(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2023-0073371
(43) 공개일자 2023년05월25일

(51) 국제특허분류(Int. Cl.)
G06F 9/50 (2018.01) *G06F 16/955* (2019.01)
G06F 9/455 (2018.01) *G06F 9/48* (2018.01)
H04L 67/289 (2022.01) *H04L 67/52* (2022.01)
(52) CPC특허분류
G06F 9/5072 (2013.01)
G06F 16/955 (2019.01)
(21) 출원번호 10-2022-0132131
(22) 출원일자 2022년10월14일
심사청구일자 없음
(30) 우선권주장
63/280,001 2021년11월16일 미국(US)
17/561,334 2021년12월23일 미국(US)

(71) 출원인
인텔 코포레이션
미합중국 캘리포니아 95054 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
라구나트, 애런
미국 97225 오리건 포틀랜드 사우스웨스트 워싱턴 스트리트 10662
쵸두리, 모하메드
미국 97124 오리건 힐스보로 노스이스트 54번 웨이 1991
(뒷면에 계속)
(74) 대리인
양영준, 김연송, 백만기

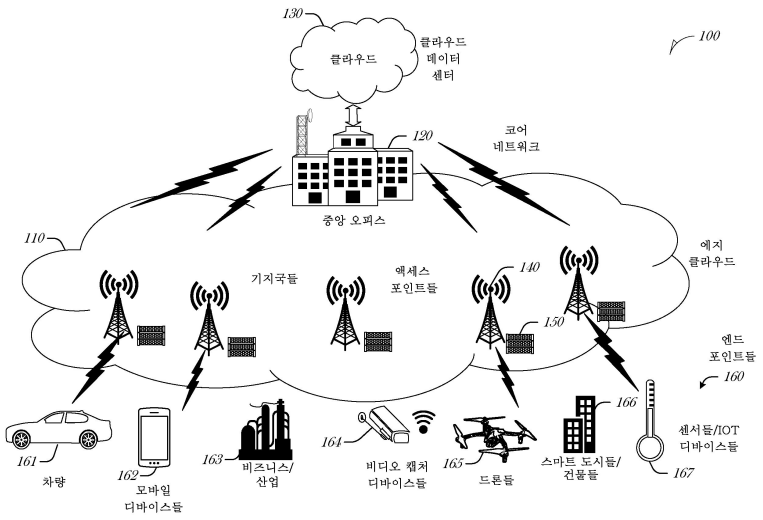
전체 청구항 수 : 총 25 항

(54) 발명의 명칭 **FAAS(FUNCTION-AS-A-SERVICE) 아키텍처에서의 계산 스토리지**

(57) 요 약

계산 스토리지를 구현하기 위한 다양한 시스템들 및 방법들이 본 명세서에 설명된다. 오케스트레이터 시스템은, 오케스트레이터 시스템에서, 등록 패키지를 수신하고 - 등록 패키지는 함수 코드, 함수 코드에 대한 입력 데이터의 논리적 로케이션, 및 함수 코드에 대한 이벤트 트리거를 포함하고, 이벤트 트리거는 입력 데이터가 수정될 때에 응답하여 트리거하도록 설정됨 -; 스토리지 서비스와 인터페이스하고 - 스토리지 서비스는 입력 데이터의 논리적 로케이션을 모니터링하고 입력 데이터가 수정될 때를 로케이션 서비스에 통지함 - ; 로케이션 서비스와 인터페이스하여 입력 데이터의 물리적 로케이션을 획득하고 - 로케이션 서비스는 입력 데이터의 논리적 로케이션으로부터 물리적 로케이션을 결정함 -; 및 입력 데이터 근처에서 실행되도록 함수 코드를 구성하도록 구성된다.

대 표 도



(52) CPC특허분류
G06F 9/45558 (2013.01)
G06F 9/4881 (2013.01)
G06F 9/5027 (2013.01)
H04L 67/289 (2022.05)
H04L 67/52 (2022.05)
G06F 2009/45583 (2019.08)
G06F 2009/45595 (2019.08)
(72) 발명자
메스니어, 마이클
미국 97056 오리건 스카포스 사우스웨스트 애슐리 코트 52365
아이어, 라비샹카 알.
미국 97229 오리건 포틀랜드 노쓰웨스트 타운센드 코트 5153
아담스, 이안
미국 97124 오리건 힐스보로 노스이스트 25번 애비뉴 2111
메치, 티즈스
독일 50321 엔더블유 브륄 월스트라쎄 33
브라운, 존 제이.
아일랜드 엘케이 아이이 리머릭 에스씨 로드 서머빌 애비뉴 컬리지 게이트 19
호반, 아드리안
아일랜드 브이95와이와이씨4 클레어 크래틀로이 캐슬쿼터
라마무르티, 베라라그하반
미국 95035 캘리포니아 밀피타스 나이트쉐이드 로드 1418 유닛 40
코에벌, 패트릭
미국 97229 오리건 포틀랜드 노스웨스트 찬나 드라이브 15200
구임 베르낫, 프란세스크
스페인 08036 바르셀로나 카소노바 148
도시, 크시티즈 아룬
미국 85282 애리조나 템페 이스트 발보아 드라이브 2101
발레, 수잔 엠.
미국 03051 뉴햄프셔 허드슨 말라드 드라이브 7
리, 빈
미국 97229 오리건 포틀랜드 노스웨스트 밀클리프 스트리트 16912

청구범위유예 : 있음
임시명세서출원 : 있음

명 세 서

청구범위

청구항 1

오케스트레이터 시스템으로서:

프로세서; 및

명령어들을 저장하는 메모리를 포함하고, 상기 명령어들은, 상기 프로세서에 의해 실행될 때, 상기 오케스트레이터 시스템으로 하여금:

상기 오케스트레이터 시스템에서, 등록 패키지를 수신하고 - 상기 등록 패키지는 함수 코드, 상기 함수 코드에 대한 입력 데이터의 논리적 로케이션, 및 상기 함수 코드에 대한 이벤트 트리거를 포함하고, 상기 이벤트 트리거는 상기 입력 데이터가 수정될 때에 응답하여 트리거하도록 설정됨 -;

스토리지 서비스와 인터페이스하고 - 상기 스토리지 서비스는 상기 입력 데이터의 논리적 로케이션을 모니터링하고 상기 입력 데이터가 수정될 때를 로케이션 서비스에 통지함 - ;

상기 로케이션 서비스와 인터페이스하여 상기 입력 데이터의 물리적 로케이션을 획득하고 - 상기 로케이션 서비스는 상기 입력 데이터의 논리적 로케이션으로부터 상기 물리적 로케이션을 결정함 -; 및

상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하게 야기하는 오케스트레이터 시스템.

청구항 2

제1항에 있어서, 상기 등록 패키지는 상기 입력 데이터에 액세스하기 위해 사용되는 크리덴셜을 포함하는 오케스트레이터 시스템.

청구항 3

제1항에 있어서, 상기 등록 패키지는 출력 데이터에 대한 논리적 로케이션을 포함하는 오케스트레이터 시스템.

청구항 4

제1항 내지 제3항 중 어느 한 항에 있어서, 상기 출력 데이터에 대한 논리적 로케이션은 URL(universal resource locator)인 오케스트레이터 시스템.

청구항 5

제1항에 있어서, 상기 스토리지 서비스는 비구조화된 객체 스토리지 서비스인 오케스트레이터 시스템.

청구항 6

제1항 내지 제3항 중 어느 한 항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하기 위해, 상기 오케스트레이터 시스템은 상기 입력 데이터의 물리적 로케이션에 있는 서버에 상기 함수 코드를 배치하는 오케스트레이터 시스템.

청구항 7

제1항 내지 제3항 중 어느 한 항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하기 위해, 상기 오케스트레이터 시스템은 상기 물리적 로케이션으로부터 상기 함수 코드가 실행되도록 스케줄링되는 컴퓨팅 노드로 상기 입력 데이터를 프리페치하는 오케스트레이터 시스템.

청구항 8

제1항 내지 제3항 중 어느 한 항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하기 위해, 상기 오케스트레이터 시스템은 상기 함수 코드의 복수의 인스턴스로부터 상기 함수 코드의 한 인스턴스를 선택하고, 상기 함수 코드의 상기 선택된 인스턴스는 적어도 하나의 성능 메트릭에 따라 상기 복수의 함수 코드

인스턴스 중 다른 것들보다 비교적 더 양호하게 수행된 한 컴퓨팅 노드에서 실행되는 오케스트레이터 시스템.

청구항 9

제1항 내지 제3항 중 어느 한 항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하기 위해, 상기 오케스트레이터 시스템은 상기 함수 코드의 복수의 인스턴스로부터 상기 함수 코드의 한 인스턴스를 선택하고, 상기 함수 코드의 상기 복수의 인스턴스는 대응하는 복수의 컴퓨팅 노드 상에서 실행되고, 상기 함수 코드의 상기 선택된 인스턴스는 상기 복수의 컴퓨팅 노드 중 다른 것들보다 상기 입력 데이터에 물리적으로 더 가까운 상기 복수의 컴퓨팅 노드 중 한 컴퓨팅 노드에서 실행되는 오케스트레이터 시스템.

청구항 10

방법으로서:

오케스트레이터 시스템에서, 등록 패키지를 수신하는 단계 - 상기 등록 패키지는 함수 코드, 상기 함수 코드에 대한 입력 데이터의 논리적 로케이션, 및 상기 함수 코드에 대한 이벤트 트리거를 포함하고, 상기 이벤트 트리거는 상기 입력 데이터가 수정될 때에 응답하여 트리거하도록 설정됨 -;

스토리지 서비스와 인터페이스하는 단계 - 상기 스토리지 서비스는 상기 입력 데이터의 논리적 로케이션을 모니터링하고 상기 입력 데이터가 수정될 때를 로케이션 서비스에 통지함 - ;

상기 로케이션 서비스와 인터페이스하여 상기 입력 데이터의 물리적 로케이션을 획득하는 단계 - 상기 로케이션 서비스는 상기 입력 데이터의 논리적 로케이션으로부터 상기 물리적 로케이션을 결정함 -; 및

상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하는 단계를 포함하는 방법.

청구항 11

제10항에 있어서, 상기 등록 패키지는 상기 입력 데이터에 액세스하기 위해 사용되는 크리덴셜을 포함하는 방법.

청구항 12

제10항에 있어서, 상기 등록 패키지는 출력 데이터에 대한 논리적 로케이션을 포함하는 방법.

청구항 13

제12항에 있어서, 상기 출력 데이터에 대한 논리적 로케이션은 URL(universal resource locator)인 방법.

청구항 14

제10항 내지 제13항 중 어느 한 항에 있어서, 상기 스토리지 서비스는 비구조화된 객체 스토리지 서비스인 방법.

청구항 15

제10항 내지 제13항 중 어느 한 항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하는 단계는 상기 입력 데이터의 물리적 로케이션에 있는 서버에 상기 함수 코드를 배치하는 단계를 포함하는 방법.

청구항 16

제10항 내지 제13항 중 어느 한 항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하는 단계는 상기 물리적 로케이션으로부터 상기 함수 코드가 실행되도록 스케줄링되는 컴퓨팅 노드로 상기 입력 데이터를 프리페치하는 단계를 포함하는 방법.

청구항 17

제10항 내지 제13항 중 어느 한 항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하는 단계는 상기 함수 코드의 복수의 인스턴스로부터 상기 함수 코드의 한 인스턴스를 선택하는 단계를 포함하고, 상기 함수 코드의 상기 선택된 인스턴스는 적어도 하나의 성능 메트릭에 따라 상기 함수 코드의 상기

복수의 인스턴스 중 다른 것들보다 비교적 더 양호하게 수행된 한 컴퓨팅 노드에서 실행되는 방법.

청구항 18

제10항 내지 제13항 중 어느 한 항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하는 단계는 상기 함수 코드의 복수의 인스턴스로부터 상기 함수 코드의 한 인스턴스를 선택하는 단계를 포함하고, 상기 함수 코드의 상기 복수의 인스턴스는 대응하는 복수의 컴퓨팅 노드 상에서 실행되고, 상기 함수 코드의 상기 선택된 인스턴스는 상기 복수의 컴퓨팅 노드 중 다른 것들보다 상기 입력 데이터에 물리적으로 더 가까운 상기 복수의 컴퓨팅 노드 중 한 컴퓨팅 노드에서 실행되는 방법.

청구항 19

머신에 의해 실행될 때, 상기 머신으로 하여금 동작들을 수행하게 야기하는 명령어들을 포함하는 적어도 하나의 머신 판독가능 매체로서, 상기 동작들은:

오케스트레이터 시스템에서, 등록 패키지를 수신하는 동작 - 상기 등록 패키지는 함수 코드, 상기 함수 코드에 대한 입력 데이터의 논리적 로케이션, 및 상기 함수 코드에 대한 이벤트 트리거를 포함하고, 상기 이벤트 트리거는 상기 입력 데이터가 수정될 때에 응답하여 트리거하도록 설정됨 -;

스토리지 서비스와 인터페이스하는 동작 - 상기 스토리지 서비스는 상기 입력 데이터의 논리적 로케이션을 모니터링하고 상기 입력 데이터가 수정될 때를 로케이션 서비스에 통지함 - ;

상기 로케이션 서비스와 인터페이스하여 상기 입력 데이터의 물리적 로케이션을 획득하는 동작 - 상기 로케이션 서비스는 상기 입력 데이터의 논리적 로케이션으로부터 상기 물리적 로케이션을 결정함 -; 및

상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하는 동작을 포함하는 적어도 하나의 머신 판독가능 매체.

청구항 20

장치로서:

오케스트레이터 시스템에서, 등록 패키지를 수신하기 위한 수단 - 상기 등록 패키지는 함수 코드, 상기 함수 코드에 대한 입력 데이터의 논리적 로케이션, 및 상기 함수 코드에 대한 이벤트 트리거를 포함하고, 상기 이벤트 트리거는 상기 입력 데이터가 수정될 때에 응답하여 트리거하도록 설정됨 -;

스토리지 서비스와 인터페이스하기 위한 수단 - 상기 스토리지 서비스는 상기 입력 데이터의 논리적 로케이션을 모니터링하고, 상기 입력 데이터가 수정될 때를 로케이션 서비스에 통지함 -;

상기 로케이션 서비스와 인터페이스하여 상기 입력 데이터의 물리적 로케이션을 획득하기 위한 수단 - 상기 로케이션 서비스는 상기 입력 데이터의 논리적 로케이션으로부터 상기 물리적 로케이션을 결정함 -; 및

상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하기 위한 수단을 포함하는 장치.

청구항 21

제20항에 있어서, 상기 등록 패키지는 상기 입력 데이터에 액세스하기 위해 사용되는 크리덴셜을 포함하는 장치.

청구항 22

제20항 또는 제21항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하는 것은 상기 입력 데이터의 물리적 로케이션에 있는 서버에 상기 함수 코드를 배치하는 것을 포함하는 장치.

청구항 23

제20항 또는 제21항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하는 것은 상기 물리적 로케이션으로부터 상기 함수 코드가 실행되도록 스케줄링되는 컴퓨팅 노드로 상기 입력 데이터를 프리페치하는 것을 포함하는 장치.

청구항 24

제20항 또는 제21항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하는 것은 상기 함수 코드의 복수의 인스턴스로부터 상기 함수 코드의 한 인스턴스를 선택하는 것을 포함하고, 상기 함수 코드의 상기 선택된 인스턴스는 적어도 하나의 성능 메트릭에 따라 상기 함수 코드의 상기 복수의 인스턴스 중 다른 것들보다 비교적 더 양호하게 수행된 한 컴퓨팅 노드에서 실행되는 장치.

청구항 25

제20항 또는 제21항에 있어서, 상기 입력 데이터 근처에서 실행되도록 상기 함수 코드를 구성하는 것은 상기 함수 코드의 복수의 인스턴스로부터 상기 함수 코드의 한 인스턴스를 선택하는 것을 포함하고, 상기 함수 코드의 상기 복수의 인스턴스는 대응하는 복수의 컴퓨팅 노드 상에서 실행되고, 상기 함수 코드의 상기 선택된 인스턴스는 상기 복수의 컴퓨팅 노드 중 다른 것들보다 상기 입력 데이터에 물리적으로 더 가까운 상기 복수의 컴퓨팅 노드 중 한 컴퓨팅 노드에서 실행되는 장치.

발명의 설명

기 술 분 야

[0001] [우선권]

[0002] 본 출원은 2021년 11월 16일자로 출원된 미국 가특허 출원 제63/280,001호를 기초로 우선권을 주장하며, 이 미국 출원은 참조에 의해 그 전체 내용이 본 명세서에 포함된다.

[0003] [기술 분야]

[0004] 본 명세서에 설명된 실시예들은 일반적으로 네트워크 모니터링 및 튜닝에 관한 것으로, 특히 계산 스토리지(computational storage)를 구현하기 위한 시스템 및 방법에 관한 것이다.

배 경 기 술

[0005] 일반적인 레벨에서, 에지 컴퓨팅(edge computing)은 총 소유 비용을 최적화하고, 애플리케이션 레이턴시를 감소시키고, 서비스 능력을 개선하고, 보안 또는 데이터 프라이버시 요건들의 준수를 개선하기 위해서 엔드포인트 디바이스들(예를 들어, 소비자 컴퓨팅 디바이스들, 사용자 장비 등)에 더 가깝게 컴퓨팅 및 스토리지 리소스들을 전이(transition)하는 것을 지칭한다. 일부 시나리오들에서, 에지 컴퓨팅은 많은 타입들의 스토리지 및 컴퓨팅 리소스들 사이에서 애플리케이션들에 대한 오케스트레이션 및 관리를 제공하는 클라우드 유사 분산형 서비스(cloud-like distributed service)를 제공할 수 있다. 그 결과, 이전에 대규모 원격 데이터 센터들에서만 이용가능했던 강력한 컴퓨팅 리소스들이 엔드포인트들에 더 가깝게 이동되고 네트워크의 "에지"에서의 소비자들에 의한 사용을 위해 이용가능하게 됨에 따라, 에지 컴퓨팅의 일부 구현들은 "에지 클라우드(edge cloud)" 또는 "포그(fog)"라고 지칭된다.

[0006] 모바일 네트워크 설정에서의 에지 컴퓨팅 사용 사례들이, "모바일 에지 컴퓨팅"으로도 알려진, 멀티 액세스 에지 컴퓨팅(multi-access edge computing, MEC) 접근법들과의 통합을 위해 개발되었다. MEC 접근법들은 애플리케이션 개발자들 및 콘텐츠 제공자들이 네트워크의 에지에서의 동적 모바일 네트워크 설정으로 컴퓨팅 능력들 및 정보 기술(IT) 서비스 환경에 액세스하게 허용하도록 설계된다. MEC 시스템들, 플랫폼들, 호스트들, 서비스들, 및 애플리케이션들의 동작을 위한 공통 인터페이스들을 정의하기 위해 ETSI(European Telecommunications Standards Institute) ISG(industry specification group)에 의해 제한된 표준들이 개발되었다.

[0007] 에지 컴퓨팅, MEC, 및 관련 기술들은 전통적인 클라우드 네트워크 서비스들 및 광역 네트워크 접속들에서 제공되는 것보다 감소된 레이턴시, 증가된 응답성, 및 더 많은 이용가능한 컴퓨팅 능력을 제공하려고 시도한다. 그러나, 이동성 및 동적으로 론칭되는 서비스들의 일부 모바일 사용 및 디바이스 처리 사용 사례들로의 통합은, 특히 많은 참여자들(디바이스들, 호스트들, 테넌트(tenant)들, 서비스 제공자들, 운영자들)이 수반되는 복잡한 이동성 설정에서, 오케스트레이션, 기능적 조정, 및 리소스 관리에 대한 한계 및 우려로 이어졌다. 유사한 방식으로, 사물 인터넷(Internet of Things, IoT) 네트워크들 및 디바이스들이 다양한 엔드포인트들로부터 분산형 컴퓨팅 배열(distributed compute arrangement)을 제공하도록 설계된다. IoT 디바이스들은 네트워크 상에서 통신할 수 있는 물리적인 또는 가상화된 객체들이고, 데이터를 수집하거나 실세계 환경에서 액션들을 수행하기 위해 사용될 수 있는 센서들, 액추에이터들, 및 다른 입력/출력 컴포넌트들을 포함할 수 있다. 예를 들어, IoT

디바이스들은, 건물들, 차량들, 패키지들 등과 같은 매일의 사물들에 임베드되거나 부착되어 그러한 사물들의 추가 레벨의 인공 감각 인식을 제공하는 저전력 엔드포인트 디바이스들을 포함할 수 있다. 최근에, IoT 디바이스들이 더 대중화되었고 따라서 이들 디바이스를 사용하는 애플리케이션들이 급증하였다.

[0008] 다양한 에지(Edge), 포그(Fog), MEC, 및 IoT 네트워크들, 디바이스들, 및 서비스들의 배치는 네트워크의 에지에서 그리고 네트워크의 에지를 향해 발생하는 몇몇 진보된 사용 사례들 및 시나리오들을 도입하였다. 그렇지만, 이 진보된 사용 사례들은 또한, 많은 다른 문제들 중에서도, 특히 보다 많은 타입들의 컴퓨팅 시스템들 및 구성들이 배치됨에 따라, 오케스트레이션, 보안, 처리 및 네트워크 리소스들, 서비스 가용성 및 효율성, 서비스 품질 보장에 관련된 몇 가지 대응하는 기술적 도전 과제들을 끌어들였다.

발명의 내용

도면의 간단한 설명

[0009] 반드시 실제 축척(scale)으로 그려진 것은 아닌 도면에서, 유사한 도면 번호가 상이한 뷰들 내의 유사한 컴포넌트를 기술할 수 있다. 상이한 문자 접미사를 갖는 유사한 도면 번호가 유사한 컴포넌트의 상이한 경우를 표현할 수 있다. 일부 실시예들은 첨부 도면들의 그림들에서 제한이 아닌 예로서 도시된다.

도 1은 실시예에 따른, 에지 컴퓨팅을 위한 에지 클라우드 구성의 개관을 도시한다.

도 2는 실시예에 따른, 엔드포인트들, 에지 클라우드, 및 클라우드 컴퓨팅 환경들 간의 동작 계층들을 도시한다.

도 3은 실시예에 따른, 에지 컴퓨팅 시스템에서의 네트워킹 및 서비스들을 위한 예시적인 접근법을 도시한다.

도 4는 실시예에 따른, 다중의 에지 노드 및 다중의 테넌트 사이에서 동작되는 에지 컴퓨팅 시스템에서의 가상 에지 구성의 배치를 도시한다.

도 5는 실시예에 따른, 에지 컴퓨팅 시스템에서 컨테이너들을 배치하는 다양한 컴퓨팅 배열들을 도시한다.

도 6a는 실시예에 따른, 에지 컴퓨팅 시스템에서의 컴퓨팅 노드에 배치된 컴퓨팅을 위한 예시적인 컴포넌트들의 개관을 제공한다.

도 6b는 실시예에 따른, 에지 컴퓨팅 시스템에서의 컴퓨팅 디바이스 내에서의 예시적인 컴포넌트들의 추가 개관을 제공한다.

도 7은 실시예에 따른, 도 6b의 예시적인 컴퓨터 판독가능 명령어와 같은 소프트웨어를 하나 이상의 디바이스에 배포하는 예시적인 소프트웨어 배포 플랫폼을 도시한다.

도 8은 실시예에 따른, 서버리스 데이터 센터를 도시하는 블록도이다.

도 9는 실시예에 따른, 다중의 하드웨어 시스템을 갖는 운영 환경을 도시하는 블록도이다.

도 10은 실시예에 따른, 오케스트레이션 제어 평면을 도시하는 블록도이다.

도 11은 실시예에 따른, 오케스트레이션 시스템에서의 데이터 및 제어 흐름을 도시하는 블록도이다.

도 12는 실시예에 따른, 의도 기반 오케스트레이션을 구현하는 방법을 도시하는 흐름도이다.

도 13은 실시예에 따른, 실시예에 따른 계산 스토리지를 오케스트레이션하기 위한 데이터 및 제어 흐름을 도시하는 블록도이다.

도 14는 실시예에 따른, 계산 스토리지를 구현하기 위한 방법을 도시하는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0010] 본 명세서에 설명된 시스템들 및 방법들은 FaaS(function as a service) 환경에서의 계산 스토리지를 제공한다. 현행의 오케스트레이션 솔루션들은 QoS(quality of service) 관리를 달성하는 데 매우 임페러티브(imperative)한 방식을 사용한다. 임페러티브라는 것은 특정 규칙들, 제약들, 임계값들, 플랫폼 파라미터들, 및 그와 유사한 것을 사용하여 QoS를 달성하는 방법을 표현하는 것을 지칭한다. QoS 관리는 리소스들의 정확한 양(예를 들어, vCPU(virtual central processing unit)들의 수), 또는 클라우드 및 에지에서 작업부하들을 지원하기 위한 하드웨어 피처들을 요청하는 것에 관하여 주로 구축된다. 이러한 방식으로 오케스트레이션을 수행하는 것은 몇

가지 문제들을 갖는다. 이것은, 어느 한 벤더에 의해 특정된 특정한 피처 세트들이 또 다른 벤더에 의해 제공된 장비에 의해 이용가능하지 않을 수 있기 때문에, 원치않는 벤더-록 인(vendor-lock in)을 생성한다. 또한, 경험은 부정확한 정보가 QoS 요청에서 선언되고 따라서 차선의 성능으로 이어진다는 것을 보여준다. 마지막으로, 이러한 선언들은 QoS 관리 스킴에 포함되지 않는 정황을 갖는다. 예를 들어, 애플리케이션이 실행되고 있는 플랫폼의 타입(예를 들어, Xeon 타입 코어 대 Atom 기반 코어 프로세서들)이 특정 QoS 요청을 만족시키기 위한 프로비저닝 판정들에 영향을 미칠 수 있다.

[0011] 더욱이, 작업부하가 모놀리식 스타일로부터 마이크로서비스 스타일로 이동함에 따라, 고객들이 작업부하들에 대한 정확한 리소스 할당들을 스스로 선택하는 것이 더 어려워진다. 이는 고객들이 리소스들을 오버프로비저닝하는 것으로 이끌 수 있고, 이는 더 높은 비용을 초래한다.

[0012] 하드웨어 플랫폼들이 (예컨대, 단일 CPU 대신에 다중의 XPU를 사용하여) 보다 이질적이 됨에 따라, 사용자가 오케스트레이션 요청에서 다수의 정황 상세 사항들을 정의할 필요가 있는 것으로부터 추상화되어야 하고, 대신에 플레이 중인 서비스들에 대한 특정 목표 세트를 달성하고자 자신이 원하는 것에 집중할 수 있어야 하는 것이 중요하다. 의도 주도 모델(intent-driven model)이 이 추상화를 사용자에게 제공하고, 그 결과로 서비스 소유자에 대한 양호한 성능은 물론이고 리소스 소유자에 대한 양호한 투자 수익(return on investment)을 가져온다. 따라서, 요구되는 레벨의 서비스 품질을 달성하기 위해 한 세트의 시스템들과 그들의 리소스들에 걸쳐 (서비스 레벨 목표로서 정의되는) 의도를 매핑하는 방법이 필요하다.

[0013] SLA(service level agreement)는 서비스 제공자와 클라이언트 사이의 계약이다. SLA들은 서비스 타겟들 또는 다른 비즈니스 목표들을 약술하는 제3자들 사이의 계약들이다. SLA들은 타겟을 만족시키는데 실패한 당사자들에 대한 페널티들을 약술할 수 있다. 예를 들어, 서비스가 90일 기간에 걸쳐 어떤 임계 백분율보다 많은 동안 이용 가능하지 않을 때 수수료의 부분 환급이 시행될 수 있다.

[0014] SLO(service level objective)들은 다양한 시스템 능력들에 대한 정확한 수치적 타겟들을 제공한다. 전형적인 SLO들은 서비스 가용성 또는 가동 시간, 서비스 레이턴시, 서비스 대역폭 프로비저닝 등을 중심에 둔다. SLO들은 "30일 기간에 걸쳐 99.9% 가동 시간" 또는 "수신된 요청들의 적어도 95%에 대해 100ms 미만의 응답 시간"을 요구하는 것과 같이, 백분율들로서 종종 표현된다.

[0015] KPI(key performance indicator)들은 특정 목표에 대한 시간 경과에 따른 성능의 정량화가능한 척도들이다. SLO 정책들을 측정하기 위해 KPI 메트릭들이 수집된다. 예시적인 KPI들은 서비스 에러들의 수, 가동 휴지 시간, 캐시 미스들, FPS(frames per second), 레이턴시, IPC(instructions per cycle), 보안 장애들, 실패한 로그인들 등을 포함하지만, 이에 제한되지는 않는다. KPI들은 도출된 값들(예를 들어, 이동하는 30초 윈도우에 걸친 평균 레이턴시)일 수 있다. KPI들은 또한 비교들(예를 들어, 이력 평균과 상이한 퍼센트)에 기초할 수 있다.

[0016] 서비스 품질 목표들은 계층화된 하향식(top-down) 구조를 형성하지만 트리 또는 DAG 패턴에 있는 것으로 제약되지는 않는 통계적 및 결정론적 사양들로서 표현될 수 있다. 예를 들어, KPI(key performance indicator)들 및 이들이 전체 QoS(quality of service) 또는 SLO(Service Level Objective)를 평가하기 위하여 어떻게 조합되는지를 표현하기 위하여 메타 언어가 이용될 수 있다. KPI들에 대한 다양한 범위들, 비교들, 임계값들, 제약들, 또는 그와 유사한 것이 메타 언어를 이용하여 표현될 수 있다. 대부분의 실세계 소프트웨어 솔루션들은 성능, 스케일, 정확도 및 가용성에 대한 많은 미묘한 요구들을 충족시켜야 하며, 이러한 요구들은 항상 또는 모든 상황들에 대해 반드시 고정되지는 않는다. C 또는 파이썬(Python)과 같은 언어로 된 프로그램이 무엇이 계산될지를 표현할 시에 프로그래머에게 높은 정도의 유연성을 허용하는 것처럼, 리소스의 스케줄러에 의해 충족될 필요가 있는 동적 요건을 표현하는 메타 언어가 사용될 수 있다. 이러한 메타 언어로 작성된 메타 프로그램은, 예를 들어, 프로그램들, 런타임들, 운영 체제들, 인프라스트럭처 서비스들, 네트워킹 기어(networking gear), 및 등등의 거동들에 대해 수행되는 다양한 캘리브레이션들을 반영하는 변수들을 도입할 수 있다. 그 후, 이러한 변수들은 메타프로그램들의 더 높은 레벨의 변수들 및 결과들이 되도록 짜여지고, 다음 차례로 메타프로그램들로부터의 정정 액션, 반응 액션, 경고 액션, 및 다른 액션을 구동한다. 메타 프로그램들이 상세 사항의 추상화 및 축소를 위해 그리고 더 높은 레벨들에서 특정된 목표에의 수렴을 위해 설계되도록 구조화가 계층화된다.

[0017] 정책들, 휴리스틱들(heuristics) 등의 계층화가 로직 단독의 규칙들에 의해 제한되지 않고, 신경망들, SVM(support vector machine)들, 결정 트리들 등과 같은 데이터 프로그래밍된 지능을 플러그인(plug-in)하는 능력을 허용하도록 개방형 아키텍처가 제공된다. 전체적인 목표는 컴퓨팅의 일부분을 이러한 메타 프로그램들로 리디렉션(redirection)하고 애플리케이션의 구조의 일부인 계산과 애플리케이션의 실행 동안 서비스 품질 목

표를 충족시키도록 설계되는 계산 간에 엄격한 분할을 짓는 것을 피하는 것이다. 따라서, 쿠버네티스(Kubernetes)와 같은 제어/관리 평면과 그것이 그 하에서 실행되는 컨테이너화된 애플리케이션 사이의 보통의 구별은 흐릿해지고, 의도에 관한 정보 및 그 의도에 대한 성능이 상이한 깊이 계층들에서 양방향으로 흐르도록 허용된다. 따라서, 자동 정정 및 자동 표시 능력들이 서비스 목표들 및 서비스 품질 목표들에서의 흐름들 사이에서 공동 엔지니어링될 수 있다. 이점은 소프트웨어가 그것이 활성화되는 환경들과의 자기 적응형 또는 공동 적응형 수용을 달성하기 시작한다는 점이다.

[0018] SLA(Service Level Agreements) 만족도를 최대화하려고 시도하는 통계적 승인 제어 정책들이 이용가능하고 지터 민감 네트워크 시스템들에 관련된다. 그러나, 추상화가 상향으로 이동하고 서버리스 시스템들 또는 고도로 가상화된 것에서의 스케줄링이 구현되고 컨테이너들을 통한 리소스들의 적시(just-in-time) 할당들이 요구 시에 생성됨에 따라, 리소스들을 스케줄링하는 복잡성이 계속 증가한다. 매우 큰 데이터 센터들은 반응성 로드 밸런싱을 수행하기 위한 상당한 양의 용량을 준비할 수 있고, 가장 극단적인 요구 버스트들을 제외한 모든 것들에의 승인 거절들을 연기할 수 있지만, 컷오프(cutoff)가 아웃라이어 테일 레이턴시(outlier tail latency)에 대한 것이기 때문에, 인프라스트럭처의 보다 덜 탄력적인 제공자들은 통계적 정책들 하에서도 어려운 호출들을 해야만 한다. 레이턴시 푸시아웃(latency pushout)들의 캐스케이드를 야기할 수 있는 큰 버스트들을 요구 시에 존중하는 것은 극히 어렵다. 대안으로서, 본 명세서에 설명된 시스템들 및 방법들은 네스팅되고 서서히 변화되는 SLA들을 구현하고 따라서 버스트 수용적이다.

[0019] 레이트 제한된 서버에서 단일 큐를 갖는 토큰 버킷 모델(token-bucket model)을 상상해보자: 가장 새로운 도착에 대한 통계적 투사(statistical projection)는 새로운 도착에 의해 보이는 큐 길이에 비례하는 평균 응답 시간을 갖는다. 순진한 정책(naive policy)은 이 투사된 응답 시간이 테일 레이턴시 제약조건(tail latency constraint)을 위반하는 경우 새로운 도착을 거부하는 것일 것이다; 약간 덜 순진한 정책은 큐 길이 및 이미 발생한 위반들의 수에 좌우되어 새로운 도착을 확률적으로 거절하는 것일 것이다. 이제 기준선 단일 큐, 단일 레이트 서버가 가장 새로운 도착이, 더 낮은 서비스 레이트가 보장되지만 더 완화된 테일 레이턴시가 허용되는 제2 큐에 배치되는 2 레벨 정책이 되도록 스플릿되는 일반화를 상상해보자. 따라서, 예를 들어, 제1 큐가 10ms의 P99 레이턴시를 갖는 경우, 제2 큐는 10ms의 P95 레이턴시와 20ms의 P99 레이턴시를 갖는 조합 SLO를 가질 수 있다. 극단적인 버스트들이 드물기 때문에, 제2 큐에 전용되는 분수 용량은 그 평균 큐 길이가 작기 때문에 높을 필요가 없다. 데이터 센터에서, 제2 큐로의 이동은 한계 요청(엄격한 레이턴시 경계를 위반할 가능성이 있는 것)을, 대응하여 더 완화된 SLO를 갖는 덜 활용되지만 리소스가 덜 주어지는 오버플로우 클러스터에 효과적으로 재할당한다. 이러한 스킴은 제2 큐가 응답 시간 동안 그의 포화에 접근함에 따라 훨씬 더 자유로운 SLO를 갖는 제3 큐가 오버플로우를 흡수하도록 일반화될 수 있다. 이러한 방식으로, 버스트들에 적응적인 SLA 내에서의 복합 네스팅된 SLO들이 제작될 수 있다. 제공자는 비탄력적 SLA를 충족시키기 위해 많은 피크 용량을 프로비저닝해야 할 필요가 없는 것으로부터 생기는 절감이 네스팅된 SLA를 수락하는 고객들에게 되돌려 넘겨지는 유사하게 서서히 변화되는 비용 모델을 협상할 수 있다.

[0020] 계층적 서비스 레벨 협약들과는 대조적으로, 여기서 설명된 시스템들 및 방법들은 네스팅되고 또한 서서히 변화되는 SLA들을 도입한다. 고전적인 임계화에 의존하는 대신에, 그리고 검사할 굳게 설정된 "단일" 절 규칙(hard set “single” clause rule)들을 갖는 대신에, SLA의 네스팅된 하위 절은 전체 SLA를 평가하기 위해 다른 하위 절들과 조합되어 평가될 수 있다. 이는 더 복잡한 규칙들을 허용한다. 또한, 하위 절들의 "파싱(parsing)"은 각각의 절의 결과들이 미사용 리소스들의 공유를 생성하도록 허용한다. 이 접근법은 SLA 규칙들에서의 더 많은 유연성 및 클러스터 리소스들의 더 나은 사용을 허용한다. 이 유연성은 리소스 사양이 아닌 "의도(intent)"로서 도입될 수 있고, 이것은 그 후 네스팅된/서서히 변화되는 SLA 규칙들에 매핑되고, 이것들은 그 후 모니터링되고 시행된다.

[0021] 생산 배치들에서, 공통 패턴은 환경의 일부분에 애플리케이션의 새로운 인스턴스들을 배치하고 더 넓은 모집단으로 롤 아웃하기 전에 사용자 베이스의 작은 서브세트로 테스트하는 것이다(즉, 카나리 롤 아웃(Canary roll-out)). 서비스 레벨 목표들의 점점 더 낮은 레벨들로의 SLA 매핑이 (최적화 및 교정 둘 다의 목적을 위해) 반복적 접근법을 요구한다는 것을 인정하면, 이 SLA 분해에 대한 새로운 워크플로우는 SLA 준수에 대한 영향을 결정하기 위해 제한된 엔드-투-엔드(end-to-end) 방식으로 또는 E2E 솔루션의 서브 컴포넌트들로서 부분적으로 배치할 잠재성을 또한 포함한다.

[0022] 요약하면, 본 명세서에 기술되는 시스템들 및 방법들은, 현행의 모델로부터, 고객이 의도(예컨대, 레이턴시, 스루풋, 및 신뢰성 속성들)만을 표현하고 오케스트레이션 스택 자체가 그 의도를 달성하기 위해 플랫폼을 셋업하는 목표 주도 접근법으로 이동하여, 오케스트레이션을 달성하는 새로운 방식을 제공한다. 서버리스 인프라스트

럭처가 보다 일반화됨에 따라, 클라우드 네이티브(마이크로서비스) 스타일 작업부하들은 결국에는 의도 주도 오케스트레이션 방식을 요구할 것이다. 이러한 함수들 및 다른 것들이 아래에 더 상세히 설명된다.

[0023] 도 1은 다음의 예들 중 다수에서 "에지 클라우드"라고 지칭되는 처리의 계층을 포함하는, 에지 컴퓨팅을 위한 구성의 개관을 보여주는 블록도(100)이다. 도시된 바와 같이, 에지 클라우드(110)가 액세스 포인트 또는 기지국(140), 로컬 처리 허브(150), 또는 중앙 오피스(120)와 같은 에지 로케이션에 공동 배치되고, 따라서 다중의 엔티티, 디바이스, 및 장비 인스턴스를 포함할 수 있다. 에지 클라우드(110)는 클라우드 데이터 센터(130)보다 엔드포인트(소비자 및 생산자) 데이터 소스들(160)(예를 들어, 자율 차량들(161), 사용자 장비(162), 비즈니스 및 산업 장비(163), 비디오 캡처 디바이스들(164), 드론들(165), 스마트 도시들 및 빌딩 디바이스들(166), 센서들 및 IoT 디바이스들(167) 등)에 훨씬 더 가깝게 위치된다. 에지 클라우드(110) 내의 에지들에 제공되는 컴퓨팅, 메모리, 및 스토리지 리소스들은 엔드포인트 데이터 소스들(160)에 의해 사용되는 서비스들 및 함수들에 대한 초저 레이턴시 응답 시간들을 제공할 뿐만 아니라 에지 클라우드(110)로부터 클라우드 데이터 센터(130)를 향하는 네트워크 백홀 트래픽을 감소시키는 데에 중요하고, 따라서 여러 이점들 중에서도 특히 에너지 소비 및 전체 네트워크 사용을 개선한다.

[0024] 컴퓨팅, 메모리, 및 스토리지는 부족한 리소스들이고, 일반적으로 에지 로케이션에 좌우되어 감소한다(예를 들어, 기지국에서보다, 중앙 오피스에서보다 소비자 엔드포인트 디바이스에서 이용가능한 처리 리소스가 더 적다). 그러나, 에지 로케이션이 엔드포인트(예를 들어, UE(user equipment))에 더 가까울수록, 그 공간 및 전력이 종종 더 많이 제약된다. 따라서, 에지 컴퓨팅은 지리적으로 그리고 네트워크 액세스 시간 둘 모두에서 더 가깝게 위치되는 더 많은 리소스들의 분포를 통해, 네트워크 서비스들을 위해 필요한 리소스들의 양을 감소시키려고 시도한다. 이러한 방식으로, 에지 컴퓨팅은 적절한 경우 컴퓨팅 리소스들을 작업부하 데이터로 가져오거나, 또는 작업부하 데이터를 컴퓨팅 리소스들로 가져오려고 시도한다.

[0025] 이하에서는 다중의 잠재적 배치를 커버하고 그리고 일부 네트워크 운영자들 또는 서비스 제공자들이 그들 자신의 인프라스트럭처들에서 가질 수 있는 제한들을 해결하는 에지 클라우드 아키텍처의 양태들을 설명한다. 이것들은 (기지국 레벨에서의 에지들이 예를 들어 멀티 테넌트 시나리오에서 더 제약된 성능 및 능력을 가질 수 있기 때문에) 에지 로케이션에 기초한 구성들의 변형; 에지 로케이션들, 로케이션들의 티어(tier)들, 또는 로케이션들의 그룹들에 이용가능한 컴퓨팅, 메모리, 스토리지, 패브릭, 가속, 또는 유사한 리소스들의 타입에 기초한 구성들; 서비스, 보안, 및 관리 및 오케스트레이션 능력들; 및 최종 서비스들의 유용성 및 성능을 달성하기 위한 관련 목표들을 포함한다. 이들 배치들은 레이턴시, 거리, 및 타이밍 특성들에 좌우되어, "근거리 에지(near edge)", "가까운 에지(close edge)", "로컬 에지(local edge)", "중간 에지(middle edge)", 또는 "원거리 에지(far edge)" 계층들로 간주될 수 있는 네트워크 계층들에서의 처리를 달성할 수 있다.

[0026] 에지 컴퓨팅은 네트워크의 "에지(Edge)"에서 또는 그에 더 가까운 곳에서, 전형적으로, 데이터를 생산하고 소비하는 엔드포인트 디바이스들에 훨씬 더 가까운 기지국들, 게이트웨이들, 네트워크 라우터들, 또는 다른 디바이스들에서 구현되는 컴퓨팅 플랫폼(예를 들어, x86 또는 ARM 컴퓨팅 하드웨어 아키텍처)의 사용을 통해 컴퓨팅이 수행되는 개발 패러다임이다. 예를 들어, 에지 게이트웨이 서버들은 접속된 클라이언트 디바이스들에 대한 저 레이턴시 사용 사례들(예를 들어, 자율 주행 또는 비디오 감시)에 대해 실시간으로 계산을 수행하기 위해 메모리 및 스토리지 리소스들의 풀을 갖출 수 있다. 또는 예로서, 기지국들은 백홀 네트워크들을 통해 데이터를 추가로 통신하지 않고서, 접속된 사용자 장비에 대한 서비스 작업부하들을 직접 처리하기 위해 컴퓨팅 및 가속 리소스들로 증강될 수 있다. 또는 또 다른 예로서, 중앙 오피스 네트워크 관리 하드웨어는 가상화된 네트워크 함수들을 수행하고 접속된 디바이스들에 대한 서비스들 및 소비자 함수들의 실행을 위한 컴퓨팅 리소스들을 제공하는 표준화된 컴퓨팅 하드웨어로 대체될 수 있다. 에지 컴퓨팅 네트워크들 내에서는, 컴퓨팅 리소스가 데이터로 "이동될(moved)" 서비스들에서의 시나리오들뿐만 아니라, 데이터가 컴퓨팅 리소스로 "이동될(moved)" 시나리오들이 있을 수 있다. 또는 예로서, 기지국 컴퓨팅, 가속 및 네트워크 리소스들은 코너 경우(corner case)들, 긴급 상황들을 관리하기 위해 또는 상당히 더 길게 구현된 라이프사이클에 걸쳐 배치된 리소스들에 대한 오랜 수명(longevity)을 제공하기 위해 휴면 중인 용량(가입, 요구에 따른 용량(capacity on demand))을 활성화함으로써 필요에 따른 기준으로 작업부하 요구들에 스케일링하기 위해 서비스들을 제공할 수 있다.

[0027] 도 2는 엔드포인트들, 에지 클라우드, 및 클라우드 컴퓨팅 환경들 사이의 동작 계층들을 도시한다. 구체적으로, 도 2는 네트워크 컴퓨팅의 다중의 예시적인 계층 중에서 에지 클라우드(110)를 활용하는 계산 사용 사례들(205)의 예들을 묘사한다. 계층들은 데이터 생성, 분석, 및 데이터 소비 활동들을 수행하기 위해 에지 클라우드(110)에 액세스하는 엔드포인트(디바이스들 및 사물들) 계층(200)에서 시작한다. 에지 클라우드(110)는 게이트웨이를 갖는 에지 디바이스 계층(210), 구내(on-premise) 서버, 또는 물리적으로 근접한 에지 시스템

들에 위치되는 네트워크 장비(노드(215)); 기지국들, 무선 처리 유닛들, 네트워크 허브들, 지역 데이터 센터들(DC), 또는 로컬 네트워크 장비(장비(225))를 포함하는 네트워크 액세스 계층(220); 및 그 사이에 위치된 임의의 장비, 디바이스들, 또는 노드들(계층(212)에서, 상세히 예시되지 않음)과 같은 다중의 네트워크 계층에 걸쳐 있을 수 있다. 에지 클라우드(110) 내에서의 그리고 다양한 계층들 중에서의 네트워크 통신들은 묘사되지 않은 커넥티비티(connectivity) 아키텍처들 및 기술들을 통해 발생하는 것을 포함하여, 임의의 수의 유선 또는 무선 매체들을 통해 발생할 수 있다.

[0028] 네트워크 통신 거리 및 처리 시간 제약들에서 비롯되는 레이턴시의 예들은 엔드포인트 계층(200) 중에 있을 때의 밀리초(ms) 미만으로부터, 에지 디바이스들 계층(210)에서의 5ms 아래, 심지어 네트워크 액세스 계층(220)에서의 노드들과 통신할 때의 10 내지 40ms까지의 범위일 수 있다. 에지 클라우드(110) 너머에는 코어 네트워크(230) 및 클라우드 데이터 센터(240) 계층들이 있고, 각각은 증가하는 레이턴시를 갖는다(예를 들어, 코어 네트워크 계층(230)에서의 50-60ms 사이에서부터 클라우드 데이터 센터 계층에서의 100ms 이상). 그 결과, 적어도 50 내지 100ms 이상의 레이턴시들을 갖는, 코어 네트워크 데이터 센터(235) 또는 클라우드 데이터 센터(245)에서의 동작들은 사용 사례들(205)의 대다수의 시간 임계적 함수들을 달성하지 못할 것이다. 이들 레이턴시 값들 각각은 예시 및 대비 목적을 위해 제공된다; 다른 액세스 네트워크 매체들 및 기술들의 사용이 레이턴시들을 추가로 감소시킬 수 있다는 것이 이해될 것이다. 일부 예들에서, 네트워크의 각자의 부분들은, 네트워크 소스 및 목적지에 대해, "근접 에지(close edge)", "로컬 에지", "근거리 에지(near edge)", "중간 에지", 또는 "원거리 에지" 계층들로서 카테고리화될 수 있다. 예를 들어, 코어 네트워크 데이터 센터(235) 또는 클라우드 데이터 센터(245)의 관점에서 보면, 중앙 오피스 또는 콘텐츠 데이터 네트워크는 "근거리 에지" 계층(클라우드에 "근거리", 사용 사례들(205)의 디바이스들 및 엔드포인트들과 통신할 때 높은 레이턴시 값들을 가짐) 내에 위치되는 것으로 간주될 수 있는 반면, 액세스 포인트, 기지국, 구내 서버, 또는 네트워크 게이트웨이는 "원거리 에지" 계층(클라우드로부터 "원거리", 사용 사례들(205)의 디바이스들 및 엔드포인트들과 통신할 때 저 레이턴시 값들을 가짐) 내에 위치되는 것으로 간주될 수 있다. "가까운", "로컬", "근거리", "중간", 또는 "원거리" 에지를 구성하는 것으로서의 특정 네트워크 계층의 다른 카테고리화들은 네트워크 계층들(200-240) 중 임의의 것에서의 소스로부터 측정되는, 레이턴시, 거리, 네트워크 홉들의 수, 또는 다른 측정가능한 특성들에 기초할 수 있다는 것이 이해될 것이다.

[0029] 다양한 사용 사례들(205)은 에지 클라우드를 활용하는 다중의 서비스로 인해, 인커밍 스트림들로부터의 사용 압박 하에서 리소스들에 액세스할 수 있다. 저 레이턴시를 갖는 결과들을 달성하기 위해, 에지 클라우드(110) 내에서 실행되는 서비스들은 (a) 우선순위(스루풋 또는 레이턴시) 및 서비스 품질(QoS)(예를 들어, 자율 주행차에 대한 트래픽은 응답 시간 요건의 관점에서 온도 센서보다 더 높은 우선순위를 가질 수 있음; 또는, 응용에 좌우되어, 컴퓨팅/가속기, 메모리, 스토리지, 또는 네트워크 리소스에 성능 민감도/병목이 존재할 수 있음); (b) 신뢰성 및 복원력(예를 들어, 일부 입력 스트림들은 액션을 받을 필요가 있고 트래픽은 미션 크리티컬 신뢰성으로 라우팅될 필요가 있는 한편, 일부 다른 입력 스트림들은 응용에 좌우되어 가끔의 실패를 허용할 수 있음); 및 (c) 물리적 제약들(예를 들어, 전력, 냉각 및 폼 팩터)의 면에서 변하는 요건들을 밸런싱한다.

[0030] 이러한 사용 사례들에 대한 엔드-투-엔드 서비스 뷰는 서비스 흐름의 개념을 수반하고 트랜잭션과 연관된다. 트랜잭션은 서비스를 소비하는 엔티티에 대한 전체 서비스 요구뿐만 아니라, 리소스들, 작업부하들, 작업 흐름들, 및 비즈니스 함수 및 비즈니스 레벨 요건들에 대한 연관된 서비스들을 상세화한다. 설명된 "조건(terms)"으로 실행되는 서비스들은 서비스의 라이프사이클 동안 트랜잭션에 대한 실시간 및 런타임 계약 준수를 보장하는 방식으로 각각의 계층에서 관리될 수 있다. 트랜잭션 내의 컴포넌트가 SLA에 합의된 것을 놓치고 있을 때, 시스템 전체(트랜잭션 내의 컴포넌트들)는 (1) SLA 위반의 영향을 이해하고, (2) 전체 트랜잭션 SLA를 재개하기 위해 시스템 내의 다른 컴포넌트들을 증강하고, (3) 교정하는 조치들을 구현하는 능력을 제공할 수 있다.

[0031] 따라서, 이들 변화 및 서비스 특징을 염두에 두고서, 에지 클라우드(110) 내의 에지 컴퓨팅은 사용 사례들(205)의 다중의 애플리케이션(예를 들어, 물체 추적, 비디오 감시, 커넥티드 카(connected car)들 등)을 실시간으로 또는 거의 실시간으로 서빙하고 그에 응답하는 능력을 제공하고, 이들 다중의 애플리케이션에 대한 초저 레이턴시 요건들을 충족시킬 수 있다. 이러한 이점들은, 레이턴시 또는 다른 제한들로 인해 클라우드 컴퓨팅을 활용할 수 없는, 완전히 새로운 클래스의 애플리케이션들(VNF들(Virtual Network Functions), FaaS(Function as a Service), EaaS(Edge as a Service), 표준 프로세스들 등)을 가능하게 한다.

[0032] 그러나, 에지 컴퓨팅의 이점들과 함께 다음의 주의사항들(caveats)이 따라온다. 에지에 위치된 디바이스들은 종종 리소스 제약되고 따라서 에지 리소스들의 사용에 대한 압박이 있다. 전형적으로, 이는 다중의 사용자(테넌트들) 및 디바이스들에 의한 사용을 위한 메모리 및 스토리지 리소스들의 풀링(pooling)을 통해 해결된다.

에지는 전력 및 냉각 제약될 수 있고 따라서 전력 사용은 가장 많은 전력을 소비하고 있는 애플리케이션들에 의해 고려될 필요가 있다. 이들 풀링된 메모리 리소스에는 내재된 전력 성능 절충들이 있을 수 있는데, 그 이유는 이들 중 다수가 더 많은 전력이 더 큰 메모리 대역폭을 요구하는 신생 메모리 기술들을 사용할 가능성이 있기 때문이다. 마찬가지로, 에지 로케이션들이 무인(unmanned)일 수 있고 심지어 허가된 액세스를 필요로 할 수 있기 때문에(예를 들어, 제3자 로케이션에 하우징될 때), 하드웨어의 개선된 보안과 신뢰 루트의 신뢰된 함수들(root of trust trusted functions)이 또한 요구된다. 이러한 이슈들은 멀티 테넌트, 멀티 오너, 또는 멀티 액세스 설정에서의 에지 클라우드(110)에서 심화되고, 여기서 서비스들 및 애플리케이션들은, 특히 네트워크 사용이 동적으로 요동하고 다중의 이해관계자, 사용 사례들, 및 서비스들의 구성(composition)이 변화함에 따라, 많은 사용자들에 의해 요청된다.

[0033] 보다 일반적인 레벨에서, 에지 컴퓨팅 시스템은, 클라이언트 및 분산 컴퓨팅 디바이스들로부터 조정을 제공하는, 에지 클라우드(110)에서 동작하는 이전에 논의된 계층들(네트워크 계층들(200-240))에서의 임의의 수의 배치들을 포괄하는 것으로 설명될 수 있다. 하나 이상의 에지 게이트웨이 노드, 하나 이상의 에지 애그리게이션 노드(edge aggregation node), 및 하나 이상의 코어 데이터 센터가 네트워크의 계층들에 걸쳐 분산되어 전기통신 서비스 제공자(telecommunication service provider)("telco", 또는 "TSP"), 사물 인터넷 서비스 제공자, 클라우드 서비스 제공자(cloud service provider, CSP), 엔터프라이즈 엔티티, 또는 임의의 다른 수의 엔티티들에 의해 또는 그를 대신하여 에지 컴퓨팅 시스템의 구현을 제공할 수 있다. 에지 컴퓨팅 시스템의 다양한 구현들 및 구성들은, 예컨대 서비스 목표들을 충족시키도록 오케스트레이션될 때, 동적으로 제공될 수 있다.

[0034] 본 명세서에 제공된 예들과 일관되게, 클라이언트 컴퓨팅 노드는 데이터의 생산자 또는 소비자로서 통신할 수 있는 임의 타입의 엔드포인트 컴포넌트, 디바이스, 기기, 또는 다른 것으로서 구체화될 수 있다. 또한, 에지 컴퓨팅 시스템에서 사용되는 라벨 "노드" 또는 "디바이스"는 그러한 노드 또는 디바이스가 클라이언트 또는 에이전트/미니온/팔로워 역할에서 동작하는 것을 반드시 의미하지는 않는다; 오히려, 에지 컴퓨팅 시스템 내의 노드들 또는 디바이스들 중 임의의 것은 에지 클라우드(110)를 용이하게 하거나 사용하기 위한 이산 또는 접속된 하드웨어 또는 소프트웨어 구성들을 포함하는 개별 엔티티들, 노드들, 또는 서브시스템들을 지칭한다.

[0035] 이와 같이, 에지 클라우드(110)는 네트워크 계층들(210-230) 중에서 에지 게이트웨이 노드들, 에지 애그리게이션 노드들, 또는 다른 에지 컴퓨팅 노드들에 의해 그리고 그 내에서 동작되는 네트워크 컴포넌트들 및 함수 피처들로부터 형성된다. 따라서 에지 클라우드(110)는, 본 명세서에서 논의되는, 라디오 액세스 네트워크(radio access network, RAN) 가능 엔드포인트 디바이스들(예를 들어, 모바일 컴퓨팅 디바이스들, IoT 디바이스들, 스마트 디바이스들 등)에 근접하게 위치되는 에지 컴퓨팅 및/또는 스토리지 리소스들을 제공하는 임의 타입의 네트워크로서 구체화될 수 있다. 다시 말해서, 에지 클라우드(110)는, 모바일 캐리어 네트워크들(예를 들어, GSM(Global System for Mobile Communications) 네트워크들, LTE(Long-Term Evolution) 네트워크들, 5G/6G 네트워크들 등)을 포함하는, 서비스 제공자 코어 네트워크들 내로의 입구 포인트의 역할을 하는 전통적인 네트워크 액세스 포인트들과 엔드포인트 디바이스들을 접속하면서, 또한 스토리지 및/또는 컴퓨팅 능력들을 제공하는 "에지"로서 구상될 수 있다. 다른 타입들 및 형식들의 네트워크 액세스(예를 들어, Wi-Fi, 장거리 무선, 광학 네트워크들을 포함하는 유선 네트워크들)가 또한 그러한 3GPP 캐리어 네트워크들 대신에 또는 그와 조합되어 활용될 수 있다.

[0036] 에지 클라우드(110)의 네트워크 컴포넌트들은 서버들, 멀티 테넌트 서버들, 기기 컴퓨팅 디바이스들, 및/또는 임의의 다른 타입의 컴퓨팅 디바이스들일 수 있다. 예를 들어, 에지 클라우드(110)는 하우징, 섀시, 케이스 또는 쉘(shell)을 포함하는 자족적인 전자 디바이스인 기기 컴퓨팅 디바이스를 포함할 수 있다. 일부 상황들에서, 하우징은 휴대성을 위해 치수가 정해져서 인간에 의해 휴대되고 및/또는 선적될 수 있도록 할 수 있다. 예시적인 하우징들은 기기의 콘텐츠를 부분적으로 또는 완전히 보호하는 하나 이상의 외부 표면을 형성하는 재료들을 포함할 수 있고, 여기서 보호는 내기후성, 위험한 환경 보호(예를 들어, EMI, 진동, 극한 온도들)를 포함할 수 있고 및/또는 수중 사용(submergibility)을 가능하게 할 수 있다. 예시적인 하우징들은, AC 전력 입력들, DC 전력 입력들, AC/DC 또는 DC/AC 컨버터(들), 전력 조정기들, 변압기들, 충전 회로, 배터리들, 유선 입력들 및/또는 무선 전력 입력들과 같은, 고정식 및/또는 휴대용 구현들을 위한 전력을 제공하는 전력 회로를 포함할 수 있다. 예시적인 하우징들 및/또는 그것의 표면들은 빌딩들, 전기통신 구조물들(예를 들어, 기둥들, 안테나 구조물들 등) 및/또는 랙들(예를 들어, 서버 랙들, 블레이드 마운트들 등)과 같은 구조물들에의 부착을 가능하게 하기 위해 장착 하드웨어에 연결되거나 이를 포함할 수 있다. 예시적인 하우징들 및/또는 그것의 표면들은 하나 이상의 센서(예를 들어, 온도 센서들, 진동 센서들, 광 센서들, 음향 센서들, 용량성 센서들, 근접 센서들 등)를 지지할 수 있다. 하나 이상의 그러한 센서는 표면에 포함되거나, 표면에 의해

휴대되거나, 또는 달리 표면에 임베드되고 및/또는 기기의 표면에 장착될 수 있다. 예시적인 하우징들 및/또는 그것의 표면들은 추진 하드웨어(예를 들어, 바퀴들, 프로펠러들 등) 및/또는 관절식 하드웨어(예를 들어, 로봇 암들, 피봇 가능한 부속물들 등)와 같은 기계적 연결을 지원할 수 있다. 일부 상황들에서, 센서들은 사용자 인터페이스 하드웨어(예를 들어, 버튼, 스위치, 다이얼, 슬라이더 등)와 같은 임의 타입의 입력 디바이스들을 포함할 수 있다. 일부 상황들에서, 예시적인 하우징들은 그 안에 포함되거나, 그에 의해 휴대되거나, 그 안에 임베드되고 및/또는 그것에 부착된 출력 디바이스들을 포함한다. 출력 디바이스들은 디스플레이들, 터치스크린들, 라이트들, LED들, 스피커들, I/O 포트들(예를 들어, USB) 등을 포함할 수 있다. 일부 상황들에서, 에지 디바이스들은 특정 목적(예를 들어, 교통 신호등)을 위해 네트워크에서 제시되는 디바이스들이지만, 다른 목적들을 위해 활용될 수 있는 처리 및/또는 다른 능력들을 가질 수 있다. 그러한 에지 디바이스들은 다른 네트워킹된 디바이스들과 독립적일 수 있으며, 그의 주요 목적에 적합한 폼 팩터를 갖는 하우징을 구비할 수 있고; 그렇지만 그의 주 태스크를 방해하지 않는 다른 컴퓨팅 태스크들을 위해 이용가능할 수 있다. 에지 디바이스들은 사물 인터넷 디바이스들을 포함한다. 기기 컴퓨팅 디바이스는 디바이스 온도, 진동, 리소스 활용, 업데이트, 전력 문제, 물리적 및 네트워크 보안 등과 같은 로컬 문제들을 관리하기 위한 하드웨어 및 소프트웨어 컴포넌트들을 포함할 수 있다. 기기 컴퓨팅 디바이스를 구현하기 위한 예시적인 하드웨어가 도 6b와 관련하여 기술된다. 에지 클라우드(110)는 하나 이상의 서버 및/또는 하나 이상의 멀티 테넌트 서버를 또한 포함할 수 있다. 이러한 서버는 운영 체제를 포함하고 가상 컴퓨팅 환경을 구현할 수 있다. 가상 컴퓨팅 환경은 하나 이상의 가상 머신, 하나 이상의 컨테이너 등을 관리하는(예를 들어, 생성(spawning), 배치, 파괴하는 등) 하이퍼바이저를 포함할 수 있다. 그러한 가상 컴퓨팅 환경들은 하나 이상의 애플리케이션 및/또는 다른 소프트웨어, 코드 또는 스크립트가 하나 이상의 다른 애플리케이션, 소프트웨어, 코드 또는 스크립트로부터 격리되면서 실행될 수 있는 실행 환경을 제공한다.

[0037] 도 3에서는, 다양한 클라이언트 엔드포인트들(310)(모바일 디바이스들, 컴퓨터들, 자율 차량들, 비즈니스 컴퓨팅 장비, 산업 처리 장비의 형식임)이 엔드포인트 네트워크 애그리게이션의 타입에 특정적인 요청들 및 응답들을 교환한다. 예를 들어, 클라이언트 엔드포인트들(310)은 구내 네트워크 시스템(332)을 통해 요청들 및 응답들(322)을 교환함으로써, 유선 광대역 네트워크를 통해 네트워크 액세스를 획득할 수 있다. 모바일 컴퓨팅 디바이스들과 같은 일부 클라이언트 엔드포인트들(310)은 액세스 포인트(예를 들어, 셀룰러 네트워크 타워)(334)를 통해 요청들 및 응답들(324)을 교환함으로써, 무선 광대역 네트워크를 통해 네트워크 액세스를 획득할 수 있다. 자율 차량들과 같은 일부 클라이언트 엔드포인트들(310)은 거리 위치 네트워크 시스템(336)을 통해 무선 차량 네트워크를 통해 요청들 및 응답들(326)에 대한 네트워크 액세스를 획득할 수 있다. 그러나, 네트워크 액세스의 타입에 관계없이, TSP는 트래픽 및 요청들을 애그리게이션하기 위해 에지 클라우드(110) 내에 애그리게이션 포인트들(342, 344)을 배치할 수 있다. 따라서, 에지 클라우드(110) 내에서, TSP는 요청된 콘텐츠를 제공하기 위해, 예컨대 에지 애그리게이션 노드들(340)에서, 다양한 컴퓨팅 및 스토리지 리소스들을 배치할 수 있다. 에지 클라우드(110)의 에지 애그리게이션 노드들(340) 및 다른 시스템들은 클라우드 또는 데이터 센터(360)에 접속되고, 이것은 백홀 네트워크(350)를 사용하여 웹사이트들, 애플리케이션들, 데이터베이스 서버들 등에 대한 클라우드/데이터 센터로부터의 더 높은 레이턴시 요청들을 충족시킨다. 단일 서버 프레임워크 상에 배치된 것들을 포함하여, 에지 애그리게이션 노드들(340) 및 애그리게이션 포인트들(342, 344)의 추가적인 또는 통합된 인스턴스들이 또한 에지 클라우드(110) 또는 TSP 인프라스트럭처의 다른 영역들 내에 존재할 수 있다.

[0038] 도 4는 다중의 에지 노드 및 이러한 에지 노드들을 사용하는 다중의 테넌트(예컨대, 사용자들, 제공자들) 간에 동작되는 에지 컴퓨팅 시스템에 걸친 가상화된 및 컨테이너 기반 에지 구성들에 대한 배치 및 오케스트레이션을 도시하고 있다. 구체적으로, 도 4는 다양한 가상 에지 인스턴스들에 액세스하는, 다양한 클라이언트 엔드포인트들(410)(예를 들어, 스마트 도시들/빌딩 시스템들, 모바일 디바이스들, 컴퓨팅 디바이스들, 비즈니스/물류 시스템들, 산업 시스템들 등)에 대한 요청들 및 응답들을 충족시키기 위한, 에지 컴퓨팅 시스템(400)에서의 제1 에지 노드(422) 및 제2 에지 노드(424)의 조정을 묘사한다. 여기서, 가상 에지 인스턴스들(432, 434)은 에지 클라우드에서의 처리 및 에지 컴퓨팅 능력들을 제공하고, 웹사이트들, 애플리케이션들, 데이터베이스 서버들 등에 대한 더 높은 레이턴시 요청들을 위해 클라우드/데이터 센터(440)에 액세스한다. 그러나, 에지 클라우드는 다중의 테넌트 또는 엔티티에 대한 다중의 에지 노드 간의 처리의 조정을 가능하게 한다.

[0039] 도 4의 예에서, 이러한 가상 에지 인스턴스들은 다음을 포함한다: 에지 스토리지, 컴퓨팅, 및 서비스들의 제1 조합을 제공하는, 제1 테넌트(테넌트 1)에게 제공되는 제1 가상 에지(432); 및 에지 스토리지, 컴퓨팅 및 서비스들의 제2 조합을 제공하는 제2 가상 에지(434). 가상 에지 인스턴스들(432, 434)은 에지 노드들(422, 424) 사이에 분산되고, 요청 및 응답이 동일한 또는 상이한 에지 노드들로부터 충족되는 시나리오들을 포함할 수 있다. 분산되지만 조정된 방식으로 동작하는 에지 노드들(422, 424)의 구성은 에지 프로비저닝 함수들(450)에 기

초하여 발생한다. 다중의 테넌트 사이에서, 애플리케이션들 및 서비스들에 대한 조정된 동작을 제공하는 에지 노드들(422, 424)의 기능성은 오케스트레이션 함수들(460)에 기초하여 발생한다.

[0040] (410)에서의 디바이스들 중 일부는 테넌트 1이 테넌트1 '슬라이스' 내에서 기능할 수 있는 한편 테넌트 2가 테넌트2 슬라이스 내에서 기능할 수 있는 (그리고, 추가 예들에서, 추가적인 또는 서브 테넌트들이 존재할 수 있고; 및 각각의 테넌트는 심지어 특정 하드웨어 피처들에 대해 하루 종일 특정의 피처들의 세트에 구체적으로 자격이 주어지고 트랜잭션적으로 결부될 수 있음) 멀티 테넌트 디바이스들이라는 것을 이해해야 한다. 신뢰된 멀티 테넌트 디바이스는 키와 슬라이스의 조합이 "RoT(root of trust)" 또는 테넌트 특정적 RoT로 고려될 수 있도록 테넌트 특정적 암호 키를 추가로 포함할 수 있다. RoT는 DICE(Device Identity Composition Engine) 아키텍처를 사용하여 동적으로 구성되도록 추가로 계산될 수 있으며, 따라서 단일 DICE 하드웨어 빌딩 블록이 (FPGA(Field Programmable Gate Array)와 같은) 디바이스 능력들의 계층화를 위한 계층화된 신뢰 컴퓨팅 베이스 컨텍스트들을 구성하기 위해 사용될 수 있다. RoT는 멀티 테넌시(multi-tenancy)를 지원하기 위해 유용한 "팬 아웃(fan-out)"을 가능하게 하기 위해 신뢰 컴퓨팅 컨텍스트를 위해 추가로 사용될 수 있다. 멀티 테넌트 환경 내에서, 각자의 에지 노드들(422, 424)은 노드당 다중의 테넌트에 할당된 로컬 리소스들에 대한 보안 피처 시행 포인트들로서 동작할 수 있다. 추가적으로, 테넌트 런타임 및 애플리케이션 실행(예를 들어, 인스턴스들(432, 434)에서)은 잠재적으로 다중의 물리적 호스팅 플랫폼에 걸쳐 있는 리소스들의 가상 에지 추상화를 생성하는 보안 피처에 대한 시행 포인트로서 역할을 할 수 있다. 마지막으로, 오케스트레이션 엔티티에서의 오케스트레이션 함수들(460)은 테넌트 경계들을 따라 리소스들을 집결시키기 위한 보안 피처 시행 포인트로서 동작할 수 있다.

[0041] 에지 컴퓨팅 노드들은 리소스들(메모리, CPU(central processing unit), GPU(graphics processing unit), 인터럽트 컨트롤러, 입력/출력 I/O 컨트롤러, 메모리 컨트롤러, 버스 컨트롤러 등)을 파티셔닝할 수 있고, 여기서 각자의 파티셔닝은 RoT 능력을 포함할 수 있고, 여기서 DICE 모델에 따른 팬 아웃 및 계층화가 에지 노드들에 추가로 적용될 수 있다. 클라우드 컴퓨팅 노드들은 종종 컨테이너들, FaaS 엔진들, 서블릿들, 서버들, 또는 각각에 대한 RoT 컨텍스트를 지원하기 위해 DICE 계층화 및 팬 아웃 구조에 따라 파티셔닝될 수 있는 다른 계산 추상화를 사용한다. 따라서, 디바이스들(410, 422, 및 440)에 걸쳐 있는 각자의 RoT들은 DTCB(distributed trusted computing base)의 확립을 조정할 수 있어서, 모든 요소들을 엔드 투 엔드(end to end)로 링크하는 테넌트 특정적 가상 신뢰 보안 채널이 확립될 수 있도록 한다.

[0042] 또한, 컨테이너는 이전의 에지 노드로부터 그것의 콘텐츠를 보호하는 데이터 또는 작업부하 특정적 키들을 가질 수 있다는 것이 이해될 것이다. 컨테이너의 마이그레이션의 일환으로서, 소스 에지 노드에서의 포드 컨트롤러(pod controller)는 타겟 에지 노드 포드 컨트롤러로부터 마이그레이션 키를 획득할 수 있고, 여기서 마이그레이션 키는 컨테이너 특정적 키들을 랩핑(wrap)하기 위해 사용된다. 컨테이너/포드가 타겟 에지 노드로 마이그레이션될 때, 언랩핑 키는 포드 컨트롤러에 노출되고 포드 컨트롤러는 그 후 랩핑된 키들을 암호 해제한다. 키들은 이제 컨테이너 특정적 데이터에 대한 동작들을 수행하기 위해 사용될 수 있다. 마이그레이션 함수들은 적절히 증명된 에지 노드들 및 포드 관리자들에 의해 게이팅될 수 있다(위에 설명된 바와 같이).

[0043] 추가 예들에서, 에지 컴퓨팅 시스템은 다중 소유자, 멀티 테넌트 환경에서 컨테이너들(코드 및 필요한 의존성들을 제공하는 컨테이닝된, 배치가능한 소프트웨어의 유닛)의 사용을 통해 다중의 애플리케이션의 오케스트레이션을 제공하도록 확장된다. 도 4에서 신뢰 '슬라이스' 개념의 프로비저닝 및 라이프사이클과 관련된 키 관리, 신뢰 앵커 관리, 및 다른 보안 함수들을 수행하기 위해 멀티 테넌트 오케스트레이터가 사용될 수 있다. 예를 들어, 에지 컴퓨팅 시스템은 다중의 가상 에지 인스턴스로부터(그리고, 클라우드 또는 원격 데이터 센터로부터) 다양한 클라이언트 엔드포인트들에 대한 요청들 및 응답들을 충족시키도록 구성될 수 있다. 이들 가상 에지 인스턴스들의 사용은 다중의 테넌트 및 다중의 애플리케이션(예를 들어, AR(augmented reality)/VR(virtual reality), 기업 애플리케이션들, 콘텐츠 전달(content delivery), 게이밍, 컴퓨팅 오프로드)을 동시에 지원할 수 있다. 또한, 가상 에지 인스턴스들 내에 다중 타입의 애플리케이션들(예를 들어, 정상 애플리케이션들; 레이턴시 민감 애플리케이션들; 레이턴시 중요 애플리케이션들; 사용자 평면 애플리케이션들; 네트워킹 애플리케이션들; 등)이 있을 수 있다. 가상 에지 인스턴스들은 또한 상이한 지리적 로케이션들에 있는 다중의 소유자의 시스템들(또는, 다중의 소유자에 의해 공동 소유되거나 공동 관리되는 각자의 컴퓨팅 시스템들 및 리소스들)에 걸쳐 있을 수 있다.

[0044] 예를 들어, 각각의 에지 노드(422, 424)는 하나 이상의 컨테이너의 그룹을 제공하는 컨테이너 "포드"(426, 428)의 사용에 의한 것과 같은 컨테이너들의 사용을 구현할 수 있다. 하나 이상의 컨테이너 포드를 사용하는 설정에서, 포드 컨트롤러 또는 오케스트레이터가 포드 내의 컨테이너들의 로컬 제어 및 오케스트레이션을 담당한

다. 각자의 에지 슬라이스들(432, 434)에 대해 제공되는 다양한 에지 노드 리소스들(예를 들어, 6각형으로 묘사된, 스토리지, 컴퓨팅, 서비스들)은 각각의 컨테이너의 필요에 따라 파티셔닝된다.

[0045] 컨테이너 포드들의 사용으로, 포드 컨트롤러가 컨테이너들 및 리소스들의 파티셔닝 및 할당을 감독한다. 포드 컨트롤러는, 예컨대 SLA 계약들에 기초하여 KPI(key performance indicator) 타겟들을 수신함으로써, 물리 리소스들을 어떻게 최상으로 파티셔닝할지에 대해 그리고 어떤 지속기간 동안 컨트롤러에게 지시하는 지시들을 오케스트레이터(예를 들어, 오케스트레이터(460))로부터 수신한다. 포드 컨트롤러는 작업부하를 완료하고 SLA를 만족시키기 위해 어느 컨테이너가 어느 리소스들을 그리고 얼마나 오랫동안 필요로 하는지를 결정한다. 포드 컨트롤러는 또한: 컨테이너를 생성하는 것, 그것에 리소스들 및 애플리케이션들을 프로비저닝하는 것, 분산된 애플리케이션 상에서 함께 작업하는 다중의 컨테이너 사이의 중간 결과들을 조정하는 것, 작업부하가 완료될 때 컨테이너들을 해체하는 것, 및 그와 유사한 것과 같은 컨테이너 라이프사이클 동작들을 관리한다. 추가적으로, 포드 컨트롤러는 올바른 테넌트가 인증할 때까지 리소스들의 할당을 방지하거나 또는 증명 결과가 만족될 때까지 컨테이너에 데이터 또는 작업부하를 프로비저닝하는 것을 방지하는 보안 역할을 서빙할 수 있다.

[0046] 또한, 컨테이너 포드들의 사용으로, 테넌트 경계들이 여전히 그러나 컨테이너들의 각각의 포드의 컨텍스트에서 존재할 수 있다. 각각의 테넌트 특정적 포드가 테넌트 특정적 포드 컨트롤러를 갖는 경우, 전형적인 리소스 고갈 상황들을 회피하기 위해 리소스 할당 요청들을 통합하는 공유 포드 컨트롤러가 있을 것이다. 포드 및 포드 컨트롤러의 증명 및 신뢰성을 보장하기 위해 추가 제어들이 제공될 수 있다. 예를 들어, 오케스트레이터(460)는 증명 검증을 수행하는 로컬 포드 컨트롤러들에게 증명 검증 정책을 프로비저닝할 수 있다. 증명이 제2 테넌트 포드 컨트롤러가 아니라 제1 테넌트 포드 컨트롤러에 대한 정책을 만족시키는 경우, 제2 포드는 그것을 만족시키는 상이한 에지 노드로 마이그레이션될 수 있다. 대안적으로, 제1 포드는 실행하도록 허용될 수 있고 상이한 공유 포드 컨트롤러가 제2 포드가 실행하기 전에 설치되고 기동된다.

[0047] 도 5는 에지 컴퓨팅 시스템에서 컨테이너들을 배치하는 추가적인 컴퓨팅 배열들을 도시한다. 단순화된 예로서, 시스템 배열들(510, 520)은 포드 컨트롤러(예를 들어, 컨테이너 관리자들(511, 521), 및 컨테이너 오케스트레이터(531))가 컴퓨팅 노드들(배열(510) 내의 (515))을 통한 실행을 통해 컨테이너화된 포드들, 함수들, 및 FaaS(functions-as-a-service) 인스턴스들을 론칭하도록, 또는 컴퓨팅 노드들(배열(520) 내의 (523))을 통한 실행을 통해 컨테이너화된 가상화된 네트워크 함수들을 개별적으로 실행하도록 적응되는 설정들을 묘사한다. 이 배열은 (컴퓨팅 노드들(537)을 사용하는) 시스템 배열(530)에서 다중의 테넌트의 사용을 위해 적응되는데, 여기서 컨테이너화된 포드들(예를 들어, 포드들(512)), 함수들(예를 들어, 함수들(513), VNF들(522, 536)), 및 FaaS(functions-as-a-service) 인스턴스들(예를 들어, FaaS 인스턴스(514))은 각자의 테넌트들에 특정적인 가상 머신들(예를 들어, 테넌트들(532, 533)에 대한 VM들(534, 535)) 내에서 론칭된다(가상화된 네트워크 함수들의 실행 이외에). 이 배열은 컨테이너들(542, 543)을 제공하는 시스템 배열(540)에서의 사용, 또는 컨테이너 기반 오케스트레이션 시스템(541)에 의해 조정되는 바와 같은, 컴퓨팅 노드들(544) 상에서의 다양한 함수들, 애플리케이션들, 및 함수들의 실행을 위해 추가로 적응된다.

[0048] 도 5에 묘사된 시스템 배열들은 애플리케이션 구성(application composition)의 면에서 VM들, 컨테이너들, 및 함수들을 동등하게 취급하는 아키텍처를 제공할 수 있다(그리고 결과적인 애플리케이션들은 이들 3개의 구성 요소의 조합들이다). 각각의 구성 요소는 로컬 백엔드로서 하나 이상의 가속기(FPGA, ASIC) 컴포넌트의 사용을 수반할 수 있다. 이러한 방식으로, 애플리케이션들은, 오케스트레이터에 의해 조정된, 다중의 에지 소유자에 걸쳐 스플릿될 수 있다.

[0049] 도 5의 컨텍스트에서, 포드 컨트롤러/컨테이너 관리자, 컨테이너 오케스트레이터, 및 개별 노드들은 보안 시행 포인트를 제공할 수 있다. 그러나, 어느 한 테넌트에 할당된 리소스들이 제2 테넌트에 할당된 리소스들과 구별되는 테넌트 격리가 오케스트레이션될 수 있지만, 에지 소유자들은 리소스 할당들이 테넌트 경계들에 걸쳐 공유되지 않도록 보장하기 위해 협력한다. 또는, 테넌트 경계들에 걸쳐 리소스 할당들이 격리될 수 있는데, 이는 테넌트들이 가입 또는 트랜잭션/계약 기준을 통해 "사용"을 허용할 수 있기 때문이다. 이들 컨텍스트에서, 테넌시(tenancy)를 시행하기 위해 에지 소유자들에 의해 가상화, 컨테이너화, 인클레이브들(enclaves) 및 하드웨어 파티셔닝 스킴들이 사용될 수 있다. 다른 격리 환경들은: 베어 메탈(bare metal)(전용) 장비, 가상 머신들, 컨테이너들, 컨테이너들 상의 가상 머신들, 또는 이들의 조합들을 포함할 수 있다.

[0050] 추가 예들에서, 소프트웨어 정의된 또는 제어된 실리콘 하드웨어, 및 다른 구성 가능 하드웨어의 양태들이 에지 컴퓨팅 시스템의 애플리케이션들, 함수들, 및 서비스들과 통합될 수 있다. SDSi(software defined silicon)가 (예를 들어, 하드웨어 구성 자체 내에서의 새로운 피처들의 업그레이드, 재구성, 또는 프로비저닝에 의해) 자체

의 일부분 또는 작업부하를 교정하는 구성 요소의 능력에 기초하여, 일부 리소스 또는 하드웨어 구성 요소가 계약 또는 서비스 레벨 협약을 충족하는 능력을 보장하기 위해 사용될 수 있다.

[0051] 추가 예들에서, 본 에지 컴퓨팅 시스템들 및 환경을 참조하여 논의된 컴퓨팅 노드들 또는 디바이스들 중 임의의 것이 도 6a 및 도 6b에 묘사된 컴포넌트들에 기초하여 이행될 수 있다. 각자의 에지 컴퓨팅 노드들은 다른 에지, 네트워킹, 또는 엔드포인트 컴포넌트들과 통신할 수 있는 어느 한 타입의 디바이스, 기기, 컴퓨터, 또는 다른 "사물(thing)"로서 구체화될 수 있다. 예를 들어, 에지 컴퓨팅 디바이스는 개인 컴퓨터, 서버, 스마트폰, 모바일 컴퓨팅 디바이스, 스마트 기기, 차량내 컴퓨팅 시스템(예를 들어, 내비게이션 시스템), 외부 케이스, 쉘 등을 갖는 자족적인(self-contained) 디바이스, 또는 설명된 함수들을 수행 가능한 다른 디바이스 또는 시스템으로서 구체화될 수 있다.

[0052] 도 6a에 묘사된 단순화된 예에서, 에지 컴퓨팅 노드(600)는 컴퓨팅 엔진(본 명세서에서 "컴퓨팅 회로"로서 또한 지칭됨)(602), 입력/출력(I/O) 서브시스템(본 명세서에서 "I/O 회로"라고도 지칭됨)(608), 데이터 스토리지(본 명세서에서 "데이터 스토리지 회로"로서 또한 지칭됨)(610), 통신 회로 서브시스템(612), 및, 선택적으로 하나 이상의 주변 디바이스(본 명세서에서 "주변 디바이스 회로"로서 또한 지칭됨)(614)를 포함한다. 다른 예들에서, 각자의 컴퓨팅 디바이스들은 컴퓨터에서 전형적으로 발견되는 것들(예를 들어, 디스플레이, 주변 디바이스들 등)과 같은 다른 또는 추가적인 컴포넌트들을 포함할 수 있다. 추가적으로, 일부 예들에서, 예시적인 컴포넌트들 중 하나 이상은 또 다른 컴포넌트에 통합되거나 다른 경우에는 또 다른 컴포넌트의 일부분을 형성할 수 있다.

[0053] 컴퓨팅 노드(600)는 다양한 컴퓨팅 함수들을 수행할 수 있는 임의 타입의 엔진, 디바이스, 또는 디바이스들의 컬렉션으로서 구체화될 수 있다. 일부 예들에서, 컴퓨팅 노드(600)는 집적 회로, 임베디드 시스템, FPGA(field-programmable gate array), SOC(system-on-a-chip), 또는 다른 통합 시스템 또는 디바이스와 같은 단일 디바이스로서 구체화될 수 있다. 예시적인 예에서, 컴퓨팅 노드(600)는 프로세서(본 명세서에서 "프로세서 회로"로서 또한 지칭됨)(604) 및 메모리(본 명세서에서 "메모리 회로"로서 또한 지칭됨)(606)를 포함하거나 이로서 구체화된다. 프로세서(604)는 본 명세서에 설명된 함수들을 수행할 수 있는(예를 들어, 애플리케이션을 실행할 수 있는) 임의 타입의 프로세서(들)로서 구체화될 수 있다. 예를 들어, 프로세서(604)는 멀티 코어 프로세서(들), 마이크로컨트롤러, 처리 유닛, 특수화된 또는 특수 목적 처리 유닛, 또는 다른 프로세서 또는 처리/제어 회로로서 구체화될 수 있다.

[0054] 일부 예들에서, 프로세서(604)는 FPGA, ASIC(application specific integrated circuit), 재구성 가능 하드웨어 또는 하드웨어 회로, 또는 본 명세서에서 설명된 함수들의 수행을 용이하게 하기 위한 다른 특수화된 하드웨어로서 구체화되거나, 이들을 포함하거나, 이들에 결합될 수 있다. 또한 일부 예들에서, 프로세서(604)는 DPU(data processing unit), IPU(infrastructure processing unit), 또는 NPU(network processing unit)라고도 알려진 특수화된 x-처리 유닛(x-PU)으로서 구체화될 수 있다. 이러한 xPU는 SOC 내에 통합되거나, 네트워킹 회로(예를 들어, SmartNIC, 또는 개선된 SmartNIC), 가속 회로, 스토리지 디바이스들, 스토리지 디스크들, 또는 AI 하드웨어(예를 들어, GPU들 또는 프로그래밍된 FPGA들)와 통합되는 독립형 회로 또는 회로 패키지로서 구체화될 수 있다. 그러한 x-PU는 하나 이상의 데이터 스트림을 처리하기 위해 프로그래밍을 수신하고, 검색하고 및/또는 다른 경우에는 획득하고 및 CPU 또는 범용 처리 하드웨어 밖에서 (마이크로서비스들을 호스팅하는 것, 서비스 관리 또는 오케스트레이션을 수행하는 것, 서버 또는 데이터 센터 하드웨어를 조직 또는 관리하는 것, 서비스 메시들을 관리하는 것, 또는 텔레메트리를 수집 및 분배하는 것과 같은) 데이터 스트림들에 대한 특정 태스크들 및 액션들을 수행하도록 설계될 수 있다. 그렇지만, x-PU, SOC, CPU, 및 프로세서(604)의 다른 변형들이 컴퓨팅 노드(600) 내에서 그리고 그를 대신하여 많은 타입의 동작들 및 명령어들을 실행하기 위해 서로 협력하여 동작할 수 있다는 것을 이해할 것이다.

[0055] 메모리(606)는 본 명세서에서 설명된 함수들을 수행할 수 있는 임의 타입의 휘발성(예를 들어, DRAM(dynamic random access memory) 등) 또는 비휘발성 메모리 또는 데이터 스토리지로서 구체화될 수 있다. 휘발성 메모리(volatile memory)는 매체에 의해 저장된 데이터의 상태를 유지하기 위해 전력을 요구하는 스토리지 매체일 수 있다. 휘발성 메모리의 비제한적인 예들은 다양한 타입의 RAM(random access memory), 예컨대 DRAM 또는 SRAM(static random access memory)을 포함할 수 있다. 메모리 모듈에서 사용될 수 있는 하나의 특정 타입의 DRAM은 SDRAM(Synchronous Dynamic Random Access Memory)이다.

[0056] 예에서, 메모리 디바이스(예를 들어, 메모리 회로)는 NAND 또는 NOR 기술들(예를 들어, SLC(Single-Level Cell), MLC(Multi-Level Cell), QLC(Quad-Level Cell), TLC(Tri-Level Cell), 또는 일부 다른 NAND)에 기초하

는 것들과 같은, 임의의 수의 블록 어드레싱가능 메모리 디바이스들이다. 일부 예들에서, 메모리 디바이스(들)는 바이트 어드레싱가능 라이트-인-플레이스(write-in-place) 3차원 크로스포인트 메모리 디바이스, 또는 다른 바이트 어드레싱가능 라이트-인-플레이스 NVM(non-volatile memory) 디바이스들, 예컨대 단일 또는 멀티-레벨 PCM(Phase Change Memory) 또는 PCMS(phase change memory with a switch), 칼코게나이드 상 변화 재료(예를 들어, 칼코게나이드 유리)를 사용하는 NVM 디바이스들, 금속 산화물 베이스(metal oxide base), 산소 베이컨시 베이스(oxygen vacancy base) 및 CB-RAM(Conductive Bridge Random Access Memory)을 포함하는 저항성 메모리, 나노와이어 메모리, FeTRAM(ferroelectric transistor random access memory), 멤리스터 기술을 포함하는 MRAM(magneto resistive random access memory), STT(spin transfer torque)-MRAM, 스핀트로닉 자기 접합 메모리 기반 디바이스, MTJ(magnetic tunneling junction) 기반 디바이스, DW(Domain Wall) 및 SOT(Spin Orbit Transfer) 기반 디바이스, 사이리스터 기반 메모리 디바이스, 상기 중 어느 것의 조합, 또는 다른 적합한 메모리를 포함한다. 메모리 디바이스는 3차원 크로스포인트 메모리 디바이스(예를 들어, Intel® 3D XPoint™ 메모리), 또는 다른 바이트 어드레싱가능 라이트-인 플레이스 비휘발성 메모리 디바이스들을 또한 포함할 수 있다. 메모리 디바이스는 다이 자체 및/또는 패키징된 메모리 제품을 지칭할 수 있다. 일부 예들에서, 3D 크로스포인트 메모리(예를 들어, Intel® 3D XPoint™ 메모리)는 메모리 셀들이 워드 라인들과 비트 라인들의 교차점에 놓이고 개별적으로 어드레싱가능하며 비트 스토리지가 벌크 저항(bulk resistance)의 변화에 기초하는 무트랜지스터 적층가능 크로스 포인트 아키텍처(transistor-less stackable cross point architecture)를 포함할 수 있다. 일부 예들에서, 메모리(606)의 전부 또는 일부는 프로세서(604)에 통합될 수 있다. 메모리(606)는 하나 이상의 애플리케이션, 애플리케이션(들)에 의해 조작되는 데이터, 라이브러리들, 및 드라이버들과 같은 동작 동안 사용되는 다양한 소프트웨어 및 데이터를 저장할 수 있다.

[0057] 일부 예들에서, 저항기 기반 및/또는 무트랜지스터 메모리 아키텍처들이 상변화 재료의 볼륨이 적어도 2개의 전극 사이에 존재하는 나노미터 스케일 PCM(phase-change memory) 디바이스들을 포함한다. 예시적인 상변화 재료의 부분들은 결정질 상들 및 비정질 상들의 변화하는 정도들을 드러내는데, 여기서 적어도 2개의 전극 사이의 저항의 변화하는 정도들이 측정될 수 있다. 일부 예들에서, 상변화 재료는 칼코게나이드 기반 유리 재료이다. 그러한 저항성 메모리 디바이스들은 때때로 이전에 그들을 통해 흐른 전류의 이력을 기억하는 멤리스티브 디바이스(memristive device)들로서 지칭된다. 저장된 데이터는 전기 저항을 측정함으로써 예시적인 PCM 디바이스들로부터 검색되며, 여기서 결정질 상들은 비교적 높은 저항 값(들)(예로서, 논리 "1")을 갖는 비정질 상들과 비교할 때 비교적 낮은 저항 값(들)(예로서, 논리 "0")을 드러낸다.

[0058] 예시적인 PCM 디바이스들은 장기간(예를 들어, 실온에서 대략 10년) 동안 데이터를 저장한다. 예시적인 PCM 디바이스들에 대한 기입 동작들(예를 들어, 논리 "0"에 설정, 논리 "1"에 설정, 중간 저항 값에 설정)은 적어도 2개의 전극에 하나 이상의 전류 펄스를 인가함으로써 달성되며, 여기서 펄스들은 특정 전류 크기 및 지속 기간을 갖는다. 예를 들어, 적어도 2개의 전극에 인가된 긴 저전류 펄스(SET)는 예시적인 PCM 디바이스로 하여금 저저항 결정질 상태에 있도록 야기하는 한편, 적어도 2개의 전극에 인가된 비교적 짧은 고전류 펄스(RESET)는 예시적인 PCM 디바이스가 고저항 비정질 상태에 있도록 야기한다.

[0059] 일부 예들에서, PCM 디바이스들의 구현은 인메모리(in-memory) 컴퓨팅 능력들을 가능하게 하는 비-폰 노이만 컴퓨팅 아키텍처들을 용이하게 한다. 일반적으로 말하면, 전통적인 컴퓨팅 아키텍처들은 버스를 통해 하나 이상의 메모리 디바이스에 통신가능하게 접속된 CPU(central processing unit)를 포함한다. 이와 같이, CPU와 메모리 사이에서 데이터를 전송하기 위해 유한한 양의 에너지 및 시간이 소비되는데, 이는 폰 노이만 컴퓨팅 아키텍처의 알려진 병목 현상이다. 그러나, PCM 디바이스들은 인메모리로 일부 컴퓨팅 동작들을 수행함으로써 CPU와 메모리 사이의 데이터 전송들을 최소화하고, 일부 경우들에서는 제거한다. 달리 말하면, PCM 디바이스들은 정보를 저장하고 계산 태스크들을 실행한다. 이러한 비-폰 노이만 컴퓨팅 아키텍처들은 10,000 비트를 갖는 벡터들과 같이, 초차원 컴퓨팅을 용이하게 하기 위해 비교적 높은 차원수를 갖는 벡터들을 구현할 수 있다. 비교적 큰 비트 폭 벡터들은 넓은 비트 벡터들과 유사한 정보를 또한 처리하는, 인간의 두뇌를 모방해 모델링된 컴퓨팅 패러다임들을 가능하게 한다.

[0060] 컴퓨팅 회로(602)는 컴퓨팅 회로(602)(예를 들어, 프로세서(604) 및/또는 메인 메모리(606)를 가짐) 및 컴퓨팅 회로(602)의 다른 컴포넌트들과의 입력/출력 동작들을 용이하게 하는 회로 및/또는 컴포넌트들로서 구체화될 수 있는 I/O 서브시스템(608)을 통해 컴퓨팅 노드(600)의 다른 컴포넌트들에 통신가능하게 결합된다. 예를 들어, I/O 서브시스템(608)은 메모리 컨트롤러 허브들, 입력/출력 제어 허브들, 통합된 센서 허브들, 펌웨어 디바이스들, 통신 링크들(예를 들어, 포인트-투-포인트 링크들, 버스 링크들, 와이어들, 케이블들, 도광체들(light guides), 인쇄 회로 보드 트레이스들 등), 및/또는 입력/출력 동작들을 용이하게 하기 위한 다른 컴포넌트들 및

서브시스템들로서 구체화되거나 그렇지 않으면 이들을 포함할 수 있다. 일부 예들에서, I/O 서브시스템(608)은 SoC(system-on-a-chip)의 일부분을 형성할 수 있고, 컴퓨팅 회로(602)의 프로세서(604), 메모리(606), 및 다른 컴포넌트들 중 하나 이상과 함께 컴퓨팅 회로(602)에 통합될 수 있다.

[0061] 하나 이상의 예시적인 데이터 스토리지 디바이스들/디스크들(610)은, 예를 들어, 메모리 디바이스들, 메모리, 회로, 메모리 카드들, 플래시 메모리, 하드 디스크 드라이브들, SSD(solid-state drive)들, 및/또는 다른 데이터 스토리지 디바이스들/디스크들과 같은, 데이터의 단기 또는 장기 저장을 위해 구성된 임의 타입(들)의 물리적 디바이스(들) 중 하나 이상으로서 구체화될 수 있다. 개별 데이터 스토리지 디바이스들/디스크들(610)은 데이터 스토리지 디바이스/디스크(610)에 대한 데이터 및 펌웨어 코드를 저장하는 시스템 파티션을 포함할 수 있다. 개별 데이터 스토리지 디바이스들/디스크들(610)은 또한, 예를 들어, 컴퓨팅 노드(600)의 타입에 좌우되어 운영 체제들에 대한 데이터 파일들 및 실행파일들을 저장하는 하나 이상의 운영 체제 파티션을 포함할 수 있다.

[0062] 통신 회로(612)는 컴퓨팅 회로(602)와 또 다른 컴퓨팅 디바이스(예를 들어, 구현 에지 컴퓨팅 시스템의 에지 게이트웨이) 사이에서 네트워크를 통한 통신을 가능하게 할 수 있는 임의의 통신 회로, 디바이스, 또는 이들의 컬렉션으로서 구체화될 수 있다. 통신 회로(612)는 이러한 통신을 달성하기 위해 임의의 하나 이상의 통신 기술(예를 들어, 유선 또는 무선 통신) 및 연관된 프로토콜들(예를 들어, 3GPP 4G 또는 5G 표준과 같은 셀룰러 네트워킹 프로토콜, IEEE 802.11/Wi-Fi®와 같은 무선 로컬 영역 네트워크 프로토콜, 무선 광역 네트워크 프로토콜, 이더넷, Bluetooth®, Bluetooth Low Energy, IEEE 802.15.4 또는 ZigBee®와 같은 IoT 프로토콜, LPWAN(low-power wide-area network) 또는 LPWA(low-power wide-area) 프로토콜 등)을 사용하도록 구성될 수 있다.

[0063] 예시적인 통신 회로(612)는 HFI(host fabric interface)로도 지칭될 수 있는 NIC(network interface controller)(620)를 포함한다. NIC(620)는 또 다른 컴퓨팅 디바이스(예를 들어, 에지 게이트웨이 노드)와 접속하기 위해 컴퓨팅 노드(600)에 의해 사용될 수 있는 하나 이상의 애드-인-보드(add-in-board), 도터 카드(daughter card), 네트워크 인터페이스 카드, 컨트롤러 칩, 칩셋, 또는 다른 디바이스들로서 구체화될 수 있다. 일부 예들에서, NIC(620)는 하나 이상의 프로세서를 포함하는 SoC(system-on-a-chip)의 일부로서 구체화되거나, 또는 하나 이상의 프로세서를 또한 포함하는 멀티칩 패키지 상에 포함될 수 있다. 일부 예들에서, NIC(620)는 둘 다 NIC(620)에 로컬인 로컬 프로세서(도시되지 않음) 및/또는 로컬 메모리(도시되지 않음)를 포함할 수 있다. 그러한 예들에서, NIC(620)의 로컬 프로세서는 본 명세서에 설명된 컴퓨팅 회로(602)의 함수들 중 하나 이상을 수행할 수 있다. 그에 부가하여 또는 대안적으로, 이러한 예들에서, NIC(620)의 로컬 메모리는 보드 레벨, 소켓 레벨, 칩 레벨, 및/또는 다른 레벨들에서 클라이언트 컴퓨팅 노드의 하나 이상의 컴포넌트에 통합될 수 있다.

[0064] 추가적으로, 일부 예들에서, 각자의 컴퓨팅 노드(600)는 하나 이상의 주변 디바이스(614)를 포함할 수 있다. 그러한 주변 디바이스들(614)은 컴퓨팅 노드(600)의 특정 타입에 좌우되어, 오디오 입력 디바이스, 디스플레이, 다른 입력/출력 디바이스, 인터페이스 디바이스, 및/또는 다른 주변 디바이스와 같은 컴퓨팅 디바이스 또는 서버에서 발견되는 임의 타입의 주변 디바이스를 포함할 수 있다. 추가 예들에서, 컴퓨팅 노드(600)는 에지 컴퓨팅 시스템 또는 유사한 형식들의 기기들, 컴퓨터들, 서브시스템들, 회로, 또는 다른 컴포넌트들에서 각자의 에지 컴퓨팅 노드(클라이언트, 게이트웨이, 또는 애그리게이션 노드이든 간에)에 의해 구체화될 수 있다.

[0065] 보다 상세한 예에서, 도 6b는 본 명세서에 설명된 기법들(예를 들어, 동작들, 프로세스들, 방법들, 및 방법론들)을 구현하기 위해 에지 컴퓨팅 노드(650)에 존재할 수 있는 컴포넌트들의 예의 블록도를 도시한다. 이 에지 컴퓨팅 노드(650)는 컴퓨팅 디바이스로서 또는 그의 일부로서(예컨대, 모바일 디바이스, 기지국, 서버, 게이트웨이 등으로서) 구현될 때 노드(600)의 각자의 컴포넌트들의 보다 가까운 뷰를 제공한다. 에지 컴퓨팅 노드(650)는 본 명세서에서 참조된 하드웨어 또는 논리 컴포넌트들의 임의의 조합을 포함할 수 있고, 에지 통신 네트워크 또는 그러한 네트워크들의 조합과 함께 사용 가능한 임의의 디바이스를 포함하거나 그와 결합될 수 있다. 컴포넌트들은 집적 회로(IC)들, 그의 부분들, 이산 전자 디바이스들, 또는 다른 모듈들, 명령어 세트들, 프로그래머블 로직 또는 알고리즘들, 하드웨어, 하드웨어 가속기들, 소프트웨어, 펌웨어, 또는 에지 컴퓨팅 노드(650) 내에서 적응된 이들의 조합으로서, 또는 보다 큰 시스템의 섀시 내에 다른 방식으로 포함된 컴포넌트들로서 구현될 수 있다.

[0066] 에지 컴퓨팅 디바이스(650)는 마이크로프로세서, 멀티코어 프로세서, 멀티스레드 프로세서, 초저 전압 프로세서, 임베디드 프로세서, x-PU/DPU/IPU/NPU, 특수 목적 처리 유닛, 특수화된 처리 유닛, 또는 다른 공지된 처리 요소들일 수 있는, 프로세서(652) 형식의 처리 회로를 포함할 수 있다. 프로세서(652)는 프로세서(652) 및 다른 컴포넌트들이 단일 집적 회로, 또는 단일 패키지, 예컨대 캘리포니아주 산타 클라라 소재의 인텔 코포

레이션으로부터의 Edison™ 또는 Galileo™ SoC 보드들이 되도록 형성되는 SoC(system on a chip)의 일부일 수 있다. 예로서, 프로세서(652)는 Quark™, Atom™, i3, i5, i7, i9, 또는 MCU 클래스 프로세서와 같은 Intel® Architecture Core™ 기반 CPU 프로세서, 또는 Intel®로부터 이용가능한 또 다른 이러한 프로세서를 포함할 수 있다. 그러나, 예를 들어, 캘리포니아주 서니베일의 Advanced Micro Devices, Inc.(AMD®)로부터 입수가능한 것, 캘리포니아주 서니베일의 MIPS Technologies, Inc.로부터의 MIPS® 기반 설계, ARM Holdings, Ltd. 또는 그의 고객, 또는 그들의 실시권자들 또는 채택자들로부터 라이센싱된 ARM® 기반 설계와 같은 임의 수의 다른 프로세서들이 사용될 수 있다. 프로세서들은 Apple® Inc.의 A5-A13 프로세서, Qualcomm® Technologies, Inc.의 Snapdragon™ 프로세서, 또는 Texas Instruments, Inc.의 OMAP™ 프로세서와 같은 유닛들을 포함할 수 있다. 프로세서(652) 및 부속 회로는 도 6b에 도시된 모든 요소들보다 적은 요소들을 포함하는 제한된 하드웨어 구성들 또는 구성들을 포함하여, 단일 소켓 폼 팩터, 다중 소켓 폼 팩터, 또는 다양한 다른 포맷들로 제공될 수 있다.

[0067] 프로세서(652)는 인터커넥트(656)(예를 들어, 버스)를 통해 시스템 메모리(654)와 통신할 수 있다. 주어진 양의 시스템 메모리를 제공하기 위해 임의 수의 메모리 디바이스들이 사용될 수 있다. 예들로서, 메모리(654)는 DDR 또는 모바일 DDR 표준들(예를 들어, LPDDR, LPDDR2, LPDDR3, 또는 LPDDR4)과 같은 JEDEC(Joint Electron Devices Engineering Council) 설계에 따른 RAM(random access memory)일 수 있다. 특정 예들에서, 메모리 컴포넌트는, DDR SDRAM에 대한 JESD79F, DDR2 SDRAM에 대한 JESD79-2F, DDR3 SDRAM에 대한 JESD79-3F, DDR4 SDRAM에 대한 JESD79-4A, LPDDR(Low Power DDR)에 대한 JESD209, LPDDR2에 대한 JESD209-2, LPDDR3에 대한 JESD209-3, 및 LPDDR4에 대한 JESD209-4와 같은, JEDEC에 의해 공표된 DRAM 표준을 준수할 수 있다. 이러한 표준들(및 유사한 표준들)은 DDR 기반 표준들이라고 지칭될 수 있고, 이러한 표준들을 구현하는 스토리지 디바이스들의 통신 인터페이스들은 DDR 기반 인터페이스들이라고 지칭될 수 있다. 다양한 구현들에서, 개별 메모리 디바이스들은 SDP(single die package), DDP(dual die package) 또는 쿼드 다이 패키지(QDP)와 같은 임의 수의 상이한 패키지 타입들의 것일 수 있다. 이 디바이스들은, 일부 예들에서, 보다 낮은 프로필의 솔루션을 제공하기 위해 마더보드 상에 직접 납땜될 수 있는 한편, 다른 예들에서, 디바이스들은 주어진 커넥터에 의해 마더보드에 다음 차례로 결합되는 하나 이상의 메모리 모듈로서 구성된다. 다른 타입들의 메모리 모듈들, 예컨대, microDIMM들 또는 MiniDIMM들을 포함하지만 이들로 제한되지는 않는 상이한 각종의 DIMM(dual inline memory module)들과 같은 임의 수의 다른 메모리 구현들이 사용될 수 있다.

[0068] 데이터, 애플리케이션들, 운영 체제들 및 등등과 같은 정보의 영구 스토리지를 제공하기 위해, 스토리지(658)는 또한 인터커넥트(656)를 통해 프로세서(652)에 결합될 수 있다. 예에서, 스토리지(658)는 SSDD(solid-state disk drive)를 통해 구현될 수 있다. 스토리지(658)에 대해 사용될 수 있는 다른 디바이스들은 SD(Secure Digital) 카드들, microSD 카드들, XD(eXtreme Digital) 픽처 카드들, 및 그와 유사한 것과 같은 플래시 메모리 카드들, 및 USB(Universal Serial Bus) 플래시 드라이브들을 포함한다. 예에서, 메모리 디바이스는 칼코게나이드 유리를 사용하는 메모리 디바이스들, 멀티 임계 레벨 NAND 플래시 메모리, NOR 플래시 메모리, 단일 또는 멀티 레벨 PCM(Phase Change Memory), 저항성 메모리, 나노와이어 메모리, FeTRAM(ferroelectric transistor random access memory), 반-강유전성 메모리, 멤리스터 기술을 포함하는 MRAM(magnetoresistive random access memory) 메모리, 금속 산화물 베이스, 산소 베이컨시 베이스 및 CB-RAM(conductive bridge Random Access Memory)를 포함하는 저항성 메모리, 또는 STT(spin transfer torque)-MRAM, 스핀트로닉 자기 접합 메모리 기반 디바이스, MTJ(magnetic tunneling junction) 기반 디바이스, DW(Domain Wall) 및 SOT(Spin Orbit Transfer) 기반 디바이스, 사이리스터 기반 메모리 디바이스, 또는 상기 중 임의의 것의 조합, 또는 다른 메모리일 수 있거나 이들을 포함할 수 있다.

[0069] 저전력 구현들에서, 스토리지(658)는 프로세서(652)와 연관된 온다이(on-die) 메모리 또는 레지스터들일 수 있다. 그러나, 일부 예에서, 스토리지(658)는 마이크로 HDD(hard disk drive)를 이용하여 구현될 수 있다. 게다가, 기술된 기술들에 부가하여 또는 그 대신에, 그 중에서도 특히, 저항 변화 메모리들, 상 변화 메모리들, 홀로그래픽 메모리들, 또는 화학 메모리들과 같은, 임의 수의 새로운 기술들이 스토리지(658)에 대해 사용될 수 있다.

[0070] 컴포넌트들은 인터커넥트(656)를 통해 통신할 수 있다. 인터커넥트(656)는 ISA(industry standard architecture), EISA(extended ISA), PCI(peripheral component interconnect), PCIx(peripheral component interconnect extended), PCIe(PCI express), 또는 임의 수의 다른 기술들을 포함하여, 임의 수의 기술들을 포함할 수 있다. 인터커넥트(656)는, 예를 들어, SoC 기반 시스템에서 사용되는 독점적 버스일 수 있다. 그 중

에서도 특히, I2C(Inter-Integrated Circuit) 인터페이스, SPI(Serial Peripheral Interface) 인터페이스, 포인트 투 포인트 인터페이스들, 및 전력 버스와 같은 다른 버스 시스템들이 포함될 수 있다.

[0071] 인터커넥트(656)는 접속된 에지 디바이스들(662)과의 통신을 위해, 프로세서(652)를 송수신기(666)에 결합할 수 있다. 트랜시버(666)는, 그 중에서도 특히, Bluetooth® Special Interest Group에 의해 정의된 바와 같은 BLE(Bluetooth® low energy) 표준, 또는 ZigBee® 표준을 사용하는, IEEE 802.15.4 표준 하에서의 2.4 GHz(Gigahertz) 송신들과 같은 임의 수의 주파수들 및 프로토콜들을 사용할 수 있다. 특정 무선 통신 프로토콜을 위해 구성된 임의 수의 라디오들이 접속된 에지 디바이스들(662)에 대한 접속들을 위해 사용될 수 있다. 예를 들어, WLAN(wireless local area network) 유닛은 IEEE(Institute of Electrical and Electronics Engineers) 802.11 표준에 따라 Wi-Fi® 통신을 구현하기 위해 사용될 수 있다. 또한, 예를 들어, 셀룰러 또는 다른 무선 광역 프로토콜에 따른 무선 광역 통신은 WWAN(wireless wide area network) 유닛을 통해 발생할 수 있다.

[0072] 무선 네트워크 송수신기(666)(또는 다중의 송수신기)는 상이한 범위에서의 통신을 위해 다중의 표준 또는 라디오를 이용하여 통신할 수 있다. 예를 들어, 에지 컴퓨팅 노드(650)는 전력을 절감하기 위해, BLE(Bluetooth Low Energy)에 기초한 로컬 송수신기, 또는 또 다른 저전력 라디오를 사용하여, 예컨대, 약 10 미터 내의 가까운 디바이스들과 통신할 수 있다. 예를 들어, 약 50 미터 내의 더 멀리 떨어진 접속된 에지 디바이스들(662)은 ZigBee® 또는 다른 중간 전력 라디오들을 통해 도달될 수 있다. 양 통신 기법들은 상이한 전력 레벨들에서 단일 라디오를 통해 발생할 수 있거나, 또는 개별 송수신기들, 예를 들어, BLE를 사용하는 로컬 송수신기 및 ZigBee®를 사용하는 별개의 메시 송수신기를 통해 발생할 수 있다.

[0073] 로컬 또는 광역 네트워크 프로토콜을 통해 클라우드(예를 들어, Edge 클라우드(695)) 내의 디바이스들 또는 서비스들과 통신하기 위해 무선 네트워크 송수신기(666)(예를 들어, 라디오 송수신기)가 포함될 수 있다. 무선 네트워크 송수신기(666)는, 그 중에서도 특히, IEEE 802.15.4, 또는 IEEE 802.15.4g 표준들을 따르는 LPWA(low-power wide-area) 송수신기일 수 있다. 에지 컴퓨팅 노드(650)는 Semtech 및 LoRa Alliance에 의해 개발된 LoRaWAN™(Long Range Wide Area Network)을 사용하여 광역에 걸쳐 통신할 수 있다. 본 명세서에 설명된 기법들은 이러한 기술들로 제한되지는 않고, Sigfox와 같은 장거리 저 대역폭 통신, 및 다른 기술들을 구현하는 임의 수의 다른 클라우드 송수신기들과 함께 사용될 수 있다. 또한, IEEE 802.15.4e 사양에 기술된 시간 슬롯 채널 호핑과 같은 다른 통신 기법들이 이용될 수 있다.

[0074] 본 명세서에 설명된 바와 같이, 무선 네트워크 송수신기(666)에 대해 언급된 시스템들에 부가하여 임의 수의 다른 라디오 통신들 및 프로토콜들이 사용될 수 있다. 예를 들어, 송수신기(666)는 고속 통신을 구현하기 위해 확산 스펙트럼(SPA/SAS) 통신을 이용하는 셀룰러 송수신기를 포함할 수 있다. 또한, 중속 통신 및 네트워크 통신의 제공을 위한 Wi-Fi® 네트워크들과 같은 임의 수의 다른 프로토콜들이 사용될 수 있다. 송수신기(666)는 본 개시내용의 끝에서 더 상세히 논의되는, LTE(Long Term Evolution) 및 5세대(5G) 통신 시스템들과 같은 임의 수의 3GPP(Third Generation Partnership Project) 사양들과 호환가능한 라디오들을 포함할 수 있다. 에지 클라우드(695)의 노드들에 또는 (예를 들어, 메시에서 동작하는) 접속된 에지 디바이스들(662)과 같은 다른 디바이스들에 유선 통신을 제공하기 위해 NIC(network interface controller)(668)가 포함될 수 있다. 유선 통신은 이더넷 접속을 제공할 수 있거나, 또는 많은 것들 중에서도 특히, CAN(Controller Area Network), LIN(Local Interconnect Network), DeviceNet, ControlNet, Data Highway+, PROFIBUS, 또는 PROFINET와 같은 다른 타입의 네트워크들에 기초할 수 있다. 추가적인 NIC(668), 예를 들어, 이더넷을 통해 클라우드에 통신을 제공하는 제1 NIC(668), 및 또 다른 타입의 네트워크를 통해 다른 디바이스들에 통신을 제공하는 제2 NIC(668)가 제2 네트워크에 대한 접속을 가능하게 하기 위해 포함될 수 있다.

[0075] 디바이스로부터 또 다른 컴포넌트 또는 네트워크로의 다양한 타입의 적용가능한 통신이 주어지면, 디바이스에 의해 사용되는 적용가능한 통신 회로는 컴포넌트들(664, 666, 668, 또는 670) 중 임의의 하나 이상을 포함하거나 이에 의해 구체화될 수 있다. 따라서, 다양한 예들에서, 통신(예를 들어, 수신, 송신 등)을 위한 적용가능한 수단은 그러한 통신 회로에 의해 구체화될 수 있다.

[0076] 에지 컴퓨팅 노드(650)는, 하나 이상의 AI(artificial intelligence) 가속기, 신경 컴퓨팅 스틱, 뉴로모픽 하드웨어, FPGA, GPU들의 배열, x-PU들/DPU들/IPU/NPU들의 배열, 하나 이상의 SoC, 하나 이상의 CPU, 하나 이상의 디지털 신호 프로세서, 전용 ASIC들, 또는 하나 이상의 특수화된 태스크를 달성하도록 설계된 다른 형식들의 특수화된 프로세서 또는 회로에 의해 구체화될 수 있는 가속 회로(664)를 포함하거나 이에 결합될 수 있다. 이들 태스크는 AI 처리(머신 러닝, 훈련, 추론, 및 분류 동작들을 포함함), 시각 데이터 처리, 네트워크 데이터

처리, 객체 검출, 규칙 분석, 또는 그와 유사한 것을 포함할 수 있다. 이들 태스크는 또한 본 문서의 다른 곳에서 논의된 서비스 관리 및 서비스 동작들을 위한 특정 에지 컴퓨팅 태스크들을 포함할 수 있다.

[0077] 인터커넥트(656)는 추가적인 디바이스들 또는 서브시스템들을 접속하기 위해 사용되는 센서 허브 또는 외부 인터페이스(670)에 프로세서(652)를 결합할 수 있다. 디바이스들은 가속도계들, 레벨 센서들, 흐름 센서들, 광학 광 센서들, 카메라 센서들, 온도 센서들, 글로벌 내비게이션 시스템(예를 들어, GPS) 센서들, 압력 센서들, 기압 센서들, 및 그와 유사한 것과 같은 센서들(672)을 포함할 수 있다. 허브 또는 인터페이스(670)는 추가로 에지 컴퓨팅 노드(650)를, 전력 스위치들, 밸브 액추에이터들, 가청 사운드 발생기, 시각적 경고 디바이스, 및 이와 유사한 것과 같은 액추에이터들(674)에 접속시키기 위해 사용될 수 있다.

[0078] 일부 선택적 예들에서, 다양한 입력/출력(I/O) 디바이스들이 에지 컴퓨팅 노드(650) 내에 존재하거나 그에 접속될 수 있다. 예를 들어, 센서 판독값들 또는 액추에이터 위치와 같은 정보를 보여주기 위해 디스플레이 또는 다른 출력 디바이스(684)가 포함될 수 있다. 입력을 수용하기 위해 터치 스크린 또는 키패드와 같은 입력 디바이스(686)가 포함될 수 있다. 출력 디바이스(684)는, 바이너리 상태 표시기들(예를 들어, 발광 다이오드(LED)들) 및 다중 문자 시각적 출력들과 같은 단순한 시각적 출력들, 또는 디스플레이 스크린들(예를 들어, 액정 디스플레이(LCD) 스크린들)과 같은 더 복잡한 출력들을 포함하는 임의 수의 형식의 오디오 또는 시각적 디스플레이를 포함할 수 있고, 문자들, 그래픽들, 멀티미디어 객체들, 및 그와 유사한 것의 출력이 에지 컴퓨팅 노드(650)의 동작으로부터 발생되거나 생성된다. 디스플레이 또는 콘솔 하드웨어는, 본 시스템의 맥락에서, 에지 컴퓨팅 시스템의 출력을 제공하고 입력을 수신하기 위해; 에지 컴퓨팅 시스템의 컴포넌트들 또는 서비스들을 관리하기 위해; 에지 컴퓨팅 컴포넌트 또는 서비스의 상태를 식별하기 위해; 또는 임의의 다른 수의 관리 또는 운영 함수들 또는 서비스 사용 사례들을 수행하기 위해 사용될 수 있다.

[0079] 배터리(676)가 에지 컴퓨팅 노드(650)에 전력을 공급할 수 있지만, 에지 컴퓨팅 노드(650)가 고정된 로케이션에 장착되는 예들에서, 그것은 전기 그리드에 결합된 전원을 가질 수 있거나, 또는 배터리는 백업으로서 또는 일시적 능력들을 위해 사용될 수 있다. 배터리(676)는 리튬 이온 배터리, 또는, 아연-공기 배터리, 알루미늄-공기 배터리, 리튬-공기 배터리, 및 그와 유사한 것과 같은 금속-공기 배터리일 수 있다.

[0080] 배터리 모니터/충전기(678)는, 포함된 경우, 배터리(676)의 충전 상태(SoCh)를 추적하기 위해 에지 컴퓨팅 노드(650)에 포함될 수 있다. 배터리 모니터/충전기(678)는, 배터리(676)의 SoH(state of health) 및 SoF(state of function)와 같은, 장애 예측들을 제공하기 위해 배터리(676)의 다른 파라미터들을 모니터링하기 위해 사용될 수 있다. 배터리 모니터/충전기(678)는, Linear Technologies로부터의 LTC4020 또는 LTC2990, 아리조나주 퍼닉스의 ON Semiconductor로부터의 ADT7488A, 또는 텍사스주 달라스의 Texas Instruments의 UCD90xxx 계열로부터의 IC와 같은 배터리 모니터링 집적 회로를 포함할 수 있다. 배터리 모니터/충전기(678)는 배터리(676)에 대한 정보를 인터커넥트(656)를 통해 프로세서(652)에 통신할 수 있다. 배터리 모니터/충전기(678)는 프로세서(652)가 배터리(676)의 전압 또는 배터리(676)로부터의 전류 흐름을 직접 모니터링할 수 있게 하는 아날로그-투-디지털(ADC) 컨버터를 또한 포함할 수 있다. 배터리 파라미터들은 송신 주파수, 메시 네트워크 동작, 주파수 감지, 및 그와 유사한 것과 같은, 에지 컴퓨팅 노드(650)가 수행할 수 있는 액션들을 결정하기 위해 사용될 수 있다.

[0081] 전력 블록(680), 또는 그리드에 결합된 다른 전원이 배터리(676)를 충전하기 위해 배터리 모니터/충전기(678)와 결합될 수 있다. 일부 예들에서, 전력 블록(680)은, 예를 들어, 에지 컴퓨팅 노드(650) 내의 루프 안테나를 통해 무선으로 전력을 획득하기 위해 무선 전력 수신기로 대체될 수 있다. 그 중에서도 특히, 캘리포니아주 밀피타스 소재의 Linear Technologies의 LTC4020 칩과 같은 무선 배터리 충전 회로가 배터리 모니터/충전기(678)에 포함될 수 있다. 특정 충전 회로들은 배터리(676)의 크기, 및 따라서 요구되는 전류에 기초하여 선택될 수 있다. 충전은, 그 중에서도 특히, Airfuel Alliance에 의해 공표된 Airfuel 표준, Wireless Power Consortium에 의해 공표된 Qi 무선 충전 표준, 또는 Alliance for Wireless Power에 의해 공표된 Rezence 충전 표준을 사용하여 수행될 수 있다.

[0082] 스토리지(658)는 본 명세서에 설명된 기법들을 구현하기 위해 소프트웨어, 펌웨어, 또는 하드웨어 커맨드들의 형식의 명령어들(682)을 포함할 수 있다. 이러한 명령어들(682)이 메모리(654) 및 스토리지(658)에 포함된 코드 블록들로서 도시되어 있지만, 코드 블록들 중 임의의 것이, 예를 들어, ASIC(application specific integrated circuit) 내에 구축된 하드와이어드 회로들로 대체될 수 있다는 것이 이해될 수 있다.

[0083] 예에서, 메모리(654), 스토리지(658), 또는 프로세서(652)를 통해 제공되는 명령어들(682)은 에지 컴퓨팅 노드(650)에서 전자적 동작들을 수행하라고 프로세서(652)에 지시하는 코드를 포함하는 비일시적 머신 판독가능 매

체(660)로서 구체화될 수 있다. 프로세서(652)는 인터커넥트(656)를 통해 비일시적 머신 판독가능 매체(660)에 액세스할 수 있다. 예를 들어, 비일시적 머신 판독가능 매체(660)는 스토리지(658)에 대해 설명된 디바이스들에 의해 구체화될 수 있거나, 광학 디스크들(예를 들어, DVD(digital versatile disk), CD(compact disk), CD-ROM, 블루레이 디스크), 플래시 드라이브들, 플로피 디스크들, 하드 드라이브들(예를 들어, SSD들), 또는 정보가 임의의 지속기간 동안 (예를 들어, 연장된 시간 기간들 동안, 영구적으로, 짧은 인스턴스들 동안, 일시적으로 버퍼링하는 동안, 및/또는 캐싱하는 동안) 저장되는 임의 수의 다른 하드웨어 디바이스들을 포함하는 스토리지 디바이스들 및/또는 스토리지 디스크들과 같은 특정 스토리지 유닛들을 포함할 수 있다. 비일시적 머신 판독가능 매체(660)는, 예를 들어, 위에 묘사된 동작들 및 기능성의 흐름도(들) 및 블록도(들)에 관하여 설명된 바와 같이, 액션들의 특정 시퀀스 또는 흐름을 수행하도록 프로세서(652)에 지시하는 명령어들을 포함할 수 있다. 본 명세서에 사용되는 바와 같이, 용어들 "머신 판독가능 매체(machine-readable medium)" 및 "컴퓨터 판독가능 매체(computer-readable medium)"는 교환가능하다. 본 명세서에서 사용될 때, "비일시적 컴퓨터 판독가능 매체"라는 용어는 임의 타입의 컴퓨터 판독가능 스토리지 디바이스 및/또는 스토리지 디스크를 포함하고, 전파 신호들을 배제하고, 송신 매체를 배제하는 것으로 명확히 정의된다.

[0084] 또한 특정 예에서, 프로세서(652) 상의 명령어들(682)은 (머신 판독가능 매체(660)의 명령어들(682)과 별개로, 또는 그와 조합되어) TEE(trusted execution environment)(690)의 실행 또는 동작을 구성할 수 있다. 예에서, TEE(690)는 명령어들의 보안 실행 및 데이터에 대한 보안 액세스를 위해 프로세서(652)가 액세스가능한 보호된 영역으로서 동작한다. TEE(690), 및 프로세서(652) 또는 메모리(654)에서의 수반되는 보안 영역의 다양한 구현들은, 예를 들어, Intel® SGX(Software Guard Extensions) 또는 ARM® TrustZone® 하드웨어 보안 확장들, Intel® ME(Management Engine), 또는 Intel® CSME(Converged Security Manageability Engine)의 사용을 통해 제공될 수 있다. 보안 강화(security hardening), 하드웨어 신뢰 루트들(roots-of-trust), 및 신뢰된 또는 보호된 동작들의 다른 양태들이 TEE(690) 및 프로세서(652)를 통해 디바이스(650)에서 구현될 수 있다.

[0085] 도 6a 및 도 6b의 예시된 예들이 제각기 컴퓨팅 노드 및 컴퓨팅 디바이스에 대한 예시적인 컴포넌트들을 포함하지만, 본 명세서에 개시된 예들은 이것들에만 제한되지는 않는다. 본 명세서에서 사용될 때, "컴퓨터"는 상이한 타입의 컴퓨팅 환경들에서 도 6a 및/또는 도 6b의 예시적인 컴포넌트들의 일부 또는 전부를 포함할 수 있다. 예시적인 컴퓨팅 환경들은, 참여하는 에지 컴퓨팅 디바이스들 중 특정의 것들이 이종(heterogenous) 또는 동종(homogeneous) 디바이스들이도록, 분산 네트워킹 배열로 된 에지 컴퓨팅 디바이스들(예컨대, 에지 컴퓨터들)을 포함한다. 본 명세서에서 사용될 때, "컴퓨터"는 개인용 컴퓨터, 서버, 사용자 장비, 가속기 등을 포함할 수 있으며, 이들의 임의의 조합들을 포함할 수 있다. 일부 예들에서, 분산 네트워킹 및/또는 분산 컴퓨팅은 도 6a 및/또는 도 6b에 예시된 바와 같은 임의의 수의 그러한 에지 컴퓨팅 디바이스들을 포함하며, 이들 각각은 상이한 서브컴포넌트들, 상이한 메모리 용량들, I/O 능력들 등을 포함할 수 있다. 예를 들어, 분산 네트워킹 및/또는 분산 컴퓨팅의 일부 구현들이 특정의 원하는 기능성과 연관되기 때문에, 본 명세서에 개시된 예들은 분산 컴퓨팅 태스크들의 기능적 목표를 만족시키기 위해 도 6a 및/또는 도 6b에 나타낸 컴포넌트들의 상이한 조합들을 포함한다. 일부 예들에서, "컴퓨팅 노드" 또는 "컴퓨터"라는 용어는 도 6a의 예시적인 프로세서(604), 메모리(606) 및 I/O 서브시스템(608)만을 포함한다. 일부 예들에서, 분산 컴퓨팅 태스크(들)의 하나 이상의 목표 함수는, 데이터 스토리지(예컨대, 예시적인 데이터 스토리지(610)), 입력/출력 능력들(예컨대, 예시적인 주변 디바이스(들)(614)), 및/또는 네트워크 통신 능력들(예컨대, 예시적인 NIC(620))를 수용하기 위한 디바이스들과 같은, 에지 네트워킹 환경의 상이한 부분들에 위치된 하나 이상의 대안 디바이스/구조물에 의존한다.

[0086] 일부 예들에서, 분산 컴퓨팅 및/또는 분산 네트워킹 환경(예컨대, 에지 네트워크)에서 동작하는 컴퓨터들은 계산 낭비를 감소시키는 방식으로 특정의 목표 기능성을 수용하도록 구조화되어 있다. 예를 들어, 컴퓨터가 도 6a 및 도 6b에 개시된 컴포넌트들의 서브세트를 포함하기 때문에, 이러한 컴퓨터들은, 그렇지 않았더라면 사용되지 않고 및/또는 충분히 이용되지 않을 컴퓨팅 구조를 포함하지 않고서, 분산 컴퓨팅 목표 함수들의 실행을 충족시킨다. 이와 같이, 본 명세서에서 사용되는 "컴퓨터"라는 용어는 분산 컴퓨팅 태스크들의 목표 함수들을 충족시키고 및/또는 그렇지 않으면 실행할 수 있는 도 6a 및/또는 도 6b의 구조의 임의의 조합을 포함한다. 일부 예들에서, 컴퓨터들은 동적 수요와 관련하여 다운스케일링 또는 업스케일링하는 방식으로 대응하는 분산 컴퓨팅 목표 함수들에 부합하는 방식으로 구조화된다. 일부 예들에서, 분산 컴퓨팅 요청(들)의 하나 이상의 태스크를 처리하는 그들의 능력을 고려하여 상이한 컴퓨터들이 기동되고 및/또는 그렇지 않은 경우에는 인스턴스화되어, 태스크들을 만족시킬 수 있는 임의의 컴퓨터가 그러한 컴퓨팅 활동으로 진행하도록 한다.

[0087] 도 6a 및 도 6b의 예시된 예들에서, 컴퓨팅 디바이스들은 운영 체제들을 포함한다. 본 명세서에서 사용되는 바와 같이, "운영 체제"는 도 6a의 예시적인 에지 컴퓨팅 노드(600) 및/또는 도 6b의 예시적인 에지 컴퓨팅 노드

(650)와 같은 예시적인 컴퓨팅 디바이스들을 제어하기 위한 소프트웨어이다. 예시적인 운영 체제들은 소비자 기반 운영 체제들(예를 들어, Microsoft® Windows® 10, Google® Android® OS, Apple® Mac® OS 등)을 포함하지만, 이에 제한되지는 않는다. 예시적인 운영 체제들은 또한 실시간 운영 체제들, 하이퍼바이저들 등과 같은 산업에 초점을 맞춘 운영 체제들을 포함하지만, 이에 제한되지는 않는다. 제1 에지 컴퓨팅 노드 상의 예시적인 운영 체제는 제2 에지 컴퓨팅 노드 상의 예시적인 운영 체제와 동일하거나 상이할 수 있다. 일부 예들에서, 운영 체제는 특정 통신 프로토콜들 및/또는 인터프리터들과 같은, 운영 체제에 고유하지 않은 하나 이상의 함수 및/또는 동작을 용이하게 하기 위한 대안 소프트웨어를 기동한다. 일부 예들에서, 운영 체제는 운영 체제에 고유하지 않은 다양한 기능성들을 인스턴스화한다. 일부 예에서, 운영 체제는 변하는 정도의 복잡성 및/또는 능력을 포함한다. 예를 들어, 제1 에지 컴퓨팅 노드에 대응하는 제1 운영 체제는 동적 입력 조건들에 대한 응답성의 특정 성능 예상들을 갖는 실시간 운영 체제를 포함하고, 제2 에지 컴퓨팅 노드에 대응하는 제2 운영 체제는 최종 사용자 I/O를 용이하게 하는 그래픽 사용자 인터페이스 능력들을 포함한다.

[0088] 명령어들(682)은 또한 다수의 WLAN(wireless local area network) 전송 프로토콜(예를 들어, 프레임 릴레이, IP(internet protocol), TCP(transmission control protocol), UDP(user datagram protocol), HTTP(hypertext transfer protocol) 등) 중 어느 하나를 활용하여 무선 네트워크 송수신기(466)를 통해 송신 매체를 이용하여 통신 네트워크를 통해 송신 또는 수신될 수 있다. 예시적인 통신 네트워크들은 LAN(local area network), WAN(wide area network), 패킷 데이터 네트워크(예를 들어, 인터넷), 모바일 폰 네트워크들(예를 들어, 셀룰러 네트워크들), POTS(Plain Old Telephone) 네트워크들, 및 무선 데이터 네트워크들을 포함할 수 있다. 네트워크들을 통한 통신은, 그 중에서도 특히, Wi-Fi라고 알려진 IEEE(Institute of Electrical and Electronics Engineers) 802.11 표준들의 계열, IEEE 802.16 표준들의 계열, IEEE 802.15.4 표준들의 계열, LTE(Long Term Evolution) 표준들의 계열, UMTS(Universal Mobile Telecommunications System) 표준들의 계열, P2P(peer-to-peer) 네트워크들, NG(next generation)/5G(5th generation) 표준들과 같은, 하나 이상의 상이한 프로토콜을 포함할 수 있다.

[0089] 본 명세서에서 사용되는 바와 같은 "회로(circuitry)"라는 용어는, 설명된 기능성을 제공하도록 구성되는, 전자 회로, 로직 회로, 프로세서(공유, 전용, 또는 그룹) 및/또는 메모리(공유, 전용, 또는 그룹), ASIC(Application Specific Integrated Circuit), FPD(field-programmable device)(예를 들어, FPGA(field-programmable gate array), PLD(programmable logic device), CPLD(complex PLD), HCPLD(high-capacity PLD), 구조화된 ASIC, 또는 프로그래머블 SoC), DSP들(digital signal processors) 등과 같은 하드웨어 컴포넌트들을 지칭하거나, 이들의 일부이거나, 또는 이들을 포함한다는 점에 유의한다. 일부 실시예들에서, 회로는 설명된 기능성의 적어도 일부를 제공하기 위해 하나 이상의 소프트웨어 또는 펌웨어 프로그램을 실행할 수 있다. "회로"라는 용어는 또한 하나 이상의 하드웨어 요소와 그 프로그램 코드의 기능성을 수행하기 위해 사용되는 프로그램 코드의 조합(또는 전기 또는 전자 시스템에서 사용되는 회로들의 조합)을 지칭할 수 있다. 이러한 실시예들에서, 하드웨어 요소들과 프로그램 코드의 조합은 특정 타입의 회로라고 지칭될 수 있다.

[0090] 따라서, 본 명세서에서 사용되는 바와 같은 "프로세서 회로" 또는 "프로세서"라는 용어는 산술 또는 논리 연산들의 시퀀스를 순차적으로 그리고 자동으로 수행하거나, 또는 디지털 데이터를 기록, 저장 및/또는 전송할 수 있는 회로를 지칭하거나, 그의 일부이거나, 그를 포함할 수 있다. 용어 "프로세서 회로" 또는 "프로세서"는, 하나 이상의 애플리케이션 프로세서, 하나 이상의 기저대역 프로세서, 물리적 CPU(central processing unit), 단일 또는 멀티 코어 프로세서, 및/또는 프로그램 코드, 소프트웨어 모듈들, 및/또는 함수 프로세스들과 같은 컴퓨터 실행가능 명령어들을 실행하거나 다른 경우에는 동작시킬 수 있는 임의의 다른 디바이스를 지칭할 수 있다.

[0091] 본 명세서에 설명되는 라디오 링크들 중 임의의 것은 이하의 것들을 포함하지만 이에 제한되지는 않는 이하의 라디오 통신 기술들 및/또는 표준들 중 임의의 하나 이상의 것에 따라 동작할 수 있다: GSM(Global System for Mobile Communications) 라디오 통신 기술, GPRS(General Packet Radio Service) 라디오 통신 기술, EDGE(Enhanced Data Rates for GSM Evolution) 라디오 통신 기술, 및/또는 3GPP(Third Generation Partnership Project) 라디오 통신 기술, 예를 들어, UMTS(Universal Mobile Telecommunications System), FOMA(Freedom of Multimedia Access), 3GPP LTE(Long Term Evolution), 3GPP LTE 어드밴스드(Advanced), CDMA2000(Code division multiple access 2000), CDPD(Cellular Digital Packet Data), 모비텍스(Mobitex), 3G(Third Generation), CSD(Circuit Switched Data), HSCSD(High-Speed Circuit-Switched Data), UMTS(3G)(Universal Mobile Telecommunications System), W-CDMA(UMTS)(Wideband Code Division Multiple Access(Universal Mobile Telecommunications System)), HSPA(High Speed Packet Access), HSDPA(High-Speed

Downlink Packet Access), HSUPA(High-Speed Uplink Packet Access), HSPA+(High Speed Packet Access Plus), UMTS-TDD(Universal Mobile Telecommunications System-Time-Division Duplex), TD-CDMA(Time Division-Code Division Multiple Access), TD-CDMA(Time Division-Synchronous Code Division Multiple Access), 3세대 파트너십 프로젝트 릴리스 8(Pre-4th Generation)(3GPP Rel. 8(Pre-4G)), 3GPP Rel. 9(3rd Generation Partnership Project Release9), 3GPP Rel. 10(3rd Generation Partnership Project Release10), 3GPP Rel. 11(3rd Generation Partnership Project Release11), 3GPP Rel. 12(3rd Generation Partnership Project Release12), 3GPP Rel. 13(3rd Generation Partnership Project Release13), 3GPP Rel. 14(3rd Generation Partnership Project Release14), 3GPP Rel. 15(3rd Generation Partnership Project Release15), 3GPP Rel. 16(3rd Generation Partnership Project Release16), 3GPP Rel. 17(3rd Generation Partnership Project Release17) 및 후속 릴리스들(예컨대 Rel. 18, Rel. 19 등), 3GPP5G, 5G, 5G NR(5G New Radio), 3GPP 5G New Radio, 3GPP LTE Extra, LTE-Advanced Pro, LTE LAA(Licensed-Assisted Access), MuLTEfire, UTRA(UMTS Terrestrial Radio Access), E-UTRA(Evolved UMTS Terrestrial Radio Access), LTE 어드밴스드(4G)(Long Term Evolution Advanced(4th generation)), cdmaOne(2G), CDMA 2000(3G)(Code division multiple access 2000(Third generation)), EV-DO(Evolution-Data Optimized or Evolution-Data Only), AMPS(1G)(Advanced Mobile Phone System(1st generation)), TACS/ETACS(Total Access Communication System/Extended Total Access Communication System), D-AMPS (2G)(Digital AMPS (2nd Generation)), PTT(Push-to-talk), MTS(Mobile Telephone System), IMTS(Improved Mobile Telephone System), AMTS(Advanced Mobile Telephone System), OLT(Norwegian for Offentlig Landmobil Telefoni, Public Land Mobile Telephony), MTD(Swedish abbreviation for Mobiltelefonisystem D, or Mobile telephony system D), Autotel/PALM(Public Automated Land Mobile), ARP(Finnish for Autoradiopuhelin, “car radio phone”), NMT(Nordic Mobile Telephony), Hicap(High capacity version of NTT(Nippon Telegraph and Telephone)), CDPD(Cellular Digital Packet Data), 모비텍스(Mobitex), DataTAC, iDEN(Integrated Digital Enhanced Network), PDC(Personal Digital Cellular), CSD(Circuit Switched Data), PHS(Personal Handy-phone System), WiDEN(Wideband Integrated Digital Enhanced Network), iBurst, 3GPP 일반 액세스 네트워크 또는 GAN 표준으로 또한 지칭되는 UMA(Unlicensed Mobile Access), Zigbee, Bluetooth(r), WiGig(Wireless Gigabit Alliance) 표준, 일반적인 mmWave 표준들(WiGig, IEEE 802.11ad, IEEE 802.11ay 등과 같이 10-300 GHz 이상에서 동작하는 무선 시스템들), 300 GHz 이상 및 THz 대역에서 동작하는 기술들, (3GPP/LTE 기반 또는 IEEE 802.11p 또는 IEEE 802.11bd 및 기타) V2V(Vehicle-to-Vehicle) 및 V2X(Vehicle-to-X) 및 V2I(Vehicle-to-Infrastructure) 및 I2V(Infrastructure-to-Vehicle) 통신 기술들, 3GPP 셀룰러 V2X, DSRC(Dedicated Short Range Communications) 통신 시스템들, 예컨대 지능형 전송 시스템들(Intelligent-Transport-Systems) 및 기타(전형적으로 5850 MHz 내지 5925 MHz 이상에서 동작함(전형적으로 CEPT Report 71에서의 변경 제안들에 따르면 최대 5935 MHz까지)), 유럽 ITS-G5 시스템(즉, IEEE 802.11p 기반 DSRC의 유럽 변형, ITS-G5A(즉, 주파수 범위 5,875 GHz 내지 5,905 GHz에서 안전성 관련 응용들을 위해 ITS에 전용되는 유럽 ITS 주파수 대역들에서의 ITS-G5의 동작), ITS-G5B(즉, 주파수 범위 5,855 GHz 내지 5,875 GHz에서의 ITS 비-안정성 응용들에 전용되는 유럽 ITS 주파수 대역들에서의 동작), ITS-G5C(즉, 주파수 범위 5,470 GHz 내지 5,725 GHz에서의 ITS 응용들의 동작)를 포함함), 700 MHz(715 MHz 내지 725 MHz 포함함) 대역에서의 일본의 DSRC, IEEE 802.11bd 기반 시스템들, 등.

[0092] 본 명세서에 기술되는 양태들은 전용 면허 스펙트럼, 비면허 스펙트럼, 면허 면제 스펙트럼, (면허) 공유 스펙트럼(예컨대, LSA = 2.3 내지 2.4GHz, 3.4 내지 3.6GHz, 3.6 내지 3.8GHz 및 추가의 주파수들에서의 면허 공유 액세스(Licensed Shared Access) 및 SAS = 3.55 내지 3.7GHz 및 추가의 주파수들에서의 스펙트럼 액세스 시스템/CBRS = 민간 광대역 라디오 시스템(Citizen Broadband Radio System))을 포함하는 임의의 스펙트럼 관리 스킴의 맥락에서 사용될 수 있다. 적용가능한 스펙트럼 대역들은 IMT(International Mobile Telecommunications) 스펙트럼뿐만 아니라 국가 할당에 의한 대역들과 같은 다른 타입들의 스펙트럼/대역들을 포함한다(국가 할당은 450-470MHz, 902-928MHz(주: 예를 들어, 미국(에 할당됨FCC 파트 15)), 863-868.6MHz(주: 예를 들어, 유럽 연합에 할당됨(ETSI EN300 220)), 915.9-929.7MHz(주: 예를 들어, 일본에 할당됨), 917-923.5MHz(주: 예를 들어, 한국에 할당됨), 755-779MHz 및 779-787MHz(주: 예를 들어, 중국에 할당됨), 790-960MHz, 1710-2025MHz, 2110-2200MHz, 2300-2400MHz, 2.4-2.4835GHz(주: 글로벌 가용성을 갖는 ISM 대역이고, 이것은 Wi-Fi 기술 계열(11b/g/n/ax)에 의해 그리고 또한 블루투스에 의해 사용됨), 2500-2690MHz, 698-790MHz, 610-790MHz, 3400-3600MHz, 3400-3800MHz, 3800-4200MHz, 3.55-3.7GHz(주: 예를 들어, 민간 광대역 라디오 서비스를 위해 미국에 할당됨), 5.15-5.25GHz 및 5.25-5.35GHz 및 5.47-5.725GHz 및 5.725-5.85GHz

대역들(주: 예를 들어, 미국에 할당됨(FCC 파트 15), 총 500MHz 스펙트럼에서 4개의 U-NII 대역들로 구성됨), 5.725-5.875GHz(주: 예를 들어, EU에 할당됨(ETSI EN301 893)), 5.47-5.65GHz(주: 예를 들어, 한국에 할당됨), 5925-7085MHz 및 5925-6425MHz 대역(주: 제각기 미국 및 EU에서 고려 중임). 차세대 Wi-Fi 시스템은 동작 대역으로서 6GHz 스펙트럼을 포함할 것으로 예상되지만, 2017년 12월부로, Wi-Fi 시스템은 이 대역에서 아직 허용되지 않는다는 점이 주목된다. 2019-2020 기간에서 규정이 완료될 것으로 예상됨), IMT-어드밴스드 스펙트럼, IMT-2020 스펙트럼(3600-3800MHz, 3800-4200MHz, 3.5GHz 대역들, 700MHz 대역들, 24.25-86GHz 범위 내의 대역들 등을 포함할 것으로 예상됨), FCC의 "스펙트럼 프론티어(Spectrum Frontier)" 5G 이니셔티브 하에서 이용가능하게 되는 스펙트럼(27.5-28.35GHz, 29.1-29.25GHz, 31-31.3GHz, 37-38.6GHz, 38.6-40GHz, 42-42.5GHz, 57-64GHz, 71-76GHz, 81-86GHz 및 92-94GHz 등을 포함함), 5.9GHz(전형적으로 5.85-5.925GHz) 및 63-64GHz의 ITS(Intelligent Transport Systems) 대역, WiGig Band 1(57.24-59.40GHz), WiGig Band 2(59.40-61.56GHz) 및 WiGig Band 3(61.56-63.72GHz) 및 WiGig Band 4(63.72-65.88GHz)와 같은 WiGig에 현재 할당된 대역들, 57-64/66GHz(주: 이 대역은 MGWS(Multi-Gigabit Wireless Systems)/WiGig에 대한 거의 글로벌 지정을 갖는다. 미국(FCC 파트 15)에서 총 14 GHz를 할당하는 한편, EU(고정된 P2P에 대한 ETSI EN 302 567 및 ETSI EN 301 217-2)는 총 9GHz 스펙트럼을 할당함), 70.2GHz-71GHz 대역, 65.88GHz와 71GHz 사이의 임의의 대역, 76-81GHz와 같은 자동차 레이더 응용들에 현재 할당된 대역들, 및 94-300GHz 및 그 이상을 포함하는 미래의 대역들). 더욱이, 이 스킴은 특히 400MHz 및 700MHz 대역들이 유망한 후보들인 TV 화이트 스페이스 대역들(전형적으로 790MHz 아래)과 같은 대역들에 대해 2차적으로 사용될 수 있다. 셀룰러 응용들 외에, PMSE(Program Making and Special Events), 의료, 건강, 수술, 자동차, 저 레이턴시, 드론 등의 응용들과 같은 수직 시장(vertical market)들에 대한 특정 응용들이 다루어질 수 있다.

[0093] 도 7은 도 6b의 예시적인 컴퓨터 판독가능 명령어들(682)과 같은 소프트웨어를 예시적인 프로세서 플랫폼(들)(710) 및/또는 예시적인 접속된 에지 디바이스들과 같은 하나 이상의 디바이스에 배포하는 예시적인 소프트웨어 배포 플랫폼(705)을 도시한다. 예시적인 소프트웨어 배포 플랫폼(705)은 소프트웨어를 저장하고 다른 컴퓨팅 디바이스들(예컨대, 제3자들, 예시적인 접속된 에지 디바이스들)에 송신할 수 있는 임의의 컴퓨터 서버, 데이터 설비, 클라우드 서비스 등에 의해 구현될 수 있다. 예시적인 접속된 에지 디바이스들은 고객들, 클라이언트들, 관리 디바이스들(예를 들어, 서버들), 제3자들(예를 들어, 소프트웨어 배포 플랫폼(705)을 소유 및/또는 운영하는 엔티티의 고객들)일 수 있다. 예시적인 접속된 에지 디바이스들은 상업 및/또는 가정 자동화 환경들에서 동작할 수 있다. 일부 예들에서, 제3자는 개발자, 판매자, 및/또는 도 6b의 예시적인 컴퓨터 판독가능 명령어들(682)과 같은 소프트웨어의 라이센서이다. 제3자는 사용 및/또는 재판매 및/또는 서브라이센싱을 위해 소프트웨어를 구매 및/또는 라이센싱하는 소비자, 사용자, 소매업자, OEM 등일 수 있다. 일부 예들에서, 분산 소프트웨어는 하나 이상의 UI(user interfaces) 및/또는 GUI(graphical user interfaces)의 디스플레이로 하여금 서로 지리적으로 및/또는 논리적으로 분리된 하나 이상의 디바이스(예를 들어, 접속된 에지 디바이스들)(예를 들어, 물 분포 제어(예를 들어, 펌프들), 전기 분포 제어(예를 들어, 릴레이들) 등의 책임을 전담하는 물리적으로 분리된 IoT 디바이스들)를 식별하게 야기한다.

[0094] 도 7의 도시된 예에서, 소프트웨어 배포 플랫폼(705)은 하나 이상의 서버 및 하나 이상의 스토리지 디바이스를 포함한다. 스토리지 디바이스들은 컴퓨터 판독가능 명령어들(682)을 저장한다. 예시적인 소프트웨어 배포 플랫폼(705)의 하나 이상의 서버는 위에서 설명된 예시적인 네트워크들 중 임의의 것 및/또는 인터넷 중 임의의 하나 이상의 것에 대응할 수 있는 네트워크(715)와 통신 상태에 있다. 일부 예들에서, 하나 이상의 서버는 상업적 트랜잭션의 일환으로서 요청 측 당사자에게 소프트웨어를 송신하라는 요청들에 응답한다. 소프트웨어의 전달, 판매 및/또는 라이센스에 대한 지불은 소프트웨어 배포 플랫폼의 하나 이상의 서버에 의해 및/또는 제3자 지불 엔티티를 통해 취급될 수 있다. 서버들은 구매자들 및/또는 라이센서들이 소프트웨어 배포 플랫폼(605)으로부터 컴퓨터 판독가능 명령어들(682)을 다운로드하는 것을 가능하게 한다. 예를 들어, 예시적인 컴퓨터 판독가능 명령어들에 대응할 수 있는 소프트웨어는 예시적인 프로세서 플랫폼(들)(700)(예를 들어, 예시적인 접속된 에지 디바이스들)에 다운로드될 수 있고, 이 프로세서 플랫폼(들)(700)은 스위치에서 콘텐츠 삽입을 구현하기 위해 컴퓨터 판독가능 명령어들(682)을 실행한다. 일부 예들에서, 소프트웨어 배포 플랫폼(705)의 하나 이상의 서버는 예시적인 컴퓨터 판독가능 명령어들(682)의 요청들 및 송신들이 통과해야만 하는 하나 이상의 보안 도메인 및/또는 보안 디바이스에 통신가능하게 접속된다. 일부 예들에서, 소프트웨어 배포 플랫폼(705)의 하나 이상의 서버는 개선, 패치, 업데이트 등이 최종 사용자 디바이스들의 소프트웨어에 배포되고 적용될 것을 보장하기 위해 소프트웨어(예를 들어, 도 6b의 예시적인 컴퓨터 판독가능 명령어들(682))에 대한 업데이트를 주기적으로 제공, 송신, 및/또는 강제한다.

[0095] 도 7의 예시된 예에서, 컴퓨터 판독가능 명령어들(682)은 소프트웨어 배포 플랫폼(705)의 스토리지 디바이스들

상에 특정 포맷으로 저장된다. 컴퓨터 판독가능 명령어들의 포맷은 특정 코드 언어(예를 들어, 자바, 자바스크립트, 파이썬, C, C#, SQL, HTML 등), 및/또는 특정 코드 상태(예를 들어, 컴파일되지 않은 코드(예를 들어, ASCII), 인터프리팅된 코드, 링크된 코드, 실행가능 코드(예를 들어, 바이너리) 등)를 포함하지만, 이에 제한되지는 않는다. 일부 예들에서, 소프트웨어 배포 플랫폼(705)에 저장된 컴퓨터 판독가능 명령어들(682)은 예시적인 프로세서 플랫폼(들)(710)에 송신될 때, 제1 포맷으로 되어 있다. 일부 예들에서, 제1 포맷은 특정 타입들의 프로세서 플랫폼(들)(710)이 실행할 수 있는 실행가능 바이너리이다. 그러나, 일부 예들에서, 제1 포맷은 예시적인 프로세서 플랫폼(들)(710) 상에서의 실행을 가능하게 하기 위해 제1 포맷을 제2 포맷으로 변환할 하나 이상의 준비 태스크를 필요로 하는 컴파일되지 않은 코드이다. 예를 들어, 수신 측 프로세서 플랫폼(들)(710)은 프로세서 플랫폼(들)(710) 상에서 실행될 수 있는 제2 포맷의 실행가능 코드를 발생시키기 위해 컴퓨터 판독가능 명령어들(682)을 제1 포맷으로 컴파일할 필요가 있을 수 있다. 또 다른 예들에서, 제1 포맷은, 프로세서 플랫폼(들)(710)에 도달할 시에, 명령어들의 실행을 용이하게 하기 위해 인터프리터에 의해 인터프리팅되는 인터프리팅된 코드이다.

[0096] 도 8은 실시예에 따른 서버리스(serverless) 데이터 센터(800)를 나타내는 블록도이다. 데이터 센터(800)는 논리적 서비스 블록들로 구성된다. 도 8에 예시된 서비스 블록들은 범용 컴퓨팅 서비스들(802), ML(Machine Learning) 및 AI(Artificial Intelligence) 서비스들(804), 계산 스토리지 서비스들(806), 및 가속 서비스들(808)을 포함한다. 서비스 블록들은 지능형 네트워크 패브릭(810)과 결합된다.

[0097] 이 타입의 서버리스 데이터 센터(800)에 의해, 오케스트레이션을 수행하는 새로운 방식이 달성될 수 있고, 이는 현재의 관행들로부터, 고객이 의도만을 표현하고 오케스트레이션 스택 자체가 그 의도를 달성하기 위해 플랫폼을 설정하는 목표 주도 접근법으로 이동한다.

[0098] 일부 기존의 SaaS(Software as a Service) 모델들은 전형적으로 그들의 소비자들에게 SLO(Service Level Objectives)의 서비스 특정적 세트를 프로모션한다. 예를 들어, 데이터 분석 플랫폼은 계산될 수 있는 시간당 다수의 작업을 프로모션할 수 있다. 이는 사용자 관점에서는 크지만, 여전히 SaaS 제공자가 요청들을 리소스 오케스트레이터에 전송하기 전에 필요한 리소스들에게 SLO들을 선제적으로 매핑할 것을 요구한다. 서비스 제공자들은 SLO들에 기초하여 리소스 타입들을 자동으로 선택할 수 없다.

[0099] 본 명세서에 설명된 시스템들 및 방법들은 의도들(목표들)이 플랫폼 설정들에 자동으로, 동적으로, 그리고 직접적으로 매핑될 수 있는 방법을 예시한다. 의도들이 상위 레벨 목표들로서 수신되고 오케스트레이션 스택 전체에 걸쳐 하위 레벨 설정들에 매핑된다.

[0100] 예를 들어, 어떤 것이 본 명세서에서 "P50 레이턴시"라고 지칭되는 시간 윈도우에서 완료되는 특정 비율이 존재한다는 애플리케이션 요건을 고려한다. 태스크들, 프로젝트들, 요청들, 또는 그와 유사한 것이 시간의 적어도 50%에서 제때에 완료할 필요가 있다는 것, 또는 태스크가 제때에 완료할 적어도 50%의 확률이 있다는 것을 나타내기 위해 "P50 타겟" 또는 "P50 완료들"과 같은, 유사할 수 있는 다른 용어들이 또한 사용될 수 있다. 이 의도는, 스레드 레벨 우선 순위를 표시하기 위해 그런 것처럼 그리고 그로부터 CPU 캐시 웨이/프로필 할당/리소스 할당으로, 하위 레벨 설정들에 매핑된다. 입력/출력(I/O) 측에서, 요청 및 응답을 송신 및 수신하기에 충분한 통신 리소스들이 이용가능하도록 보장하기 위해 의도가 네트워크 설정들에 매핑될 수 있다.

[0101] 변하는 조건들에 동적으로 적응하기 위해, 시스템들 및 방법들은 아키텍처 내의 다양한 레벨들에서 제어 루프들을 이용한다. 제어 루프들은 신용 시스템, 유틸리티/비용 함수, 계획 및 해결기(solver) 등을 채택할 수 있다. 제어 루프들은 모니터링되는 시스템이 얼마나 빨리 적응할 필요가 있는지에 좌우되어 상이한 속도들로 실행될 수 있다. 이러한 제어 루프들은 컴퓨팅 시스템들의 역동성을 고려할 때 특히 중요하고 유용하다. 예를 들어, 네트워크 트래픽은 매일 달라질 수 있다. 고객들이 리소스 요건들 자체를 변경해야만 하는 경우, 서비스 제공자를 사용하는 데 그들에게 추가적인 부담이 있다. 덧붙여, 이러한 결정을 내리는 데 필요한 정보가 고객에게 노출되지 않을 수 있기 때문에, 이러한 변경이 어려울 수 있다. 그 대신에, 본 명세서에 기술된 시스템 및 방법을 사용하여, 고객은 의도를 지정하기만 하면 되고, 시스템은 리소스들을 자동으로 조절한다. 이는 고객과 서비스 제공자 둘 모두에 대한 비용 절감으로 이어질 수 있다.

[0102] 시스템에서의 역동성은 생성되는 계획들에도 적용된다. 이러한 계획들은 시간적 요소를 갖는다. 예를 들어, 계획은 일시적으로(예를 들어, 수 분 정도) 고속 제어 루프의 감독 하에서 SLA를 따라잡기 위해 작업부하에게 풍부한 리소스들을 할당하는 것을 목표로 할 수 있고, 그 후 더 긴 기간(예를 들어, 월별)에 걸쳐 더 느린 제어 루프의 감독에 의해 수용 가능하고 알맞은 리소스 할당 패턴을 달성하려고 기대할 수 있다. 계획은, 자동으로 트리거될 수 있거나 또는 예를 들어, 인간 안내를 가능하게 하기 위해 계획을 인간 운영자에게 추천으로서 먼저

전송함으로써 부분적으로 수동으로 트리거될 수 있다.

[0103] 덧붙여, 계획의 시간적 양태는 또한 연속적인 개선 관점에서 유용할 수 있다. 시스템-오브-시스템들(system-of-systems)에서 모든 SLO들을 만족시키는 계획은 모든 SLO들을 또한 만족시키지만 상이한 셋업, 구성, 또는 정책들의 세트를 트리거하는 또 다른 계획으로 대체될 수 있다. 예를 들어, 유지보수를 준비하기 위해, SLO는 리소스를 더 효율적으로 사용하거나, 인커밍 작업부하들/서비스들을 위한 공간을 만드는 것, 또는 그와 유사한 것을 할 수 있다.

[0104] 분산형 서비스 지향 플랫폼에서, 오케스트레이션의 복잡도가 훨씬 더 증가한다. 여기 기술되는 시스템들 및 방법들은 상이한 이해 관계자들에 의해 소유될 수 있는 다중의 사이트에 걸친 오케스트레이션을 다룬다. 개개의 부분들이 무엇보다 중요한 의도를 달성하기 위해 협상 및 조정을 구현할 수 있으며, 이는 어떤 중앙집중식 오케스트레이션도 없을 때 가장 중요하다.

[0105] 구현에서, 서비스 제공자는 애플리케이션 컨텍스트 메타데이터를 갖는 컴퓨팅 유닛(예를 들어, 포드(pod))을 배치할 수 있어서, 노드가, 빠른 제어 루프에서, 보다 적절한 애플리케이션 결정들을 행하고 가능하게는 그의 엔드-투-엔드 영향 속성으로 인해 낮은 우선순위 작업부하를 탈우선순위화(deprioritize)하지 않기로 선택할 수 있도록 한다. 그러나, 이는 E2E 뷰를 요구하고, 그것은 스택의 상위 계층들에서 동작들을 수행하는 다른 계산 노드들과 함께 데이터(데이터 출처, 준수, 및 기록을 포함함)에 대한 보안 함수들을 제공해야만 하는 노드들의 공동-스케줄링 및 공동-우선순위화의 개념들을 포함한다. 쿠버네티스(Kubernetes)에서, 포드(Pod)는 당신이 생성하고 관리할 수 있는 컴퓨팅의 최소 배치가능 단위이다. 포드는 공유된 스토리지 및 네트워크 리소스들, 및 컨테이너들을 실행하는 방법에 대한 사양을 갖는 하나 이상의 컨테이너의 그룹이다. 포드의 콘텐츠들은 항상 동일 장소에 위치되고 공동-스케줄링되며, 공유된 컨텍스트에서 실행된다.

[0106] 의도 기반 시스템들로의 전환에 의해, 시스템은 배포 기능으로부터 리소스들이 잠재적으로 이용가능하게 될 수 있는지 및 리소스들을 이용가능하게 만들기 위해 시스템을 재구성하는 방법으로의 전환에 중점을 둔다. 리소스들은 동일한 카테고리에 있으면서 상이한 타입들의 것일 수 있다: 예컨대, 고성능 프로세서들(예컨대, Intel® Xeon CPU들) 대 몇 개의 감소된 계산 프로세서들(예컨대, Xeon-D들), 또는 단일의 대형 XPU 대 몇 개의 소형 XPU들. 게다가, 리소스 자체는 상이한 타입들의 서브리소스들/컴포넌트들, 예를 들어, 효율적이고 성능 좋은 코어들 등을 포함할 수 있다.

[0107] 일부 구현들에서, 시스템은 클러스터 및 인프라스트럭처 관리자들의 수중에 서비스 레벨 목표 지향 제어들을 두고 또한 가능하게는 잘못된 저레벨 구성 상세 사항들을 지정해야만 하는 부담을 제거함으로써 복잡하고 분산된 인프라스트럭처를 관리 통제해야 하는 도전 과제를 해결한다. 관리 정책 컨트롤러(Admin Policy Controller) 및 관리 정책 변환 모듈(Admin Policy Translation module)이 오케스트레이션 아키텍처에서 구현될 수 있고, 다중의 티어의 의도 주도 관리 정책들을 관리하는 워크플로가 작업부하 분배 워크플로에서 사용될 수 있다. 임페러티브 구성들에 대한 다중의 반복으로 관리 정책들을 달성하는 방법을 포함하여 관리 정책들의 폐루프 제어가 사용될 수 있다. 이는 정책들 또는 정책 비준수(policy non-compliance)를 변경하는 것에 직면하여 작업부하 배치 및 재배치에 영향을 미칠 수 있다.

[0108] 시스템은 쿠버네티스 관리자의 가정(assumption)들보다는 애플리케이션의 필요들을 반영하기 위해 쿠버네티스의 LCM(life-cycle management)을 실행하기 위한 새로운 모델을 구현할 수 있다. 애플리케이션은 (빠른 시작을 지원하기 위해) 일시적으로 배치되고, 후속하여 쿠버네티스의 새로운 인스턴스가 애플리케이션의 필요를 더 잘 반영하는 클러스터 및 노드 레벨 정책들과 함께 배치된다. 작업부하는 후속하여 새로운 클러스터로 이동된다.

[0109] CAT(Cache Allocation Technology) 및 MBA(Memory Bandwidth Allocation), 및 전력과 같은 공동 리소스들을 공유하는 현재의 방법들은 애플리케이션의 KPI들이 특정 동적 컨트롤러(예를 들어, RMD(Resource Management Daemon), DRC(Dynamic Resource Controller), Appqos 등)에 의해 모니터링되거나 또는 소진(exhaustion)을 다루기 위해 텔레메트리 인식 스케줄링(Telemetry Aware Scheduling)을 사용하는 것에 의존한다. "리소스 요청 브로커"를 오케스트레이션에 통합시키는 것은 애플리케이션 KPI(이것은 애플리케이션 자체에 의해 모니터링됨)와 독립적으로, 메시지 기반 시스템을 가질 수 있다. 브로커는 시간이 공유에서의 인자이기 때문에 플랫폼 상에 로컬 에이전트를 가질 수 있으며, 클러스터 "공유" 정책에 기초하여 로컬 판정들을 할 수 있다. 개별 포드들은 정적 계약들, "보장된 리소스 계약들", 동등한 공유 계약들, 또는 동적(입찰) 계약들에 가입할 수 있다. 경쟁 요청들이 매수/매도(bid/offer) 방식, 예를 들어, "나는 기간 N동안 리소스 Y를 위해 리소스 X를 기꺼이 포기하려고 한다"를 운영할 수 있는 로컬 브로커 에이전트에 의해 중재된다.

[0110] 컴포넌트들이 여러 리소스들에 걸쳐 분산될 때 보안 위험이 발생한다. 의도 기반 오케스트레이션에서 그리고 QoS 클래스들과 유사하게, 시스템은 사용자들이 환경에서의 보안 또는 신뢰의 특정 양태들에 대해 그들이 얼마나 "민감"한지 또는 그들이 얼마나 많은 위험을 기꺼이 수락할지를 정의하도록 허용할 수 있다. 시스템은 위험을 모니터링, 분석, 및 액세스하기 위한 오케스트레이션 제어 평면에 대한 부가의 컴포넌트들의 세트는 물론이고 평가를 정책들, 리소스 할당들 등의 세트로 변환할 수 있는 컴포넌트를 포함할 수 있다.

[0111] 도 9는 실시예에 따른, 다중의 하드웨어 시스템(902A 및 902B)을 갖는 운영 환경(900)을 예시하는 블록도이다. 운영 환경(900)은 북남쪽(예컨대, 전체 스택) 및 동서쪽(예컨대, E2E)으로 뻗어 있는 "시스템들의 시스템(system of systems)"으로 간주될 수 있다. 시스템(902A 또는 902B) 내의 각각의 계층에서, 상이한 종류의 의도가 북쪽에서 남쪽으로 또는 동쪽에서 서쪽으로 SLO들에 매핑될 수 있다. 스택 내의 다양한 계층들에서의 SLO들은 FPS(frames per second), 레이턴시, IPC(instructions per cycle) 등으로 기술될 수 있다. 기업들 또는 시스템들 사이의 SLO들은 서비스를 구성하는 개별 애플리케이션들이 전체 의도 목표들을 달성하기 위해 어떻게 상호작용할 필요가 있는지의 관점에서 기술될 수 있다. 예를 들어, E2E SLO들은 P99 레이턴시 < 100ms 요건, 프론트엔드 최대 10ms, 백엔드 5ms, 캐싱 최대 10ms 등을 포함할 수 있다. 풀 스택 SLO들 및 E2E SLO들의 목표를 달성하기 위해, 시스템은 상위 계층들로부터 하위 계층들로 정책들을 시행하고, 시스템들에 걸쳐 계층 내에서의 컴포넌트들 사이에서 조정하거나 협상한다.

[0112] SLO들, 풀 스택 또는 E2E(예를 들어, 시스템 대 시스템)는 시간에 따라 변화될 수 있고, 범위(예를 들어, 최소 및 최대 허용가능 값)로 표현될 수 있고, 특정 시간 기간 동안 범위들 밖으로의 편차 또는 분산을 허용할 수 있고, 선호되는 것으로서 또는 다른 우선순위화 방식들을 사용하여 표현될 수 있고, 또는 등등과 같이 된다. 예는 태스크가 10ms를 초과하지 않고 5ms의 바람직한 준수 레벨에서 실행되면서 60분 동안 시간의 99%를 P99 준수해야만 한다는 것일 수 있다.

[0113] 일부 구현들에서, 그들의 작업부하와 연관된 사이드카(sidecar)들의 세트가 노드 상에서 사용된다. 사이드카들은 노드, 플랫폼, 설비들 등이 정확하게 셋업되고 따라서 그들의 작업부하의 목표들이 충족될 수 있도록 확실하게 하기 위해 조정한다. 사이드카들은 특정의 설정을 모니터링하고 올바른 정책들을 시행하며, 올바른 리소스들을 할당하고, 설정들을 튜닝하여, 포드들의 의도들이 충족될 수 있도록 할 수 있고, 등등이다.

[0114] 일부 설치들에서, 작업부하에 대한 서비스 레벨 목표(Service Level Objective) 기반 입력 파라미터들이 제시될 때 오케스트레이션 시스템에 의해 판정되는 리소스 할당에 대해 차징/빌링(charging/billing) 제약조건들을 부여하는 도전 과제가 존재한다. 2개의 새로운 컴포넌트가 의도 기반 차징 및 의도 기반 빌링을 가능하게 한다: 1) 차징 가드레일 함수 및 2) SLO 계획 함수. 차징 가드레일 함수(Charging Guardrails Function)의 개념은 오케스트레이션 아키텍처에서 논리적으로 중심 점인 배치 워크플로에 도입된다. 그것은 할당된 리소스들이 사용자에게 알맞게 되도록 확실히 하기 위해 SLO 지향 리소스 계획 컴포넌트를 제어하고 안내하는 것을 담당하는 엔티티로서 작용한다. SLO 계획 함수는, 작업부하 SLA에 대한 알맞은 준수를 위한 그들의 적합성만이 아니라, 할당된 리소스에 대한 비용 기반을 고려할 필요가 있다.

[0115] 도 10은 실시예에 따른 오케스트레이션 제어 평면(1000)을 도시하는 블록도이다. 오케스트레이션 제어 평면(1000)은 API(application programming interface)(1002)를 사용하여 액세스가능하거나 구성가능하다. 설정들, 데이터, 또는 다른 정보가 데이터베이스(1004)에 저장될 수 있다. 오케스트레이션 제어 평면(1000)은 리소스 관리자들(1006), 컨트롤러들(1008), 스케줄러들(1010), 플래너(1012), 모니터(1014), CIM(continuous improvement module)(1016), 및 관측가능성 스택(1018)을 포함한다. 플래너(1012), CIM(1016), 모니터(1014), 또는 오케스트레이션 제어 평면(1000)의 다른 컴포넌트들은 지식 데이터베이스(1020) 내의 데이터에 액세스하거나 지식 데이터베이스(1020)에 데이터를 저장할 수 있다. 지식 데이터베이스(1020)는, 예를 들어, 플래너 또는 스케줄러들에 대한 입력으로서 사용되는 다양한 데이터를 포함할 수 있다. 지식 데이터베이스(1020)는 태스크들의 배치 및 리소스들의 활용을 결정하기 위해 사용될 수 있는 네트워크 토폴로지 정보를 포함할 수 있다.

[0116] 플래너(1012)는 하드와이어드 회로, 프로그래머블 하드웨어 디바이스(예를 들어, ASIC), 또는 하드웨어 플랫폼 상에서(예를 들어, 일반 CPU 상에서) 실행되는 명령어들을 사용하여 구현될 수 있다. 플래너(1012)는 의도들 또는 목표들을 SLO들에 매핑하도록 구성, 설계, 프로그래밍, 또는 달리 적응될 수 있다. 플래너(1012)는 또한 SLO들을 액션가능 계획들로 분해(break down)할 수 있다. 플래너(1012)는 SLO 요건들을 적절한 포드 사양들로 자동으로 번역 또는 매핑하기 위해 사용될 수 있는데, 이 포드 사양들은 컴퓨팅, 네트워크, 스토리지, 및 설비(예를 들어, 전력) 도메인들에 걸친 리소스 요건들, 플랫폼 피처들, 또는 정책들을 포함할 수 있다. 정책들 및 하위 레벨 목표 설정들에 대한 목표들의 매핑은 휴리스틱, ML(machine learning) 또는 AI(artificial

intelligence) 메커니즘들, 작업부하 특성화들(예를 들어, 온라인 데이터로부터 또는 오프라인 실험을 통해 또는 샌드박싱을 통해 도출됨), 또는 소유자의 시스템을 어떻게 사용할지를 안내하는 리소스 소유자들로부터의 정책들을 구현할 수 있다.

[0117] 플래너(1012)는 또 다른 플래너(1022)와 시스템-대-시스템(예를 들어, E2E)을 조정하고 잠재적으로 상이한 이해관계자들에 속하는 다중의 PoP들(points-of-presence)을 매핑하기 위해 추가로 사용될 수 있다. 다양한 타입들의 조정 및 협상 방식들이 사용될 수 있다. 예를 들어, MAUT(multiple attribute utility theory) 모델이 리소스 할당을 협상하기 위해 사용될 수 있다.

[0118] 플래너(1012)는 의도들을 액션 정책들, 시스템 설정들, 리소스 요건들, 및 그와 유사한 것으로 변환하는 것을 감독한다. 이것은 지식 데이터베이스(1020)에 저장된 통찰에 기초하여 이를 행한다. 일단 계획이 이용가능하면, 이것은 스케줄러(1010)에 의해 구현되고 시행된다.

[0119] 계획은 오케스트레이션 제어 평면(1000) 내의 다양한 레벨들에서 다중의 SLO를 사용하여 정의될 수 있다. 예를 들어, 오케스트레이션 제어 평면(1000)의 상위 레벨들 상의 SLO들은 FPS를 레귤레이트(regulate) 및 제어하기 위해 사용될 수 있고, 오케스트레이션 제어 평면(1000)의 하위 레벨들 상의 SLO들은 IPC를 레귤레이트 및 제어하기 위해 사용될 수 있다. 벤더들, 사용자들, 및 서비스 제공자들은 표준화된 포맷을 사용하여 SLO들을 정의할 수 있다. SLO들은 또한 타겟 또는 한계 값들로부터 일부 변동을 제공하는 가드레일(guardrail)들을 포함할 수 있다. 예를 들어, 가드레일은 최대 10분 동안 P95의 10% 위반을 허용할 수 있다. 가드레일들은 또한 조건부일 수 있다. 예를 들어, 가드레일은 시스템이 그 후에 P99 준수를 보증할 경우 최대 10분 동안 P95의 10% 위반을 허용할 수 있다.

[0120] 이종 컴퓨팅 셋업들 및 XPU 환경들이 증가함에 따라, 하위 레벨 오케스트레이션 및 리소스 관리자들은 아주 세분화된 레벨에서 작업부하들의 공동 스케줄링 및 공동 우선순위화를 지원하도록 구성될 수 있다.

[0121] 태스크들을 리소스들에 할당할 때, 배치 문제는 해결하기 어렵다. 클러스터 기반 스케줄링과는 달리, 이 시스템은 필요한 완료 기한에 대한 동작의 근접성에 기초하여 로컬 리소스들이 재우선순위화되는 것을 허용한다. 부하 균형 및 자동 스케일 액션들은 중앙 집중식 스케줄러로부터 구동되어야만 할 필요 없이 그래프 병목들에서 국소적으로 개시된다. 이러한 액션들은 또한 중앙 집중식 또는 계층적 스케줄러로부터의 지속성 있는 명령어들과 비교하여 시간 제한된다.

[0122] 프로세서들과 같은 일부 리소스들은 전력 사용을 최적화하는 방식으로 오케스트레이션될 수 있다. 프로세서 전력 레벨들(예를 들어, PL1, PL2, PL3 등)은 임계 전력 인출(threshold power draw)을 정의하기 위해 사용된다. 시스템 플랫폼은 이러한 PL 값들 및 다른 전력 값들(캡핑, P 상태, 언코어 주파수)을 파인 그레인드(fine-grained) 방식으로 계속 조절한다. 그러나, 이것은 성능 대 전력(performance to power)을 SLA들에 링크하는 전달 함수의 매우 시변적인 상황적 성질이 주어지면 복잡한 매핑이기 때문에, 조절들은 사전 훈련된 모델들을 통해 주도된다. 사전 훈련된 모델들은 활용들 및 전력에 걸쳐 다양한 시계열 도출된 트렌드 신호들을 입력들로서 취하는데, 이들은 일반적으로 코스 그레인드(coarse-grained) 타입들의 작업부하 믹스들(예를 들어, "AI", "gRPC 헤비", "미디어" 등)에 대해 훈련되는 모델들일 수 있다.

[0123] 고객이 함수들을 등록할 때 스토리지 액세스 의도들을 표현할 때, 이러한 등록 의도들은 스토리지 계층으로부터 함수로의 효율적인 데이터 이동을 가능하게 하기에 충분한 컨텍스트를 인프라스트럭처 계층에 제공한다. 이는 데이터를 보유하는 서버들 상에서 컴퓨팅 가속기들을 사용하여 데이터 바로 옆에서 함수를 스케줄링하거나 또는 서버들 내에서 또는 서버들에 걸쳐 효율적인 데이터 전송 메커니즘들을 활용함으로써 행해진다. 중요한 점은 이것이 서버 로케이션에 대한 특정 지식을 필요로 하거나 또는 특정 컴퓨팅 리소스들 또는 전송 메커니즘들을 프로그래밍해야 하는 함수 없이 달성된다는 것이다. 스케줄링/배치는 프로세스의 시작에서의 함수가다. 액세스 패턴들이 배치 시간에 명확하지 않을 수 있으므로, 이 시스템은 또한 데이터 국소성 및 전송률들의 런타임 평가를 구현하고, 그러한 통찰력을 이용하여 자동 리스케줄링 이벤트들, 예를 들어, 어느 함수/마이크로서비스가 어떤 종류의 데이터를 처리하는 데 양호한지를 나타내는 히스토그램들을 트리거한다.

[0124] CIM(1016)은 정책에서의 절충들 또는 변경들을 분석하면서, 현재 동작들을 보다 효율적으로 만들기 위한 옵션들을 스카우트(scout)하고, 시스템이 가까운 미래에 어떻게 거동할지를 예측하는 것을 담당한다. 이 예측은 사전적 계획을 가능하게 하는 핵심이다. 반응적(reactive) 또는 사전적(proactive) 계획은 (다양한 고속/저속 루프들을 만족시키기 위해) 다양한 시간 스케일로 사용될 수 있다.

[0125] 모니터(1014)는 관측가능성 스택(1018)으로부터 입력들을 취하고, 플래너(1012) 및 CIM(1016)을 피드(feed) 또

는 트리거하기 위해 이 정보를 사용한다. 게다가, 관측가능성 스택(1018)은 지식 데이터베이스(1020)에서의 통찰, 휴리스틱 등이 온라인 및 오프라인 학습 프로세스들을 사용하여 정확한 것을 보증하는 것을 담당한다. 오프라인 학습은 실험 기반 작업부하 특성화를 통해 달성될 수 있다. 관측가능성 스택(1018)은 수집된 데이터를 사용하는 분석을 통해 아키텍처를 자체 진화시키기 위해 훈련 데이터를 수집할 수 있다. 훈련 데이터는 모니터들에 의해 캡처된 실세계 데이터로부터 도출될 수 있다. 대안적으로, 훈련 데이터는 오프라인으로 준비되어 관측가능성 스택(1018)에게 제공될 수 있다. CIM(1016)의 사용은 연속 개선 능력들을 갖고 시스템에서 자가 적응, 자가 치유, 및 최적화를 가능하게 하는 이점을 제공한다.

[0126] 연속 개선을 제공하기 위해, 다중의 제어 루프가 구현될 수 있다. 도 11은 실시예에 따른, 오케스트레이션 시스템에서의 데이터 및 제어 흐름을 예시하는 블록도이다. 의도 기반 SLO들이 SLO 변환기(1102)에서 수신되고, SLO 변환기(1102)는 변환된 SLO들을 서비스 모니터(1104)에 피드한다. SLO 변환기(1102)는 플래너(1012)의 인스턴스이거나, 그것의 컴포넌트이거나, 또는 그것을 포함할 수 있다. 서비스 모니터(1104)는 모니터(1014)의 인스턴스이거나, 그것의 컴포넌트이거나, 그것을 포함할 수 있다.

[0127] SLO 변환기(1102)는 규칙들 및 사전들의 데이터베이스(예를 들어, 지식 데이터베이스(1020))를 사용하여 의도 기반 SLO를 3개의 양태: 1) 모니터링 파라미터들, 2) 구성 파라미터들, 및 3) 시간 도메인 파라미터들에 매핑할 수 있다.

[0128] 모니터링 파라미터들은 서비스들, 태스크들, 또는 작업들을 모니터링할 때 서비스 모니터(1104)에 의해 사용된다. 모니터링 파라미터들은 서비스 모니터(1104)가 동작들을 능동적으로 모니터링하기 위한 일련의 유연한 가드레일들 및 요구되는 텔레메트리를 포함할 수 있다.

[0129] 가드레일들은 시간 제한적(time-bounded) 및 범위 제한적(range-bounded) 유연성을 제공할 수 있으며, 따라서 이들은 컨텍스트에 민감하고 상황 적응적이 된다. 따라서, 오케스트레이터의 함수에 대해 적용되는 가드레일들은 매우 느리게 그러나 매끄럽게 시프트하여, 일련의 국소화된 여유를 이룸으로써 바람직한 엔드-투-엔드 목표를 최대화하는 것을 허용할 수 있다. 이 아이디어는 3가지 중요한 유연성을 달성한다: a) P99.9 레이턴시와 같은 가장 강한 제약조건을 취하고, 그 제약조건이 고정된 임계값으로부터 임계값 범위(즉, 시간상 짧은 지속기간 동안 +10 퍼센트)로 이동하여, 나중에 연장된 시간 기간 동안 훨씬 더 나은 P99.9 레이턴시만큼 보상되도록 허용하는 것, b) 한 번에 리소스 할당에서 가능하게는 비용이 많이 드는 조절을 취하고, 그것을 제한된 양의 시간만큼 벗어나게 이동시키며, 그 사이에, 만족스러운 해결책들이 계속 실행될 수 있도록 가드레일들을 부드럽게 해주는 것, 및 c) 수리 및 후속 정상 동작들이 픽업할 수 있을 때까지 리소스들이 보다 유연하게 공유될 것을 필요로 하는 데이터 센터의 다른 부분들에서 과도적 장애들을 다루기 위해 응급 요구, 수요 등에 대한 보다 풍부한 시스템 응답을 허용하는 것.

[0130] 가드레일들에 대한 현재의 방법들은 가드레일 크로싱(guardrail crossing) 및 후속 교정을 평가하기 위해 사전 정의된 기간에 걸쳐 임계화 및 통합을 이용하는 것을 수반한다. 가드레일들이 "상태유지(stateful)" 또는 "컨텍스트(context)" 기반이 되도록 허용하는 것은 강한 제약들이 특정 조건들 동안 부드럽게 되도록 허용한다. 예를 들어, CPU 활용은 작업부하가 데이터를 캐싱함에 따라 작업부하의 제1 배치에서 최고가 되도록 허용될 수 있지만, 그 시간 후에, 정상 동작에서, 과도한 CPU 활용은 가드레일 크로싱을 트리거할 수 있다. 시스템은 가드레일들을 구현하기 위한 컨텍스트 정보로서 라이프 사이클 스테이징, 인서비스 업그레이드, HA 장애 조치, 서비스 메시 재시작 등을 포함하는 컨텍스트 지식을 추가한다.

[0131] 구성 파라미터들은 오케스트레이터들 및 리소스 관리자들에 의해 컴퓨팅, 네트워크, 메모리, 및 기타 리소스들을 구성하기 위해 사용된다. 구성 파라미터들은 일련의 오케스트레이션 목표들로서 표현될 수 있는데, 이것들은 최고 레벨 오케스트레이션 시스템에 피드되고 오케스트레이션 스택의 각각의 하위 계층에 의해 일련의 서브 목표들로 전환된다. 오케스트레이션 스택의 하단에는 파인 그레인드 제어로 튜닝될 수 있는 물리적 하드웨어 컴포넌트들이 있다. 이와 같이, 일련의 리소스 컨트롤러들이 컴퓨팅 플랫폼 상에서 정책들을 시행할 수 있다. 컴퓨팅 플랫폼은 CPU, GPU, FPGA, 가속기, IPU, 또는 다른 타입의 처리 디바이스일 수 있다.

[0132] 시간 도메인 파라미터들이 모니터링 사이클들 및 구성 파라미터들에 대한 변경들이 얼마나 자주 이루어지는지를 설정하도록 제어 루프들을 구성하기 위해 사용된다. SLO 변환기(1102)는 비실시간 모니터링으로부터 실시간 모니터링까지의 범위에 있는, SLO 모니터링을 위한 시간 도메인들을 생성한다. 시간 도메인들은 대응하는 SLO에 대한 엄격한 모니터링 및 오케스트레이션 피드백 응답 시간들을 지정한다. 시간 도메인들은 도 11에서 주관적인 용어로 "더 느린", "느린", 및 "더 빠른" 것으로서 도시되지만, SLO 변환기(1102)에 의해 시간 도메인에 매핑되는 요건들에 좌우되어, 마이크로초, 시간, 일, 또는 그와 유사한 것과 같은 임의의 시간 측정으로 특정될

수 있다. 이들 시간 도메인 파라미터들은 고정되거나, 자동으로 업데이트되거나, 또는 별개로 구성가능할 수 있다.

[0133] 서비스 모니터(1104)는 E2E 텔레메트리(1106), 비실시간 텔레메트리(1108), 거의 실시간 텔레메트리 또는 실시간 텔레메트리(1110)를 모니터링하는 계층들을 갖는다. 각각의 계층은 상이한 시간 도메인 파라미터들을 가질 수 있는 대응하는 제어 루프를 갖는다.

[0134] SLO 변환기(1102)는 의도들을 E2E에 대한 "서비스 레벨 모니터링", 느린 리소스 모니터링, 및 빠른 리소스 모니터링으로 변환한다. 규칙, 정책, 및 학습된 통찰에 기초하여, SLO 변환기(1102)는 의도의 타입, 요구되는 반응 속도, 또는 기타의 요건에 기초하여 인스턴스화될 수 있는 하나 이상의 서비스 모니터에 의도를 매핑한다. SLO 변환기(1102)는 "물리적 SLA들" 또는 "물리적 SLO들"이 경계들을 벗어날 때 오케스트레이션 스택 내의 엔티티들에게 통지들을 제공하도록 서비스 모니터들을 구성한다. 서비스 모니터들은 수동 또는 능동 프로브들, SDN(software defined networking) 컨트롤러들, 또는 SDN 분석 시스템들을 이용하여 E2E SLA들을 모니터링하는 고전적인 서비스 보증 솔루션들을 포함할 수 있다. 매핑된 SLO에 의해 요구되는 응답 시간을 달성하기 위해, 필요에 따라, 더 빠른 서비스 모니터들이 플랫폼들 상에 함께 위치될 수 있다.

[0135] 시스템은, 예를 들어, 컴포넌트들의 배치에 대한 스마트 관측가능성, 및 정확한 로케이션에서 모니터링을 자동 주입하는 방법을 주입하는 방법들을 채택할 수 있다. 컴포넌트들의 배치 후에, 시스템은 자동 SLW 교정을 달성하는 제어 루프의 일부를 형성할 수 있다. 교정을 위한 경우가 있을 때, 시스템은 효과적이고 또한 부주의로 다른 서비스들에 영향을 주지 않는 정정 조치를 제공하기 위해 충분한 제어들이 준비되도록 보장한다.

[0136] 도 11에 도시된 바와 같이, 의도 기반 오케스트레이션은 오케스트레이터들(1112-1114)의 계층 구조를 이용하여 구현될 수 있다. 표준 오케스트레이터는 사용자가 애플리케이션을 컴포넌트들의 세트 및 그 컴포넌트들을 플랫폼들 상에 랜딩시키기 위한 요건들로서 기술하도록 허용한다. 계층적 오케스트레이션은 문제가 분해되고 부분들로 분산되는 것을 허용하기 위해 사용될 수 있으며, 여기서 서브 오케스트레이터는 하나의 노드들의 서브세트 상에서 애플리케이션의 서브세트를 스케줄링하는 것을 담당하고, 또 다른 서브 오케스트레이터는 상이한 노드들의 서브세트 상에서 상이한 애플리케이션의 서브세트를 스케줄링하는 것을 담당한다.

[0137] 표준 임페러티브 오케스트레이션과는 대조적으로, 사용자가, 컴포넌트의 요건들의 일부로서, 의도를 기술할 수 있게 함으로써 의도 기반 오케스트레이션이 가능하게 될 수 있다. 이것은 사용자가 원하는 결과를 선언하고 있는 선언 메커니즘이다. 따라서, (임페러티브 메커니즘에서와 같이) 특정 규칙들 또는 파라미터들을 표현하는 대신에, 사용자는 원하는 결과를 표현할 수 있다. 한 예는 어떤 주어진 레벨의 가용성 또는 최대 처리 레이턴시를 달성하는 것일 수 있고, 이를 위해 다중 인스턴스가 그 의도를 달성하기 위해 특정한 특성들을 갖는 노드들 상에 배치될 수 있다. 의도는 선언적 표현이다.

[0138] 의도 기반 오케스트레이션을 표준 오케스트레이터의 스케줄러에 깊게 통합하기보다는, 의도 기반 스케줄링 알고리즘이 서브 오케스트레이터에 배치될 수 있다. 이 경우, 최상위 레벨 표준 오케스트레이터가 애플리케이션 기술을 수신할 때, 이것은 애플리케이션의 하나 이상의 컴포넌트에 대해 지정된 의도를 볼 수 있다. 이것은 이들 컴포넌트들의 스케줄링을 행하기 위해 의도 기반 서브 오케스트레이터에게 요청하는 것을 선택할 수 있다. 각각의 의도 기반 오케스트레이터 또는 서브 오케스트레이터는 특정 종류의 의도를 충족시키는데 전문화될 수 있다. 의도 기반 서브 오케스트레이터는 문제를 다른 서브 오케스트레이터들로 더 분해할 수 있다.

[0139] 예를 들어, 인제스트(ingest) 단계, 처리 단계, 및 액추에이션 단계로 구성되는 비디오 분석 파이프라인을 고려한다. 전체적 응용은 각각의 카메라에 대한 인제스트 컴포넌트, 100ms 이하의 레이턴시의 의도를 갖는 처리 단계, 및 각각의 액추에이터에 대한 액추에이션 컴포넌트로서 기술될 수 있다. 최상위 레벨 오케스트레이터는 인제스천(ingestion) 및 액추에이션 컴포넌트 배치들을 취급할 수 있다. 처리는 부하 균형을 맞추고 원하는 레이턴시를 달성하기 위해 얼마나 많은 처리 컴포넌트들이 필요한지를 결정할 수 있는 의도 기반 오케스트레이터에 넘겨질 수 있다. 의도 기반 오케스트레이터는 심지어 태스크를 추가적인 서브 오케스트레이터들로 세분할 수 있으며, 따라서 다중의 노드들의 클러스터가 의도를 달성하는 데 (또는 아마도 클러스터 레벨에서 높은 가용성의 추가적인 의도를 가능하게 하는 데) 사용된다.

[0140] 이 접근법의 몇 가지 이점이 있다. 첫째, 기존의 오케스트레이터들에서의 기존의 스케줄링 알고리즘들의 복잡한 의사 결정을 의도 기반 오케스트레이션의 똑같이 복잡한 의사 결정과 병합할 필요가 없다. 각각은 문제의 핵심 부분들에 대해 사용될 수 있다. 그에 부가하여, 분산 결정을 하는 것은 결정이 처리에 가깝게 푸시되도록 허용한다. 이는 더 빠른 반응성을 허용할 수 있으며, 이는 산업적 사용 사례들에서 빠른 제어 루프들과 같은

것들을 가능하게 하는 것을 도울 것이다.

[0141] 다양한 실시예에서, 최상위 레벨 오케스트레이터는 고객으로부터 의도들을 수신하고, 의도 기반 서브 오케스트레이션이 요구되는 때를 식별하고, 최상위 레벨 오케스트레이터와 서브 오케스트레이터들 사이의 상호작용들을 정의하도록 구성, 프로그래밍, 또는 달리 적응된다. 최상위 레벨 오케스트레이터는 표준 오케스트레이터일 수 있고, 서브 오케스트레이터들은 도 10에 설명된 것과 같은 의도 기반 오케스트레이터일 수 있다. 오케스트레이터들의 계층 구조를 사용함으로써, SLA가 상위 레벨 오케스트레이터와 서브 오케스트레이터 사이에 합의된 경우 문제가 해결된다. 서브 오케스트레이터는 그것이 더 이상 SLA를 충족시킬 수 없을 때를 요청 측 오케스트레이터에 표시할 수 있다.

[0142] 이러한 오케스트레이터들의 조직을 달성하기 위해, 의도들은 표준 오케스트레이터들과 호환되는 방식으로 표현되어야 하며, 이러한 표준 오케스트레이터들은 의도 기반 서브 오케스트레이션이 요구될 때를 식별할 수 있어야 한다. 요청 측 오케스트레이터와 요청 측 오케스트레이터를 충족시키기 위해 사용되는 서브 오케스트레이터들 사이에 프로토콜이 사용될 수 있다.

[0143] 게다가, 애플리케이션이, 개별적으로 오케스트레이션되는 컴포넌트들로 스플릿될 때, 이들은 전체적인 오케스트레이션에 영향을 미치는 순서화 의존관계들을 가질 수 있다. 이러한 순서화 의존관계들이 존재하는 경우, 이들은 추상적 용어로 기술될 수 있다. 예를 들어, 생산자-소비자 흐름의 사용 사례에서, 생산 컴포넌트는 바람직하게는 X 단위의 데이터, 이벤트들, 프레임들 등을 소비 컴포넌트의 앞에 유지하는 것으로서 지정될 수 있다. 따라서, 각각의 컴포넌트에 대한 서브 오케스트레이터들은 리소스들의 할당에 대한 조건부 책임들을 질 수 있다(소비자는 시간 T0+delta에서보다 시간 T0에서 더 적은 리소스들을 필요로 하는 한편, 생산자는 시간 T0에서보다 시간 T0-delta에서 더 많은 리소스들을 필요로 한다). 이러한 "리소스 흐름들"은 서브 오케스트레이터들 사이에서 타이트하게 조정되게 되며, 따라서 본 예에서의 생산자 및 소비자는 집합적으로 CPU, 캐시, 메모리 대역폭 등의 X 분수(X-fraction)를 얻지만, 여기서 리소스들은 그들 사이의 막후 조종된 공유(choreographed sharing)에 따라 끊김 없이 흐른다. 유사하게, 우선순위 반전들이 방지될 필요가 있다; 따라서, 예를 들어, 소비자가 생산자가 그만큼 자신의 앞에 있는 거리를 따라잡고 감소시키려고 열심히 시도하고 있기 때문에, 생산자가 더 낮은 우선 순위를 가질 수 있지만, 소비자가 거리를 너무 빠르게 계속해서 줄여서 생산자가 이제 앞에 머무르려고 노력해야만 하는 경우, 스케줄링 소프트웨어의 끼어드는 튜닝을 요구하는 것 대신에, 우선 순위들이 거리의 함수로서 그들 사이에 빠르게 흐르게 하는 것이 합리적이다.

[0144] 요청된 의도/목표들을 달성하기 위해, 가속 기술(FPGA들, GPU들 등)뿐만 아니라 전체 셋업(예를 들어, DRC)을 지원하는 하드웨어 피처들에 대한 요구가 구현될 수 있다. 종종, 가속기들 및 XPU들은 CPU 코드 개발자가 명령어들을 통해 직접적으로 제어하지는 않는 동작들을 실행한다. 따라서, 가속기 또는 XPU에서의 특정 하드웨어 특권으로, 또는 스택의 상위 레벨들에서의 통상의 소프트웨어에 불투명한 일부 방식으로 수행되는 민감한 동작들은, 그렇지 않았더라면 보안 필터링 동작들의 공동 스케줄링을 요구했을 보안 경계 제약들을 단순화할 수 있다. 게다가, 하위 레벨 피처들은 고속 제어 루프들에서 발생하고 있는 것을 다양한 제어 루프들에게 알릴 필요가 있다. 이러한 타입의 보고는 추적, 로깅, 및 모니터링을 위해 관측가능성 프레임워크의 확장/사용을 필요로 한다.

[0145] 도 12는 실시예에 따른, 의도 기반 오케스트레이션을 구현하는 방법(1200)을 예시하는 흐름도이다. (1202)에서, 태스크의 실행을 위한 의도 기반 SLO(service level objective)가 오케스트레이터에서 수신된다.

[0146] (1204)에서, SLO는 복수의 정책에 매핑된다. 매핑은 정적 맵에 기초할 수 있다. 대안적으로, 매핑은 휴리스틱 또는 다른 지능적 메커니즘들을 사용하여 수행될 수 있다.

[0147] (1206)에서, 복수의 정책이 복수의 서브 오케스트레이터에 분배되고, 복수의 서브 오케스트레이터 각각은 태스크의 일부분의 실행을 관리한다. 정책들은 타입별로, 리소스별로, 또는 다른 인자들별로 그룹화되거나 분리될 수 있다. 서브 오케스트레이터들은 리소스들의 그룹, 리소스의 타입, 특정 노드 또는 노드들의 세트, 또는 그와 유사한 것을 담당할 수 있다.

[0148] (1208)에서, 태스크의 실행이 모니터링된다. 태스크 모니터링은 관심 있는 KPI들을 정의하고 그 후 그 KPI들에 대해 데이터를 되풀이해서 획득함으로써 수행될 수 있다.

[0149] (1210)에서, 모니터링에 기초하여 교정 동작이 개시된다. 교정 동작들은 마이그레이션, 리소스 할당 또는 할당 해제, 프로세스의 재시작 또는 중단, 컨테이너, 포드, 또는 마이크로서비스, 또는 그와 유사한 것과 같은 동작들을 포함할 수 있다.

[0150] 기존의 오케스트레이션된 분산 컴퓨팅 환경들에서, 데이터 이동은 병목현상 문제이다. 분산 환경에서 실행되는 함수들은 제거 블록 스토리지 서비스들에서 저장될 수 있는 데이터에 액세스한다. 수많은 효율적인 데이터 전송 옵션들이 통상적으로 이용가능하지만, 함수가 실행될 곳의 로케이션이 고정되어 있지 않기 때문에, 스토리지에 효율적으로 액세스하는 것은 부담스러운 일이 된다. 상이한 서버들이 상이한 능력들을 가질 수 있기 때문에, 정상 IP를 통한 네트워크 전송의 최소 공통 분모가 사용된다. 데이터에 가깝게 함수를 실행하는 대안은 실현가능하지 않은데, 그 이유는, 설계에 의해, 데이터 로케이션이 함수로부터 멀어져 추상화되기 때문이다. 또한, 서버의 리소스들은 함수로부터 숨겨진다. 이와 같이, 계산 스토리지가 활용될 수 없다.

[0151] 기존의 아키텍처들은 데이터가 블록 스토리지 서버 내에서 또는 블록 스토리지 서버에 근접하여 처리되는 계산 스토리지를 사용하지 않는다. 그 대신에, 데이터가 먼저 블록 스토리지 서버로부터 (예컨대, EC2와 같은) 컴퓨팅 서버, (예컨대, S3과 같은) 객체 스토리지 서버, 또는 (예컨대, AWS(Amazon Web Services) AQUA와 같은) 중간 단기 계층(intermediate ephemeral layer)으로 복사되고, 여기서 데이터가 처리될 수 있다. 설계에 의해, 스토리지 서비스 클러스터에서의 데이터의 실제 로케이션은 함수로부터 떨어져 추상화된다. 따라서, 블록 스토리지 레벨에 함수들을 배치하기 위한 추가적인 메커니즘들을 추가하지 않고서는 계산 스토리지가 활용될 수 없다.

[0152] 현재의 접근법은 함수들이 가능한 한 일반적인 방식으로 원격 객체 스토리지 서비스들로부터의 데이터에 액세스하는 것이다. 이것은 클러스터 내의 임의의 서버로 스케일링 아웃하는 것을 가능하게 한다. 그러나, 이것은 큰 레이턴시들 또는 추가의 네트워크 홉들을 갖는 스토리지 서비스로부터의 비효율적인 데이터 전송들의 대가를 치루고 이뤄진다. 스토리지 서비스들에서 지속되는 데이터는 FaaS 함수들이 배치되는 컴퓨팅 클러스터 내의 서버들과 구별되는 스토리지 클러스터의 일부인 서버들에 전형적으로 상주한다. 결과적으로, 데이터가 FaaS 함수에 의해 판독 또는 기입될 때, 그것은 2개의 클러스터 사이에서 네트워크를 통해 앞뒤로 출하될 필요가 있다. FaaS 함수들의 온 디맨드 오토 스케일링 능력은, 특히 큰 훈련 데이터 세트들에 걸쳐 동작할 필요가 있는 AI/ML 파이프라인들에 대해 매우 매력적이지만, 별개의 컴퓨팅 및 스토리지 클러스터들 사이의 데이터 이동은 매우 비효율적이다. 데이터 이동은 전체 애플리케이션에 상당한 레이턴시를 추가하고 많은 양의 네트워크 대역폭을 소비한다.

[0153] 일부 기존 구현들이 이러한 문제들을 해결하려고 시도하지만, 단기 스토리지 계층을 생성하는 것은 이것이 스토리지와 컴퓨팅 클러스터들 사이에 고가의 서버들을 제공하는 것을 수반하기 때문에 매우 비용이 많이 든다. 이것은 또한, 데이터가 (데이터가 지속되는) 스토리지 서비스로부터 단기 계층으로 미리 이동될 것을 요구한다. 다른 구현들은 CS(computational storage) 커맨드들을 실행하기 위해 데이터 로케이션에 대한 지식을 요구하며, 이는 데이터가 분산 스토리지 서비스에 존재하는 경우 최종 애플리케이션들이 CS를 활용하는 것을 막는다. 또한, 기존의 객체 스토리지 서비스들(예를 들어, 아마존 S3)이 사용자들이 객체 스토리지 시스템에서 쿼리들(예를 들어, 아마존의 FaaS 람다 기술로 구현되는 아마존의 S3 선택)을 실행하거나 또는 애플리케이션(예를 들어, 아마존의 S3 객체 람다)으로 리턴하기 전에 S3 GET 요청으로부터 리턴된 데이터를 변환하는 것을 허용하지만, 그러한 계산 서비스들은 객체 레벨에서 계산을 종료하고 블록 레벨에서 계산 스토리지를 제공하지 않는다. 함수들에 대한 보다 효율적인 데이터 액세스를 제공하는 시스템이 필요하다.

[0154] 본 명세서에 기술된 의도 기반 오케스트레이션 환경에서, 시스템은 데이터가 존재하는 곳과 동일한 노드 근방에서 또는 그 노드에서 함수의 실행을 지능적으로 스케줄링할 수 있다. 사용자는 함수들을 등록할 때 스토리지 액세스 의도들을 표현할 수 있다. 이들 등록 의도들은 스토리지 계층과 함수 사이의 효율적인 데이터 이동을 가능하게 하기 위해 인프라스트럭처 계층에 충분한 컨텍스트를 제공한다. 이는 데이터를 보유하는 서버들 상에서 컴퓨팅 가속기들을 사용하여 데이터 바로 옆에서 함수를 스케줄링하거나 또는 서버들 내에서 또는 서버들에 걸쳐 효율적인 데이터 전송 메커니즘들을 활용함으로써 행해진다. 중요한 점은 이것이 서버 로케이션에 대한 특정 지식을 필요로 하거나 또는 특정 컴퓨팅 리소스들 또는 전송 메커니즘들을 프로그래밍해야 하는 함수 없이 달성된다는 것이다. 스케줄링 및 배치는 프로세스의 시작에서의 함수이다. 액세스 패턴들이 배치 시간에 명백하지 않을 수 있기 때문에, 시스템은 또한 데이터 국소석 및 전송 레이트들의 런타임 평가를 포함하고 자동 리스케줄링 이벤트들을 트리거하기 위해 그 통찰력을 사용한다. 예를 들어, 어느 함수/마이크로서비스가 어느 종류의 데이터를 처리하는 데 좋은지를 보여주는 히스토그램들이 개발될 수 있다. 일부 설명들이 함수들을 논의할 수 있지만, 마이크로서비스들은 본 개시내용에 의해 포괄되고 함수 스케줄링 및 오케스트레이션 대신에 또는 함수 스케줄링 및 오케스트레이션과 함께 사용될 수 있다는 점이 이해된다.

[0155] 스토리지 액세스 의도들을 갖는 의도 기반 오케스트레이션을 구현함으로써 몇 가지 장점들이 존재한다. 시스템은 스케일 아웃을 가능하게 하는 일반적 함수 구현들을 유지하지만, 더 효율적인 데이터 전송들을 생성한다.

시스템은 또한 스토리지 서비스들과 FaaS 함수들 사이에서 데이터를 앞뒤로 이동시킬 필요성을 감소시키고, 그에 의해 네트워크 대역폭 소비를 감소시킨다. 시스템은 또한 최종 사용자 애플리케이션으로부터 수많은 가능한 계산 스토리지 구현 특정적 상세사항들을 추상화하고 애플리케이션들에 대한 프로그래밍 부담을 감소시킨다. 시스템은 최종 사용자 애플리케이션들이 분산형 스토리지에 지속된 데이터에 대한 계산 스토리지를 이용하는 것을 허용하며, 이는 CS 커맨드를 개시하기 위한 데이터의 로케이션을 알 필요성으로 인해 이전에는 가능하지 않았다. 시스템은 또한, CS 특정적 상세사항들을 사용자에게 투명하게 함으로써 안전한 방식으로 데이터 부근에서 사용자 정의된 함수들을 실행하는 능력을 생성한다. 데이터 로케이션은 사용자 정의된 함수에 결코 노출되지 않는다. 더욱이, 사용자 크리덴셜들을 제공함으로써 액세스가능한 데이터만이 사용자 정의된 함수에 제시된다. 시스템은 또한 데이터 액세스 패턴들 및 속성들을 변경하는 것에 대해 동적으로 적응한다.

[0156] FaaS(Function-as-a-Service)는 사용자들이 함수(지원된 프로그래밍 언어로 함수 소유자에 의해 작성됨) 및 함수가 언제 기동되어야 하는지를 표시하는 이벤트 트리거를 등록할 수 있는 클라우드 컴퓨팅 서비스의 타입이다. 논의의 나머지에서, 함수의 개념은 현대의 클라우드 네이티브 서비스들에서 액세스된 데이터가 어떻게 의도 주도 모델을 따를 수 있는지에 대한 예로서 사용된다. 그러나, 메커니즘들은 함수들에 제한되지 않고, 여기서 설명되는 시스템들은 마이크로서비스 스타일 배치들을 위해 또한 사용될 수 있다는 점이 이해된다. 서비스 제공자는 함수를 배치하는 모든 양태들(서버 로케이션, 부하 기반 스케일링 등)을 관리한다. 제공된 함수들은 전형적으로 FaaS 프레임워크 코드 내에 랩핑되고 데이터 센터 내의 적절한 서버 상의 컨테이너들에 배치된다. FaaS 함수들이 동작하는 입력/출력 데이터는 전형적으로 객체 또는 파일 스토리지 서비스들을 포함하는 다른 클라우드 서비스들로부터 액세스되어야만 한다. 전형적으로, 함수 코드는 데이터를 판독 또는 기입하기 위해 이러한 서비스들에 대한 관련 API 호출들을 기동한다. 데이터 이동은, 스토리지 서비스들에 대한 높은 레이턴시들 및 큰 네트워크 대역폭 소비로 인해, FaaS 배치들에서 널리 문서화된 문제이다. 인프라스트럭처는 FaaS 함수 내로부터의 데이터 액세스들에 대한 지식을 갖지 않는다. 반대로, FaaS 함수는 이러한 상세사항들이 스토리지 서비스에 의해 추상화되기 때문에 데이터 로케이션에 대한 지식을 갖지 않는다. 결과적으로. 배치된 함수들은 데이터 스토리지 서비스들을 제공하는 서버들과는 구별되는 컴퓨팅 클러스터들 내의 서버들 상에서 종종 실행된다.

[0157] 본 시스템은, 동일한 것에 대해 어떻게 오케스트레이션할지에 대한 메커니즘들 및 의도들/목표들이 충족되고 있는 것을 보장하기 위한 방식들을 포함하여, 스토리지 관련 활동들에 대한 의도 선언들을 제공한다.

[0158] 사용자 특정된 스토리지 액세스 의도는 새로운 타입의 데이터이다. 사용자 특정된 스토리지 액세스 의도에 의해, 사용자는 등록된 함수가 저장된 데이터에 대해 필터링, 변환, 또는 다른 동작들을 수행하는 것을 나타낸다. 사용자가 그들이 등록하는 함수들에 대해 표현할 수 있는 수많은 타입의 스토리지 액세스 의도가 있다. 이 의도들은 오케스트레이션 인프라스트럭처에게 스토리지 서비스들 내에서 사용되는 함수 실행 및 데이터 액세스 메커니즘들의 다양한 양태들을 오케스트레이션하는 정보를 제공한다.

[0159] 스토리지 액세스 의도들의 비제한적인 리스트는 다음을 포함한다: 1) 데이터 처리 QoS: 함수는 데이터에 대해 동작하고 레이턴시 한계 내에서 완료되어야 한다; 2) 데이터 액세스 QoS: 함수는 스토리지에 액세스하고, 처리 QoS 요건을 충족시키기 위해 특정 대역폭 및 레이턴시 한계들로 그렇게 해야 한다; 3) 데이터 전송 QoS: 함수는 데이터를 전송하고 사용자는 데이터 처리 및 데이터 액세스 QoS 요건들을 충족시키기 위해 데이터 전송 속도 요건들을 특정한다; 및 4) 데이터 복원성 QoS: ECC 메모리로 처리하고 RAS(Reliability Availability Serviceability)와 같이 가능해진 메트릭들을 가짐.

[0160] 또한, 고려해야 할 임페러티브들이 있다. 비 제한적인 리스트는 다음을 포함한다: 1) 데이터 어피니티: 함수는 데이터 근처에서 실행되어야 한다; 2) 데이터 경계: 데이터는 함수가 데이터 경계 내에서 이동해야만 하는 프라이버시 요건 또는 보안으로 인해 주어진 관리적 또는 지리적 경계를 넘을 수 없다; 3) 데이터 중력: 데이터 이동들이 데이터 관련 스킴들을 계산하고 데이터 관련 스킴들로의 이동을 계산할 수 있게 한다; 4) 데이터 프라이버시/보안; 및 5) 데이터 복원성.

[0161] 데이터 어피니티 임페러티브(data affinity imperative)는 스케줄링 및 오케스트레이션 동안 사용된다. FaaS(FaaS-CS 함수)의 "계산 스토리지 함수"는 함수가 충족할 수 있는 가능한 의도를 위해 가드레일들을 지정하기 위해 함수 소유자에 의해 사용될 수 있다. 따라서, 실제 함수 코드 및 이벤트 트리거 외에도, 함수 소유자는 스토리지 서비스 내의 데이터 컨테이너(들) URL(들) 및 임의의 필요한 액세스 크리덴셜들을 제공한다. 데이터 컨테이너 URL은 데이터의 실제 로케이션을 제공하지 않고 데이터에 액세스하는 방법을 제공하는 논리적 추상화 계층이다. 오케스트레이션 시스템은 데이터의 물리적 로케이션을 결정하고 데이터와 동일한 로케이션에 또

는 그 근방에 FaaS-CS 함수를 스케줄링하기 위해 데이터 컨테이너 URL을 사용할 수 있다.

[0162] FaaS-CS 패키지는 <f, I, O, C, E>로서 보여질 수 있고, 여기서 f는 사용자 정의된 함수 코드이고, I는 입력 데이터 컨테이너 URL(예를 들어, AWS(Amazon Web Services) S3과 같은 객체 스토리지 서비스 내의 버킷)이고, O는 선택적 출력 컨테이너 URL(AWS S3 버킷일 수 있음)이고, C는 데이터(예를 들어, 키)에 액세스하기 위한 사용자 크리덴셜들이고, E는 함수에 대한 이벤트 트리거이다.

[0163] 새로운 인프라스트럭처 능력들은 FaaS-CS 함수의 등록에서 제공되는 의도를 활용한다. 로케이터 서비스는 주어진 URL의 실제 서버 및 데이터 로케이션을 결정하기 위해 스토리지 서비스들과 인터페이스할 수 있다. FaaS 인프라스트럭처는 로케이터를 사용하여 데이터가 어디에 지속되는지를 찾고 데이터를 보유하는 서버 상에서 또는 그 근처에서 FaaS-CS 함수의 실행을 트리거할 수 있다. 이 점에 있어서, FaaS-CS 함수에 대한 데이터 어피니티의 사용자 의도는 데이터 근처에서의 함수의 실행을 오케스트레이션하기 위해 사용될 수 있다.

[0164] 도 13은 실시예에 따른, 계산 스토리지를 오케스트레이션하기 위한 데이터 및 제어 흐름을 도시하는 블록도이다. 사용자(1302)는 게이트웨이(1304)에서 작업부하를 등록하기 위해 작업부하 패키지를 송신할 수 있다. 작업부하 패키지는 함수 코드, 데이터 컨테이너 URL, 선택적인 출력 로케이션, 사용자 크리덴셜들(필요한 경우), 및 함수 코드 실행을 트리거하기 위해 이용되는 이벤트 트리거를 포함한다. 작업부하 패키지는 FaaS-CS 패키지의 형식일 수 있다. 또한, 사용자는, 예를 들어, 함수 소유자의 의도 가드레일들에 따라 함수 실행 시간 상에 바인딩된 레이턴시를 특정하는 데이터 처리 QoS 의도를 포함하는 하나 이상의 스토리지 액세스 의도를 표현할 수 있다.

[0165] 게이트웨이(1304)는 특정 구성 파라미터들로 특정 서버에서 실행하도록 함수(예를 들어, 작업부하 컨테이너(1306))를 인스턴스화, 스케줄링, 또는 구성하기 위한 오케스트레이터로서 작용할 수 있다. 분산 컴퓨팅 환경은, 상이한 타입의 CPU, GPU, FPGA, 가속기들, 계산 드라이브들 등과 같은, 함수가 실행될 수 있는 수많은 이용가능한 컴퓨팅 리소스들을 가질 수 있다. 애플리케이션 프로그래밍 인터페이스(예를 들어, Intel®로부터의 oneAPI)는 이러한 이질성을 다루기 위해 사용될 수 있다. 함수가 API 추상화를 호출하는 경우, 게이트웨이(1304)는 임의의 원하는 리소스 상에서 함수 실행을 스케줄링할 수 있다. 게이트웨이(1304)는 각각의 실행가능한 리소스 타입에 대해 함수 실행 시간들에 대한 타이밍 통계를 유지할 수 있다. 실행가능한 컴퓨팅 리소스 상의 실제 부하가 오버로드되는 경우, 함수가 근방의 이용가능한 리소스에 초기에 스케줄링되고 데이터가 관련 서버로 전송될 수 있다.

[0166] 게이트웨이(1304)는 이벤트 트리거를 스토리지 서비스(1308)(예를 들어, MinIO, Inc.로부터의 MinIO)에 등록한다. 이벤트 트리거는 데이터가 작업부하 패키지에서의 URL에 의해 식별된 버킷에 추가될 때 통지를 요청할 수 있다. 스토리지 서비스(1308)는 데이터 버킷을 모니터링하고, 변경이 버킷에서 검출될 때, URL을 로케이션 서비스(1310)에 송신한다. 로케이션 서비스(1310)는 제공된 URL에 기초하여 데이터의 물리적 로케이션을 결정하기 위해 룩업을 수행한다. 그 후, 로케이션 서비스(1310)는 물리적 로케이션을 게이트웨이(1304)에 송신한다. 데이터의 물리적 로케이션이 알려지면, 게이트웨이(1304)는 데이터의 로케이션 근처의 노드에서 작업부하 컨테이너(1306)를 스케줄링할 수 있다. FaaS 컨테이너(1306)는, 예를 들어, 통신 오버헤드의 많은 부분을 감소시키기 위해 스토리지 클러스터에서의 노드에 배치될 수 있다. 필요에 따라, 데이터는 데이터의 로케이션(예를 들어, 원격 스토리지(1314))으로부터, 작업부하 컨테이너(1306)와 동일한 노드 상에 있는 로컬 스토리지(1312)로 전송된다. 네트워크를 통해 데이터를 신속하게 전송하기 위해 다양한 통신 메커니즘들, 예를 들어, TCP/IP를 통한 NVMe(NVMe-oT) 또는 패브릭들을 통한 NVMe(NVMe-oF)가 사용될 수 있다. 작업부하 컨테이너(1306)가 데이터와 동일한 노드 상에서 스케줄링되는 경우, 필요하지 않을 수 있다.

[0167] 이와 같이, 레이턴시 한계 QoS 의도를 충족시키기 위해, 게이트웨이(1304)는 여러 옵션들을 갖는다. 함수의 (콜드(cold), 웜(warm) 또는 핫(hot)) 인스턴스화에 응답하여, 스케줄러는 함수를 배치하거나 또는 그것이 동작할 데이터에 가능한 한 "근접"하게 이미 배치된 함수를 활성화할 수 있다. 이벤트 트리거들에 응답하여, 프레임워크(예를 들어, 게이트웨이(1304) 또는 중간 함수)는: 1)(예를 들어, 히스토그램/히트맵/등의 형식으로 과거 실행들의 분석에 기초하여) 특정 세트의 데이터에 대한 동작을 위한 최상의 응답 시간을 보여준 함수 인스턴스를 결정하거나, 또는 2) 액세스되고 있는 데이터에 더 가까운 새로운 함수의 프로비저닝을 트리거할 수 있다. 또한, 스케일 아웃/인 동안, 게이트웨이(1304)는 원하는 의도들을 충족시키기 위해 콜드, 웜, 및 핫 함수 인스턴스들의 최상의 로케이션을 결정할 수 있다.

[0168] 덧붙여, 함수들이 체인화될 수 있고, 함수 내에서의 오케스트레이션 능력들(예컨대, AWS StepFunction, Azure Durable 함수들 등)이 전체적인 의도를 충족시키는 데 사용될 수 있다. 이것은 이벤트 트리거에서의 정보가 또

한 함수 실행의 최적 로케이션을 결정하기 위해 사용될 수 있기 때문에 필요할 수 있다. 이 이벤트 트리거 정보는 일반화된 게이트웨이가 동작하기에는 너무 애플리케이션 특정적이다.

[0169] 게다가, 스케줄링/트리거 라우팅 판정은 데이터를 보유하는 서버 상의 또는 그에 가까운 특정 컴퓨팅 리소스들의 존재에 기초하여 행해진다. 이 정보는, 요청된 레이턴시 한계 QoS 의도를 충족시키는 함수를 어디에서 실행할지를 결정하기 위해 FaaS 스케줄링 인프라스트럭처에 의해 이용될 수 있다.

[0170] 데이터 처리 QoS(예를 들어, 레이턴시 한계 실행)에 추가하여, 또는 그 대신에, 데이터 전송 QoS는 사용자 FaaS-CS 사양에 의해 지정될 수 있다. 사용자는 스토리지에 액세스할 필요가 있는 함수를 지정하고 이들 액세스에 대해 특정한 데이터 전송 속도가 유지될 것을 요청할 수 있다. FaaS 인프라스트럭처(예컨대, 게이트웨이(1304))는 이러한 데이터 전송 의도를 지원하기 위해 다양한 이용가능한 메커니즘들을 활용할 수 있다. DMA 전송들은 그로부터 컴퓨팅 리소스들이 데이터를 액세스할 수 있는 메모리로 데이터를 이동시키는 데 사용될 수 있다. 컴퓨팅 리소스들이 데이터를 보유하는 드라이브에 대한 피어 디바이스들로서 이용가능할 때, 피어 투 피어 전송들이 요청될 수 있다. 컴퓨팅 리소스가 원격에 있는 경우, RDMA와 같은 효율적인 전송 메커니즘들이 사용될 수 있다. 또한, 스위치들과 같은 네트워크 인프라스트럭처가 데이터 전송을 위한 네트워크 QoS를 제공하도록 프로그래밍될 수 있다. 게이트웨이(1304)는 요청된 전송 속도들을 충족시키기 위해서 함수로의 데이터 전송을 오케스트레이션하기 위해 이 메커니즘들 중 하나 이상을 사용할 수 있다.

[0171] 종종, 데이터는 프라이버시 법률들, 보안 요건들 등으로 인해 관리적 또는 지리적 경계들에 제한되어야 할 수 있다. 특정 경계들을 통과하지 못하는 데이터의 이러한 의도는 전형적으로 스토리지 서비스 내에서의 데이터 컨테이너들에 적용되는 정책으로서 표현된다. 함수가 의도된 경계 외부로부터 데이터에 액세스하려고 시도할 때, 스토리지 서비스는 그것을 방지할 것이다. 이러한 의도를 충족시키는 실시예는 사용자가 함수를 등록할 때 함수 실행의 원하는 경계를 부가된 파라미터로서 지정하는 것이다. 대안적인 실시예는 FaaS 인프라스트럭처를 확장하여, 경계 위반들로 인한 데이터 액세스 실패들에, 요구된 경계 내의 함수를 리스케줄링함으로써 동적으로 응답한다.

[0172] 현대의 데이터 센터들은 메모리 벽 또는 전력 벽이 아니라, 점점 더 데이터 이동 벽에 직면하고 있다. 데이터 이동은 데이터를 그것이 계산되는 곳으로 이동시켜야 하는 것으로 인한 전력, 대역폭, 및 레이턴시 비용들뿐만 아니라, 보안을 제공하는 것, 보안을 제공하는 프로세스에서 데이터를 변환하는 것, 대역폭 증폭, 스토리지에서의 기입 증폭들, 불균형한 혼잡들, 및 등등의 다른 연관된 비용들에 의해 제한된다. 데이터를 컴퓨팅에 이동시키는 대신에 컴퓨팅을 데이터에 이동시키는 것은 새로운 말이지만, 이것이 항상 가능하거나 실용적인 것은 아닌데, 그 이유는 일부 경우들에서는 컴퓨팅이 가속 능력과 같은 움직일 수 없는 어떤 것과 결부되고, 다른 경우들에서는 데이터의 생산자(소스)가 그것이 제공할 수 있는 컴퓨팅의 양에 있어서 제한될 수 있기 때문이다. 따라서, 데이터 샤딩(data-sharding)과 유사한 컴퓨팅 샤딩(compute-sharding) 개념을 제공하여, 컴퓨팅 샤드(compute-shard)들에 있을 필요가 있는 데이터는 거기에 사전 위치되고, 계산이 데이터 중력의 원리에 따라 분산되도록 하는 것이 이상적이다. 데이터 중력(data gravity)은 생산자들 및 소비자들, 가속 규정들(acceleration provisions) 등이 데이터 위치결정 및 배치 아키텍처에 따라 동일 장소에 배치되도록 설계되는 것을 의도한다. 대용량 CXL-연결된 풀들에서의 메모리 풀링은 가까운 이웃(예를 들어, 1-홉) 서버들의 세트가 스위치를 통해 서로 작업할 필요 없이 계산 및 메모리 리소스들을 동시에 풀링하기 위한 능력을 최대화한다. 이것은 또한 자연스럽게 매우 낮은 오버헤드의 DSM(distributed shared memory) 구현들로 이끄는데, 그 이유는 주어진 컴퓨팅 샤드가 DSM에 있지만 동일한 메모리 풀에 연결되는 상이한 컴퓨팅 샤드에 의해 작업되고 있는 데이터를 드물게 푸시하거나 풀링하기 때문이다.

[0173] 시스템은 "의도"를 데이터에 할당하여, 보안, 전력, 냉각, 지리 국소성, 또는 프라이버시를 포함하는 속성들을 연관시켜 데이터에 대한 무게 중심(center of gravity)을 생성한다. 이 무게 중심은 컴퓨팅에 의해 다이제스트(digest)될 준비가 되도록 데이터를 위치시키기 위해 사용된다. 이것은 컴퓨팅을 직접적으로 할당하는 것과 구별되며, 컴퓨팅 샤딩에 결부될 수 있다. 이 접근법은 리소스 오케스트레이션에 통합될 수 있는데, 그 이유는 데이터 "무게 중심"이 새로운 스케줄링 방법/운영자로서 제공될 수 있어서, 컴퓨팅을 데이터와 동일 장소에 배치하기 위해 스케줄링 알고리즘에 대한 입력을 사용하기 때문이다.

[0174] 데이터 액세스 패턴들은 크기, 양, 주파수, 및 레이턴시 요건들에서 가능한 차이들에 관련된 변경들과 같이, 시간 경과에 따라 변경될 것으로 예상된다. 시스템은 데이터 액세스 패턴들을 지속적으로 평가하고 이들을 정의된 의도 및 정책과 비교하여 필요에 따라 조절하는 폐루프 제어 메커니즘을 포함한다.

[0175] 도 14는 실시예에 따른, 계산 스토리지를 구현하기 위한 방법(1400)을 도시하는 흐름도이다. (1402)에서, 등록

패키지가 오케스트레이터 시스템에서 수신된다. 등록 패키지는 함수 코드, 함수 코드에 대한 입력 데이터의 논리적 로케이션, 및 함수 코드에 대한 이벤트 트리거를 포함하고, 이벤트 트리거는 입력 데이터가 수정될 때에 응답하여 트리거하도록 설정된다.

[0176] (1404)에서, 스토리지 서비스는 스토리지 서비스가 입력 데이터의 논리적 로케이션을 모니터링하고 입력 데이터가 수정될 때를 로케이션 서비스에 통지하는 곳과 인터페이스된다. 실시예에서, 등록 패키지는 입력 데이터에 액세스하기 위해 사용되는 크리덴셜을 포함한다. 실시예에서, 등록 패키지는 출력 데이터에 대한 논리적 로케이션을 포함한다. 추가 실시예에서, 출력 데이터에 대한 논리적 로케이션은 URL(universal resource locator)이다. 실시예에서, 스토리지 서비스는 비구조화된 객체 스토리지 서비스이다. 비구조화된 객체 스토리지 서비스의 예는 MinIO이다.

[0177] (1406)에서, 로케이션 서비스는 입력 데이터의 물리적 로케이션을 획득하기 위한 것과 인터페이스되고, 로케이션 서비스는 입력 데이터의 논리적 로케이션으로부터 물리적 로케이션을 결정한다. 역 DNS(domain name service) 룩업은 논리적 로케이션으로부터 물리적 로케이션을 결정하기 위해 사용될 수 있다.

[0178] (1408)에서, 함수 코드가 입력 데이터 근처에서 실행되도록 구성된다. 실시예에서, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 것은 입력 데이터의 물리적 로케이션에 있는 서버에 함수 코드를 배치하는 것을 포함한다.

[0179] 실시예에서, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 것은 물리적 로케이션으로부터 함수 코드가 실행되도록 스케줄링되는 컴퓨팅 노드로 입력 데이터의 적어도 일부를 프리페치하는 것을 포함한다. 함수 코드는 프리페치된 입력 데이터를 사용하고 그 후 원래의 입력 데이터 소스로부터 데이터 스트림으로 스위칭하도록 구성될 수 있다. 이러한 방식으로, 함수 코드는 근처 데이터를 신속하게 처리할 수 있고, 그 후 더 많은 데이터가 필요함에 따라 네트워크를 통해 데이터에 스위칭할 수 있다.

[0180] 실시예에서, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 것은 함수 코드의 복수의 인스턴스로부터 함수 코드의 한 인스턴스를 선택하는 것을 포함하고, 함수 코드의 선택된 인스턴스는 적어도 하나의 성능 메트릭에 따라 함수 코드의 복수의 인스턴스 중 다른 것들보다 비교적 더 양호하게 수행된 컴퓨팅 노드에서 실행된다. 성능 메트릭들은 레이턴시, 실행 시간, 및 메모리 사용을 포함하지만 이들로 제한되지는 않는다.

[0181] 실시예에서, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 것은 함수 코드의 복수의 인스턴스로부터 함수 코드의 한 인스턴스를 선택하는 것을 포함하고, 함수 코드의 복수의 인스턴스는 대응하는 복수의 컴퓨팅 노드 상에서 실행되고, 함수 코드의 선택된 인스턴스는 복수의 컴퓨팅 노드 중 다른 것들보다 입력 데이터에 물리적으로 더 가까운 복수의 컴퓨팅 노드들 중 한 컴퓨팅 노드에서 실행된다.

[0182] 본 문서에 기술된 오케스트레이터 또는 오케스트레이션 시스템들은 기기, 엔드포인트 컴포넌트, 디바이스, 클라이언트 디바이스, 컴퓨팅 디바이스, 노드, 서버, 개별적으로 판매되고 상이한 시스템에 통합되는 독립형 노드 등에서 구현될 수 있다. 시스템 아키텍트는 임의의 설치에서 오케스트레이션 시스템을 구현할 수 있다.

[0183] 실시예들은 하드웨어, 펌웨어, 및 소프트웨어 중 하나 또는 조합으로 구현될 수 있다. 실시예들은 또한 본 명세서에서 설명된 동작을 수행하기 위해 적어도 하나의 프로세서에 의해 판독 및 실행될 수 있는 머신 판독가능 스토리지 디바이스 상에 저장된 명령어들로서 구현될 수 있다. 머신 판독가능 스토리지 디바이스는 머신(예컨대, 컴퓨터)에 의해 판독가능한 형식으로 정보를 저장하기 위한 임의의 비일시적인 메커니즘을 포함할 수 있다. 예를 들어, 컴퓨터 판독가능 스토리지 디바이스는 ROM(read-only memory), RAM(random-access memory), 자기 디스크 저장 매체, 광 저장 매체, 플래시 메모리(flash-memory) 디바이스들, 및 다른 저장 디바이스들 및 매체를 포함할 수 있다.

[0184] 예들은, 본 명세서에 설명되는 바와 같이, 모듈들, IP(intellectual property) 블록들 또는 코어들, 엔진들, 또는 메커니즘들과 같은, 로직 또는 다수의 컴포넌트를 포함할 수 있거나, 또는 이들 상에서 동작할 수 있다. 이러한 로직 또는 컴포넌트들은 본 명세서에 설명된 동작들을 수행하기 위해 하나 이상의 프로세서에 통신가능하게 결합된 하드웨어, 소프트웨어 구성된 하드웨어, 또는 펌웨어일 수 있다. 로직 또는 컴포넌트들은 하드웨어 모듈들(예를 들어, IP 블록)일 수 있고, 이와 같이 특정된 동작들을 수행할 수 있는 유형의 엔티티들로 간주될 수 있고 특정 방식으로 구성되거나 배열될 수 있다. 예에서, 회로들은 IP 블록, IP 코어, SoC(system-on-chip), 또는 그와 유사한 것으로서 특정된 방식으로 (예를 들어, 내부적으로 또는 다른 회로들과 같은 외부 엔티티들에 대해) 배열될 수 있다.

[0185] 예에서, 하나 이상의 컴퓨터 시스템(예를 들어, 독립적인, 클라이언트 또는 서버 컴퓨터 시스템) 또는 하나 이

상의 하드웨어 프로세서의 전부 또는 일부는 특정된 동작들을 수행하도록 동작하는 모듈로서 펌웨어 또는 소프트웨어(예를 들어, 명령어들, 애플리케이션 부분, 또는 애플리케이션)에 의해 구성될 수 있다. 예에서, 소프트웨어는 머신 판독가능 매체 상에 상주할 수 있다. 예에서, 소프트웨어는, 모듈의 기반 하드웨어에 의해 실행될 때, 하드웨어가 특정된 동작들을 수행하도록 야기한다. 따라서, 용어 하드웨어 모듈은 유형의 엔티티를 포괄하며, 특정 방식으로 동작하거나, 또는 본 명세서에서 설명된 임의의 동작의 일부 또는 전부를 수행하도록 물리적으로 구성되거나, 구체적으로 구성되거나(예를 들어, 하드와이어링되거나), 또는 임시로(예를 들어, 일시적으로) 구성되는 (예를 들어, 프로그래밍되는) 엔티티인 것으로 이해된다.

[0186] 모듈들이 임시로 구성되는 예들을 고려하면, 모듈들 각각이 시간 상 임의의 한 순간에 인스턴트화될 필요는 없다. 예를 들어, 모듈들이 소프트웨어를 사용하여 구성된 범용 하드웨어 프로세서를 포함하는 경우에; 범용 하드웨어 프로세서는 상이한 시간들에 각자의 상이한 모듈들로서 구성될 수 있다. 따라서, 소프트웨어는, 예를 들어, 하나의 시간 인스턴스에서 특정 모듈을 구성하고, 상이한 시간 인스턴스에서 상이한 모듈을 구성하도록 하드웨어 프로세서를 구성할 수 있다. 모듈들은 또한 소프트웨어 또는 펌웨어 모듈들일 수 있으며, 이것들은 본 명세서에서 설명된 방법론들을 수행하도록 동작한다.

[0187] IP 블록(IP 코어라고도 지칭됨)은 로직, 셀, 또는 집적 회로의 재사용가능 유닛이다. IP 블록은 FPGA(field programmable gate array), ASIC(application-specific integrated circuit), PLD(programmable logic device), SoC(system on a chip), 또는 그와 유사한 것의 일부로서 사용될 수 있다. 그것은 디지털 신호 처리 또는 이미지 처리와 같은 특정 목적을 위해 구성될 수 있다. 예시적인 IP 코어들은 CPU(central processing unit) 코어들, 통합된 그래픽들, 보안, 입력/출력(I/O) 제어, 시스템 에이전트, GPU(graphics processing unit), 인공 지능, 신경 프로세서들, 이미지 처리 유닛, 통신 인터페이스들, 메모리 컨트롤러, 주변 디바이스 제어, 플랫폼 컨트롤러 허브, 또는 그와 유사한 것을 포함한다.

[0188] 추가적인 유의사항들 & 예들:

[0189] 예 1은 오케스트레이터 시스템으로서: 프로세서; 및 명령어들을 저장하는 메모리를 포함하고, 명령어들은 프로세서에 의해 실행될 때, 오케스트레이터 시스템으로 하여금: 오케스트레이터 시스템에서, 등록 패키지를 수신하고 - 등록 패키지는 함수 코드, 함수 코드에 대한 입력 데이터의 논리적 로케이션, 및 함수 코드에 대한 이벤트 트리거를 포함하고, 이벤트 트리거는 입력 데이터가 수정될 때에 응답하여 트리거하도록 설정됨 -; 스토리지 서비스와 인터페이스하고 - 스토리지 서비스는 입력 데이터의 논리적 로케이션을 모니터링하고 입력 데이터가 수정될 때를 로케이션 서비스에 통지함 - ; 로케이션 서비스와 인터페이스하여 입력 데이터의 물리적 로케이션을 획득하고 - 로케이션 서비스는 입력 데이터의 논리적 로케이션으로부터 물리적 로케이션을 결정함 -; 및 입력 데이터 근처에서 실행되도록 함수 코드를 구성하게 야기한다.

[0190] 예 2에서, 예 1의 주제는, 등록 패키지가 입력 데이터에 액세스하기 위해 사용되는 크리덴셜을 포함하는 것을 포함한다.

[0191] 예 3에서, 예 1 또는 예 2의 주제는, 등록 패키지가 출력 데이터에 대한 논리적 로케이션을 포함하는 것을 포함한다.

[0192] 예 4에서, 예 3의 주제는, 출력 데이터에 대한 논리적 로케이션이 URL(universal resource locator)인 것을 포함한다.

[0193] 예 5에서, 예들 1-4의 주제는, 스토리지 서비스가 비구조화된 객체 스토리지 서비스인 것을 포함한다.

[0194] 예 6에서, 예들 1-5의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하기 위해, 오케스트레이터 시스템이 입력 데이터의 물리적 로케이션에 있는 서버에 함수 코드를 배치하는 것을 포함한다.

[0195] 예 7에서, 예들 1-6의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하기 위해, 오케스트레이터 시스템이 물리적 로케이션으로부터 함수 코드가 실행되도록 스케줄링되는 컴퓨팅 노드로 입력 데이터의 적어도 일부를 프리페치하는 것을 포함한다.

[0196] 예 8에서, 예들 1-7의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하기 위해, 오케스트레이터 시스템이 함수 코드의 복수의 인스턴스로부터 함수 코드의 한 인스턴스를 선택하고, 함수 코드의 선택된 인스턴스는 적어도 하나의 성능 메트릭에 따라 함수 코드의 복수의 인스턴스 중 다른 것들보다 비교적 더 양호하게 수행된 한 컴퓨팅 노드에서 실행되는 것을 포함한다.

[0197] 예 9에서, 예들 1-8의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하기 위해, 오케스트레이터

시스템이 함수 코드의 복수의 인스턴스로부터 함수 코드의 한 인스턴스를 선택하고, 함수 코드의 복수의 인스턴스는 대응하는 복수의 컴퓨팅 노드 상에서 실행되고, 함수 코드의 선택된 인스턴스는 복수의 컴퓨팅 노드 중 다른 것들보다 입력 데이터에 물리적으로 더 가까운 복수의 컴퓨팅 노드 중 한 컴퓨팅 노드에서 실행되는 것을 포함한다.

[0198] 예 10은 방법으로서: 오케스트레이터 시스템에서, 등록 패키지를 수신하는 단계 - 등록 패키지는 함수 코드, 함수 코드에 대한 입력 데이터의 논리적 로케이션, 및 함수 코드에 대한 이벤트 트리거를 포함하고, 이벤트 트리거는 입력 데이터가 수정될 때에 응답하여 트리거하도록 설정됨 -; 스토리지 서비스와 인터페이스하는 단계 - 스토리지 서비스는 입력 데이터의 논리적 로케이션을 모니터링하고 입력 데이터가 수정될 때를 로케이션 서비스에 통지함 - ; 로케이션 서비스와 인터페이스하여 입력 데이터의 물리적 로케이션을 획득하는 단계 - 로케이션 서비스는 입력 데이터의 논리적 로케이션으로부터 물리적 로케이션을 결정함 -; 및 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 단계를 포함한다.

[0199] 예 11에서, 예 10의 주제는, 등록 패키지가 입력 데이터에 액세스하기 위해 사용되는 크리덴셜을 포함하는 것을 포함한다.

[0200] 예 12에서, 예 10 또는 예 11의 주제는, 등록 패키지가 출력 데이터에 대한 논리적 로케이션을 포함하는 것을 포함한다.

[0201] 예 13에서, 예 12의 주제는, 출력 데이터에 대한 논리적 로케이션이 URL(universal resource locator)인 것을 포함한다.

[0202] 예 14에서, 예들 10-13의 주제는, 스토리지 서비스가 비구조화된 객체 스토리지 서비스인 것을 포함한다.

[0203] 예 15에서, 예들 10-14의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 단계가 입력 데이터의 물리적 로케이션에 있는 서버에 함수 코드를 배치하는 단계를 포함하는 것을 포함한다.

[0204] 예 16에서, 예들 10-15의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 단계가 물리적 로케이션으로부터 함수 코드가 실행되도록 스케줄링되는 컴퓨팅 노드로 입력 데이터의 적어도 일부를 프리페치하는 단계를 포함하는 것을 포함한다.

[0205] 예 17에서, 예들 10-16의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 단계가 함수 코드의 복수의 인스턴스로부터 함수 코드의 한 인스턴스를 선택하는 단계를 포함하고, 함수 코드의 선택된 인스턴스는 적어도 하나의 성능 메트릭에 따라 함수 코드의 복수의 인스턴스 중 다른 것들보다 비교적 더 양호하게 수행된 한 컴퓨팅 노드에서 실행되는 것을 포함한다.

[0206] 예 18에서, 예들 10-17의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 단계가 함수 코드의 복수의 인스턴스로부터 함수 코드의 한 인스턴스를 선택하는 단계를 포함하고, 함수 코드의 복수의 인스턴스는 대응하는 복수의 컴퓨팅 노드 상에서 실행되고, 함수 코드의 선택된 인스턴스는 복수의 컴퓨팅 노드 중 다른 것들보다 입력 데이터에 물리적으로 더 가까운 복수의 컴퓨팅 노드 중 한 컴퓨팅 노드에서 실행되는 것을 포함한다.

[0207] 예 19는, 머신에 의해 실행될 때, 머신으로 하여금 예들 10-18의 방법들 중 임의의 것의 동작들을 수행하게 야기하는 명령어들을 포함하는 적어도 하나의 머신 판독가능 매체이다.

[0208] 예 20은 예들 10-18의 방법들 중 임의의 것을 수행하기 위한 수단을 포함하는 장치이다.

[0209] 예 21은 명령어들을 포함하는 적어도 하나의 머신 판독가능 매체이고, 명령어들은, 머신에 의해 실행될 때, 머신으로 하여금 동작들을 수행하도록 야기하고, 동작들은: 오케스트레이터 시스템에서, 등록 패키지를 수신하는 동작 - 등록 패키지는 함수 코드, 함수 코드에 대한 입력 데이터의 논리적 로케이션, 및 함수 코드에 대한 이벤트 트리거를 포함하고, 이벤트 트리거는 입력 데이터가 수정될 때에 응답하여 트리거하도록 설정됨 -; 스토리지 서비스와 인터페이스하는 동작 - 스토리지 서비스는 입력 데이터의 논리적 로케이션을 모니터링하고 입력 데이터가 수정될 때를 로케이션 서비스에 통지함 - ; 로케이션 서비스와 인터페이스하여 입력 데이터의 물리적 로케이션을 획득하는 동작 - 로케이션 서비스는 입력 데이터의 논리적 로케이션으로부터 물리적 로케이션을 결정함 -; 및 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 동작을 포함한다.

[0210] 예 22에서, 예 21의 주제는, 등록 패키지가 입력 데이터에 액세스하기 위해 사용되는 크리덴셜을 포함하는 것을 포함한다.

[0211] 예 23에서, 예 21 또는 예 22의 주제는, 등록 패키지가 출력 데이터에 대한 논리적 로케이션을 포함하는 것을 포함한다.

[0212] 예 24에서, 예 23의 주제는, 출력 데이터에 대한 논리적 로케이션이 URL(universal resource locator)인 것을 포함한다.

[0213] 예 25에서, 예들 21-24의 주제는, 스토리지 서비스가 비구조화된 객체 스토리지 서비스인 것을 포함한다.

[0214] 예 26에서, 예들 21-25의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 동작은 입력 데이터의 물리적 로케이션에 있는 서버에 함수 코드를 배치하는 동작을 포함하는 것을 포함한다.

[0215] 예 27에서, 예들 21-26의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 동작은 물리적 로케이션으로부터 함수 코드가 실행되도록 스케줄링되는 컴퓨팅 노드로 입력 데이터의 적어도 일부를 프리페치하는 동작을 포함하는 것을 포함한다.

[0216] 예 28에서, 예들 21-27의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 동작은 함수 코드의 복수의 인스턴스로부터 함수 코드의 한 인스턴스를 선택하는 동작을 포함하는 - 함수 코드의 선택된 인스턴스는 적어도 하나의 성능 메트릭에 따라 함수 코드의 복수의 인스턴스 중 다른 것들보다 비교적 더 양호하게 수행된 한 컴퓨팅 노드에서 실행됨 - 것을 포함한다.

[0217] 예 29에서, 예들 21-28의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 동작은 함수 코드의 복수의 인스턴스로부터 함수 코드의 한 인스턴스를 선택하는 동작을 포함하는 - 함수 코드의 복수의 인스턴스는 대응하는 복수의 컴퓨팅 노드 상에서 실행되고, 함수 코드의 선택된 인스턴스는 복수의 컴퓨팅 노드 중 다른 것들보다 입력 데이터에 물리적으로 더 가까운 복수의 컴퓨팅 노드 중 한 컴퓨팅 노드에서 실행됨 - 것을 포함한다.

[0218] 예 30은 장치로서: 오케스트레이터 시스템에서, 등록 패키지를 수신하기 위한 수단 - 등록 패키지는 함수 코드, 함수 코드에 대한 입력 데이터의 논리적 로케이션, 및 함수 코드에 대한 이벤트 트리거를 포함하고, 이벤트 트리거는 입력 데이터가 수정될 때에 응답하여 트리거하도록 설정됨 -; 스토리지 서비스와 인터페이스하기 위한 수단 - 스토리지 서비스는 입력 데이터의 논리적 로케이션을 모니터링하고, 입력 스토리지이터가 수정될 때를 로케이션 서비스에 통지함 -; 로케이션 서비스와 인터페이스하여 입력 데이터의 물리적 로케이션을 획득하기 위한 수단 - 로케이션 서비스는 입력 데이터의 논리적 로케이션으로부터 물리적 로케이션을 결정함 -; 및 입력 데이터 근처에서 실행되도록 함수 코드를 구성하기 위한 수단을 포함한다.

[0219] 예 31에서, 예 30의 주제는, 등록 패키지가 입력 데이터에 액세스하기 위해 사용되는 크리덴셜을 포함하는 것을 포함한다.

[0220] 예 32에서, 예 30 또는 예 31의 주제는, 등록 패키지가 출력 데이터에 대한 논리적 로케이션을 포함하는 것을 포함한다.

[0221] 예 33에서, 예 32의 주제는, 출력 데이터에 대한 논리적 로케이션이 URL(universal resource locator)인 것을 포함한다.

[0222] 예 34에서, 예들 30-33의 주제는, 스토리지 서비스가 비구조화된 객체 스토리지 서비스인 것을 포함한다.

[0223] 예 35에서, 예들 30-34의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 것이 입력 데이터의 물리적 로케이션에 있는 서버에 함수 코드를 배치하는 것을 포함하는 것을 포함한다.

[0224] 예 36에서, 예들 30-35의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 것이 물리적 로케이션으로부터 함수 코드가 실행되도록 스케줄링되는 컴퓨팅 노드로 입력 데이터의 적어도 일부를 프리페치하는 것을 포함하는 것을 포함한다.

[0225] 예 37에서, 예들 30-36의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 것이 함수 코드의 복수의 인스턴스로부터 함수 코드의 한 인스턴스를 선택하는 것을 포함하는 - 함수 코드의 선택된 인스턴스는 적어도 하나의 성능 메트릭에 따라 함수 코드의 복수의 인스턴스 중 다른 것들보다 비교적 더 양호하게 수행된 한 컴퓨팅 노드에서 실행됨 - 것을 포함한다.

[0226] 예 38에서, 예들 30-37의 주제는, 입력 데이터 근처에서 실행되도록 함수 코드를 구성하는 것이 함수 코드의 복수의 인스턴스로부터 함수 코드의 한 인스턴스를 선택하는 것을 포함하는 - 함수 코드의 복수의 인스턴스는 대

응하는 복수의 컴퓨팅 노드 상에서 실행되고, 함수 코드의 선택된 인스턴스는 복수의 컴퓨팅 노드 중 다른 것들보다 입력 데이터에 물리적으로 더 가까운 복수의 컴퓨팅 노드 중 한 컴퓨팅 노드에서 실행됨 - 것을 포함한다.

[0227] 예 39는, 처리 회로에 의해 실행될 때, 처리 회로로 하여금 예 1 내지 예 38 중 임의의 예를 구현하는 동작들을 수행하게 야기하는 명령어들을 포함하는 적어도 하나의 머신 판독가능 매체이다.

[0228] 예 40은 예 1 내지 예 38 중 임의의 예를 구현하는 수단을 포함하는 장치이다.

[0229] 예 41은 예 1 내지 예 38 중 임의의 예를 구현하는 시스템이다.

[0230] 예 42는 예 1 내지 예 38 중 임의의 예를 구현하는 방법이다.

[0231] 위의 상세한 설명은 상세한 설명의 일부를 형성하는 첨부 도면들에 대한 참조들을 포함한다. 도면들은, 예시로서, 실시될 수 있는 특정 실시예들을 보여준다. 이러한 실시예들은 본 명세서에서 "예들(examples)"이라고 또한 지칭된다. 이러한 예들은 도시되거나 설명된 것 이외의 요소들을 포함할 수 있다. 그러나, 도시되거나 설명되는 요소들을 포함하는 예들이 또한 고려된다. 더욱이, 특정한 예(또는 그의 하나 이상의 양태)에 관련하여, 또는 본 명세서에 도시되거나 설명된 다른 예들(또는 그의 하나 이상의 양태)에 관련하여, 도시되거나 설명된 해당 요소들(또는 그의 하나 이상의 양태)의 임의의 조합 또는 치환을 사용하는 예들이 또한 고려된다.

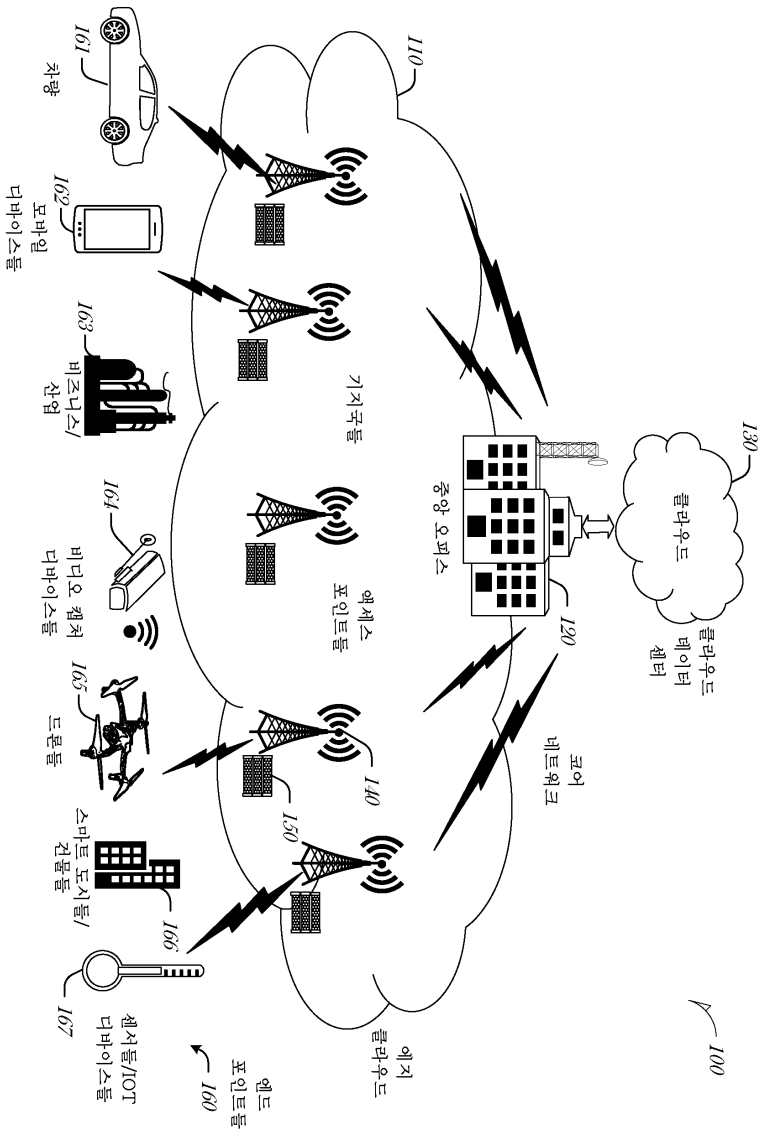
[0232] 본 문서에서 참조되는 간행물, 특허, 및 특허 문서는, 참조 문헌으로서 개별적으로 포함되기나 한 것처럼, 그 전체가 참조 문헌으로서 본 명세서에 포함된다. 이 문서와 참조로 그렇게 포함된 문서들 사이의 불일치한 사용들의 경우에, 포함된 참조문헌(들)에서의 사용은 이 문서의 사용에 대해 보충적이다; 양립할 수 없는 불일치들에 대해서는, 이 문서에서의 사용이 우선한다.

[0233] 본 문서에서, "하나(a 또는 an)"라는 용어는, 특허 문헌들에서 흔한 것처럼, "적어도 하나" 또는 "하나 이상"의 임의의 다른 경우들 또는 사용들에 독립적인, 하나 또는 하나보다 많은 것을 포함하기 위해 사용된다. 이 문서에서, 용어 "또는"은, 달리 표시되지 않는 한, "A 또는 B"가 "B가 아닌 A", "A가 아닌 B", 및 "A 및 B"를 포함하도록, 비배타적 또는을 지칭하기 위해 사용된다. 첨부된 청구범위에서, 용어 "포함하는(including)" 및 "여기서(in which)"는 각자의 용어 "포함하는(comprising)" 및 "여기서(wherein)"의 평이한 동등어(plain-English equivalents)로서 사용된다. 또한, 다음의 청구항들에서, 용어 "포함하는(including)" 및 "포함하는(comprising)"은 개방적인데(open-ended), 즉, 청구항에서 그러한 용어 이후에 나열된 것들에 추가하여 요소들을 포함하는 시스템, 디바이스, 물품, 또는 프로세스는 여전히 그 청구항의 범위 내에 드는 것으로 간주된다. 더욱이, 다음의 청구항들에서, "제1(first)", "제2(second)", 및 "제3(third)" 등의 용어들은 단지 레이블들로서 사용되고, 그것들의 대상들에 수치적 순서를 제안하려고 의도되는 것은 아니다.

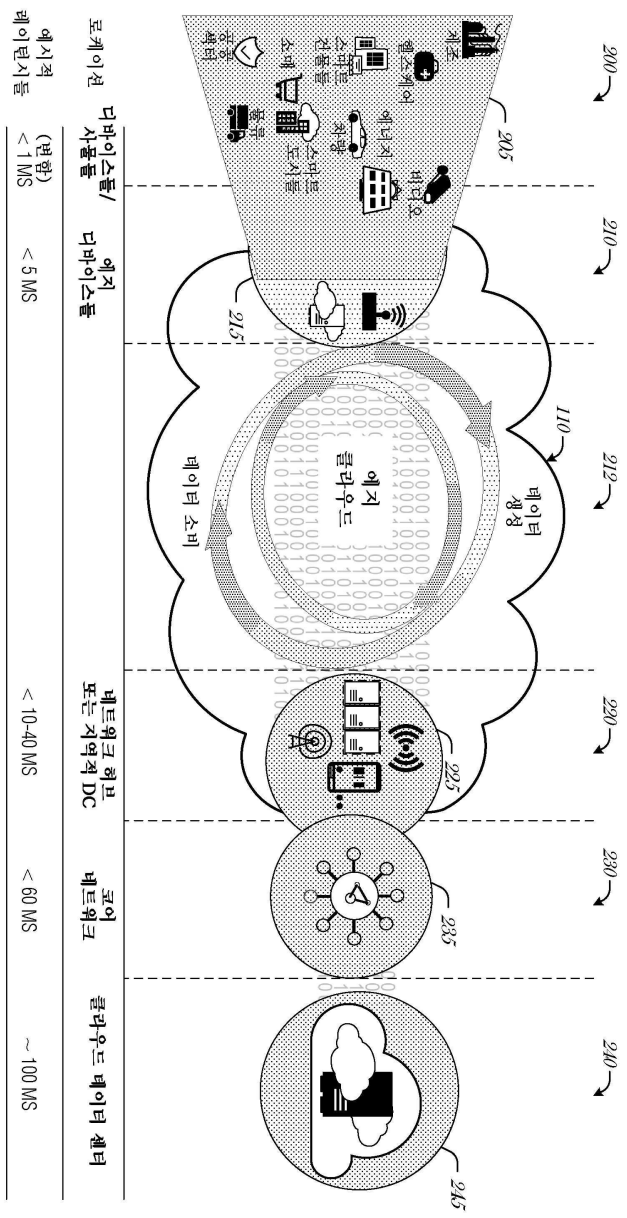
[0234] 이상의 설명은 제한이 아니라 예시를 위한 것이다. 예를 들어, 위에 설명된 예들(또는 그의 하나 이상의 양태)은 다른 것들과 조합되어 사용될 수 있다. 다른 실시예들이, 위의 설명의 검토시에 본 기술분야의 통상의 기술자에 의해 그런 것처럼 사용될 수 있다. 요약서는 독자가 기술적 개시내용의 본질을 신속하게 확인하게 하는 것이다. 그것이 청구항들의 범위 또는 의미를 해석하거나 제한하는 데 이용되지는 않을 것이라는 이해 하에서 제출된다. 또한, 위의 상세한 설명에서, 본 개시내용을 간소화하기 위해 다양한 특징들이 함께 그룹화될 수 있다. 그러나, 실시예들이 상기 특징들의 서브세트를 특징으로 할 수 있기 때문에, 청구항들은 본 명세서에 개시된 모든 특징을 제시하지 않을 수 있다. 또한, 실시예들은 특정 예에서 개시되는 것보다 더 적은 특징들을 포함할 수 있다. 따라서, 이하의 청구항들은 이에 의해 상세한 설명에 포함되고, 청구항은 그 자체로 별도의 실시예로서 성립한다. 본 명세서에 개시되는 실시예들의 범위는, 첨부된 청구항들에 부여되는 등가물들의 전체 범위와 함께, 이러한 청구항들을 참조하여 결정되어야 한다.

도면

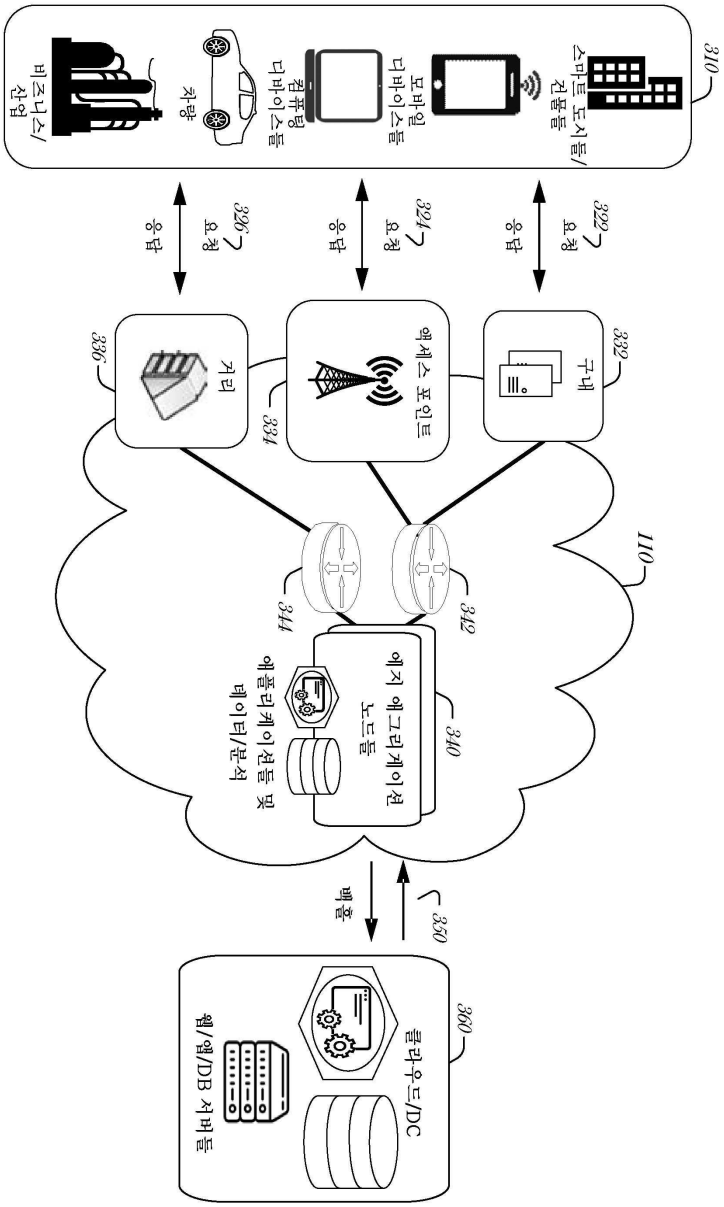
도면1


100
110
에지 클라우드
120
중앙 오피스
130
클라우드 데이터 센터
140
액세스 포인트들
150
기지국들
160
엔드 포인트들
161
차량
162
모바일 디바이스들
163
비즈니스/산업
164
비디오 캡처 디바이스들
165
드론들
166
스마트 도시들/건물들
167
센서들/IOT 디바이스들
코어 네트워크

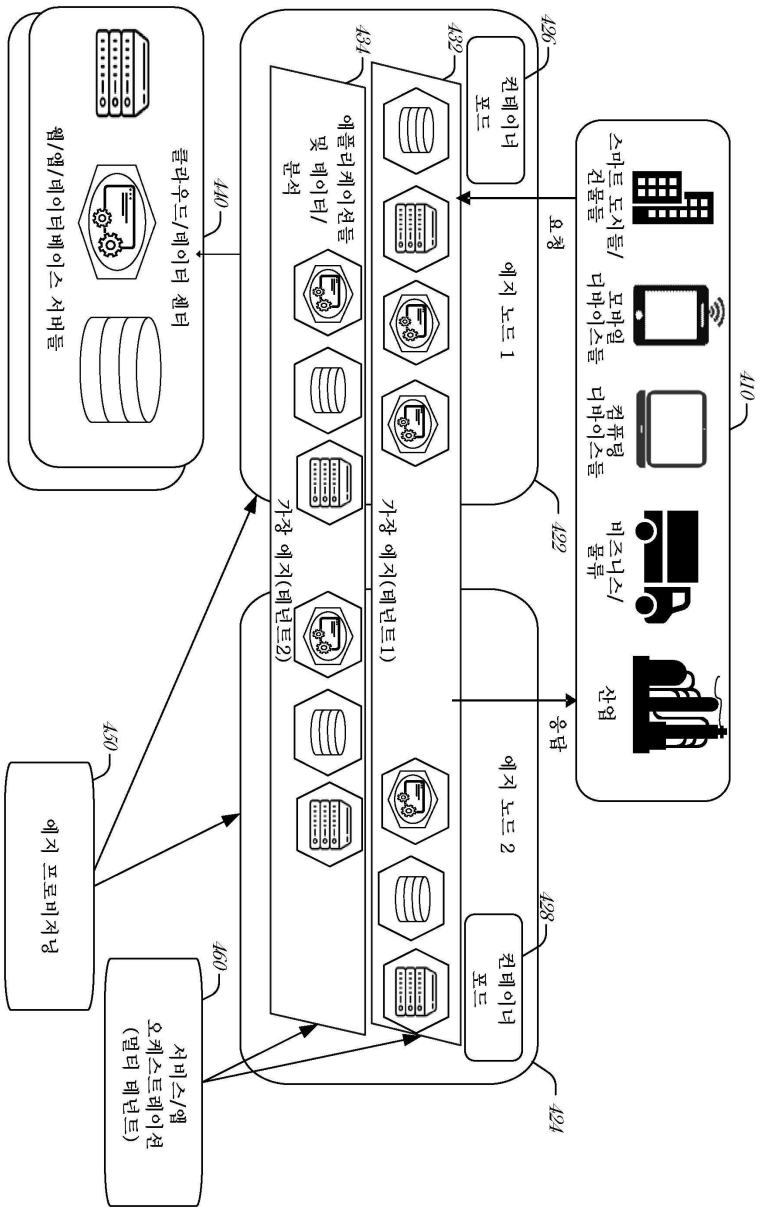
도면2


200
205
210
215
110
212
데이터 생성
에지 클라우드
데이터 소비
220
225
230
235
240
245
로케이션
디바이스들/사물들
에지 디바이스들
네트워크 허브 또는 지역적 DC
코어 네트워크
클라우드 데이터 센터
예시적 레이턴시들
(변함) < 1 MS
< 5 MS
< 10-40 MS
< 60 MS
~ 100 MS

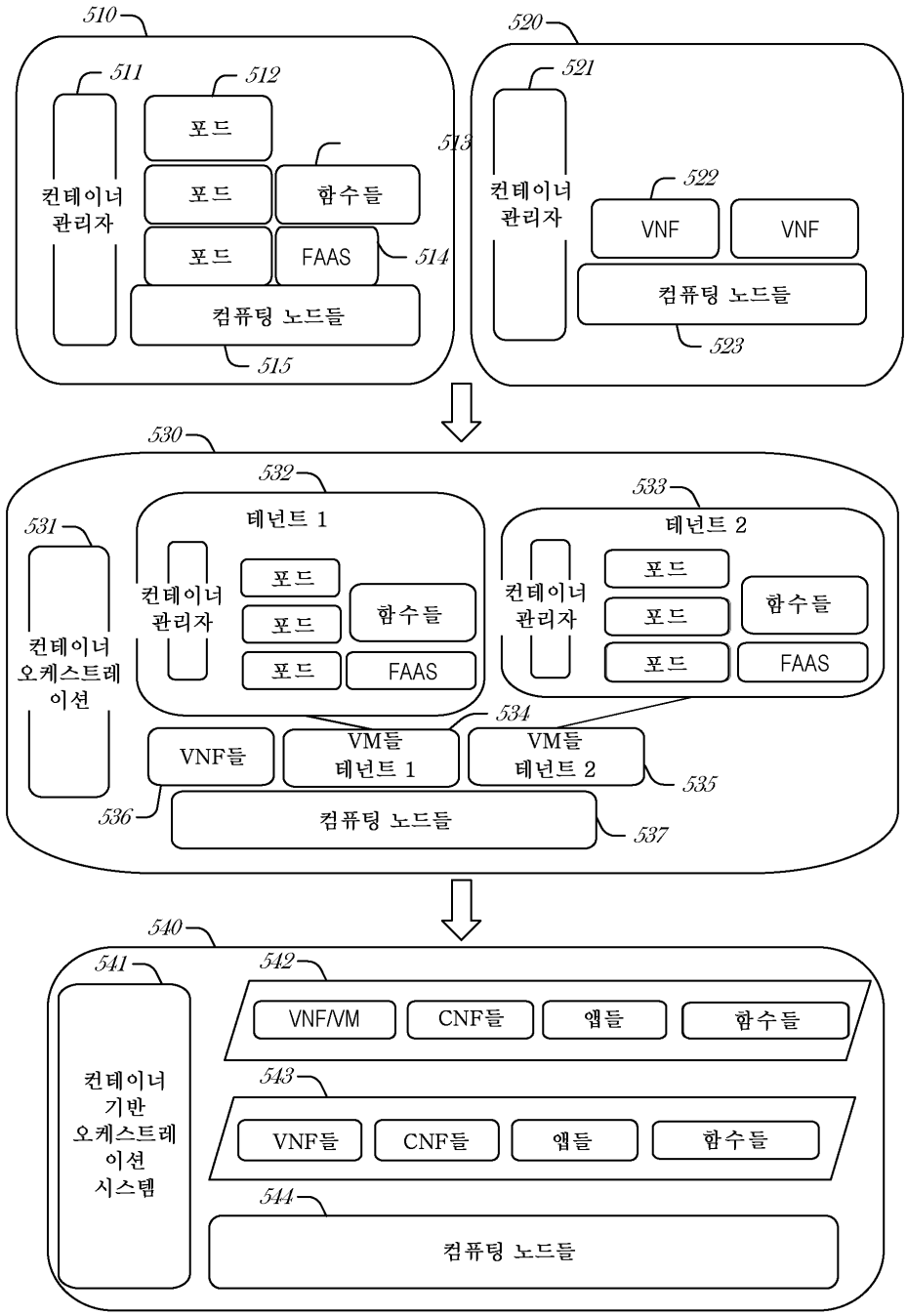
도면3


310
스마트 도시들/
건물들
모바일
디바이스들
컴퓨팅
디바이스들
차량
비즈니스/
산업
326
요청
응답
324
요청
응답
322
요청
응답
332
구내
334
액세스 포인트
336
거리
110
342
344
340
엣지 애그리게이션
노드들
애플리케이션들 및
데이터/분석
350
백홀
360
클라우드/DC
웹/앱/DB 서버들

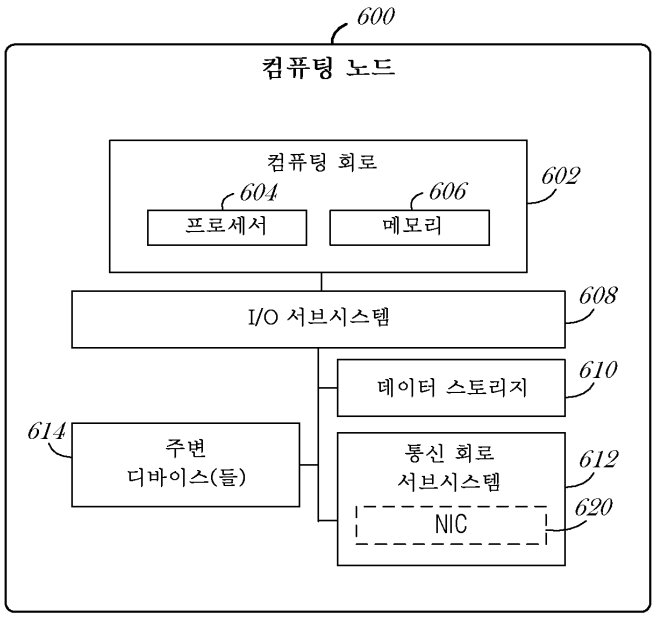
도면4


410
스마트 도시들/
건물들
모바일
디바이스들
컴퓨팅
디바이스들
비즈니스/
물류
산업
요청
응답
420
422
424
426
428
430
432
434
440
450
460
컨테이너
포드
에지 노드 1
에지 노드 2
가장 에지(테넌트1)
가장 에지(테넌트2)
애플리케이션들
및 데이터/
분석
클라우드/데이터 센터
웹/앱/데이터베이스 서버들
에지 프로비저닝
서비스/앱
오케스트레이션
(멀티 테넌트)

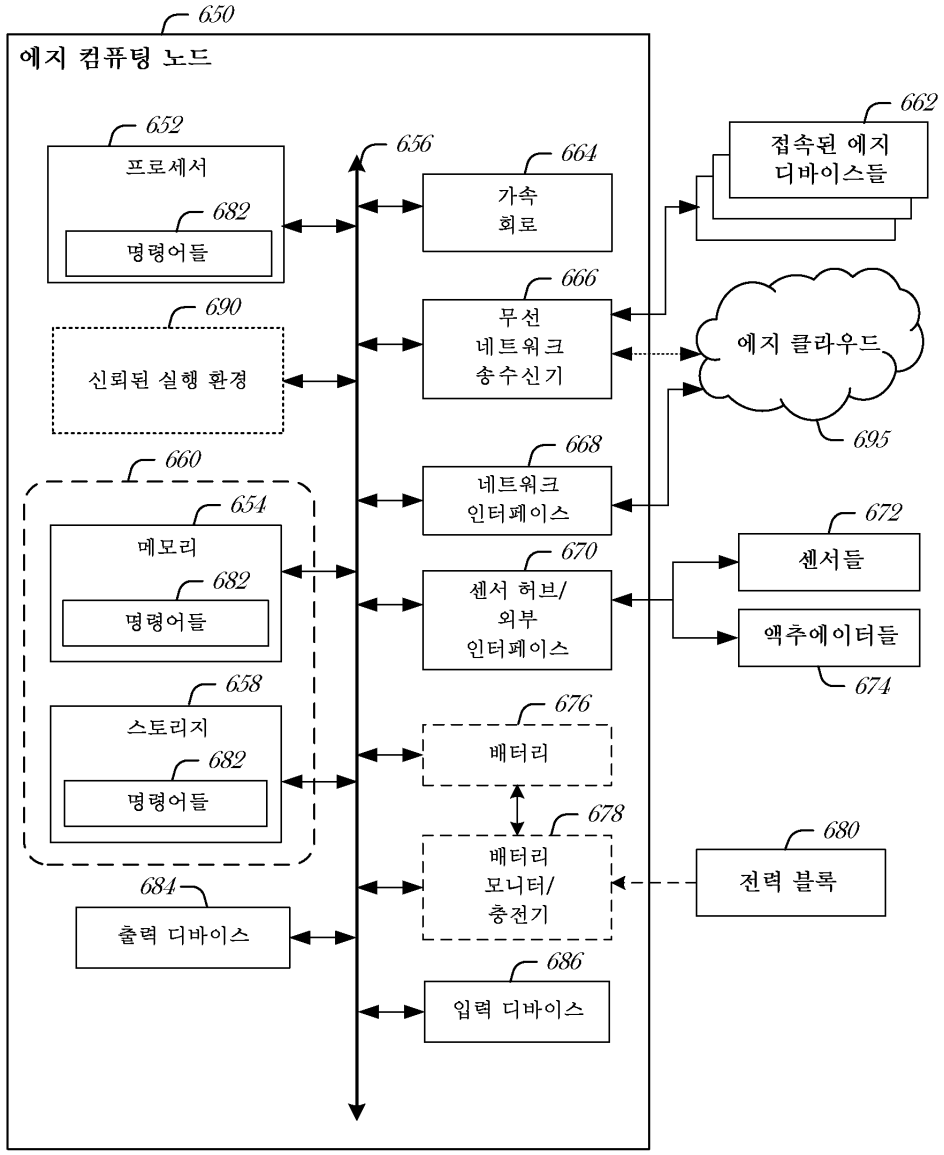
도면5


510
511
512
포드
포드
포드
함수들
FAAS
514
컨테이너 관리자
513
컴퓨팅 노드들
515
520
521
컨테이너 관리자
522
VNF
VNF
컴퓨팅 노드들
523
530
531
컨테이너 오케스트레이션
532
테넌트 1
컨테이너 관리자
포드
포드
포드
함수들
FAAS
533
테넌트 2
컨테이너 관리자
포드
포드
포드
함수들
FAAS
534
VNF들
VM들 테넌트 1
VM들 테넌트 2
535
536
컴퓨팅 노드들
537
540
541
컨테이너 기반 오케스트레이션 시스템
542
VNF/VM
CNF들
앱들
함수들
543
VNF들
CNF들
앱들
함수들
544
컴퓨팅 노드들

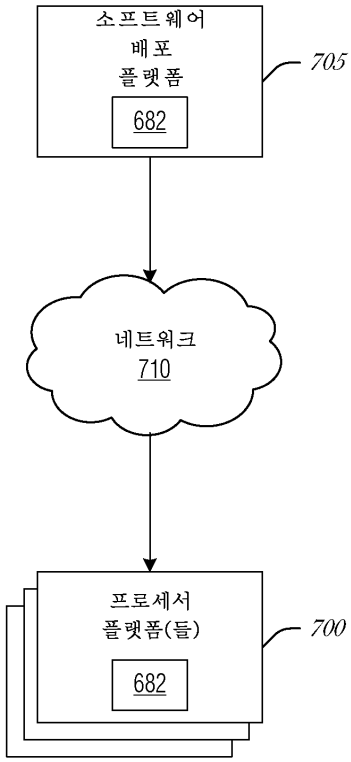
도면6a


600
컴퓨팅 노드
컴퓨팅 회로
604
프로세서
606
메모리
602
I/O 서브시스템
608
데이터 스토리지
610
614
주변
디바이스(들)
통신 회로
서브시스템
612
NIC
620

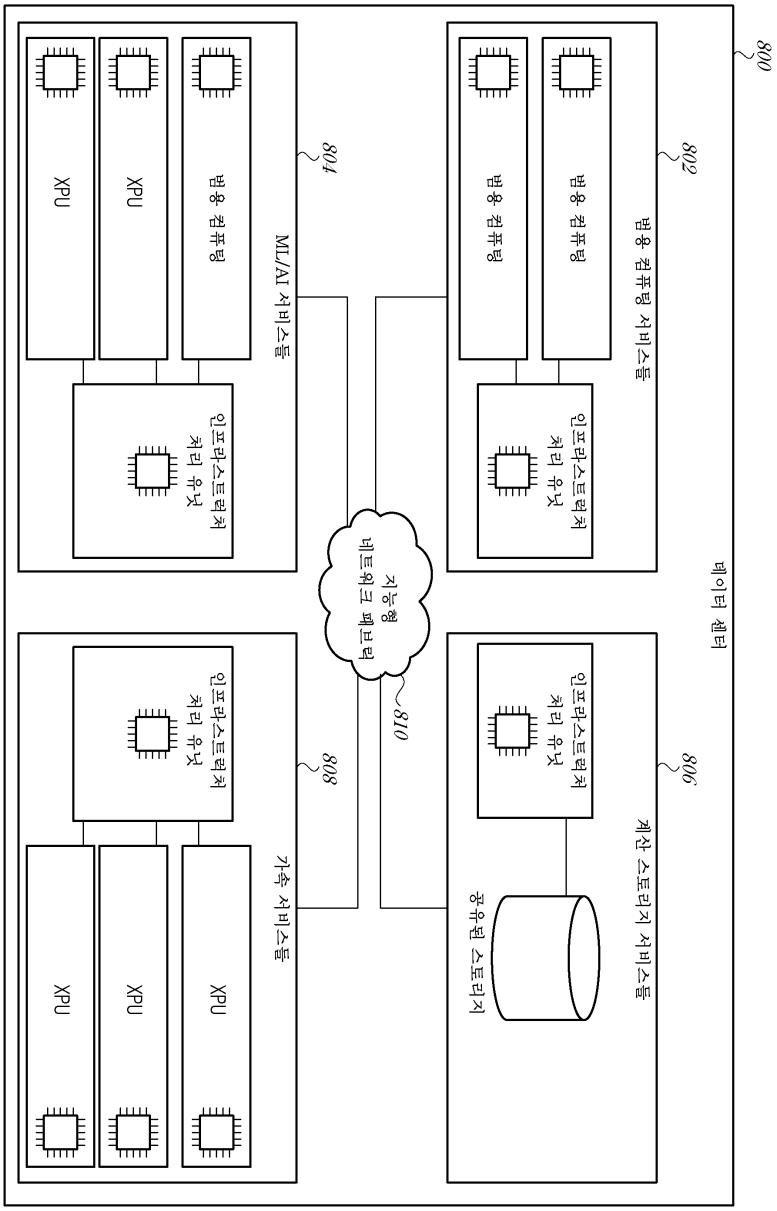
도면6b


650
에지 컴퓨팅 노드
652
프로세서
682
명령어들
656
664
가속
회로
662
접속된 에지
디바이스들
666
무선
네트워크
송수신기
690
신뢰된 실행 환경
에지 클라우드
695
660
668
네트워크
인터페이스
654
메모리
682
명령어들
672
센서들
670
센서 허브/
외부
인터페이스
액추에이터들
674
658
스토리지
682
명령어들
676
배터리
678
배터리
모니터/
충전기
680
전력 블록
684
출력 디바이스
686
입력 디바이스

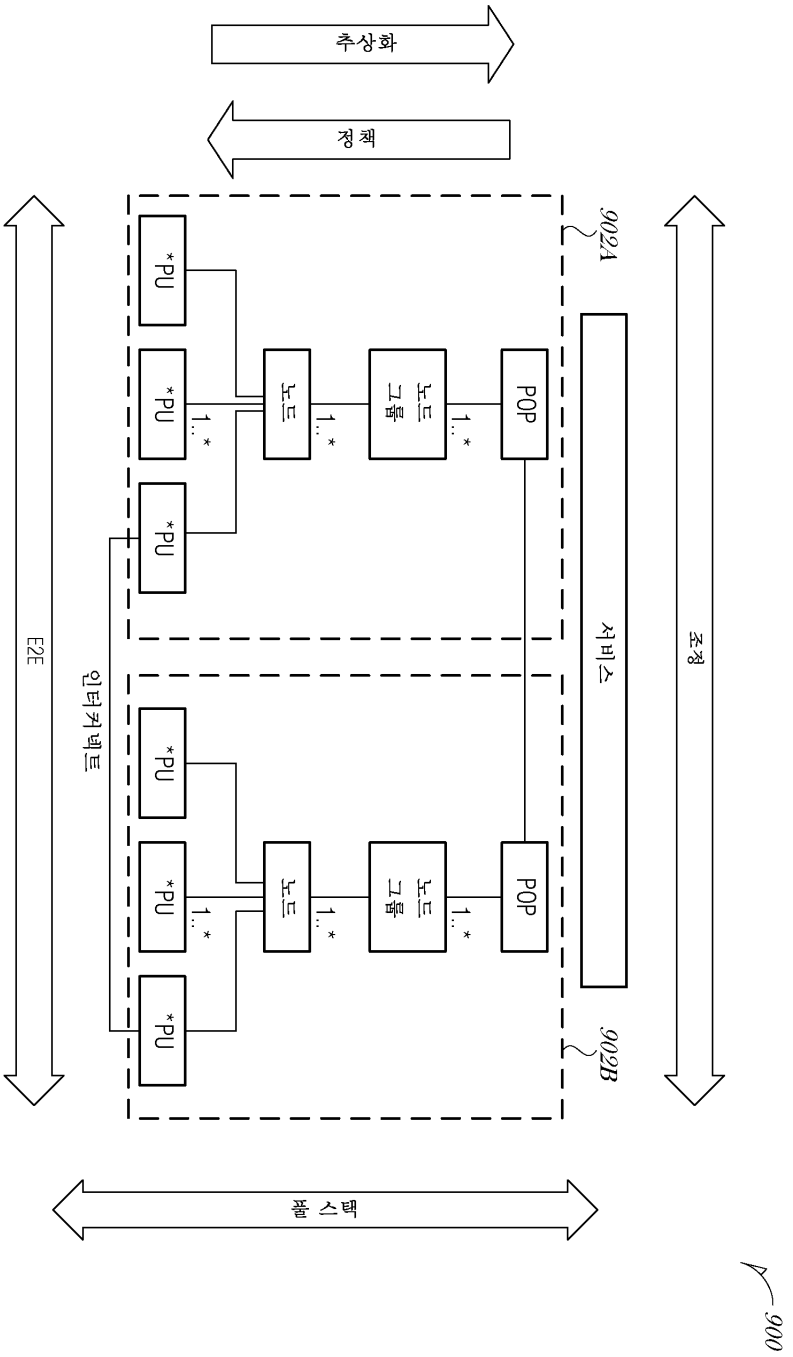
도면7



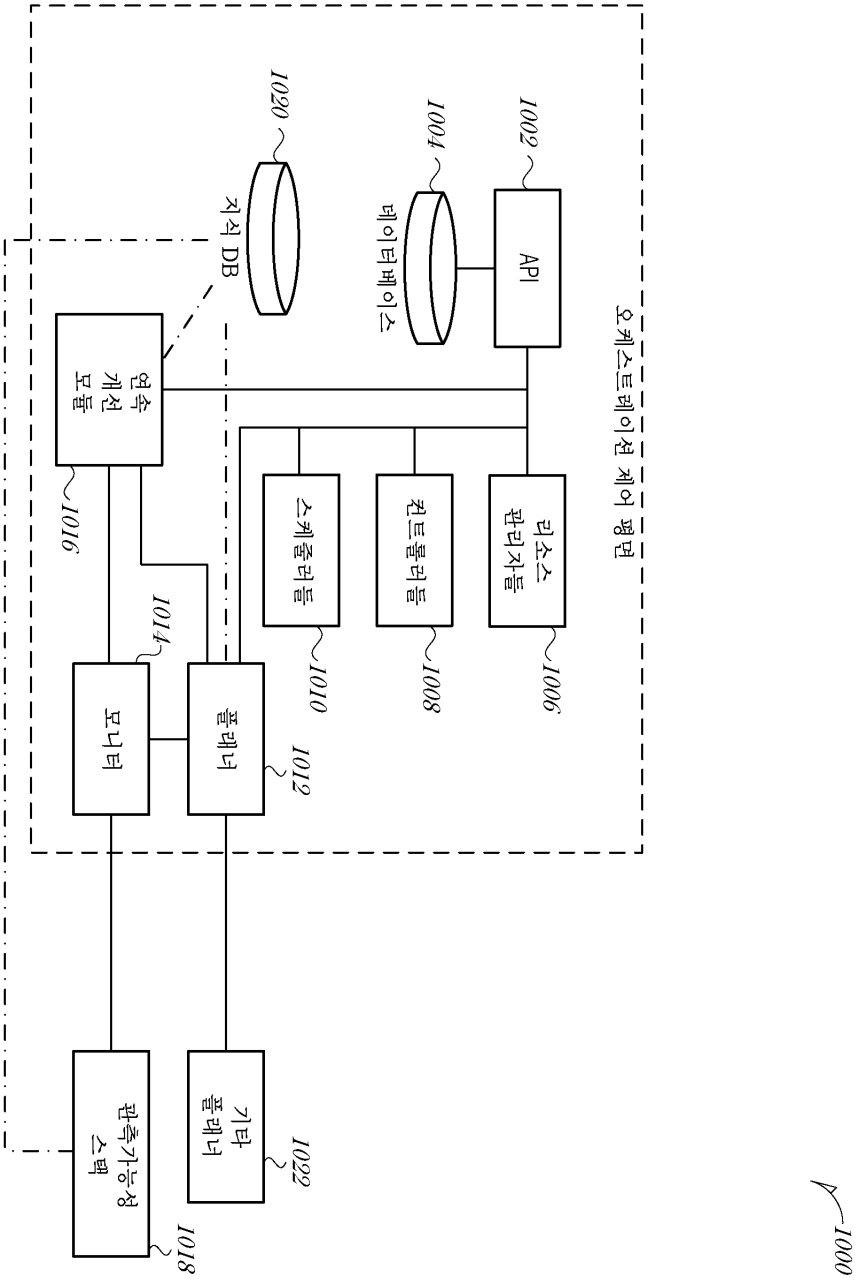
도면8


800
데이터 센터
802
범용 컴퓨팅 서비스들
범용 컴퓨팅
범용 컴퓨팅
인프라스트럭처 처리 유닛
804
ML/AI 서비스들
범용 컴퓨팅
XPU
XPU
인프라스트럭처 처리 유닛
806
계산 스토리지 서비스들
인프라스트럭처 처리 유닛
공유된 스토리지
808
가속 서비스들
XPU
XPU
XPU
인프라스트럭처 처리 유닛
810
지능형 네트워크 패브릭

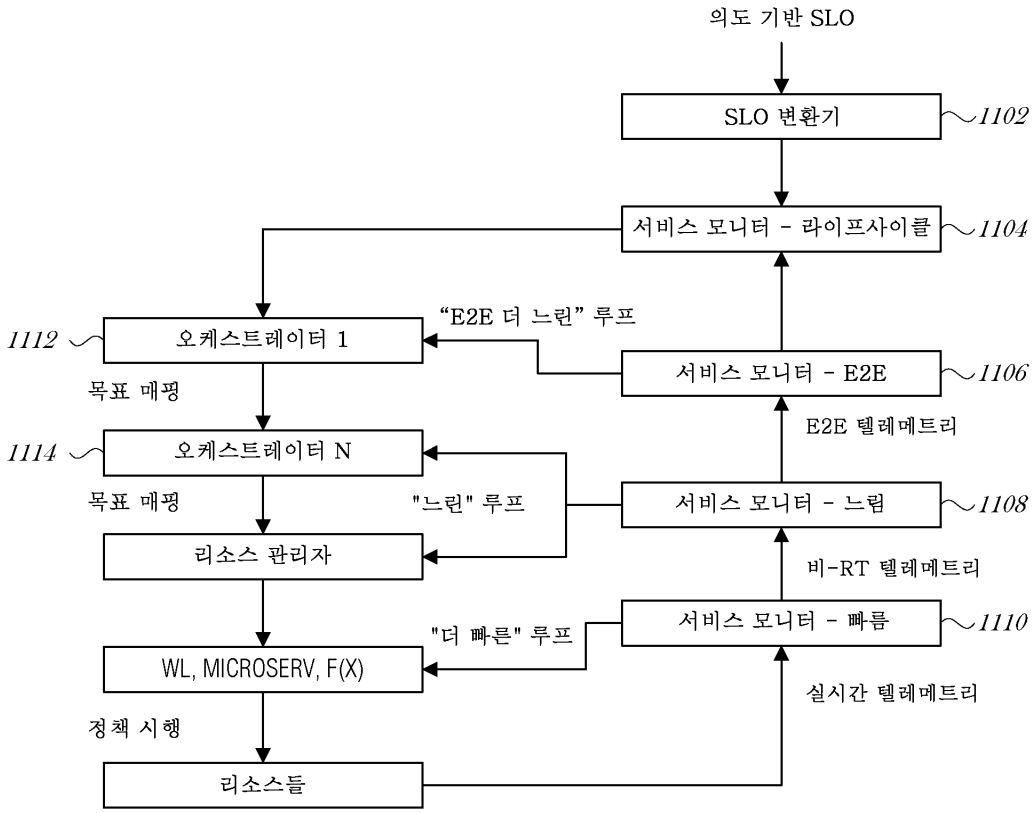
도면9


추상화
정책
902A
902B
*PU
*PU
*PU
노드
노드 그룹
POP
1..*
1..*
1..*
서비스
조정
E2E
인터커넥트
*PU
*PU
*PU
노드
노드 그룹
POP
1..*
1..*
1..*
풀 스택
900

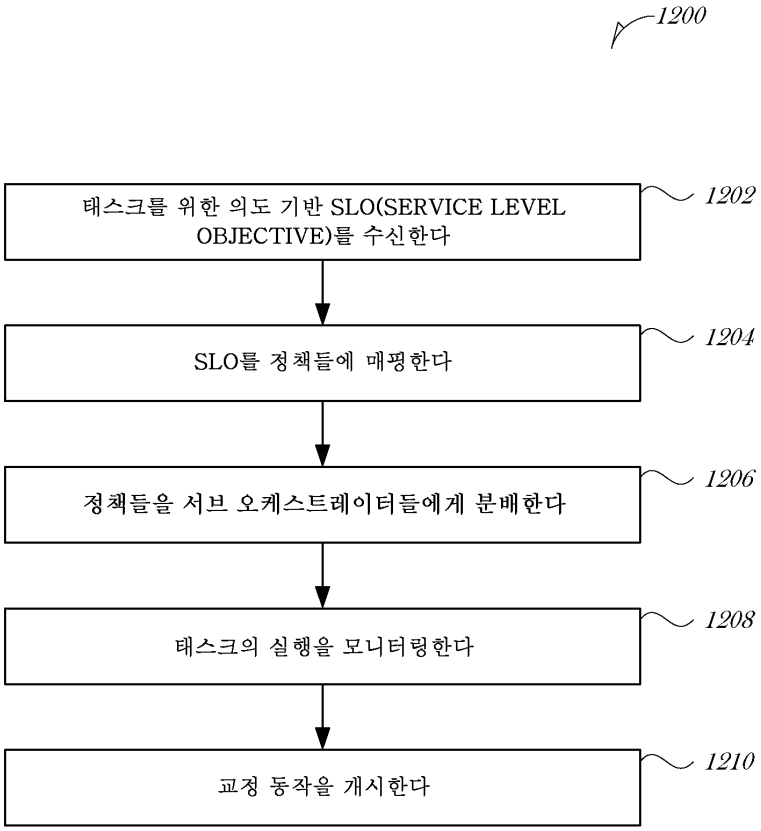
도면10


오케스트레이션 제어 평면
API
데이터베이스
지식 DB
리소스 관리자들
컨트롤러들
스케줄러들
플래너
모니터
연속 개선 모듈
기타 플래너
관측가능성 스택
1000
1002
1004
1006
1008
1010
1012
1014
1016
1018
1020
1022

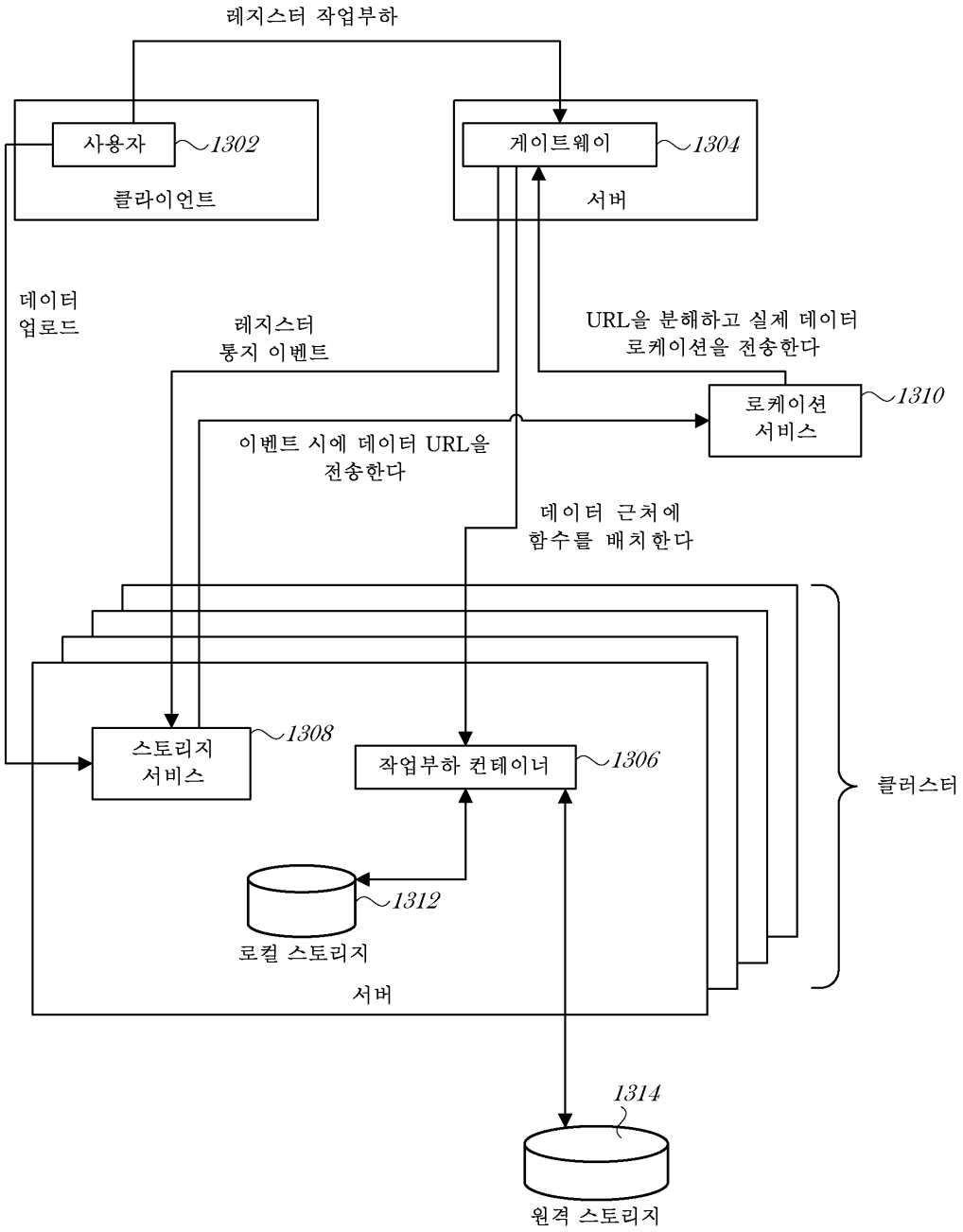
도면11


의도 기반 SLO
SLO 변환기
1102
서비스 모니터 - 라이프사이클
1104
"E2E 더 느린" 루프
1112
오케스트레이터 1
서비스 모니터 - E2E
1106
목표 매핑
E2E 텔레메트리
1114
오케스트레이터 N
목표 매핑
"느린" 루프
서비스 모니터 - 느림
1108
리소스 관리자
비-RT 텔레메트리
"더 빠른" 루프
서비스 모니터 - 빠름
1110
WL, MICROSERV, F(X)
실시간 텔레메트리
정책 시행
리소스들

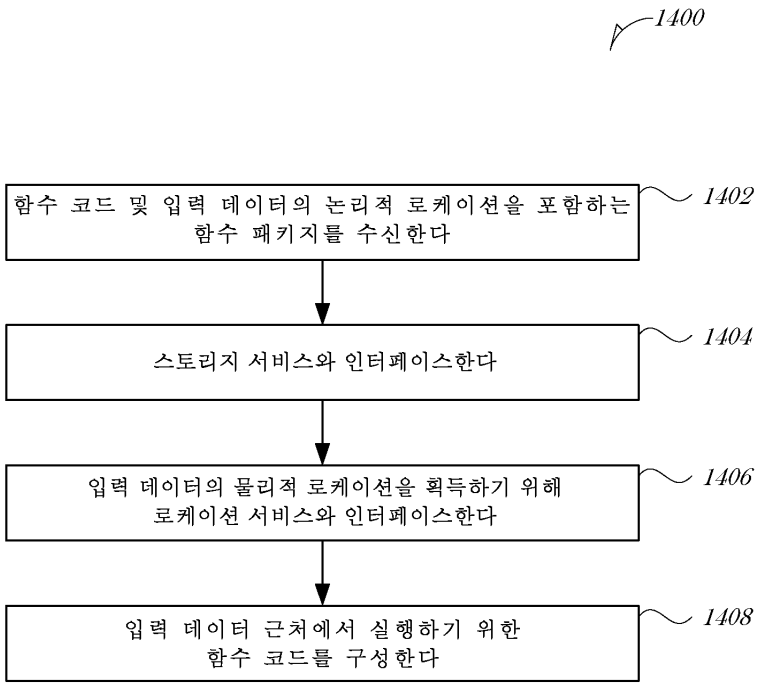
도면12


1200
태스크를 위한 의도 기반 SLO(SERVICE LEVEL OBJECTIVE)를 수신한다
1202
SLO를 정책들에 매핑한다
1204
정책들을 서브 오케스트레이터들에게 분배한다
1206
태스크의 실행을 모니터링한다
1208
교정 동작을 개시한다
1210

도면13


레지스터 작업부하
사용자
1302
클라이언트
게이트웨이
1304
서버
데이터
업로드
레지스터
통지 이벤트
URL을 분해하고 실제 데이터
로케이션을 전송한다
로케이션
서비스
1310
이벤트 시에 데이터 URL을
전송한다
데이터 근처에
함수를 배치한다
스토리지
서비스
1308
작업부하 컨테이너
1306
클러스터
1312
로컬 스토리지
서버
1314
원격 스토리지

도면14


1400
함수 코드 및 입력 데이터의 논리적 로케이션을 포함하는 함수 패키지를 수신한다
1402
스토리지 서비스와 인터페이스한다
1404
입력 데이터의 물리적 로케이션을 획득하기 위해 로케이션 서비스와 인터페이스한다
1406
입력 데이터 근처에서 실행하기 위한 함수 코드를 구성한다
1408

임시명세서(첨부)

아이콘을 더블 클릭하시면 임시명세서 파일이 열립니다.

본 공보 PDF는 첨부파일을 가지고 있습니다. Acrobat Reader PDF뷰어를 제공하지 않는 브라우저(크롬, 파이어폭스, 사파리 등)의 경우 첨부파일 열기가 제한되어 있으므로 Acrobat Reader PDF뷰어 설치 후 공보 PDF를 다운로드 받아 해당 뷰어에서 조회해주시기 바랍니다.