Java 기반 리소스 수집 라이브러리(OSHI)를 활용한 컨테이너 자원 모니터링 시스템 설계에 관한 연구

김채은, 이혜빈, 구성희, 차예원, 이장수*, 김동균

경북대학교 컴퓨터학부, *인터데이터

chaeeun7478@naver.com, dlgpqls1367@naver.com, elev082@naver.com,

yewon51317@naver.com, jslee@interdatalabs.com*, dongkyun@knu.ac.kr

A Study on the Design of a Containe Resource Monitoring System Using the Java-Based Resource Collection Libaray (OSHI)

Chaeeun Kim, Hyebin Lee, Seonghee Gu, Yewon Cha, Jangsoo Lee*, Dongkyun Kim

Kyungpook National University, *InterData

요 약

본 논문은 다양한 운영체제와 컨테이너 환경에서도 일관된 리소스 모니터링을 제공하기 위해, Java 기반 OSHI 라이브러리와 Apache Kafka를 결합한 시스템을 설계하였다. 보안성이 요구되는 폐쇄망에서도 대용량 데이터를 안정적으로 수집·전송하며, 실시간 시각화 및 경고 체계를 제공한다.

Ⅰ. 서 론

현대 IT 인프라는 모놀리식 서버 중심 구조를 넘어, 컨테이너 기반의 분산 하이브리드 클라우드 아키텍처로 재편되었다. 이러한 복잡한 시스템 환경에서는 리소스 상태를 세밀하게 추적하고, 혼재된 여러 플랫폼 및 운영체제 간의 일관된 관찰을 요구한다[1]. 특히 대규모 처리 환경에서는 높은 신뢰성과 확장성 또한 요구된다.

따라서 분산 환경 전반의 리소스를 효율적으로 관리하고, 수집된 대용량 데이터를 실시간으로 안전하게 처리·분석할 수 있는 인프라가 필수적이다. 그러나 기존의 HTTP 기반 모니터링은 폐쇄형 사내망이나 대규모 트래픽 환경에서 보안성과 확장성에 한계를 드러낸다[2]. 나아가, 각 호스트·컨테이너가 HTTP 수집 포트를 노출하기 어려운 환경에서는, 대상 시스템이 중앙 서버로 직접 데이터를 전송하는 구조가 요구되며, 이질적인 시스템 전반에서 일관된 모니터링을 필요로 한다.

이에 본 논문에서는 OSHI 및 Kafka 기반 모니터링 시스템을 제안한다. OSHI를 사용해 다양한 플랫폼에서의 일관된 리소스 수집을 진행하고, Kafka를 활용하여 대용량 데이터 전송 및 처리를 보장함으로써[3] 기존의 HTTP 방식 대비 높은 보안성과 대용량 데이터 처리 성능을 확보하는 방식을 제안한다. 나아가, 이러한 아키텍처를 기반으로, 실시간 데이터 처리 및 경고 체계를 갖춘 시스템 설계 확장을 기대해볼 수 있다.

Ⅱ. 본 론

OSHI(Operating System and Hardware Information)는 Java 기반의 JNA(Java Native Access) 라이브러리로, 별도의 네이티브 설치 없이 다양한 운영체제(Windows, macOS, Linux, Unix 등)에서 시스템 및 하드웨어 정보를 손쉽게 수집할 수 있다. 또한, CPU, 메모리, 디스크, 네트워크 등 주요 자원을 실시간으로 모니터링할 수 있으며, Java 환경과의 높은 통합성 덕분에 컨테이너 기반 시스템이나 하이브리드 클라우드 환경에서도 일관된 리소스 수집이 가능하다.

컨테이너는 경량화된 가상화 기술로 신속한 배포와 확장성을 제공하지만, 리소스 사용량을 제한하거나 모니터링하지 않으면 성능 저하나 장애가 발생할 수 있다. 따라서, 각 컨테이너의 CPU, 메모리, 네트워크, 디스크 사용량 등 주요 리소스의 실시간 모니터링이 필수적이다. Docker의 docker stats 명령 등은 운영체제별 차이로 인해 통합 관리에 한계가 있으나, OSHI를 활용하면 다양한 운영체제에서 일관된 방식으로 컨테이너 및 호스트 자원 정보를 수집할 수 있다.

분류	설명	단위	OSHI 메서드/필드
CPU	CPU 사용률	%	CentralProcessor.getSystemCpuLoadBetweenTicks()
메모리	총 메모리 크기	Byte	GlobalMemory.getTotal()
	사용 가능한 메모리 크기	Byte	GlobalMemory.getAvailable()
디스크	전체 디스크 용량	Byte	OSFileStore.getTotalSpace()
	사용 가능한 디스크 용량	Byte	OSFileStore.getUsableSpace()
	디스크에서 읽은 총 데이터량	Byte	HWDiskStore.getReadBytes()
	디스크에 쓴 총 데이터량	Byte	HWDiskStore.getWriteBytes()
네트워크	네트워크 인터페이스 이름	String	NetworkIF.getName()
	네트워크 속도	bps	NetworkIF.getSpeed()
	받은 데이터량	Byte	NetworkIF.getBytesRecv()
	보낸 데이터량	Byte	NetworkIF.getBytesSent()

표 1 OSHI를 통해 수집하는 데이터 종류

보안 제약이 심한 폐쇄망에서는 HTTP ‘pull’ 기반 모니터링의 보안 및 확장성의 한계를 극복하기 위해 Apache Kafka를 활용한 ‘push’ 방식을 도입한다[4]. 에이전트 모듈이 OSHI로부터 수집한 메트릭을 이진 프로토콜로 Kafka에 전송하며, 전송 실패 시 로컬 디스크 큐잉을 통해 무손실 복구를 보장한다. 이러한 구조는 대용량 데이터를 안정적으로 처리하는 동시에 네트워크 단절이나 브로커 장애 시에도 데이터를 안전하게 유지할 수 있다.

본 논문에서 제안하는 시스템은 OSHI를 활용하여 주기적으로 컨테이너와 호스트의 리소스 사용량을 수집한다. 수집한 데이터를 실시간으로 모니터링 컨테이너에 전달하며, 이 컨테이너는 수집된 정보를 Apache Kafka로 전송한다. Kafka는 대용량의 실시간 데이터를 안전하게 처리할 수 있고, HTTP가 제한된 환경에서도 보안성을 유지할 수 있다. 수집된 데이터는 대시보드에 실시간 시각화되며, 사용자가 설정한 임계값을 초과할 경우, 프론트엔드에서 즉시 경고를 제공한다.

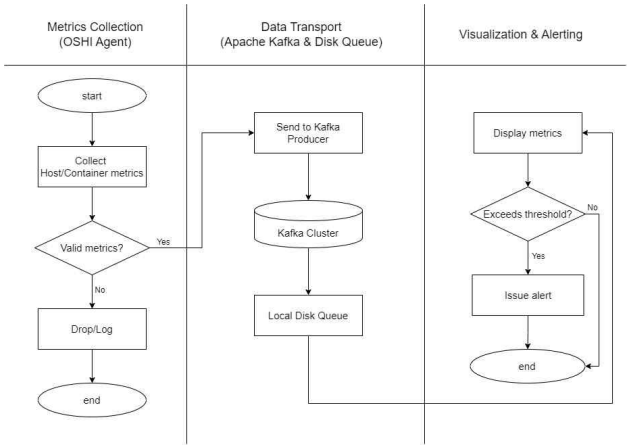

그림 1 OSHI와 Apache Kafka를 도입한 모니터링 시스템의 흐름도

OSHI는 오픈소스이며, Maven/Gradle 등과 쉽게 통합할 수 있다. 또한, 다양한 운영체제와 컨테이너 플랫폼에서 활용할 수 있어, 이기종 환경에서도 일관된 모니터링이 가능하다. 본 시스템은 기존의 HTTP 기반의 오픈소스 모니터링 도구를 대체하여 기업 내부 네트워크 전용의 Kafka 기반 시스템을 구축함으로써 보안성 확보가 가능할 것으로 기대한다.

Ⅲ. 결 론

본 논문에서는 Java 기반 OSHI 라이브러리와 Apache Kafka를 결합한 컨테이너 자원 모니터링 시스템에 관한 연구에 대해 살펴보았다. OSHI를 통해 Docker, Kubernetes 같은 여러 컨테이너 플랫폼에서도 일관된 방식으로 자원 상태를 파악할 수 있어, 기존 운영체제별로 달랐던 docker stats 명령의 한계를 극복하였다. OSHI가 제공하는 메트릭 수집 오버헤드는 기존 라이브러리 대비 경량화가 가능하며, Maven/Gradle과의 손쉬운 통합으로 개발·배포 과정에서도 별도의 네이티브 설치 없이 즉시 적용할 수 있다. 또한, 실시간 시각화 대시보드를 통해 임계값 기반 경고를 제공함으로써 운영자가 즉각적으로 이상 상태를 파악하고 대응할 수 있어, 운영 효율성이 크게 향상될 것으로 기대된다.

향후, 수집된 메트릭과 경고 이력을 기반으로 AI 기법을 연동하여 이상 징후 자동 탐지 및 장애 예측 기능을 추가할 수 있다. 아울러, 엣지 컴퓨팅 환경에 최적화된 경량 모니터링 에이전트와 동적 계측·샘플링 전략을 구현하여, 대규모 분산 시스템에서도 실시간성을 유지하는 차세대 모니터링 프레임워크로 발전시키고자 한다.

ACKNOWLEDGMENT

* 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학 사업의 연구결과로 수행되었음. (2021-0-01082)

참 고 문 헌

[1] 이현민, 김현우, “멀티테넌트 환경에서 테넌트별 리소스 미터링 및 모니터링 시스템 구현,” 한국통신학회 종합 학술 발표회 논문집(동계), 2015, pp.541-542.

[2] Henrik Rexed, “What is Prometheus and Where Does It Fail for Enterprises?,” Medium, 2020.

[3] Jap Kreps, Neha Narkhede, and Jun Rao, “Kafka: a Distributed Messaging System for Log Processing,” NetDB, 2011.

[4] M. Park, H.Choi, “보안 제약 환경에서의 푸시 기반 모니터링 아키텍처 설계,” 인공지능연구학회지, vol. 12, no. 1, 2024.