클라우드 미디어 DJ 플랫폼 개발

정진웅, 이주현, 염상길, 추현승
성균관대학교 소프트웨어대학
e-mail:{sjud325, joohyun7, sanggil12, choo}@skku.edu

Development of Cloud Media DJ Platform

Jinwoong Jung, Joohyun Lee, Sanggil Yeoum, Hyunseung Choo
College of Software, Sungkyunkwan Univ.

요 약

모바일 디바이스 및 클라우드의 진보와 함께, 사람들이 이용해 온 많은 종류의 오프라인 서비스들이 온라인 서비스로 대체되었다. 과거에는 음악다방의 DJ들이 고객이 요청한 음악을 재생해주고 사연을 읽어주는 역할을 하였다. 우리는 이러한 서비스를 공용 클라우드, 스트리밍 서버, 스트리밍 클라이언트, 사용자 기기로 구성된 온라인 서비스로 대체하였다. 고객은 모바일 디바이스를 사용하여 음악, 영상, 사연을 요청하고, 현재 고객이 있는 공간의 스크린과 스피커를 통해 서비스를 수신한다. 우리는 또한 해당 서비스가 운용되는 클라우드 미디어 DJ 플랫폼을 설계하고 구현한다. 본 플랫폼은 대규모 트래픽을 수용하기 위해 Docker 컨테이너 배포 기술을 사용한다. 성능 실험결과에서 본 플랫폼은 동시 사용자 수에 따른 응답 시간이 최소 25%에서 최대 75%까지 줄었다.

1. 서론

1970년대에 인기를 얻은 한국의 음악다방에는 DJ가 있었다. 음악다방은 현재와 카페와 같이 음료수를 판매하고, 사람들이 서로 만나는 장소였다. 음악다방의 DJ는 다방 고객이 요청한 음악을 재생해주고, 노래를 부르며, 사연을 읽어주었다. 그러나, 개인 음악 장치의 발달로 인해 음악다방의 DJ가 많이 사라지고 음악다방만이 남게 되었다. 여전히 많은 사람이 좋아하는 음악을 일행과 공유하고 싶어 한다. 음악다방의 DJ들이 제공하는 오프라인 서비스는 어떻게 구현해야 할까?

클라우드 미디어 DJ 서비스에서 가장 중요한 요소는 음악다방 DJ의 역할을 어떻게 대체할 수 있는가이다. 우리는 DJ 대신 클라우드를 사용한다. DJ의 음악 재생은 음악 및 영상 서비스 공급자의 스트리밍 API로 대체한다. 종이에 적은 사연을 DJ에게 직접 건네던 행동은 안드로이드 또는 웹 앱으로 대체한다. DJ의 목소리는 카페의 스피커가 대신하고, 스크린을 통해 고객의 사연을 본다.

클라우드 미디어 DJ 서비스는 안정적이고 합리적인 서버 플랫폼이 필요하다. 카페 주인은 서비스 이용료 지급에 부담을 느껴서는 안 되며, 서비스를 사용하면서 보안 위협을 느껴서는 안 된다. 제안 플랫폼은 Google Compute Engine[1](GCE)에서 운용되며, 제안 서비스의 안정적인 운영을 위해 Docker[2] 컨테이너 별로 관리된다. 카페 주인은 물리적 서버가 필요 없으며, 복잡한 관리 및 높은 지출 없이 고객에게 새로운 서비스를 제공할 수 있다. Docker 컨테이너로 관리되는 서버는 단일 서버 구조보다 높은 안정성 및 낮은 대기 시간을 보여준다.

본 논문에서는 클라우드 미디어 DJ 서비스와 서비스가 설치된 제안 플랫폼에 관해 설명한다. 먼저, 제안 서비스가 무엇인지 소개하고, 음악다방 DJ의 역할이 서버와 클라이언트 측면에서 어떻게 구현되는지 설명한다. 다음으로, 대규모 트래픽에도 잘 동작하는 저비용, 고효율 플랫폼을 제안하고 이 플랫폼이 단일 서버 구조보다 낮은 응답 시간과 높은 안정성을 갖고 있다는 것을 입증한다. 마지막으로, 결론과 향후 연구에 관해 기술한다.

2. 관련 연구

2.1 모바일 앱과 웹 앱

21세기 IT 시대의 가장 큰 혁신은 모바일 디바이스의 탄생이다 [3]. 스마트폰으로 언제 어디서나 인터넷에 접근할 수 있는 기술은 최근 발생한 가장 큰 업적이다. 또한, 무선 네트워크 통신 품질이 발전함에 따라 모바일 디바이스와 모바일 앱 서비스도 함께 발전했다. 이제는 서비스 대부분을 모바일로 실행할 수 있다. 온라인 앱 서비스는 미래 IT 시대에서 중요성이 더욱 높아질 것이다.

모바일에서 주로 실행되는 앱은 네이티브 앱과 웹 앱이다 [4]. 네이티브 앱은 다른 실행 환경을 거치지 않고 모바일 운영체제에서 바로 실행된다. 따라서 성능이 좋고, 많은 권한을 취득할 수 있다는 장점이 있다. 그러나 운영체제 종속적이라는 단점이 있다. 웹 앱은 웹 뷰를 거쳐 실행되므로 네이티브 앱보다는 성능이 떨어진다. 그러나 웹 앱에서 준수되는 HTML5 표준으로 인해 모든 장치에서 해당 웹 앱을 실행할 수 있다. 개발자는 자산 및 앱 시나리오를 고려하여 적절한 앱 스타일을 선택해야 한다.

2.2 클라우드와 가상화

클라우드 컴퓨팅[5]은 오늘날 매우 관심받는 트렌드 중 하나이다. 이 기술은 우리가 실제 컴퓨터가 아닌 인터넷에 연결된 다른 컴퓨터로 정보를 처리하는 것을 의미한다. IT 대기업은 전 세계에 자체 데이터 센터를 구축하고 있고, 많은 중소기업이 대기업이 제공하는 클라우드 컴퓨팅 서비스를 이용하고 있다. 이는 높은 비용과 물리적 관리가 필요하지 않고, 보안 위협에서 안전하다. 관련 기술에는 예전부터 꾸준히 사용되어 온 VM(Virtual Machine)과 최근에 주목받는 컨테이너가 존재한다.

VM[6]은 컴퓨터를 에뮬레이션한 것이다. 기존 물리적 하드웨어의 Hypervisor 기술을 통해 운영체제를 가상화하면, 해당 VM은 호스트와 분리되어 실행된다. 따라서 호스트와 다른 운영체제를 사용할 수 있으며, 하나의 VM이 공격을 받더라도 호스트 및 다른 VM은 안전하게 유지될 수 있다. 그러나 운영체제를 가상화했을 때의 필요 컴퓨팅 자원이 매우 크다. 또한, 가상화된 운영체제는 일반적으로 호스트보다 성능이 매우 떨어진다.

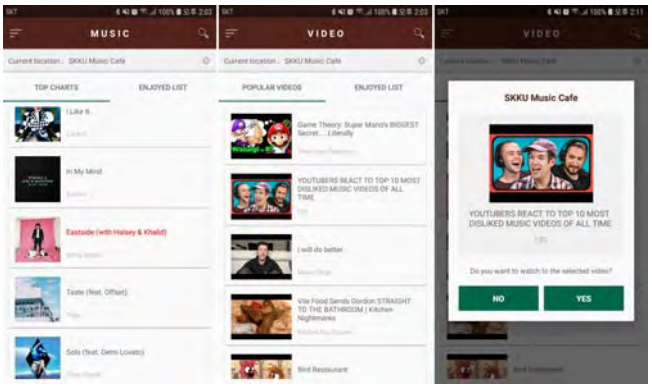
컨테이너는 VM과는 달리 프로세스를 가상화한다. 프로세스 가상화는 호스트와 비교해 성능이 저하되지 않으며, 프로세스가 필요한 자원만을 사용하기 때문에 컴퓨팅 자원이 많이 필요하지 않다. 그러나 운영체제 가상화가 아니므로 호스트와 다른 운영체제를 사용할 수 없다. 또한, 호스트와 연결되어 있으므로 VM보다는 보안 위협에 더 취약할 수 있다. 그런데도 컨테이너의 최근 인기는 Docker의 탄생에서 비롯된다. Docker는 컨테이너를 만들고, 관리하며, 배포할 수 있도록 도와준다. Docker 사용자는 패키지를 Docker 이미지로 변환하여 공개된 저장소에 게시한다. 다른 사용자는 언제든지 공개된 이미지를 내려받아 해당 패키지를 최적의 상태로 이용할 수 있다.

앞서 언급했듯 VM과 Docker 컨테이너는 각자 장단점이 있다. 개발자가 서버를 관리하기 위해 사용하는 가상화 기술은 서비스 또는 플랫폼 특성을 기반으로 한다. 본 플랫폼은 각각의 카페마다 다른 서버를 배치하여 안정성을 높인다. 예를 들어, 모든 카페가 단일 서버에서 관리된다고 가정해보자. 하나의 카페에서 오류가 발생하면, 서버가 중단되고 모든 카페에서 서비스를 이용할 수 없게 된다. 그렇다고 해서 각각의 카페에 클라우드 컴퓨팅 인스턴스를 하나씩 둔다면, 천문학적인 비용이 필요하다. 또한, 본 서비스를 사용하는 카페의 수가 많다고 가정하므로 컴퓨팅 자원을 많이 사용하는 VM은 적절하지 않다. 이럴 때 컨테이너를 사용하여 각 카페의 서버를 분리하고 관리하면 보다 안정적이고 효율적인 서버 환경이 구성된다.

3. 플랫폼 설계

3.1 클라우드 미디어 DJ 서비스

클라우드 미디어 DJ 서비스는 카페 고객이 음악, 영상, 사연을 신청하면 현재 위치한 카페에서 해당 콘텐츠를 고객 모두가 함께 수신하는 온라인 서비스다. 고객은 안드로이드 또는 웹 앱을 사용하여 콘텐츠를 신청할 수 있으며, 콘텐츠는 카페의 스크린과 스피커에서 재생된다. 고객은 카페에서 본인이 원하는 음악을 들을 수 있고, 카페 주인은 일반 고객을 단골로 바꿀 수 있는 기반을 마련한다. 고객의 신청 이력은 항상 저장되며, 고객은 본 서비스를 통해 음악과 사연을 일행과 공유할 수 있다. 카페 주인은 본 플랫폼의 웹 앱을 사용하여 별도의 장치 없이 모바일 및 노트북에서 콘텐츠 재생 환경을 구성할 수 있다.



(그림 1) 본 서비스의 앱 화면

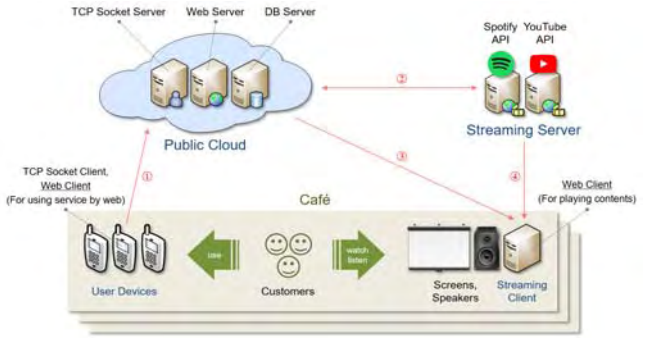
본 서비스는 고객이 카페에서 신청한 음악 및 영상을 재생할 수 있도록 음악 및 영상 스트리밍 기능을 제공한다. 고객이 인기차트 또는 검색을 통해 콘텐츠를 선택하면 카페에서 실시간으로 스트리밍된다. 이 기능은 본 서비스의 핵심이며, 음악다방 DJ의 가장 큰 역할을 대체한다. 또한, 고객은 카페 스크린에 출력할 사연을 신청할 수 있다. 고객이 앱을 통해 특정 문장을 작성하면, 해당 문장이 스크린에 출력된다. 이 기능은 고객의 사연을 읽어주던 음악다방 DJ의 부수적인 역할을 대체한다.

3.2 플랫폼 구조 및 서비스 시나리오

클라우드 미디어 DJ 서비스는 공용 클라우드, 스트리밍 서버, 스트리밍 클라이언트, 사용자 기기로 이루어진 플랫폼에서 운영된다. 공용 클라우드는 카페와 고객 및 스트리밍 서버를 연결한다. 스트리밍 서버는 음악 또는 영상 제공 서비스 업체를 의미하며, 스트리밍 클라이언트에 실질적인 콘텐츠 제공 역할을 맡는다. 스트리밍 클라이언트는 해당 콘텐츠를 재생한다. 카페 고객은 사용자 기기에 설치된 앱으로 원하는 콘텐츠를 신청한다. 카페 내부에는 사용자 기기와 스트리밍 클라이언트가 존재하며, 카페 외부에는 공용 클라우드와 스트리밍 서버가 존재한다.

공용 클라우드는 TCP Socket Server, Web Server, DB Server로 이루어진다. TCP Socket Server는 안드로이드 앱과 통신하기 위해 사용된다. Web Server는 고객에게 콘텐츠 신청 웹 앱을 제공하는 용도 및 카페 주인에게 콘텐츠 스트리밍 웹 앱을 제공하는 용도로 사용된다. DB Server는 고객의 정보와 콘텐츠 신청 이력 및 카페 정보를 저장하는 역할을 한다. 본 플랫폼은 스트리밍 서버로 2개 업체를 선정하여 사용했다. 음악 콘텐츠는 Spotify API를 사용했으며, 영상 콘텐츠는 Youtube API를 사용했

다. 스트리밍 클라이언트는 웹 앱이므로 웹 페이지의 화면과 소리를 스크린과 스피커에 출력해줄 수 있는 기기라면 어떤 기기든 해당 역할을 해낼 수 있다.



(그림 2) 본 서비스의 메인 시나리오

① 사용자가 앱을 통해 콘텐츠를 신청하면, 해당 콘텐츠의 정보와 카페 및 고객 정보가 공용 클라우드에 전송된다. 해당 데이터는 Socket 또는 Web Socket 통신을 통해 JSON 문자열로 전송된다. 프로토콜 데이터는 어떤 기능을 실행할지 결정하며, 나머지는 해당 기능을 수행할 때 필요한 부가정보가 된다.

② 공용 클라우드는 데이터를 수신하여 프로토콜을 통해 특정 기능을 수행한다. 또한, 부가정보에 포함된 해당 콘텐츠 정보를 스트리밍 서버가 제공하는 REST API를 사용하여 검색한다. 이후, 검색 결과로 출력되는 데이터에서 해당 콘텐츠의 스트리밍 웹 주소를 수집한다.

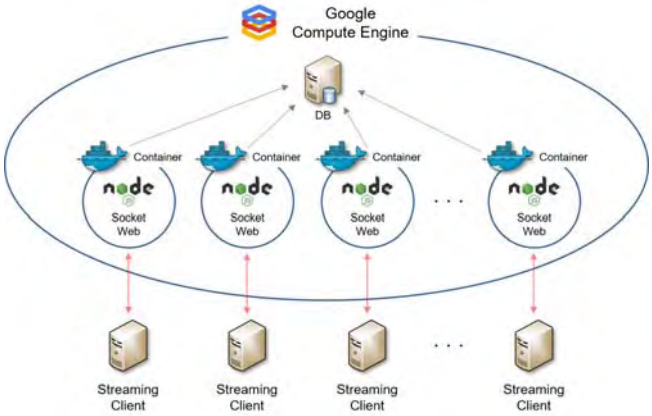
③ 공용 클라우드는 해당 카페와 일치하는 스트리밍 클라이언트로 스트리밍 웹 주소와 사용자 정보를 전송한다. 프로토콜은 콘텐츠 신청 데이터를 수신할 때 받은 프로토콜을 그대로 전송한다. 스트리밍 클라이언트도 해당 프로토콜로 정해진 기능을 수행한다.

④ 스트리밍 클라이언트에서 콘텐츠 재생 방법은 두 가지로 나뉜다. 첫째는 HTML5 video 태그를 활용한 음악 재생이다. video 태그의 src 속성에 스트리밍 웹 주소를 부여하면 음악이 재생된다. 둘째는 YouTube IFrame Player API를 활용한 YouTube 영상 재생이다. 해당 API는 HTML5의 iframe 태그를 이용하여 YouTube 영상을 불러와 재생하는 방식이다. videoId 속성에 재생하고 싶은 YouTube 영상 고유 ID를 입력하면 영상이 재생된다. 두 API 모두 재생 완료 이벤트가 존재하므로 해당 이벤트를 통해 다음 콘텐츠를 순서대로 재생시킬 수 있다.

3.3 클라이언트 – 서버 통신

공용 클라우드는 GCE 속에서 하나의 인스턴스로 구동된다. GCE는 구글 클라우드 플랫폼의 핵심 서비스이다. 사용자가 돈을 지급하면 원하는 사양의 가상 컴퓨터를 제공해준다. 사용자가 한 번 설정한 가상 컴퓨터 사양은 언제든지 바꿀 수 있다. 구글에서 제공하는 것이기 때문에 보안이 뛰어나며, 물리 서버보다 유지 보수가 편리하다.

공용 클라우드는 앞서 말했듯 TCP Socket Server와 Web Server 및 DB Server로 구성된다. Node.js에서 구동되는 TCP Socket Server는 Node.js의 net 모듈로 구현하였다. net 모듈에서 TCP Socket Server를 생성하면 안드로이드의 Socket API를 통하여 해당 서버로 접속할 수 있다. Web Server는 express 모듈을 사용하여 구현하였다. express 모듈로 Web Server를 만들고 실시간 통신 기술을 위해 Web Socket 모듈을 추가로 사용하였다. Web Client는 웹 주소를 입력하여 해당 Web Server에 접속할 수 있고, HTML5에서 추가된 Web Socket 라이브러리를 사용하여 해당 서버와 실시간 통신할 수 있다. DB Server와의 통신은 mysql 모듈을 통해 Back-end로 이루어진다.



(그림 3) 공용 클라우드 구조

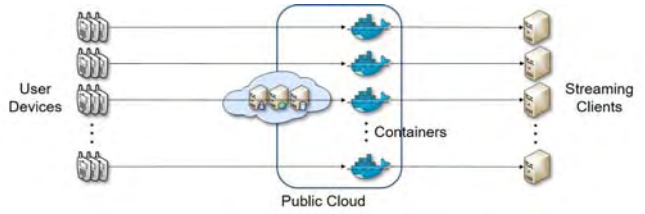
우리는 GCE에 지급하는 비용은 최소화하고, 성능은 합리적으로 끌어올리며, 서버의 안정성을 높여주는 방안으로 컨테이너 구조를 생각해냈다. 컨테이너 별로 카페를 나누면 한 카페에서 오류가 발생하더라도 다른 카페에 영향을 미치지 않는다. 또한, 이벤트가 발생할 때만 작업을 하는 Node.js의 Event Driven 특징 및 프로세스 가상화 방식을 사용해 컴퓨팅 자원을 필요한 만큼만 사용하는 컨테이너의 특징이 맞물려 카페 별 서버 운영을 매우 적은 수의 GCE 인스턴스로 구성할 수 있다.

본 서비스의 기능이 구현된 서버 프로세스는 Docker 이미지로 패키징하였다. 해당 이미지를 이용하면 수백 개의 컨테이너도 한 줄의 명령어로 생성할 수 있다. 이를 통해 공용 클라우드에는 TCP Socket Server와 Web Server 및 DB 통신 Back-end가 포함된 Node.js 서버 프로세스가 컨테이너로 나누어지고, 해당 컨테이너는 카페 수만큼 존재하는 것이 된다. DB Server는 모든 콘텐츠 신청 이력이 하나의 공간에 저장되기 위하여 분리하지 않는다.

4. 성능 비교측정

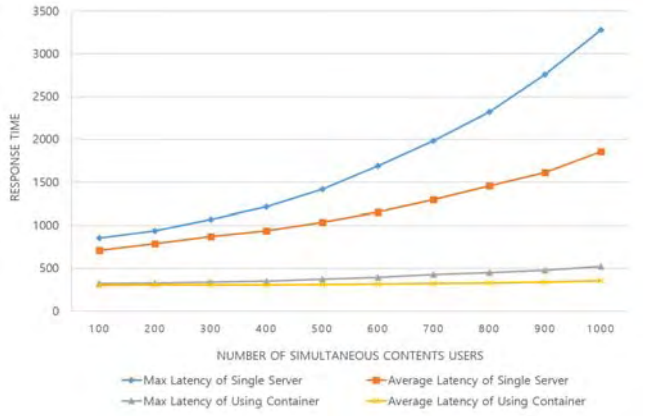
본 플랫폼의 성능을 비교 측정하기 위해 일반적인 서버 구조인 단일 서버 구조를 구현하였다. 단일 서버 구조는 하나의 GCE 인스턴스에 TCP Socket Server, Web Server, DB Server를 단 하나씩만 생성하는 구조다. 하나의 TCP Socket Server와 Web Server가 다수의 카페와 통신하며, 하나의 DB Server가 모든 정보를 저장한다. 인스턴스 사양은 vCPU 4개와 16GB 메모리로 설정하였다.

(그림 4) 단일 서버 구조 : 비교 대상

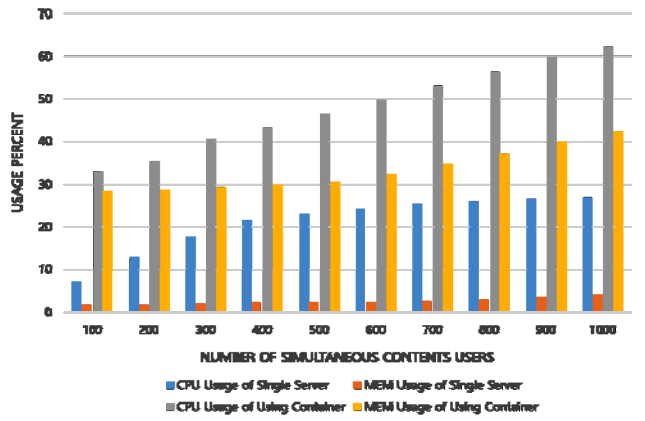


(그림 5) 컨테이너 사용 구조 : 본 플랫폼

본 플랫폼은 하나의 GCE 인스턴스에서 컨테이너를 활용하여 카페 별로 서버를 달리 제공한다. TCP Socket Server와 Web Server는 카페 수만큼 존재하는 컨테이너에 하나씩 포함된다. 따라서 TCP Socket Server 하나와 Web Server 하나가 단 하나의 카페와 통신한다. 단일 서버 구조와 마찬가지로 하나의 DB Server가 모든 정보를 저장하며, 인스턴스 사양도 같게 설정하였다.



(그림 6) 동시 신청 횟수에 따른 응답 시간



(그림 7) 동시 신청 횟수에 따른 자원 사용률

위 그래프는 제안 서비스를 이용하는 200개의 카페가 있다고 가정하고, 다수의 고객이 하나의 콘텐츠를 신청했을 때 공용 클라우드의 콘텐츠 신청 완료 응답 시간과 컴퓨팅 자원 사용률을 보여준다. 콘텐츠 신청 완료 응답 시간은 대규모 트래픽이 발생했을 때 카페 고객의 불편함을 파악하기 위함이며, 컴퓨팅 자원 사용률은 인스턴스를 효율적으로 사용할 수 있나 확인하기 위함이다.

응답 시간 그래프를 보면 컨테이너 사용 구조가 단일 서버 구조와 비교해 최소 25%에서 최대 75%만큼 짧은 지연 시간을 가진다는 것을 알 수 있다. 컴퓨팅 자원 사용률 그래프를 보면 컨테이너 사용 구조가 단일 서버 구조와 비교해 CPU와 메모리를 최소 2배에서 최대 14배만큼 효율적으로 사용한다는 것을 알 수 있다. 단일 서버 구조는 대규모 트래픽에서 간혹 데이터 손실이 발생하는데, 그 이유는 대량의 콘텐츠 신청 데이터를 한 서버에서 받아들여 데이터 충돌이 발생하기 때문이다. 본 플랫폼은 온라인 서비스 측면에서, 안정성과 고효율 및 짧은 지연 시간이 최우선이기 때문에 컨테이너 사용 구조가 적합하다.

5. 결론 및 향후 연구

본 플랫폼을 통해 고객은 콘텐츠 신청 서비스를 안정적으로 이용할 수 있으며, 카페 주인은 저비용 및 고효율의 콘텐츠 서비스를 제공할 수 있다. 현재 제공되는 추천 기능은 이전에 신청한 콘텐츠를 추천할 뿐이기 때문에, 장르와 연령대를 사용하는 실질적인 맞춤형 추천 기능을 개발할 예정이다. 또한, 실내에서 정확도가 떨어지는 GPS 정보를 대신하여 비 가청 주파수를 이용해 고객의 정확한 실내 위치를 파악하는 기능을 개발할 계획이다.

ACKNOWLEDGEMENT

본 논문은 기초연구사업 (NRF-2010-0020210)과 과학기술정보통신부 및 정보통신기술진흥센터의 Grand ICT연구센터지원사업 (IITP-2018-2015-0-00742)의 연구결과로 수행되었음

참고문헌

[1] S. P. T. Krishnan and J. L. U. Gonzalez, “Google compute engine,” In Building Your Next Big Thing with Google Cloud Platform, pp. 53-81, Apress, Berkeley, CA, 2015.

[2] D. Merkel, “Docker: lightweight linux containers for consistent development and deployment,” Linux Journal, 2014(239), 2, 2014.

[3] Q. Bi, G. L. Zysman, and H. Menkes, “Wireless mobile communications at the start of the 21st century,” IEEE communications magazine, 39(1), 110-116, 2001.

[4] A. Charland and B. Leroux, “Mobile application development: web vs. native,” Queue, 9(4), 20, 2011.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, and M. Zaharia, “A view of cloud computing,” Communications of the ACM, 53(4), 50-58, 2010.

[6] Y. Li, W. Li, and C. Jiang, “A survey of virtual machine system: Current technology and future trends,” In Electronic Commerce and Security (ISECS), 2010 Third International Symposium on (pp. 332-336), IEEE, 2010.