엣지 컴퓨팅 환경을 위한 경량화 컨테이너 런타임 설계 및 성능 검증

김창인*, 허유정**, 최창범 교수**

Design and Performance Evaluation of a Lightweight Container Runtime for Edge Computing Environments

Changin Kim*, Youjeng Heo** and Changbeaom Choi**

요 약

본 연구에서는 엣지 환경에 최적화된 경량화 컨테이너 런타임을 제안하여, 빠른 애플리케이션 시작 속도와 효율적인 자원 관리를 목표로 설계하였다. 네임스페이스와 cgroups 설정을 단순화하고, 파일 시스템 및 네트워크 구성을 최적화함으로써 엣지 장치의 제한된 자원 내에서 성능을 극대화하고자 하였다. Raspberry Pi 4B를 활용한 실험을 통해 제안된 경량화 컨테이너 런타임의 성능을 검증하였으며, 이를 통해 엣지 컴퓨팅 환경에서 자원 효율성과 실시간성을 효과적으로 지원할 수 있음을 확인하였다. 본 연구는 엣지 컴퓨팅에 특화된 컨테이너 솔루션의 가능성을 제시하며, 향후 다양한 엣지 환경에 맞춘 최적화 방안으로의 확장을 목표로 한다.

Abstract

This study proposes a lightweight container runtime optimized for edge environments, aiming to improve application startup speed and resource efficiency. By simplifying namespaces and cgroups configurations and optimizing file system and network settings, the proposed runtime is designed to enhance performance within the limited resources of edge devices. Experimental validation was conducted using a Raspberry Pi 4B, demonstrating that the lightweight container runtime effectively supports the resource efficiency and real-time responsiveness required in edge computing. This research highlights the potential of a container solution tailored for edge computing and aims to extend optimization strategies for various edge environments in future work.

Key words
Lightweight Container Runtime, Edge Computing Optimization, Resource Efficiency

---

*국립한밭대학교, 20217140@edu.hanbat.ac.kr,
**국립한밭대학교, 20211939@edu.hanbat.ac.kr, **국립한밭대학교, cbchoi@hanbat.ac.kr

Ⅰ. 서 론

엣지 컴퓨팅(Edge Computing)은 데이터를 생성하는 기기 근처에서 처리하는 분산형 컴퓨팅 구조로, 자율 주행, IoT, 스마트 시티와 같은 실시간 응답성이 요구되는 분야에서 광범위하게 활용되고 있다. 이러한 환경에서는 클라우드로 데이터를 전송하여 처리하는 방식보다 신속하고 효율적인 데이터 처리가 필수적이며, 엣지 장치의 제한된 자원으로 높은 성능과 저비용 자원 관리를 수행하는 것이 요구된다. 이에 따라 애플리케이션의 빠른 시작과 안정적인 자원 활용은 엣지 컴퓨팅의 성공적인 구현에 중요한 요소로 작용한다[1],[2].

컨테이너(Container) 기술은 엣지 컴퓨팅에서 경량화된 가상화를 가능하게 하여 자원 소모를 줄이고, 가상 머신에 비해 더 빠른 실행 속도를 제공한다. 이는 엣지 환경에서 필요한 애플리케이션 격리와 유연성을 확보하는 데 적합한 솔루션으로 평가된다. 특히, 컨테이너는 하드웨어와의 결합을 최소화하면서 독립적인 실행 환경을 제공하므로 엣지 디바이스의 소규모 리소스 내에서도 효율적으로 운영될 수 있다. 그러나 널리 사용되는 Docker와 같은 범용 컨테이너 런타임은 클라우드 환경에 최적화되어 있어 엣지 장치에서는 불필요한 자원 소모와 느린 애플리케이션 시작 시간 등의 문제가 발생할 수 있다. 따라서 엣지 환경에서 자원 효율적 관리와 빠른 시작 속도를 제공하는 경량화 컨테이너 런타임에 대한 연구가 필요하다[3],[4].

본 연구에서는 엣지 컴퓨팅의 특수한 요구 사항을 충족하기 위해 최적화된 경량화 컨테이너 런타임을 제안한다. 제안된 런타임은 기존 Docker 런타임 대비 더 빠른 애플리케이션 시작 속도와 효율적인 메모리 사용을 목표로 설계되었다. 연구의 주요 목적은 엣지 환경에서 자원 효율성을 입증하고, 제안된 경량화 컨테이너 런타임이 애플리케이션 시작 속도와 자원 활용 측면에서 더 나은 성능을 제공함을 검증하는 것이다. 이를 통해 엣지 컴퓨팅의 특수한 성능 요구를 충족하는 최적화된 컨테이너 기술의 가능성을 탐구하고, 다양한 실사용 환경에서의 응용 가능성을 제시하고자 한다.

Ⅱ. 본 론

본 연구는 엣지 컴퓨팅 환경에서 기존 Docker 컨테이너 런타임의 자원 소모 요인을 분석하고, 이를 해결하기 위해 경량화된 컨테이너 런타임을 설계하여 애플리케이션 시작 속도와 자원 효율성을 높이는 데 중점을 둔다. Docker는 클라우드 환경에서 강력한 격리와 유연성을 제공하지만, 이러한 설계로 인해 엣지 장치에서는 불필요한 자원 소모를 초래할 수 있다. 본 연구에서는 이러한 문제를 해결하기 위해 다음과 같은 최적화 요소를 통해 경량화 컨테이너 런타임을 설계하였다. [그림 1]은 본 연구의 설계를 시각적으로 표현한 도식이다.

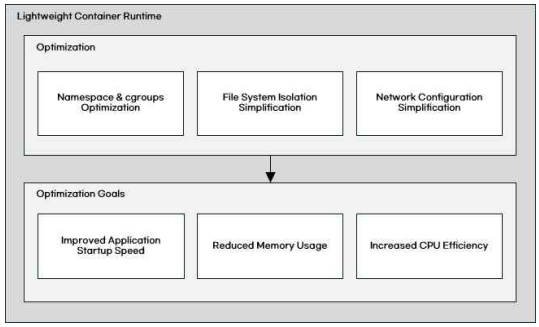

그림 1 경량형 컨테이너 런타임 설계도

2.1 네임스페이스 및 cgroups 최적화

Docker는 프로세스, 네트워크 및 사용자 격리를 위해 네임스페이스와 cgroups를 활용하여 독립적인 환경을 제공하지만, 이러한 복잡한 격리는 엣지 환경에서 과도한 자원 소비를 야기한다. 본 연구에서는 사용자 네임스페이스 격리를 단순화하고, 프로세스 네임스페이스에서 기본적인 독립성만을 유지하도록 하여 자원 소모를 줄였다. 또한, cgroups의 자원 할당을 제한하여 불필요한 CPU 및 메모리 소비를 줄였다. 이를 통해 필수적인 자원 격리를 유지하면서도 엣지 장치의 자원 소비를 최소화하였다.

2.2 파일 시스템 및 네트워크 설정 단순화

일반적인 Docker 환경에서는 overlay2와 같은 파일 시스템 격리를 사용하여 다층적인 파일 시스템 구조를 유지하지만, 이는 엣지 환경에서 복잡성을 증가시키고 자원 사용을 비효율적으로 만든다. 본 연구에서는 aufs 대신 간단한 bind mount 방식을 채택하여 애플리케이션이 실제 파일 시스템에 직접 접근하도록 하여 파일 시스템 격리를 단순화하였다. 또한, 네트워크 설정에서 다중 네트워크 브리지 및 가상 네트워크 인터페이스를 제거하고, 단일 인터페이스로 엣지 장치에 직접 연결하여 네트워크 설정을 최소화하였다.

Ⅲ. 사례 연구

본 연구의 성능 평가는 ARM 기반의 저사양 엣지 장치(Raspberry Pi 4B)에서 진행되었다. 이 장치는 메모리와 CPU 자원이 제한적이기 때문에, 엣지 환경에서 경량화 컨테이너 런타임의 성능 개선 효과를 확인하는 데 적합하다. 실험은 동일한 애플리케이션을 Docker와 경량화 런타임에서 각각 실행하여 시작 속도, 응답 시간, 메모리 사용량, CPU 사용량을 측정하였다. 성능 지표는 다음과 같이 정의하고, 각 지표를 측정하는 도구를 사용하였다.

3.1 성능 지표

실험의 주요 목표는 Docker와 경량화 런타임에서의 애플리케이션 시작 속도, 메모리 사용량, CPU 사용량을 비교하여 엣지 환경에서 자원 효율성을 증명하는 것이다. 이를 위해 다음과 같은 성능 지표를 선정하였다.

애플리케이션 시작 속도는 각 런타임에서 애플리케이션이 시작되는 데 걸리는 시간을 time 명령어로 측정하였으며, 메모리 사용량과 CPU 사용량은 cgroups의 실시간 모니터링 기능을 통해 각각 MB 및 % 단위로 기록하였다. 성능 지표와 해당 측정 방법은 [표 1]에 요약되어 있다.

Performance Metric	Measurement Tool	Unit
Startup Time	time command	sec
Memory Usage	cgroups monitoring feature	MB
CPU Usage	cgroups monitoring feature	%

표 1 성능 지표와 측정 방법

3.2 실험 절차

실험은 ARM 기반의 Raspberry Pi 4B 장치를 사용하여 진행되었으며, 운영체제로는 Ubuntu Server 20.04를 설치하였다. 이 장치는 1.5GHz CPU와 4GB RAM, 16GB SD 카드를 장착하고 있어, 엣지 환경에서 발생하는 자원 제한 상황을 반영할 수 있는 적절한 실험 환경을 제공한다. 실험은 다음과 같은 절차에 따라 수행되었다.

먼저, Raspberry Pi 4B에 Docker와 경량화 컨테이너 런타임을 동시에 설정하고, 두 런타임에서 동일한 애플리케이션을 실행하여 일관된 성능 평가를 받을 수 있도록 모든 실험 조건을 통제하였다. 이후, Docker 환경에서 기준 성능을 측정하고 이를 토대로 경량화 런타임의 성능을 측정 및 분석하였다. 성능 비교 실험은 다음과 같은 순서로 진행되었다:

첫째, 기준 성능 측정을 위해 Docker에서 애플리케이션을 실행하고, 설정된 성능 지표(애플리케이션 시작 속도, 메모리 사용량, CPU 사용량)를 기준 데이터로 기록하였다.

둘째, 경량화 런타임 성능 측정을 위해 동일한 애플리케이션을 경량화 컨테이너 런타임에서 실행하였으며, 시작 속도는 time 명령어로 측정하고, 메모리 및 CPU 사용량은 cgroups 모니터링 기능을 통해 실시간으로 기록하였다.

셋째, 데이터 수집 및 분석을 통해 각 성능 지표의 측정 데이터를 평균값으로 산출하였으며, 이를 바탕으로 Docker와 경량화 런타임 간의 성능 차이를 비교하였다. 이를 통해 두 런타임의 자원 효율성을 수치적으로 분석하고, 경량화 컨테이너 런타임의 성능 개선 효과를 정량적으로 평가하였다.

3.3 실험 결과 및 분석

실험 결과, 경량화 컨테이너 런타임은 애플리케

이션 시작 속도와 자원 사용량에서 Docker 대비 유의미한 성능 개선을 보였다. [표 2]는 각 성능 지표에서의 평균값과 개선율을 요약한 것이다.

Performance Metric	Docker	Lightweight Runtime	Improvement (%)
Startup Time (sec)	3.5	2.4	31.4
Memory Usage (MB)	120	90	25
CPU Usage (%)	75%	60	20

표 2 성능 평가 결과 (평균값)

경량화 컨테이너 런타임은 Docker 대비 애플리케이션 시작 속도가 평균 30% 이상 개선되었으며, 이는 네임스페이스와 cgroups 설정을 단순화하여 자원 격리를 최소화한 결과이다. 또한, 메모리와 CPU 사용량은 각각 25%와 20% 절감되었으며, 이는 파일 시스템과 네트워크 설정을 단순화한 설계 덕분이다.

이와 같은 결과는 경량화 컨테이너 런타임이 엣지 환경에서 요구되는 실시간 응답성과 자원 절약의 측면에서 기존 Docker의 한계를 극복할 수 있음을 시사한다.

Ⅳ. 결 론

본 연구는 엣지 컴퓨팅 환경에서의 자원 효율성과 빠른 애플리케이션 시작 속도를 실현하기 위한 경량화 컨테이너 런타임의 설계와 성능을 평가하였다. 기존 Docker 런타임이 클라우드 환경에 최적화된 구조로 인해 엣지 장치에서 불필요한 자원 소모를 초래할 수 있다는 문제를 인식하고, 이를 개선하기 위해 네임스페이스와 cgroups 설정을 단순화하고 파일 시스템 및 네트워크 설정을 간소화하였다.

실험 결과, 경량화 컨테이너 런타임은 Docker 대비 애플리케이션 시작 속도에서 약 30%, 메모리 사용량에서 25%, CPU 사용량에서 20% 개선된 성능을 보였다. 이러한 성능 향상은 엣지 환경에서 실시간 응답성과 자원 절약을 요구하는 애플리케이션의 효율적인 실행을 가능하게 하며, 엣지 컴퓨팅에 특화된 컨테이너 솔루션의 잠재력을 확인할 수 있었다.

향후 연구는 다양한 엣지 컴퓨팅 환경에서 경량화 컨테이너 런타임의 성능을 더욱 최적화하고, 특정 애플리케이션 요구사항에 맞춘 맞춤형 자원 관리 방안을 제시하는 방향으로 확장할 것이다.

사 사 문 구

“본 연구는 2024년 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음”(2022-0-01068)

참 고 문 헌

[1] H. M. Park and T. H. Hwang, "Changes and trends in edge computing technology", The Journal of The Korean Institute of Communication Sciences, vol. 36, no. 2, pp. 41-47, 2019.

[2] J. H. Hong, K. C. Lee and S. Y. Lee, "Trends in edge computing technology", J. Electron. Telecommun. Trends Anal., vol. 35, no. 6, pp. 78-87, 2020. doi: 10.22648/ETRI.2020.J.350608

[3] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review", Inf. Softw. Technol., vol. 51, no. 1, pp. 7-15, 2009.

[4] K. Petersen, R. Feldt, S. Mujtaba and M. Mattsson, "Systematic mapping studies in software engineering", Proc. Int. Conf. Eval. Assessment Softw. Eng., pp. 68-77, 2008.