 스마트 컨트랙트인 체인코드를 검증할 수 있도록 설계되어 높은 무결성과 보안성을 제공할 수 있다. 이러한 이점으로 HLF는 아마존, IBM 등의 클라우드 환경을 제공하는 기업에서 널리 사용되고 있다[2].

HLF를 포함하여 Hyperledger Besu, Sawtooth, Quorum, Parity[3,4], R3 Corda[5]과 같이 다양한 허가형 블록체인 플랫폼은 배포 및 운영의 편의성, 확장성, 격리성 보장을 위해 Docker, 쿠버네티스와 같은 컨테이너 기반으로 구축되고 있다[6].

HFS Research의 2020년 보고서에 따르면[7], IBM, Accenture 등 12개 대형 서비스 기업이 수행한 약 425건 중 88% 이상의 기업용 블록체인 프로젝트가 구축된 것으로 나타났다. 이처럼 컨테이너 기반 구조는 많은 허가형 블록체인에서 주로 채택되고 있으며, HLF 또한 native 컨테이너 런타임(runC[8])을 기반으로 peer, orderer 등 다양한 노드를 구성하고 실행한다. 이러한 구조는 운영 편의성과 확장성, 그리고 자원 효율성을 확보할 수 있으며, 대부분 자원이 공유되는 클라우드 환경에서 경량성이 중요한 요소로 작용한다[9].

그러나 native 컨테이너는 호스트 커널을 공유하는 구조이기 때문에, 이미지 크기나 부팅 속도에서는 유리하더라도, 보안 격리 측면에서는 구조적인 한계를 가진다. 예를 들어, 관리 취약점(CVE-2017-5123)이나 권한 상승 취약점(CVE-2016-5195)[10]과 같은 커널 단위의 공격이 발생할 경우, 단일 컨테이너의 침해가 전체 시스템으로 확산될 위험이 존재한다. 특히 HLF는 여러 종류의 노드(peer, orderer 등)가 서로 통신하고 상태를 검증하는 구조로 되어 있어, 하나의 노드라도 침해될 경우 전체 블록체인 네트워크의 무결성과 신뢰성을 위협할 수 있다. 이는 HLF가 활용되는 의료, 금융, 공급망 관리 시스템 등 고신뢰 서비스 분야에서 치명적인 보안 문제가 될 수 있다[11].

하지만 현재까지의 HLF 관련 연구들은 대부분 노드 배치나 확장성에 따른 성능 평가[12-15]에 집중되어 있으며, 컨테이너 기반 환경에서 발생할 수 있는 보안 문제를 해결하기 위한 실행 구조에 대한 성능 분석은 충분히 이루어지지 않았다. 또한 기존의 컨테이너 플랫폼 간 성능 비교 연구들[16-20]은 주로 단순 요청-응답 워크로드를 기반으로 하여, 노드 간 복잡한 연산과 통신이 빈번하게 발생하는 블록체인 환경의 특성을 제대로 반영하지 못하고 있다.

이에 본 논문에서는 기존의 native 컨테이너 기반 실행 환경의 보안 한계를 극복하기 위한 대안으로, 보안 컨테이너인 Kata 컨테이너 기반 HLF 실행 구조를 선택하였다. 보안 컨테이너는 호스트 운영체제와 컨테이너 사이에 계층을 하나 추가하여, 악의적인 접근이나 보안 취약점을 방지한다. 대표적인 보안 컨테이너는 gVisor, Firecracker, Kata 등이 있으며 각 런타임은 각기 다른 보안 및 격리 메커니즘을 가진다.

gVisor는 시스템 콜을 사용자 공간에서 에뮬레이션하여 높은 수준의 격리를 제공하지만 사용자 공간과 커널 공간 간의 빈번한 context switch로 인해 CPU 오버헤드와 처리 지연이 증가하는 문제가 있다[17-20]. Firecracker는 경량화된 가상 머신인 MicroVM 기반의 구조를 통해 호스트 커널과의 격리를 제공하지만 I/O 요청을 단일 이벤트 루프에서 순차적으로 처리하는 방식으로 인해 병렬 처리가 불가능하고 I/O 처리량이 제한됨이 알려져 있다[18,19].

반면, Kata 컨테이너[21]는 각 컨테이너를 독립된 MicroVM 내에서 실행하며[22] I/O 요청을 병렬로 처리할 수 있는 구조로 복잡한 연산과 빈번한 I/O 요청이 발생하는 상대적으로 낮은 CPU 오버헤드와 높은 처리량을 보이는데[17-20], 이러한 구조는 보안성과 성능 간 균형이 요구되는 HLF 환경에 적합하다.

따라서 본 논문에서는 Kata 컨테이너 기반의 HLF

실행 환경을 구성하여 HLF의 성능과 자원 소모량을 상세히 분석한다.

HLF를 보안 컨테이너로 구동할 때 가장 큰 문제점은, 보안을 위한 MicroVM과 같은 추가적인 계층으로 인해 처리량 등의 HLF 성능 하락과 CPU 등의 추가적인 자원 소모가 발생할 수 있다는 점이다.

따라서 본 논문은 Kata를 기반으로 HLF를 운용할 때, 성능 및 자원 소모량의 변화를 상세히 분석한다. 또한, HLF를 구성하는 peer, orderer 와 같이 특성이 상이한 노드들이 보안 컨테이너로 운용될 때 각 노드가 필요한 자원의 양과 성능 관의 관계를 분석하여, Kata를 기반으로 HLF 시스템을 구성하고 운용할 때 native 컨테이너 기반 시스템과 유사한 성능을 가지는 방안을 제시한다. 요약하면, 본 논문의 기여점은 아래와 같다.

- 기존 runC 컨테이너 대비 보안성이 강화된 Kata 컨테이너를 활용하여 HLF를 구동하고, 해당 환경에서의 성능(TPS, 지연)과 CPU 자원 소모량의 변화를 최초로 정량적으로 분석한다.
- Kata 컨테이너 기반 HLF는 native 컨테이너에 비해 보안성을 제공하지만, 추가 가상화 계층으로 인해 HLF의 처리량이 감소할 수 있으며, 물리 CPU의 개수에 비례하여 CPU 사용량이 증가함을 확인한다.
- HLF를 구성하는 노드의 종류별로 CPU 할당량을 차등하여, native 컨테이너와 유사한 성능에 도달할 수 있음을 확인한다.
- 보안성을 강화한 HLF 시스템 구성 시 자원 배분 전략 및 최적 운영을 위한 방안을 제시한다.

Ⅱ. 하이퍼레저 패브릭과 컨테이너 가상화 기술

2.1 HLF

그림 1은 HLF을 기반으로 구축된 기본적인 형태의 블록체인 구조를 보여주며, 1) peer 노드, 2) orderer 노드, 및 3) 채널로 구성된다. 첫째, peer 노드는 트랜잭션을 저장하는 원장(그림1에서의 Leger1, Leger2)과 원장에 저장된 트랜잭션을 읽거나 쓰기 위한 체인코드(그림 1의 CC1, CC2)를 관리한다. 두개 이상의 peer 노드가 하나의 조직(organization, 그림 1의 Org0, Org1)에 속하게 되고, 각 조직은 사용자 및 노드의 인증서 발급과 관리를 위해 certificate authority (CA)를 운영한다.

둘째, orderer 노드는 peer 노드에서 검증된 트랜잭션을 정렬하고 블록을 생성하는 역할을 수행한다. 특정 조직이 orderer를 독점하는 것을 방지하고 여러 조직이 신뢰할 수 있도록, orderer는 peer와 독립된 조직에서 운영된다.

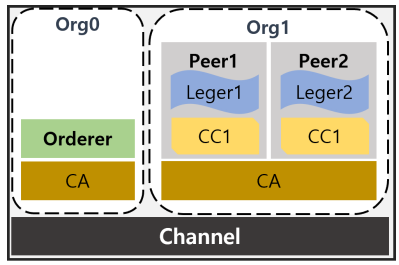


그림 1. HLF 구조
Fig. 1. HLF Structure

마지막으로, 채널은 여러 조직이 트랜잭션을 안전하게 공유할 수 있도록 서로 격리시킨다. 즉, 같은 채널에 속한 노드들은 트랜잭션, 원장, 및 체인코드를 공유하지만, 서로 다른 채널 간에는 정보를 공유할 수 없다. 이러한 블록체인 시스템은 클라이언트로부터 받은 트랜잭션마다 무결성을 보장하기 위해 실행, 정렬, 검증의 세 단계를 거쳐 처리한다[23]. 각 단계별 연산은 다음과 같다.

1) 실행: peer 노드들이 트랜잭션을 검증하고 체인코드를 실행하여 유효성을 검증하고 유효성이 승인된 트랜잭션을 orderer 노드에게 전송

2) 정렬: peer 노드로부터 승인된 트랜잭션들을 합의 프로토콜에 따라 순서대로 정렬하고, 이를 블록 단위로 묶어 모든 peer 노드에게 배포

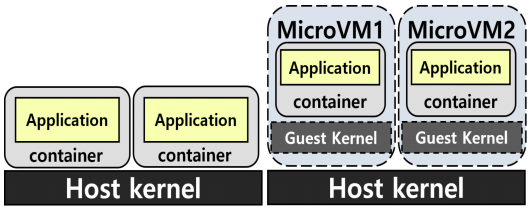
3) 검증: peer 노드는 블록 내 트랜잭션들을 최종 검증하고 유효한 트랜잭션만 원장에 기록

HLF의 개별 노드는 응용 프로그램(application)의 형태인 독립적인 컨테이너로 구동한다. 또한, 트랜잭션을 처리하는 다수 노드를 손쉽게 운용하기 위해 HLF는, 컨테이너들의 생성과 배포를 손쉽게 관할하기 위해 컨테이너 운용 플랫폼인 쿠버네티스(Kubernetes)[24]를 함께 사용하고 있다.

2.2 Native 컨테이너와 Kata 컨테이너

2.2.1 native 컨테이너

그림 2a는 쿠버네티스에서 표준으로 사용하는 runC 기반의 native 컨테이너구조를 나타낸다. Native 컨테이너는 모든 컨테이너가 하나의 운영체제 커널을 공유하며, 컨테이너 내에는 프로세스 구동에 필요한 라이브러리를 포함하며 별도의 운영체제 커널을 포함하지 않아 실행 속도가 빠르다. 또한, 컨테이너 내의 I/O 요청은 호스트 커널을 통해 직접 처리되므로 상대적으로 성능이 높고 지연이 적다. 그러나, 운영체제 커널을 공유하는 구조적 특성상 컨테이너들과 커널 사이의 격리 수준



(a) Native container (b) Kata container

그림 2. Native 및 Kata 컨테이너의 구조
Fig. 2. Architecture of Native and Kata container

이 낮다. 가령, 하나의 컨테이너에서 악성 코드나 보안 공격이 발생한 경우, 동일 호스트의 다른 컨테이너에도 영향을 줄 수 있다[25].

CPU 할당 방식: Native 컨테이너는 CPU를 별도로 가상화하지 않고 Linux cgroup과 쿠버네티스 스케줄러를 통해 물리 CPU(physical CPU, pCPU) 코어를 time-slicing 방식으로 여러 컨테이너가 공유한다. 쿠버네티스 스케줄러는 각 컨테이너별 CPU(vCPU)에 대한 최소 요구 자원(requests)과 최대 사용 자원(limits)를 설정할 수 있으며, 요청된 requests의 vCPU의 합계가 pCPU를 초과할 경우 해당 컨테이너를 배치하지 못하고 스케줄링은 실패한다[26].

또한, 컨테이너의 requests, limits를 설정하지 않으면 쿠버네티스 스케줄러는 리눅스 운영체제의 CPU 스케줄러인 completely fair scheduler(CFS) 메커니즘을 사용하여 자원을 동적으로 분배한다. 이러한 방식은 최소한의 자원 보장이 어려울 수 있지만, 전체적인 CPU 활용도를 높이고 유연한 자원 할당이 가능하다.

2.2.2 Kata 컨테이너

그림 2b는 Kata 컨테이너의 구조를 나타낸다. Kata 컨테이너는 각 컨테이너가 독립적인 운영체제 커널을 갖는 경량 가상 머신(MicroVM) 내부에서 실행되며, MicroVM은 리눅스의 QEMU[27]-KVM[28]를 기반으로 생성된다. 이러한 구조는 컨테이너 별 독립된 커널 환경을 제공하여, 특정 컨테이너가 공격을 받더라도 다른 컨테이너나 호스트에 영향을 미치지 않는 높은 격리 수준을 보장한다.

MicroVM은 컨테이너 구동에 필요한 핵심적인 커널 기능만 포함하기 때문에 일반적인 가상머신에 비해 가볍지만 native 컨테이너보다는 상대적으로 많은 연산을 거친다. 가령, Kata 컨테이너에서 발생한 I/O 요청은 호스트 사용자 영역에 위치한 QEMU의 virtio 드라이버를 거쳐 호스트 커널 영역으로 전달되는데, 이는 native 컨테이너에서 요청이 바로 호스트 커널로 전달되는 과정에 비해 추가적인 연산을 필요로 한다.

CPU 할당 방식: 개별 Kata 컨테이너 또한 native 컨테이너와 마찬가지로 requests와 limits를 설정할 수 있는데, Kata 컨테이너는 QEMU을 통하여 pCPU를 vCPU로 가상화하여 할당한다[29]. 이는 컨테이너의 vCPU 할당 방식과 다르게 하나의 pCPU를 여러 개의 vCPU로 가상화할 수 있으며, 각 vCPU는 QEMU 프로세스 내의 리눅스 커널 쓰레드로서 동작한다. 이러한 vCPU 쓰레드는 native 컨테이너가 호스트 커널에서 컨테이너의 개별 프로세스를 직접 스케줄링하는 방식과 구조적으로 차이가 있다. Kata 컨테이너에서는 호스트 커널이 QEMU의 vCPU 쓰레드를 스케줄링하고 컨테이너 내부의 프로세스 및 쓰레드는 MicroVM 내부의 커널에 의해 관리 및 실행된다. 또한, vCPU는 호스트 시스템의 가상화 지원(ex. Intel VT-x, AMD-V) 상태, pCPU 코어 개수, 그리고 OS와 하이퍼바이저의 CPU 스케줄링 정책에 따라 그 수를 제어할 수 있다. 즉, Kata 컨테이너들의 limits의 합이 pCPU를 초과하더라도 생성 가능한 vCPU 수 범위 내라면 할당이 가능하다(오버커밋)[30]. 다만, 과도한 오버커밋은 vCPU 간의 스케줄링 지연, TLB flush 증가, context switching 증가로 인한 성능 저하를 유발함이 알려져 있다[31].

Ⅲ. 관련 연구 비교 및 분석

본 장에서는 관련연구로서 1) HLF 성능 분석과 2) 컨테이너 플랫폼 성능 비교 측면의 기존 연구들을 요약하고, 본 연구의 필요성을 설명한다.

여기서는 소단원에 관한 내용을 간단히 살펴본다. 여기서는 소단원에 관한 내용을 간단히 살펴본다.

3.1 HLF 성능 분석

현재까지 HLF 자체에 대한 논문은 대부분 플랫폼 자체의 성능 평가에 국한되어 있다[13-15]. 가령, HLF의 처리량과 지연을 향상시키기 위한 시스템 구성(orderer와 peer 노드 개수) 및 내부 파라미터 튜닝에 관한 연구들이 다수 진행되어 있으며[22,32], HLF와 이더리움의 트랜잭션 처리 성능을 비교한 연구[11]가 존재한다. 해당 연구들 모두 native 컨테이너 기반의 분석 논문으로서, 컨테이너의 보안 취약점을 보완한 보안 컨테이너에 대한 고려 및 그에 따른 성능 차이를 다룬 연구는 부족하다.

3.2 컨테이너 플랫폼 성능 비교 분석

또 다른 축으로서, native 컨테이너와 Kata 컨테이너 간의 성능을 비교한 연구들이 존재한다.

표 1. 실험 환경 구성
Table 1. Experimental environment setup

	Version
K8S	1.23
Kata	3.2.0
runC	1.1.12
Hyperledger fabric	2.4.4
Caliper	0.6.0

먼저, 기존연구들은 웹/DB 서비스(Redis, Nginx, MySQL)[16-19] 또는 마이크로 서비스(TeaStore, Spark)[20] 기반 워크로드를 구동하는 상황에서 native 및 Kata 컨테이너의 성능을 비교한다. 이러한 워크로드는 대부분 요청-응답 기반의 서비스 또는 한 컨테이너를 생성한 상태에서 성능을 분석한다. 반면, 본 연구는 HLF를 대상으로 분석을 수행하는데, HLF 환경은 1) 다수의 컨테이너가 동시에 구동되며, 2) 컨테이너 간 합의(consensus) 프로세스, 원장 데이터 저장 및 관리, 체인코드 실행 등 빈번한 통신과 복잡한 연산을 수행한다. 이러한 특징은 기존 웹/DB 워크로드와 큰 차이를 보이며, 특히 Kata 컨테이너의 복잡한 I/O 연산 과정에서 발생하는 병목 현상이 더욱 두드러질 수 있다.

Ⅳ. Native 컨테이너와 Kata 컨테이너의 성능 비교 분석

4.1 실험 환경

본 실험에서는 Google cloud platform을 기반으로, 두 대의 서버를 생성하여 수행한다. 두 대의 서버는 각각 us-west1-b 데이터센터에 위치한 C3(8 CPU 코어와 32 GB 메모리)과 E2(2 CPU 코어와 4 GB 메모리) 서버들이며, 서로 1 Gbps 속도의 네트워크로 연결된다. HLF를 구동하는 컨테이너들은 C3 서버에서 실행한다. 또한, HLF의 성능(처리량 및 지연)을 측정하기 위해 트랜잭션을 생성하는 Caliper benchmark[33]를 E2 서버에서 실행한다. HLF는 그림 1에서 설명한 것과 같이 가장 기본적인 구조인 orderer 노드 1개와 peer 노드 2개, 각 peer 노드에 대한 chaincode와 CA 2개로 구성하며, 총 7개의 컨테이너가 사용된다.

실험에 사용하는 소프트웨어 및 버전은 표1과 같다. 컨테이너 플랫폼으로서 기본 런타임인 runC 1.1.12 기반 native 컨테이너와 Kata 컨테이너 3.2.0를 사용하며, 모두 쿠버네티스(Kubernetes, K8S)를 기반으로 생성된다. HLF는 2.4.4, Caliper는 0.6.0를 사용한다.

매 실험 시 Caliper는 HLF에 초당 300, 400, 500, 600, 700개의 트랜잭션을 전송하고, HLF은 수신받은 트랜잭션을 각각 처리한다. 실험 결과 서로 다른 트랜잭션 처리 성능의 경향성이 유사하게 나타나, 본 논문에서는 초당 300개의 트랜잭션을 전송한 실험 결과를 기준으로 분석을 수행한다.

Kata 컨테이너가 native 컨테이너 대비 지니는 성능 차이 및 CPU 소모량을 비교 분석하기 위해, 본 연구에서는 HLF가 구동되는 C3머신에서 컨테이너가 사용가능한 물리 CPU의 수를 2, 4, 6, 8 코어로 증가시키며 실험한다. 매 상황마다, HLF의 초당 트랜잭션 처리량(transactions per second, TPS)과 트랜잭션별 평균 지연을 측정한다.

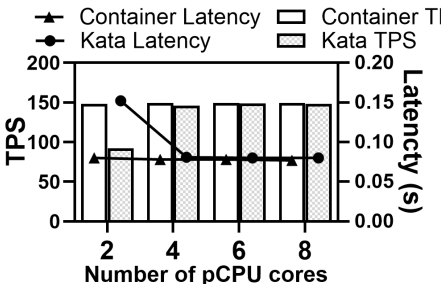
4.2 HLF 성능 및 CPU 사용량 분석

본 장에서는 초당 300개의 트랜잭션을 전송하는 상황에서 2, 4, 6, 8개 코어들(pCPU)이 존재할 때, native 컨테이너와 Kata 컨테이너 간 HLF 성능(TPS, 지연)과 CPU 사용량을 분석한다. pCPU들은 존재하는 컨테이너들에게 모두 동일한 양만큼 균등하게 할당하여 실험한다. CPU 사용량을 성능분석 지표로 선정한 이유는 HLF이 체인코드 실행, 블록 생성 및 검증 등 과정에서 CPU 연산이 빈번하게 발생하기 때문에, CPU 자원 사용의 효율성이 HLF의 성능에 직접적인 영향을 미친다[34-36]. 이에 비해 메모리 사용량 등 다른 리소스 사용량은 HLF 성능과 무관하게 일정 수준으로 유지된다는 선행연구 결과가 있다[37]. 따라서 본 연구에서는 HLF 환경에서 컨테이너 런타임에 따른 성능 차이를 정량적으로 평가할 수 있는 주요 지표로 CPU 사용량을 선정하여 분석을 수행한다.

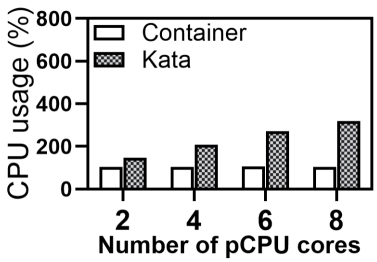
pCPU 개수에 따른 HLF 성능. 그림 3a는pCPU가 2, 4, 6, 8 코어(x축)로 주어질 때, HLF의 평균 TPS(왼쪽 y축 및 막대 그래프)와 평균 지연(오른쪽 y축 및 실선 그래프)을 보여준다. 첫째, 2 코어가 주어진 상황에서는 Kata 컨테이너가 native 컨테이너 대비 TPS와 평균 지연이 각각 38.02%, 89.99%의 성능 저하를 보인다. 이러한 점은 Kata 컨테이너의 추가 계층인 MicroVM이 HLF의 개별 트랜잭션 처리에 추가적인 지연과 병목을 발생시킴을 드러낸다.

둘째, pCPU가 4코어 이상인 상황에서는 native 컨테이너와 Kata 컨테이너의 성능 차이가 크게 줄어든다. 예컨대, 4코어 환경에서 Kata 컨테이너의 TPS는 native 대비 2.3% 낮고, 지연은 3.8% 높다. 이 주어진 실험 환경에서 pCPU가 4코어 이상일 경우, HLF CPU 자원 부족으로 인한 병목 현상이 거의 사라짐을 의미한다.

CPU 사용량 상세분석. 그림 3b는 그림 3a와동일하



(a) Transaction performance (TPS and latency)



(b) CPU usage

그림 3. Native container와 Kata container의 HLF 성능 및 CPU 사용량
Fig. 3. HLF performance and CPU usage comparison between native and Kata containers

게 pCPU 코어 수가 증가할 때, HLF를 구동하는 컨테이너들의 평균 CPU 사용량을 보여준다. 첫째, 코어 수가 변화하는 모든 상황에서 Kata 컨테이너가 native 컨테이너보다 많은 CPU 자원을 소모한다. 구체적으로, 코어가 2개일 때, Kata 컨테이너는 native 컨테이너보다 41.35% 많은 CPU 자원을 사용한다. 둘째, 코어의 개수가 증가할수록 native 컨테이너 대비 Kata 컨테이너가 추가적으로 소모하는 CPU 사용량 또한 증가하는 경향을 보인다. 구체적으로, pCPU가 2코어에서 8코어로 증가할 때 native 컨테이너 대비 Kata 컨테이너의 추가 CPU 사용량은 41.35%에서 204%로 증가한다.

이러한 결과를 세부적으로 살펴보면, 우선 HLF를 구동하는 컨테이너의 기본 구조는 native와 Kata 모두 유사하다[12]. 두 컨테이너 간의 주요 차이는 보안성을 강화하기 위해 추가된 커널 수준의 가상화 계층(KVM, QEMU 등)이다. 즉, Native 컨테이너와 Kata 컨테이너의 CPU 사용량 차이는 바로 이 추가 가상화 계층에서 소모되는 자원에 기인한다. 즉, 그림 3b의 실험 결과는 pCPU 코어 수가 증가할수록 Kata 컨테이너의 추가 가상화 계층에서 소비되는 CPU 자원도 비례하여 늘어난다는 점을 시사한다.

Ⅴ. HLF 노드 종류별 CPU 자원 및 성능 관계 분석

5.1 실험 환경

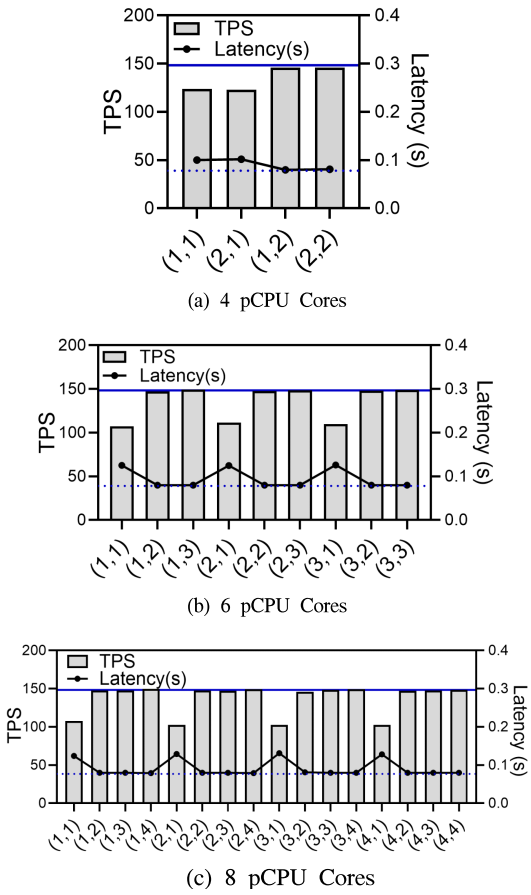
본 장에서는 Kata 컨테이너 기반의 HLF에서 블록체인 시스템을 구성하는 두 가지 종류의 노드인 orderer와 peer 노드를 나누어 분석한다. 동일한 수의 pCPU가 존재하더라도, 노드 종류별 서로 다른 양의 CPU(vCPU)를 할당 가능하다. 이에 우리는 pCPU 코어가 4, 6, 8개 존재할 때, 두 종류의 노드에 서로 상이한 vCPU를 할당하면서 HLF의 성능을 분석한다. pCPU 코어가 2개일 때는, 두 종류의 노드에 항상 1개의 vCPU를 할당하게 되어 IV장의 분석 결과와 동일하다. 모든 실험은 IV장과 동일한 환경에서 수행하고, vCPU의 할당만 상이하게 수행한다.

5.2 노드별 CPU 할당에 따른 HLF 성능 분석

그림 4는 4, 6, 8개의 pCPU 코어가 존재할 때의 실험 결과를 보여준다. 각 그래프의 x축은 orderer 및 peer 노드에 할당하는 vCPU의 개수를 나타내며, 즉, (2, 1)은 orderer 노드와 peer 노드에 각각 vCPU 2개와 1개를 할당한다. 각 그래프는 HLF의 처리량(왼쪽 y축, 막대)과 지연(우측 y축, 검은색 실선)을 보여준다. 또한, 그래프에는 파란색 실선과 점선이 그어져 있는데 (가령 그림 4a에서 왼쪽 y축 기준 150 TPS 근처 실선과 우측 y축 기준 0.9 근처의 점선), 이들은 각각 native 컨테이너 환경에서 동일 실험을 수행할 때 달성하는 최대 TPS와 최소 지연을 나타낸다. native 컨테이너는 물리 CPU를 가상화하지 않기 때문에 유한한 pCPU 코어를 HLF 컨테이너들이 공유하여 사용할 수 있도록 별도의 자원 제한을 설정하지 않았다. 이러한 설정에서 native 컨테이너는 pCPU 자원을 최대로 활용하여 최대 성능을 달성하는 것을 확인하였다.

첫째, 모든 pCPU 코어 실험에서 peer 노드에 vCPU를 2개 이상 할당하면, Kata 기반의 HLF는 native 컨테이너에서의 최고 TPS(파란 실선)와 최소 지연(파란 점선)과 매우 근접한 성능을 달성한다. 예를 들어, 4pCPU 코어의 (1, 2)에서 TPS는 최고 TPS와 지연 대비 2.15%, 지연은 2.56%의 차이만을 보이고, 6pCPU 코어의 (1, 3)에서도 TPS는 0.12%,

지연은 2.56%의 증가하였다. 또한, 8pCPU의 (1, 4)에서도 TPS, 지연이 각 0.24%, 2.59% 증가하는 것을 확인하였다. 이는 peer 노드가 트랜잭션 수신, 체인코드 실행, 트랜잭션 검증 등 많은 연산을 수행하기에 최소 2개 이상의 CPU가 필요함을 보여준다. 가령, peer 노드



(a) 4 pCPU Cores

(b) 6 pCPU Cores

(c) 8 pCPU Cores

그림 4. HLF 노드별 vCPU 차등할당 시 HLF 성능 분석
Fig. 4. HLF performance with different vCPU allocations per HLF node

에 vCPU를 적게 할당하면, 최초 트랜잭션 처리를 담당하는 peer 노드의 병목으로 전체 HLF 성능이 저하된다.

둘째, orderer 노드의 경우, vCPU 수가 증가하더라도 HLF의 성능에 큰 영향을 주지 않는다. 가령, 8pCPU 코어의 (4, 1)를 보면 vCPU 수가 늘어나더라도 HLF의 TPS와 지연은 각 31.27%, 66.23% 차이가 나타난다. orderer 노드는 각 peer가 수신하고 검증한 트랜잭션을 합의 프로토콜에 따라 정렬하여 다시 peer로 전달한다[38]. 즉, 트랜잭션 데이터에 대한 해싱 등 CPU 집약적인 작업을 수행하는 peer에 비해 orderer는 상대적으로 적은 연산을 수행한다[23]. 따라서, 적은 양의 vCPU로도 HLF에 필요한 연산을 충분히 처리할 수 있다.

상기 결과들은 HLF 성능 최적화를 위해 자원 배분에 차별화가 필요함을 시사한다. peer 노드에는 최소 2개 이상의 vCPU를 할당하여 트랜잭션 처리에 따른 병목 현상을 방지해야 하며, 반면 orderer 노드는 적은 vCPU 할당으로도 충분한 성능을 발휘할 수 있다.

Ⅵ. 결 론

본 연구는 HLF의 격리와 보안성을 높이는 Kata 컨테이너 기반의 구동 방안을 제시하고, 이를 native 컨테이너와 비교 분석하였다. 실험 결과, Kata 컨테이너는 추가 가상화 계층으로 인한 CPU 오버헤드와 성능 저하가 발생하나, 특히 트랜잭션 처리 작업이 많은 peer 노드에 충분한 vCPU를 할당하면 native 컨테이너 수준의 성능을 유지할 수 있음을 확인하였다. 상기 결과는 보안성을 강화한 HLF 시스템 운용 시 노드별 자원 배분 전략이 필수적임을 시사하며, 향후 가상화 계층 최적화를 통한 오버헤드 감소 방안을 위한 연구의 필요성을 제시한다.

References

[1] *Hyperledger Fabric Docs*, Retrieved Feb. 20 2025, from https://hyperledger-fabric.readthedocs.io/en/release-2.5

[2] E. Androulaki, et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. Thirteenth EuroSys Conf. 2018,* Article 30, pp. 1-15, Association for Computing Machinery, New York, NY, USA, 2018. (https://doi.org/10.1145/3190508.3190538)

[3] P. W. Eklund and R. Beck, "Factors that impact blockchain scalability," in *Proc. 11th Int. Conf. Manag. Digital EcoSystems (MEDES '19),* pp. 126-133, Association for Computing Machinery, New York, NY, USA, 2020. (https://doi.org/10.1145/3297662.3365818)

[4] G. Volpe, A. M. Mangini, and M. P. Fanti, "An architecture combining blockchain, docker and cloud storage for improving digital processes in cloud manufacturing," in *IEEE Access*, vol. 10, pp. 79141-79151, 2022. (https://doi.org/10.1109/ACCESS.2022.3194264)

[5] *r3 - Corda Enterprise 4.8*, Retrieved May, 08 2025, from https://docs.r3.com/en/platform/corda/4.8/enterprise/docker-image.html

[6] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "BLOCKBENCH: A framework for analyzing private blockchains,"

in *Proc. 2017 ACM Int. Conf. Manag. Data (SIGMOD '17),* pp. 1085-1100, Association for Computing Machinery, New York, NY, USA, 2017. (https://doi.org/10.1145/3035918.3064033)

[7] *HFS Research, Top 10 Enterprise Blockchain Service Providers 2020* (2020), Retrieved May 9, 2025, from https://walton.uark.edu/departments/information-systems/files/hfs_top10_2004-enterpriseblockchainservices2020-1.pdf

[8] *runC*, Retrieved Feb. 20, 2025, from https://github.com/opencontainers/runc

[9] Y. Kang, T. Roh, J. Lee, W. Choi, Y. Yoo, G. Yang, and C. Yoo, "Exploring the secure container for permissioned blockchains," in *Proc. Symp. KICS*, pp. 1853-1854, 2024.

[10] S. Sultan, I. Ahmad, and T. Dimitriou, "Container security: Issues, challenges, and the road ahead," in *IEEE Access,* vol. 7, pp. 52976-52996, 2019. (https://doi.org/10.1109/ACCESS.2019.2911732)

[11] R. Yang, et al., "Public and private blockchain in construction business process and information integration," *Automat. in Construction,* vol. 118, p. 103276, 2020. (https://doi.org/10.1016/j.autcon.2020.103276)

[12] Y. Ucbas, A. Eleyan, M. Hammoudeh, and M. Alohaly, "Performance and scalability analysis of ethereum and hyperledger Fabric," in *IEEE Access,* vol. 11, pp. 67156-67167, 2023. (https://doi.org/10.1109/ACCESS.2023.3291618)

[13] H. H. Pajooh, M. A. Rashid, F. Alam, and S. Demidenko, "Experimental performance analysis of a scalable distributed hyperledger fabric for a large-scale IoT testbed," *Sensors*, vol. 22, p. 4868, 2022. (https://doi.org/10.3390/s22134868)

[14] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability," *2019 IEEE Int. Conf. Blockchain(Blockchain)*, pp. 536-540, Atlanta, GA, USA, 2019. (https://doi.org/10.1109/Blockchain.2019.00003)

[15] S. Shalaby, A. A. Abdellatif, A. Al-Ali, A. Mohamed, A. Erbad, and M. Guizani, "Performance evaluation of hyperledger fabric," *2020 IEEE Int. Conf. Inf., IoT, and Enabling Technol. (ICIoT)*, pp. 608-613, Doha, Qatar, 2020. (https://doi.org/10.1109/ICIoT48696.2020.9089614)

[16] R. Kumar and B. Thangaraju, "Performance analysis between runC and kata container runtime," *2020 IEEE Int. Conf. Electr., Computing and Commun. Technol. (CONECCT)*, pp. 1-4, Bangalore, India, 2020. (https://doi.org/10.1109/CONECCT50063.2020.9198653)

[17] X. Wang, J. Du, and H. Liu, "Performance and isolation analysis of runC, gVisor and kata containers runtimes," *Cluster Comput 25*, pp. 1497-1513, 2022. (https://doi.org/10.1007/s10586-021-03517-8)

[18] G. Li, K. Takahashi, K. Ichikawa, H. Iida, P. Thiengburanathum, and P. Phannachitta, "Comparative performance study of lightweight hypervisors used in container environment," in *CLOSER 2021*, pp. 215-223, Apr. 2021. (https://doi.org/10.5220/0010440502150223)

[19] S. Jeon, S. Shim, H. Chung, and Y. Nah, "OCI runtime comparison and analysis study," *2022 IEEE Fifth Int. Conf. AIKE*, pp. 65-66, Laguna Hills, CA, USA, 2022. (https://doi.org/10.1109/AIKE55402.2022.00017)

[20] W. Viktorsson, C. Klein, and J. Tordsson, "Security-performance trade-offs of kubernetes container runtimes," *2020 28th Int. Symp. MASCOTS*, pp. 1-4, Nice, France, 2020. (https://doi.org/10.1109/MASCOTS50786.2020.9285946)

[21] Kata Containers, *The speed of containers, the security of VMs*, Retrieved Feb. 20, 2025, from https://katacontainers.io/

[22] A. Randazzo and I. Tinnirello, "Kata containers: An emerging architecture for enabling MEC services in fast and secure way," *2019 Sixth Int. Conf. IOTSMS*, pp. 209-214, Granada, Spain, 2019.

(https://doi.org/10.1109/IOTSMS48152.2019.8939164)

[23] G. Yang, K. Lee, K. Lee, Y. Yoo, H. Lee, and C. Yoo, "Resource analysis of blockchain consensus algorithms in hyperledger fabric," in *IEEE Access*, vol. 10, pp. 74902-74920, 2022. (https://doi.org/10.1109/ACCESS.2022.3190979)

[24] *Kubernetes*, Retrieved Feb. 20, 2025, from https://kubernetes.io/docs/home/

[25] S. Sultan, I. Ahmad, and T. Dimitriou, "Container security: Issues, challenges, and the road ahead," in *IEEE Access*, vol. 7, pp. 52976-52996, 2019. (https://doi.org/10.1109/ACCESS.2019.2911732)

[26] C. Carrión, "Kubernetes scheduling: Taxonomy, ongoing issues and challenges," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 1-37, Jul. 2023. (https://doi.org/10.1145/3539606)

[27] *QEMU*, Retrieved Feb. 20, 2025, from https://www.qemu.org

[28] *KVM*, Retrieved Feb. 20, 2025, from https://linux-kvm.org/page/Main_Page

[29] S. Schildermans, J. Shan, K. Aerts, J. Jackrel, and X. Ding, "Virtualization overhead of multithreading in X86 state-of-the-art & remaining challenges," in *IEEE Trans. Parall. and Distrib. Syst.*, vol. 32, no. 10, pp. 2557-2570, Oct. 2021. (https://doi.org/10.1109/TPDS.2021.3064709)

[30] O. Kilic, et al., "Overcoming virtualization overheads for large-CPU virtual machines," in *2018 IEEE 26th Int. Symp. MASCOTS*, pp. 369-380, 2018. (https://doi.org/10.1109/MASCOTS.2018.00042)

[31] T. Xing, C. Xiong, C. Ye, Q. Wei, J. Picorel, and A. Barbalace, "Maximizing VMs' IO performance on overcommitted CPUs with fairness," in *Proc. 2023 ACM Symp. Cloud Computing (SoCC '23)*, pp. 93-108, Association for Computing Machinery, New York, NY, USA, 2023. (https://doi.org/10.1145/3620678.3624649)

[32] S. Jung, et al., "Prediction of permissioned blockchain performance for resource scaling configurations," *ICT Express*, vol. 10, no. 6, pp. 1253-1258, 2024. (https://doi.org/10.1016/j.icte.2024.09.003)

[33] *Hyperledger Caliper - hyperledger.github.io*, Retrieved Feb., 20 2025, from https://hyperledger.github.io/caliper/0.6.0.

[34] C. Melo, G. Gonçalves, F. A. Silva, et al., "A comprehensive hyperledger fabric performance evaluation based on resources capacity planning," *Cluster Comput*, vol. 27, pp. 12395-12410, 2024. (https://doi.org/10.1007/s10586-024-04591-4)

[35] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," *2018 IEEE 26th Int. Symp. MASCOTS*, pp. 264-276, Milwaukee, WI, USA, 2018. (https://doi.org/10.1109/MASCOTS.2018.00034)

[36] L. Foschini, A. Gavagna, G. Martuscelli, and R. Montanari, "Hyperledger fabric blockchain: Chaincode performance analysis," *ICC 2020*, pp. 1-6, Dublin, Ireland, 2020. (https://doi.org/10.1109/ICC40277.2020.9149080)

[37] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat, and S. Chatterjee, "Performance characterization of hyperledger fabric," *2018 CVCBT*, pp. 65-74, Zug, Switzerland, 2018. (https://doi.org/10.1109/CVCBT.2018.00013)

[38] J. Lee, Y. Yoo, C. Yoo, and G. Yang, "Predictive placement of geo-distributed blockchain nodes for performance guarantee," *2024 IEEE 17th CLOUD*, pp. 66-68, Shenzhen, China, 2024. (https://doi.org/10.1109/CLOUD62652.2024.00017)

강 윤 경 (Yunkyung Kang)

2023년 2월 : 순천향대학교 컴퓨터소프트웨어공학과 졸업
2023년 9월~현재 : 고려대학교 컴퓨터학과 석사과정
<관심분야> 보안 컨테이너, 가상화 기반 컨테이너 기술, 블록체인 네트워크
[ORCID:0009-0009-2012-1153]

양 경 식 (Gyeongsik Yang)

2015년 2월 : 고려대학교 컴퓨터학과 졸업
2017년 2월 : 고려대학교 컴퓨터학과 석사
2019년 8월 : 고려대학교 컴퓨터학과 박사
2023년 9월~현재 : 고려대학교 컴퓨터학과 조교수
<관심분야> 시스템 소프트웨어, 네트워크 시스템, AI 시스템, 데이터센터 인프라
[ORCID:0000-0003-4560-2972]

최 원 미 (Wonmi Choi)

2021년 2월 : 고려대학교 컴퓨터학과 졸업
2021년 3월~현재 : 고려대학교 컴퓨터학과 석박통합 과정
<관심분야> 컨테이너 가상화, 컨테이너 오케스트레이션, 커널 네트워킹 스택, 연합 학습
[ORCID: 0009-0001-3290-8262]

유 혁 (Chuck Yoo)

1982년 2월 : 서울대학교 전자공학과 졸업
1986년 8월 : University of Michigan At Ann arbor 석사
1990년 8월 : University of Michigan At Ann arbor 박사
1990년 8월~1995년 2월 : Sun Microsystems Lab 연구원
1995년 3월~현재 : 고려대학교 정보대학 교수
<관심분야> 운영체제, 소프트웨어 정의 네트워크, 디지털 헬스
[ORCID:0000-0002-1115-1862]