(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2023년04월25일
(11) 등록번호 10-2526180
(24) 등록일자 2023년04월21일

(51) 국제특허분류(Int. Cl.)
H04W 36/38 (2009.01) *H04W 36/02* (2009.01)
H04W 36/08 (2009.01) *H04W 88/18* (2019.01)
(52) CPC특허분류
H04W 36/385 (2013.01)
H04W 36/02 (2013.01)
(21) 출원번호 10-2021-0149441
(22) 출원일자 2021년11월03일
심사청구일자 2021년11월03일
(56) 선행기술조사문헌
공개특허공보 제10-2017-0010834호 (2017.02.01.)*
공개특허공보 제10-2018-0138438호 (2018.12.31.)*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
한림대학교 산학협력단
강원도 춘천시 한림대학길 1, 한림대학교(옥천동)
(72) 발명자
김태운
강원도 춘천시 백령로 215, 108동 602호
(74) 대리인
김남혁

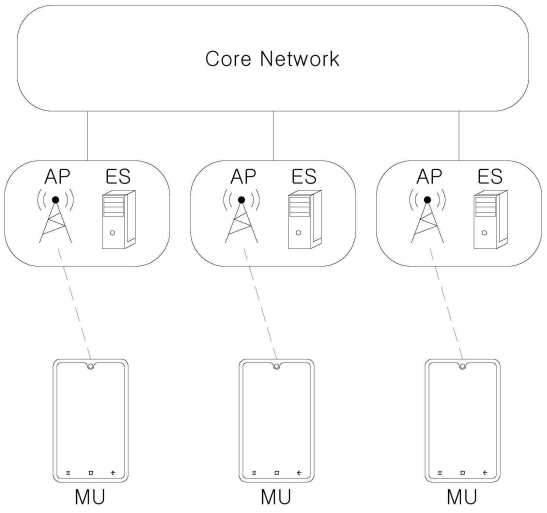
전체 청구항 수 : 총 5 항 심사관 : 조희정

(54) 발명의 명칭 **핸드오프 과정에서 컨테이너의 마이그레이션을 수행하는 모바일 엣지 컴퓨팅 시스템**

(57) 요 약

모바일 엣지 컴퓨팅 시스템이 개시된다. 본 시스템은, 제1 AP(Access Point), 제1 AP를 통해 적어도 하나의 단말과 연동되어 서비스를 제공하는 제1 ES(Edge Server), 제2 AP, 제2 AP를 통해 적어도 하나의 단말과 연동되어 서비스를 제공하는 제2 ES, 코어 네트워크에 포함된 적어도 하나의 서버를 포함한다. 서버는, 제1 ES에 포함된 컨테이너의 동작에 따라 사용자 단말로 서비스가 제공되는 동안 핸드오프 이벤트가 감지되면, DC(Differential-Copy) 및 LR(Log-Replay) 중 하나의 마이그레이션 절차를 선택하고, 선택된 마이그레이션 절차에 따라, 제1 ES에서 동작 중인 컨테이너를 제2 ES로 전달하는 마이그레이션을 진행한다.

대 표 도 - 도1



(52) CPC특허분류
H04W 36/08 (2013.01)
H04W 88/18 (2019.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711136073
과제번호	2021R1F1A1059109
부처명	과학기술정보통신부
과제관리(전문)기관명	한국연구재단
연구사업명	기본연구
연구과제명	공공안전 및 재난 통신을 위한 지능형 무선 네트워크 최적화 기술 연구
기 여 율	1/1
과제수행기관명	한림대학교 산학협력단
연구기간	2021.06.01 ~ 2024.02.29

명 세 서

청구범위

청구항 1

모바일 엣지 컴퓨팅 시스템에 있어서,

제1 AP(Access Point);

상기 제1 AP를 통해 적어도 하나의 단말과 연동되어 서비스를 제공하는 제1 ES(Edge Server);

제2 AP;

상기 제2 AP를 통해 적어도 하나의 단말과 연동되어 서비스를 제공하는 제2 ES; 및

코어 네트워크에 포함된 적어도 하나의 서버;를 포함하고,

상기 서버는,

상기 제1 ES에 포함된 컨테이너의 동작에 따라 사용자 단말로 서비스가 제공되는 동안 핸드오프 이벤트가 감지되면, DC(Differential-Copy) 및 LR(Log-Replay) 중 하나의 마이그레이션 절차를 선택하고,

상기 선택된 마이그레이션 절차에 따라, 상기 제1 ES에서 동작 중인 상기 컨테이너를 상기 제2 ES로 전달하는 마이그레이션을 진행하며

상기 DC는,

상기 제1 ES가 상기 컨테이너의 쓰기 가능한 계층을 상기 제2 ES로 전송하고, 상기 제1 ES가 체크포인트를 생성하여 상기 제2 ES로 전송하며, 상기 제2 ES는 상기 쓰기 가능한 계층 및 상기 체크포인트를 기반으로 상기 컨테이너를 최신 상태에서 재개하는, 모바일 엣지 컴퓨팅 시스템.

청구항 2

삭제

청구항 3

제1항에 있어서,

상기 LR은,

상기 제1 ES가 상기 제1 ES에서 실행된 command trace를 상기 제2 ES로 전송하고, 상기 제2 ES가 상기 command trace를 리플레이 하여 상기 컨테이너를 최신 상태에서 재개하는, 모바일 엣지 컴퓨팅 시스템.

청구항 4

제1항에 있어서,

상기 서버는,

이하 수학식에 따라, 상기 DC 및 상기 LR 중 하나의 마이그레이션 절차를 선택하고,

$$\min_x f(x)=\frac{1}{\alpha}\{x\cdot T_{DC}+(1-x)T_{LR}\}+w\frac{1}{\beta}\{x\cdot L_{DC}+(1-x)L_{LR}\}$$

상기 T_{DC}는 상기 DC의 마이그레이션 시간이고, 상기 T_{LR}은 상기 LR의 마이그레이션 시간이고, 상기 L_{DC}는 상기 DC에 대한 실행의 결과로 발생하는 네트워크 부하이고, 상기 L_{LR}은 상기 LR에 대한 실행의 결과로 발생하는 네트워크 부하이고, 상기 w는 네트워크 부하에 대한 가중치를 나타내는 음이 아닌 디자인 파라미터이고,

상기 수학식에서, x는 이하 범위로 설정되는,

$x \in \{0,1\}$

모바일 엣지 컴퓨팅 시스템.

청구항 5

모바일 엣지 컴퓨팅 시스템의 제어 방법에 있어서,

제1 ES(Edge Server)가 제1 AP(Access Point)를 통해 사용자 단말로 서비스를 제공하는 단계;

상기 사용자 단말과 관련된 핸드오프 이벤트가 감지되면, 코어 네트워크의 서버가 DC(Differential-Copy) 및 LR(Log-Replay) 중 하나의 마이그레이션 절차를 선택하는 단계;

상기 서버가, 상기 선택된 마이그레이션 절차에 따라, 상기 제1 ES에서 동작 중인 컨테이너를 제2 AP와 매칭되는 제2 ES로 전달하는 마이그레이션을 진행하는 단계; 및

상기 마이그레이션이 수행되면, 상기 제2 ES가 제2 AP를 통해 상기 사용자 단말로 상기 서비스를 제공하는 단계;를 포함하고,

상기 DC는,

상기 제1 ES가 상기 컨테이너의 쓰기 가능한 계층을 상기 제2 ES로 전송하고, 상기 제1 ES가 체크포인트를 생성하여 상기 제2 ES로 전송하며, 상기 제2 ES는 상기 쓰기 가능한 계층 및 상기 체크포인트를 기반으로 상기 컨테이너를 최신 상태에서 재개하는, 모바일 엣지 컴퓨팅 시스템의 제어 방법.

청구항 6

모바일 엣지 컴퓨팅 시스템에 포함되는 서버의 제어 방법에 있어서,

제1 ES(Edge Server)가 제1 AP(Access Point)를 통해 사용자 단말로 서비스를 제공하는 동안, 상기 사용자 단말과 관련된 핸드오프 이벤트를 감지하는 단계;

상기 핸드오프 이벤트가 감지되면, DC(Differential-Copy) 및 LR(Log-Replay) 중 하나의 마이그레이션 절차를 선택하는 단계; 및

상기 선택된 마이그레이션 절차에 따라, 상기 제1 ES에서 동작 중인 컨테이너를 제2 AP와 매칭되는 제2 ES로 전달하는 마이그레이션을 진행하는 단계;를 포함하고

상기 DC는,

상기 제1 ES가 상기 컨테이너의 쓰기 가능한 계층을 상기 제2 ES로 전송하고, 상기 제1 ES가 체크포인트를 생성하여 상기 제2 ES로 전송하며, 상기 제2 ES는 상기 쓰기 가능한 계층 및 상기 체크포인트를 기반으로 상기 컨테이너를 최신 상태에서 재개하는, 서버의 제어 방법.

발명의 설명

기 술 분 야

[0001] 본 개시는 복수의 엣지 서버를 포함하는 모바일 엣지 컴퓨팅 시스템에 관한 것으로, 보다 상세하게는, 컨테이너의 마이그레이션을 통해 서비스의 연속성을 유지하면서 핸드오프를 수행하는 시스템의 제어 방법에 관한 것이다.

배 경 기 술

[0002] 우리는 사물 인터넷 시대에 살고 있다. 사용자 주변의 (IoT) 장치는 지속적으로 방대한 데이터를 생성하고 이를 처리하여 스마트 핸드헬드 장치에서 실행되는 사용자 중심 애플리케이션을 가능하게 한다. 특히, 빅 데이터 분석, 가상 현실(VR)/증강 현실(AR), 이미지 처리 등과 같이 계산량이 많은 작업은 이러한 애플리케이션 프로그램의 성공을 위한 핵심 요소이다. 그러나, IoT 장치나 핸드헬드 장치는 제한된 리소스로 인해 계산량이 많은 응용 프로그램을 실행하는 동안 실시간 상호 작용을 제공하기에 충분하지 않다(L. Zhao et al).

[0003] 풍부한 사용자 경험에 대한 필요성으로 인해 저전력 장치가 계산적으로 무거운 작업을 오프로드할 수 있는 클라우드 컴퓨팅이 대중화되었다. 클라우드 컴퓨팅은 방대한 양의 IT 리소스를 중앙 집중하여 필요에 따라 컴퓨팅 리소스를 프로비저닝하고 다양한 서비스 수요에 활용될 수 있다(예: 자동 크기 조정). 또한 가상화는 클라우드 컴퓨팅에서 높은 리소스 활용도(resource utilization), 리소스 격리(resource isolation) 및 멀티 테넌시(multi-tenancy)를 달성하는 데 핵심적인 역할을 한다(C. Puliafito et al).

[0004] 리소스가 제한된 장치의 경우 작업을 클라우드 컴퓨팅으로 오프로딩하면 작업 처리 시간과 배터리 사용량을 줄이는 데 크게 도움이 된다. 그러나, 단말 장치까지의 거리가 멀수록 높은 지연이 발생할 수 있어 의료, 커넥티드 차량, AR, 감시/모니터링과 같은 지연에 민감한 애플리케이션에는 적합하지 않다.

[0005] 자원이 제한된 단말 장치에 추가적인 컴퓨팅 자원을 제공함과 동시에 대기 시간을 최소화하기 위해 클라우드 컴퓨팅의 유망한 대안으로 엣지 컴퓨팅이 등장했다. 엣지 컴퓨팅은 클라우드 리소스를 네트워크 엣지에 배치한다. 따라서 최종 장치까지의 거리를 줄이면 네트워크 지연이 효과적으로 줄어들 수 있다.

[0006] 도 1은 에지 컴퓨팅을 사용하는 네트워크 다이어그램의 예를 보여준다. 에지 서버(ES)는 액세스 포인트(AP)와 함께 배치되며, AP와 연결된 사용자는 배치된 ES로 작업을 오프로드 할 수 있다. 각 사용자는 ES에서 실행되는 자체 서비스를 가지고 있으며 가상화 및 멀티 테넌시로 인해 동일한 ES에서 실행되는 다른 서비스의 운영을 위반하지 않는다. 사용자가 연결된 AP의 무선 범위를 벗어나면 사용자의 연결을 다른 AP로 전송하기 위해 핸드오프 절차가 트리거 된다.

[0007] 사용자가 이동하는 동안 서비스 지연을 최소화하기 위해 ES에서 실행 중인 서비스를 다른 ES로 이전하는 서비스 마이그레이션의 개념이 제안되었다. 특히 상태 저장 및 라이브 서비스 마이그레이션은 진행 중인 서비스를 중단하지 않고 실행 중인 서비스를 완전히 마이그레이션하기 때문에 쉽지 않다(W Shi et al, S. Wang et al). 서비스 마이그레이션 프로세스를 가속화하기 위해 Docker(Docker [Online]. https://www.docker.com)와 같은 경량 컨테이너화된 가상화의 사용이 많은 주목을 받았다. Docker는 네이티브에 가까운 성능을 달성할 수 있다.

[0008] 서비스 마이그레이션 기법을 제안하는 연구는 몇 가지 있었지만 현재 진행 중인 서비스의 특성과 마이그레이션 기법을 함께 고려하지 않았다는 것이 가장 큰 한계점이다. 서비스마다 속성이 다르기 때문에 컨테이너화된 서비스를 효율적으로 마이그레이션하려면 자율적으로 다른 마이그레이션 기법을 선택해야 한다. 또한, 도 1에서 볼 수 있듯이 컨테이너 마이그레이션으로 인해 발생하는 네트워크 트래픽이 코어 네트워크에 주입되며 트래픽이 많을 경우 네트워크 혼잡을 유발할 수 있다. 따라서 마이그레이션 시간을 최소화하면서 네트워크 트래픽이 너무 많이 발생하지 않도록 마이그레이션 방법을 신중하게 설계할 필요가 있다.

[0009] 클라우드 컴퓨팅과 에지 컴퓨팅 모두 작업 오프로딩을 허용하여 최종 장치의 배터리 수명을 연장하는 데 도움이 된다. 엣지 컴퓨팅은 특히 지연에 민감한 애플리케이션을 위한 클라우드 컴퓨팅의 유망한 대안이다. 또한, 엣지 컴퓨팅은 사용자 주변에서만 네트워크 전송을 발생시켜 대역폭 비용과 프라이버시 위협을 줄일 수 있다.

[0010] 관련하여, 컨테이너 마이그레이션 및 구현에 대한 이전 연구를 검토할 필요가 있다. 엣지 컴퓨팅 테스트베드나 오프로딩 정책 설계와 같은 관련 주제는 각각 종래의 문헌들(E. Ahmed et al, Y. Mao et al)을 통해 설명될 수 있다.

[0011] 상태 저장 라이브 컨테이너 마이그레이션은 다음 두 가지 이유로 어려울 수 있다.

[0012] 하나는, 컨테이너화된 가상 플랫폼마다 고유한 아키텍처가 있으며 상태 저장 마이그레이션을 위한 표준 또는 공통 절차가 없다는 것이다. 따라서, Stateful 마이그레이션을 지원하려면 해당 플랫폼에 대한 심층적인 이해가 필요하다. 일반적으로, 상태 저장 마이그레이션에는 로컬 파일 시스템에 저장된 지속 파일(persistent files)과 실행 상태(예: CPU 컨텍스트, 메모리 내용 등)를 소스 에지 서버 ES(src)에서 대상 에지 서버 ES(tgt)로 전송하는 것이 포함된다. 여기서, ES(src)는 사용자가 서비스를 받는 에지 서버 호스트이고 ES(tgt)는 사용자의 서비스가 마이그레이션될 에지 서버 호스트이다.

[0013] 다른 하나는, 라이브 컨테이너 마이그레이션이 서비스 중단을 최소화함으로써, 사용자가 진행 중인 마이그레이션을 알아차리지 못해야 한다는 것이다.

[0014] 위의 두 가지가 없으면 원활하고 투명한 마이그레이션이 불가능할 수 있고, 다음을 통해 해당 목표가 달성될 수 있다.

[0015] - ES(src)에서 ES(tgt)로 상태를 전송하는 상태 전송 마이그레이션, 또는

[0016] - 대상 에지 서버를 점진적으로 구축하여 해당 상태가 소스 에지 서버와 동기화되도록 하는 상태 재구축 마이그레이션.

[0017] Puliafito et al.은 마이그레이션 시간, 서비스 중단 시간 및 전송된 데이터 양 측면에서 사전 복사, 사후 복사 및 하이브리드 마이그레이션 성능을 비교한다. 즉, 사전 복사는 ES(src)를 중지하기 전에 대부분의 상태(즉, 더티 페이지)를 반복적으로 전송하는 반면 사후 복사는 사전 복사의 양을 최소화하고 ES(tgt)에서 요청하면 더티 페이지를 전송한다. 하이브리드는 이 둘의 조합이다. 이 문서에서 사용된 도구는 체크포인트를 위한 CRIU, 파일 전송을 위한 rsync, 컨테이너 런타임을 위한 runC이다. 다만, 저자들은 일부 응용 프로그램에서 디스크에 쓰기가 허용되지 않음을 전제로 지속 파일(persistent files)이 전송되지 않도록 했다.

[0018] Karhula et al. 는 체크포인트를 위해 Docker 및 CRIU를 사용하여 IoT 에지 기능의 마이그레이션을 시연했다. 그러나, 그들이 제안하는 방법은 ES(src)와 ES(tgt) 사이에 지속 파일이 동기화되지 않고 체크포인트가 만들어진 후 변경된 상태에 대한 고려가 없다는 점에서 제한적이다.

[0019] Nadgowdaet al. 은 Docker 플랫폼에서 상태 저장 마이그레이션을 제안했다. 제안된 Voyager는 메모리 상태를 ES(tgt)로 전달한다. 또한, 로컬 파일 시스템 마이그레이션 시 서비스 다운타임을 최소화하기 위해 네트워크 파일 시스템을 이용한 듀얼 밴드 데이터 전송을 제안한다. 그러나 네트워크 연결 저장소를 사용할 수 없는 경우 마이그레이션된 서비스는 백그라운드에서 복사되는 파일에 대한 잦은 장애로 인해 서비스 품질(QoS)이 크게 저하될 수 있다.

[0020] Dupont et al. 은 Cloud4IoT라는 마이그레이션 오케스트레이션 시스템을 제안했다. 해당 시스템은 Docker와 Kubernetes를 사용하여 IoT 기능의 수평 및 수직 마이그레이션을 수행할 수 있다. 다만, Cloud4IoT의 한계는 서비스가 상태 비저장이라는 기본 가정이고, 따라서 Cloud4IoT는 상태를 전송하지 않으며 Stateful 마이그레이션이 필요한 경우 사용될 수 없다는 문제가 있다.

[0021] Al-Dhuraibi et al은 ElasticDocker라는 Docker 컨테이너용 자동 수직 확장 시스템을 제안했다. 서비스 마이그레이션과는 다르지만 ElasticDocker는 호스트에 스케일업을 위한 리소스가 남아 있지 않을 때 라이브 마이그레이션을 수행한다. ElasticDocker의 라이브 마이그레이션은 먼저 파일 시스템 차이를 전송한 다음 메모리 상태를 전송한다. 라이브 마이그레이션은 본 개시에 따라 제안될 FC와 유사하다. 다만, ElasticDocker의 한계는 최종 메모리 덤프 단계에서 컨테이너를 고정한다는 것이다. ES(tgt)의 컨테이너가 시작되기 전에 사용자가 요청을 보내면 요청이 손실되어 QoS가 저하될 수 있다.

[0022] Ma et al. 는 Docker의 계층화된 스토리지를 활용한 컨테이너 라이브 마이그레이션을 제안했다. 그들이 제안한 방법은 세 단계로 요약될 수 있다. 1) 이미지 레이어 동기화: 서로 다른 이미지 레이어가 ES(tgt)로 전송된다. 2) 메모리 차이 전송: 이 단계는 일관된 메모리 상태에 대한 체크포인트를 전송한다. 3) 컨테이너 중지 및 컨테이너 계층 동기화: 이 단계는 쓰기 가능한 컨테이너 계층을 동기화한다. 즉, 파일/디렉토리에 대한 사용자의 수정 사항이 ES(tgt)로 전송된다. 이 작업의 한계는 두 가지이다. 사전 덤프 컨테이너 단계에서 메모리 스냅샷은 모든 잠재적 대상 서버에 전송되며 불필요한 네트워크 부하가 발생할 수 있다. 또한, 마이그레이션의 나중 단계에서는 ES(tgt)가 서비스를 제공할 준비가 되기 전에 ES(src)를 종료한다. 이로 인해, 짧은 기간의 서비스 중단이 발생하거나 최악의 경우 ES(tgt)가 결함 상태가 될 때 영구적인 서비스 다운이 발생할 수 있다.

[0023] Yu et al. 은 로깅 및 리플레이를 사용한 컨테이너 마이그레이션을 제안했다. 제안하는 3단계 마이그레이션은 1) 컨테이너 레이어를 포함한 전체 이미지 내보내기 및 전송, 2) 이미지 전송이 진행되는 동안 변경 사항을 기록하고 ES(tgt)에서 변경 사항 리플레이, 3) ES(src) 및 ES(tgt)를 재개, 이다. 마이그레이션 체계는 지속 파일(persistent files)을 마이그레이션할 수 있지만 일부 메모리 페이지는 마이그레이션되지 않는다. 이는 상태 저장 응용 프로그램에 대한 제한 사항일 수 있다.

[0024] 가상 머신(VM) 환경에 대해서도 유사한 접근 방식이 제안되었다. Liu et al. 은 실행 중인 VM의 체크포인트를 먼저 전송하는 VM 마이그레이션 방식을 제안했다. 대상 호스트의 VM이 소스 호스트와 일관된 상태가 될 때까지 시스템 이벤트가 기록되고 대상 호스트에서 리플레이 된다. 그러나, 이 접근 방식은 소스 호스트의 로컬 파일 시스템에 있는 지속 파일(persistent files)의 전송을 고려하지 않는다. 또한 일부 응용 프로그램에서는 로깅-및-리플레이 자체가 체크 포인트와 로깅-및-리플레이의 조합보다 더 나은 솔루션이 될 수 있다.

[0025] 이전 문헌에서 볼 수 있듯이 상태 저장 마이그레이션에는 상태 이전 또는 상태 재구축과 같은 다양한 방법이 있다. 또한 서비스마다 속성이 다르며 이에 대해서는 후술한다. 그러나, 앞서 언급한 연구들 중 어느 것도 서비스 중단과 네트워크 혼잡을 최소화하기 위해 마이그레이션 시간과 네트워크 부하 측면에서 최적의 마이그레이션 기

법을 선택하는 것을 공동으로 고려하지 않았다.

[0026] 여러 서비스가 제한된 수의 에지 서버에서 실행될 수 있으므로 가상화는 에지 컴퓨팅을 가능하게 하는 핵심 구성 요소이다. 따라서 리소스 활용을 극대화하기 위해서는 리소스 격리(resource isolation) 및 멀티 테넌시(multi-tenancy)가 필수적이다. 가상 머신 및 컨테이너와 같은 다양한 가상화 기술 중에서 컨테이너화는 고성능 및 경량 특성으로 인해 에지 컴퓨팅 영역에서 더 대중화되었다. Docker가 유일한 컨테이너화된 가상화 솔루션(예: LXC 및 OpenVZ)은 아니지만 약 25%의 큰 시장 점유율(예: 약 25%)로 널리 사용된다는 점에서 본 개시에서 주로 다루어진다. 또한 Docker 컨테이너는 단일 보드 컴퓨터에서도 기본에 가까운 성능으로 작동하는 것으로 나타났다(R. Morabito). 그러나 Docker의 한 가지 단점은 다른 컨테이너화 솔루션보다 더 많은 리소스를 소비한다는 것이다(O. Oleghe). 라이브 컨테이너 마이그레이션 및 구현에서 핵심적인 역할을 하는 몇 가지 Docker 기능을 소개한다.

[0027] 컨테이너는 애플리케이션과 그것의 종속성을 포함하는 완전한 런타임 환경이다. 특히 Docker는 namespace를 사용하여 서로 다른 컨테이너 간의 리소스를 분리하고 CPU 및 메모리와 같은 리소스를 제어/모니터링하기 위해 제어 그룹(cgroup)을 사용한다. 사용자 공간에서 격리된 프로세스로 실행되는 컨테이너는 운영 체제 커널을 호스트 시스템과 공유한다. 커널을 공유함으로써 컨테이너는 경량화될 수 있으며, 이는 VirtualBox 및 VMWare와 같은 가상 머신 소프트웨어와 구별되는 기능이다.

[0028] Docker 컨테이너는 애플리케이션 또는 서비스를 구성하는 읽기 전용 계층의 집합인 이미지에서 생성된다. Docker 이미지는 공용(예: Docker Hub) 또는 개인 저장소에서 다운로드하거나 사용자 정의 Dockerfile 스크립트에서 빌드할 수 있다. 컨테이너가 이미지에서 시작되면 컨테이너 레이어라는 쓰기 가능한 레이어가 최상위 레이어로 추가된다. 모든 변경 사항은 기록 중 복사 방식으로 컨테이너 계층에 저장된다. Docker는 이미지 레이어(예: overlay2, btrfs 및 aufs)를 저장하기 위해 다양한 스토리지 드라이버를 지원하며 오버레이2가 공식적으로 권장되고 있다. 예를 들어, 쓰기 가능한 컨테이너 계층은 오버레이2 드라이버에서 상위 계층으로 표시되며 파일 시스템의 차이점 또는 변경 사항(즉, 수정, 추가 또는 삭제된 파일/디렉토리)을 포함한다. 즉, 이미지와 다른 목록 또는 파일 및 디렉토리는 Docker diff 명령을 사용하여 사용할 수 있는 상위 디렉토리에서 찾을 수 있다.

[0029] Docker는 또한 컨테이너의 STDOUT 및 STDERR을 로그로 기록한다. 로그는 기본적으로 JSON 형식으로 저장된다. 로그 파일은 docker 검사 결과에서 경로가 있는 LogPath 디렉토리에서 찾을 수 있다. 또한 docker logs 명령을 사용하여 로그를 검색할 수 있다. 이는 주로 사용자가 실행하고 컨테이너에 적용된 작업의 내역을 추적하는 데 특히 유용하다.

[0030] CRIU는 Linux 프로세스의 상태를 검사하고 복원하는 유틸리티이다. CRIU는 메모리 상태와 프로세스 상태, 오픈 파일, 네트워크 소켓 등을 캡처하고 파일 모음으로 덤프한다. 체크포인트의 크기는 프로세스의 페이지맵 크기에 따라 다르다. Dockers의 실험적 체크포인트 기능[36]은 CRIU에 의존하며, 이는 ES(tgt)의 휘발성 상태가 ES(src)의 휘발성 상태와 동일하도록 마이그레이션하는 데 핵심적인 역할을 한다.

발명의 내용

해결하려는 과제

[0031] 본 개시는, 시간과 네트워크 로드 모두에 효율적인 방식으로 서비스를 마이그레이션하기 위해 마이그레이션할 서비스 응용 프로그램과 다른 마이그레이션 기술의 속성을 함께 고려하는 모바일 엣지 컴퓨팅 시스템을 제공한다.

[0032] 본 개시는 Full-Copy(FC), Differential-Copy(DC) 및 Log-Replay(LR) 마이그레이션이라는 라이브 및 상태 저장 서비스 마이그레이션을 달성하기 위한 세 가지 컨테이너 마이그레이션 기술을 제안한다.

[0033] 또한, 본 개시에 따른 마이그레이션 기술은 마이그레이션 중 리플레이 버퍼 및 패킷 릴레이 기술을 사용함으로써 서비스 중단을 최소화하여, 원활하고(seamless) 상태 유지에 용이한(stateful) 방식으로 컨테이너를 일 에지 서버에서 다른 에지 서버로 재배치할 수 있다. 또한, 널리 사용되는 컨테이너 기반 가상화 시스템인 docker 플랫폼에서 실제 구현을 위한 시스템 설계를 제안한다.

[0034] 마이그레이션 시간을 최소화하는 것에 더하여 네트워크 트래픽 양이 줄어드는 것도 중요하기 때문에, 두 가지가 동시에 고려된 최적의 마이그레이션 기법 선택 알고리즘이 차용된다.

[0035] 본 개시의 목적들은 이상에서 언급한 목적으로 제한되지 않으며, 언급되지 않은 본 개시의 다른 목적 및 장점들

은 하기의 설명에 의해서 이해될 수 있고, 본 개시의 실시 예에 의해 보다 분명하게 이해될 것이다. 또한, 본 개시의 목적 및 장점들은 특허 청구 범위에 나타낸 수단 및 그 조합에 의해 실현될 수 있음을 쉽게 알 수 있을 것이다.

과제의 해결 수단

[0036] 본 개시의 일 실시 예에 따른 모바일 엣지 컴퓨팅 시스템은, 제1 AP(Access Point), 상기 제1 AP를 통해 적어도 하나의 단말과 연동되어 서비스를 제공하는 제1 ES(Edge Server), 제2 AP, 상기 제2 AP를 통해 적어도 하나의 단말과 연동되어 서비스를 제공하는 제2 ES, 코어 네트워크에 포함된 적어도 하나의 서버를 포함한다. 상기 서버는, 상기 제1 ES에 포함된 컨테이너의 동작에 따라 사용자 단말로 서비스가 제공되는 동안 핸드오프 이벤트가 감지되면, DC(Differential-Copy) 및 LR(Log-Replay) 중 하나의 마이그레이션 절차를 선택하고, 상기 선택된 마이그레이션 절차에 따라, 상기 제1 ES에서 동작 중인 상기 컨테이너를 상기 제2 ES로 전달하는 마이그레이션을 진행한다.

[0037] 상기 DC는, 상기 제1 ES가 상기 컨테이너의 쓰기 가능한 계층을 상기 제2 ES로 전송하고, 상기 제1 ES가 체크포인트를 생성하여 상기 제2 ES로 전송하며, 상기 제2 ES는 상기 쓰기 가능한 계층 및 상기 체크포인트를 기반으로 상기 컨테이너를 최신 상태에서 재개할 수 있다.

[0038] 상기 LR은, 상기 제1 ES가 상기 제1 ES에서 실행된 command trace를 상기 제2 ES로 전송하고, 상기 제2 ES가 상기 command trace를 리플레이 하여 상기 컨테이너를 최신 상태에서 재개할 수 있다.

[0039] 상기 서버는, 이하 수학식에 따라, 상기 DC 및 상기 LR 중 하나의 마이그레이션 절차를 선택할 수 있다. $\min_x f(x)=\frac{1}{\alpha}\{x\cdot T_{DC}+(1-x)T_{LR}\}+w\frac{1}{\beta}\{x\cdot L_{DC}+(1-x)L_{LR}\}$. 상기 T_{DC}는 상기 DC의 마이그레이션 시간이고, 상기 T_{LR}은 상기 LR의 마이그레이션 시간이고, 상기 L_{DC}는 상기 DC 실행의 결과로 발생하는 네트워크 부하이고, 상기 L_{LR}은 상기 LR 실행의 결과로 발생하는 네트워크 부하이고, 상기 w는 네트워크 부하에 대한 가중치를 나타내는 음이 아닌 디자인 파라미터이다. 상기 수학식에서, x는 이하 범위로 설정될 수 있다. $x\in\{0,1\}$

[0040] 본 개시의 일 실시 예에 따른 모바일 엣지 컴퓨팅 시스템의 제어 방법은, 제1 ES(Edge Server)가 제1 AP(Access Point)를 통해 사용자 단말로 서비스를 제공하는 단계, 상기 사용자 단말과 관련된 핸드오프 이벤트가 감지되면, 코어 네트워크의 서버가 DC(Differential-Copy) 및 LR(Log-Replay) 중 하나의 마이그레이션 절차를 선택하는 단계, 상기 서버가, 상기 선택된 마이그레이션 절차에 따라, 상기 제1 ES에서 동작 중인 컨테이너를 제2 AP와 매칭되는 제2 ES로 전달하는 마이그레이션을 진행하는 단계, 상기 마이그레이션이 수행되면, 상기 제2 ES가 제2 AP를 통해 상기 사용자 단말로 상기 서비스를 제공하는 단계를 포함한다.

[0041] 본 개시의 일 실시 예에 따라 모바일 엣지 컴퓨팅 시스템에 포함되는 서버의 제어 방법은, 제1 ES(Edge Server)가 제1 AP(Access Point)를 통해 사용자 단말로 서비스를 제공하는 동안, 상기 사용자 단말과 관련된 핸드오프 이벤트를 감지하는 단계, 상기 핸드오프 이벤트가 감지되면, DC(Differential-Copy) 및 LR(Log-Replay) 중 하나의 마이그레이션 절차를 선택하는 단계, 상기 선택된 마이그레이션 절차에 따라, 상기 제1 ES에서 동작 중인 컨테이너를 제2 AP와 매칭되는 제2 ES로 전달하는 마이그레이션을 진행하는 단계를 포함한다.

발명의 효과

[0042] 본 개시에 따른 모바일 엣지 컴퓨팅 시스템의 효과는 이하 설명된다.

[0043] 본 개시에 따라 엣지 컴퓨팅용 세 가지 라이브 컨테이너 마이그레이션 기술이 제공된다. 본 개시에 따른 모바일 엣지 컴퓨팅 시스템은 지속 파일(persistent file)과 휘발성 상태(예: CPU 컨텍스트 및 메모리 상태)를 모두 마이그레이션하여 상태 기반 마이그레이션을 달성할 수 있다. 또한, 마이그레이션 중 패킷 릴레이와 리플레이 버퍼를 사용하여 서비스 다운타임을 0으로 줄이고 소스 및 대상 에지 서버 모두 효율적인 방식으로 상태를 동기화할 수 있다.

[0044] 본 개시에 따른 모바일 엣지 컴퓨팅 시스템은, 세 가지 마이그레이션 기술의 특성 및 마이그레이션할 컨테이너화된 서비스의 주요 속성에 따라 마이그레이션 시간 및 네트워크 부하 측면에서 우수한 마이그레이션 기술을 선택적으로 활성화할 수 있다.

[0045] 특히, 본 개시에 따라 널리 사용되는 Docker 플랫폼에서 마이그레이션 기술을 구현하는 방법에 대한 실용적이고

기술적인 세부 정보가 소개된다.

[0046] 본 개시에 따른 모바일 엣지 컴퓨팅 시스템은, 마이그레이션하고자 하는 서비스의 속성, 서로 다른 마이그레이션 기법의 특성, 예상 마이그레이션 시간 및 예상 트래픽 발생량 등이 함께 반영된 최적의 마이그레이션 기법 선택 알고리즘을 활용한다.

[0047] 본 개시에 따른 모바일 엣지 컴퓨팅 시스템은, 최적의 마이그레이션 선택과 마이그레이션 절차를 자율적으로 수행할 수 있다.

[0048] 본 개시에 따른 모바일 엣지 컴퓨팅 시스템은, 기성 컴퓨팅 장치에서 제안된 마이그레이션 기술과 자동화된 최적의 마이그레이션 선택 시스템을 포함하여 구현된다.

[0049] 후술할 실증적 평가에 따라 세 가지 마이그레이션 기법이 서로 비교되고, 본 개시에 따른 모바일 엣지 컴퓨팅 시스템이 서비스 중단을 최소화하는 방식으로 서비스를 효과적이고 효율적으로 마이그레이션할 수 있음이 설명된다.

도면의 간단한 설명

[0050] 도 1은 모바일 사용자가 개별적으로 AP에 연결되는 형태의 모바일 엣지 컴퓨팅 시스템의 구성을 설명하기 위한 도면,

도 2는 엣지 서버 간 컨테이너화된 서비스의 핸드오프 과정을 개략적으로 설명하기 위한 도면,

도 3은 본 개시의 일 실시 예에 따른 모바일 엣지 컴퓨팅 시스템이 FC(Full-Copy) 마이그레이션을 수행하는 동작을 설명하기 위한 흐름도,

도 4는 본 개시의 일 실시 예에 따른 모바일 엣지 컴퓨팅 시스템이 DC(Differential-Copy) 마이그레이션을 수행하는 동작을 설명하기 위한 흐름도,

도 5는 본 개시의 일 실시 예에 따른 모바일 엣지 컴퓨팅 시스템이 LR(Log-Replay) 마이그레이션을 수행하는 동작을 설명하기 위한 흐름도,

도 6은 본 개시의 일 실시 예에 따른 모바일 엣지 컴퓨팅 시스템에 포함되는 각 장치의 기능적 구성을 설명하기 위한 블록도,

도 7은 본 개시의 일 실시 예에 따른 AC-LF/NC-L 시나리오 상에서 FC, DC, LR 각각에 따른 마이그레이션의 응답 시간을 도시한 그래프들,

도 8은 본 개시의 일 실시 예에 따른 AC-LF/NC-L 시나리오 상에서 FC, DC, LR의 요청 별 응답 시간의 평균을 도시한 그래프,

도 9은 본 개시의 일 실시 예에 따른 AC-LF/NC-H 시나리오 상에서 FC, DC, LR 각각에 따른 마이그레이션의 응답 시간을 도시한 그래프들,

도 10은 본 개시의 일 실시 예에 따른 AC-LF/NC-H 시나리오 상에서 FC, DC, LR의 요청 별 응답 시간의 평균을 도시한 그래프,

도 11은 본 개시의 일 실시 예에 따른 AC-LF/NC-H 시나리오 상에서 선택된 마이그레이션(x=0 or 1) 각각에 대하여 가중치 w에 대한 목적 값의 분포를 도시한 그래프,

도 12는 AC-LS/NC-L 시나리오 및 AC-LS/NC-H 시나리오 각각에 있어 DC 및 LR의 요청 별 응답 시간의 평균을 도시한 그래프들,

도 13은 AC-LS/NC-L 시나리오 및 AC-LS/NC-H 시나리오 각각에 있어 DC 및 LR의 가중치 w에 대한 목적 값의 분포를 도시한 그래프들,

도 14는 AC-HF/NC-L, AC-HF/NC-H, AC-HS/NC-L, AC-HS/NC-H 시나리오 각각에 있어 DC 및 LR의 요청 별 응답 시간의 평균을 도시한 그래프들, 그리고

도 15는 AC-HF/NC-L, AC-HF/NC-H, AC-HS/NC-L, AC-HS/NC-H 시나리오 각각에 있어 가중치 w에 대한 목적 값의 분포를 도시한 그래프들이다.

발명을 실시하기 위한 구체적인 내용

[0051] 본 개시에 대하여 구체적으로 설명하기에 앞서, 본 명세서 및 도면의 기재 방법에 대하여 설명한다.

[0052] 먼저, 본 명세서 및 청구범위에서 사용되는 용어는 본 개시의 다양한 실시 예들에서의 기능을 고려하여 일반적인 용어들을 선택하였다. 하지만, 이러한 용어들은 당해 기술 분야에 종사하는 기술자의 의도나 법률적 또는 기술적 해석 및 새로운 기술의 출현 등에 따라 달라질 수 있다. 또한, 일부 용어는 출원인이 임의로 선정한 용어도 있다. 이러한 용어에 대해서는 본 명세서에서 정의된 의미로 해석될 수 있으며, 구체적인 용어 정의가 없으면 본 명세서의 전반적인 내용 및 당해 기술 분야의 통상적인 기술 상식을 토대로 해석될 수도 있다.

[0053] 또한, 본 명세서에 첨부된 각 도면에 기재된 동일한 참조번호 또는 부호는 실질적으로 동일한 기능을 수행하는 부품 또는 구성요소를 나타낸다. 설명 및 이해의 편의를 위해서 서로 다른 실시 예들에서도 동일한 참조번호 또는 부호를 사용하여 설명한다. 즉, 복수의 도면에서 동일한 참조 번호를 가지는 구성요소를 모두 도시되어 있다고 하더라도, 복수의 도면들이 하나의 실시 예를 의미하는 것은 아니다.

[0054] 또한, 본 명세서 및 청구범위에서는 구성요소들 간의 구별을 위하여 "제1", "제2" 등과 같이 서수를 포함하는 용어가 사용될 수 있다. 이러한 서수는 동일 또는 유사한 구성요소들을 서로 구별하기 위하여 사용하는 것이며 이러한 서수 사용으로 인하여 용어의 의미가 한정 해석되어서는 안 된다. 일 예로, 이러한 서수와 결합된 구성요소는 그 숫자에 의해 사용 순서나 배치 순서 등이 제한되어서는 안 된다. 필요에 따라서는, 각 서수들은 서로 교체되어 사용될 수도 있다.

[0055] 본 명세서에서 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함하다" 또는 "구성되다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0056] 본 개시의 실시 예에서 "모듈", "유닛", "부(part)" 등과 같은 용어는 적어도 하나의 기능이나 동작을 수행하는 구성요소를 지칭하기 위한 용어이며, 이러한 구성요소는 하드웨어 또는 소프트웨어로 구현되거나 하드웨어 및 소프트웨어의 결합으로 구현될 수 있다. 또한, 복수의 "모듈", "유닛", "부(part)" 등은 각각이 개별적인 특정한 하드웨어로 구현될 필요가 있는 경우를 제외하고는, 적어도 하나의 모듈이나 칩으로 일체화되어 적어도 하나의 프로세서로 구현될 수 있다.

[0057] 또한, 본 개시의 실시 예에서, 어떤 부분이 다른 부분과 연결되어 있다고 할 때, 이는 직접적인 연결뿐 아니라, 다른 매체를 통한 간접적인 연결의 경우도 포함한다. 또한, 어떤 부분이 어떤 구성요소를 포함한다는 의미는, 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다.

[0058] 이하에서, 첨부된 도면을 이용하여 본 발명의 다양한 실시 예들에 대하여 구체적으로 설명한다.

[0059] 본 개시에 따라 제안되는 기술적 사상은, 컨테이너화된 서비스 마이그레이션에 대한 최초의 포괄적인 연구에 해당한다. 구현 세부 사항과 함께 세 가지 상태 이전 또는 상태 재구축 마이그레이션 솔루션이 제안된다. 또한, 마이그레이션할 서비스의 특성을 고려하여 최적의 마이그레이션 기법을 선택하는 저비용 알고리즘이 활용된다.

표 1

[0060]

AP	Access Point
ES	Edge Server
MU	Mobile User
QoS	Quality of Service
RTT	Round Trip Time
ES	Edge Server (or Edge computing Server)
ES(src)	ES from which the running container is migrated
ES(tgt)	ES to which the running container on ES(src) is migrated
DC	Proposed Differential-Copy migration technique
FC	Proposed Full-Copy migration technique
LR	Proposed Log-Replay migration technique

[0061] 컨테이너를 ES(src)에서 ES(tgt)로 마이그레이션할 때, ES(src)와 ES(tgt)가 동일한 지속 상태(즉, 파일 시스템의 파일 및 디렉토리) 및 휘발성 상태(즉, 메모리 레이아웃, 네트워크 소켓, 오픈 파일 등)를 갖는 경우, 상태 저장 마이그레이션(stateful migration)이 수행될 수 있다. 이를 위해 상태를 ES(tgt)로 전송하거나 ES(tgt)의 상태를 ES(src)와 동일하게 만드는 일련의 작업을 실행하는 세 가지 마이그레이션 기술이 제안된다.

[0062] AP에 ES를 배치하면 서비스 지연이 줄어들고 네트워크 혼잡이 방지된다. 따라서 본 개시의 실시 예에 따르면 ES는 AP와 함께 위치한다. 일 실시 예로, 모바일 사용자는 이더넷 케이블을 통해 코어 네트워크에 유선으로 연결된 IEEE 802.11 AP를 통해 네트워크에 액세스할 수 있다. AP의 커버리지에 진입한 사용자는 무선 연결을 위해 AP와 연결하고, 사용자가 다른 AP와 연결되어 있으면 핸드오프 절차를 트리거한다.

[0063] AP-n과 연결된 사용자는 AP-n과 같은 위치에 있는 ES-n에서 실행되는 전용 컨테이너에서 계산 오프로딩 서비스를 받는다. 사용자가 다른 AP로 핸드오프 하면 컨테이너 마이그레이션이 트리거되고 컨테이너를 새 AP로 이동하여 오프로딩 서비스에 대한 네트워크 지연을 최소화할 수 있다.

[0064] 한편, 이하 내용이 추가로 가정될 수 있다. 구체적으로, ES(src)에서 ES(tgt)로의 마이그레이션이 트리거되면 널리 사용 가능한 Docker 이미지(예: 공개 또는 비공개 저장소에서 액세스할 수 있는 이미지)가 ES(tgt)에서 로컬로 사용 가능(또는 pre-cached)하게 된다[34]. 그러나 커스터마이징된 이미지를 실행하는 ES의 경우 이러한 가정이 성립하지 않으며, 후술할 FC를 사용하여 처리할 수 있다. ES 호스트는 유사한 컴퓨팅 성능을 가지고 있으며 기본 리소스 제한 구성(Runtime options with Memory, CPUs, and GPUs [Online]. Available: https://docs.docker.com/config/containers/resource_constraints, Accessed on: Sep. 24, 2021.)은 모든 Docker 컨테이너에 적용될 수 있다.

[0065] 본 개시의 일 실시 예에 따른 모바일 엣지 컴퓨팅 시스템의 마이그레이션 기술은, ES(src) 및 ES(tgt)를 동기화시켜, 결국 ES(tgt)가 ES(src)와 동일한 지속 상태 및 휘발성 상태를 갖도록 한다. 제안되는 마이그레이션은 크게 상태 복제(state duplication)와 상태 재생(state reproduction)의 두 가지 방법으로 분류될 수 있다. 상태 복제는 ES(src) 전송에서 ES(tgt)로 상태를 복사하여 ES(tgt)가 ES(src)의 가장 최근 상태에서 시작할 수 있도록 한다. 반면에 상태 재생은 명령 실행 log/trace를 ES(tgt)에 전달하고 ES(tgt)는 ES(src)에 일관된 상태의 컨테이너를 재생할 수 있다.

[0066] 본 개시에서 사용되는 예시로 도 2와 같은 시나리오를 가정할 수 있다. AP-1과 관련된 사용자는 ES-1에서 실행되는 컨테이너에서 서비스를 받고 있다. 사용자가 AP-1에서 멀어지면 컨테이너화된 서비스 마이그레이션이 시작되는 AP-2와 다시 연결된다.

[0067] 컨테이너 마이그레이션은 앞서 설명된 컨트롤러로부터 해당 메시지를 수신하여 트리거된다. 컨트롤러는 ES(src) 및 ES(tgt)에 서로 다른 메시지를 전송할 수 있다, 구체적으로, ES(tgt)의 ID(즉, IP 주소)를 ES(src)로, 기본 이미지의 이름을 ES(tgt)로 전송할 수 있다. 기본 이미지 이름은, 전체 이미지를 전송하는 FC에서는 사용되지 않지만, DC와 LR 모두에서, 이미지가 ES(tgt)에 로컬로 캐시되었는지 확인하는 데 사용될 수 있다.

[0068] **1) Full-Copy Migration, FC**

[0069] FC는 본 개시의 일 실시 예에 따라 주로 성능 기준으로 사용되는 네이브한(naive) 상태 복제 방법이다. AP-1의 ES(src)에 있는 컨테이너가 AP-2의 ES(tgt)로 마이그레이션되면, 모든 컨테이너 계층(즉, 쓰기 가능 계층 및 읽기 전용 계층)과 실행 상태가 ES(tgt)로 전송되므로 일관된 상태의 컨테이너는 전송된 것을 사용해야만 ES(tgt)에서 시작할 수 있다. FC 마이그레이션의 전체 절차는 도 3에 도시되어 있다.

[0070] 마이그레이션이 트리거되면, ES(tgt)로 마이그레이션 될(실행 중인) 컨테이너가 내보내어지고 쓰기 가능한 레이어와 읽기 전용 레이어를 모두 포함하는 새 이미지로 저장될 수 있다. 이 작업에는 Docker commit 및 save 명령이 사용될 수 있다. 컨테이너의 지속 상태를 포함하는 새 이미지는 ES(tgt)로 전송된다. 다음 단계는 실행 상태를 전달하는 것으로, 체크포인팅을 수행하여 ES(tgt)로 전송할 수 있다. ES(src)에서 checkpoint create 명령을 사용하여 체크포인트가 만들어지면, ES(src)는 미리 정의된 경로나 사용자가 정의한 경로에 바이너리 파일 집합을 생성한다. 모든 파일 전송은 보안 복사 프로토콜 도구인 scp를 사용하여 수행될 수 있다.

[0071] 컨테이너를 내보내고 검사할 때, --pause=false 및 --leave-running=true 옵션을 사용하여 컨테이너가 계속 실행됨에 따라, ES(tgt)에서 복제본이 준비될 때까지 사용자에게 지속적으로 서비스가 제공될 수 있다. MU가 AP-2로 핸드오프하는 순간부터 컨테이너가 ES(tgt)에서 준비될 때까지 무시할 수 없는 시간이 흐르며 그 동안 컨테

이너의 상태가 변경될 수 있다. 변경 사항을 추적하기 위해, 제안된 마이그레이션은 패킷 릴레이 및 버퍼링 기술을 사용할 수 있다.

[0072] 사용자가 AP-2로 핸드오프한 직후 해당 서빙 컨테이너는 ES(src)에서 작동하는 반면 사용자는 통신을 위해 AP-2에 액세스할 수 있다. ES(tgt)의 컨테이너의 서비스 제공 준비가 될 때까지 AP-2는 MU의 메시지를 AP-1에 중계하여 다음을 수행한다. ES(src)는 서비스를 제공할 수 있다. 그 동안 AP-2는 동일한 사용자 요청을 로컬로 버퍼링 할 수 있다. ES(tgt)의 컨테이너가 시작되면 버퍼링된 모든 요청이 해당 컨테이너에서 리플레이될 수 있다. 버퍼가 비게 되면 ES(tgt)는 AP-2에 준비 상태를 알린다. 그러면 AP-2는 자신의 로컬 ES, 즉 ES(tgt)가 사용자에게 서비스를 제공할 수 있도록 패킷 릴레이 및 버퍼링을 중지할 수 있다. 또한 AP-2는 ES(src)가 시스템 자원을 중지하고 해제할 수 있도록 하는 서비스 중지 요청을 ES(src)로 보낼 수 있다. 대기열에 대기 작업이 있는 경우 ES(src)는 유예 기간이라고 하는 특정 시간 동안 컨테이너 중지를 지연할 수 있다.

[0073] **2) Differential-Copy Migration, DC**

[0074] DC는 읽기 전용 계층을 전송하지 않으므로 효율적인 상태 복제 방법이다. 감소된 데이터 전송량은 마이그레이션 시간을 단축할 뿐만 아니라 FC에 비해 네트워크 혼잡을 완화한다. 앞서 언급했듯이 컨테이너는 상단에 쓰기 가능한 레이어와 아래에 하나 이상의 읽기 전용 레이어로 구성된다. 컨테이너가 시작된 기본 이미지가 ES(tgt)에 로컬로 캐시된 경우, 쓰기 가능 계층(writable layer)을 마이그레이션해야만 지속 상태 측면에서 일관된 상태가 획득될 수 있다. 체크포인팅은 DC에서 휘발성 상태를 ES(tgt)로 전송하는 데에도 사용된다. DC의 전체 절차는 도 4에 도시되어 있다. 도 4는, 컨테이너화된 서비스를 마이그레이션할 때 기본 이미지가 ES(tgt)에 미리 캐시되어 있거나 또는 가까운 저장소에서 즉시 가져올 수 있는 상황임을 가정한다.

[0075] 컨트롤러에서 해당 메시지를 수신하여 마이그레이션이 트리거되면 ES(src)는 파일 시스템 변경 사항(즉, 쓰기 가능 계층 내용)을 추출하고 ES(tgt)는 기본 이미지가 로컬에서 사용 가능한지 확인할 수 있다. 기본 이미지의 이름은 컨트롤러에 의해 ES(tgt)에 알려질 수 있다.

[0076] 쓰기 가능 계층의 콘텐츠는 두 가지 방법으로 추출될 수 있다. 하나는 docker diff 명령을 사용하여 변경 사항을 나열하고 docker cp로 추출하는 것이다. 그리고, 다른 하나는 ES 호스트의 파일 시스템에서 변경 사항이 저장되는 경로를 찾는 것이다. 경로는 UpperDir 레이블 값을 사용하여 docker 검사 결과에서 식별될 수 있다. 전자는 안전하지만 docker diff 결과를 파싱해야 하므로 본 개시에 따른 DC는 효율성을 위해 후자를 사용할 수 있다.

[0077] 쓰기 가능 계층 내용을 전송한 후 ES(src)는 체크포인트를 만들어 ES(tgt)로 전송할 수 있다. 둘 다 수신되면 ES(tgt)는 컨테이너를 최신 상태에서 재개할 수 있다. 쓰기 가능 계층이나 체크포인트에 포함되지 않은 모든 변경 사항은 패킷 리플레이 및 버퍼링 방법에 의해 처리될 수 있다.

[0078] DC는 상태를 ES(tgt)로 전송한다는 점에서 FC와 유사하다. 그러나 FC는 훨씬 더 큰 데이터를 전송하므로 DC보다 시간이 오래 걸리고 네트워크 부하가 더 크다.

[0079] **3) Log-Replay Migration, LR**

[0080] LR은 FC 및 DC와 다른 상태 재생 방식이다. ES(src)로부터 상태 관련 데이터를 받는 대신, ES(tgt)는 ES(src)에서 실행된 command trace을 수집하고 기본 이미지를 처음부터 시작하고 로컬에서 명령을 리플레이 할 수 있다. 따라서 ES(tgt)는 ES(src)가 시작된 동일한 기본 이미지에서 시작하고 ES(src)와 동일한 실행 이력을 따라 일관된 상태를 재구성할 수 있다. LR 마이그레이션의 전체 절차는 도 5에 도시되어 있다.

[0081] 제안되는 LR은, 커맨드 로그(command logs)를 추출하기 위해 docker 로그를 활용할 수 있다. docker logs 명령은 컨테이너의 STDOUT 및 STDERR 스트림을 표시한다. 온전한 전체 로그는 타임스탬프와 함께 JSON 형식의 로그 파일에서 획득될 수 있다. 이 경우, 각 사용자 요청에 해당하는 각 작업에 대해 시작과 끝이 STDOUT에 출력되도록 서비스가 구성될 수 있다. 어떠한 애플리케이션/서비스이건 래퍼 모듈(wrapper module) 내에서 실행되어 간단하게 수행될 수 있다. LogPath 레이블 값을 사용하여 docker inspect 결과에서 로그 파일의 경로가 검색될 수도 있다.

[0082] LR은 로그를 구문 분석하고 그 안의 명령을 추출하여 command trace을 검색할 수 있다. 그런 다음, command trace은 ES(tgt)에서 리플레이 될 수 있도록 ES(tgt)로 전송될 수 있다. 컨테이너가 ES(tgt)에서 준비되는 동안 패킷 리플레이 및 버퍼링은 전송된 command trace에 포함되지 않은 상태 변경을 캡처하는 데 사용될 수 있다. ES(tgt)의 컨테이너가 수신된 command trace에 대한 리플레이를 완료하면 버퍼링된 커맨드 로그(command log)를

리플레이하고 마지막으로 AP-2에 준비 상태를 알릴 수 있다.

[0083] 한편, LR은 각 명령 또는 사용자 요청에 걸리는 시간을 계산하는 등 최적의 마이그레이션 기술을 계산하는 데 사용될 수 있다. 자세한 내용은 후술한다.

[0084] **4) Packet Relay and Replay Buffer**

[0085] 패킷 리플레이(packet replay) 및 리플레이 버퍼(replay buffer)는 ES(src)에서 ES(tgt)로 전송된 내용에 포함되지 않은 마지막 순간의 변경 사항을 캡처하는 데 중요한 역할을 한다. FC는 완전한 컨테이너 이미지와 체크포인트를 전송하고 DC는 쓰기 가능한 레이어와 체크포인트를 전송하며 LR은 커맨드 로그(command log)를 전송한다. ES(tgt)의 컨테이너는 리플레이 버퍼가 비어 있고 대기 중인 모든 명령이 ES(tgt)에서 성공적으로 재생될 때만 AP-2에 준비 상태를 알릴 수 있다. 한편, AP-2가 수신한 MU의 서비스 요청은 AP-1의 ES(src)로 중계될 수 있다.

[0086] 상술한 패킷 릴레이 및 리플레이 버퍼 방식은 다음 두 가지 이유로 서비스 마이그레이션에 특히 유용하다. 일반적인 상태 저장 기반의(stateful) 컨테이너 마이그레이션 접근 방식은 미리 정의된 반복 횟수에 도달할 때까지 메모리 덤프의 반복적인 전송을 기반으로 한다(C. Puliafito et al). 반복 제한 후에 전송할 덤프가 남아 있으면 ES(tgt)에서 정확히 동일한 컨테이너를 복구하지 못할 수도 있다. 또한 전송할 변경된 상태(즉, 더티 페이지)의 양이 변경을 일으킨 명령 자체보다 클 수 있다. 그렇다면 대역폭과 시간 측면에서 더티 페이지를 보내는 것보다 커맨드 로그(command log)를 전송하는 것이 더 효율적일 수 있다.

[0087] 다른 하나는 서비스 가용성 및 안전 장치 속성과 관련이 있다. 일반적인 컨테이너 마이그레이션 기술은 ES(src)에서 컨테이너를 중지한 다음 마이그레이션 절차의 일부 지점에서 ES(tgt)에서 복제본을 재개할 수 있다. 그러나 이러한 접근 방식은 기간이 아무리 짧아도 서비스 다운타임이 발생한다. 또한 어떤 이유로 ES(tgt)에서 복제 컨테이너를 시작하지 못하면 컨테이너화된 서비스를 사용할 수 없게 되어 QoS 요구 사항을 위반하게 될 수 있다.

[0088] 반면에 제안된 마이그레이션 기술은 상술한 두 가지 문제를 효과적으로 극복할 수 있다. ES(src)의 컨테이너는 ES(tgt)의 복제본이 준비되고 완전히 작동할 때까지 서비스를 계속 제공하므로 결과적으로 서비스를 사용할 수 없는 기간이 없다. 또한 ES(tgt)가 컨테이너 복제본을 런칭하는 데 실패하면 ES(src)의 원본(컨테이너)이 서비스를 계속할 수 있다.

[0089] 제안된 세 가지 마이그레이션 기술은 특성과 사용 사례가 다르다. 마이그레이션 시간과 네트워크 부하 측면에서 효율적인 마이그레이션 기술을 선택하는 것은 각 마이그레이션 기술의 작동 방식과 마이그레이션할 컨테이너화된 서비스의 속성에 따라 다르다.

[0090] FC는 전송 중심의 마이그레이션 기술이며 기본 이미지 또한 전송하므로 가장 많은 양의 네트워크 트래픽을 생성한다. 겉보기에는 비효율적이지만, 마이그레이션할 컨테이너의 기본 이미지가 서비스 ES, 즉 ES(src) 외부에서 사용될 수 없을 때 작동하는 유일한 솔루션에 해당한다. 그러나, 그렇지 않은 경우 FC는 대역폭과 시간 측면에서 항상 DC보다 열등하다.

[0091] DC는 상태 복원이 오래 걸릴 때 특히 유용하다. 예를 들어, J. Hang et al을 통해 알려진 바와 같이 식물 잎 질병 감지 애플리케이션을 위한 Inception3 딥 러닝 모델을 훈련하는 데 약 2시간이 걸렸다. 그러나, 결과 모델의 크기는 약 90MB에 불과하다. 이러한 경우, ES(src)와 ES(tgt) 사이의 네트워크 지연이 작고 대역폭이 충분히 크다는 점을 전제로, 훈련된 모델, 즉 DC를 전송하는 것이 훈련 데이터 세트를 수신하여 ES(tgt)에서 모델을 훈련시키는 것(: LR)보다 시간과 대역폭 면에서 더 효율적이다.

[0092] 반면에 LR은 로그 재생에 의한 상태 복원이 변경된 상태(즉, 파일 시스템의 영구 파일, CPU/MEM 컨텍스트 등)를 전송하는 것보다 빠른 경우 다른 두 가지보다 효율적이다. 예를 들어, Augmentation(F. Chollet)은 훈련된 모델의 일반화 성능을 향상시키기 위해 이미지 관련 응용 프로그램에 대한 딥 러닝의 데이터 세트를 곱하는 일반적인 트릭이다. 이는 몇 줄의 코드(또는 명령)만으로 구현될 수 있다. Augmentation 기술은 회전, 이동, 전단, 확대/축소 및 뒤집기와 같은 간단한 선형 작업을 기반으로 하며 빠르게 수행될 수 있으며 추가로 방대한 데이터 세트를 생성할 수 있다. 이러한 경우 모든 증강 데이터를 전송하는(: DC) 대신 ES(tgt)에서 증강을 리플레이 함으로써 시간과 대역폭이 절약될 수 있다.

[0093] 이하 내용에서는, 마이그레이션 시간과 네트워크 부하 모두에 대해 최상의 마이그레이션 기법을 선택하는 최적의 마이그레이션 결정 알고리즘이 개시된다. 본 알고리즘은 마이그레이션 기술과 함께 마이그레이션할 애플리케

이션의 특성₩이 고려된 것이다. 또한, 본 개시의 일 실시 에에 따라 자율적으로 최적의 마이그레이션을 수행할 수 있는 시스템 설계를 제안한다.

[0094] 최적의 라이브 마이그레이션을 수행할 수 있는 제안된 자율 시스템은 도 6에 도시되어 있다. 본 시스템은 다음과 같은 모듈들로 구성된다. 본 모듈들은, 각 장치 내에서 하드웨어 및/또는 소프트웨어의 결합으로 구현될 수 있다.

[0095] - 컨트롤러(Controller): Logger에서 핸드오프 이벤트가 보고되면 컨트롤러는 최적의 마이그레이션 기술을 결정하고 ES(src) 및 ES(tgt)에 결정과 함께 선택한 마이그레이션을 수행하는 데 필요한 추가 정보를 알릴 수 있다.

[0096] - Logger: 모든 이벤트를 수집하고 로컬 파일 시스템에 저장한다. MU의 핸드오프 이벤트가 AP에 의해 보고되면 Logger는 이벤트를 컨트롤러에 푸시한다.

[0097] 컨트롤러 및 로커는 코어 네트워크에 포함된 적어도 하나의 서버(100)에 포함될 수 있다. 서버(100)는 하나 이상의 컴퓨터를 포함한다.

[0098] - ES: Edge 서버는 컨테이너화된 서비스 및 마이그레이션 절차를 수행한다.

[0099] - AP: 연결된 MU에 대한 네트워크 액세스를 제공하는 것 외에도 AP는 패킷 릴레이 및 리플레이 버퍼를 모두 제어할 수 있다.

[0100] - 모바일 사용자(MU. 사용자 단말): MU는 컨테이너에 지속적으로 작업을 오프로드하고, 이동성에 따라, 주변 AP 간에 발생하는 핸드오프의 대상이 된다.

[0101] 각 모듈 내부의 구성 요소는 상술한 예시적 시나리오(도 2)를 통해 이하 소개된다. MU가 네트워크에 가입하고 AP-1과 연결한다고 가정될 수 있다. MU는 또한 ES-1에서 컨테이너화된 서비스를 시작하고 AP-1은 Event Reporter에 의해 모든 이벤트 로그를 Logger로 보낼 수 있다.

[0102] Logger는 로그를 수신한 다음 Log Writer에 의해 로그를 로컬에 저장할 수 있다. Logger가 수신하는 로그 중 이벤트 푸시 구성 요소에 의해 특정 이벤트가 컨트롤러에 푸시될 수 있다. 예를 들어, MU와 AP-1의 연결, 및 컨테이너화된 서비스에 사용되는 기본 이미지의 이름은 컨트롤러에 푸시되는 특수 이벤트에 해당할 수 있다. 그러면, Controller는 AP ID(예: SSID, IP 주소, MAC 주소)와 이미지 ID(예: 이미지 이름, 태그)를 Association DB와 Image DB에 각각 저장할 수 있다.

[0103] 본 개시에서는, 사용자가 핸드오프 이벤트를 발생시키면 항상 마이그레이션이 실행된다고 가정한다. 이는, 특히 AP의 적용 범위가 크고 사용자 단말이 느린 속도로 이동할 때 합리적인 가정에 해당한다. IEEE 802.11의 경우 핸드오프 과정에서 MU는 새로운 AP(즉, 도 2의 AP-2)에게 REASSOCIATION REQUEST 메시지를 전송하며, 이는 AUTHENTICATION 메시지의 교환을 제외하고 핸드오프 이벤트가 감지되는 가장 빠른 순간이다. 재연결 이벤트 로그(re-association event log)도 Logger로 전송되고 Logger의 이벤트 푸시 구성 요소는 핸드오프 이벤트를 컨트롤러로 전달할 수 있다.

[0104] Controller가 핸드오프 이벤트를 통보받으면 Migration Decision 구성은 최적의 마이그레이션을 선택하는 데 필요한 프로필 등의 정보를 ES-1에 요청하는 것으로 시작한다. ES-1의 Profile Extractor는 프로필을 수집하여 다시 Controller로 보낸다. 프로필에 포함된 정보 집합은 후술한다. 또한 Controller는 AP-1 ID를 AP-2로, AP-2 ID를 AP-1로 보낼 수 있다. 프로필을 수신하면 Controller는 최적의 마이그레이션 기법을 결정하고 AP-1과 AP-2 모두에 결정을 통보할 수 있다. ES의 Migration Agent는 결정에 응답하고 컨테이너 마이그레이션을 시작할 수 있다. AP-1이 전송을 마치면, AP-1은 AP-2에 컨테이너 시작 요청을 보낼 수 있다. 그런 다음, AP-2의 마이그레이션 에이전트는 Docker 엔진이 컨테이너를 시작하도록 허용하고 리플레이 버퍼(Replay Buffer)에서 커맨드를 리플레이하기 전에 남아 있는 마이그레이션 절차(있는 경우)를 수행할 수 있다. 마이그레이션 절차가 완료되면, 버퍼가 비워질 때까지 리플레이 버퍼의 명령이 ES-2의 컨테이너에서 재생될 수 있다. 더 이상 리플레이 할 명령이 없으면 ES-2는 AP-2에 가용성을 알릴 수 있으며, AP-1까지 전달된 결과, AP-1이 ES-1의 컨테이너를 중지할 수 있다.

[0105] 제안된 최적의 마이그레이션 결정 알고리즘은 마이그레이션 시간과 마이그레이션 결과 네트워크에 주입될 트래픽 부하를 함께 고려할 수 있다. 마이그레이션 시간은 ES(src)에서 마이그레이션이 시작된 시점부터 ES(tgt)에서 마이그레이션된 컨테이너가 오프로딩 서비스를 제공할 준비가 된 시점까지의 시간 간격을 의미한다. 분명히, 마이그레이션 시간은 서비스 지연과 같은 QoS에 영향을 미친다. 마이그레이션이 진행 중일 때 AP-2에서 수신한 사용자 요청은 AP-1의 ES(src)로 릴레이되어 서비스 지연이 증가할 수 있다. 따라서, 미리 정해진 QoS 또는

SLA(서비스 수준 계약)를 충족하려면 마이그레이션을 최대한 빨리 완료할 필요가 있다. 또한 ES(src)의 컨테이너는 마이그레이션이 완료된 후에만 시스템 리소스를 중지하고 해제할 수 있으므로 마이그레이션 시간을 단축하면 AP-1의 컴퓨팅 리소스가 절약될 수 있다.

[0106] 선택할 마이그레이션은 각 마이그레이션 기술이 네트워크에 트래픽을 주입하기 때문에 네트워크 로드 또는 정체에도 영향을 준다. ES(src)에서 ES(tgt)까지의 라우팅 경로가 아무리 짧더라도 마이그레이션 결과 네트워크 트래픽은 코어 네트워크로 진입하게 된다(도 1). 따라서 선택한 마이그레이션 기법이 네트워크 트래픽을 너무 많이 생성하면 네트워크 혼잡이 발생할 수 있다. 전송은 전송 오류의 가능성을 증가시킬 수 있으며, 이러한 오류는 재전송을 유발하여 마이그레이션 시간을 증가시킬 수 있다.

[0107] 최적의 마이그레이션 결정 문제를 공식화하기 위해, 먼저 각 마이그레이션 기술의 예상 마이그레이션 시간이 표현될 수 있다. T_{FC}를 FC의 마이그레이션 시간이라고 했을 때, 이미지 생성 시간 t_{imggen}, 체크포인트 생성 시간 t_{chkgen}, 컨테이너 이미지 전송 시간 t_{imgtx}, 체크포인트 전송 시간 t_{chktx}, 컨테이너 시작 시간 t_{ctst}, 마이그레이션 중 버퍼링된 커맨드의 리플레이 시간 t_{bufrep}의 합이 T_{FC}로 정의될 수 있다. 또한, T_{DC}를 DC의 마이그레이션 시간이라고 했을 때, 체크포인트 생성 시간 t_{chkgen}, 쓰기 가능한 레이어 콘텐츠 전송 시간 t_{difftx}, 체크포인트 전송 시간 t_{chktx}, 컨테이너 시작 시간 t_{ctst}, 마이그레이션 중 버퍼링된 명령을 재생할 시간 t_{bufrep}의 합이 T_{DC}로 정의될 수 있다. 또한, T_{LR}을 LR의 마이그레이션 시간이라고 했을 때, command trace의 생성 시간 t_{trgen}, trace 전송 시간 t_{trtx}, 컨테이너 시작 시간 t_{ctst}, 수신된 trace의 리플레이 시간 t_{rep}, 마이그레이션 중 버퍼링된 명령을 리플레이할 시간 t_{bufrep}의 합이 T_{LR}로 정의될 수 있다.

[0108] 컨테이너에서 실행되는 사용자 명령의 양이 메모리 상태 변경의 양에 영향을 미친다는 점을 감안할 때 t_{chkgen} ~ t_{trgen}이라고 가정할 수 있다. 그리고, C는 T_{FC}, T_{DC} 및 T_{LR}에 포함된 공통 시간 요소로 정의될 수 있다. 구체적으로, T_{FC} 및 T_{DC}의 경우 $C = t_{chkgen} + t_{ctst} + t_{bufrep}$이고 T_{LR}의 경우 $C = t_{trgen} + t_{ctst} + t_{bufrep}$에 해당할 수 있다. 결과적으로 $T_{FC} = t_{imggen} + t_{imgtx} + t_{chktx} + C$, $T_{DC} = t_{difftx} + t_{chktx} + C$, $T_{LR} = t_{trtx} + t_{rep} + C$와 같이 마이그레이션 시간이 단축될 수 있다(언급된 용어는 초 단위일 수 있음). 전송 시간은 전송할 비트 수와 대역폭 B(초당 비트 수)의 함수이며 후자는 알려진 것으로 가정할 수 있다. 후술할 실험에서 평가될 C의 값은 여러 실험의 평균으로 구해진다.

[0109] 또한, 다음과 같이 세 가지 마이그레이션 기술에 의해 생성될 네트워크 트래픽의 양이 정의될 수 있다. L_{FC}를 FC 실행 결과 발생하는 네트워크 부하라고 하고, 이미지를 전송하기 위한 데이터 양 l_{imgtx} 과 체크포인트를 전송하기 위한 데이터 양 l_{chktx}의 합으로 정의될 수 있다. L_{DC}를 DC 실행의 결과로 발생하는 네트워크 부하라고 하고, 쓰기 가능 계층을 전송하기 위한 데이터 양 l_{difftx} 과 체크포인트를 전송하기 위한 데이터 양 l_{chktx}이라는 용어의 합으로 정의될 수 있다. L_{LR}을 LR 실행 결과 발생하는 네트워크 부하라고 하고, command trace를 전송하기 위한 데이터의 양 l_{trtx}에 해당할 수 있다. 여기에서 언급된 용어는 비트 단위이다.

[0110] 마이그레이션 시간 및 네트워크 부하에 대한 위의 방정식에서 DC가 항상 FC보다 우수하다는 것은 분명하다. 전체 컨테이너 이미지, 즉 읽기 전용 레이어와 쓰기 가능 레이어는 쓰기 가능 레이어를 포함하고, 경우에 따라 전체 이미지의 크기가 쓰기 가능 레이어보다 훨씬 크기 때문이다. 따라서, 마이그레이션 시간과 생성되는 네트워크 트래픽 측면에서 DC가 항상 FC를 능가할 수 있다. 이는 후술할 평가 결과와 밀접하게 일치한다. 예를 들어 FC의 마이그레이션 시간은 표 4 및 표 5와 같이 DC의 마이그레이션 시간보다 10~20배 더 클 수 있다. 따라서, 세 가지 기술 중 최적의 마이그레이션 결정은 DC와 LR 중 최상의 마이그레이션을 선택하는 것으로 좁혀질 수 있다. 해당하는 최적의 마이그레이션 결정 문제는 다음과 같다(P. 1로 표시).

수학식 1

[0111] $$\min_x f(x) = \frac{1}{\alpha}\{x \cdot T_{DC} + (1-x)T_{LR}\} + w\frac{1}{\beta}\{x \cdot L_{DC} + (1-x)L_{LR}\}$$

[0112] (1a)

[0113] subject to: $x \in \{0,1\}$ (1b)

[0114] 여기서 $w \in R_+$는 네트워크 부하에 대한 가중치를 나타내는 음이 아닌 디자인 파라미터이다. $t_{difftx} = l_{difftx}/B$, $t_{chktx} = l_{chktx}/B$, $t_{trtx} = l_{trtx}/B$이다. w -> 0일 수록 마이그레이션 시간을 최소화하는 마이그레이션 기법이 최적의 솔루션으로 선택되고, w가 커질수록 네트워크 트래픽을 최소화하는 것이 더 중요해진다. 대역폭 B는 packet pair probing(R. Prasad et al)과 같이 잘 알려진 기술을 통해 추정될 수 있다.

[0115] T_{DC} 및 T_{LR}과 같은 시간 관련 용어는 초 단위로 측정되는 반면 네트워크 부하 관련 용어, 즉 L_{DC} 및 L_{LR}은 비트 단위이다. 이것은 본질적으로 다른 것보다 훨씬 큰 가치를 갖는 문제를 초래할 수 있다. 다른 유형의 항이 동일한 척도를 갖도록 하기 위해 α와 β를 모두 도입하여 목적 함수(1a)에서 해당 항을 정규화할 수 있다. P.1을 푸는 데 필요한 값은 문제를 풀기 직전에 계산될 수 있다. 예를 들어, command trace의 크기는 앞서 언급한 대로 로그 파일을 구문 분석한 결과 제공될 수 있다. 쓰기 가능한 계층의 크기는 UpperDir에서 linux 명령 du -h -apparent-size를 사용하여 계산될 수 있다.

[0116] 체크포인트의 크기는 체크포인트를 생성함으로써 빠르게 계산될 수 있다. 컨테이너의 실시간 메모리 사용량을 보여주는 docker stats 명령을 통해 대략적인 크기가 식별될 수도 있다.

[0117] 주어진 P.1은 이진 제약 조건(1b)을 갖는 정수(이진) 최적화 문제이다. 정수 문제는 일반적으로 다루기 힘들지만 P.1의 작은 크기로 인해, 예를 들어 branch and bound algorithm(R. E. davis et al)을 사용하여 최적의 솔루션을 빠르게 찾을 수 있다. 그러나, 제안된 알고리즘은 기존의 솔루션 방법을 사용하여 P.1에 대한 최적값을 계산하는 대신 x의 각 가능한 값으로 개별적으로 문제를 해결하고 가장 작은 목적 값이 나오는 최상의 것을 선택할 수 있다. 결정변수 x는 0과 1 사이(0 또는 1)의 값만 가지며 f(x)에 포함된 항은 선형이므로 제안되는 무차별 대입 알고리즘은 빠르게 종료될 수 있다.

[0118] 상술한 알고리즘의 성능을 평가하기 위해 자동 마이그레이션 시스템(도 6)과 세 가지 마이그레이션 기법 FC(도 3), DC(도 4) 및 LR(도 5)이 구현되었다. 각 사용자는 오프로딩 서비스를 위해 전용 컨테이너에 독점적으로 액세스할 수 있다. 사용자의 오프로딩 요청 다음에 ES에서 실행되는 전용 컨테이너의 애플리케이션 계층 ACK(긍정적 승인)가 이어지는 바, 응답 시간 또는 서비스 지연은 사용자가 요청을 보낸 순간부터 사용자가 해당 ACK를 수신한 순간 사이의 경과 시간으로 정의될 수 있다. 핸드오버 이벤트를 정확하게 스케줄링하기 위해 단순화된 AP를 구현하고 사용자의 움직임을 에뮬레이션하여 지정된 시간에 핸드오버가 발생하도록 설계되었다.

[0119] 테스트베드는 3대의 PC와 노트북으로 구성되었다. PC #1은 컨트롤러와 Logger를 모두 실행하고, PC #2는 AP-1과 ES-1을 모두 실행하고, PC #3은 AP-2와 ES-2를 모두 실행하고, 노트북은 MU로 동작하도록 구현되었다. 예시된 시나리오(도 2)에서 설명한 것처럼 MU는 AP-1과 연결되고 ES(src)인 ES-1으로부터 서비스를 받을 수 있다. MU가 AP-2로 이동함에 따라 핸드오프가 발생하고 컨테이너화된 서비스는 ES(tgt)인 ES-2에서 실행되도록 AP-2로 마이그레이션된다. 4대의 기계는 이더넷 케이블과 IEEE 802.11ac 채널을 통해 PC와 노트북이 각각 연결되는 단일 근거리 통신망 내에 위치하였다. 마이그레이션 중 서비스 지연을 정확하게 측정하기 위해 MU는 일정 간격으로 서비스 요청을 생성하도록 구성되었으며, 기본적으로 MU의 요청 속도는 초당 1개 요청으로 설계되었다. AP의 대역폭과 대기 시간을 모두 제어하기 위해 Linux Traffic Control을 구성하는 Linux 도구인 tc가 사용되었다. 두 AP 간의 일련의 ICMP 메시지 교환에 의해 측정된 RTT(왕복 시간)는 평균 200ms인 반면, MU 및 MU와 관련된 AP, AP 및 해당 AP의 ES, AP 및 컨트롤러, AP 및 Logger 등 다양한 두 개체 사이의 평균 RTT는 약 20ms이다.

[0120] 다양한 설정 또는 시나리오에서 제안된 방법을 평가하기 위해, 두 가지 서비스인 서비스 1(SVC-1)과 서비스 2(SVC-2), 두 서비스 모두에 적용 가능하며 사용자 액션(또는 명령)의 두 가지 다른 시퀀스인 ACT-1 및 ACT-2가 활용되었다. SVC-1은 빠르게 시작할 수 있고 쓰기 가능한 계층에 저장할 소량의 데이터를 생성할 수 있는 간단하고 가벼운 애플리케이션 컨테이너이다. 반면에 SVC-2는 상대적으로 무거운 애플리케이션 컨테이너이며 시작하는 데 시간이 더 오래 걸리고 사용자 요청으로 인해 저장할 많은 데이터가 생성될 수 있다. ACT-1은 완료하는 데 오래 걸리지 않는 작업으로 구성된다. 그러나, ACT-2는 ACT-1보다 더 오래 걸리는 일련의 작업이다. 또한, 작은 대역폭과 큰 대역폭의 두 가지 구성으로 네트워크 대역폭이 제어되었다. 이렇듯 두 가지 응용 프로그램, 두 가지 작업 시퀀스, 및 두 가지 네트워크 대역폭 설정을 사용하여 표 2에 요약된 대로 8개의 서로 다른 프로필 또는 시나리오가 구현되었다.

[0121] 각 애플리케이션에 대한 각 작업 시퀀스의 영향을 분석하여 DC에 대해 전송할 데이터의 양과 LR의 일관된 상태

를 복원하는 데 걸리는 시간을 강조하기 위해 다음과 같은 두 가지 구성 목록으로 8개의 프로필이 재구성되었다. 하나는 작업 시퀀스를 포함하는 컨테이너화된 서비스 애플리케이션을 위한 것이고 다른 하나는 네트워크 대역폭 구성을 위한 것이다. AC-st를 s 및 t로 레이블이 지정된 애플리케이션 구성이라고 하고 NC-b를 b로 레이블이 지정된 네트워크 구성이라고 각각 가정할 수 있다.

표 2

[0122]

Scenario Label	Service Type	Action Sequence	Network
AC-LF/NC-L	SVC-1	ACT-1	Low bandwidth
AC-LS/NC-L	SVC-1	ACT-2	Low bandwidth
AC-HF/NC-L	SVC-2	ACT-1	Low bandwidth
AC-HS/NC-L	SVC-2	ACT-2	Low bandwidth
AC-LF/NC-H	SVC-1	ACT-1	High bandwidth
AC-LS/NC-H	SVC-1	ACT-2	High bandwidth
AC-HF/NC-H	SVC-2	ACT-1	High bandwidth
AC-HS/NC-H	SVC-2	ACT-2	High bandwidth

표 3

[0123]

Application Configuration			Network Config.	
ID	Memory use + WL size (MB)	Trace replay time (second)	ID	Bandwidth (MB/s)
AC-LF	50	5	NC-L	10
AC-HF	200	5	NC-H	50
AC-LS	50	10		
AC-HS	200	10		

[0124] 여기서, 이하와 같이 정의된다.

[0125] - s∈{L, H} (Low, High): 쓰기 가능한 계층에 저장될 수 있는 데이터의 양 및 체크포인트 사이즈(for DC)

[0126] - t∈{F, S} (fast, slow): 새로 실행된 컨테이너가 일관된 상태를 갖도록 LR이 수행해야 하는 작업에 해당하는, command trace에 로그된 명령을 실행하는 데 걸리는 시간 (참고: 이는 패킷 릴레이 동안 버퍼링된 명령을 리플레이 하는 것과 다름).

[0127] - b∈{L, H} (Low, High): 네트워크 대역폭참고적으로, FC는 전체 컨테이너 이미지(즉, 초기 이미지와 쓰기 가능한 계층)와 체크포인트를 전송한다. 마찬가지로 FC는 초기 이미지를 전송하고 DC가 전송하는 것으로 간주할 수 있다.

[0128] 구성 레이블 s 및 t는 각각 DC 및 LR과 관련된 반면 레이블 b는 세 가지 마이그레이션 기술 모두의 성능에 영향을 준다. FC의 경우 전송할 데이터의 양은 전체 컨테이너 이미지와 체크포인트의 크기이며 전자는 위에서 언급하지 않았다. 앞서 언급했듯이 FC는 항상 DC보다 성능이 뛰어나므로 DC와 LR의 성능 비교에 주로 중점을 둔다. 구성 세부 정보는 표 3에 나열되어 있다. 기본 이미지는 크기가 약 909MB인 공식 python:3.8이다. 이미지는 DockerHub에서 가져올 수 있다. 각 시나리오에 대해 평가는 200초 또는 100초 동안 실행되고 MU는 기본 속도로 서비스 요청을 생성한다. MU는 처음으로부터 50초가 경과한 시점 또는 연결된 AP에 50번째 요청을 보내는 시점에 AP-1에서 AP-2로 핸드오프한다. 개시된 각 데이터는 5회 실험의 평균이다.

[0129] 언급한 바와 같이, P.1에 대한 최적해는 가능한 x로 목적함수 f(x)를 평가함으로써 찾을 수 있다. 최적의 x는 더 작은 f(x)를 가져온 것이다. 평가를 위해, 고려하는 모든 시나리오 프로필에서 LR에 대해서는 다른 두 가지보다 네트워크에 더 적은 트래픽이 주입되었다. 따라서, DC는 마이그레이션 시간이 LR보다 짧은 경우에만 최적의 마이그레이션을 위해 선택할 수 있다. 그럼에도 불구하고, w가 커질수록 마이그레이션 결과 발생하는 네트워크 트래픽의 양에 더 많은 가중치가 부여되므로, LR이 최적으로 선택될 가능성이 높다. 주어진 시나리오 프로필에 대해 최적 솔루션이 w로 변경되면 f(0)과 f(1)에 의해 형성된 두 선이 교차할 때 해가 한 번 도출된다. 구체

적으로, f(0) = f(1)을 넣으면 f(0)과 f(1)이 교차하는 w의 정확한 값이 다음과 같이 생성될 수 있다.

수학식 2

[0130] $$w=\frac{\beta}{\alpha}\times\frac{T_{LR}-T_{DC}}{L_{DC}-L_{LR}}$$

[0131] 먼저, AC-LF/NC-L 및 AC-LF/NC-의 두 프로필에 대한 각 마이그레이션 기술의 작동 방식과 전반적인 서비스 지연(각 사용 요청의 응답 시간) 변화가 모니터링되었다. AC-LF/NC-L과 AC-LF/NC-H는 모두 동일한 애플리케이션 구성을 가지고 있다. 그러나, 네트워크 대역폭이 다르기 때문에, 마이그레이션 관련 데이터를 전송하는 데 걸리는 시간에 영향이 있어 전체 마이그레이션 시간이 달라진다. 여기서, 마이그레이션 시간(migration time)은 MU의 핸드오프(이전의 AP로부터)에 의해 마이그레이션이 트리거되는 순간부터 일관된 상태의 마이그레이션된 컨테이너가 서비스를 제공할 준비가 된 순간(새로운 AP 상에서) 사이의 경과 시간으로 정의된다. 도 7과 도 8은 AC-LF/NC-L 시나리오 프로파일에 대한 실험 결과를 보여준다. 구체적으로, 도 7은 FC(7(a)), DC(7(b)) 및 LR(7(c))이 마이그레이션에 사용될 때 요청당 응답 시간의 자취를 보여준다. 도 8은 3개의 마이그레이션에 대하여 평균 요청당 응답 시간만 보여준다.

[0132] 예상대로 FC는 마이그레이션에 가장 많은 시간을 할애했으며, 그 결과 세 가지 마이그레이션 기술 중 응답 시간이 가장 컸다. 마이그레이션하는 동안 AP-1과 AP-2 간에 패킷 릴레이가 발생하고 둘 사이의 RTT가 크기 때문에 응답 시간은 마이그레이션 시간에 비례한다. 마이그레이션이 진행되는 동안 AP-2와 연결된 MU는 AP-2와 통신할 수 있다. 그러나, 작동하는 컨테이너는 여전히 AP-1의 ES-1에 있으며 AP-2와 같은 위치에 있는 컨테이너는 아직 시작되지 않을 수 있다. 따라서 AP-2는 ES-1에 있는 MU의 컨테이너가 요청에 응답할 수 있도록 MU의 요청을 AP-1에 전달해야 한다. 이 중계로 인해 응답 시간이 길어질 수 있다.

[0133] FC가 있는 ES(src)는 마이그레이션을 위해 다음 동작을 순서대로 수행할 수 있다. 1) 읽기 전용 계층과 쓰기 가능 계층을 모두 포함하는 완전한 컨테이너 이미지를 생성하고, 2) 이미지를 전송하며, 3) 체크포인트를 생성하고, 4) 체크포인트를 전송한다. 동작 1)은 이미지 크기가 크기 때문에 실험에서 몇 초가 걸린다. 또한, ACLF/NC-L 시나리오 프로파일의 낮은 대역폭의 경우 이미지를 전송하는 데 약 100초가 소요되었다. 이는 FC가 생성하는 전체 트래픽이 약 960MB이기 때문이다(즉, 읽기 전용 이미지의 경우 909MB, 체크포인트 및 쓰기 가능한 레이어 콘텐츠 모두의 경우 50MB). 전체 컨테이너 이미지와 체크포인트를 수신하면, ES-2는 1) 수신된 이미지 로드, 2) 이미지에서 컨테이너 생성, 3) 수신된 체크포인트로 컨테이너 시작 등의 동작들을 수행할 수 있다. 본 세 가지 동작에도 몇 초의 시간이 필요하다.

[0134] 반면 DC와 LR은 모두 FC보다 마이그레이션 시간이 훨씬 짧다. 주된 이유는 전송할 데이터의 양이 감소하기 때문이다. DC가 쓰기 가능한 계층과 체크포인트의 합인 50MB의 데이터를 전송하는 데에 약 5초가 소요된다. LR은 약 1MB의 작은 크기의 텍스트, 즉 커맨드 로그(command log)만 전송하고 수신된 command trace를 재생하는 데 5초를 소비한다. LR 마이그레이션은 DC보다 약간 짧았지만 차이는 무시할 수 있을 정도이다. 이는 주로 체크포인트로부터 생성하고 복원하는 데 걸리는 시간 때문이다.

표 4

[0135]

Average (second)	FC	DC	LR
Migration time	124.9040	6.9080	5.9728
Per-request response time	0.2029	0.0289	0.0266

[0136] 마이그레이션 시간과 평균 응답 시간을 표 4에 요약한 것처럼, 마이그레이션 시간은 서비스 지연 또는 사용자 요청에 대한 응답 시간에 직접적인 영향을 미친다. 그 결과 FC가 평균 0.2029초라는 가장 큰 응답 시간을 기록했다. DC의 마이그레이션 시간은 LR의 마이그레이션 시간보다 약간 길지만 둘 사이의 평균 응답 시간은 크게 다르지 않았다. 실험이 진행됨에 따라 LR에 비해 DC에 의해 발생하는 추가 지연이 미미하게 되었기 때문이다. 실험이 더 오래 지속되면 DC와 LR 사이의 평균 지연이 훨씬 더 가까워졌다.

[0137] 최적의 마이그레이션 관점에서 LR은 항상 이 시나리오 프로필에 대한 최적의 솔루션이 된다. 첫째, LR은 항상 네트워크에 더 적은 트래픽을 주입한다. 또한 ACLF/NC-L 구성의 경우 LR을 사용하면 마이그레이션 시간이 단축되었다. 따라서 LR은 가중치 매개변수 w의 모든 값에 대해 최적으로 선택될 수 있다.

표 5

[0138]

Average (second)	FC	DC	LR
Migration time	37.1064	3.4032	6.1295
Per-request response time	0.1390	0.0293	0.0351

[0139] 도 9, 도 10 및 표 5는 AC-LF/NC-H 시나리오 프로필에 대한 평가 결과를 보여준다. 평가는 200초가 아닌 100초로 진행되며, 이는 평균 응답시간에 영향을 미친다. 많은 데이터를 전송하는 마이그레이션 방법이 증가된 대역폭을 활용한다는 점은 주목할 가치가 있다. 그 결과, FC와 DC의 마이그레이션 시간이 크게 감소한 반면 LR의 마이그레이션 시간은 이전 AC-LF/NC-L 시나리오 프로파일에 비해 크게 변하지 않았다. 그 결과, FC의 마이그레이션 시간이 37.1064초로 단축되어 요청당 응답 시간이 단축되었다. 증가된 대역폭은 또한 DC의 마이그레이션을 줄였다. 그러나, LR은 ES-1에서 ES-2로 많이 전송하지 않기 때문에 LR의 마이그레이션 시간은 크게 변하지 않았다. 전송할 데이터는 작은 크기의 텍스트 전용 파일인 command trace 뿐이다. DC의 마이그레이션 시간 단축으로 평균적으로 LR보다 응답 시간이 단축되었다.

[0140] 최적의 마이그레이션 관점에서 도 11은 흥미로운 결과를 나타낸다. 비록 LR이 네트워크에 더 적은 트래픽을 주입하지만 DC는 AC-LF/NC-H 구성에 대해 더 짧은 마이그레이션 시간을 초래했다. 따라서, 이전 AC-LF/NC-L 시나리오 프로필과 달리 LR은 AC-LF/NC-H에서 항상 최적의 마이그레이션이 아니다. w의 작은 값은 생성할 네트워크 트래픽 양의 영향을 무시하는 경향이 있으므로 DC가 최적의 솔루션으로 선택될 수 있다. 그러나 w > 11:1277일 때 마이그레이션 시간에 대한 중요성이 사라지므로 LR이 최적으로 선택될 수 있다.

[0141] AC-LF/NC-L과 ACLF/NC-H 모두에 대한 평가 결과, 대역폭의 변화가 상대적으로 전송할 트래픽이 큰 FC와 DC 모두의 성능에 영향을 미치는 것으로 나타났다. 반면 전송할 것이 많지 않은 LR의 성능은 대역폭 변화에 따라 크게 달라지지 않는다.

[0142] 앞서 논의된 바와 같이 FC는 항상 DC보다 성능이 뛰어나다. 따라서, FC를 제외한 DC와 LR의 성능을 비교하는 실험이 수행되었다. 100초 동안 평가가 수행되었다.

[0143] 도 12와 표 6은 AC-LS/NC-L과 AC-LS/NC-H 시나리오 프로필에 대한 DC와 LR의 평가 결과를 보여준다. 두 시나리오 모두에서 볼 수 있듯이 네트워크 대역폭 차이에 대한 두 시나리오 프로파일 간에 DC의 마이그레이션 시간이 많이 변경되었지만 LR의 경우는 그렇지 않았다. ACT-2 동작 시퀀스를 사용하면 ES-2가 로그의 명령을 재생하는 데만 약 10초가 소요되었다. 반면에 DC는 총 50MB에 달하는 쓰기 가능 계층과 체크포인트만 전송하면 되므로 마이그레이션 시간이 LR보다 짧았다.

표 6

[0144]

Profile	Average (second)	DC	LR
AC-LS/ NC-L	Migration time	7.4572	10.9406
	Per-request response time	0.0411	0.0488
AC-LS/ NC-H	Migration time	3.7277	10.8831
	Per-request response time	0.0324	0.0487

[0145] 도 13은 AC-LS/NC-L과 AC-LS/NC-H 시나리오 프로필에 대해 평가된 P.1의 목적 값의 변화를 보여준다. 마이그레이션 시간의 관점에서 볼 때 LR은 DC보다 성능이 우수하므로, w가 증가함에 따라 최적의 마이그레이션이 변경된다. AC-LS/NC-L 및 AC-LS/NC-H 시나리오 프로필의 경우 최적 솔루션은 w = 14:2179 및 w = 29:2057일 때 각각 DC에서 LR로 변경된다. ACLS/NC-H 시나리오 프로필에서 AC-LS/NC-L 시나리오 프로필에 비해 최적 솔루션을 DC에서 LR로 전환하려면 더 큰 w 값이 필요하다. 이는 AC-LS/NC-H에서 DC의 마이그레이션 시간이 AC-LS/NC-L의 마이

그레이션 시간보다 짧고 LR이 두 시나리오 프로필에서 거의 동일한 마이그레이션 시간을 초래하기 때문이다.

[0146] SVC-2 애플리케이션 컨테이너를 처음부터 시작하는 절차는 SVC-1보다 복잡하며 LR의 마이그레이션 시간에 영향을 미친다. 마이그레이션하는 동안 DC는 중지되거나 일시 중지된 컨테이너를 다시 시작하며 컨테이너를 처음부터 시작하지 않는다. 반면에 LR은 ES-2에서 컨테이너를 처음부터 시작한 다음 수신된 추적 로그(trace log)를 리플레이 한다. 시간이 많이 걸리는 SVC-2의 시작은 전체 LR 마이그레이션 프로세스를 지연시키며, 이는 실험 결과에서 눈에 띄게 나타난다.

표 7

[0147]

Profile	Average (second)	DC	LR
AC-HF/ NC-L	Migration time	27.7610	8.9276
	Per-request response time	0.0943	0.0341
AC-HF/ NC-H	Migration time	11.4055	10.2165
	Per-request response time	0.0457	0.0357
AC-HS/ NC-L	Migration time	27.5847	15.7138
	Per-request response time	0.0913	0.0472
AC-HS/ NC-H	Migration time	13.8344	16.5808
	Per-request response time	0.0505	0.0509

[0148] SVC-2를 실행하는 컨테이너에 대한 평가는 다양한 작업 순서 및 네트워크 대역폭 구성에서 수행되었으며 결과는 도 14 및 표 7에 나와 있다. SVC-2 컨테이너는 SVC-1 컨테이너보다 더 큰 체크포인트와 쓰기 가능 계층을 생성하며, DC에서 생성에 활용되는 네트워크 트래픽 양이 200MB로 증가한다. 이는 네트워크 대역폭이 작을 때 DC가 LR보다 마이그레이션에 훨씬 더 오래 걸리는 이유이다(14(a) 및 14(c) 참조). 반면, 네트워크 대역폭이 클 경우 데이터 전송 시간이 줄어들어 DC의 마이그레이션 시간은 LR의 마이그레이션 시간에 가깝거나 짧다.

[0149] DC의 마이그레이션 시간은 주로 네트워크 대역폭에 따라 달라지므로 14(b) 및 14(d)와 같이 DC는 대역폭이 클수록 마이그레이션 시간이 짧아진다. 반면, LR의 마이그레이션 시간은 전송된 로그를 재생하는 데 걸리는 시간에 크게 영향을 받는다. 따라서 LR은 재생 시간이 더 짧은 경우(즉, 14(a) 및 14(b))에서만 더 짧은 마이그레이션 시간을 초래한다.

[0150] 4가지 시나리오 프로필 중 DC는 마이그레이션 시간과 관련하여 AC-HS/NC-H 시나리오 프로필에서만 LR을 능가한다. 따라서, 다른 세 가지 시나리오 프로필에 대해 LR은 도 15와 같이 최적으로 선택될 수 있다. AC-HS/NC-H 시나리오 프로필에서 DC의 마이그레이션 시간은 LR의 마이그레이션 시간보다 약간 짧다. 따라서 작은 값의 w에 대해서는 DC가 최적으로 선택될 수 있다. 그러나 $w > 2{:}7602$의 값에 대해서는 LR이 최적의 마이그레이션 기술로 해석될 수 있다.

[0151] 본 개시를 통해 컨테이너화된 서비스를 위한 세 가지 원활한 상태 저장 마이그레이션 기술이 개시되었다. FC와 DC는 모두 대상 에지 서버로 상태를 전송한다는 점에서 상태 복제 방식이다. 반면 LR은 command trace를 리플레이하여 일관된 상태의 컨테이너를 만드는 상태 재생 방식이다. 대상 에지 서버로 전송된 내용에 포함되지 않은 상태 변경인 마지막 순간의 상태 변경을 캡처하기 위해 패킷 릴레이 및 버퍼 릴레이 방법이 제안되었다.

[0152] 그런 다음 마이그레이션할 애플리케이션의 특성과 마이그레이션 방법을 함께 고려하여 최적의 마이그레이션 기법을 선택하는 자율 마이그레이션 시스템을 위한 시스템 설계가 개시되었다. 제안하는 최적의 마이그레이션 선택 문제는 마이그레이션 시간과 네트워크 부하를 모두 고려하여 튜닝 가능한 가중치 파라미터에 따라 최적의 결정을 내린다. 세 가지 마이그레이션 기술과 자율 마이그레이션 시스템이 구현되었고 실험도 진행되었다. 그 결과, 대역폭의 변화가 DC의 성능에 영향을 미치는 것으로 나타났다. 반면에 LR의 성능은 추적 재생 시간(trace replay time)의 응용 속성에 따라 다르다. 그 결과 최적의 마이그레이션을 결정할 때 마이그레이션 기법의 특성과 마이그레이션에 적용되는 속성을 함께 고려해야 한다는 점이 시사된다.

[0153] 한편, 이상에서 설명된 다양한 실시 예들은 서로 저촉되지 않는 한 둘 이상의 실시 예가 함께 구현될 수 있다.

[0154] 한편, 이상에서 설명된 다양한 실시 예들은 소프트웨어(software), 하드웨어(hardware) 또는 이들의 조합된 것

을 이용하여 컴퓨터 또는 이와 유사한 장치로 읽을 수 있는 기록 매체 내에서 구현될 수 있다.

[0155] 하드웨어적인 구현에 의하면, 본 개시에서 설명되는 실시 예들은 ASICs(Application Specific Integrated Circuits), DSPs(digital signal processors), DSPDs(digital signal processing devices), PLDs(Programmable logic devices), FPGAs(field programmable gate arrays), 프로세서(processor), 제어기(controller), 마이크로 컨트롤러(micro-controllers), 마이크로 프로세서(microprocessor), 기타 기능 수행을 위한 전기적인 유닛(unit) 중 적어도 하나를 이용하여 구현될 수 있다.

[0156] 일부의 경우에 본 명세서에서 설명되는 실시 예들이 프로세서 자체로 구현될 수 있다. 소프트웨어적인 구현에 의하면 본 명세서에서 설명되는 절차 및 기능과 같은 실시 예들은 별도의 소프트웨어 모듈들로 구현될 수 있다. 상술한 소프트웨어 모듈들 각각은 본 명세서에서 설명되는 하나 이상의 기능 및 작동을 수행할 수 있다.

[0157] 한편, 상술한 본 개시의 다양한 실시 예들에 따른 서버 또는 단말의 처리동작을 수행하기 위한 컴퓨터 명령어(computer instructions)는 비일시적 컴퓨터 판독 가능 매체(non-transitory computer-readable medium)에 저장될 수 있다. 이러한 비일시적 컴퓨터 판독 가능 매체에 저장된 컴퓨터 명령어는 특정 기기의 프로세서에 의해 실행되었을 때 상술한 다양한 실시 예에 따른 서버, ES, 사용자 단말 등의 처리 동작을 상술한 기기가 수행하도록 한다.

[0158] 비일시적 판독 가능 매체란 레지스터, 캐쉬, 메모리 등과 같이 짧은 순간 동안 데이터를 저장하는 매체가 아니라 반영구적으로 데이터를 저장하며, 기기에 의해 판독(reading)이 가능한 매체를 의미한다. 구체적으로는, 상술한 다양한 어플리케이션 또는 프로그램들은 CD, DVD, 하드 디스크, 블루레이 디스크, USB, 메모리카드, ROM 등과 같은 비일시적 판독 가능 매체에 저장되어 제공될 수 있다.

[0159] 또한, 이상에서는 본 발명의 바람직한 실시 예에 대하여 도시하고 설명하였지만, 본 발명은 상술한 특정의 실시 예에 한정되지 아니하며, 청구범위에서 청구하는 본 발명의 요지를 벗어남이 없이 당해 발명이 속하는 기술분야에서 통상의 지식을 가진 자에 의해 다양한 변형실시가 가능한 것은 물론이고, 이러한 변형실시들은 본 발명의 기술적 사상이나 전망으로부터 개별적으로 이해돼서는 안 될 것이다.

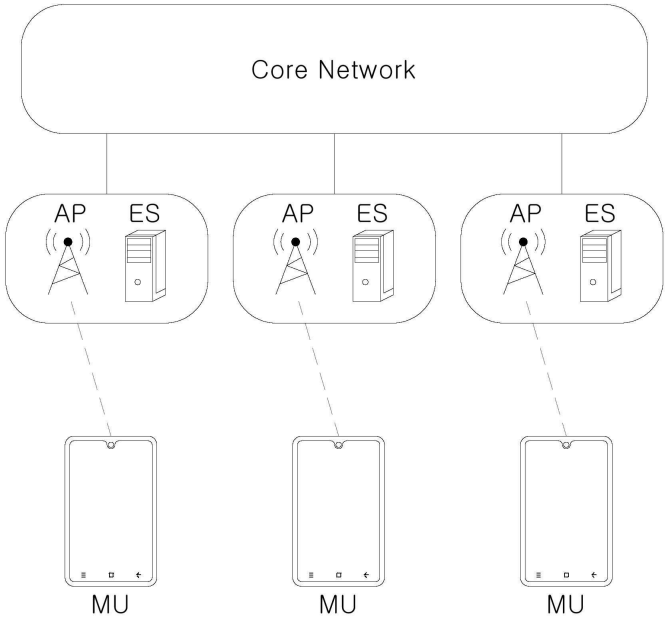
부호의 설명

[0160] 100: Server(서버) 200: ES(Edge Server)

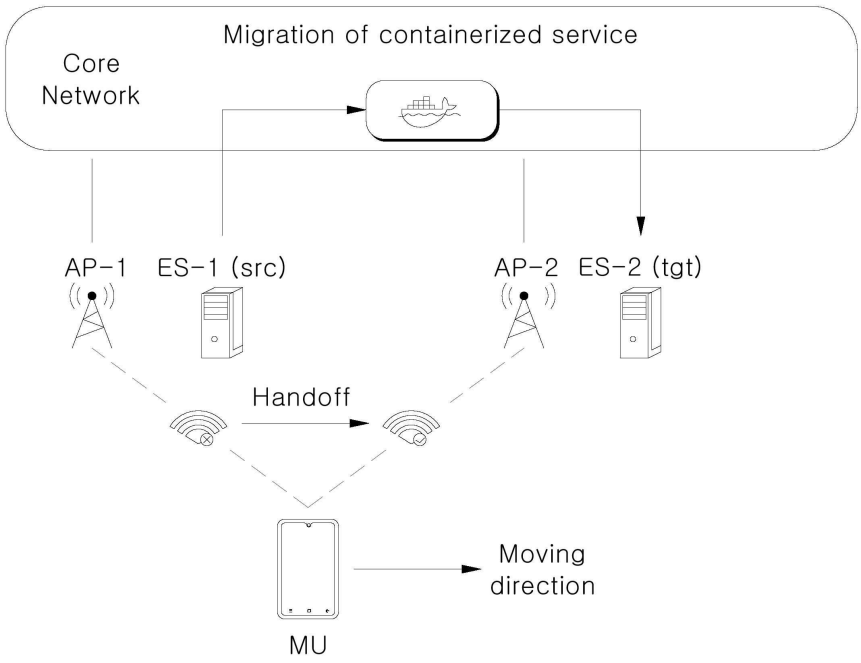
300: AP(Access Point) 400: MU(Mobile User)

도면

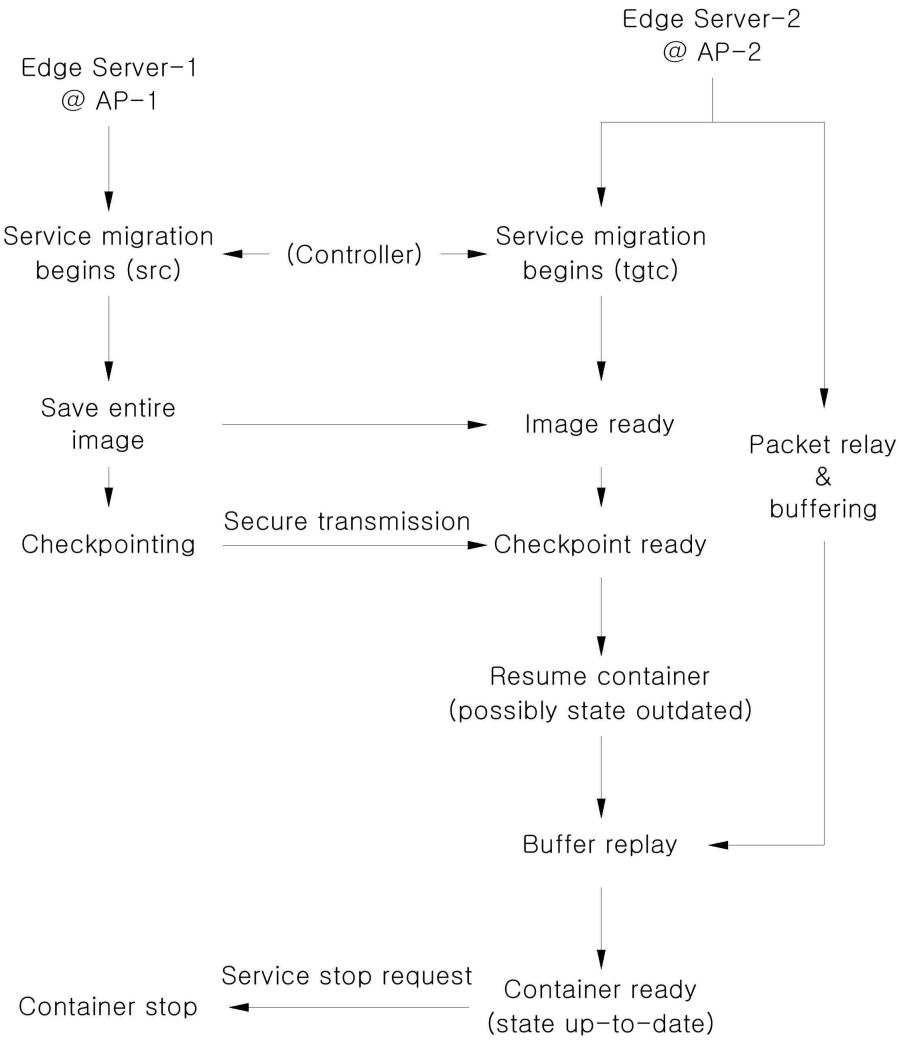
도면1



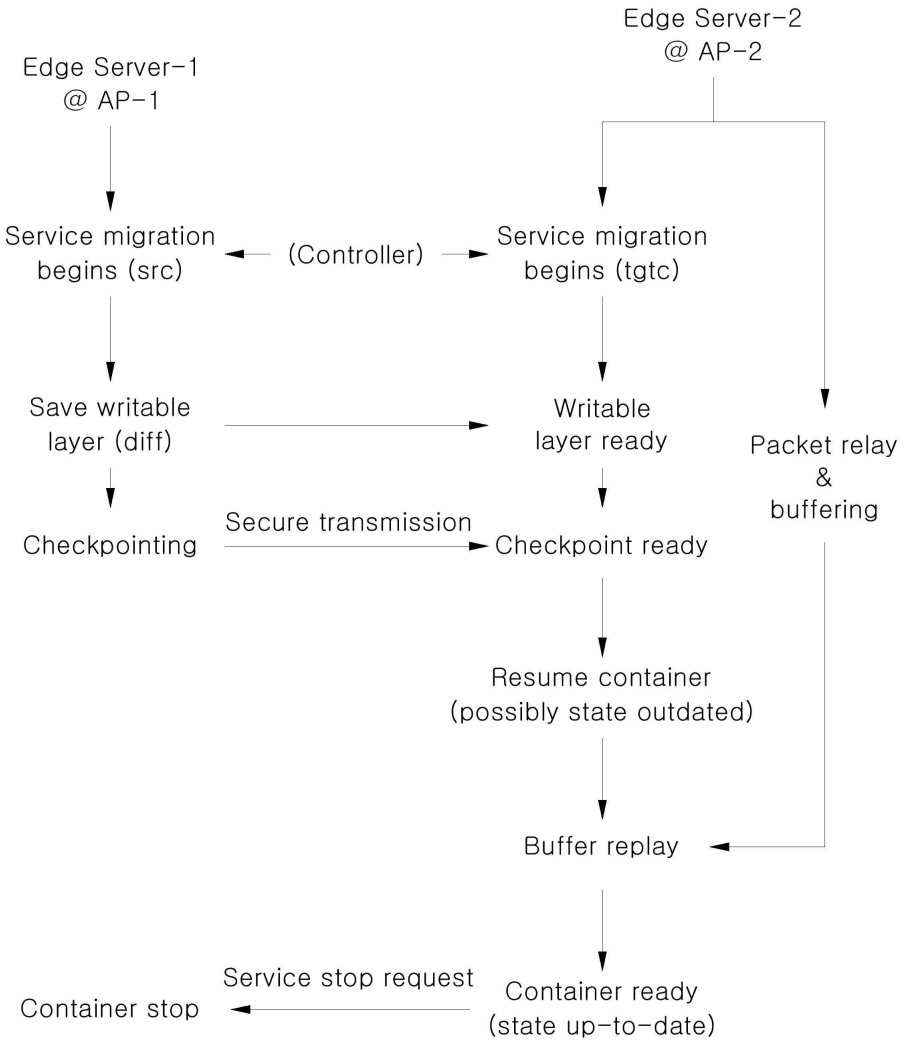
도면2


Migration of containerized service
Core
Network
AP-1
ES-1 (src)
AP-2
ES-2 (tgt)
Handoff
Moving
direction
MU

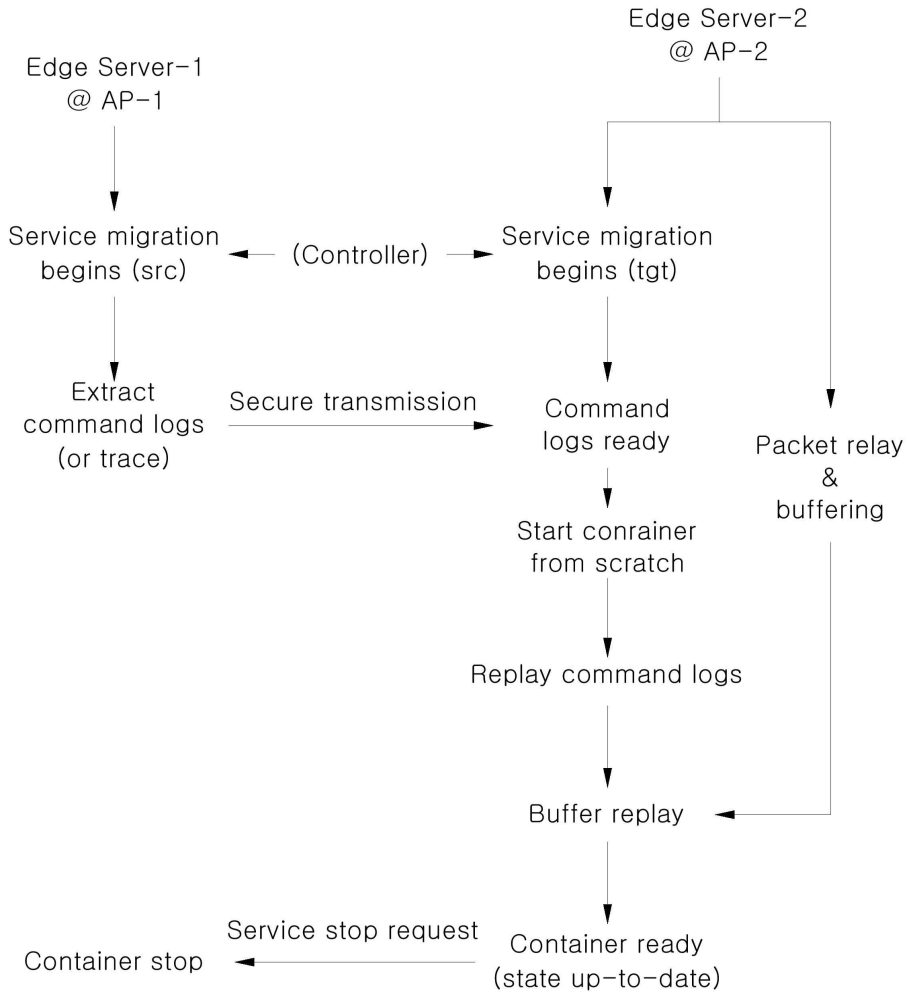
도면3


Edge Server-2
@ AP-2
Edge Server-1
@ AP-1
Service migration
begins (src)
(Controller)
Service migration
begins (tgtc)
Save entire
image
Image ready
Packet relay
&
buffering
Checkpointing
Secure transmission
Checkpoint ready
Resume container
(possibly state outdated)
Buffer replay
Container stop
Service stop request
Container ready
(state up-to-date)

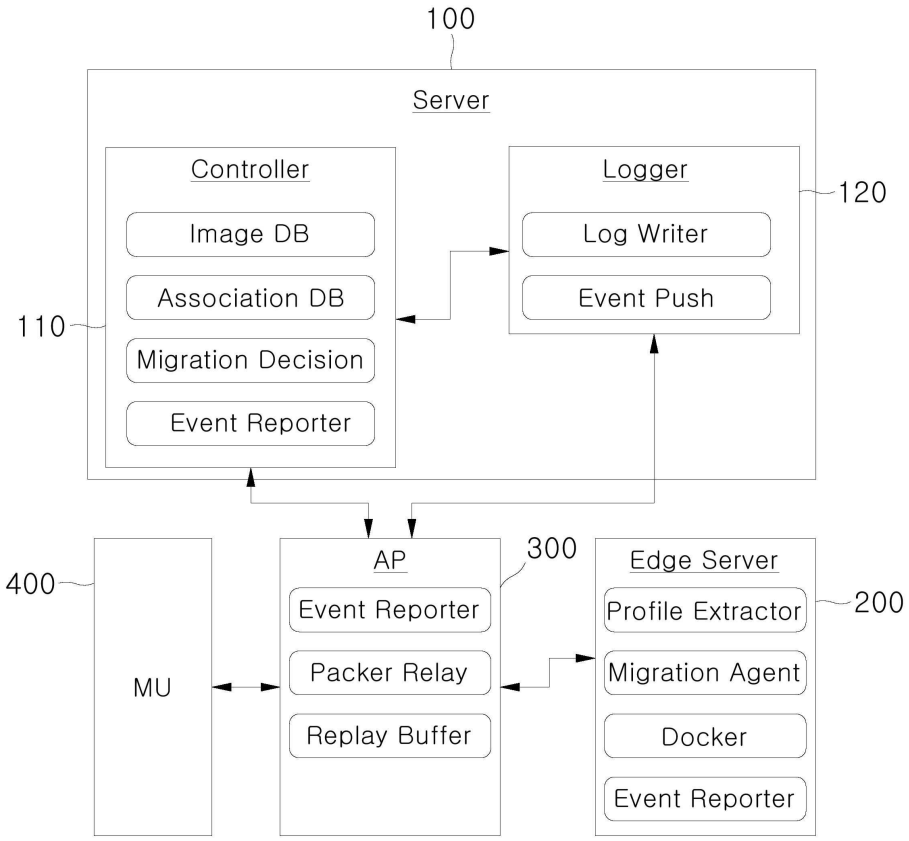
도면4


Edge Server-2
@ AP-2
Edge Server-1
@ AP-1
Service migration
begins (src)
(Controller)
Service migration
begins (tgtc)
Save writable
layer (diff)
Writable
layer ready
Packet relay
&
buffering
Checkpointing
Secure transmission
Checkpoint ready
Resume container
(possibly state outdated)
Buffer replay
Container stop
Service stop request
Container ready
(state up-to-date)

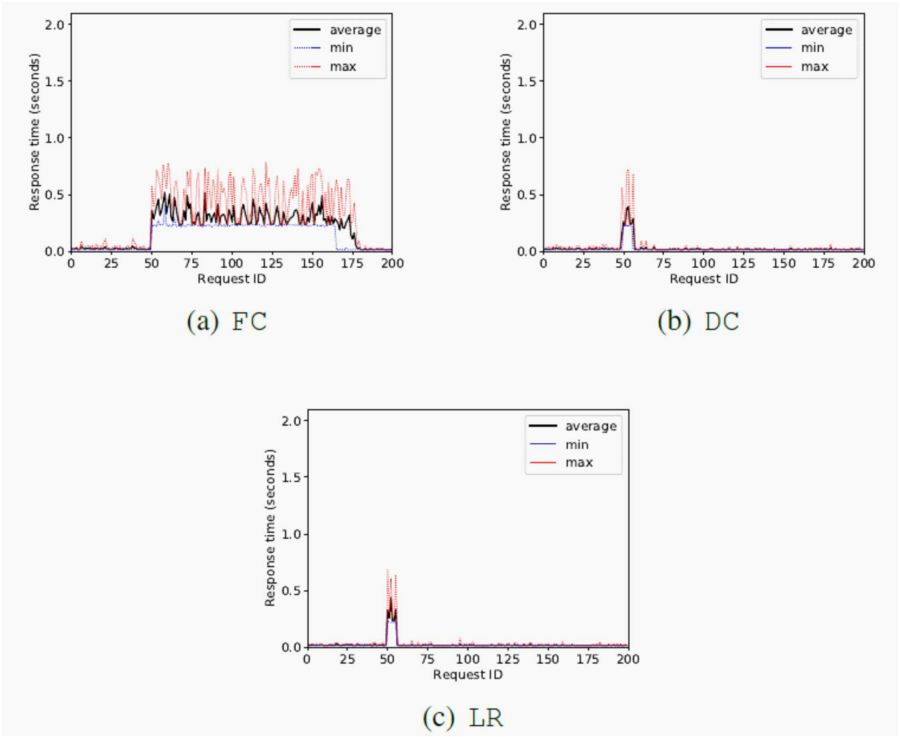
도면5


Edge Server-2
@ AP-2
Edge Server-1
@ AP-1
Service migration
begins (src)
(Controller)
Service migration
begins (tgt)
Extract
command logs
(or trace)
Secure transmission
Command
logs ready
Packet relay
&
buffering
Start conrainer
from scratch
Replay command logs
Buffer replay
Container stop
Service stop request
Container ready
(state up-to-date)

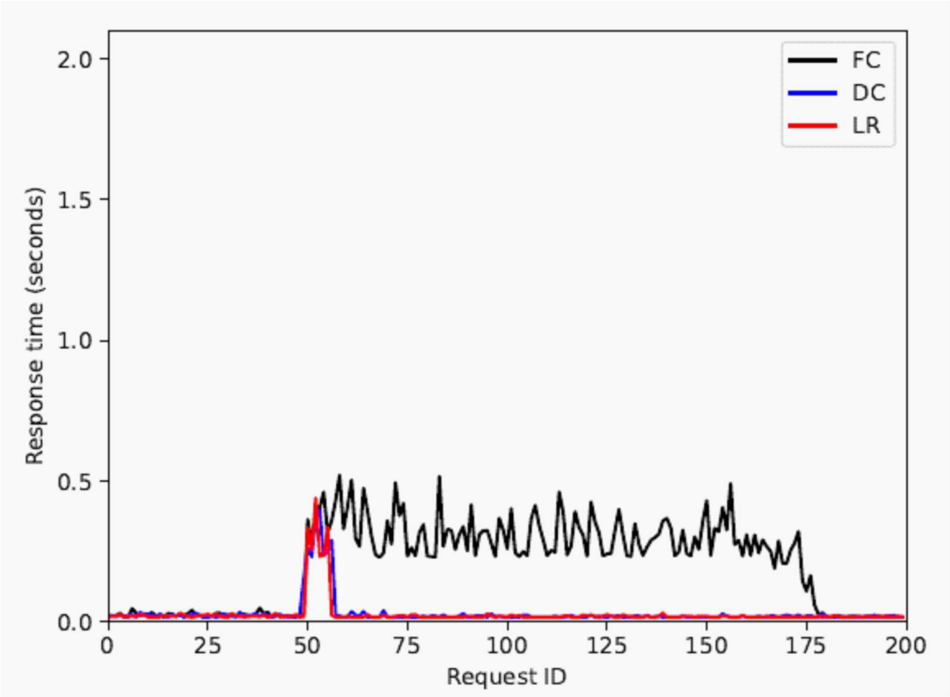
도면6


100
Server
Controller
Image DB
Association DB
Migration Decision
Event Reporter
110
Logger
Log Writer
Event Push
120
400
MU
AP
Event Reporter
Packer Relay
Replay Buffer
300
Edge Server
Profile Extractor
Migration Agent
Docker
Event Reporter
200

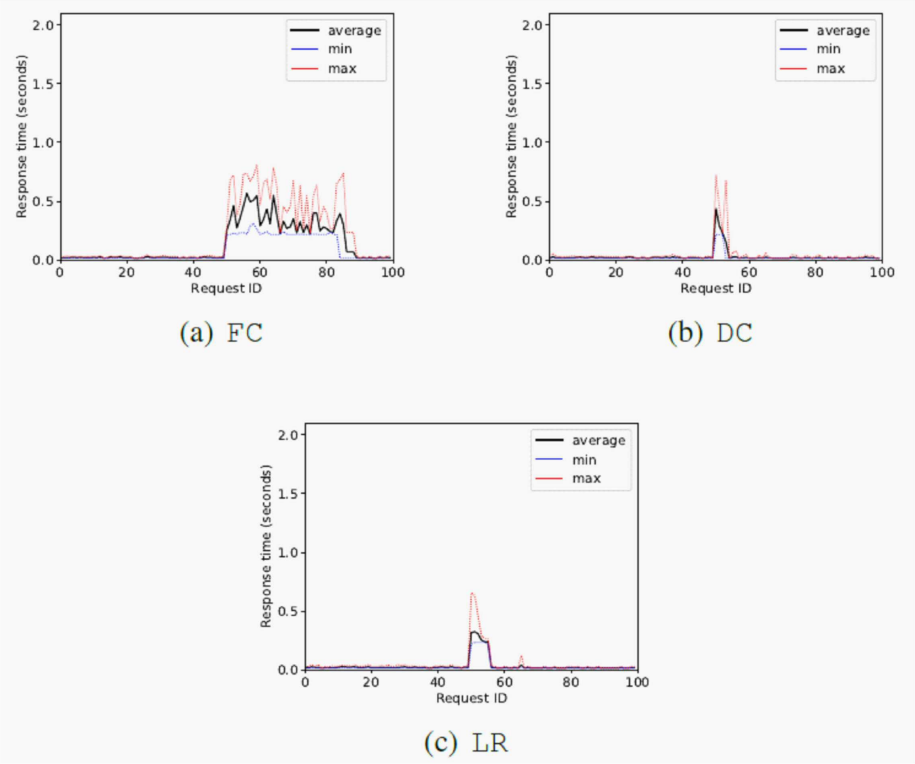
도면7


average
min
max
Response time (seconds)
Request ID
(a) FC
(b) DC
(c) LR

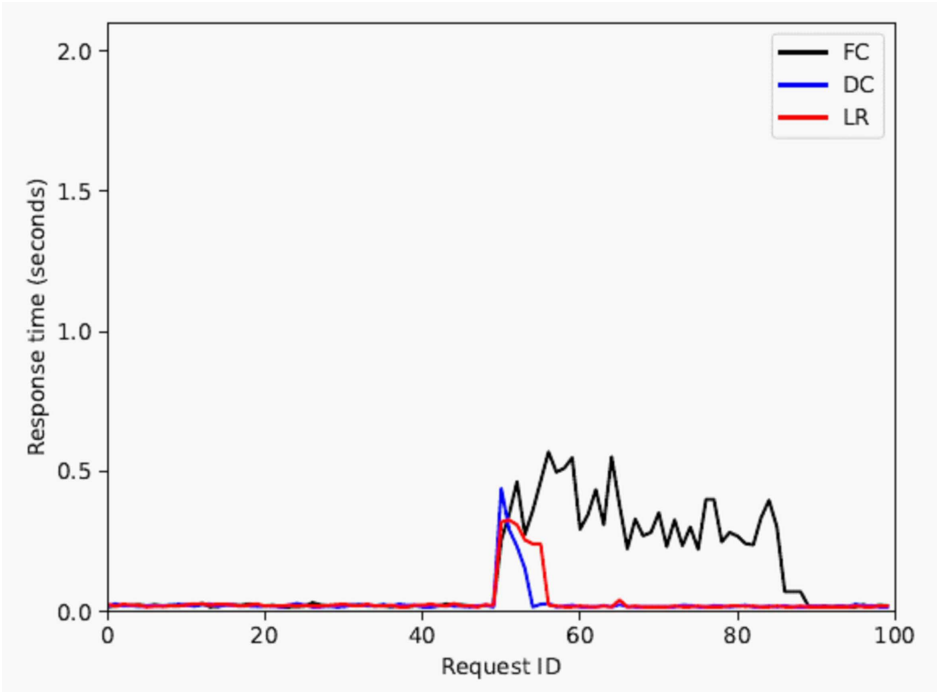
도면8


FC
DC
LR
Response time (seconds)
Request ID
2.0
1.5
1.0
0.5
0.0
0
25
50
75
100
125
150
175
200

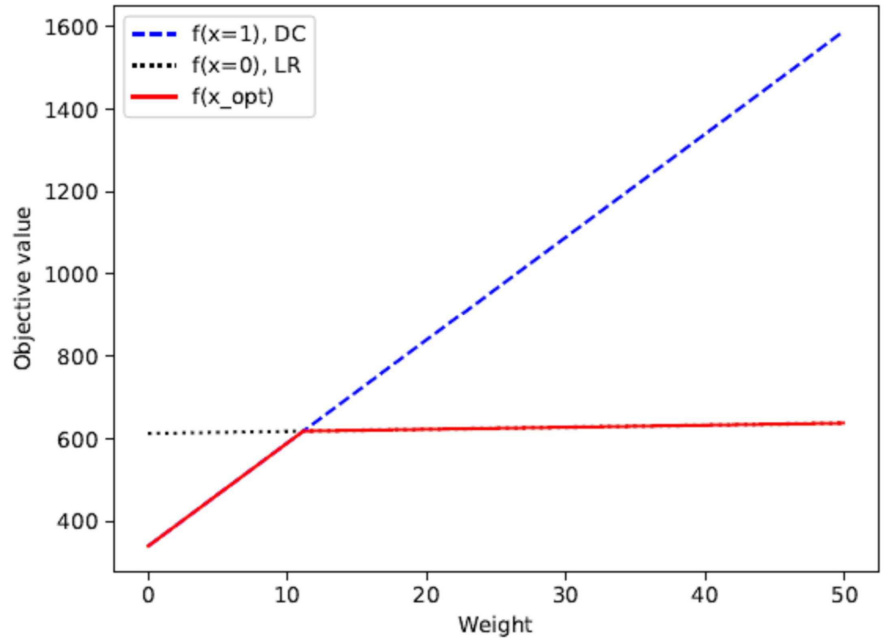
도면9


average
min
max
Response time (seconds)
Request ID
(a) FC
(b) DC
(c) LR

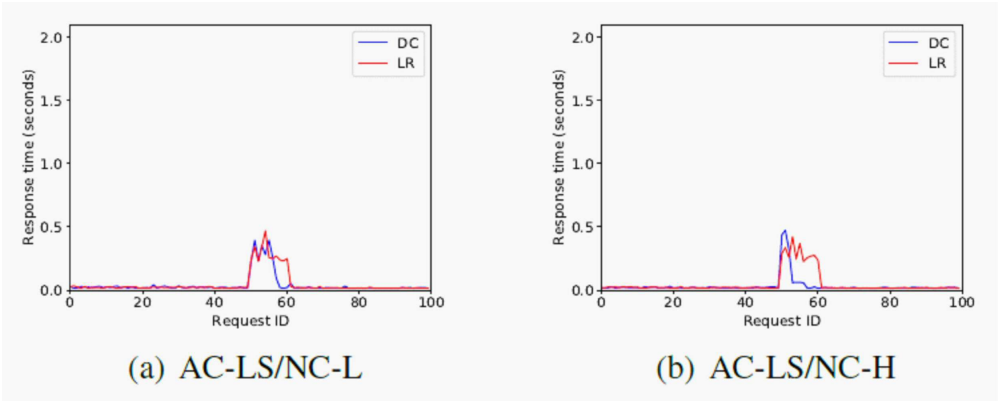
도면10


FC
DC
LR
Response time (seconds)
Request ID
0.0
0.5
1.0
1.5
2.0
0
20
40
60
80
100

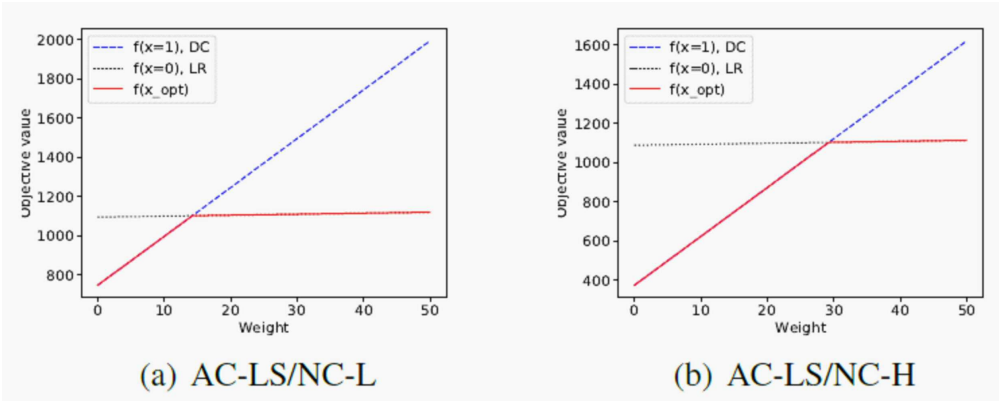
도면11


f(x=1), DC
f(x=0), LR
f(x_opt)
Objective value
Weight
400
600
800
1000
1200
1400
1600
0
10
20
30
40
50

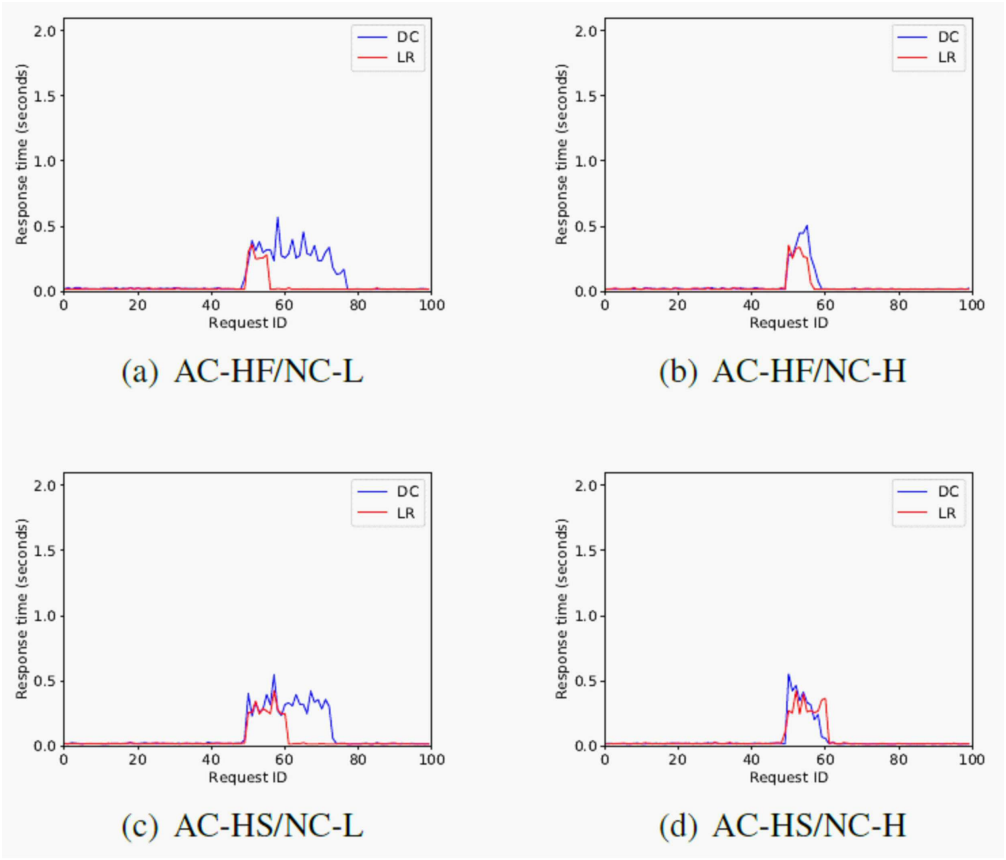
도면12


DC
LR
Response time (seconds)
Request ID
(a) AC-LS/NC-L
(b) AC-LS/NC-H

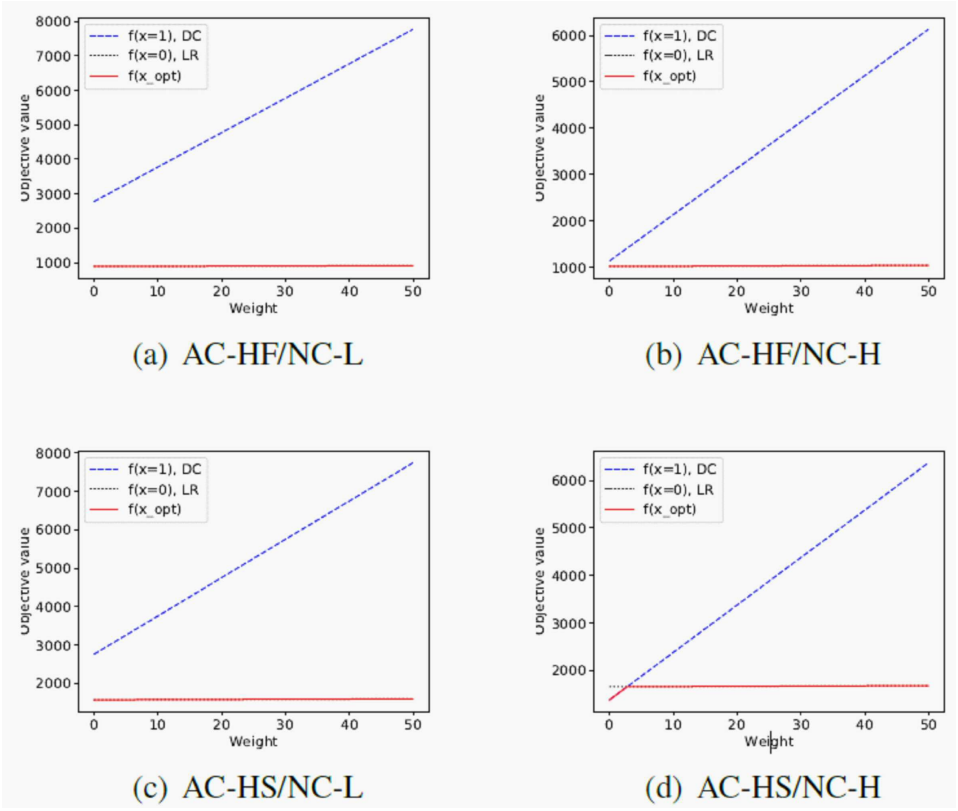
도면13


f(x=1), DC
f(x=0), LR
f(x_opt)
Objective value
Weight
(a) AC-LS/NC-L
(b) AC-LS/NC-H

도면14


DC
LR
Response time (seconds)
Request ID
(a) AC-HF/NC-L
(b) AC-HF/NC-H
(c) AC-HS/NC-L
(d) AC-HS/NC-H

도면15


f(x=1), DC
f(x=0), LR
f(x_opt)
Objective value
Weight
(a) AC-HF/NC-L
(b) AC-HF/NC-H
(c) AC-HS/NC-L
(d) AC-HS/NC-H