2025년도 한국통신학회 추계종합학술발표회

DPU 기반 투명화된 S3 트랜잭션 모니터링 오프로딩 아키텍처

박상준, 김종원*
광주과학기술원 AI융합학과
sjoonpark@gm.gist.ac.kr, *jongwon@gist.ac.kr

DPU-Based Transparent S3 Transaction Monitoring Offloading Architecture

SangJoon Park, JongWon Kim*
Department of AI Convergence, Gwangju Institute of Science and Technology (GIST)

요 약

현대 클라우드 환경에서 객체 스토리지는 AI 학습 데이터와 빅데이터 분석의 핵심 인프라로, 페타바이트 규모의 트래픽을 처리한다. 이러한 환경에서 트랜잭션 레벨 모니터링은 보안 감사와 성능 최적화를 위해 필수적이다. 그러나 전통적인 호스트 기반 패킷 캡처 모니터링은 상당한 CPU 오버헤드를 유발하며, 호스트 운영체제 공격 시 감사 로그가 조작될 수 있는 취약점이 존재한다. 본 연구는 NVIDIA BlueField-2 DPU의 하드웨어 가속 기능을 활용하여 네트워크 계층에서 S3 트래픽을 캡처하고 트랜잭션 메타데이터를 전처리하여 호스트로부터 완전히 오프로딩하는 아키텍처를 제안한다. eSwitch 기반 하드웨어 미러링, 플로우 분류, DPDK 기반 TCP 재조립을 통해 HTTP 헤더만을 추출하여 중앙 서버로 전송한다. 25GbE 네트워크 환경에서 수행한 실험 결과, 제안 방식은 극심한 성능 저하를 유발하는 호스트 기반 모니터링과 달리 호스트 CPU에 거의 부하를 주지 않으면서, 인증 정보, 객체 경로, 데이터 해시값 등 트랜잭션 분석에 필요한 핵심 메타데이터를 성공적으로 수집하였다.

Ⅰ. 서 론

현대 클라우드 인프라에서 객체 스토리지 시스템은 AI 학습 데이터 저장, 빅데이터 분석의 핵심 요소로 자리잡았다. MinIO, AWS S3와 같은 솔루션은 페타바이트 규모의 데이터를 처리하며 기하급수적으로 증가하는 트래픽을 생성한다.[1] 이러한 환경에서 트랜잭션 레벨의 모니터링은 보안 감사, 성능 최적화, 규정 준수를 위해 필수적이다. 누가 어떤 객체에 접근하고 어떤 작업을 수행했는지에 대한 가시성은 보안 위협 탐지의 기반이 된다.[2]

그러나 전통적인 호스트 기반 패킷 캡처 모니터링은 상당한 CPU 자원을 소비하여 스토리지 노드의 성능을 저하시키며, 이는 고속 네트워크 환경에서 더욱 심각해진다. 또한 호스트, 운영체제에서 실행되는 모니터링 소프트웨어는 루트킷이나 악성코드에 의해 무력화될 수 있어, 공격자가 감사 로그를 조작하여 보안 이벤트 감지를 우회할 수 있다는 취약점이 존재한다.[3]

본 연구는 기존의 모니터링 방식의 한계를 극복하기 위해, BlueField-2 DPU[4]의 하드웨어 가속 기능을 체계적으로 활용하여 네트워크 계층에서 S3 트래픽의 캡처하고 트랜잭션 메타데이터를 전처리하여 호스트로부터 완전히 오프로딩하는 아키텍처를 제시한다. 제안하는 아키텍처에서 DPU는 트랜잭션 레벨의 메타데이터를 추출하여 중앙 분석 서버로 전달하며, 이를 통해 호스트 성능 영향 없이 투명한 트래픽 가시성을 확보한다.

Ⅱ. 제안 아키텍처

제안하는 아키텍처는 그림 1과 같이 DPU 기반의 투명 모니터링 구조로 구성된다. 호스트로 향하는 S3 트래픽은 원본 경로를 그대로 유지하고, eSwitch에서 하드웨어 미러링된 복제본만 DPU로 전달되어 모니터링을 수행한다. 이 방식은 OVS-Kernel/TC 규칙을 ASAP² 하드웨어로 오프로딩해 처리하므로, 데이터 경로 지연이나 호스트 CPU 부하 없이 동작한다. 원본 패킷은 목적지(MinIO 서버)로 그대로 가고, 복제본만 DPU 내부에서 처리된다. DPU 내부 처리는 네 단계로 이뤄진다.

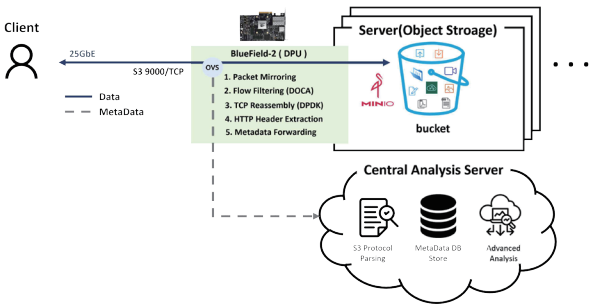

그림 1 DPU 기반 S3 트래픽 모니터링 아키텍처

1) 하드웨어 미러링(OVS-Kernel/TC + ASAP²): eSwitch가 OVS/TC 플로우를 하드웨어로 오프로딩(ASAP²) 하여 패킷을 복제한다. 원본 트래픽은 라인레이트로 포워딩되고, 복제본만 DPU의 파이프라인으로 전달된다. 이 동작은 eSwitch 하드웨어 레벨에서 수행되므로, ARM 코어의 개입과 CPU 오버헤드를 최소화한다.

2) 플로우 분류(DOCA Flow 오프로딩): 미러링된 트래픽을 5-tuple 등 헤더 필드로 하드웨어 플로우 테이블에서 빠르게 분류·스티어링한다. 이로써 CPU 오버헤드 없이 S3 요청/응답의 양방향(클라이언트↔서버)을 모두 캡처하도록 송·수신 플로우를 함께 잡고, 이후 큐 분배·필터링 등 프로그래밍적 후속 처리를 적용한다.

3) 헤더 우선 파싱 (DPDK 기반 TCP 재조립 + HTTP 헤더 추출): 분류된 패킷은 TCP 시퀀스를 기준으로 헤더 부분까지 재조립하며, HTTP 헤더 경계 검출을 통해 헤더만을 추출한다. 바디 데이터는 재조립 대상에서 제외하여 처리 오버헤드를 최소화한다. 추출된 헤더에는 인증 정보(Authorization), 데이터 크기(Content-Length), 무결성 해시값(ETag), 전송 패턴(메서드/URI 조합) 등 트랜잭션 식별 및 분석에 필요한 핵심 메타데이터들이 포함되어 있다.

4) 헤더 전달 : 추출된 HTTP 헤더는 TCP 소켓을 통해 중앙 서버로 전송

된다. 이를 통해 DPU는 호스트에 영향을 주지 않고 S3 트랜잭션 메타데이터를 중앙 집중식으로 수집할 수 있다.

Ⅲ. 실험 및 평가

3.1 실험 환경

실험은 AMD EPYC 7513 32코어 프로세서(64 스레드), 128GB 메모리를 탑재한 호스트 노드와 NVIDIA BlueField-2 DPU(ConnectX-6)를 25GbE 네트워크로 연결한 환경에서 수행되었다. 단일 노드에 MinIO 서버를 구성하고 Warp 클라이언트로 트래픽을 생성하여 모니터링 오버헤드를 측정하였다.

3.2 실험 시나리오

실제 S3 서비스 운영 패턴을 반영하기 위해 Warp 벤치마크 도구를 활용한 2단계 워크로드를 설계하였다. 1단계는 초기 데이터 적재를 모사한 워밍업(5분, PUT 중심), 2단계는 일반적인 운영 상황을 재현한 혼합 워크로드(10분, GET 80% / PUT 15% / DELETE 5%)로 구성하였다. 이는 클라우드 스토리지의 전형적인 사용 패턴인 초기 대량 업로드 후 읽기 중심 운영을 반영한다. 본 실험은 HTTP/1.1 평문 트래픽을 대상으로 하며, TLS 암호화 환경 및 HTTP/2, 멀티파트 업로드는 고려하지 않았다.

3.3 비교 대상

모니터링 방식에 따른 성능 영향을 정량화하기 위해 세 가지 구성을 비교하였다.

- **Baseline:** 모니터링 없이 순수 워크로드만 실행
- **Host Monitoring:** tshark를 통한 호스트 기반 트래픽 모니터링
- **DPU Offloading**: 제안하는 DPU 기반 오프로딩 모니터링

호스트 모니터링 시 tshark를 사용하여 S3 트래픽의 HTTP 요청 메소드, URI, 응답 코드와 같은 특정 필드를 추출하도록 구성하였다. 이는 단순 패킷 캡처를 넘어 호스트 CPU에서 직접 L7 메타데이터를 파싱하는 실제적인 분석 오버헤드를 측정하기 위함이다. 각 구성에서 CPU 사용률, 메모리 사용량, 네트워크 처리량을 수집하였으며, 실험의 신뢰성 확보를 위해 동일 조건에서 5회 반복 수행 후 평균값을 산출하였다.

3.4 실험 결과

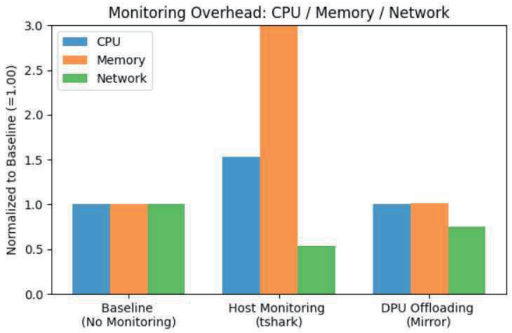

그림 2 모니터링 구성별 호스트 자원 사용률

그림 2는 세 가지 구성에서 측정한 호스트 CPU 사용률, 메모리 사용량, 네트워크 처리량을 baseline 대비 정규화한 결과를 보여준다.

tshark를 이용한 Host Monitoring은 L7 파싱으로 인해 CPU 사용량이 53% 증가하고 네트워크 처리량은 47% 급감하는 등 호스트에 성능 부하를 주었다. 특히 메모리 사용량은 3배 이상 증가하여 자원 고갈의 위험을 보였다. 반면, DPU Offloading은 호스트의 CPU와 메모리 자원을 baseline 수준으로 완벽히 보존하였다. 네트워크 처리량이 약 25% 감소했으나, 이는 DPU 내부 처리로 인한 미세 지연(latency)이 제한된 플로우 벤치마크 환경에서 증폭되었기 때문으로 분석된다. 그럼에도 불구하고, 호스트의 핵심 자원을 전혀 소모하지 않고 L7 모니터링을 수행했다는 점에서 제안 아키텍처의 효율성은 명확히 입증된다.

```
[Request] Client -> MinIO Server
GET /sample/?location= HTTP/1.1
Authorization: AWS4-HMAC-SHA256 Credential=admin/20250921/...
X-Amz-Date: 20250921T063923Z

HEAD /sample/ HTTP/1.1
Authorization: AWS4-HMAC-SHA256 Credential=admin/20250921/...
X-Amz-Content-Sha256: e3b0c44298fc7ae41...

PUT /sample/payload_sample.json HTTP/1.1
Content-Length: 2855
Content-Type: application/json
X-Amz-Content-Sha256: STREAMING-AWS4...
X-Amz-Decoded-Content-Length: 2681
------------------------------------------------------------
[Response] MinIO Server -> Client
HTTP/1.1 200 OK
Content-Length: 128
X-Amz-Request-Id: 186738EB8439F90A
Server: MinIO

HTTP/1.1 404 Not Found
<Code>ObjectLockConfigurationNotFoundError</Code>
X-Amz-Request-Id: 186738EB844C9A63

HTTP/1.1 200 OK
ETag: "e5387c7bc1c314e5cb0e4071a21c1b5e"
Content-Length: 0
```

그림 3 S3 트랜잭션 메타데이터 예시

그림 3은 제안한 아키텍처를 통해 추출된 S3 트랜잭션 메타데이터의 일부 예시를 보여준다. PUT, GET, HEAD, DELETE 등 다양한 S3 작업의 HTTP 헤더가 성공적으로 추출되었으며, 인증 정보(Authorization), 객체 경로, 응답 상태 코드 등 트랜잭션 분석에 필요한 핵심 정보를 포함하고 있다.

Ⅳ. 결론

본 연구는 BlueField-2 DPU를 활용하여 S3 트래픽 모니터링을 호스트로부터 오프로딩하는 아키텍처를 제안하고 구현하였다. 네트워크 하드웨어 레벨에서 독립적으로 동작하는 DPU 기반 모니터링은 호스트 운영체제 공격으로부터 자유로우며, 호스트 성능 영향 없이 트랜잭션 레벨 가시성을 확보할 수 있음을 입증하였다. 실험 결과 호스트 기반 모니터링과 달리 호스트 CPU 부하를 거의 발생시키지 않음을 확인하였다.

하지만 현재 구현은 HTTP/1.1 평문 트래픽과 제한된 동시성을 갖는 단일 노드 환경에서 검증되었으며, 향후 TLS 암호화·HTTP/2 지원 및 수집된 메타데이터를 중앙에서 통합·분석하는 후단 시스템 구현이 요구된다. 그럼에도 불구하고, 본 연구는 호스트로부터 완전히 독립된 DPU 기반 S3 트랜잭션 모니터링의 실현 가능성을 보였다는 점에서 의의가 있으며, 이는 향후 다중 노드 클라우드 환경으로 확장하여 대규모 분산 시스템의 보안 감사 체계를 구축하는 중요한 단초를 제공한다.

ACKNOWLEDGMENT

본 연구는 2025년도 산업통상부 및 한국산업기술기획평가원(KEIT) 연구비 지원에 의한 연구(RS-2025-25448249)와 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2019-0-01842, 인공지능대학원지원(광주과학기술원))

참 고 문 헌

[1] M. Zhao et al., "Tectonic-Shift: A Composite Storage Fabric for Large-Scale ML Training," in Proc. USENIX Annu. Tech. Conf. (ATC), 2023, pp. 433-449.

[2] M. Vijayasanthi and G. Narsimha, "Design and Integration of a Level-based Access Control and Selective Auditing Framework for Cloud Services," Int. J. Intell. Eng. Syst., vol. 18, no. 7, 2025.

[3] A. G. Pennington et al., "Storage-based intrusion detection," ACM Trans. Inf. Syst. Secur., vol. 13, no. 4, pp. 1-27, 2010.

[4] I. Burstein, "Nvidia data center processing unit (dpu) architecture," in Proc. IEEE Hot Chips Symp. (HCS), Aug. 2021, pp. 1-20.