파일 I/O 시나리오를 통한 컨테이너 런타임별 격리 수준 분석 연구

김미연[1], 최상훈[2], 박기웅[3†]
[1]세종대학교 Syscore Lab. 석사과정
[2]세종대학교 Syscore Lab. 연구교수
[3]세종대학교 정보보호학과 교수
miyeon2002@naver.com, csh0052@gmail.com, woongbak@sejong.ac.kr

A Study on Analyzing Isolation Levels by Container Runtime Through File I/O Scenario

Mi-Yeon Kim[1], Sang-Hoon Choi[2], Ki-Woong Park[3†]
[1,2]Syscore Lab., Sejong University
[3]Dept. of Computer and Information Security, Sejong University

요 약

최근 컨테이너가 클라우드의 핵심 기반 기술로 자리하게 되었다. 컨테이너는 느린 프로비저닝과 과도한 리소스를 사용하는 VM의 단점을 완화하지만, 호스트의 커널을 공유하므로 격리 수준이 낮다는 또 다른 한계가 존재한다. 컨테이너를 실행할 수 있는 런타임에는 여러 종류가 있으며, 다양한 컨테이너의 보안 문제를 해결하기 위해 보안 강화 런타임이 제안되고 있다. 그러나 각 컨테이너 런타임의 격리 수준을 정량적으로 분석한 연구는 충분하지 않은 실정이다. 본 논문에서는 컨테이너 런타임이 디스크 측면에서의 격리를 얼마나 잘 수행하는지를 분석하기 위해 32GB/128GB의 대용량 파일을 생성하여 부하를 유발하고, 부하로 인해 발생하는 디스크 성능의 변화를 정량적으로 제시한다.

1. 서론

최근 컨테이너는 클라우드 기술의 핵심 기반이 되었다 [1]. 커널 기능을 통해 OS 수준의 가상화를 제공하는 컨테이너는 무겁고 느린 VM(Virtual Machine)의 단점을 완화하지만, 호스트의 커널을 공유하기 때문에 낮은 수준의 격리와 넓은 공격 표면이라는 한계를 가지고 있다. 컨테이너의 낮은 격리 수준으로 인한 문제는 클라우드 환경에서 noisy neighbor로 인한 자원 경쟁이 발생할 때 더욱 두드러질 수 있다.

noisy neighbor는 클라우드 환경에서 자원을 과도하게 사용하는 특정 테넌트를 지칭한다. noisy neighbor로 인한 문제는 물리 자원을 공유하는 다른 테넌트나 시스템 전체 성능에 영향을 미칠 수 있고, 공격자가 이를 악용하면, 서비스 거부 공격(DoS, Denial of Service)으로까지 이어질 수 있다 [2, 3].

컨테이너 런타임이란 컨테이너 엔진에 해당하는 소프트웨어 계층을 의미하며, 컨테이너 런타임에는 여러 종류가 있다. 최근에는 컨테이너의 다양한 보안 문제를 완화하기 위해 Google의 gVisor와 같은 보안 강화 런타임도 제안되고 있다 [4]. 그러나 각 런타임의 격리 수준을 정량적으로 분석한 연구는 아직 충분하지 않은 실정이다. 본 논문에서는 런타임별 디스크 격리 수준을 분석하기 위해 시스템 컨테이너인 LXC(LinuX Containers), 대표적인 애플리케이션 컨테이너 런타임인 runc, 보안 강화 런타임인 gVisor에서 대용량의 파일을 생성한다. 이후, 부하를 발생시켰을 때 디스크의 성능이 어떻게 변화하는지를 관측하여 디스크 측면에서의 런타임별 격리 수준을 정량적으로 분석하고자 한다. 파일 생성은 디스크에 직접적인 부하를 일으킬 수 있는 단순한 작업이지만, 이를 악용하여도 탐지나 제제가 쉽지 않기 때문에 현실적인 위협 모델로 작용할 수 있다.

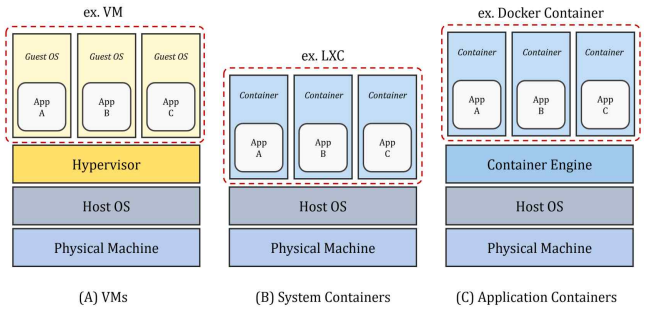
본 논문의 구성은 다음과 같다. 2장에서는 배경지식 및 관련 연구, 3장에서는 컨테이너 기반 멀티테넌시 환경에서 디스크 간섭을 유도하는 접근 방식에 대해 기술한다. 4장에서는 실험 및 평가 결과에 대해 기술하고, 5장에서는 결론 및 향후 연구를 제시한다.

2. 배경지식 및 관련 연구

본 장에서는 실험에 사용된 컨테이너 런타임에 대한 설명과 논문과 관련된 선행 연구를 분석하여 기술한다.

2.1. 컨테이너 런타임

2.1.1 LXC



(그림 1) VM vs. Containers(System/Application) 아키텍처

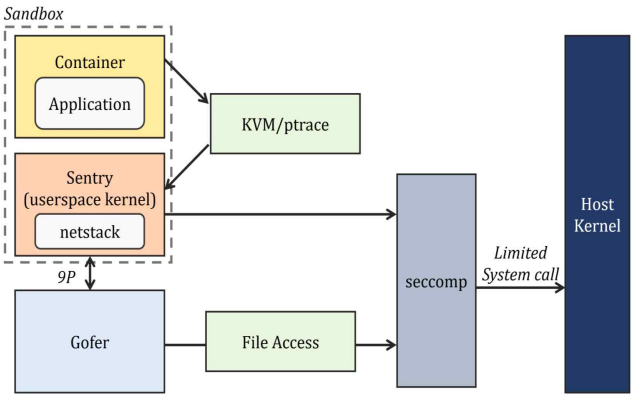
VM은 (그림 1)의 (A)와 같이 Guest OS를 필요로 하기 때문에 속도가 느리고 무겁다는 단점이 있다. 컨테이너는 이러한 단점을 완화하기 위해 등장한 경량화된 가상화 기술이다. 가장 초기의 컨테이너 중 하나인 LXC는 리눅스에 의해 2008년에 개발되었으며, LXC의 아키텍처는 (그림 1)의 (B)와 같다. LXC는 low-level 컨테이너 런타임으로 최소한의 기능만 유지하여 가볍다는 장점이 있지만, 오케스트레이션 등의 기능이 전혀 제공되지 않는다는 한계가 있다 [5].

2.1.2. runc

컨테이너 라이프 사이클을 관리하기 위한 OCI에서는 런타임과 관련된 표준인 runtime-spec을 정의하고 있으며, runc는 OCI의 표준을 준수하는 Docker 컨테이너의 기본 런타임이다 [6]. Docker의 아키텍처는 (그림 1)의 (C)와 같다. runc는 리눅스의 네임스페이스, cgroup, capabilities를 활용하여 컨테이너를 생성하는 기능을 제공하며, 보안 정책을 적용하기 위해 seccomp, SElinux, Apparmor에 의존한다 [7].

2.1.3. gVisor(runsc)

gVisor는 (그림 2)와 같이 Sentry와 Gofer로 구성된 샌드박스 환경을 통해 강화된 보안을 제공하는 컨테이너 런타임이다. 유저스페이스 커널인 Sentry는 애플리케이션이 필요로 하는 커널의 기능들을 구현한다. 애플리케이션이 시스템 콜을 호출하면 Sentry로 리디렉션되기 때문에 대부분의 시스템 콜은 호스트 커널로 넘어가지 않는다. 만약, 파일 열기 등과 같이 파일 시스템에 접근해야 하는 상황이 발생하면, Gofer로 전달하여 간접적으로 처리한다. 이때 Gofer는 9P 프로토콜을 사용하여 소켓이나 공유 메모리 채널을 통해 Sentry와 통신한다.



(그림 2) gVisor 아키텍처

이러한 구성으로 gVisor는 기존 컨테이너 런타임에 비해 보안성을 크게 향상시킨다는 장점이 있지만, 네트워크 스택 및 파일 시스템 처리에서 성능 오버헤드가 증가한다는 한계가 존재한다.

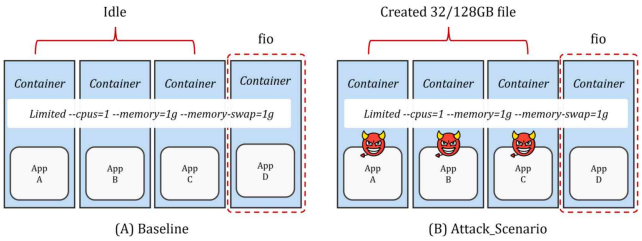
2.2. 관련 연구

Li et al. [8]은 애플리케이션 컨테이너인 Docker와 gVisor, Kata, Qemu 기반의 환경에서 Sequential Read/Write Latency와 Random Read/Write Latency를 통해 디스크 I/O를 평가하였다. 그러나 본 연구에서는 시스템 컨테이너인 LXC를 추가하여 분석 유형을 확대하였다. 또한, IOPS를 추가 지표로 활용하여 보다 종합적인 디스크 I/O 성능을 측정하였다.

Volpert et al. [9]는 gVsior, Kata, podmen 환경에서 CPU, Disk, Memory, Network 성능을 측정하였지만, gVisor에 대한 디스크 성능 결과를 제외(n/a로 작성)하였다. 실험을 위해서는 블록 장치를 컨테이너 내부와 직접 연결하는 기능이 필수적인데 gVisor는 해당 기능을 지원하지 않기 때문이라고 설명했다. 본 연구에서는 이러한 한계를 고려하여 각 런타임으로 실행된 컨테이너의 내부에서 디스크의 성능을 측정하고자 한다.

3. 컨테이너 기반 멀티테넌시 환경에서 디스크 간섭을 유도하는 접근 방식

본 장에서는 멀티테넌시 환경에서 대용량의 파일을 생성함으로써, 디스크 간섭을 유도할 수 있는 방식에 대하여 기술한다. 우리는 본 연구의 실험을 위해 (그림 3)과 같은 컨테이너를 구성하였다. 먼저, 베이스라인을 측정하기 위해 (그림 3)의 (A)와 같이 컨테이너 A, B, C, D를 생성한 후, A, B, C는 유휴 상태로 두고, D에서 디스크의 성능을 측정하였다.



(그림 3) 실험을 위한 컨테이너 구성

이후, (그림 3)의 (B)와 같이 컨테이너 A, B, C에서 각각 32GB, 128GB만큼의 파일을 동시에 생성하였다. 이때 컨테이너 A, B, C에서 발생한 부하로 인해 컨테이너 D의 디스크 성능이 어떻게 변화하는지를 확인하였다.

이 과정은 단일 CPU 코어와 1GB 메모리로 제한된 LXC, runc(using Docker), gVisor(using Docker) 컨테이너에서 수행되었다. 또한, gVisor의 디스크 제한이 공식적으로 지원되지 않아 동일한 조건에서 경향을 파악하기 위해 모든 런타임의 디스크 제한을 제외하였다. 간섭을 유도하기 위한 파일의 용량은 32GB와 128GB로 선택하였는데 32GB 파일은 메모리보다 작은 용량으로써 캐시 효과를 받았을 때의 성능을 측정하기 위함이고, 메모리보다 큰 용량의 128GB 파일은 캐시 효과를 무력화했을 때의 성능을 측정하기 위함이다.

우리는 오차를 최대한 제거하여 객관성을 확보하기 위해 각 세트당 100번의 반복 실험을 수행하였으며, 회차마다 모든 컨테이너를 삭제하고 재생성하였다.

4. 실험 및 평가

본 장에서는 3장에서 언급된 접근 방식을 기반으로 실험을 수행하고, 측정된 디스크 성능을 통해 런타임별 디스크 격리 수준을 평가한다. 측정 지표는 Read/Write IOPS 처리량과 Read/Write Latency로 선택하였다.

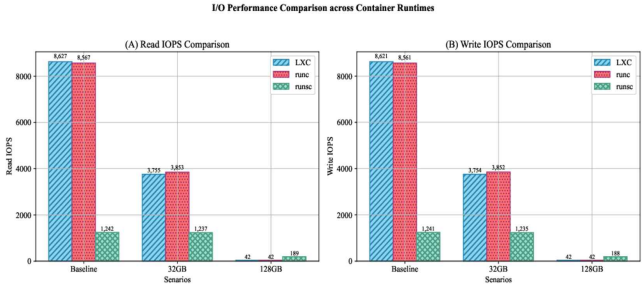
4.1. 실험 환경

본 연구를 위한 실험 환경은 <표 1>과 같다.

<표 1> 실험 환경

항목	Spec/Version
CPU	Intel Xeon Silver 4310 CPU @ 2.10GHz
Memory	64GB
OS	Ubuntu 22.04
Kernel	5.15.0-143-generic
Docker	28.2.2
LXC	5.0.0
fio	3.28

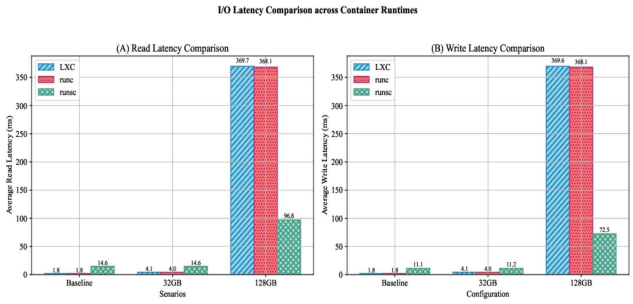
4.2. 평가



(그림 4) 런타임 간 Read/Write IOPS 처리량 비교

(그림 4)는 컨테이너 런타임에 대한 Read/Write IOPS 처리량을 나타낸다. 대부분의 경우, LXC와 runc는 유사한 성능을 보이는 것으로 나타났다. 32GB 시나리오에서는 디스크 성능이 베이스라인 대비 50% 이상 감소했고, 128GB의 경우, 약 99.5% 감소하였다. 반면, gVisor에서 측정된 베이스라인은 LXC와 runc 베이스라인의 약 14% 수준에서 그쳤다.

gVisor는 여러 계층을 통해 강화된 보안을 제공하므로 베이스라인 자체가 낮게 측정되지만, 32GB 시나리오에서의 성능이 베이스라인과 유사한 것을 확인할 수 있다. 이는 LXC와 runc의 경향과 비교했을 때 gVisor가 디스크 격리를 어느 정도 잘 수행한다고 판단할 수 있는 근거가 된다. 그러나 128GB 시나리오에서의 성능은 베이스라인 대비 약 85% 감소하는 것으로 나타났다. 이러한 결과는 보안 강화 런타임인 gVisor조차 디스크 측면에서의 완전한 격리를 제공하지 않음을 나타내며, 일정 크기 이상의 부하가 발생했을 때는 noisy neighbor 문제로 인한 영향을 받을 수 있음을 의미한다.



(그림 5) 런타임 간 Read/Write Latency 비교

(그림 5)는 컨테이너 런타임의 Read/Write Latency를 나타낸다. IOPS 처리량과 마찬가지로 LXC와 runc는 비슷한 양상을 보였다. 32GB 시나리오의 Latency는 베이스라인 대비 약 2배 증가하였고, 128GB의 경우에는 약 200배 이상 증가하였다. gVisor에서 32

GB 시나리오의 경우, 베이스라인과 매우 유사한 Latency가 발생하는 것으로 관측되었지만, 128GB 시나리오에서는 Read/Write Latency가 각각 96.8ms, 72.5ms로 약 6.5배 증가하는 것을 확인할 수 있었다.

우리는 gVisor의 Write Latency보다 Read Latency가 더 높게 측정된 것을 확인할 수 있었는데, 이는 IOPS 처리량에서 Read의 성능이 더 높게 측정되었던 것을 고려했을 때 대비되는 결과이다. 이러한 현상이 관측되는 이유는 gVisor가 유저스페이스에서 시스템 콜을 인터셉트하여 에뮬레이션하는 설계 방식 때문이다. 즉, gVisor 환경에서 발생하는 Write의 경우, 버퍼 처리를 위한 flush를 비동기로 처리하기 때문에 Delay가 발생하지 않는 반면, 읽기 작업에서는 시스템 콜 인터셉트로 인한 오버헤드가 발생하므로 비교적 높은 Latency가 측정된다.

5. 결론

본 논문에서는 컨테이너 런타임별 디스크 격리 수준을 분석하기 위해 대용량의 파일을 생성하여 부하를 발생시킨 후, 부하가 미치는 영향을 확인하였다. 실험 결과, LXC와 runc 환경에서는 시나리오별 지표에서 뚜렷한 성능저하가 관측되었다. gVisor는 32GB 시나리오에서 베이스라인과 유사한 성능을 유지했지만, 128GB 시나리오에서는 명확한 성능저하가 나타났다. 특히, gVisor의 Latency에서는 IOPS 처리량과는 상반된 결과가 도출되었는데 이는 gVisor의 설계 방식과 버퍼 처리 방식에 기인한 것이었다. 우리는 실험을 통해 noisy neighbor로 인한 파일 I/O가 다른 테넌트의 디스크 성능에 영향을 미칠 수 있음을 입증하였다. 또한, 보안 컨테이너 런타임으로 평가되는 gVisor도 디스크 격리를 완전하게 수행하지 못한다는 것을 확인하였다.

향후 연구에서는 정상 행위임에도 성능 간섭을 유발할 수 있는 추가 시나리오를 정립하고, 컨테이너 기반 멀티테넌시 환경에서 성능 간섭을 완화할 수 있는 방안을 연구하고자 한다.

Acknowledgement

본 논문은 과학기술정보통신부의 재원으로 실감콘텐츠핵심기술개발(Project No. RS-2023-00228996, 40%), 한국연구재단(NRF) 중견후속 연구사업(Project No. RS-2023-00208460, 30%), 한국콘텐츠진흥원(KOCCA) 저작권기술 글로벌 인재 양성사업 (Project No. RS-2025-02221620, 30%)의 지원을 받아 수행된 연구임.

참고문헌

[1] Yang, Yutian, et al, "Security challenges in the container cloud", 2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA). IEEE, Atlanta, GA, USA, 2021, p. 137-145.

[2] Shethiya, Aditya S, "Ensuring Optimal Performance in Secure Multi-Tenant Cloud Deployments", Spectrum of Research, 4, 2,2024.

[3] Vajpayee, Prashant, and Gahangir Hossain, "Multi-Tenant Cloud Security-Risk Prediction through Cyber-Value-at-Risk (CVaR)", 2024 12th International Symposium on Digital Forensics and Security (ISDFS). IEEE, San Antonio, Texas, USA, 2024, p. 1-7.

[4] Lee, Kyungwoon, et al, "Impact of secure container runtimes on file i/o performance in edge computing", Applied Sciences, 13, 24, 2023.

[5] Wiedner, Florian, et al, "Control groups added latency in NFVs: An update needed?", 2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE, Dresden, Germany, 2023, p. 40-45.

[6] Cochak, Henrique Zanela, et al, "RunC and Kata runtime using Docker: A network perspective comparison", 2021 IEEE Latin-American Conference on Communications (LATINCOM). IEEE, Santo Domingo, Dominican Republic, 2021, p. 1-6.

[7] Kumar, Rakesh, and B, Thangaraju. "Performance analysis between runc and kata container runtime", 2020 ieee international conference on electronics, computing and communication technologies (conecct). IEEE, Bangalore, India, 2020, p. 1-4.

[8] Li, Guoqing, et al, "Comparative Performance Study of Lightweight Hypervisors Used in Container Environment", CLOSER, Bozen-Bolzano, Italy,2021, p. 215-223.

[9] Volpert, Simon, et al, "An Empirical Analysis of Common OCI Runtimes' Performance Isolation Capabilities", Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering, London United Kingdom, 2024, p. 60-70.