(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년03월10일
(11) 등록번호 10-2372568
(24) 등록일자 2022년03월04일

(51) 국제특허분류(Int. Cl.)
H04L 65/40 (2022.01) *G06F 21/53* (2013.01)
(52) CPC특허분류
H04L 67/2804 (2013.01)
G06F 21/53 (2013.01)
(21) 출원번호 10-2017-7016846
(22) 출원일자(국제) 2015년11월25일
심사청구일자 2020년09월10일
(85) 번역문제출일자 2017년06월19일
(65) 공개번호 10-2017-0107431
(43) 공개일자 2017년09월25일
(86) 국제출원번호 PCT/US2015/062635
(87) 국제공개번호 WO 2016/086111
국제공개일자 2016년06월02일
(30) 우선권주장
62/084,511 2014년11월25일 미국(US)
14/951,223 2015년11월24일 미국(US)
(56) 선행기술조사문헌
US20090172792 A1*
US20130340028 A1*
US20140096221 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
어스0 인코포레이티드
미국 워싱턴 98004 벨뷰 스위트 700 노스이스트 10900 8번가 어스0 인코포레이티드내
(72) 발명자
얀추크, 토마스
미국 워싱턴 98004 벨뷰 수트 204 메인 스트리트 10777 어스0 인코포레이티드 내
워로스키, 마티아스
미국 워싱턴 98004 벨뷰 수트 204 메인 스트리트 10777 어스0 인코포레이티드 내
(74) 대리인
특허법인씨엔에스

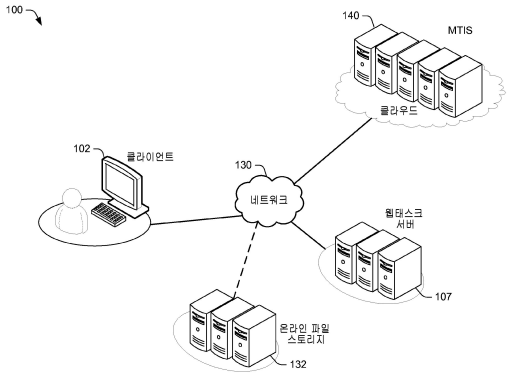
전체 청구항 수 : 총 20 항 심사관 : 이준석

(54) 발명의 명칭 **서버 요청에서 캡슐화된 코드를 통한 멀티테넌시**

(57) 요 약

멀티테넌트 인프라 스트럭처 서버(multitenant infrastructure server(MTIS))는 임의의 애플리케이션의 컴퓨터 루틴을 실행하기 위한 환경을 제공하도록 구성된다. MTIS는, 웹태스크 컨테이너(webtask container) 내에서 컴퓨터 루틴을 실행하기 위한 요청을 웹태스크 서버로부터 수신한다. 컴퓨터 루틴은 상기 MTIS에서 웹태스크 컨테이너 내에서 실행된다. 컴퓨터 루틴의 성공적인 실행에 따라, 결과 세트가 웹태스크 서버에 리턴된다. 컴퓨터 루틴의 실행이 성공적이지 않으면, 에러 통지가 웹태스크 서버로 리턴된다. 컴퓨터 루틴의 실행 동안 소비된 리소스가 판단된다. 웹태스크 컨테이너는 MTIS 상에서 컴퓨터 루틴의 지속적인 저장을 방지하기 위하여 파기된다.

대 표 도



(52) CPC특허분류
H04L 67/02 (2013.01)
H04L 67/1097 (2013.01)

명 세 서

청구범위

청구항 1

임의의 애플리케이션의 컴퓨터 루틴을 실행하기 위한 환경을 제공하도록 구성된 멀티테넌트 인프라 스트럭처 서버(multitenant infrastructure server(MTIS))에 있어서,

프로세서;

상기 프로세서에 결합되고 통신 네트워크를 통한 통신을 가능하게 하도록 구성된 네트워크 인터페이스;

콘텐츠와 프로그래밍을 위한 저장 장치; 및

상기 저장 장치에 저장된 프로그램

을 포함하고,

상기 프로그램의 실행은,

웹태스크 컨테이너(webtask container) 내에서 상기 컴퓨터 루틴을 실행하기 위한 요청을 웹태스크 서버로부터 수신하는 단계로서, 상기 요청은,

클라이언트 데이터;

실행 가능한 컴퓨터 코드 형태의 상기 컴퓨터 루틴 또는 상기 컴퓨터 루틴에 대한 URL(uniform resource identifier) 링크; 및

상기 컴퓨터 루틴과 연관된 적어도 하나의 클라이언트 비밀(secret)

을 포함하는 단계;

상기 MTIS에서 상기 웹태스크 컨테이너 내에서 상기 컴퓨터 루틴을 실행시키는 단계;

상기 컴퓨터 루틴의 성공적인 실행에 따라, 상기 웹태스크 서버에 결과 세트를 리턴하는 단계;

상기 컴퓨터 루틴의 성공적이지 않은 실행에 따라, 상기 웹태스크 서버로 에러 컴퓨터 루틴을 리턴하는 단계;

상기 컴퓨터 루틴의 실행 동안 소비된 리소스를 판단하는 단계로서,

상기 웹태스크 컨테이너와 연관된 요청 식별자(ID)를 획득하는 단계; 및

요청당(per request) 기반으로 소비된 상기 리소스를 계량하도록 상기 요청 ID를 스레드(thread) ID와 연관시키는 단계

를 포함하는 단계; 및

상기 MTIS 상에서 상기 컴퓨터 루틴의 지속적인 저장을 방지하기 위하여 상기 웹태스크 컨테이너를 파기하는 단계

를 포함하는 동작을 수행하도록 상기 MTIS를 구성하는,

MTIS.

청구항 2

제1항에 있어서,

상기 요청은 웹태스크 토큰을 포함하는,

MTIS.

청구항 3

삭제

청구항 4

제2항에 있어서,

상기 프로그래밍의 실행은, 상기 웹태스크 토큰에 제공된 상기 URL로부터 컴퓨터 루틴을 검색하는 단계를 포함하는 동작을 수행하도록 상기 MTIS를 구성하는,

MTIS.

청구항 5

삭제

청구항 6

제2항에 있어서,

상기 웹태스크 토큰은 부당 변경(tampering)과 공개(disclosure)로부터 암호로 보호되는,

MTIS.

청구항 7

제1항에 있어서,

상기 웹태스크 컨테이너는 상기 MTIS 내의 분리된 환경 내에서 실행되기 위한 상기 컴퓨터 루틴의 모든 종속성(dependency)을 포함하는,

MTIS.

청구항 8

제7항에 있어서,

상기 분리된 환경은 상기 MTIS 내의 샌드박스(sandbox)인,

MTIS.

청구항 9

제1항에 있어서,

소비된 상기 리소스는 상기 컴퓨터 루틴의 실행 동안 사용된 처리 능력(processing power) 및 메모리를 포함하는,

MTIS.

청구항 10

제1항에 있어서,

상기 웹태스크 컨테이너를 파기하는 단계는, 상기 결과 세트 또는 상기 에러 컴퓨터 루틴이 상기 웹태스크 컨테이너에 의해 수신되었다는 상기 웹태스크 서버로부터의 확인을 수신한 것에 응답하는,

MTIS.

청구항 11

제1항에 있어서,

상기 컴퓨터 루틴은 상기 MTIS의 운영 체제에 독립적인,

MTIS.

청구항 12

제1항에 있어서,

상기 컴퓨터 루틴을 실행하기 위한 시간은 HTTP 요청/응답 사이클에 대하여 걸린 시간의 기간에 한정되는,

MTIS.

청구항 13

제12항에 있어서,

상기 웹태스크 서버는 HTTP 서버이고, 상기 요청은 HTTP 요청인,

MTIS.

청구항 14

제1항에 있어서,

리턴된 상기 결과 세트 및 상기 에러 컴퓨터 루틴은 JavaScript Object Notation(JSON) 형태를 갖는,

MTIS.

청구항 15

제1항에 있어서,

상기 MTIS는, 한 테넌트의 컴퓨터 루틴이 다른 테넌트의 컴퓨터 루틴 또는 데이터를 액세스하는 것을 방지하도록 각각의 테넌트에 대한 데이터 분리를 제공하는,

MTIS.

청구항 16

임의의 애플리케이션의 컴퓨터 루틴을 실행하는 방법에 있어서,

멀티테넌트 인프라 스트럭처 서버(multitenant infrastructure server(MTIS))의 웹태스크 컨테이너(webtask container) 내에서 상기 컴퓨터 루틴을 실행하기 위한 요청을 웹태스크 서버로부터 수신하는 단계로서, 상기 요청은 웹태스크 토큰 및 클라이언트 데이터를 포함하고, 상기 웹태스크 토큰은,

실행 가능한 컴퓨터 코드 형태의 상기 컴퓨터 루틴 또는 상기 컴퓨터 루틴에 대한 URL(uniform resource identifier) 링크; 및

상기 컴퓨터 루틴과 연관된 적어도 하나의 클라이언트 비밀(secret)

을 포함하는 단계;

상기 MTIS에서 상기 웹태스크 컨테이너 내에서 상기 컴퓨터 루틴을 실행시키는 단계;

상기 컴퓨터 루틴의 성공적인 실행에 따라, 상기 웹태스크 서버에 결과 세트를 리턴하는 단계;

상기 컴퓨터 루틴의 성공적이지 않은 실행에 따라, 상기 웹태스크 서버로 에러 통지를 리턴하는 단계;

상기 컴퓨터 루틴의 실행 동안 소비된 리소스를 판단하는 단계로서,

상기 웹태스크 컨테이너와 연관된 요청 식별자(ID)를 획득하는 단계; 및

요청당(per request) 기반으로 소비된 상기 리소스를 계량하도록 상기 요청 ID를 스레드(thread) ID와 연관시키는 단계

를 포함하는 단계; 및

상기 MTIS 상에서 상기 컴퓨터 루틴의 지속적인 저장을 방지하기 위하여 상기 웹태스크 컨테이너를 파기하는 단

계

를 포함하는,

임의의 애플리케이션의 컴퓨터 루틴을 실행하는 방법.

청구항 17

삭제

청구항 18

제16항에 있어서,

상기 웹태스크 토큰에 제공된 상기 URL로부터 컴퓨터 루틴을 검색하는 단계를 더 포함하는,

임의의 애플리케이션의 컴퓨터 루틴을 실행하는 방법.

청구항 19

삭제

청구항 20

제16항에 있어서,

상기 웹태스크 컨테이너는 상기 MTIS 내의 분리된 환경 내에서 실행되기 위한 상기 컴퓨터 루틴의 모든 종속성(dependency)을 포함하는,

임의의 애플리케이션의 컴퓨터 루틴을 실행하는 방법.

청구항 21

제16항에 있어서,

상기 웹태스크 컨테이너를 파기하는 단계는, 상기 결과 세트 또는 상기 에러 통지가 상기 웹태스크 컨테이너에 의해 수신되었다는 상기 웹태스크 서버로부터의 확인을 수신한 것에 응답하는,

임의의 애플리케이션의 컴퓨터 루틴을 실행하는 방법.

청구항 22

제16항에 있어서,

상기 컴퓨터 루틴을 실행하기 위한 시간은 HTTP 요청/응답 사이클에 대하여 걸린 시간의 기간에 한정되고;

상기 웹태스크 서버는 HTTP 서버이고; 그리고,

상기 요청은 HTTP 요청인,

임의의 애플리케이션의 컴퓨터 루틴을 실행하는 방법.

청구항 23

제16항에 있어서,

리턴된 상기 결과 세트 또는 상기 에러 통지는 JavaScript Object Notation(JSON) 형태를 갖는,

임의의 애플리케이션의 컴퓨터 루틴을 실행하는 방법.

청구항 24

제16항에 있어서,

한 테넌트의 컴퓨터 루틴이 다른 테넌트의 컴퓨터 루틴 또는 데이터를 액세스하는 것을 방지하도록 각각의 테넌트에 대한 데이터 분리를 제공하는 단계를 더포함하는,

임의의 애플리케이션의 컴퓨터 루틴을 실행하는 방법.

발명의 설명

기 술 분 야

배 경 기 술

[0001] 컴퓨터 애플리케이션 아키텍처는 다중 컴퓨터에 걸쳐 상이한 컴퓨팅 기능을 분산하는 방향으로 흘러 왔다. 사실, 대부분의 현대 모바일 및 웹 애플리케이션은 분산 아키텍처에 기초한다. 예를 들어, 클라이언트-서버 아키텍처에서, 프론트엔드(frontend)와 백엔드(backend) 사이의 애플리케이션 기능의 분리는 여러 클라이언트에 걸쳐 백엔드 컴퓨팅 리소스를 재사용하는 것을 돕는다. 또한, 이는 클라이언트와 서버 사이의 신뢰 경계(trust boundary)를 형성하고, 이는 서버가 데이터 또는 기능을 보호하기 위한 액세스를 인가할 수 있게 한다. 전형적인 클라이언트-서버 애플리케이션에서, 클라이언트는 자신을 인증한 후에 처리를 위한 데이터를 백엔드에 제출하고, 백엔드는 보호된 리소스를 이용하여 클라이언트 요청의 처리 후에 응답한다.

[0002] 애플리케이션(예를 들어, 모바일 또는 웹)의 전형적인 클라우드 백엔드는 클라이언트와는 상이한 운영 체제 및 애플리케이션 프레임워크 능력뿐만 아니라 원시(raw) 컴퓨팅 리소스(예를 들어, CPU, 메모리, 디스크, 네트워크 등)를 제공한다. 백엔드는 애플리케이션 로직의 일부를 구현하는 서버 코드와, 이 코드가 데이터베이스 연결 스트링, 애플리케이션 프로그램 인터페이스(application program interfaces(API)), 보안 키 등과 같은 보호된 데이터 또는 기능을 액세스하는데 필요로 하는 비밀(secret)을 캡슐화한다.

[0003] 백엔드 코드를 호스팅하는 인프라 스트럭처(infrastructure)를 관리하는 것은 다양한 서버의 크기를 설정하고, 이의 권한을 설정하고, 이의 크기를 조정하며, 운영 체제 업데이트를 관리하고, 보안 문제를 다루고, 필요에 따라 하드웨어를 업데이트하고, 가능성 있는 오동작에 대하여 모든 이러한 요소들을 모니터링하는 것을 포함할 수 있다. 따라서, 전형적으로 백엔드를 관리하는 것의 실행 계획에만 많은 노력이 소요된다. 이 노력은 컴퓨터 애플리케이션을 개발하고, 최적화하고 그리고/또는 효율적으로 사용하는데 더 많이 소요될 수 있다.

[0004] 수년간, 정보 기술(Information Technology(IT)) 비용을 감소시키고 서버 컴퓨팅 능력을 편리한 것/유용한 것으로서 사용 가능한 것으로 만들기 때문에, 클라우드 컴퓨팅이 증가하여 왔다. 이전에, 비용을 줄이기 위한 주된 접근 방식은 서버 컴퓨팅 기능을 클라우드 컴퓨팅 벤더로 아웃소싱함으로써 IT 스태프를 줄이는 것이었다. 그러나, 현재, 여러 경쟁력 있는 클라우드 컴퓨팅 벤더가 있으며, 따라서 비용 감소가 이제 본질적으로 주요한 기술이다.

[0005] 비용을 줄이기 위한 하나의 기술적 접근 방식은 애플리케이션 밀도를 증가시키는 것이다. 구체적으로는, 애플리케이션을 호스팅하는 것은 메모리 및 CPU와 같은 리소스 비용을 가진다. 애플리케이션들에 걸쳐 이러한 리소스 비용을 공유하는 방법이 있다면, 이 리소스 비용은 이 애플리케이션들에 걸쳐 분산될 수 있다. 따라서, 가상 기계 리소스를 공유하기 위하여 멀티테넌시(multi-tenancy) 기술이 생기게 되어, 애플리케이션 밀도를 증가시킨다.

[0006] 물리적 기계를 할당하고 이의 권한을 설정하기 위한 비용은 가상 기계를 할당하고 이의 권한을 설정하기 위한 비용보다 더 크다. 다음으로, 가상 기계를 할당하고 이의 권한을 설정하기 위한 비용은 멀티테넌트(multi-tenant) 컨테이너(container)를 할당하고 이의 권한을 설정하기 위한 비용보다 더 크다. 마지막으로, 컨테이너에서 프로세스를 실행하기 위한 비용은 스레드(thread)를 실행하기 위한 비용보다 더 비싸다. 이상적으로, 일종의 경량 웹 애플리케이션에 대하여, 애플리케이션 밀도는 최대화될 수 있지만, 각각의 애플리케이션을 스레드당 기반으로 실행시킨다. 그러나, 운영 체제는 프로세스가 리소스를 관리할 수 있게 하지만, 이는 리소스를 스레드 레벨에서 관리하기 위한 적합한 기능을 제공하지 않는다. 구체적으로, 상이한 테넌트(tenant)들의 정보 자산은 멀티테넌트 컨테이너에서와 같이 서로 분리되어야 하며, 리소스 사용은 서비스 품질을 유지하고 클라우드 컴퓨팅 벤더에 의한 청구(billing)를 가능하게 하도록 관리되고 측정되어야 한다.

[0007] 앱을 개발하고 런칭하는 것과 전형적으로 연관된 인프라 스트럭처를 구축하고 관리하는 복잡성 없이 고객이 웹 애플리케이션을 개발하고, 운영하고, 관리할 수 있게 하는 PaaS(platform as a service) 솔루션이 존재하지만, 이는 다양한 우려를 가지고 있다. 예를 들어, 알려진 PaaS 플랫폼은 매력적인 비용 구조를 제공하지 않을 수 있고, 비동기 프로그래밍 모델 상에서 실행될 수 있어 대기 시간(latency)에 악영향을 미치는 계산 결과에 대한

폴링(polling)을 필요로 한다. 또한, 알려진 PaaS 아키텍처는 업로드되어야 할 뿐만 아니라 지속적으로 저장되어야 하는 코드를 필요로 할 수 있다. 그 다음, 코드는 이의 작업을 완료하기 위하여 이벤트를 대기한다. 그러나, 이러한 접근 방식은 코드가 다른 곳에서 관리되는 보안 위험을 포함하여, 복제 및 해킹에 취약하게 한다. 본 개시 내용이 작성된 것은 이러한 고려 사항 및 다른 고려 사항에 관한 것이다.

도면의 간단한 설명

[0008] 도면은 본 교시 내용에 따라 하나 이상의 구현예를 한정이 아닌 예로서만 도시한다. 도면에서, 유사한 도면 부호는 동일하거나 유사한 요소를 나타낸다.

도 1은 안전한 멀티테넌트 환경에서 임의의 애플리케이션의 코드를 실행하기 위한 예시적인 아키텍처를 도시한다.

도 2는 멀티테넌트 컨테이너에서 코드를 실행시키기 위한 데이터 프레임워크의 블록도를 도시한다.

도 3은 도 1의 멀티테넌트 인프라 스트럭처로서 사용될 수 있는 예시적인 웹태스크(webtask) 가상 기계를 도시한다.

도 4는 멀티테넌트 인프라 스트럭처 환경에서 코드를 실행시키기 위한 상위 레벨의 예시적인 호출 흐름 프로세스를 도시한다.

도 5는 클라우드 상에 있을 수 있는 네트워크 또는 호스트 컴퓨터를 도시한다.

도 6은 사용자 인터페이스 요소를 갖는 컴퓨터를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0009] 본 개시 내용은 일반적으로 가상 환경에서 컴퓨터 루틴을 실행하는 방법 및 시스템에 관한 것이다. 본 명세서에서 논의되는 컴퓨팅 환경은 컴퓨터 개발자로부터 컴퓨터 코드의 형태로 컴퓨터 루틴을 공급받는다. 루틴에서 어떤 컴퓨팅 언어 또는 언어들이 사용되고 있는지에 대한 판단이 이루어진다. 컨테이너는, 루틴 및 루틴에 의해 활용되거나 호출되는 임의의 다른 인프라 스트럭처에서 언어를 지원하도록 권한 설정되도록, 루틴에 대하여 특별히 형성된다. 따라서, 컨테이너는 예를 들어 코드나 그에 대한 링크를 실행하기 위한 요소, 시스템 도구, 시스템 라이브러리 등을 포함하는 완전한 환경 내에서 루틴을 감추어, 이에 의해 루틴이 이의 가상 목적지 환경에서 실행될 것임을 보장한다. 가상 목적지 환경은, 메모리, 처리 능력(processing power), 네트워킹 등을 포함하는, 루틴을 실행하는데 필요한 원시 리소스(raw resource)를 제공한다. 알려진 접근 방식과는 다르게, 루틴은 영구 저장을 위하여 업로드되지 않고, 이벤트에 매핑되지 않으며, 목적지 컴퓨팅 환경(즉, 가상 기계)에 영구 저장되지 않는다. 오히려, 루틴은 코드가 다시 실행될 것이라는 예상 없이 실행을 위한 대응하는 환경을 갖는 가상 기계로 라우팅된다. 따라서, 코드는 영구 저장되지 않고, 코드의 실행 완료에 따라 파기된다.

[0010] 루틴을 실행시키기 위한 요청은 가상 목적지 환경의 게스트(guest) 운영 체제에 독립적이란 점에서 임의의 애플리케이션을 위한 것일 수 있다. 컴퓨터 루틴 실행 요청은 멀티테넌시 인프라 스트럭처에서 처리될 수 있어, 이에 의해 분리(isolation) 및 측정(measuring) 능력을 제공한다. 요청은, 멀티테넌시 인프라 스트럭처에 대한 언어 바인딩(binding)이 사용 가능하다면, 임의의 프로그래밍 언어로 될 수 있다.

[0011] 유익하게는, 사용자 자신의 컴퓨터(들) 상에 컴퓨터 루틴(예를 들어, 애플리케이션)을 설치하고 실행시키는 필요성은 이에 의해 불필요하게 되고, 이는 메인터넌스, 확장성(scalability), 보안 및 지원을 단순화한다. 또한, 일 실시예에서, 클라우드 컴퓨팅의 사용은 선행 투자 인프라 스트럭처 비용을 방지하는데 도움을 주고 비즈니스가 개선된 통제 능력 및 더 적은 메인터넌스를 가지면서 코드가 더 빠르게 실행될 수 있게 한다. 실행될 코드가 이의 컨테이너에서 분리되고, 컨테이너가 실행 완료에 따라 파기되기 때문에, 추가적인 보안이 제공된다.

[0012] *예시적인 시스템 아키텍처*

[0013] 도 1은 안전한 멀티테넌트 환경에서 임의의 애플리케이션의 루틴을 실행시키기 위한 예시적인 아키텍처를 도시한다. 시스템(100)은 클라이언트(102), 웹태스크 서버(107), 다수의 컨테이너를 포함할 수 있는 멀티테넌트 인프라 스트럭처 서버(multi-tenant infrastructure server(MTIS))(140) 및 온라인 파일 저장 서버(132)를 포함한다. 시스템(100)의 다른 컴포넌트들 사이의 통신을 가능하게 하는 네트워크(130)가 있다. 네트워크(120)는, 한정이 아닌 것으로, 근거리 통신망(local area network(LAN)), 가상 사설망(virtual private network(VPN)), 이동 통신망 또는 인터넷일 수 있다. 따라서, 클라이언트는 네트워크(130)를 통해 웹태스크 서버(107)와 통신하

여, 처리될 임의의 애플리케이션의 컴퓨터 루틴을 전송하고 대응하는 처리 결과를 그로부터 수신할 수 있다. 일 실시예에서, 루틴을 웹태스크 서버(107)로 전송하는 대신에, 온라인 파일 저장 서버(132)와 같은 루틴의 코드가 저장되는 저장소(repository)로 링크가 전송된다.

[0014] 웹태스크 서버(107)는 클라이언트(102)로부터 정보를 수신하고, 인프라 스트럭처를 포함하는 완전한 패키지에 루틴을 래핑(wrap)하여 MTIS(140)에서 분리를 실행하도록 구성된다. 그 다음, 웹태스크 서버(107)는 "패키지(package)"를 MTIS(140)로 전송한다. 웹태스크 서버(107)는 MTIS(140)로부터 컴퓨터 루틴에 의해 사용될 모든 인프라 스트럭처를 갖는 컨테이너를 요청한다. 그 다음, 웹태스크 서버(107)는 컴퓨터 루틴을 메타 데이터와 함께 패키지의 형태로 MTIS(140)에 의해 특정된 컨테이너로 발송한다. 그 다음, 컨테이너는 메타 데이터에 따라 패키지 내의 루틴을 실행한다. 컨테이너는 루틴을 실행시킨 결과를 웹태스크 서버(107)로 다시 리턴한다. 그 다음, 웹태스크 서버(107)는 결과를 클라이언트(102)로 (예를 들어, 이의 브라우저 상에) 다시 리턴한다.

[0015] MTIS(140)는 클라우드에서 동작할 수 있고, 웹태스크 서버(107)로부터 수신된 컴퓨터 루틴을 실행하는데 사용될 수 있는 원시 컴퓨팅 리소스를 포함할 수 있다. 실행될 이 루틴은 MTIS(140)에 저장되지 않는다는 점에서 비지속적이다. 대신에, 루틴은 컴퓨터 루틴의 실행 완료에 따라 폐기된다.

[0016] *예시적인 기능 블록*

[0017] 이제, 멀티테넌트 컨테이너에서 코드(예를 들어, 컴퓨터 루틴)를 실행하기 위한 데이터 프레임워크의 블록도를 도시하는 도 2를 참조한다. 시스템(100)은 네트워크(130)를 통해 서버(104)와 통신하는 클라이언트(102)를 보여준다.

[0018] 서버(140)는 클라우드에서 동작할 수 있고, CPU, 메모리, 디스크 등과 같은 원시 컴퓨팅 리소스(142)와, 운영 체제(144)를 포함할 수 있다. 따라서, 서버(140)는 클라우드 상품(cloud commodity)으로서 보일 수 있는 원시 클라우드 컴퓨팅 리소스(142)와 운영 체제를 제공한다. 서버(140)는 실행될 컴퓨터 루틴의 형태로 애플리케이션에 백엔드를 제공한다. 서버(140)로부터 눈에 띄게 빠져 있는 것은 애플리케이션 로직의 일부를 구현하는 서버 코드와, 이 코드가 데이터베이스 연결 스트링, API 키 등과 같은 보호된 데이터 또는 기능을 액세스하는데 필요로 하는 비밀(secret)이다. 서버 코드 및 비밀 양자는 하나의 번들(108)로 함께 직렬화되고, 이제 클라이언트(102)에서 발견될 수 있는 데이터이다. 또한, 일 실시예에서, 클라이언트(102)의 번들(108) 내에 모든 서버 코드를 저장하는 대신에, 코드는 GitHub® 또는 아마존 심플 스토리지 서비스(Amazon Simple Storage Service(S3))와 같은 URL(uniform resource identifier)(110)로 참조될 수 있는 위치로 구체화된다. 따라서, 코드는 온라인 파일 스토리지 서버(132)로 링크될 수 있다.

[0019] 임의의 애플리케이션을 위한 코드(예를 들어, 컴퓨터 루틴)(또는 그에 대한 URL 링크)(110)와 클라이언트 비밀(112)을 포함하는 번들(108)은 본 명세서에서 실행을 위하여 비밀과 함께 백엔드 애플리케이션 로직을 정의하는 웹태스크 토큰(webtask token)(110)이라 한다. 일 실시예에서, 이는 부당 변경(tampering)과 공개(disclosure)로부터 암호로 보호된다(114). 이는 네트워크(130)와 같은 신뢰할 수 없는 채널을 통해 안전하게 저장되거나 전달될 수 있다.

[0020] 웹태스크 서버(예를 들어, 도 1에서의 107)가 웹태스크 토큰(110)을 해독하고 이미 해독된 자료를 서버(140)(예를 들어, MTIS)에 제공한다. 예를 들어, 서버(140)(예를 들어, MTIS) 내의 웹태스크 컨테이너가 신뢰할 수 없는 코드를 실행하고 따라서 웹태스크 토큰(110)을 해독하기 위한 암호 키를 제공하지 않기 때문에, 해독은 웹태스크 서버(107)에 의해 수행된다.

[0021] 웹태스크 토큰(108)이 코드(예를 들어, 컴퓨터 루틴) 자체가 아닌 서버 코드에 대한 URL 링크(110)를 포함할 수 있기 때문에, 토큰의 직렬화된 크기는 오늘날의 대역폭 기준을 고려하면 상대적으로 작다. 따라서, 웹태스크 토큰(108)은 하이퍼텍스트 전송 프로토콜(hypertext transfer protocol(HTTP))을 포함하는 다양한 프로토콜에서 페이로드(payload)의 일부로서 전달될 수 있는 유연성을 제공한다. 도 1의 예에서, 웹태스크 토큰은 클라이언트(102) 애플리케이션 자체에 저장된다. 따라서, 클라이언트(102) 엔드에서 웹태스크 토큰(108)에 포함된 백엔드 비밀(112)을 유지함으로써, 비밀은 암호화(114) 덕분에 공개로부터 안전하게 유지된다. 따라서, 웹태스크 토큰은 부당 변경과 위장(spoofing)에 저항력이 있다.

[0022] 임의의 애플리케이션을 위한 컴퓨터 루틴을 실행하기 위하여 요청(124)이 클라이언트(102)에서 시작되고 네트워크(130)를 통해 서버(140)로 전송될 때, 요청(124)은 웹태스크 요청을 생성하는 웹태스크 토큰(108) 및 클라이언트 전용 데이터(106)를 포함할 수 있다. 따라서, 웹태스크 요청은 보통의 클라이언트 요청 데이터(106)에 더하여 웹태스크 토큰(110)을 포함하는 클라이언트(102)로부터의 요청이다.

[0023] 일 실시예에서, 서버(140)는 웹태스크 요청(124)(즉, 웹태스크 토큰(108)과 클라이언트 데이터(106)를 포함함)를 수신하고, 웹태스크 토큰(108)에 제공된 URL(100)에 기초하여 온라인 파일 스토리지(132)로부터 컴퓨터 루틴을 검색하고, 웹태스크 요청을 실행하기 위하여 적합한 컴퓨팅 리소스(142)를 적용한다. 따라서, 서버(140)는, 특정의 애플리케이션 로직에 초점이 맞추어지는 대신에, 임의의 웹태스크 요청을 실행하기 위한 포괄적인 실행 환경을 제공한다. 웹태스크를 위한 포괄적이고 균일한 실행 환경을 제공하는 서버(140)와 같은 서버는 본 명세서에서 때때로 MTIS라 한다.

[0024] 일 실시예에서, 포괄성을 유지하기 위하여, MTIS(140)는 균일한 실행 환경을 모든 웹태스크에 제공한다. 따라서, MTIS(140)에서 실행되는 다양한 애플리케이션의 백엔드 로직은 운영 체제(OS)(144) 및 사전 설치된 소프트웨어 패키지에 의해 제공된 동일한 기능에 액세스할 수 있다.

[0025] 웹태스크 모델에 의해 부과된 애플리케이션 전용 상태가 없는 것과 함께 MTIS(140)의 균일성은 전통적인 백엔드에 비하여 여러 이점을 가진다. 예를 들어, 웹태스크 런타임(runtime)은 다양한 이질적인 애플리케이션에 의해 쉽게 조정될 수 있다(104). 따라서, 이는 애플리케이션 로직 레이어로 하여금 하드웨어 레벨에서 대형 데이터 센터가 활용하는 것과 동일한 규모의 경제의 일부를 강화하게 할 수 있다. 따라서, MTIS(140)는 알려진 PaaS보다 더 높은 레벨의 추상적 개념으로 애플리케이션 로직 처리의 상품화를 가능하게 한다.

[0026] *신뢰할 수 없는 컴퓨터 루틴을 샌드박싱(Sandboxing)하기 위한 예시적인 멀티테넌트 모델*

[0027] 일 실시예에서, MTIS(140) 아키텍처는 멀티테넌트이다. 멀티테넌시는 컴퓨터 루틴(예를 들어, 소프트웨어)의 단일 인스턴스가 다수의 테넌트(tenant)에 서비스하는 동안 서버 상에서 실행하는 아키텍처를 말한다. 테넌트는 공용 액세스(common access)를 소프트웨어에 대한 특정 권한과 공유하는 사용자 그룹을 포함한다. 이러한 멀티테넌트 아키텍처를 이용하여, 소프트웨어 애플리케이션은 데이터, 구성, 사용자 관리, 테넌트 개별 기능 및 비기능적 특성의 공유를 모든 테넌트에 제공한다. 전술한 바와 같이, 본 명세서에서 논의된 MTIS(140)의 멀티테넌시 아키텍처는 애플리케이션 밀도를 증가시킨다.

[0028] MTIS(140)의 멀티테넌트 아키텍처에서의 한 가지 고려 사항은 한 테넌트의 악의적인(또는 단순히 부정확하게 기록된) 컴퓨터 루틴이 다른 테넌트의 데이터를 액세스하는 것을 어떻게 방지하는가 이다. 이점에서, 본 명세서에서 논의된 웹태스크 요청은 MTIS(140) 내의 샌드박스(sandbox)를 호출할 수 있다. 샌드박스는 실행 중인 프로그램을 분리하기 위한 보안(security) 메커니즘이다. 이는, 메모리 상의 전용 공간과 같은, 게스트 프로그램이 실행되는 엄격하게 통제되는 리소스 세트를 제공한다. 그 목적으로, Docker®가 웹태스크 요청 주위로 안전한 (컨테이너(CONTAINER)) 샌드박스를 형성하는데 사용될 수 있다. 예를 들어, Docker는, 가상 기계가 베어 메탈(bare metal)로부터 운영 체제를 분리하는 방법과 유사하게, 컨테이너 기술을 이용하여 인프라 스트럭처로부터 애플리케이션을 분리한다. 웹태스크 요청은 컴퓨터 루틴에 대한 링크를 제공하고. 런타임, 시스템 도구, 시스템 라이브러리 등과 같은 실행될 컴포넌트를 본질적으로 포함하는 완전한 파일 시스템에서 컴퓨터 루틴을 래핑하는 Docker 컨테이너로서 구현될 수 있어, 이에 의해 이것이 이의 목적지 환경(즉, MTIS(140))에서 실행될 것임을 보장한다.

[0029] 일 실시예에서, 웹태스크 요청(124)을 이용하여 실행되는 커스텀(custom) 컴퓨터 루틴은 HTTP 요청과 연계된다. 실행 시간은 HTTP 요청의 전형적인 수명으로 제한될 수 있다. 달리 말하면, 웹태스크 요청(124)은 HTTP 요청/응답 사이클 또는 동등한 사이클에 의해 만족되기에 충분히 짧은 지속 시간을 가진다.

[0030] 예를 들어, 웹태스크 요청(124)은 웹태스크 요청(124) 본문(body)에서 서버 코드(또는 그에 대한 링크)를 포함하는 클라이언트(102)로부터의 HTTP POST 요청을 수락한다. 일 실시예에서, 웹태스크 요청(124)은 또한 컴퓨터 루틴이 MTIS(140)에서 실행할 분리 경계(isolation boundary)를 나타내는 웹태스크 컨테이너 이름을 특정한다. 웹태스크 컨테이너에 대한 고객(customer)의 1:1 맵이 있을 수 있으며, 이는 하나의 구독자에 관련된 컴퓨터 루틴이 다른 구독자의 컴퓨터 루틴으로부터 항상 분리된다는 것을 의미한다. MTIS(140)는 본 명세서에서 때때로 웹태스크 컨테이너라 불리는 분리된 환경 내에 커스텀 컴퓨터 루틴을 실행하고, 결과를 갖는 응답을 다시 전송한다. 일 실시예에서, 응답은 JavaScript Object Notation(JSON) 형식을 가질 수 있다. 따라서, 웹태스크 요청(124)을 통해 제공된 커스텀 컴퓨터 루틴은 모든 테넌트에 걸쳐 균일한 환경에서 실행된다.

[0031] 웹태스크 요청(124)은 이의 실행 동안 필요한 상황 데이터(contextual data)뿐만 아니라 컴퓨터 루틴(또는 그에 대한 링크)을 포함한다. 예를 들어, 클라이언트(102)는 JavaScript 함수 클로저(function closure)를 제출한다. MTIS(140)는 그 함수를 호출하고 단일 콜백(callback) 파라미터를 제공한다. 웹태스크 요청에서의 커스텀 컴퓨터 루틴이 웹태스크 컨테이너 내에서 실행을 완료할 때, 이는 콜백을 호출하고 에러 표시 또는 단일

결과 값을 제공한다. 일 실시예에서, 그 결과 값은 그 다음 JSON으로서 직렬화되어 HTTP 요청에서 클라이언트(102)로 리턴된다.

[0032] 일 실시예에서, MTIS(140)는 Node.js에 기초하고, 이는 커스텀 컴퓨터 루틴이 웹태스크 환경에서 사전에 권한 설정된 고정된 세트의 Node.js 모듈을 활용할 수 있게 한다. 지원된 모듈 세트는 다양한 확장 가능성 시나리오의 특정 요건에 의해 제공될 수 있다. MTIS(140)에서 모든 테넌트에 걸친 컴퓨팅 환경의 균일성은 요청(124)이 도착할 때 예열된 웹태스크 컨테이너의 풀(pool)이 테넌트에 할당될 준비가 된 상태를 유지할 수 있게 한다. 이것은 콜드 스타트업(cold startup) 대기 시간을 감소시킨다.

[0033] MTIS(140)는 웹태스크 요청(124)을 처리하기 위하여 리소스를 할당하는데 있어서의 간접비의 양을 감소시키도록 구성된다. 리소스 할당 간접비는 가상 기계를 증식하고, 프로세스를 증식하고, 스레드를 증식하고, 메모리를 할당하는 형태로 나타날 수 있다. 따라서, MTIS(140)는 리소스 풀링(pooling)을 사용할 수 있다.

[0034] 일 실시예에서, 컴퓨터 루틴 환경은 Docker(오픈 소스)에 의해 제공된 것과 같은 서드 파티 인프라 스트럭처 컴파트먼트를 이용하여 분리될 수 있다. Docker는 단지 환경을 추상화하지만 멀티테넌시를 제공하지 않는다. 가상 기계는 또한 벤더의 클라우드 인프라 스트럭처에 의해 그리고/또는 MTIS(140)에 의한 요청에 의해 풀링될 수 있다.

[0035] MTIS(140)는 프로세스 풀을 증식하고, 할당 해제(de-allocating) 프로세스 대신에, 프로세스를 풀에 리턴할 수 있다. 그러나, 클라우드 간접비를 감소시키기 위하여, 실제로, 할당된 프로세스의 개수는 요청이 단일 스레딩되는 것으로 가정되기 때문에 단일 디지트로 있을 수 있다. 프로세스 관리는 또한 Java 가상 기계 또는 Node.js® 런타임과 같은 실행 환경에 의해 관리될 수 있다.

[0036] MTIS(140)는 또한 분리된 방식으로 컴퓨터 루틴의 실행을 보장하기 위하여 v8::isolate 및 v8::context와 같은 분리 및 콘텍스트 프리미티브(primitive)를 이용할 수 있다. 일 실시예에서, MTIS(140)는 자신의 메모리를 관리할 수 있다. 이 대신에, Java 가상 기계 또는 Node.js®과 같은 실행 환경이 자신의 메모리를 관리할 수 있다. 실행 환경이 자신의 메모리 할당기 및 자신의 가비지 수집기(garbage collection)를 가질 수 있다는 점을 주목하라.

[0037] 일 실시예에서, 보안은 분리 프리미티브를 이용하여 구현될 수 있다. 구체적으로는, 실행 환경은 해당하는 샌드박스에서 컴퓨터 루틴을 실행할 수 있다. 추가적인 보안 및 인증이 MTIS(140)에 의해 수행될 수 있다. 더욱 전형적으로, 초기 인증이 클라우드 인프라 스트럭처로의 공용 계정에 대한 것일 수 있다. 따라서, 인증은 요청당(per request) 기반일 필요가 없으며, 이에 의해 성능을 개선한다.

[0038] 일 실시예에서, 언어 바인딩은 실행 환경(즉, MTIS(140))에 의해 관리된다. 바인딩은 실행 환경에 고유하거나, 이 대신에 전형적으로 동적으로 링크된 라이브러리 형태로 애드온(add-on)을 통할 수 있다. 다양한 실행 환경으로 사전 구성될 수 있는 샌드박스가 이러한 실행 환경을 프로그래밍적으로 열거할 수 있기 때문에, 실행 환경(상이한 언어를 가짐)은 또한 동적으로 발견될 수 있다. 따라서, MTIS(140)는 또한 무엇이 지원되는지를 판단할 수 있고, 샌드박스를 증식/호출하여야 하는 대신에 에러 메시지로 신속하게 응답할 수 있다.

[0039] 프리컴파일(pre-compilation)은 MTIS(140)를 통해 구현되는 최적화일 수 있다. 예를 들어, 웹태스크 요청(124)에 임베디드된 컴퓨터 루틴은 소스 코드 대신에 바이트 코드일 수 있다. 저장된 프로시저가 호출되는 경우에, 서버 측 데이터베이스는 저장된 프로시저(저장된 프로시저가 MTIS(140)에 상주할 수 있다는 점에 주목하라)를 프리컴파일하였을 수 있다. 이러한 방식으로, 웹태스크 요청(124)이 전송된 컴퓨터 루틴의 파라미터에만 종속하여 이루어질 수 있다.

[0040] 다양한 실시예에서, 본 명세서에서 설명된 멀티테넌트 시스템은 안전한 컴퓨터 루틴 실행을 위한 다양한 보장을 제공한다. 첫째, 한 테넌트의 컴퓨터 루틴이 다른 테넌트의 컴퓨터 루틴 또는 데이터를 액세스하는 것을 방지하는 데이터 분리가 있다. 예를 들어, 한 테넌트가 임베디드된 패스워드를 갖는 URL 또는 연결 스트링을 이용하여 커스텀 데이터베이스를 액세스하는 컴퓨터 루틴 또는 데이터를 실행시키면, 동일한 시스템에서 실행되는 다른 테넌트의 컴퓨터 루틴이 그 패스워드를 발견하는 것이 방지된다.

[0041] 둘째, 인증된 서비스 거부(Denial of Service(DOS)) 공격을 완화시키 위한 제어된 리소스 소비가 제공된다. 그 목적으로, 일 실시예에서, 웹태스크 요청(124)의 샌드박스는 임의의 한 테넌트가 사용할 수 있는 메모리, CPU 및 다른 시스템 리소스를 제한한다.

[0042] 이제 도 1의 MTIS(140)로서 사용될 수 있는 예시적인 웹태스크 가상 기계(virtual machine(VM))를 도시하는 도

3이 참조된다. 한 테넌트의 컴퓨터 루틴과 데이터를 다른 것으로부터 분리하기 위하여, 모든 테넌트의 컴퓨터 루틴은 샌드박스(310) 내의 자신의 웹태스크 컨테이너(예를 들어, Docker)에서 실행된다. HTTP 요청이 MTIS(140)와 같은 웹태스크 VM에 도착하면, 이는 프록시(306)에 의해 먼저 처리된다. 프록시(306)는 테넌트와 컨테이너 사이의 연관을 나타내는 상태를 유지한다. 일 실시예에서, 프록시(306)는 웹태스크 컨테이너가 이미 특정 테넌트에 할당되었는지 판단하기 위하여 etcd 구성을 검토한다. 그러한 경우에, 프록시(306)는 요청을 그 웹태스크 컨테이너(310)에 포워딩한다. 그렇지 않은 경우에, 프록시는 웹태스크 클러스터에서 사용 가능한 예열된 웹태스크 컨테이너의 풀로부터 그 테넌트를 위한 새로운 웹태스크 컨테이너를 할당한다. 그 다음, 프록시는 후속 요청을 위하여 etcd에서 그 연관을 기록한다(즉, 다양한 구성 데이터가 클러스터 멤버들 사이에 공유될 수 있게 하는, 클러스터에서 모든 컴퓨터에 걸쳐 실행되고 동적 구성 레지스트리를 제공하는 데몬(daemon)).

[0043] 웹태스크 컨테이너의 예열된 풀은 모든 테넌트에 대한 균일한 실행 환경에 의해 가능하게 된다. 풀에서 예열된 컨테이너를 선택할 수 있는 것은, Docker 컨테이너의 이미 낮은 스타트업 대기 시간을 고려하더라도, 컨테이너를 온 더 플라이(on the fly)로 권한 설정하는 것에 비하여 콜드 스타트업 대기 시간을 감소시킨다.

[0044] 일 실시예에서, 임의의 단일 웹태스크 컨테이너는 단지 다수의 동시에 발생하는 요청을 단일 테넌트를 대신하여 처리될 수 있게 하는 간단한 HTTP 서버이다. 특정 웹태스크 컨테이너 내에서 실행되는 요청은 서로 분리되지 않는다. 웹태스크 컨테이너의 수명은 신뢰할 수 있는 Docker 컨테이너에서 실행되고 따라서 사전 구성된 수명 관리 정책에 따라 클러스터 내에 임의의 웹태스크 컨테이너를 종료시킬 수 있는 컨트롤러 데몬에 의해 관리된다.

[0045] 일 실시예에서, 모든 테넌트의 컴퓨터 루틴을 자신의 Docker 컨테이너(310)에서 실행시키는 것에 더하여, 출구(egress) 방화벽 규칙이 웹태스크 클러스터에서 구성된다. 이러한 규칙은 하나의 웹태스크 컨테이너 내의 신뢰할 수 없는 컴퓨터 루틴이 웹태스크 컨테이너 또는 웹태스크 인프라 스트럭처와 통신하는 것을 방지한다. 웹태스크 컨테이너의 HTTP 서버가 브리지(308)(예를 들어, Docker에 의해 생성됨)에 의해 호스트의 네트워크로부터 분리된 로컬 네트워크에서 실행되고 있기 때문에, 방화벽 규칙을 설정하는 것이 가능하다. 일 실시예에서, 웹태스크 컨테이너에서 실행되는 컴퓨터 루틴은 공용 인터넷에 대한 발신 호(outbound call)를 개시할 수 있다. 이것은 커스텀 컴퓨터 루틴으로부터 고객의 데이터베이스 또는 회사 엣지(edge) 서비스와 같은 외부 데이터 소스 및 서비스로의 발신 통신을 가능하게 한다.

[0046] 메모리 및 CPU 소비를 제한하기 위하여, 제어 그룹(cgroup) 메커니즘(예를 들어, Docker®에 의해 제공됨)이 사용될 수 있다. cgroup은 리눅스에 의해 지원되는 메커니즘이고. Docker®는 cgroup의 상부에 구축되는 기술이라는 점이 주목되어야 한다. 또한, 모든 웹태스크 컨테이너는 일시적인 리눅스 사용자를 생성하여 스타트업에 있어서 그 사용자에 대한 장착형 인증 모듈(Pluggable Authentication Module(PAM)) 제한을 구성한다. 이러한 2개의 메커니즘은 함께 포크 밤(fork bomb)과 같은 메모리 및 CPU에 대한 다양한 공격을 방지하는데 도움을 준다.

[0047] *예시적인 호출(call) 흐름 프로세스*

[0048] 서버 요청 시스템에서 캡슐화된 코드를 통한 멀티테넌시에 대한 전술한 개요로, 예시적인 호출 흐름 프로세스의 상위 레벨 논의를 고려하는 것이 이제 도움이 될 수 있다. 그 목적으로, 도 4는 멀티테넌트 인프라 스트럭처 환경에서 코드(예를 들어, 컴퓨터 루틴)를 실행시키기 위한 상위 레벨 호출 흐름 프로세스를 도시하고, 실행된 컴퓨터 루틴은 지속적이지 않다. 호출 흐름 프로세스(400)는 하드웨어, 소프트웨어 또는 이들의 조합으로 구현될 수 있는 동작 시퀀스를 나타내는 논리적인 호출 흐름에서의 단계들의 집합으로 도시된다. 소프트웨어와 연계하여, 단계들은, 하나 이상의 프로세서에 의해 실행될 때, 언급된 동작들을 수행하는 컴퓨터 실행 가능한 명령어를 나타낸다. 일반적으로, 컴퓨터 실행 가능한 명령어는 특정 기능을 수행하거나 특정 추상 데이터 타입을 구현하는 루틴, 프로그램, 오브젝트, 컴포넌트, 데이터 구조 및 이와 유사한 것을 포함할 수 있다. 동작이 설명되는 순서는 한정하는 것으로 고려되도록 의도되지 않으며, 임의의 개수의 설명된 블록은 프로세스를 구현하기 위하여 임의의 순서로 그리고/또는 동시에 결합될 수 있다. 논의의 목적을 위하여, 프로세스(400)는 도 1의 시스템(100)을 참조하여 설명된다. 예시적인 호출 흐름(400)에서, 클라우드에 클라이언트(102), 웹태스크 서버(107) 및 멀티테넌트 인프라 스트럭처 서버(MTIS)(140)가 있다.

[0049] 단계 408에서, 개발자는 흐름(400)에서 클라이언트(102)로 표현되는 컴퓨팅 장치에서 실행될 하나의 코드(예를 들어, 컴퓨터 루틴)를 준비한다. 일 실시예에서, 컴퓨터 루틴은 온라인 파일 스토리지 서버(132)에 저장된다.

[0050] 단계 410에서, 웹태스크 서버(107)와의 연결이 구축되고, 요청이 잘 정의된 종점(endpoint)에 전송되고, 여기에서 실행될 임의의 애플리케이션을 위한 컴퓨터 루틴은 요청의 파라미터이다. 웹태스크 요청(124)은 웹태스크 토

큰(108)과 클라이언트 데이터(106)를 포함한다. 다양한 실시예에서, 웹태스크 토큰(108)은 컴퓨터 루틴(또는 컴퓨터 루틴(130)을 저장하는 온라인 파일 스토리지에 대한 URL 링크)과, 컴퓨터 루틴과 연관된 클라이언트 비밀(112)을 포함할 수 있다. 일 실시예에서, 웹태스크 서버(107)가 클라이언트(102)에 의해 도달되지 못할 수 있다면, 실패된 연결 에러가 클라이언트(102)로 리턴된다. 그 목적으로, 컴퓨터 루틴은 이 에러를 해결하기 위한 핸들러를 가질 수 있다.

[0051] 단계 412에서, 웹태스크 서버(107)는 웹태스크 토큰을 클라이언트 데이터(106)와 함께 수신하고, 사용된 컴퓨터 루틴의 종류를 판단한다. 컴퓨터 루틴에 기초하여, 웹태스크 서버(107)는 웹태스크 토큰(108)과 클라이언트 데이터를 포함하는 웹태스크 요청(124)을 생성한다. 다양한 실시예에서, 웹태스크 토큰은 MTIS(140)에서 분리되어 실행되는 컴포넌트를 포함하는 완전한 환경에서 컴퓨터 루틴을 래핑하는 멀티테넌트 컨테이너(예를 들어, Docker와 같은)를 호출할 수 있다.

[0052] 일 실시예에서, 웹태스크 서버(107)는 HTTP 서버이고, 클라이언트(102)와 웹태스크 서버(107) 사이의 통신(410) 동안의 연결은 HTTP 연결이고, 웹태스크 토큰과 데이터(106)를 포함하는 웹태스크 요청(124)은 HTTP 요청이다. 이 대신에, 단지 단일의 일반화된 종점이 개발자에게 노출된다면, 다른 프로토콜, 또는 원격 프로시저 호출(Remote Procedure Call(RPC))이 사용될 수 있다.

[0053] 단계 414에서, 웹태스크 서버(107)는 웹태스크 요청(124)을 MTIS(140)에 전송한다. 이 점에서, 웹태스크 서버(107)는 MTIS(140)로부터 컴퓨터 루틴에 의해 사용될 모든 인프라 스트럭처를 갖는 컨테이너를 요청한다. 일 실시예에서, MTIS(140)는 다양하게 지원되는 언어를 위한 언어 바인딩을 포함할 수 있다. 예를 들어, MTIS(140)는 JavaScript 바인딩 및 C# 바인딩을 가질 수 있다. 지원되지 않는 언어가 요청에서 도착하는 경우에, MTIS(140)는 적합한 에러 메시지를 제공할 수 있다.

[0054] 단계 416에서, MTIS(140)는 실행될 컴퓨터 루틴을 추출하고, MTIS(140)의 분리된 환경(즉, 웹태스크 컨테이너)에서 컴퓨터 루틴을 실행시킨다. 웹태스크 컨테이너는 MTIS(140)의 샌드박스 환경에서 실행될 수 있다. 또한, MTIS(140)는 웹태스크 서버(107)에 의해 사용된 프로토콜과 양립 가능한 포맷으로 응답을 구성한다. 다양한 실시예에서, 응답은 XML, JSON 및 이와 유사한 것의 형태를 가질 수 있다. 따라서, MTIS는 수신된 웹태스크 요청을 위한 포괄적이고 균일한 환경을 제공한다.

[0055] 일 실시예에서, MTIS는, 청구서 발부(즉, 단계 418) 목적으로, 요청 식별자(ID)와 연관된 MTIS(140)의 웹태스크 컨테이너에서의 컴퓨터 루틴의 실행 동안 소비된 리소스(예를 들어, CPU, 메모리 등)를 추적한다. MTIS(140)는 웹태스크 서버(107)가 생성한 요청 ID를 사용하고, 이를 JavaScript 런타임, 운영 체제 또는 MTIS(140)에 의해 내부적으로 생성된 것 중 하나로부터의 스레드 ID와 연관시킨다. 일 실시예에서, 사용된 스레드 리소스를 요청 ID와 연관시킴으로써, 요청 마다 소비된 리소스 기반으로의 계량이 실현된다. 리소스 추적은 CPU, 메모리(예를 들어, 랜덤 액세스 메모리(RAM), 하드 디스크)에 한정될 필요는 없고, 컴퓨터 루틴의 실행 동안 활용된 네트워크 리소스와 같은 임의의 계량 가능한 리소스를 포함할 수 있다.

[0056] 일 실시예에서, 단계 420에서, MTIS는 웹태스크 서버(107)로 응답을 전송한다. 응답은 MTIS(140)에서 실행된 컴퓨터 루틴에 기초한 계산 결과일 수 있다. MTIS가 요청을 만족할 수 없거나 요청을 시간에 맞추어 만족시키지 않을 수 있다면, MTIS는 적합한 응답(즉, 에러 메시지)으로 웹태스크 서버에 응답을 리턴할 수 있다. 따라서, MTIS로부터의 응답은 실행된 컴퓨터 루틴 또는 적합한 에러 메시지에 기초한 계산 결과일 수 있다.

[0057] 단계 422에서, 응답은 웹태스크 서버(107)로부터 클라이언트(102)로 포워딩된다. 이 대신에 또는 이에 더하여, 응답은 MTIS로부터 클라이언트(102)로 직접 포워딩될 수 있다.

[0058] 선택적으로, 단계 424에서, 결과가 웹태스크 서버에 의해 수신되었다는 확인이 MTIS(140)에 의해 웹태스크 서버로부터 수신될 수 있다. 단계 426에서, MTIS(140)는, 적합한 바에 따라, 리소스 할당 해제를 수행한다. 일 실시예에서, 사용된 컨테이너는 풀로 복귀되지 않는다; 대신에, 폐기되고 교체된다. MTIS(140)는 웹태스크 컨테이너를 폐기하고, 이에 의해 컴퓨터 루틴이 영구 저장되지 않는 것을 보장한다.

[0059] *예시적인 사용 케이스*

[0060] 서버 요청에 컴퓨터 루틴을 캡슐화하는 시스템 및 방법에 대한 전술한 설명으로, 일부 예시적인 사용 케이스의 상위 레벨 논의를 제공하는 것이 도움이 될 수 있다. 본 명세서에서 논의된 개념 및 시스템은 다양한 사용 케이스에 적용될 수 있다. 예를 들어, 이는 분산 애플리케이션에 적용될 수 있고, 여기에서 애플리케이션은 각각 서로 독립적으로 동작하도록 설계된 개별 컴포넌트로 구성될 수 있다. 이러한 개별 컴포넌트는 실행될 컴퓨터 루

틴을 MTIS(140)를 통해 상이한 인스턴스로 전송할 수 있다.

[0061] 일례에서, 본 명세서에 설명된 시스템은 오프로딩(offloading)을 위하여 사용될 수 있다. 그 목적으로, 애플리케이션은 MTIS(140)를 이용하여 로컬 또는 원격으로 루틴을 실행시킬 수 있다. 로컬 컴퓨팅 리소스가 사용 가능하지 않거나 불충분한 상황에서, 애플리케이션은 MTIS(140)를 통해 클라우드로 컴퓨팅 요청을 오프로딩할 수 있다.

[0062] 일례에서, 본 명세서에서 논의된 개념은 WEB 서비스를 스크립팅(scripting)하기 위하여 사용될 수 있다. 애플리케이션은 최종 사용자가 컴퓨터 루틴에 구체화된 상이한 기능에 자체로 도움이 되는 스크립트를 작성하는 기능을 제공할 수 있다. 스크립트의 일부는 MTIS(140)를 통해 클라우드에서 실행될 수 있다.

[0063] 일례에서, 본 명세서에서 설명된 시스템은 비동기 실행 애플리케이션에서 사용될 수 있다. 이러한 시나리오에서, MTIS(140)에서 실행되는 컴퓨터 루틴은 동기하여 실행될 필요는 없다. 이러한 점에서, 웹태스크 서버(107)는 수명이 길고/장시간 실행되는 프로세스를 위한 디스패처(dispatcher) 역할을 할 수 있다.

[0064] 일례에서, 본 명세서에서 논의된 개념은 수직 애플리케이션/보안을 위하여 사용될 수 있다. 보안 API는 샌드박스를 유지하는 멀티테넌트 인프라 스트럭처 서버 애플리케이션에서 실행되도록 구현될 수 있다. 일 실시예에서, MTIS(140)는 클라이언트와 멀티테넌트 인프라 스트럭처 서버 애플리케이션 사이의 연결을 안전하게 보호하기 위한 암호화를 지원할 수 있다. 컴퓨터 루틴이 MTIS(140)에 상주하지 않기 때문에, 그리고 모든 컴퓨터 루틴이 안전한 샌드박스에서 실행되기 때문에, MTIS(140)는 보안 API를 통해 노출될 때 인증 기능을 위한 안전한 실행 환경을 제공한다. 보안에 관하여, 일 구현예에서, MTIS(140)는 멀티티어 애플리케이션에서의 프록시 레벨에서 인증, 인가, 감사(auditing) 및/또는 계량 기능을 구현하는데 사용될 수 있다.

[0065] *예시적인 컴퓨터 플랫폼*

[0066] 위에서 논의된 바와 같이, 웹태스크 서버에 대한 연결을 구축하고, 컴퓨터 루틴을 실행하기 위한 요청을 전송하고, 메시지를 전송 및 수신하고, 웹태스크 토큰을 전송 및 수신하고, 웹태스크 컨테이너를 생성하고, 분리된 환경에서 컴퓨터 루틴을 실행하기 위한 기능 및 다른 기능은, 도 5에 도시된 바와 같이, 네트워크(130)를 통해 데이터 통신을 위하여 연결되고, 클라이언트 서버(102), 웹태스크 서버(107) 및 MTIS(140)로서 동작하는 컴퓨터들 상에서 구현될 수 있다. 특수 목적 장치가 사용될 수 있지만, 이러한 장치는, 또한, 데이터 통신을 위한 적합한 네트워크 연결을 이용하지만, 본 명세서에서 논의된 기능들과 같은 기능들을 구현하도록 "서버" 프로그래밍을 실행하는데 일반적으로 사용되는 일반적인 종류의 데이터 처리 장치를 나타내도록 의도된 하나 이상의 하드웨어를 이용하여 구현될 수 있다.

[0067] 도 5 및 6은 역시 클라우드 컴퓨팅에 대하여 적용될 수 있는 범용 컴퓨터 하드웨어 플랫폼의 기능 블록도를 제공한다. 도 5는 웹태스크 서버(107) 또는 MTIS(140)와 같은 서버를 구현하는데 전형적으로 사용될 수 있는 네트워크 또는 호스트 컴퓨터 플랫폼을 도시한다. 도 6은 클라이언트(102)와 같은 개인용 컴퓨터 또는 클라이언트 컴퓨팅 장치를 구현하는데 사용될 수 있는 사용자 인터페이스 요소를 갖는 장치를 도시한다. 도 5 및 6에 도시된 바와 같은 장비의 일반적인 구조와 일반적인 동작은 상위 레벨의 예시로부터 자명하여야 한다.

[0068] 서버로서 구성된 범용 컴퓨터는, 예를 들어, 네트워크(130)를 통한 패킷 데이터 통신을 위한 데이터 통신 인터페이스를 포함한다. 또한, 서버 컴퓨터는 프로그램 명령어를 실행시키기 위하여, 하나 이상의 프로세서 형태로, 중앙 처리 유닛(central processing unit(CPU))을 포함한다. 서버 플랫폼은 전형적으로 서버에 의해 처리되고 그리고/또는 통신될 다양한 데이터 파일을 위한 내부 통신 버스, 프로그램 스토리지 및 데이터 스토리지를 포함한다. 이러한 서버의 하드웨어 요소, 운영 체제 및 프로그래밍 언어는 본질적으로 통상적이다. 물론, 서버 기능은 프로세싱 부하를 분산시키기 위하여 다수의 유사한 플랫폼에서 분산 방식으로 구현될 수 있다.

[0069] 위에서 논의된 바와 같이, MTIS에 대한 요청은 클라이언트 기계로부터 수행될 수 있다. 클라이언트 기계는, 도 6과 유사하게, 프로세서, 메모리 및 직접 또는 인터넷을 통해 클라우드 서버에 연결되기에 충분한 네트워크 연결을 갖는 임의의 장치일 수 있다. 전형적으로, 운영 체제가 있을 것이다. 전형적인 구성은 중앙 처리 유닛, RAM 및 WiFi나 이더넷 연결이다. 메모리는 컴퓨터 판독 가능한 매체일 것이고 그리고/또는 다른 컴퓨터 판독 가능한 매체에 액세스할 수 있을 것이며, 메모리 및/또는 다른 컴퓨터 판독 가능한 매체에 상주하는 컴퓨터 실행 가능한 코드로 이루어진 클라이언트 애플리케이션을 실행시킬 것이다.

[0070] 유사하게, 도 5에 도시된 것과 같은 클라우드 서버는 프로세서, 메모리 및 직접 또는 인터넷을 통해 클라이언트 기계에 연결되기에 충분한 네트워크 연결을 갖는 장치일 수 있다. 클라이언트 기계에서와 같이, 전형적으로 운영 체제가 있을 것이다. 전형적인 구성은 중앙 처리 유닛, RAM 및 WiFi나 이더넷 연결이다. 메모리는 컴퓨터 판

독 가능한 매체일 것이고 그리고/또는 다른 컴퓨터 판독 가능한 매체에 액세스할 수 있을 것이며, 메모리 및/또는 다른 컴퓨터 판독 가능한 매체에 상주하는 컴퓨터 실행 가능한 코드로 이루어진 클라이언트 애플리케이션을 실행시킬 것이다.

[0071] 클라우드 서버는 일반적으로 가상 기계를 생성할 수 있는 가상화 환경을 실행시킬 것이다. 각각의 가상 기계에서, 운영 체제 또는 시스템 레벨 환경이 있을 수 있다. 각각의 가상 기계는 각각이 스레드를 증식할 수 있는 프로세스들을 증식할 수 있다. Java Virtual Machine 또는 .NET 런타임과 같은 실행 환경이 가상 기계에서 실행되어 프로세스와 스레드를 관리할 수 있다.

[0072] 도 5 및 6에 도시된 RAM과 ROM과 같은 컴퓨터 판독 가능한 매체는, 적어도, 2가지 종류의 컴퓨터 판독 가능한 매체, 즉 컴퓨터 저장 매체 및 통신 매체를 포함한다. 컴퓨터 저장 매체는 컴퓨터 판독 가능한 명령어, 데이터 구조, 프로그램 모듈 또는 다른 데이터와 같은 정보의 저장을 위한 임의의 방법 또는 기술로 구현된 휘발성 및 비휘발성의 제거 가능 및 제거 가능하지 않은 매체를 포함한다. 컴퓨터 저장 매체는 RAM, ROM, EEPROM, 플래시 메모리 또는 다른 메모리 기술, CD-ROM, DVD(digital versatile disk) 또는 다른 광학 스토리지, 자기 카세트, 자기 테이프, 자기 디스크 스토리지 또는 다른 자기 저장 장치, 또는 컴퓨팅 장치에 의한 액세스를 위하여 정보를 저장하는데 사용될 수 있는 임의의 다른 비전송 매체를 포함하지만 이에 한정되지 않는다. 대조적으로, 통신 매체는 컴퓨터 판독 가능한 명령어, 데이터 구조, 프로그램 모듈 또는 다른 데이터를 반송파 또는 다른 전송 메커니즘과 같은 변조된 데이터 신호로 구체화할 수 있다. 본 명세서에서 정의되는 바와 같이, 컴퓨터 저장 매체는 통신 매체를 포함하지 않는다.

[0073] 소프트웨어 기능은 실행 가능한 코드 및 연관된 저장 데이터, 예를 들어, 컴퓨터 루틴을 실행하기 위한 요청을 전송하는 것, 메시지를 전송 및 수신하는 것, 웹태스크 토큰을 전송 및 수신하는 것, 웹태스크 컨테이너를 생성하는 것, 분리된 환경에서 컴퓨터 루틴을 실행하는 것 및 다른 기능을 위하여 웹태스크 서버(107) 및 MTIS(140)에서 애플리케이션을 위하여 사용되는 파일을 포함하는 프로그래밍을 수반한다. 소프트웨어 코드는 컴퓨팅 장치에 의해 실행 가능하다. 동작시, 코드는 컴퓨팅 장치 내에 저장된다. 그러나, 다른 때에는 소프트웨어는 다른 위치에 저장되고 그리고/또는 적합한 컴퓨팅 장치 시스템으로의 로딩을 위하여 전송될 수 있다. 컴퓨팅 장치의 프로세서에 의한 이러한 코드의 실행은 컴퓨팅 장치가, 본질적으로 본 명세서에서 논의되고 예시된 구현예에서 수행되는 방식으로, 다양한 기능을 수행할 수 있게 한다.

[0074] 따라서, 위에서 개요가 서술된 바와 같은 노드 데이터를 수신하고 처리하는 방법의 양태들은 프로그래밍으로 구체화될 수 있다. 기술의 프로그램 양태는 전형적으로 실행 가능한 코드 및/또는 일종의 비일시적인 기계 판독 가능한 매체 상에 반송되거나 그 내에서 구체화되는 연관된 데이터의 형태로 "제품" 또는 "제조 물품"으로서 고려될 수 있다.

[0075] *결론*

[0076] 전술한 것이 최선의 형태 및/또는 다른 예가 되는 것으로 고려되는 것을 설명하였지만, 그에 다양한 변경이 이루어질 수 있고, 본 명세서에 개시된 내용이 다양한 형태 및 예로 구현될 수 있으며, 일부만이 교시 내용이 본 명세에서 설명된 많은 적용예에 적용될 수 있다는 것이 이해되어야 한다. 본 교시 내용의 진정한 범위 내에 있는 임의의 그리고 모든 적용, 변경 및 수정을 청구하는 것이 다음의 청구범위에 의해 의도된다.

[0077] 바로 위에 언급된 바를 제외하고는, 설명되거나 예시된 어떠한 것도, 청구범위에 인용되었는지 여부에 관계없이, 공중에 대한 임의의 컴포넌트, 단계, 특징, 목적, 이익, 이점 또는 균등물의 헌납을 발생시키도록 의도되지 않으며, 또한 그와 같이 이해되어서도 안 된다.

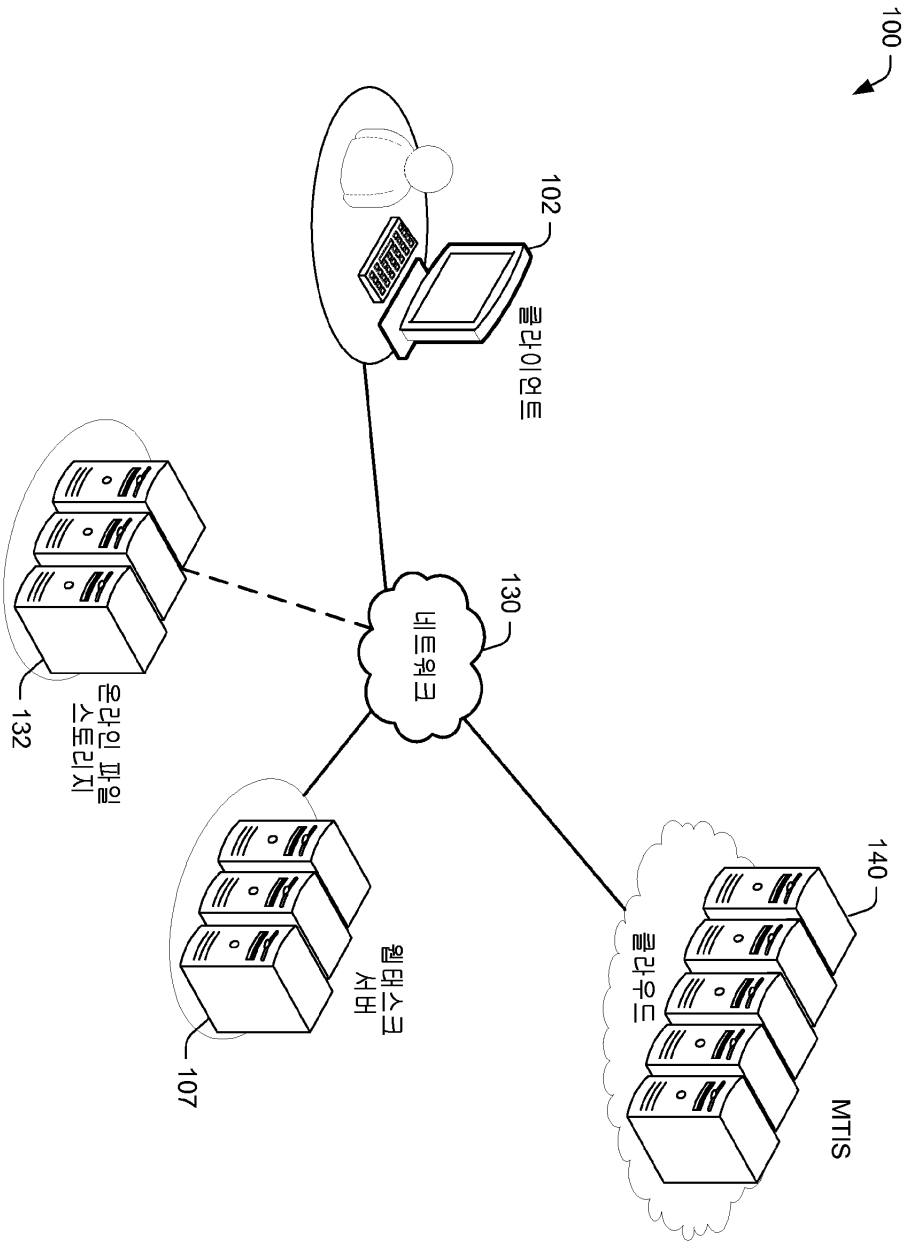
[0078] 여기에서 사용된 용어 및 표현은, 특정 의미가 본 명세서에서 달리 설명된 것을 제외하고는, 탐구 및 연구의 대응하는 영역에 관하여 이러한 용어 및 표현에 부합되는 바와 같은 통상적인 의미를 갖는다는 것이 이해될 것이다. 제1 및 제2 등과 같은 관계형 용어는 하나의 엔티티 또는 동작을 다른 것으로부터, 이러한 엔티티 또는 동작 사이의 임의의 실제적인 관계 또는 순서를 반드시 필요로 하거나 암시하지 않으면서, 구별하기 위하여만 사용된다. "포함한다", "포함하는"이라는 용어 및 이의 임의의 다른 파생어는, 요소의 리스트를 포함하는 프로세스, 방법, 물품 또는 장치가 단지 이러한 요소들만을 포함하지 않고 명시적으로 열거되지 않거나 이러한 프로세스, 방법, 물품 또는 장치에 고유한 다른 요소를 포함할 수 있도록, 비배타적인 포함을 아우르도록 의도된다. 단수로 표시된 요소는, 추가적인 제한 사항 없이, 요소를 포함하는 프로세스, 방법, 물품 또는 장치에서 추가적인 동일한 요소의 존재를 배제하지 않는다.

[0079] 요약서는 독자가 기술적 개시 내용의 본질을 신속하게 확인하는 것을 돕기 위하여 제공된다. 이는, 이것이 청구

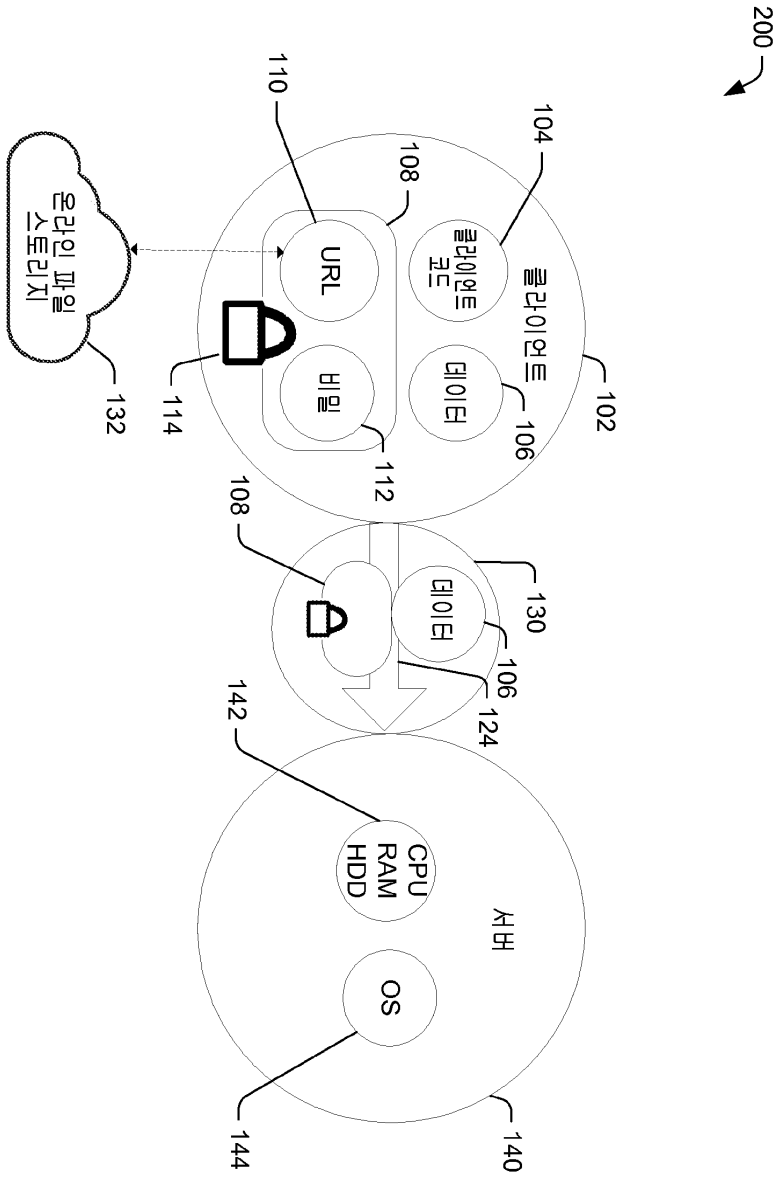
범위의 범위 및 의미를 해석하거나 제한하는데 사용되지 않을 것이라고 이해하면서 제출된다. 또한, 전술한 발명의 상세한 설명에서, 다양한 특징은 본 개시 내용을 간소화하기 위하여 다양한 실시예에서 함께 그루핑된다는 것을 알 수 있다. 이러한 개시 방법은 청구된 실시예가 각 청구항에서 명시적으로 인용된 것보다 더 많은 특징을 필요로 한다는 의도를 반영하는 것으로 이해되어서는 안 된다. 오히려, 이어지는 청구범위가 반영하는 바와 같이, 본 발명의 내용은 단일의 개시된 실시예의 모든 특징 이내에 있다. 따라서, 이어지는 청구범위는 발명의 상세한 설명에 편입되며, 각각의 청구항은 개별적으로 청구된 내용으로서 자신을 대표한다.

도면

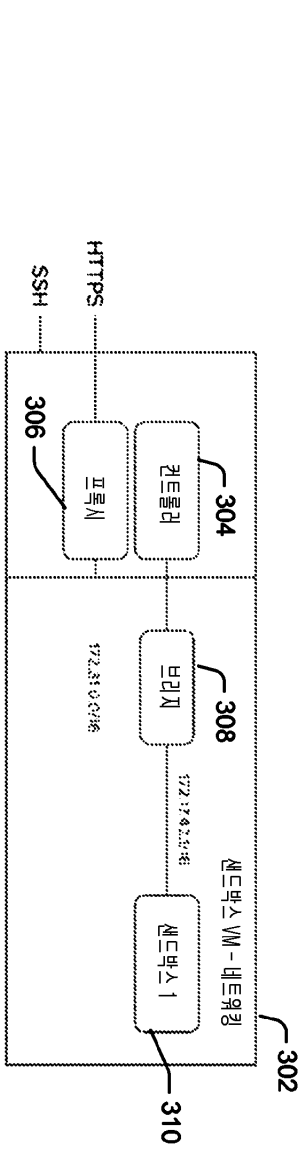
도면1



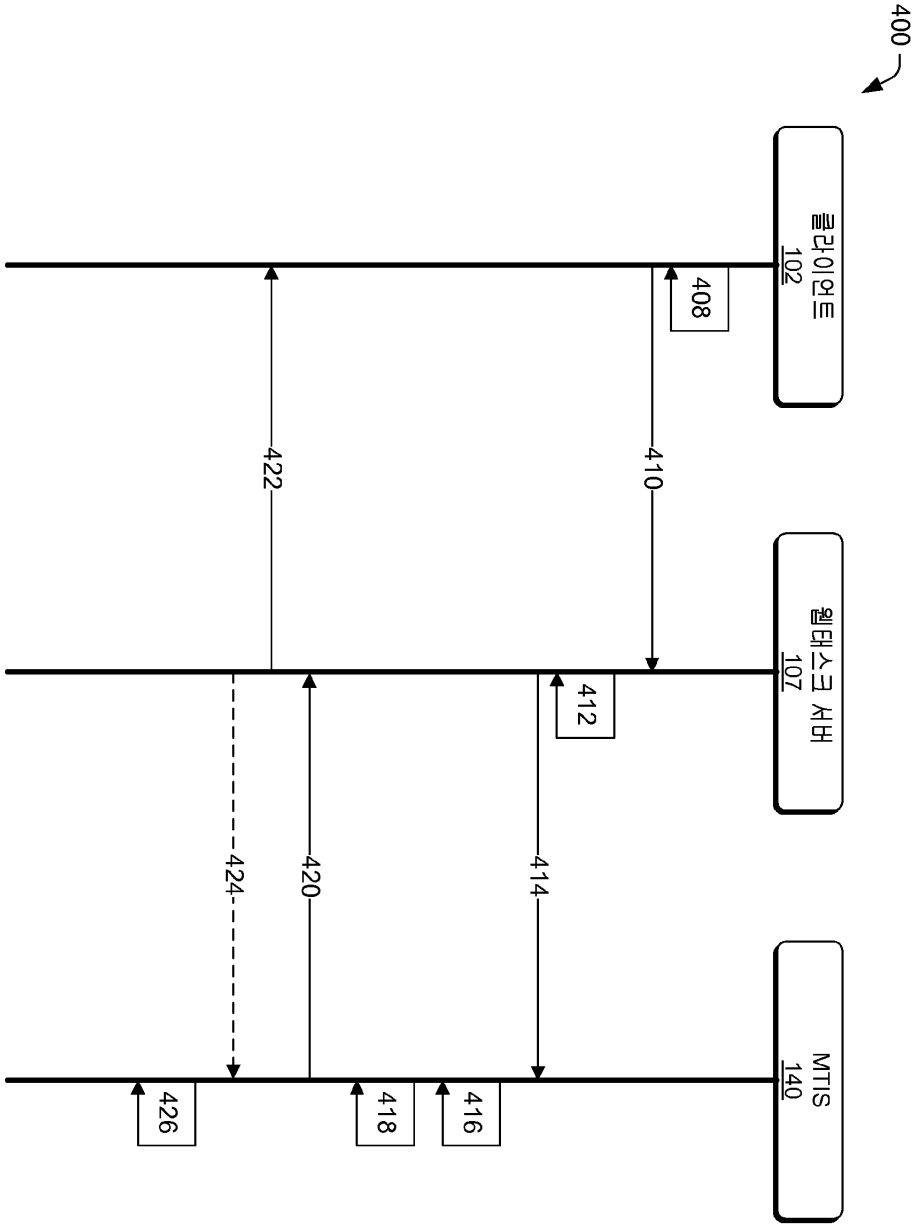
도면2


200
110
104
108
온라인 파일
스토리지
URL
클라이언트
코드
클라이언트
102
비밀
데이터
106
114
132
112
108
130
데이터
106
124
142
CPU
RAM
HDD
서버
OS
144
140

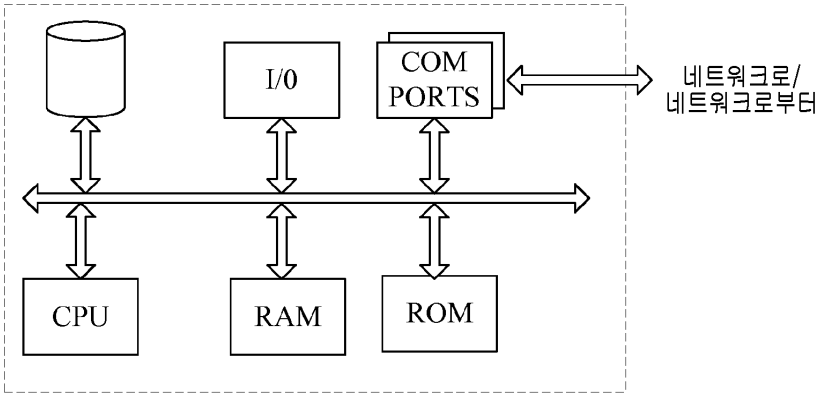
도면3


300
HTTPS
SSH
306
프록시
304
컨트롤러
308
브리지
샌드박스 VM - 네트워킹
샌드박스 1
302
310

도면4


400
클라이언트
102
웹태스크 서버
107
MTIS
140
408
422
410
412
424
420
414
426
418
416

도면5


I/0
COM
PORTS
네트워크로/
네트워크로부터
CPU
RAM
ROM

도면6