Edge Computing for the Internet of Things
:
A Case Study

Gopika Premsankar, Mario Di Francesco, and Tarik Taleb

K2025029 금동환

Table of Contents

Abstract

Abstract

배경

IoT 단말의 데이터가 폭증 -> 네트워크 대역폭 부담, 통신 지연

제안

IoT 애플리케이션을 위한 엣지 컴퓨팅

기여

1. 엣지 컴퓨팅 아키텍처/플랫폼 분류 및 조사, 엣지에 적합한 IoT 시나리오를 설명
2. 모바일 게임의 사용 사례로, 엣지 컴퓨팅과 확설화 기술에 대한 실험적 평가 수행

방법

- 응답 지연을 측정,비교

결과

- 엣지 컴퓨팅은 지연 시간 요구 충족에 필요

결론

- 현재 엣지 플랫폼으로 가능한 범위
- 새로운 기술이 IoT 어플리케이션에 미치는 영향

I. Introduction

I. Introduction

배경과 문제

- 모바일·IoT 단말의 데이터 생성이 급증, 연산·에너지 제약이 큼.
- 이를 완화하기 위해 클라우드 오프로딩이 널리 사용, 대규모 원거리 데이터센터 의존
  - 통신 지연
  - 네트워크 부하

엣지/포그 컴퓨팅의 개념과 효과

개념

네트워크 엣지에 연산/저장 자원을 배치

효과

- 통신 지연 감소
- 대역폭 부담 감소
- 이동성/지리적 분산

CDN/ICN과의 차이 및 이점

CDN/ICN

- 비대화형 콘텐츠를 사용자 근처에 배치/라우팅 최적화

엣지 컴퓨팅

- 대화형 콘텐츠 + 연산 능력
- 프라이버시
- 에너지 소비 감소

논문의 기여

1 **엣지 컴퓨팅 아키텍처·플랫폼 분류 및 조사**

엣지에 적합한 IoT 애플리케이션 시나리오 제시

2 **모바일 게임을 대표 사례로 성능 평가**

엣지·핵심 기술(예: 오프로딩)의 성능 평가 수행

I. Introduction

논문 구성 안내

- Section II: 엣지 플랫폼 분류/아키텍처.
- Section III: 엣지의 핵심 구현 기술.
- Section IV: IoT 요구사항과 엣지의 이점.
- Section V: 모바일 게임 성능 평가 결과.
- Section VI: 현 기술과의 관련성.
- Section VII: 결론 및 향후 연구 방향.

II. Edge Computing
:
Classes and Architectures

II. Edge Computing : Classes and Architectures

II-1. Resource-Rich Edge

엣지가 연결되는 네트워크에 고성능 서버를 배치

대표 설계

- **Cloudlet:** 엣지에서 1홉 거리의 Wi-Fi AP/기지국에 VM 기반 "data center in a box" 배치
- **Micro Cloud:** Wi-Fi AP /기지국에 소수 서버를 배치하는 형태
- **MEC (Multi-access Edge Computing):** 이모바일 네트워크의 무선 엑세스 컴포넌트에 배치

장점

낮은 지연, 높은 처리량, 무선 상태 정보 활용.

II. Edge Computing : Classes and Architectures

II-2. Heterogeneous Edge Nodes

다양한 컴퓨팅 자원을 활용

대표 설계

- Fot Platform: 가상화된 이기종 노드들의 시스템
- Local Cloud: 인근 단말들이 협력으로 로컬 클라우드를 형성
- Low power mini-cloud: Raspberry Pi 등 소형·저전력 노드를 묶은 휴대형 미니 클라우드

장점

다양한 자원 활용, 배치 유연성, 자원이 제한된 환경에서 운영 탄력성

II. Edge Computing : Classes and Architectures

II-3. Edge-Cloud Federation

엣지 자원과 데이터센터를 연합

대표 설계

- Edge Cloud: 엣지 앱이 엣지와 클라우드에서 모두 서비스
- Integrated Private and Public cloud: 엣지 노드가 동적 오케스트레이션
- FUSION Architecture: 인터넷에 클라두으 인프라에 서비스 배포
- Mirrored Edge Cloud: 공개 클라우드 서비스를 엣지에 미러링

장점

배치의 지리적인 유연성, 작업 처리량 극대화

III. Enabling Technologies

III. Enabling Technologies

III-1. Virtualization

단일 물리 서버에서 여러 격리된 인스턴스 실행

VM(하이퍼바이저 기반)

- 강력한 격리
- 오버헤드가 존재

컨테이너

- 호스트 OS 자원 공유
- 기동시간 단축
- 일반적 성능 우수

마이그레이션

- VM/컨테이너를 서버 간 이동

III. Enabling Technologies

III-2. Network Function Virtualization and Software Defined Network

NFV : 소프트웨어 모듈 형태로 네트워크 기능을 구현
SDN : 데이터와 제어 분리

NFV

- 범용 하드웨어에서 구동
- 향상된 유연성

SDN

- 중앙 논리 컨트롤러
- 빠른 신규 서비스

결합효과

- 구성/운영 단순화
- 비용 절감

III. Enabling Technologies

III-3. Computation Offloading

계산과 저장을 클라우드로 이전

효과

- 전력 소모 감소
- 다양한 앱 구동 가능

IV. Edge Computing for IoT Applications

IoT와 엣지 컴퓨팅 플랫폼

IoT 는 자원이 제한된 디바이스로 구성
기계친화적인 데이터로 다양한 서비스

IoT 특성별 엣지의 필요성

저지연 통신

커넥티드카, 모바일 게임, 원격 헬스 모니터링, 물류, 산업 제어

대역폭 집약 데이터

감시카메라, 순찰차, 사용자 단말의 영상

지리적 분산 처리

차량의 충돌 회피 시스템

단말 이동성

단말 이동에 맞춰 가상화 자원 마이그레이션을 수행, QoE 유지.

새로운 형태의 애플리케이션

요구

- 매우 짧은 반응 시간
- 사용자 움직임/상호작용에 신뢰도
- 추가적인 웨어러블 센서 데이터 활용

장점

배치의 지리적인 유연성, 작업 처리량 극대화

해법

왜 클라우드만으로는 부족한가

- 연산, 저장 자원이 풍부하지만 **물리적인 거리** 문제
  - 지연 시간 증가
  - 네트워크 부담 증가

엣지가 제공하는 해법

- 사용자 근접 연산/저장으로 지연 단축

V. Use Case
:
Mobile Gaming

V. Use Case : Mobile Gaming

V-1. Testbed Setup

- **플랫폼/게임**
  - 클라우드 게이밍 플랫폼: GamingAnywhere(오픈소스)
  - 게임: Neverball
- **응답 지연 정의**
  - PD (Processing delay) : 서버의 입력 처리 + 프레임 렌더링 시간
  - OD (Playout delay) : 클라이언트 디코딩 + 화면 표시 시간
  - ND (Network delay) : 클라이언트 <> 서버 RTT (round trip time)
- **액세스/네트워크**
  - Wi-Fi, LTE(NetLeap 4G, Nokia Solutions and Networks)
  - 동일 대학 네트워크 환경
- **클라이언트/서버 사양**
  - 클라이언트: Google Nexus 5, Android 5.1.1
  - 서버: Intel Xeon E3-1230 (4Core), 16GB Ram, Nvidia Quadro 2000 * 2

V. Use Case : Mobile Gaming

V-1. Testbed Setup

- **서버 배치 시나리오**
  - 네트워크 엣지 : LTE 기지국 공위치 / Wi-Fi는 클라이언트와 동일 무선망
  - 특수 목적 클라우드 : CSC cPouta(OpenStack), Kajaani(핀란드)
  - 퍼블릭 클라우드 : AWS EC2 (Frankfurt, Ireland 리전)

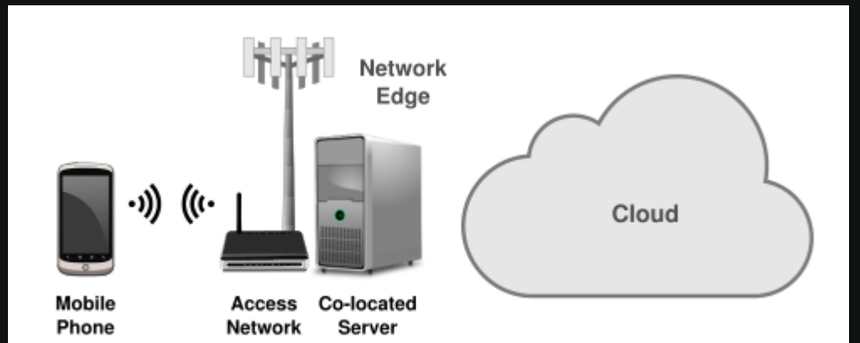

Fig. 2: Testbed setup used for the network edge scenario.

- **가상화 구성**
  - OS: Ubuntu 14.04.4
  - 구성: 베어메탈(B), 컨테이너(C, Docker 1.10.3), VM(V, QEMU 2.5.0)
  - GPU 1개를 컨테이너/VM에 직접 할당
  - EC2 인스턴스: g2.2xlarge(1 GPU, 8 vCPU), g2.8xlarge(4 GPU, 32 vCPU) — Dedicated instances 사용

- **실험 절차**
  - 각 조건 4회 반복 (통계적 유의성 확보)
  - 사전 녹화된 게임 세션 재생 (입력 타이밍 편차 최소화)
  - 세션 길이 1-3분, 스트리밍 30 FPS, 4.5 Mbps 고정

V. Use Case : Mobile Gaming

V-2. Experimental Results

- **네트워크 지연 (ND)**
  - 핀란드 내(엣지·전용 클라우드): < 25 ms, 엣지(LTE)에서는 < 20 ms 달성(현행 무선 통신의 상한선 수준).
  - 퍼블릭 클라우드(원격 리전): ≥ 50 ms (최소 2배 ↑). 거리(약 1,500-2,000 km)가 커질수록 ND 증가.
  - Wi-Fi vs LTE: 평균 Wi-Fi가 더 짧지만 분산(지터) ↑, LTE는 지터 ↓로 스트리밍에 더 안정적

- **처리 지연(PD) & 재생 지연(OD) vs 해상도 & 가상화**
  - 컨테이너 ≈ 베어메탈: 해상도와 무관하게 동일 수준 성능
  - VM은 PD 약 +30% 증가 → 1280×720에서 컨테이너는 30 FPS 유지 가능, VM은 미달
  - 1920×1080(Full HD): 모든 구성에서 목표 30 FPS 불가.
  - OD(클라이언트): FHD 미만에서 PD와 비슷한 크기, 해상도↑에 따라 증가하나 평균 25 ms 이내
  - 분산은 OD > PD, 해상도↑ 및 VM 사용 시 분산 더 큼.

- **클라우드의 추가 연산자원이 ND를 상쇄할 수 있는가?**
  - EC2 g2.2xlarge: 800×600, 1280×720에서 PD↓, 그러나 1920×1080에서는 +5 ms (베어메탈 대비).
  - EC2 g2.8xlarge: 모든 해상도에서 베어메탈보다 PD↓.
  - 핵심 원인: PD의 대부분이 인코딩 지연(렌더링보다)에서 발생 → GPU 늘려도 체감 이득 제한적.

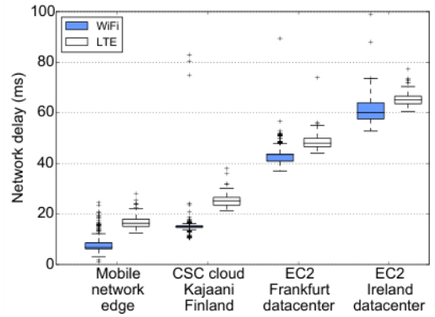

Fig. 3: Impact of network access technologies and server deployment on the network delay.

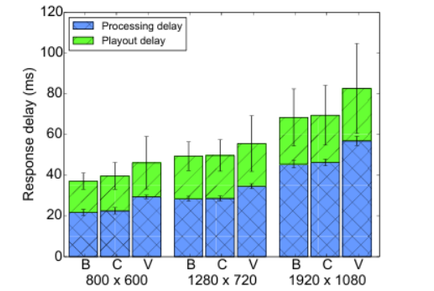

Fig. 4: Impact of the resolution on the response delay for different configurations: bare metal (B), containers (C) and virtual machines (V).

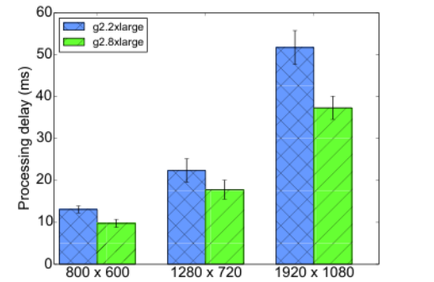

Fig. 5: Average processing delay as a function of the resolution for different Amazon GPU instances.

V. Use Case : Mobile Gaming

V-2. Experimental Results

- **시사점**
  - 근접 배치(엣지)가 ND를 압도적으로 단축 → 총 응답지연에서 결정적
  - 가상화 선택: 컨테이너 최우선(베어메탈급, VM 대비 PD 30% 절감)
  - 클라이언트 디코딩(OD) 관리와 지터 최소화 중요(LTE 유리)
  - 스케일업 한계: 클라우드 연산 증설만으론 네트워크 지연을 대체 못함 → 엣지에 최소 연산 자원이라도 배치가 효과적

VI. Discussion

VI. Discussion

논의

- **엣지 배치가 QoE 달성의 사실상 유일한 해법**
  - **지연 임계값:** 일반 상호작용은 < 150 ms 수용 가능하지만, 빠른 상호작용은 70 ms 초과 시 체감 품질 악화.
  - **엣지 vs 클라우드의 본질적 차이:** 클라우드의 고성능 자원만 늘려도 네트워크 지연을 상쇄하지 못함.

- **단일 사용자 실험의 일반화 가능성**
  - 여러 사용자가 관련된 상황에서도 주요 지표는 동일하게 유지
  - 핵심 결과는 다중 사용자 시나리오에서도 유효

- **NFV/SDN 오케스트레이션의 역할**
  - NFV : 가상화된 게이밍 모듈을 엣지에 배포. 실시간 트래픽을 애플리케이션 파라미터를 통해 조정
  - SDN : 네트워킹 관리/제어
  - 오케스트레이터 : 특정 게임의 요구에 따라 고성능의 VM/컨테이너 가동
  - 마이그레이션 : 사용자 이동성에 따라 VM/컨테이너의 실시간 마이그레이션

VI. Discussion

논의

- **한계와 목표**
  - 현 기술의 지연이 성능의 주요 제약
  - 10ms 이하 목표 : 무선 통신과 컴퓨팅 기술의 근본적이 발전 필요

- **확실한 점**
  - 엣지 컴퓨팅이 필수

VII. Conclusion

VII. Conclusion

결론

- 빠른 상호작용형 모바일 게임의 만족스러운 QoE 를 위해 엣지 컴퓨팅은 필수
- 근거리 데이터 센터가 지연을 줄이지만, 엣지에서 호스팅 리소스를 제공해야만 가능
- 클라우드 서버 성능 증대만으로 네트워크 지연 증가를 상쇄 불가
- 엣지에 제한된 자원만 배치해도 체감품질이 개선

향후 연구 방향

- 엣지 시반 모바일 게임의 대규모 평가
- 엣지 아키텍쳐를 응용 시나리오 별 비교

Contributions

엣지 컴퓨팅 아키텍쳐/플랫폼의 분류 및 조사

모바일 게임의 사례로 엣지 컴퓨팅과 활성화 기술에 대한 실험적 평가