(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년08월21일
(11) 등록번호 10-2013005
(24) 등록일자 2019년08월14일

(51) 국제특허분류(Int. Cl.)
G06F 9/50 (2018.01) *G06F 15/16* (2018.01)
(21) 출원번호 10-2014-7021738
(22) 출원일자(국제) 2013년02월01일
심사청구일자 2018년01월03일
(85) 번역문제출일자 2014년08월01일
(65) 공개번호 10-2014-0122240
(43) 공개일자 2014년10월17일
(86) 국제출원번호 PCT/US2013/024242
(87) 국제공개번호 WO 2013/116581
국제공개일자 2013년08월08일
(30) 우선권주장
13/366,039 2012년02월03일 미국(US)
(56) 선행기술조사문헌
WO2011159842 A2*
(뒷면에 계속)

(73) 특허권자
마이크로소프트 테크놀로지 라이센싱, 엘엘씨
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
왕 주
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 페이턴츠 마이크로소프트 코포레이션
콜더 브래들리 진
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 페이턴츠 마이크로소프트 코포레이션
스크졸스볼드 아릴드 이
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 페이턴츠 마이크로소프트 코포레이션
(74) 대리인
제일특허법인(유)

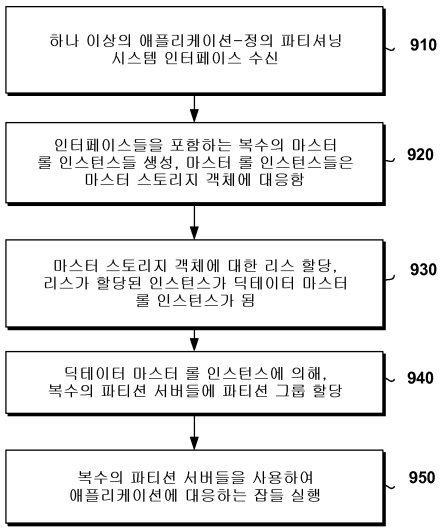
전체 청구항 수 : 총 3 항 심사관 : 유진태

(54) 발명의 명칭 **확장 가능한 환경에서의 파티션 관리 기법**

(57) 요 약

확장 가능한 환경에서 애플리케이션-정의 잡들을 파티션하기 위한 일반적인 프레임워크를 가능하게 하는 시스템 및 방법이 제공된다. 일반적인 프레임워크는 잡의 파티셔닝을 잡의 다른 양태들과 구분 짓는다. 그 결과, 사용자는 파티셔닝 알고리즘을 제공할 필요가 없게 되어, 애플리케이션-정의 잡을 정의하는 데 필요한 노력이 줄거나 최소화된다. 또한 일반적인 프레임워크는 분산 환경에서 계산을 수행하는 마스터와 서버의 관리를 용이하게 한다.

대 표 도 - 도9



(56) 선행기술조사문헌
KR1020110082529 A*
JP2004508616 A
US20040203378 A1
US20070233803 A1
*는 심사관에 의하여 인용된 문헌

명 세 서

청구범위

청구항 1

분산 컴퓨팅 환경에서 계산을 실행하는 방법으로서,

하나 이상의 애플리케이션-정의 파티셔닝 시스템 인터페이스를 수신하는 단계와,

상기 하나 이상의 애플리케이션-정의 파티셔닝 시스템 인터페이스를 포함하는 복수의 마스터 롤 인스턴스(master role instances)를 생성하는 단계 - 상기 마스터 롤 인스턴스들은 마스터 스토리지 객체에 대응함 -와,

상기 마스터 스토리지 객체에 대한 리스(lease)를 할당하는 단계 - 각각의 마스터 롤 인스턴스는 상기 리스를 얻기 위해 경쟁하고, 상기 리스를 할당받은 상기 마스터 롤 인스턴스가 딕테이터(dictator) 마스터 롤 인스턴스임 -와,

상기 딕테이터 마스터 롤 인스턴스에 의해, 복수의 파티션 서버에 파티션 그룹을 할당하는 단계- 상기 복수의 파티션 서버 중 한 파티션 서버에 초기 파티션 할당이 수행되면, 상기 한 파티션 서버는 상기 한 파티션 서버에 사유화된(private to) 스토리지 객체에 대한 리스를 유지하고, 사유화된 상기 스토리지 객체는 상기 한 파티션 서버에 의해 서빙되는 파티션을 식별함 -와,

상기 복수의 파티션 서버를 사용하여 애플리케이션에 대응하는 하나 이상의 계산을 실행하는 단계와,

상기 딕테이터 마스터 롤 인스턴스에 의해, 상기 복수의 파티션 서버로 하트비트 메시지를 전송하는 단계와,

상기 딕테이터 마스터 롤 인스턴스에 의해 상기 복수의 파티션 서버 중 하나의 파티션 서버로부터, 상기 하트비트 메시지에 대한 응답을 수신하는 단계- 상기 응답은 상기 하나의 파티션 서버가 현재 서빙하고 있는 하나 이상의 파티션을 나타냄 -와,

상기 딕테이터 마스터 롤 인스턴스에 의해, 상기 하나 이상의 나타낸 파티션과 파티션 테이블에 유지된 파티션 할당 정보 간에 불일치가 있으면, 대응하는 상기 사유화된 스토리지 객체에 대한 상기 하나의 파티션 서버의 리스를 중단하는 단계- 상기 파티션 테이블은 상기 파티션 그룹 중 하나의 파티션이 상기 복수의 파티션 서버 중 하나에 할당될 때 업데이트됨 -를 포함하는

방법.

청구항 2

제 1 항에 있어서,

하트비트 메시지에 대한 응답으로 상기 파티션 서버들로부터 하나 이상의 메트릭이 수신되고 상기 하나 이상의 메트릭 중 적어도 하나는 애플리케이션-정의 메트릭인

방법.

청구항 3

삭제

청구항 4

삭제

청구항 5

삭제

청구항 6

삭제

청구항 7

삭제

청구항 8

분산 컴퓨팅 환경에서 계산 태스크를 실행하는 시스템에 있어서,

실행 시에, 상기 시스템으로 하여금 청구항 제 1 항 또는 제 2 항의 방법을 수행하게 하는 컴퓨터 사용-가능 명령어를 실행하는 복수의 프로세서를 포함하는

시스템.

청구항 9

삭제

청구항 10

삭제

발명의 설명

기 술 분 야

배 경 기 술

[0001] 클라우드 컴퓨팅 환경은 사용자들이 해당 인프라에 투자할 필요 없이 많은 양의 컴퓨팅 자원에 액세스 가능하게 하는 잠재적인 플랫폼을 제공한다. 대신에, 컴퓨팅 환경은 클라우드 컴퓨팅 제공자에 의해 서비스로서 제공될 수도 있다. 이를 통해 사용자는 컴퓨팅 잡(job)의 크기와 중요도에 부합하도록 요청 컴퓨팅 자원들을 조정할 수 있다. 그러나, 서비스로서 이용 가능한 추가 컴퓨팅 자원을 완전히 이용하려면, 태스크를 실행하는 데 다수의 물리적 또는 가상 프로세서들이 사용될 수 있도록 컴퓨팅 태스크들을 더 작은 부분으로 나누어야만 한다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0002] 다양한 실시예에서, 확장 가능한 환경에서 애플리케이션-정의 잡들(application-defined jobs)을 파티션하기 위한 일반적인 프레임워크를 가능하게 하는 시스템 및 방법이 제공된다. 일반적인 프레임워크는 잡의 파티셔닝을 잡의 다른 양태들과 구분 짓는다. 그 결과, 사용자는 파티셔닝 알고리즘을 제공할 필요가 없게 되어, 애플리케이션-정의 잡을 정의하는 데 필요한 노력이 줄거나 최소화된다. 또한 일반적인 프레임워크는 분산 환경에서 계산을 수행하는 마스터(master)와 서버의 관리를 용이하게 한다.

[0003] 본 요약은 아래의 상세한 설명에서 추가로 기술되는 개념들의 선택을 단순화된 형식으로 소개하기 위해 제공되었다. 본 요약은 특허청구된 대상의 핵심 특성 또는 중요 특성을 식별하기 위한 것이 아니며, 특허청구된 대상의 범위를 결정하기 위해 따로 사용되기 위한 것도 아니다.

도면의 간단한 설명

[0004] 본 발명의 예시적인 실시예는 첨부된 도면들을 참조하여 아래에서 상세하게 기술되었다.

도 1에서 3은 본 발명의 양태에 따른, 분산 컴퓨팅 환경의 일례의 다양한 양태를 개략적으로 도시한다.

도 4는 본 발명의 양태에 따른, 일반 파티셔닝 환경에서 잡을 수행하는 다양한 인터페이스들 간의 인터랙션을 개략적으로 도시한다.

도 5는 본 발명의 양태에 따른, 일반 파티셔닝 환경의 일례를 도시한다.

도 6은 본 발명의 양태에 따른, 액티브 마스터 롤에 대한 백업 머신들을 제공하는 일례를 도시한다.

도 7은 본 발명의 실시예의 구현에 사용하기 적합한 예시적인 컴퓨팅 환경의 블록도이다.

도 8은 본 발명의 양태에 따른, 마스터 롤에 대한 백업 머신을 제공하는 다른 예시를 도시한다.

도 9에서 10은 본 발명의 다양한 실시예에 따른 방법들의 예시를 도시한다.

도 11은 본 발명의 양태에 따른, 결함 도메인(fault domain)과 업그레이드 도메인(upgrade domain)에 관련된 마스터 롤에 대해 백업 머신들을 제공하는 일 양태를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0005] 개관

[0006] 네트워크 상에서의 데이터 전송 속도의 증가와 기타 네트워크 특징의 개선으로 인해, 컴퓨팅 자원이 대규모 네트워크에 분산되어 있는 환경에서 대규모의 컴퓨팅 태스크를 실행하는 것이 점차 가능해졌다. 제 1 위치에 있는 사용자가 컴퓨팅 서비스에 잡 또는 컴퓨팅 태스크를 제출하여 사용자가 직접적으로 알지 못하는 컴퓨터 그룹에서 태스크를 실행시킬 수 있다. 사용자의 태스크를 실행하는 컴퓨팅 자원들은 다수의 장소에 분산되어 있을 수 있다. 하나 이상의 장소에 있는 제 1 컴퓨팅 자원 그룹이 사용자의 컴퓨팅 태스크를 실행하기 위한 데이터 및 기타 정보를 저장하는 한편, 동일한 장소 또는 한 군데 이상의 다른 장소에 있을 수 있는 제 2 컴퓨팅 자원 그룹은 컴퓨팅 태스크를 실행하는 데 사용될 수 있다.

[0007] 다양한 분산 컴퓨팅 자원들에 대한 액세스를 통해 사용자는 컴퓨팅 자원들의 위치에 대한 걱정 없이 잡 태스크를 실행할 수 있다. 또한 분산 자원들은, 특정 시간까지 컴퓨팅 태스크를 완료시키는 등 컴퓨팅 태스크에 대한 목표를 충족시키기 위해, 사용자가 사용되는 자원량을 확대시킬 (또는 축소시킬) 기회를 제공하게 된다. 그러나, 분산 컴퓨팅 자원들을 사용하는 것은 사용자에게 많은 문제를 야기한다. 종래에, 사용자 또는 애플리케이션 디자이너는 특정 분산 컴퓨팅 환경을 이용할 방법을 결정하기 위해 애플리케이션이나 잡의 설계에서 자원을 별도로 고려할 필요가 있었다.

[0008] 다양한 실시예에서, 클라우드 컴퓨팅 환경과 같은 분산 환경에서 확장 가능한 애플리케이션을 구축하는 일반 프레임워크를 가능하게 하는 시스템 및 방법이 제공된다. 일반 프레임워크는 잡 또는 워크 아이템(work item)의 파티셔닝을 그 잡 또는 워크 아이템과 관련된 계산의 실행과 분리시킬 수 있다. 이를 통해 분산 환경의 소유주는 분산 자원들에 대한 높은 가용성(availability)을 제공하면서도 액세스 가능하거나 단순화된 방식으로 사용자에게 자원들을 제공할 수 있게 된다. 일반 프레임워크는, 확장성(scalability), 결함 허용성(fault tolerance), 및/또는 가용성과 같은 특징을 다루기 위해 필요한 노력을 줄이거나 최소화하는 한편, 이들 특징을 처리하는 프레임워크를 제공함으로써 분산 환경에 대한 애플리케이션 설계의 복잡도(complexity)를 줄이게 된다. 결과적으로, 프레임워크는 애플리케이션 디자이너로 하여금 분산 환경의 요건에 추가 시간을 소비하는 대신 애플리케이션에 중점을 둘 수 있게 한다.

[0009] 정의

[0010] "키(key)"란 일반 파티션 프레임워크에서 사용되는 기본 개념들 중 하나이다. 키는 네임스페이스(namespace) 또는 도메인의 값이다. 네임스페이스의 일례로 클라우드 분산 환경의 모든 스토리지 계정에 대응하는 식별자를 들 수 있다. 본 예시에서, 키는 계정 이름이나, 계정 번호나, 또는 특정 계정을 참조 표시할 수 있는 다른 식별자에 대응할 수 있다. 네임스페이스의 일례로 잡에 대한 입력 파라미터의 가능한 영숫자 값의 범위를 들 수 있다. 바람직하게는, 네임스페이스를 포괄한 키들은, 키들이 순차 값들의 범위로 기술될 수 있도록 하는 체계화 방법을 갖고 있을 것이다. 예를 들어, 키들은 해시값(hashed values)에 기반하여 숫자로, 알파벳 순으로 체계화되거

나, 또는 범위 시작 및 범위 종료로써 두 개의 키를 특정하는 것에 기반하여 키들의 범위를 정의하는 기타 임의의 편리한 순차 방식으로 체계화될 수 있다.

[0011] "파티션(partition)"이란 로우(low, 포괄적인) 키 및 하이(high, 배타적인) 키에 의해 정의되는 범위이다. 파티션은 공백인 범위일 수 없다. 필요하다면, 하나의 파티션이 도메인의 전체 범위를 포함할 수도 있다. 두 개의 상이한 파티션의 범위 사이에 중첩이 없도록 파티션들은 상호 배타적이라고 정의된다. 모든 파티션들의 합집합은 전체 도메인 또는 네임스페이스를 포괄할 것이다(span).

[0012] "파티션 서버"란 0 이상의 파티션을 서빙하는 롤 인스턴스(role instance)에 대응하는 클라우드 컴퓨팅 환경 내의 가상 머신이다. 파티션은 동시에 둘 이상의 파티션 서버에 의해 서빙되지 못한다. 반면, 특정 파티션 서버는 그 파티션 서버에 의해 현재 서빙되고 있는 어떤 파티션들도 갖고 있지 않을 수 있다. 파티션 서버는 다양한 액션을 수행하기 위해 애플리케이션-정의 인터페이스 및 (파티셔닝 시스템에 의해 정의된) 고정 인터페이스 모두 포함할 수 있다.

[0013] "파티션 마스터"란 파티션 서버에 파티션들을 할당하거나 할당하지 않는 등에 의해 특정 유형의 롤에 대한 파티션 서버들을 관리하는 롤이다. 또한 파티션 마스터 롤은 예를 들어, 파티션 서버들이 할당된 파티션들을 여전히 서빙하고 있음을 확인함으로써 파티션 서버들을 모니터링한다. 일반적으로, 이런 롤은 결함 허용성을 위한 추가적인 부분이다. 파티션 마스터는 다양한 액션을 수행하기 위해 애플리케이션-정의 인터페이스 및 (파티셔닝 시스템에 의해 정의된) 고정 인터페이스 모두 포함할 수 있다.

[0014] "애플리케이션-정의 인터페이스"란 롤 인스턴스에 의한 수행을 위해 클라이언트에 의해 정의된 계산, 동작, 또는 기타 기능을 지칭한다. 파티션 마스터 또는 파티션 서버 롤 중 어느 하나를 위해 애플리케이션-정의 인터페이스가 생성될 수 있다. 애플리케이션-정의 인터페이스는 파티셔닝 시스템의 "고정 인터페이스"와는 대조적이다. 고정 인터페이스는 파티셔닝 시스템의 일부로서 제공되는 애플리케이션 인터페이스를 가리킨다. 클라이언트는 고정 인터페이스의 액션을 수정할 수 없다. 그러나, 애플리케이션-정의 인터페이스는 고정 인터페이스와 함께 사용될 수 있다. 예를 들어, 마스터 롤에 대한 고정 인터페이스는 마스터 롤이 관리하고 있는 서버들의 상태를 확인하기 위해 매 주기마다 하트비트(heartbeat) 메시지를 전송할 수 있다. 서버 롤에 대한 대응 고정 인터페이스는 하트비트 메시지에 적절한 방식으로 대응하기 위한 인터페이스일 수 있다. 애플리케이션-정의 인터페이스의 일례로 추가 정보를 하트비트 메시지에 추가하는 인터페이스를 들 수 있고, 다른 예로 하트비트 메시지로부터 이런 추가 정보를 추출하기 위한 인터페이스를 들 수 있다. 이런 상황에서, 고정 인터페이스가 하트비트 메시지 자체를 전송한다. 사용자 또는 애플리케이션은 메시지 전송 프로토콜을 수정할 수 없다. 그러나, 사용자는 메시지 컨텐츠를 정의하기 위한 인터페이스를 수정할 수 있다. 하트비트 메시지의 정보를 보충하기 위한 인터페이스의 기본 구조는 시스템에서 제공될 수 있다. 그러나, 클라이언트는 이런 인터페이스에서 제공되는 컨텐츠를 수정할 수 있으므로, 본원에서 이런 인터페이스는 애플리케이션-정의 인터페이스로 정의된다.

[0015] 다른 예를 들면, 애플리케이션-정의 인터페이스는 로드 밸런싱(load balancing)의 전체적인 특징을 제공하기 위해 고정 인터페이스와 함께 사용될 수 있다. 파티션 서버에 파티션을 할당하거나 파티션을 두 개의 파티션으로 나누기 위한 인터페이스는 고정 인터페이스들이다. 그러나, 애플리케이션-정의 인터페이스는 머신들 간의 파티션 할당 변경 또는 파티션을 나눌 시간 결정과 같은 로드 밸런싱 작업이 행해져야 하는 시간에 대한 수식(expression)을 제공할 수 있다.

[0016] 각각의 액티브 파티션 마스터 롤 또는 파티션 서버 롤은 그 롤에 대응하는 데이터를 제어하는 대응 스토리지 객체를 갖고 있다. 스토리지 객체의 일례로 블랍(binary large object or blob)을 들 수 있다. 파티션 서버의 경우, 스토리지 객체는 서빙되고 있는 파티션들의 신원을 포함한다. 파티션 서버에 대한 스토리지 객체는 일반적으로 파티션에 대응하는 기저 데이터(underlying data)를 포함하지 않을 것이라는 점을 유의한다. 기저 데이터를 별개의 데이터 스토리지 장소에 둔 채 파티션 신원만을 저장함으로써, 최소의 데이터 이전으로 한 쪽 서버에서 다른 쪽 서버로 파티션들을 이동시킬 수 있다. 파티션 마스터에서, 스토리지 객체는 파티션 마스터의 특정 인스턴스를 액티브 인스턴스로 설정하는 정보를 포함한다. 스토리지 객체는 마스터 서버에 의해 관리되고 있는 서버들의 모든 스토리지 객체들에 관련된 정보도 선택적으로 포함할 수 있다. 가동 중에, 마스터 및 서버 롤들은 대응 스토리지 객체에 대한 리스(lease)를 유지할 수 있다. 롤의 스토리지 객체에 대한 리스가 중단될 때, 대응 롤도 종료될 수 있다.

[0017] 앞서 언급한 대로, 결함이 생기면 적어도 하나의 추가 파티션 마스터를 이용할 수 있도록, 특정 유형의 롤에 대한 파티션 마스터들이 중복되는 것이 바람직하다. "딕테이터(dictator)"란 특정 유형의 롤에 대한 파티션 마스터 기능을 현재 실행하는 파티션 마스터로 정의된다. 딕테이터는 특정 파티션 마스터 롤에 관련된 스토리지 객

체에 대한 리스(lease)를 유지하는 파티션 마스터다.

[0018] 액티브 파티션 마스터(딕테이터)는 하트비트를 통해 파티션 서버들과 통신할 수 있다. 기본 하트비트는 "킵얼라이브(keepalive)" 유형으로서 항상 사용된다. 앞서 언급한 대로, 선택적인 명령어 및/또는 정보를 이 하트비트 메시지에 추가하기 위해 애플리케이션-정의 인터페이스를 사용할 수 있다.

[0019] 네임스페이스, 키, 및 파티션

[0020] 다양한 실시예에서, 분산 컴퓨팅 환경에서의 처리를 위해 자동으로 파티션된 계산을 사용자 또는 애플리케이션이 정의할 수 있도록 일반 파티셔닝 프레임워크가 제공된다. 일반 파티셔닝 프레임워크를 이용하기 위해, 사용자는 네임스페이스에 기반하여 계산을 정의할 수 있다. 네임스페이스는 그 네임스페이스와 관련된 데이터에 대해 선택적으로 실행되는 하나 이상의 계산 또는 연산(calculation)에 대응한다. 아래에서 설명되는 바와 같이 사용자-정의 네임스페이스는 특정 특성을 갖고 있을 수 있다. 네임스페이스가 필수적인 특성을 갖고 있는 한, 일반 파티셔닝 프레임워크는 애플리케이션-정의 인터페이스에 기반하여 네임스페이스를 자동으로 파티션할 수 있을 것이다. 이와 같은 파티셔닝은 파티셔닝 시스템과 함께 제공되는 사용자-정의 네임스페이스와 고정 인터페이스만을 사용하여 수행된다. 일반 파티셔닝 시스템의 고정 인터페이스들만을 사용하여 네임스페이스를 파티션함으로써, 이 파티셔닝은 사용자가 제공하는 임의의 애플리케이션-정의 인터페이스와 구분될 뿐만 아니라 실행중인 계산(들)의 본질로부터도 구분된다.

[0021] 잡에 대한 네임스페이스 또는 도메인은 그 위에서 동작되거나 및/또는 계산에 의해 생성되는 모든 범위의 데이터 유형에 대한 일련의 식별자에 대응할 수 있다. 추가적으로 또는 그 대신에, 네임스페이스 또는 도메인은 그 위에서 동작되거나 및/또는 계산에 의해 생성될 수 있는 가능한 상태 목록 또는 범위에 대응할 수 있다. 또 다른 옵션으로, 네임스페이스는 분산 환경에서 실행되는 계산의 다수의 인스턴스에 대한 일련의 식별자에 대응할 수 있다. 계산을 위해 가능한 모든 네임스페이스 또는 도메인이 식별될 필요는 없다. 대신에, 사용자는 파티셔닝을 위해 사용될 네임스페이스 또는 상태를 식별할 수 있다. 식별된 네임스페이스 또는 도메인은 사용자가 특정 롤을 사용하여 실행하기 원하는 계산의 전부를 포괄해야 한다. 그 위에서 동작되는 데이터 셋 전체를 계산에 의해 포괄하거나, 연산 인스턴스들의 전체를 계산에 포괄하거나, 또는 기타 임의의 편리한 방식으로, 계산의 전부를 포괄할 수 있다. 네임스페이스 내에서 이산 값이나 상태를 정하는 데 키를 사용할 수 있다. 또한 값의 범위를 정하는 데 키를 사용할 수도 있다. 키가 범위를 정하는 데 사용될 수 있으므로, 네임스페이스가 순차 배열(serial ordering) 유형을 포함해야 하고, 따라서 상위 키(upper key)와 하위 키(lower key)가 범위를 고유하게 식별할 것임을 알 것이다. 이러한 배열은 알파벳에 기반한 배열을 비롯한 종래의 배열에 기반할 수 있다. 그 대신에, 네임스페이스 내의 값 목록을 포함하는 파일에 기반한 순차 배열과 같이 배열이 임의적일 수 있다.

[0022] (네임스페이스를 사용하여 실행되는 임의의 계산을 포함하는) 네임스페이스의 정의에 더해서, 사용자는 서버 롤에 관련된 하나 이상의 애플리케이션-정의 인터페이스 또한 제공할 수 있다. 애플리케이션-정의 인터페이스는 파티션 서버가 실행할 수 있는 하나 이상의 태스크 또는 기능을 나타내고, 이런 태스크 또는 기능은 하나 이상의 네임스페이스를 선택적으로 포함한다. 서버 롤에 대한 애플리케이션-정의 인터페이스는 특정 네임스페이스에서 실행될 모든 태스크나 기능을 포함할 수 있어서, 다른 서버 롤들이 동일한 데이터 또는 상태에 액세스하려고 시도하지 않는다. 선택적으로, 예시적인 양태로 서버 롤이 서브-롤들(sub-roles)을 포함하여, 어느 하나의 롤의 일부 애플리케이션-정의 인터페이스들이 동일한 롤의 다른 애플리케이션-정의 인터페이스들과 다른 네임스페이스에서 동작하는 경우도 고려된다. 간단한 예를 들면, 클라이언트는 적어도 하나의 유형의 연산을 데이터 셋에 실행하는 것을 포함하는 계산을 실행하기 원할 수 있다. 이런 상황에서, 클라이언트는 데이터 셋으로부터 요청된 구성 요소(들)에 대한 적어도 하나의 유형의 연산을 실행하는 서버의 단일 롤(single role)을 정의할 수 있다. 이러한 단일 롤은 과학적 연산, 데이터 셋에 대한 하나 이상의 관련 데이터 마이닝 기능, 또는 기타 임의의 편리한 계산을 실행하도록 구성된 롤 인스턴스들에 대응하는 하나 이상의 서버를 나타낼 수 있다. 계산 및 관련 데이터는 계산을 실행하기 위한 네임스페이스의 정의의 일부로서 정의될 수 있다. 추가로, 계산에 관한 메트릭(metrics)을 마스터 롤 인스턴스에 제공하기 위한 애플리케이션-정의 인터페이스와 같이, 한 개 이상의 애플리케이션-정의 인터페이스를 서버 롤에 제공할 수 있다. 서버 롤 인스턴스는 서버에 전달된 키 값들에 기반하여 동작할 데이터 셋의 구성 요소 또는 구성 요소들을 결정할 수 있다. 적어도 하나의 추가 마스터 롤이 서버들을 관리할 수 있고, 이는 하나 이상의 서버에 데이터 셋 파티션들을 할당하는 것을 포함한다.

[0023] 네임스페이스에 기반하여, 다수의 파티션 서버들이 네임스페이스에 대한 상이한 처리 부분들을 다루거나 실행하도록 계산이 파티션될 수 있다. 각각의 파티션은 키 값의 범위에 대응한다. 파티션 서버에 파티션이 할당되면, 서버는 할당된 파티션에 대응하는 범위 내의 키 값을 포함하는 요청에 대해 예상된 계산을 실행한다. 서버에 할

당된 파티션들은 네임스페이스의 순차 배열에 대해 연속일 필요는 없다.

[0024] 일반 파티셔닝 환경에서, 파티션 서버들에 대한 파티션들의 현재 할당을 추적하기 위해 파티션 테이블을 사용할 수 있다. 액티브 마스터 혹은 딕테이터가 서버에 파티션을 할당할 때, 먼저 그 할당을 반영하도록 파티션 테이블이 업데이트될 수 있다. 그리고 나서, 클라이언트 요청에서 지정된 키에 기반하여 클라이언트 요청을 다룰 파티션 서버를 결정하기 위해 파티션 테이블이 사용될 수 있다. 선택적으로, 파티션된 각각의 네임스페이스에 대해 다른 파티션 테이블을 갖는 것과는 달리, 다수의 롤에 대한 파티션 할당을 추적하기 위해서 한 개의 파티션 테이블을 사용할 수 있다. 예를 들어, 파티션 테이블의 엔트리는 범위에 대한 로우 키, 범위에 대한 하이 키, 및 요청된 키에 대응하는 데이터 또는 상태에 요청 태스크를 실행할 롤 인스턴스 또는 서버 인스턴스를 포함할 수 있다. 파티션 테이블은 에포크 번호(epoch number) 또는 버전 번호와 같은 기타 데이터를 더 포함할 수 있으며, 이는 아래에서 더욱 상세하게 논의될 것이다.

[0025] 마스터 및 서버 롤 인스턴스 관리

[0026] 잡을 실행할 때, 중복성(redundancy)을 제공하기 위해 몇몇 마스터 롤 인스턴스들을 갖고 있는 것이 보통 바람직할 것이다. 그러나, 충돌을 피하기 위해, 특정 시간에 단지 하나의 마스터 롤 인스턴스만이 액티브 마스터일 수 있다. 복수의 마스터 롤 인스턴스가 존재할 때, 마스터 롤 인스턴스들은 전체 네임스페이스에 대응하는 스토리지 객체에 대한 리스를 얻기 위해 경쟁한다. 리스를 얻은 마스터 롤 인스턴스가 액티브 마스터 혹은 딕테이터가 된다. 또한, 그 마스터 롤 인스턴스의 스토리지 객체에 마스터 에포크 번호가 저장된다. 마스터가 딕테이터가 되면, 마스터는 이 번호를 증가시키고, 이를 마스터 롤 스토리지 객체에 기록하고, 이어서 에포크 번호를 사용하여 대응 파티션 서버들과 통신한다. 액티브 마스터 인스턴스 또는 딕테이터는 파티션 테이블 또는 파티션 테이블의 적어도 일부에 대한 리스를 얻을 수 있다. 파티션 서버들은 이미 목격한 최고 에포크 번호보다 낮은 마스터 에포크를 갖는 하트비트를 무시함으로써, 더 이상 딕테이터가 아닌 마스터 롤로부터의 오래된 하트비트를 피할 수 있다.

[0027] 한 가지 옵션은 스토리지 객체 리스를 통해 딕테이터쉽(dictatorship)을 구현하기 위해 별도의 딕테이터쉽 라이브러리를 사용하는 것이다. 별도의 딕테이터쉽 라이브러리의 존재는 파티셔닝 시스템 외부에 있는 롤로 하여금 중복성을 구현할 수 있게 하는 것을 비롯한 여러 이점을 제공할 수 있다. 이를 통해 파티셔닝에 관여하지 않는 롤들이 액티브 마스터 롤을 선택하는 데 동일한 방법을 사용할 수 있게 된다.

[0028] 각각의 서버는 자신의 스토리지 객체에 대한 리스를 유지한다. 마스터는 서버가 자신의 첫 번째 파티션 할당을 받을 때마다 서버 스토리지 객체 이름을 생성한다. 또한, 각각의 서버는 자신의 현재 파티션 할당(에포크 번호를 포함하는 파티션 목록)을 스토리지 객체에 저장한다. 제삼자는 서버와 그 스토리지 객체 간의 리스를 강제로 종료시킬 수 있다. 파티션 마스터 딕테이터는 파티션 할당 프로토콜의 서버 리스를 중단시키기 위해 이런 기능을 사용할 수 있다.

[0029] 마스터 롤의 중복 인스턴스들을 가짐으로써 장애 이벤트가 일어날 때 향상된 성능을 제공할 수 있지만, 오래된 업데이트로 인해 중복 인스턴스들이 문제를 일으킬 가능성도 있다. 오래된 업데이트란 새로운 딕테이터가 선택된 후에 서버 또는 데이터 스토어에서 이전 딕테이터로부터의 메시지나 명령을 수신하는 상황을 말한다. 오래된 업데이트의 문제는 외부 스토어에 상태를 유지하는 롤 또는 일부 코드에 영향을 미칠 수 있고, 외부 스토어로의 메시지가 (예컨대, 인터넷 상에서) 지연 또는 재배열되거나 롤의 장애 조치(failover) 시에 그 스토어로부터 상태가 판독될 수 있다.

[0030] 예를 들어 롤의 파티션 테이블을 업데이트하는 도중에 장애 조치하는 파티션 마스터(딕테이터)를 생각해보자. 먼저, 오래된 마스터는 파티션 테이블의 행 'X'의 업데이트를 시작한다. 그리고 나서 오래된 마스터는 작동을 멈춘다. 마스터의 중복 인스턴스가 새로운 딕테이터로서 선택된다. 이 새로운 액티브 마스터는 파티션 테이블의 행 'X'를 사용하여 업데이트, 판독, 또는 기타 동작을 실행한다. 새로운 액티브 마스터에 의한 행 'X'에 대한 액션 후에 오래된 마스터로부터의 업데이트가 행해진다. 오래된 마스터로부터의 업데이트가 파티션 테이블에 통합되면, 업데이트는 새로운(현재) 마스터가 변경을 인식하지 못한 채 파티션 테이블을 변경하게 될 것이다. 이로 인해 파티션 테이블에서 모순된 상태가 생길 수 있다.

[0031] 이런 문제점에 대한 한 가지 해결책은 오래된 마스터로부터의 오래된 업데이트를 어떻게든 막는 것이다. 한 가지 옵션으로 파티션 마스터들로 하여금 파티션 테이블의 부분들에 대한 리스를 얻게 하는 것이 있다. 마스터의 리스는 특정 네임스페이스에 대응하는 테이블의 파티션 전부에 대응할 수 있다. 리스 범위를 지정하기 위한 임의의 편리한 방법이 사용될 수 있다. 예를 들어, 필요한 경우, 리스는 파티션의 일부만을 포함하거나 및/또는

다수의 파티션들을 포함할 수 있다. 새로운 딕테이터가 선택될 때, 마스터들은 리스를 통해 딕테이터쉽을 위해 여전히 경쟁할 것이고, 서버들이 오래된 하트비트 메시지와 같이 오래된 업데이트를 막을 수 있도록 에포크 번호가 제공된다. 게다가, 마스터 딕테이터는 그 상태를 구축할 때 테이블 판독 전에 파티션 테이블(연관 부분)에 대한 리스를 취득할 수 있다.

[0032] 보다 일반적으로, 마스터 또는 서버가 데이터 구조를 처리하도록 배정될 때 마스터와 서버 양쪽 모두는 블랍, 테이블, 및/또는 기타 데이터 구조에 대한 리스를 취득함으로써 리스 메커니즘을 사용하여 오래된 업데이트를 막을 수 있다. 장애 조치 또는 할당을 변경하기 위한 마스터의 명시적인 요청 등으로 인해 마스터 또는 서버에 데이터 구조가 더 이상 할당되지 않을 때, 리스가 해제된다.

[0033] 오래된 업데이트는 마스터와 서버 간의 통신에서 문제가 될 수도 있다. 예를 들어, 마스터가 딕테이터가 될 때 마스터가 에포크 번호를 얻게 함으로써 오래된 마스터로부터의 메시지 처리를 피할 수 있다. 딕테이터쉽 장애 조치마다 에포크 번호가 증가된다. 이 에포크 번호는 모든 서버 하트비트 및/또는 마스터에서 서버로의 기타 메시지에서 전송될 수 있다. 서버는 자신이 목격한 최고 에포크 번호보다 낮은 에포크 번호를 가진 하트비트를 무시할 수 있다. 가장 높은 에포크 번호가 서버의 소프트 상태에 저장될 수 있다. 마스터와 서버가 스토리지 객체 리스를 통해 상태를 전달하고 있는 경우, 오래된 메시지 문제는 상기의 방법으로 충분히 피할 수 있다. 그 대신에, 마스터와 서버 간의 오래된 메시지를 피하는 기타 편리한 해결책도 사용할 수 있다.

[0034] 이전에 할당된 서버로부터의 오래된 업데이트를 피하기 위해 각각의 범위 파티션에 대해 유사한 에포크 번호 방법을 사용할 수 있다. 예를 들어, 각각의 파티션은 자신의 파티션에 변화가 생겼을 때 마스터에 의해 업데이트된 현재 에포크 번호를 갖고 있을 수 있다. 파티션에 대한 변화의 예시들로는 새로운 서버에 파티션 할당, 파티션 분할, 및 두 개의 파티션 병합을 들 수 있다. 새로운 서버에의 파티션 할당으로 인해 에포크 번호는 1만큼 증가할 수 있다. 파티션을 두 개 이상의 새로운 파티션들로 나누면 각각의 자식 파티션이 1만큼 증가된 부모 에포크 번호를 수신할 수 있다. 두 개의 파티션이 합쳐질 때, 합쳐진 파티션의 에포크 번호는 합치기 전의 파티션들 중 최대 에포크 번호가 1만큼 증가된 것일 수 있다. 그 대신에, 파티션의 에포크 번호 증가를 추적하는 기타 임의의 편리한 방법을 사용할 수도 있다.

[0035] 파티션 에포크 번호를 사용하는 방법의 일례로서, 파티션 마스터, 두 파티션 서버들 S1 및 S2, 및 제 3 서버 X가 있는 시스템을 생각해보자. 제 3 서버 X는, 예를 들어, 파티션 마스터 및 서버들 S1 및 S2에 의해 동작하는 네임스페이스의 실제 데이터 셋을 포함하는 데이터 서버일 수 있다. 서버들 S1 및 S2는 X에게 명령(또는 기타 메시지들)을 내린다. X의 프론트 엔드에 의한 처리 등으로 인해 S1 또는 S2로부터 X까지 오는 동안 메시지들이 지연될 수 있다고 가정하자. X가 특정 범위 파티션에 대해 목격된 최고 에포크를 파악하지 않는 경우, X가 오래된 메시지를 어떻게 받아들이는지를 보기 쉽다. 예를 들어, 에포크가 3인 파티션 P는 처음에 서버 S1에 의해 서빙될 수 있다. S1은 X로 메시지 M1을 전송할 수 있다. 메시지 M1은 파티션 P는 물론 에포크 번호 3에 대응하는 키를 포함한다. 그리고 나서 파티션 마스터는 파티션 P를 S1에서 S2로 이동시킨다. 할당 후에, S2는 새로운 에포크 번호를 포함하는 메시지 M2를 서버 X에 전송한다. X는 메시지 M1을 수신하기 전에 메시지 M2를 수신한다. 이로 인해 M1은 오래된 메시지가 된다. 그 후에 X는 오래된 메시지 M1을 수신한다. 에포크 번호를 추적함으로써, 서버 X는 오래된 메시지 M1이 파티션 P에 책임이 없는 서버로부터 온 것임을 인식할 수 있다.

[0036] 오래된 업데이트의 가능성을 배제하기 위해, 에포크 평가 라이브러리(epoch validation library)를 사용할 수 있다. 에포크 평가 라이브러리는 수신된 메시지가 오래된 것이 아님을 인증한다. 예를 들어, 서버가 새로운 딕테이터로부터 메시지를 수신하거나 새로운 파티션 범위에 관련된 요청을 수신할 때, 서버는 메시지가 마스터 또는 파티션의 현재 에포크 번호를 포함하는지를 확인하기 위해 에포크 평가 라이브러리를 살펴볼 수 있다.

[0037] 도 4는 롤, 파티션, 및 키들 간의 관계의 개략적인 예시를 도시한다. 도 4에서, 사용자(405)는 사용자 계정의 정보에 액세스하는 등 원하는 데이터 부분에 대한 액션 수행 요청을 입력한다. 데이터는 키에 의해 식별된다. 클라이언트 요청과 키는 클라이언트 또는 사용자(405)가 한 요청의 유형을 다루는 롤(420)로 전달된다. 롤(420)은 롤이 특정 유형의 요청을 처리하는 방법을 정의하는 클라이언트 라이브러리(430)를 포함한다. 요청의 유형 및 키에 기반하여, 롤(420)은 요청에서의 키에 대응하는 키 범위를 처리하는 현재 서버를 찾기 위해 파티션 테이블(455)을 검색해본다. 파티션 테이블(455)의 정보는 파티션 마스터(460)가 내린 파티션 결정에 기반하여 채워진다. 파티션 마스터(460)는 도 4에서 복수의 잠재적인 파티션 마스터 중 하나로서 도시된다. 추가 잠재적인 파티션 마스터들은 중복성을 위한 것으로 필요할 때까지 액티브하지 않다. 도 4의 예시에서, 복수의 파티션 서버들(465)은 롤(420)에서 요청한 태스크들을 실행하기 위한 롤 인스턴스들로서 이용될 수 있다. 파티션 테이블(455)에 기반하여, 복수의 파티션 서버들(465) 중 파티션 서버 N이 클라이언트 요청에서의 키에 대응하는 키 범

위를 다룬다.

[0038] 일반 파티셔닝 프레임워크의 예시

[0039] 도 5는 본 발명에 따른 일반 파티셔닝 프레임워크를 제공하는 모듈 및/또는 인터페이스 그룹의 일례를 도시한다. 도 5는 또한 일반 파티셔닝 환경을 이용할 수 있는 계산 잡을 제공하기 위한 애플리케이션-정의 인터페이스의 예시들도 디스플레이한다. 도 5에서, 사용자 또는 애플리케이션에 의해 키 또는 네임스페이스가 제공되었지만, 네임스페이스를 파티션하기 위한 인터페이스는 파티셔닝 시스템에서 제공되는 고정 인터페이스이다. 이는 네임스페이스에 대응하는 데이터에 대해 동작하는 잡 또는 워크 아이템 실행으로부터 네임스페이스의 파티셔닝을 구분한다.

[0040] 도 5에서, 사용자가 제공하거나 지정한 적어도 두 가지 유형의 컴포넌트 또는 인터페이스가 있다. 사용자는 키(510)(및 대응 네임스페이스) 및 애플리케이션에 대한 복수의 서버 인터페이스들(520)에 대한 정의를 제공한다. 전술한 바와 같이, 키는 애플리케이션의 네임스페이스를 설명한다. 이를 통해 파티셔닝 시스템은 가능한 파티셔닝에 대한 변수, 상태, 및/또는 계산의 범위를 알 수 있다. 키의 정의에 더해서, 사용자는 키에 관련된 적어도 세 개의 인터페이스들도 제공한다. 키(510)에 관련된 인터페이스들은 네임스페이스에서 키 직렬화 기능, 키 역직렬화(deserialize) 기능, 및 두 가지 키 비교 기능을 제공한다. 사용자가 네임스페이스 및 키 값을 선택하였으므로, 이 인터페이스들의 동작은 파티셔닝 시스템에 의한 고정 인터페이스로서 제공되지는 않는다. 대신에, 키 및 네임스페이스에 관련된 인터페이스는 애플리케이션-정의 인터페이스로서 사용자에 의해 제공된다.

[0041] 또한 사용자는 서버 롤 인스턴스들에서 사용하는 애플리케이션-정의 인터페이스들(520)을 제공한다. 서버 롤 인스턴스들의 애플리케이션-정의 인터페이스들(520)은 서버가 키에 대해 동작하는 요청을 수신할 때 서버가 수행할 동작들에 관한 기능을 포함한다. 예를 들어, StartServeKeyRange의 인터페이스는 특정 키 범위 서빙을 시작할 때 서버가 실행할 수 있는 하나 이상의 동작을 정의할 수 있다. StopServeKeyRange의 대응 인터페이스는 서버로 하여금 순서대로 키 범위의 서빙을 종료시키게 할 수 있다. 또한, 서버로부터 정보를 수집하는 것이 바람직할 수 있다. OnReceivedHeartbeatRequest 인터페이스와 같은 인터페이스는 서버가 현재 파티션 마스터로부터 수신된 하트비트 메시지에서 추가 정보를 추출하는 방법을 정의할 수 있다. BeforeSendingHeartbeatResponse와 같은 또 다른 인터페이스는 하트비트 메시지에 대한 응답으로 서버가 추가 정보로서 포함할 것을 정의할 수 있다. 이를 통해, 예를 들어, 서버는 로드 밸런싱에서의 사용을 위해 부하 정보를 파티션 마스터에 전달할 수 있다.

[0042] 사용자가 하트비트(또는 기타 메시지) 교환의 일부로서 추가 기능 또는 정보를 포함하면, 사용자는 마스터 롤(530)에 대한 사용자 정의 인터페이스들도 제공할 수 있다. 마스터 롤(530)의 사용자 정의 인터페이스들은 GPMaster(535)로서 개략적으로 도시된 마스터 롤에 대한 고정 인터페이스들의 보완물이다. 마스터 롤(530)의 사용자 정의 인터페이스는 불필요하다.

[0043] 도 5에 도시된 실시예에서, 일반적인 파티셔닝 환경은 5 개의 모듈 및/또는 고정 파티셔닝 시스템 인터페이스 그룹으로 동작할 수 있다. 이는 GPClient 모듈(515), GPServer 모듈(525), GPMaster 모듈(535), GPDictator 모듈(536), 및 GPLease 모듈(545)의 일부로서 도시된 고정 인터페이스들을 포함한다. 물론, 도 5에 도시된 모듈, 인터페이스, 및/또는 기능을 배열하는 다른 방식들도 사용될 수 있다.

[0044] 도 5에 도시된 GPClient 모듈(515)은 클라이언트 애플리케이션이나 사용자(599)가 지정하는 키를 사용자 요청을 처리할 서버에 대한 주소로 번역하게 하는 인터페이스들을 갖고 있다. GPClient 모듈(515)의 인터페이스들은 파티션 지도 혹은 파티션 테이블(559)을 참고하여 이러한 검색을 실행한다. GPClient 모듈 인터페이스들은, 예를 들어, 클라이언트 애플리케이션이 올바른 서버로 요청을 보낼 수 있도록 클라이언트 애플리케이션(599)에 대한 키에 대응하는 서버 주소를 반환할 수 있다.

[0045] GPServer 모듈(525)은 원하는 사용자 태스크를 실행하기 위해 애플리케이션-정의 인터페이스들(520)과 함께 작동하는 인터페이스들을 갖고 있다. GPServer 모듈(525)은 마스터 롤 인스턴스들과 서버 롤 인스턴스들 간의 통신을 위한 인터페이스들을 포함한다. GPServer 모듈 인터페이스들은 각각의 서버 롤 인스턴스와 관련된 리스 객체 및 리스 컨텐츠를 관리하기 위해 GPLease 모듈(545)의 인터페이스들과 통신한다.

[0046] GPMaster 모듈(535)은 마스터 롤 인스턴스들을 관리하는 핵심 기능을 위한 인터페이스들을 제공한다. GPMaster 모듈(535)은 마스터 롤 인스턴스들 중에서 딕테이터 선출, (하트비트 메시지를 통하는 등) 딕테이터와 서버 롤 인스턴스들 간의 통신, 및 파티션 관리를 다룬다. 파티션 관리는 파티션 테이블(559)에 대한 업데이트 제공을 포함할 수 있다. 로드 밸런싱의 경우, GPMaster 모듈(535)의 인터페이스들은 내부 알고리즘을 사용하여 로드 밸

런싱을 실행할 수 있고, 또는 GPMaster는 애플리케이션-정의 인터페이스로서 사용자가 제공하는 대안적인 로드 밸런싱 수식(560)을 수신할 수 있다. 선택적으로, GPMaster 모듈(535)에서 처리하는 메시징 기능은 하나 이상의 별도의 메시징 인터페이스로 실행될 수 있다.

[0047] GPDictator 모듈(536)은 딕테이터로서 동작할 마스터 롤 인스턴스에 관련된 기능을 처리하는 인터페이스들을 제공한다. GPDictator 인터페이스들은 딕테이터쉽을 주장하고(가령, GPMaster 모듈(535)이 행한 선거에서 딕테이터쉽을 획득한 이후), 딕테이터쉽을 해제하고(가령, 장애 조치 이후), 오래된 업데이트를 피하기 위해 시퀀스 또는 에포크 번호의 변경을 처리하는 마스터 롤 인스턴스를 다룰 수 있다.

[0048] GPLease 모듈(545)은 일반 파티셔닝 환경에서의 리스를 관리하는 인터페이스들을 제공한다. 이를 통해 마스터나 서버는 관련 스토리지 객체에, 파티션에, 또는 리스를 얻을 수 있는 임의의 기타 유형의 데이터 구조에 대한 리스를 얻을 수 있다.

[0049] 기본 마스터/서버 관리 프로토콜

[0050] 파티션 마스터가 파티션 서버들과 통신할 수 있는 하나의 방식은 규칙적으로 전송된 킵얼라이브('핑(ping)') 하트비트를 통해서이다. 이런 하트비트는 어떤 파티션도 서빙하지 않는 서버들을 포함하는 모든 서버들에 전송될 수 있다. 서버는 서버가 현재 서빙하고 있는 파티션들의 목록을 가지고 이런 킵얼라이브 하트비트(또는 다른 유형의 하트비트 메시지)에 대응할 수 있다. 마스터는 서버가 올바른 파티션들을 서빙하고 있음을 확인하기 위해 서버 파티션 리스와 함께 하트비트 응답을 사용할 수 있다. 서버가 하나 이상의 파티션을 서빙하고 있을 때, 서버는 바이너리 객체 또는 블랍 등의 자신의 사유 스토리지 객체에 대한 리스 또한 유지한다. 서버가 가동될 때, 어떤 리스도 보유하지 않고, 다만 초기 파티션 할당에 따라 리스를 보유할 것이다. 스토리지 객체 리스는 서버가 서빙하고 있는 파티션들의 목록 역시 포함해야 한다. 하트비트 응답과 스토리지 객체 리스의 정보 중 어느 하나가 마스터가 예상하는 파티션 정보와 다른 경우, 파티션 정보의 충돌이 존재한다.

[0051] 마스터와 서빙되고 있는 파티션들에 관한 서버 사이에 충돌이 일어나면, 그리고 서버가 하나 이상의 파티션을 서빙하려고 시도하고 있으면, 충돌은 치명적인 오류로 간주된다. 예를 들어, 마스터는 서버가 P1 및 P2를 서빙하고 있으면서 서버가 P2, P4 및 P5를 보고하고 있는 것으로 생각할 수 있다. 이런 상황에서, 마스터는 대응 파티션 서버 블랍에 대한 서버의 리스를 중단할 것이다. 마스터는 경보를 발행하고 파티션 테이블에 대한 적절한 상태를 재구축할 것이다. 이는 마스터의 재가동 시에 적절한 상태가 재구축되도록 마스터를 종료시키는 것을 포함할 수 있다.

[0052] 또한, 마스터와 서버가 다르고 어떤 파티션들도 서빙되고 있지 않다고 서버가 보고할 때 충돌이 있게 된다. 그러나, 이는 치명적인 오류로 간주되지 않는다. 이는, 예를 들어, 서버가 두 개의 하트비트 사이에서 장애가 생기거나 또는 여전히 리스를 보유하면서 하나 이상의 하트비트 동안 다운된 후에 재가동되어 다음 하트비트에 대응하는 경우에 발생할 수 있다. 어떤 파티션도 서빙하고 있지 않다고 보고한 서버와 충돌이 발생할 때, 파티션 마스터는 서버의 스토리지 객체를 삭제하려고 시도할 수 있다. 성공한다면, 이용 가능한 다른 서버에 파티션들이 재할당될 수 있다. 그러나, 서버의 이전 인스턴스가 아직 만료되지 않은 경우에 서버의 스토리지 객체의 리스의 삭제가 실패할 수 있다. 따라서, 리스 기간이 다 되어가면 삭제를 다시 시도할 필요가 있다. 한 번의 리스 기간 후의 스토리지 객체 삭제 실패는 전술한 바와 같이 처리될 수 있는 치명적인 오류가 된다. 마스터가 재가동될 때, 마스터는 파티션 할당으로 진행하기 전에 모르는 모든 스토리지 객체들이 확실히 삭제되게 할 것이다.

[0053] 다른 가능한 장애 상황은 서버가 리스를 보유하고 있지만 타임아웃 기간 내에 서버가 ('킵얼라이브'와 같은) 하트비트에 대응하지 않은 경우이다. 또 다시, 마스터는 충돌을 해결하기 위해 서버의 스토리지 객체를 삭제하려 시도할 수 있다. 스토리지 객체가 성공적으로 삭제되면, 서버는 더 이상 기능하지 않는다. 서버가 서빙했던 파티션들이 다른 서버들에게 재할당될 수 있다. 스토리지 객체를 삭제할 수 없는 경우에, 마스터는 파티션 할당에 관한 마스터와 서버 간의 충돌을 확인하기 위해 스토리지 객체 컨텐츠를 판독할 수 있다. 충돌이 없으면, 서버가 계속해서 서빙하고 마스터는 정상적인 하트비트 처리를 통해 곧 서버에 '핑'을 다시 시도할 수 있다. 바람직하게는, 마스터가 스토리지 객체에 대한 서버의 리스를 중단하기 전에 하트비트를 통해 서버와 통신할 수 없는 기간에 대한 제한이 있다. 충돌이 발견되면, 전술한 바와 같이 충돌을 처리할 수 있다.

[0054] 파티션 할당에서, 마스터는 적절한 파티션 서버로 파티션 할당 요청을 하트비트에 피기백한다(이 경우에 하트비트가 가속화된다). 파티션 할당 요청은 서버 스토리지 객체 이름과 서빙될 새로운 파티션 목록 전체로 구성된다. 예를 들어, 현재 P1을 서빙하고 있는 서버에 파티션 P2를 할당하기 위해, 파티션 할당은 P1과 P2로 구성된다. 이는 비할당을 할당에 직교하게 만들어, P1과 P2를 서빙하는 서버에서 P1을 제거하고 간단히 P2만으

로 구성된 할당을 전송한다. 파티션 할당이 일어날 때 마스터는 이미 파티션 테이블을 업데이트 해두었다. 파티션 테이블은 파티션이 서버에 할당될 때(그 전에)만 업데이트되고, 파티션이 서버에서 제거될 때는 업데이트가 불필요하다.

[0055] 파티션 서버는 제 1 파티션이 서버에 할당될 때 시작하는 스토리지 객체 리스를 유지한다. 스토리지 객체 리스는 서버가 죽을 때까지, 또는 파티션 테이블에서의 서버에 대한 할당 정보와 서버에 의해 마스터에 보고된 할당 정보 간에 충돌 등으로 인해 마스터가 리스를 중단시킬 때까지 유지된다. 파티션 할당과 함께 스토리지 객체 이름이 전달된다. 그 후의 모든 파티션 할당은 동일한 스토리지 객체 이름을 포함할 것이다. 서버가 파티션 할당을 수신할 때, 기존의 스토리지 객체 이름이 없는 서버는 파티션 할당에서 제공되는 스토리지 객체 이름을 얻을 것이다. 서버가 이미 이름이 있는 스토리지 객체를 갖고 있는 경우, 서버는 요청에서 제공된 이름과 기존 이름을 비교할 수 있다. 이름이 상이한 경우, 상이한 이름은 파티션 할당 정보에서 충돌을 나타내기 때문에 서버는 경보를 발행하고 종료될 수 있다. 파티션 할당을 수신한 후에, 서버는 관련 정보를 서버에 대한 스토리지 객체에 기록할 수 있다. 관련 정보는, 예를 들어, 서빙할 키 범위, 딕테이터에 대한 에포크 번호, 및/또는 파티션들에 대한 에포크 번호를 포함할 수 있다. 다음으로 서버는 파티션 할당을 제공한 마스터에 대응하고, 새로운 파티션들의 서빙을 시작하고, 제거된 파티션들의 서빙을 중단한다.

[0056] 파티션 할당 후에, 마스터 롤 인스턴스는 할당을 확인하는 서버로부터 응답을 기다릴 것이다. 응답이 할당에 매치하지 않거나, 혹은 응답이 지연되는 경우, 마스터 롤 인스턴스는 서버에 대한 리스를 종료할 수 있다. 그 대신에, 마스터 롤 인스턴스는 서버 상태를 결정하기 위해 서버에 대한 블랍을 조사할 수 있다. 예를 들어, 서버의 스토리지 객체가 할당이 성공했음을 나타내면, 그리고 응답이 부정확한 것이 아니라 단지 지연되거나 손실된 것이면, 마스터 롤 인스턴스는 서버가 차후의 하트비트 또는 기타 메시지에 대해 올바로 대응하는지를 기다려 볼 수 있다. 오류가 발견되고 마스터가 서버에 대한 스토리지 객체 리스를 중단할 수 없는 경우, 마스터는 새로운 마스터 롤 인스턴스에 의한 파티션 지도의 재구성을 강제하기 위해 종료될 수 있다.

[0057] 또한, 서버 롤 인스턴스는 자신이 서빙하고 있는 각각의 범위(즉, 파티션)에 대한 통계를 제공할 수 있다. 일반적인 파티셔닝 시스템에서 통계는 불분명하고, 통계는 이름/값 쌍의 속성 백(property bag)으로서 표현될 수 있다. 이런 선택적인 통계는 사용자가 제공하는 로드 밸런싱 공식을 포함하는 로드 밸런싱 공식에 포함될 수 있다.

[0058] 마스터가 액티브 마스터 혹은 딕테이터가 될 때, 마스터는 먼저 리스를 유지하고 있는 스토리지 객체로부터 새로운 에포크 번호를 얻는다. 그리고 나서 마스터는 시스템의 뷰를 구축하고 모순되는 정보를 정정하기 위해 세 종류의 정보를 수집한다. 먼저, 마스터는 파티션 테이블을 판독한다. 파티션 테이블은 어떤 파티션들이 존재하는지에 대한 진상을 포함한다. 바람직하게는, 마스터는 이전 마스터에 의한 오래된 기록을 막기 위해 파티션 테이블을 판독하기 전에 적어도 파티션 테이블의 연관 부분에 대한 리스를 취득한다. 다음으로, 마스터는 기존의 모든 서버 스토리지 객체들의 목록을 얻는다. 이는 서버 스토리지 객체들의 목록을 유지하거나, 특정 장소에 위치한 모든 서버 스토리지 객체들을 요구하거나, 또는 다른 편리한 방법을 통해 실행될 수 있다. 서버의 현재 파티션 할당에 관해 각각의 서버에 질의하기 위해 하트비트 또는 다른 유형의 메시지를 사용하기도 한다. 이런 질의는 서버에 대한 스토리지 객체의 이름에 관한 질의를 포함한다. 상기의 태스크들은 병렬로 진행될 수 있음을 유의한다.

[0059] 수집된 정보에 기반하여, 마스터는 파티션 테이블의 할당과 각각의 서버가 보고한 할당 간의 모순을 식별할 수 있다. 불일치가 존재하는 경우, 서버 객체에 대한 서버의 리스를 중단하고 마스터를 재가동하는 등의 하나 이상의 정정 액션이 취해질 수 있다. 또한, 파티션 테이블에서 언급되지 않은 서버 스토리지 객체들이 식별되는 경우, 이런 스토리지 객체들은 삭제될 수 있다. 마지막으로, 서버에 의해 확인되지 않은 파티션 테이블의 할당은 새로운 파티션 서버로의 할당을 위해 대기할 수 있다. 충돌을 해결한 후에, 마스터는 정상적인 하트비트 처리, 파티션 할당, 및 기타 임의의 마스터 기능을 시작할 수 있다.

[0060] 로드 밸런싱

[0061] 로드 밸런싱은 대략 세 가지 활동으로 나누어질 수 있다. 로드 밸런싱은 하나의 서버에서 다른 서버로 파티션을 이동시키기, 파티션을 다수의 파티션들로 나누기, 또는 다수의 파티션들을 하나의 파티션으로 합치기를 포함할 수 있다. 일반적으로, 제 1 서버로부터 제 2 서버로 파티션 할당을 변경하는 것은 하나 이상의 메트릭에 기반하여 충분히 높은 부하를 갖고 있는 제 1 서버에 대한 대응일 것이다. 하나의 파티션이 대규모의 부하를 책임지는 경우, 높은 부하가 다수의 서버에 나누어지도록 파티션을 다수의 파티션들로 나눌 수 있다. 파티션들을 합침으로써 적은 활동량을 갖는 파티션들을 결합시킬 수 있다. 이는 데이터 셋의 다양한 파티션들을 추적하고 유지하

는 데 필요한 오버헤드(overhead)를 감소시킨다. 선택적으로, 사용자는 네임스페이스에 대한 파티션들의 개수에 상한선을 정의할 수 있다. 파티션의 개수가 상한선에 근접할수록 파티션들의 병합을 시작하게 되는 임계치가 감소할 수 있다. 파티션의 개수의 상한선은 동적으로 설정될 수 있다.

[0062] 파티션을 언제 나누거나 이동시킬지에 관한 결정의 일례로, 부하에 기반하여 네임스페이스의 모든 파티션들을 소팅할 수 있다. 부하는 파티션에 대한 계산 실행에 관련된 하나 이상의 메트릭을 나타낼 수 있다. 따라서, 부하는 서버 또는 각각의 파티션의 전체 CPU 사용량, 서버 또는 파티션에 사용된 스토리지, 서버 전체에 의해서 혹은 하나의 파티션에 대해서 수신된 요청 개수, 또는 서버에 의해 실행되는 작업량 및/또는 특정 파티션에 대한 작업량을 나타내는 임의의 기타 편리한 값을 나타낼 수 있다. 부하에 근거하여, 평균 파티션 부하의 설정 가능한 배수보다 높은 부하를 갖는 상위 N개의 파티션들이 나뉘어질 수 있다. N은 동적으로 설정 가능하다. 예를 들어, 이는 파티션의 현재 개수의 로그(logarithm)에 기반하는 등 시스템에서 파티션의 현재 개수의 함수이거나, 또는 시스템에서 파티션의 허용 가능한 최대 개수의 함수일 수 있다. 추가적으로 또는 그 대신에, 서버들의 파티션 부하를 합산함으로써 각각의 서버의 부하를 계산할 수 있다. 그 후에 서버들이 부하에 따라 소팅될 수 있고, 서버들 간의 파티션 이동을 위해 평균의 설정 가능한 배수보다 높은 부하를 갖는 상위 N개의 서버들이 선택된다. 마찬가지로, 더 높은 부하의 서버로부터 파티션들을 받기 위해 평균 부하보다 낮은 부하를 갖는 복수의 서버가 선택된다. 바람직하게는, 제 1 서버의 부하가 평균 부하보다 낮게 하지는 않으면서 제 1 서버의 부하를 평균 부하에 가까운 값으로 줄이기 위해 제 1 서버로부터 제 2 서버로 파티션이 이동한다. 이는 모든 파티션들이 상당히 비슷한 부하를 갖고 있는 경우에 실행하기 더 쉽다. 파티션들 간의 부하의 불균형을 줄이기 위해 전술한 바와 같은 파티션 분할을 사용할 수 있다.

[0063] 앞서 언급한 대로, 파티션의 부하는 서버 롤 인스턴스들에서 수집한 통계로부터 얻어질 수 있다. 이런 정보는 예컨대, 하트비트 메시지를 통해 일정한 간격으로 마스터 롤 인스턴스로 전달된다. 바람직하게는, 부하 메트릭이 가산적이도록(additive) 부하 통계가 정의된다. 이를 통해 서버에 대한 부하는 서버에서의 각각의 파티션에 대한 부하들의 합산에 기반하여 결정될 수 있다. 파티션 및/또는 서버에 대한 부하를 결정하기 위한 하나 이상의 공식이 별도의 블랍 또는 기타 스토리지 영역에 저장될 수 있다. 파티셔닝을 위한 규칙이나 수식은 일반 파티셔닝 환경에서 제공하는 기본 규칙일 수 있고, 또는 사용자가 규칙 및/또는 수식을 제공할 수 있다.

[0064] 사용자가 제공하는 로드 밸런싱 규칙 및/또는 수식의 경우, 사용자는 먼저 로드 밸런싱 메트릭으로서 필요한 하나 이상의 메트릭을 식별할 수 있다. 적절한 메트릭의 예시들로 CPU 사용량, 네트워크 대역폭 사용량, 기간 당 처리되는 요청 개수, 또는 기타 임의의 편리한 메트릭을 포함한다. 일부 메트릭은 파티션에 한정되는 반면, 다른 메트릭은 파티션 서버의 모든 파티션들에 대한 값에 대응할 수 있다. 원하는 메트릭에 기반하여, 사용자는 각각의 서버에 대해 원하는 메트릭을 수집하기 위한 하나 이상의 인터페이스를 제공한다. 선택적으로, CPU 사용량이나 기간 당 요청 개수와 같은 보통의 메트릭은 사용자가 간단하게 액세스하는 표준 인터페이스로서 제공될 수 있다. 다음으로, 마스터를 통해 서버의 현재 상태를 확인하기 위해 서버가 사용하는 하트비트 메시지를 비롯한 메시지들을 사용하여 파티션 서버들로부터 수집된 메트릭이 대응 마스터로 전달된다.

[0065] 사용자에 의해 식별된 각각의 메트릭에 대해서, 일반적인 파티셔닝 시스템이 인식한 일련의 값들을 계산할 수 있다. 예를 들어, 일반 파티셔닝 시스템은 사용자가 정의한 변수로서 "차원들(dimensions)"을 인식할 수 있다. 일반적인 파티셔닝 시스템에서 차원은 예상 포맷을 가질 수 있다. 차원은 파티션에 대한 메트릭 값에 대응하는 파티션 메트릭(Partition Metric)에 관한 수식을 포함할 수 있다. 다른 수식은 서버에 대한 메트릭 값에 대응하는 서버 메트릭(Server Metric)에 관한 것일 수 있다. 또 다른 수식은 액션이 실행되는 상황을 정의하는 메트릭에 대한 컨디션(Condition) 값에 관한 것일 수 있다.

[0066] 단순한 상황으로, CPU 사용이 사용자에 의해 하나의 차원으로서 정의될 수 있다. 본 예시에서, CPU 사용 차원은 파티션이 언제 다른 서버로 이동되어야 할 정도로 충분히 바쁜지를 결정하는 데 사용된다. 사용자에 의해 정의된 차원에서, 특정 파티션에 대한 요청 처리 전용인 가상 머신에서 CPU 사용의 백분율이 파티션 메트릭으로 정의된다. 가상 머신에서의 모든 파티션들에 대한 CPU 사용 백분율의 합계가 서버 메트릭으로 정의될 수 있다. 본 예시에서, 컨디션은 전체 CPU 사용량의 80%를 넘는 서버 메트릭 사용으로 정의될 수 있다. 이런 컨디션이 발생할 때, 파티션이 다른 서버로 이동할 수 있다. 파티션 메트릭에 기반하여 이동될 파티션을 선택한다. 파티션 메트릭과 서버 메트릭 모두는 사용자에 의해 정의됨을 유의한다. 따라서, 사용자는 서버 메트릭과 유사한 파티션 메트릭을 갖고 있을 필요가 없다. 예를 들어, 서버 메트릭은 CPU 사용량 및 네트워크 대역폭 사용량의 조합일 수 있는 한편, 파티션 메트릭은 요청율에만 관련될 수 있다.

[0067] 사용자는 파티션의 재할당을 위한 차원 정의뿐만 아니라, 파티션 나누기를 트리거링하기 위한 차원을 정의할 수

도 있다. 파티션 분할을 트리거링하기 위한 차원의 정의는 파티션의 재할당을 위한 차원과 유사할 수 있으며, 또는 다른 차원 포맷이 사용될 수도 있다. 예를 들어, 파티션을 언제 나눌지를 결정하는 데에는 파티션 메트릭 수식이 더 유용할 것이므로, 파티션 분할을 트리거링하기 위한 차원에서는 서버 메트릭 차원이 필요하지 않을 수 있다. 또한, 파티션 분할을 트리거링하기 위한 차원은 파티션을 나누기 위한 컨디션이 충족될 때 파티션을 나누는 방법에 관한 차원을 포함할 수 있다. 파티션 분할을 트리거링하기 위한 차원은 언제 두 개의 파티션들을 합칠 것인지를 식별하는 데에도 유용할 수 있음을 유의한다. 그 대신에, 사용자가 파티션 병합에 관한 별도의 차원을 정의할 수도 있다.

[0068] 보다 일반적으로, 로드 밸런싱 작업이 언제 수행되어야 하는지를 결정하기 위한 편리한 컨디션 개수가 특정될 수 있다. 컨디션은 복수의 차원에 대응하는 메트릭과 같이, 하나 이상의 차원에 대응하는 메트릭을 포함할 수 있다. 컨디션이 로드 밸런싱의 실행을 위한 특정 순서대로 평가되도록, 컨디션이 정렬될 수 있다. 예를 들어, 파티션 분할에 관련된 컨디션은 파티션을 다른 파티션 서버들로 이동시키는 조건보다 순서 상 앞에 배치될 수 있다. 이런 예시에서, 파티션 서버에서 어느 하나의 파티션이 부하의 대부분을 책임지는 경우, 다른 파티션들을 이동시키는 것은 복수의 서버들 사이에서의 로드 밸런싱에 효율적이지 않을 수 있다. 파티션을 나눌지 여부를 먼저 확인함으로써, 문제를 일으킨 파티션이 (아마도) 더 낮은 부하들을 갖는 부분들로 나뉘어질 수 있다. 마찬가지로, 컨디션의 배열에서 일찍 파티션들을 합치는 것이 바람직할 수 있다. 예를 들어, 서버에서의 전체 부하가 평균 아래여도, 다수의 낮은 부하의 파티션들을 갖는 파티션 서버는 과도한 수의 파티션으로 인해 이용 불가능한 것으로 보일 수 있다. 이동 할당 전에 파티션들을 병합함으로써 이러한 서버는 추가 파티션 할당을 수신할 수 있게 된다.

[0069] 로드 밸런싱 활동을 시작하기 위한 컨디션을 정할 때, 편리한 유형의 메트릭이 컨디션에 포함될 수 있다. 따라서, 하나의 파티션에 대한 부하, 복수의 파티션들에 대한 부하, 서버에 대한 부하, 복수의 서버에 대한 부하에 관한 메트릭은 필요한 대로 함께 또는 따로 사용될 수 있다. 복수의 파티션들 또는 복수의 서버들에 관련된 메트릭의 경우, 단순한 예시는 평균 부하를 정의하기 위해서 모든 서버에 대한 부하를 결정하는 것이다. 로드 밸런싱 실행을 위한 컨디션은 서버 부하 대 평균 부하의 절대값의 차이, 또는 평균 서버 부하로부터의 표준 편차를 갖는 서버 부하의 비교 등 서버의 부하와 평균 서버 부하 간의 차이에 관련이 있을 수 있다. 복수의 파티션 부하를 사용할 때, 서로가 서버에 대해 가장 높은 부하를 갖는 많은 파티션들에 대한 부하를 고려하는 것이 바람직할 수 있다. 하나의 높은 부하의 파티션을 갖고 있는 파티션 서버가 아니라 유사한 부하를 갖는 다수의 파티션을 갖고 있는 파티션 서버에서는 바람직한 로드 밸런싱 작업이 다를 수 있다.

[0070] 사용자는, 파티션 재할당, 분할, 및 병합을 위한 차원의 정의뿐만 아니라, 차원에 기반하여 파티션에 대한 액션을 제한하기 위한 한 개 이상의 필터를 정의할 수 있다. 예를 들어, 서버가 70%가 넘는 CPU 사용량 서버 메트릭을 갖거나 혹은 파티션의 개수가 10개 이상일 때, 서버가 새로운 파티션 할당을 받지 않도록 만드는 것이 바람직하다. 이러한 할당을 막는 할당 필터 값(Assignment Filter value)이 정의될 수 있다. 가능한 필터들의 다른 예로 기존 파티션 서버로부터의 파티션 이동을 막거나, 파티션 분할을 막거나, 파티션의 병합을 막는 필터를 들 수 있다. 필터의 유형에 따라서, 필터는 로드 밸런싱 작업이 실행되는 것을 막을 수 있다. 그 대신에, 필터는 컨디션 고려 순서를 바꿀 수도 있고, 또는 필터는 로드 밸런싱 계산 사이클 중에 컨디션을 전부 생략하게 할 수 있다.

[0071] 예를 들어, 모든 서버 요청이 동일한 양의 자원을 사용하는 가상 시스템을 생각해보자. 이런 시스템에서, 적절한 부하 메트릭은 요청율에 기반할 수 있다. 본 예시에서, 각각의 서버 롤 인스턴스는 긴 기간 동안의 요청율(RR_SlowMA)의 평균과 짧은 기간 동안의 요청율(RR_FastMA)의 평균을 수집한다. 이와 같은 요청율 평균은 속성 백의 이름/값 쌍으로서 마스터 롤 인스턴스에 다시 전송된다. 간단한 부하 메트릭이 파티션 메트릭 = max(RR_FastMA, RR_SlowMA)인 로드 밸런싱 규칙의 공식으로 정의될 수 있다. 차원 "부하"에 대한 파티션 메트릭의 좌변은 파티셔닝 시스템의 마스터 컴포넌트가 인식하는 식별자에 대응한다. 이 경우에, 차원 "부하"는 사용자에 의해 미리 식별될 것이다. 우변은 파티션 메트릭에 할당된 부하 값을 생성하는 임의의 수식일 수 있다. 이 경우에, 부하는 복수의 이동 평균 중 어느 하나에 기반하는 요청 개수에 대응한다. 파티션 부하 값 및/또는 기타 값들에 기반하여, 파티션 분할, 병합, 또는 서버들 사이에서의 이동을 실행해야 할지에 관한 하나 이상의 컨디션을 정의할 수 있다.

[0072] 보다 일반적으로, 사용자는 로드 밸런싱 결정을 위한 메트릭 및 (컨디션과 같은) 수식의 임의의 조합을 정의할 수 있다. 로드 밸런싱 결정을 위해 사용자가 정의한 수식은 조건부 논리를 포함할 수 있고 및/또는 다차원적인 제약/최적화 목표를 지원할 수도 있다. 따라서, 사용자는 수식을 언제 어떻게 평가할지에 관한 배열을 제공하기 위해 의사 결정 트리(decision tree)를 정의하거나, 다른 조건부 논리를 사용할 수 있다. 예를 들어, 사용자는

제 1 수식의 평가를 하고 나서, 제 1 수식 값에 기반하여 평가하기 위해 복수의 가능한 추가 수식들에서 선택을 할 수 있다. 이는 "조건문(if-then-else)" 형태의 조건부 논리, 결정된 값에 기반한 다음 수식을 위한 룩업 테이블, 또는 임의의 다른 편리한 유형의 조건부 논리에 기반할 수 있다. 결과적으로, 사용자는, 특정 수식의 평가 여부는 물론 이런 수식의 평가를 위한 순서의 제공 여부를 포함하여, 로드 밸런싱 시에 사용될 메트릭 및 수식의 유형을 지정하는 융통성을 갖고 있다. 수식의 평가 순서는 이전에 평가된 수식 값에 따라 동적으로 결정될 수 있음을 유의한다.

[0073] 로드 밸런싱 결정을 위해 사용자가 정의한 수식의 다른 예로 다차원적인 제약/최적화 목표를 들 수 있다. 예를 들어, 사용자는 복수의 차원(예컨대, 2)을 정의하고, 각각의 차원에 대해 사용자는 별도의 최적화 목표 또는 제약을 정의할 수 있다. CPU 이용 및 요청 레이턴시(latency)가 예시적인 두 차원이다. 사용자는 파티션 서버의 CPU 사용량이 제 1 임계치(예컨대, 90%) 아래에 있도록 규칙을 지정하는 동시에 정의된 파티션 서버 집합(예컨대, 모든 파티션 서버들)에서 평균 요청 레이턴시를 최소화시킬 수 있다. 이런 접근 방법은 사용자가 정확히 무엇을 할지를 지정하는 조건문 형태의 논리와 다를 수 있다. 이런 모델에서는, 사용자가 제한 및 최적화 목표를 정의하고, 이를 통해 시스템은 자동으로 해결책을 찾게 된다.

[0074] 다른 가상의 시스템에서는, 사용자가 평가할 복수의 컨디션 또는 수식을 제공할 수 있다. 수식은 특정 네임스페이스를 서빙하는 파티션 서버들에 대한 다양한 CPU 사용량 메트릭에 기반한다. 제 1 수식은 임의의 파티션 서버의 네임스페이스에 관련된 CPU 사용량이 60%보다 큰지 여부를 평가한다. 이 사용자의 경우, CPU의 사용량이 60%를 넘지 않으면, 사용자는 로드 밸런싱을 원하지 않는다. 따라서, 제 1 수식의 결과가 거짓이면(즉, 어떤 파티션 서버도 60%보다 큰 CPU 사용량을 갖고 있지 않으면), 로드 밸런싱이 필요하지 않으므로 추가 수식이 평가되지 않는다. 적어도 하나의 파티션 서버가 60%를 넘는 CPU 사용량을 갖고 있다면, 실행할 로드 밸런싱 작업을 결정하기 위해 일련의 수식이 평가될 수 있다.

[0075] 로드 밸런싱 결과로 인해 파티션이 이동하는 상황에서는, 마스터 롤 인스턴스가 두 개의 할당 요청을 발행하게 함으로써 파티션이 제 1 서버에서 제 2 서버로 이동할 수 있다. 제 1 서버에 대한 할당 요청에는 파티션이 포함되지 않으므로, 제 1 서버는 그 파티션에 대한 서비스를 중단하게 된다. 제 2 서버에 대한 제 2 할당 요청에는 파티션이 포함된다.

[0076] 파티션이 둘 이상의 파티션들로 나뉘어지는 상황에서는, 마스터 롤 인스턴스가 분할 키(split key)를 결정하여 분할을 시작할 수 있고, 분할 키는 새로운 파티션들 중 어느 하나에 대한 포괄적인 범위의 끝부분을 형성할 키 값에 대응한다. 분할 키는 임의의 편리한 방식으로 선택될 수 있다. 마스터 또는 서버 롤 인스턴스는 파티션의 범위의 중간 또는 그 근처에 있는 키 값을 선택하는 것과 같이 파티션에 기반하여 분할 키를 선택할 수 있다. 그 대신에, 서버가 파티션에 관한 추가 통계에 기반하여 분할 키를 선택할 수 있다. 예를 들어, 전체 파티션에 대한 부하 결정과 유사한 방식으로 파티션의 각종 부분에 대한 부하를 추적하기 위해 샘플링-기반 버킷(bucket) 메커니즘이 사용될 수 있다. 부하가 새로운 파티션에 할당된 버킷에 유사해지도록 분할 키를 선택할 수 있다.

[0077] 특정 롤에서, (액티브) 마스터는 파티션 서버들에 부하를 분산시키는 책임이 있다. 바람직하게는, 마스터는 하나 이상의 서버가 과부화되어 요청을 처리하지 못하게 되는 것을 막을 것이다. 대안적인 실시예에서, 서버 당 한 개의 파티션/범위를 유지하고 이들 범위를 조정함으로써 부하가 조정될 수 있다. 그 대신에 파티션을 이동시킴으로써, 적은 수의 서버에만 영향을 미치면서 부하에 대한 조정이 실행될 수 있다.

[0078] 부하가 파티션 재할당을 통해 매끄럽게 이동할 수 있도록 서버 당 최소 개수의 파티션을 갖는 것이 보통 바람직하다. 파티션 개수가 최소 수준으로 떨어지면, 추가 병합이 실행되지 못한다. 마찬가지로, 너무 많은 파티션을 갖는 것도 피하는 것이 바람직하다. 서버에서 파티션의 최대 개수에 근접할수록, 파티션들의 병합 가능성이 증가할 수 있다. 예를 들어, 서버 당 5개 내지 8개의 파티션들을 유지하는 것이 바람직하다. 물론, 본 발명의 다양한 실시예는 서버 당 1개에서 서버당 수백개까지 서버당 임의의 개수의 파티션들과도 동작할 수 있다.

[0079] 바람직하게는, 분할 및 병합 프로토콜 모두는 상태 비보존형이다(stateless). 마스터 또는 관련 서버(들)은 파티셔닝 시스템에 오류를 일으키지 않고 언제든지 장애가 생길 수 있다. 다시 말해서, 마스터나 서버 중 어느 하나가 분할 또는 병합 프로세스 동안 장애가 생기면, 장애가 생긴 시간에 상관 없이 다음 마스터 또는 서버가 유효한 파티션 할당 목록을 구성할 수 있을 것이다. 상태 비보존형 분할 프로토콜에서, 참여하고 있는 서버는 분할 액션 중 어느 것도 실행할 필요가 없다. 예를 들어, 파티션 테이블은 범위가 로우 키 값 D에서 하이 키 값 H까지 이르는 서버 S1의 파티션을 포함할 수 있다. 본 예시에서, 그 파티션의 에포크 번호는 2이다. 사용자-정의 로드 밸런싱 방정식에 기반해서, 파티션의 일부가 다른 서버에 할당될 수 있도록 파티션이 분할되어야 하는지 여부가 결정된다. 마스터 롤 인스턴스는 서버 S1에 분할 키를 요청한다. 서버 S1은 분할 키로서 키 G를 반환한

다. 다음으로 마스터는 파티션 테이블을 수정한다. 앞서 언급한 하나의 엔트리 대신에, 테이블은 이제 두 개의 파티션을 포함하게 된다. 한쪽은 로우 키 값 D와 하이 키 값 G를 갖고 있는 한편, 두 번째 파티션은 로우 키 값 G와 하이 키 값 H를 갖고 있다. 앞서 언급한 대로, 로우 키 값 및 하이 키 값에 기반한 파티션 범위 정의는 로우 키 값을 포함하지만 하이 키 값은 제외한다. 기존 엔트리를 수정하고 새로운 엔트리를 추가함으로써, 기존 엔트리를 제거하고 새로운 두 개의 엔트리를 추가함으로써, 또는 기타 임의의 편리한 방법에 의해 파티션 테이블에서의 변경이 발생할 수 있다.

[0080] 다음 하트비트 사이클에서, 마스터는 서버 S1이 서빙하는 파티션들과 파티션 테이블의 정보 간의 충돌을 검출한다. 분할이 방금 일어났기 때문에, 마스터가 서버 S1의 블랍 리스를 종료하지 못 했다. 대신에, 마스터는 D에서 G까지의 파티션 범위와 에포크가 3인 할당을 서버 S1에 전송한다. 이는 파티션 테이블의 분할 파티션들 중 어느 하나와 매치하도록 S1에서 파티션의 할당을 수정하게 된다. 서버 S1으로부터 새로운 할당의 접수 통지(acknowledgement)를 수신한 후에, 마스터는 두 번째 분할 파티션을 다른 서버에 할당할 수 있다. 두 번째 할당 파티션의 에포크 번호는 역시 3이 될 것이다. 그 대신에, 분할 파티션 양쪽 모두가 처음에 S1에 할당되었다가, 로드 밸런싱을 실행하기 위해 어느 한쪽의 파티션 혹은 양쪽의 파티션이 나중에 이동할 수도 있다.

[0081] 두 개의 파티션의 병합 역시 상태 비보존형으로 처리될 수 있다. 파티션들이 합쳐질 때, 첫 단계로서 병합을 위한 파티션들은 현재 서버로부터 할당되지 않는다. 예를 들어, 서버 S2의 첫 번째 파티션은 로우 키 값 K와 하이 키 값 M을 가질 수 있다. 본 예시에서, 첫 번째 파티션에 대한 에포크 번호는 7이다. 서버 S2의 두 번째 파티션은 로우 키 값 M과 하이 키 값 N을 가질 수 있다. 본 예시에서 두 번째 파티션에 대한 에포크 값은 9이다. 첫 단계로서, 파티션들이 각각의 서버로부터 할당되지 않아서, 파티션 테이블이 서버에 대한 비-할당 값을 보여줄 수 있다. 다음으로 두 개의 파티션 엔트리가 로우 키 값 K와 하이 키 값 N을 갖는 하나의 엔트리로 대체된다. 이 파티션에 할당된 에포크 번호는 병합된 파티션들의 최고치보다 1만큼 클 수 있고, 본 예시에서는 10에 대응한다. 그 다음에, 새로운 파티션이 서버에 할당될 수 있다.

[0082] 추가 예시

[0083] 본 발명을 설명하기 위한 컨텍스트를 제공하기 위해서, 분산 네트워크 혹은 클라우드 컴퓨팅 환경에서 컴퓨팅 자원들을 체계화하는 일례가 제공된다. 클라우드 컴퓨팅 환경의 다음 설명은 설명적 예시로서 제공된다. 당업자라면 특허청구된 발명이 다른 유형의 체계를 갖는 분산 네트워크 환경과 함께 사용될 수 있음을 인식할 것이다. 설명적 예시에서 아래의 정의가 사용된다.

[0084] "클라이언트"란 네임스페이스 또는 도메인에 대해 애플리케이션-정의 인터페이스에 의한 하나 이상의 액션 요청을 발행하는 롤로서 정의된다. 클라이언트는 사용자 혹은 사용자를 대신해 착수된 프로세스에 대응할 수 있다. 예를 들어, 특정 계정의 조회 요청은 원하는 계정에 대응하는 키를 가지고 모든 계정의 도메인에 대해 실행되는 계정 조회를 위한 애플리케이션으로 보내지는 요청에 대응한다.

[0085] "워크 아이템"이란 클라우드 컴퓨팅 환경에서 실행될 잡의 정적 표현이다. 워크 아이템은 잡 바이너리, 처리될 데이터에 대한 포인터, 및 선택적으로는 잡 실행을 위한 태스크를 시작하기 위한 명령어 라인을 포함하는 잡의 다양한 양태를 특정할 수 있다. 또한, 워크 아이템은 반복 스케줄, 우선 순위 및 제약을 특정할 수도 있다. 예를 들어, 워크 아이템은 매일 오후 5시에 시작되도록 특정할 수 있다.

[0086] "잡"이란 워크 아이템의 실행 중인 인스턴스이다. 잡은 분산 계산을 실행하기 위해 함께 작동하는 일련의 태스크들을 포함한다. 태스크는 클라우드 컴퓨팅 환경의 하나 이상의 가상 머신에서 실행될 수 있다.

[0087] "태스크"란 잡의 기본 실행 유닛이다. 각각의 태스크는 가상 머신에서 실행된다. 사용자는 각각의 태스크에 대한 데이터를 입력하기 위해 명령어 라인 및 포인터에 대한 추가 입력을 특정할 수 있다. 태스크의 실행 중에 태스크를 실행하는 가상 머신의 그 작업 디렉토리 밑에 파일의 위계 구조를 생성할 수 있다.

[0088] 클라우드 컴퓨팅 환경의 사용자는 일반적으로 클라우드 컴퓨팅 자원을 사용하여 잡을 실행하기를 원할 것이다. 잡은 일반적으로 클라우드 컴퓨팅 환경을 통해 액세스 가능한 장소에 저장되어 있는 데이터에 대한 잡의 실행을 포함할 것이다. 운영자가 클라우드 컴퓨팅 환경을 제공하는 한 가지 방법으로 많은 계층으로서 환경을 제공하는 것이 있다. 도 1은 클라우드 컴퓨팅 환경에서 태스크를 실행하기에 적합한 시스템의 일례를 개략적으로 도시한다. 도 1의 시스템은 태스크 런타임 계층(task runtime layer, 110), 제삼자 태스크 런타임 계층(120), 자원 관리 계층(130), 및 스케줄링 및 실행 계층(140)을 포함한다.

[0089] 도 1에 도시된 실시예에서, 태스크 런타임 계층(110)은 사용자(105)로부터의 태스크를 위한 실행 환경 및 보안 컨텍스트(security context)을 셋업할 책임이 있다. 태스크 런타임 계층(110)은 태스크들을 시작하고 태스크들

의 상태를 모니터링할 수도 있다. 태스크 런타임 계층(110)은 각각의 가상 머신에서 실행되는 시스템 에이전트의 형태를 취할 수 있다. 태스크 런타임 계층은 사용자의 태스크 실행 파일에 연결될 수 있는 런타임 라이브러리 또한 포함할 수 있다. 태스크 런타임 계층(110)의 일부로서 런타임 라이브러리를 가짐으로써 시스템 에이전트에 의해 실행되는 태스크들에 풍부한 기능을 잠재적으로 제공할 수 있다. 런타임 라이브러리의 예시로 태스크들 간의 빠른 통신을 가능하게 하는 하나 이상의 효율적인 통신 라이브러리, 다른 가상 머신들 및/또는 다른 태스크들로부터의 파일을 판독하기 위한 효율적인 원격 파일 액세스 라이브러리 지원, 태스크들이 (예컨대, 블랍으로) 체크포인트하고 재개하도록 하는 체크포인트 라이브러리, 로깅(logging) 라이브러리, 및 가상 머신 풀(pool) 내에서 특정 태스크를 실행하는 가상 머신들에서 사용될 분산 파일 시스템을 제공하는 라이브러리를 들 수 있다.

[0090] 제삼자 태스크 런타임 계층(120)은 태스크 런타임 계층(110) 위에 추가 런타임을 구축하여 실행시킨다. 또한 제삼자 태스크 런타임 계층(120)은 잡의 태스크들의 실행을 조정하는 추가 기능을 제공할 수 있다. 예로서 가상 머신 풀 내에서 특정 태스크를 실행하는 가상 머신들에서 사용될 분산 파일 시스템을 제공하는 라이브러리에 대한 맵리듀스(MapReduce) 런타임을 포함할 수 있다. 이를 통해 사용자는 자신의 잡 또는 태스크에 맞는 방식으로 클라우드 컴퓨팅 환경을 체계화시킬 수 있다. 몇몇 실시예에서, 잡 관리자 태스크는 클라우드 컴퓨팅 자원을 실행 및/또는 제어하기 위해 사용자가 제삼자 런타임 계층의 사용하는 것을 용이하게 해줄 수 있다.

[0091] 자원 관리 계층(130)은 클라우드 컴퓨팅 환경에서 이용 가능한 컴퓨팅 자원의 관리를 다룬다. 한 가지 옵션은 자원 관리 계층(130)이 세 개의 다른 레벨로 자원을 관리하게 하는 것이다. 첫 번째 레벨에서, 자원 관리 계층(130)은 잡(즉, 워크 아이템의 실행)에 관련된 가상 머신들은 물론 태스크에 관련된 각각의 가상 머신에 저장된 파일들의 할당(allocation) 및 반납(deallocation)을 관리한다. 두 번째 레벨에서, 잡에 관련된 가상 머신들이 머신 풀로 그룹화될 수 있다. 풀은 하나 이상의 잡 및/또는 워크 아이템과 관련된 가상 머신들을 포함할 수 있다. 실시예에 따라서, 하나의 풀이 데이터 센터의 모든 가상 머신 클러스터들, 하나의 지리적 영역(geographic region) 내의 복수의 데이터 센터에서의 복수의 클러스터들, 또는 복수의 지리적 영역에서의 데이터 센터들의 복수의 클러스터들과 같은 다수의 가상 머신 클러스터를 포함할 수 있다. 하나의 풀이 수백만 개의 많은 가상 머신을 포함할 수도 있다. 가상 머신들은 수십억 개의 대규모 풀에 포함될 수 있다. 세 번째 레벨에서, 자원 관리 계층은 특정 풀 그룹에서의 잡 또는 워크 아이템과 연계 가능한 가상 머신의 수를 관리한다. 이를 통해 시스템의 현재 부하에 기반하여 사용되는 컴퓨팅 자원량의 동적 조정이 가능해진다. 또한, 현재 풀 그룹에서 사용되고 있지 않는 가상 머신들은 다른 풀 그룹과의 통합을 위해 다시 클라우드 컴퓨팅 환경에 돌려보내진다.

[0092] 도 1에 도시된 실시예에서, 스케줄링 및 실행 계층(140)은 사용자에 의해 실행 중인 워크 아이템, 잡, 및 태스크를 관리한다. 스케줄링 및 실행 계층(140)은 스케줄링 결정을 하고, 잡과 태스크의 시작은 물론 장애 시 재시도를 책임진다. 이런 스케줄링 및 실행 계층(140)은 다양한 레벨로 잡 및/또는 태스크를 관리하기 위한 컴포넌트를 포함할 수 있다.

[0093] 전술한 계층들은 다수의 지리적인 위치에서의 프로세서들을 포함하는 클라우드 컴퓨팅 환경에서 구현될 수 있다. 도 2는 다른 장소에 있는 프로세서들이 하나의 클라우드 컴퓨팅 아키텍처 내에 통합되는 방법의 일례를 개략적으로 도시한다.

[0094] 도 2에서, 가상 머신의 풀을 관리하기 위해 하나 이상의 태스크 테넌트(215)가 사용될 수 있다. 태스크 테넌트(215)는 일련의 가상 머신을 유지할 수 있다. 하나 이상의 사용자의 잡이 가상 머신의 하나 이상의 풀의 일부로서 태스크 테넌트(215) 내의 가상 머신에서 실행될 수 있다. 특정 지리적 영역에서 하나 이상의 태스크 테넌트(215)가 사용될 수 있다. 태스크 테넌트(215)의 책임에 일련의 가상 머신을 유지하기 및 태스크 테넌트 내에서의 자원 이용에 따라 태스크 테넌트를 동적으로 늘리거나 줄이기가 포함될 수 있다. 이를 통해 태스크 테넌트(215)는 증가된 고객 요구도를 수용하기 위해 태스크 테넌트 내의 가상 머신의 개수를 증가시킬 수 있다. 또한, 이를 통해 태스크 테넌트(215)는 다른 고객들을 위한 서비스를 다루는 데이터 센터에서 호스팅된 다른 서비스들에 가상 머신이 할당될 수 있도록 미사용 가상 머신을 해제할 수 있다. 태스크 테넌트(215)의 다른 책임은 풀 할당/반납/관리 논리의 일부를 구현하는 것일 수 있다. 이를 통해 태스크 테넌트(215)는 고객을 위한 태스크에 관련된 풀에 가상 머신을 할당하는 방법의 결정에 참여할 수 있다. 태스크 테넌트(215)는 또한 태스크 테넌트 내의 가상 머신에서의 태스크 스케줄링 및 실행을 책임질 수 있다.

[0095] 도 2에 도시된 실시예에서, 복수의 태스크 테넌트(215)를 제어하는 하나 이상의 태스크 위치 서비스(225)가 제공된다. 복수의 태스크 테넌트는 특정 지리적 영역에서의 모든 태스크 테넌트, 세계로부터의 다양한 태스크 테넌트, 또는 기타 임의의 편리한 태스크 테넌트 그룹화에 대응할 수 있다. 도 2에서, 태스크 위치 서비스(225)는

“미국 북부" 및 “미국 남부”로 표시된 지역을 서빙하는 것으로 도시된다. 태스크 위치 서비스(225)의 책임은 특정 지리적 영역에 대한 태스크 계정의 관리를 포함할 수 있다. 태스크 위치 서비스(225)는 사용자들이 클라우드 컴퓨팅 환경과 인터랙션할 수 있게 하는 애플리케이션 프로그래밍 인터페이스(API)를 제공할 수도 있다. 이런 API는 가상 머신의 풀에 관련된 API 처리, 풀 관리 논리, 및 특정 지리적 영역의 태스크 테넌트에 대한 풀 관리 논리의 조정을 포함할 수 있다. 또한, API는 사용자가 제출한 태스크 처리는 물론, 사용자 태스트와 관련된 워크 아이템이나 잡의 유지, 스케줄링, 및 종료를 포함할 수 있다. API는 지리적 영역의 모든 워크 아이템, 잡, 태스크 및 풀에 대한 통계 수집, 종합, 및 보고를 위한 API를 더 포함할 수 있다. 게다가, API는 가상 머신의 현물 시장에 기반하여 단기간 동안 사용자에게 선점형(preemptible) 가상 머신으로서 이용 가능한 가상 머신의 경매를 할 수 있게 하는 API를 포함할 수 있다. API는 사용량 측정 및 결제 지원 제공을 위한 API도 포함할 수 있다.

[0096] 태스크 위치 서비스(225)는 글로벌 위치 서비스(235)에 의해 서로 연결될 수 있다. 글로벌 위치 서비스(235)는 계정 생성, 및 태스크 위치 서비스 테넌트(225)와 함께 태스크 계정을 관리하는 것을 포함하는 계정 관리를 책임질 수 있다. 이는 재해 복구에 대한 책임, 및 주요 데이터 센터 재해가 있는 경우에 워크 아이템 및 잡의 가용성에 대한 책임을 포함한다. 이는 어떤 이유로든 데이터 센터의 이용 불가로 인해 다른 장소에서 워크 아이템이나 잡을 실행하는 것을 포함할 수 있다. 이는 또한 고객들이 그들의 워크 아이템, 잡, 및 풀을 한 데이터 센터에서 다른 데이터 센터로 이전시키는 것도 포함할 수 있다. 일반적으로 오직 한 개의 액티브 글로벌 위치 서비스(235)가 있을 것이다. 이런 액티브 글로벌 위치 서비스(235)는 다양한 태스크 위치 서비스(225)는 물론, 데이터 저장을 관리하는 서비스 컴포넌트들(도시되지 않음)과도 통신하고 있다. 글로벌 위치 서비스는 글로벌 계정 네임스페이스(237)를 유지할 수 있다.

[0097] 도 3은 태스크 위치 서비스에 대해 가능한 구성을 도시한다. 도 3에 도시된 구성에서, 태스크 위치 서비스는 하나 이상의 계정 서버(321)를 포함할 수 있다. 계정 서버는 생성, 삭제, 또는 속성 업데이트를 포함하는, 특정 지리적 영역의 계정들에 대한 계정 관리를 처리한다. 계정 프론트 엔드(322)는 계정 서비스의 프론트 엔드 노드로서 동작한다. 계정 프론트 엔드(322)는 도면에 도시된 바와 같이 계정 가상 IP 주소(324) 뒤에 있다. 계정 프론트 엔드(322)는 계정 생성 또는 계정 삭제의 API 요청을 비롯해 글로벌 위치 서비스로부터 들어오는 계정 IP 요청을 처리한다.

[0098] 또한 도 3의 구성은 하나 이상의 풀 서버(331)를 포함한다. 풀 서버(331)는 특정 지리적 영역의 가상 머신 풀에 대한 풀 관리 및 풀 트랜잭션을 다룬다. 풀 서버(331)는 풀 생성, 삭제 및 속성 업데이트를 다룬다. 풀 서버(331)는 또한 다수의 태스크 테넌트들에 대한 상위 가상 머신 할당 알고리즘을 관리한다. 가상 머신 할당에서는 가상 머신의 특정 사용자에 대한 스토리지와의 연결을 고려할 수 있다. 풀 서버는 가상 머신의 할당과 관련된 다른 태스크들도 실행할 수 있다.

[0099] 하나 이상의 워크 아이템 또는 잡 서버(WIJ, 336)가 도 3의 구성에 포함된다. WIJ 서버(336)는 워크 아이템 또는 잡의 생성, 삭제, 및 업데이트를 다룬다. 또한, 워크 아이템이나 잡이 시작하거나 종료될 때 사용자가 풀의 자동 생성 및/또는 해체를 요청하면, WIJ 서버(336)는 그 워크 아이템이나 잡에 관련된 풀의 생성 및 삭제를 시작할 수 있다. WIJ 서버(336)는 또한 확장(scaling)에 관한 일반 파티셔닝 메커니즘을 사용한다. 일 실시예에서, 각각의 태스크 위치 서비스에 다수의 WIJ 서버(336)가 있고, 각각의 WIJ 서버는 워크 아이템 범위를 다룬다.

[0100] 풀 서버(331)와 W IJ 서버(336)는 태스크 위치 서비스 프론트 엔드(338)를 통해 사용자로부터 요청을 수신한다. 태스크 위치 서비스 프론트 엔드(338)는 사용자로부터의 요청을 처리하기 위해 대응 컴포넌트를 호출하는 책임이 있다. 태스크 위치 서비스 프론트 엔드(338)는 도면에 도시된 바와 같이 계정 가상 IP 주소(334) 뒤에 있다.

[0101] 도 3의 구성은 태스크 위치 서비스 마스터(342)를 더 포함한다. 일 실시예에서, 태스크 위치 서비스 마스터(342)는 두 가지 주된 책임을 갖는다. 먼저, 태스크 위치 서비스 마스터(342)는 태스크 위치 서비스(225)의 대응 서버에 대한 파티셔닝 논리를 구현하는 마스터 시스템으로서 동작한다. 또한, 태스크 위치 서비스 마스터(342)는 태스크 위치 서비스의 전체 지리적 영역에 대한 각 현물 주기(spot period)의 처음에 선점형 가상 머신의 새로운 시장 가격을 계산할 책임을 질 수 있다. 태스크 위치 서비스 마스터는 풀 서버 및 태스크 테넌트로부터 현재 입찰 및 자원 가용성 정보를 수집할 수 있고, 그에 따라 새로운 시장 가격을 계산한다. 그 대신에, 태스크 위치 서비스 마스터는 입찰 및 자원 가용성 정보를 현물 가격 마켓 서비스에 전송할 수 있다. 태스크 위치 서비스 마스터는 하나의 지리적 영역의 모든 태스크 테넌트들에 대한 선점형 가상 머신에 대한 상위 레벨 할당 안내를 풀 서버에게 하게 된다.

[0102] 컴퓨팅 환경의 활동 및 행동을 추적하기 위해, 태스크 위치 서비스 마스터(342)는 하나 이상의 통계 종합 서버(statistics aggregation server, 355)와 통신할 수 있다. 통계 종합 서버는 태스크, 잡, 워크 아이템 및 풀에 대한 상세한 통계를 수집 및 종합하는 책임을 진다. 시스템의 다른 컴포넌트들이 태스크 및 가상 머신에 대한 세립(fine-grained) 통계를 낼 수 있다. 통계 종합 서버는 태스크 레벨이나 가상 머신 레벨 통계로부터의 이들 세립 통계를 워크 아이템, 계정 레벨, 및/또는 풀 레벨 통계로 종합한다. 통계는 사용할 수 있게 API를 통해 노출될 수 있다. 게다가, 통계 종합 서버는 결제 시에 사용되도록 각각의 계정에 대한 매시간 측정 기록을 만들어 낼 책임이 있을 수 있다.

[0103] 좀 더 구체적인 예를 들면, 일반 파티셔닝은 도 3에 도시된 태스크 위치 서비스에서의 롤 및 서브-롤에 적용될 수 있다. 도 3에 도시된 최상위 롤은 태스크 위치 서비스 또는 테넌트이다. 태스크 위치 서비스의 다수의 인스턴스들이 존재하면, 그 인스턴스들 중 어느 하나가 태스크 위치 서비스 마스터(또는 딕테이터)(342)에 대응할 것이다. 계정 서버 롤(321), 풀 서버 롤(332), 및 워크 아이템-잡 서버 롤(336)이 테넌트 안에 있다. 또한 이들 롤 각각은 태스크 위치 서비스의 인스턴스를 나타내는 반면, 이들 롤 인스턴스는 전체 테넌트 내에서의 일련의 기능을 처리한다. 예를 들어, 테넌트의 계정 서버 롤에 의해 계좌 정보 요청이 처리된다. 태스크 위치 서비스 또는 테넌트의 다수의 인스턴스들이 존재하는 경우, 테넌트 안의 각각의 롤에 대한 마스터는 다른 인스턴스에 대응할 수 있다.

[0104] 도 6은 다수의 마스터 롤들에 대해 중복성을 제공할 수 있는 방법의 종래의 예시를 도시한다. 이런 종래의 예시에서, 각각의 마스터 롤은 가용성을 향상시키기 위해 두 개의 추가 인스턴스를 가질 필요가 있다. 결함 도메인은 공통의 장애 패턴을 갖고 있는 노드들을 포함하고, 이 노드들은 함께 장애가 발생할 수 있다. 예를 들어, 동일한 전원을 공유하는 동일한 랙의 노드들은 공통의 문제로 인해 장애가 발생할 수 있으므로, 이 노드들은 공통 결함 도메인에 있을 수 있다. 업그레이드 도메인은 시스템 업그레이드 동안 동시에 오프라인이 될 수 있는 일련의 노드들에 대응한다. 도 6에 도시되어 있는 바와 같이, 업그레이드 혹은 장애로 인해 공통된 시간에 다운되지 않기 위해 이 롤들은 다른 "결함 도메인" 및 "업그레이드 도메인"에 퍼져 있다.

[0105] 종래의 방법에서, 태스크 위치 서비스의 세 개의 롤들에 필요한 추가 인스턴스들을 제공하려면 각각의 롤에 대해 별도의 추가 인스턴스를 마련할 필요가 있다. 도 6에서, 이는 각각의 유형의 마스터에 대한 추가 인스턴스들을 제공하는 명시적인 머신을 구비하는 것으로 도시된다. 따라서, 계정 서버에 대한 마스터(621)는 추가 인스턴스들(622 및 623)을 필요로 할 것이다. 마찬가지로, 풀 서버에 대한 마스터(632)는 백업 인스턴스들(631 및 633)을 갖고 있다. WIJ 서버에 대한 마스터(643)는 백업 인스턴스들(642 및 641)을 갖고 있다.

[0106] 도 8은 롤에 대한 다양한 인스턴스와 마스터를 제공하기 위해 일반 파티셔닝을 사용하는 분산 컴퓨팅 환경에서 가상 머신을 체계화시키는 방법을 일례를 도시한다. 도 8에서, 계정 서버, 풀 서버, 및 WIJ 서버에 대해 서로 다른 GP 마스터들(821, 831, 및 841)이 각각 도시된다. 관리 중인 롤에 상관 없이 GP 마스터 모듈과 임의의 고정 인터페이스들이 동일하기 때문에, 필요한 GP 마스터들(821, 831, 및 841)에 대한 백업 서버가 하나의 머신에서 결합될 수 있다. 따라서, 세 개의 GP 마스터들에 대한 백업으로서 하나의 백업(852)이 제공될 수 있다. GP 마스터들(821, 831, 및 841) 중 어느 하나에 장애가 생기면, 동일한 GP 마스터 모듈과 고정 인터페이스들이 사용될 수 있다. 본 예시에서, 장애가 생긴 GP 마스터 롤을 대체하기 위해 장애 조치 백업에서 필요한 유일한 추가 정보 유형이 대응 네임스페이스 및 임의의 애플리케이션-정의 인터페이스들에 관한 키이다. 마찬가지로, 세 개의 GP 마스터들(821, 831, 및 841) 전부에 대해 하나의 제 2 백업(853)이 사용될 수 있다. 그 결과, 본 예시에서, 적어도 세 개의 GP 마스터 롤들에 대해 두 개의 GP 마스터 백업 서버들(852 및 853)만이 사용된다. 세 개의 GP 마스터 롤이 공통 머신에 의해 백업된 것으로 도시되어 있지만, 동일한 사용자나 계정에 속하는 임의의 편리한 개수의 GP 마스터 롤들이 공통 머신에 의해 백업될 수 있다.

[0107] 도 11은 본 발명의 양태에 따른, 결함 도메인과 업그레이드 도메인에 관한 마스터 롤에 대해 백업 머신을 제공하는 예시적인 양태를 도시한다. 도 8과 관련하여 상기에서 논의한 개념과 유사하게, 다수의 GP 마스터 롤이 더 적은 개수의 서버에 대해 백업될 수 있다. 예를 들어, 도 11은 제 1 결함 도메인 및 제 1 업그레이드 도메인의 계정 GP 마스터(1202), 제 2 결함 도메인 및 제 2 업그레이드 도메인의 풀 GP 마스터(1204), 및 제 3 결함 도메인 및 제 3 업그레이드 도메인의 WIJ GP 마스터(1210), 제 1 GP 백업(1206) 및 제 2 GP 백업(1208)을 도시한다. 제 1 GP 백업(1206) 및 제 2 GP 백업(1208)은 각각 GP 마스터 롤들과 다른 결함 도메인과 업그레이드 도메인에 있다. 도시된 예시에서, 세 개의 롤에 대한 마스터들 전부를 호스트하기 위해 하나의 일반 파티셔닝 시스템은 5 개의 서버(또는 4 개의 서버와 한 개의 백업도 가능함)만을 필요로 한다. 도 6에 도시된 예시에서, 세 개의 동일한 마스터 롤에 대해 9 개의 다른 서버들이 필요할 수 있다. 도 8에 도시된 접근 방법은 시스템이 호스트하고 있는 임의의 유형의 롤에 대해 사용될 수 있는 두 개의 추가 서버들을 이용하여 수행될 수 있다. 따

라서, 결함 도메인의 장애 또는 이용 불가능한 업그레이드 도메인으로 인해 하나 이상의 마스터 롤이 이용 불가능해지면 백업 서버(예컨대, GP 백업(1206))가 사용될 수 있다. (도 6에 관해 논의된 것과 비교하여) 본 예시에서는 더 적은 수의 서버가 필요로 하지만, 백업 서버의 가용성을 보장하기 위해 추가 결함 도메인과 업그레이드 도메인을 구현하는 경우도 고려된다. 앞서 논의된 도 8에서, 예시적인 양태로, 공통 머신에 의해 임의의 개수의 마스터 롤이 백업되는 경우도 고려된다.

[0108] 본 발명의 실시예들의 개관을 간략하게 설명하였으며, 본 발명의 실행에 적합한 예시적인 운영 환경에 대해 이제부터 설명한다. 일반적으로 도면을 참조하면, 특히 처음에는 도 7을 참조하면, 본 발명의 실시예를 구현하기 위한 예시적인 운영 환경은 일반적으로 컴퓨팅 장치(700)로써 도시되고 설계된다. 컴퓨팅 장치(700)는 적합한 컴퓨팅 환경의 일례에 불과하며, 본 발명의 사용 또는 기능의 범위에 대한 어떠한 한정도 제시하는 것이 아니다. 또한 컴퓨팅 장치(700)는 설명될 컴포넌트들 중 어느 하나 또는 이들의 조합과 관련된 어떠한 종속성 또는 요건을 갖는 것으로 해석되어서는 안 된다.

[0109] 본 발명의 실시예는 일반적으로, PDA(personal data assistant) 또는 기타 핸드헬드 장치와 같은 컴퓨터 또는 기타 기계에 의해 실행되는, 프로그램 모듈과 같은 컴퓨터 실행 가능 명령어를 포함하는 컴퓨터 코드 또는 기계-사용 가능 명령어로 기술될 수 있다. 일반적으로, 루틴, 프로그램, 객체, 컴포넌트, 데이터 구조 등을 포함하는 프로그램 모듈은 특정 작업을 수행하거나 특정 추상 데이터 유형을 구현하는 코드를 지칭한다. 본 발명은 핸드-헬드 장치, 소비자 가전, 범용 컴퓨터, 더 특수화된 컴퓨팅 장치 등과 같은 다양한 시스템 구성에서 실시될 수 있다. 본 발명은 또한, 통신 네트워크를 통해 연결된 원격-처리 장치에 의해 작업이 수행되는 분산 컴퓨팅 환경에서 실시될 수 있다.

[0110] 도 7을 계속 참조하여, 컴퓨팅 장치(700)는 다음의 장치들, 메모리(712), 하나 이상의 프로세서(714), 하나 이상의 프레젠테이션 컴포넌트(presentation component)(716), 입/출력(I/O) 포트(718), I/O 컴포넌트(720), 및 예시적인 전원(722)을 직접 또는 간접적으로 연결하는 버스(710)를 포함한다. 상기 버스(710)는 하나 이상의 버스들(예컨대, 주소 버스, 데이터 버스, 또는 이들의 조합)일 수 있는 것을 나타낸다. 도 7의 다양한 블록은 간결성을 위해 선으로 나타나지만, 실제로, 다양한 컴포넌트를 묘사하는 것이 그렇게 명료하지 않고, 비유적으로, 선들은 더 정확하게 애매하고(grey) 흐려질(fuzzy) 것이다. 예를 들어, 디스플레이 장치와 같은 프레젠테이션 컴포넌트가 I/O 컴포넌트인 것을 고려할 수 있다. 추가로, 많은 프로세서들이 메모리를 가진다. 본 발명자는 이러한 것이 해당 분야의 속성이며, 도 7의 다이어그램은 본 발명의 하나 이상의 실시예와 함께 사용될 수 있는 예시적인 컴퓨팅 장치의 예에 불과함을 강조한다. "워크스테이션", "서버", "랩톱", "핸드헬드 장치" 등의 카테고리들 간에 구분이 없는데, 이는 이들 모두 도 7의 범위 내에 있고 "컴퓨팅 장치"를 지칭하기 때문이다.

[0111] 일반적으로 컴퓨팅 장치(700)는 다양한 컴퓨터 판독 가능 매체를 포함한다. 컴퓨터 판독 가능 매체는 컴퓨팅 장치(700)에 의해 액세스될 수 있고 휘발성 및 비휘발성 매체, 이동식 및 비이동식 매체 모두를 포함하는 임의의 이용 가능한 매체일 수 있다. 예를 들어, 그러나 제한 없이, 컴퓨터 판독 가능 매체는 컴퓨터 저장 매체 및 통신 매체를 포함할 수 있다. 컴퓨터 저장 매체는 컴퓨터 판독 가능 명령어, 데이터 구조, 프로그램 모듈, 또는 기타 데이터와 같은 정보의 저장을 위한 임의의 방법 또는 기술로 구현되는 휘발성 및 비휘발성, 이동식 및 비이동식 매체를 포함한다. 컴퓨터 저장 매체는, RAM(Random Access Memory), ROM(Read Only Memory), EEPROM(Electronically Erasable Programmable Read Only Memory), 플래시 메모리 또는 기타 메모리 기술, CD-ROM, DVD(digital versatile disk) 또는 기타 홀로그램 메모리, 자기 카세트, 자기 테이프, 자기 디스크 저장 장치 또는 기타 자기 저장 장치, 또는 원하는 정보를 인코딩하기 위해 사용될 수 있고 컴퓨팅 장치(700)가 액세스할 수 있는 기타 임의의 매체를 포함하지만, 이에 국한되지는 않는다. 일 실시예에서, 컴퓨터 저장 매체는 유형의 컴퓨터 저장 매체로부터 선택될 수 있다. 또 다른 실시예에서, 컴퓨터 저장 매체는 비 일시적 컴퓨터 저장 매체로부터 선택될 수 있다.

[0112] "통신 매체"는 통상적으로 컴퓨터 판독 가능 명령어, 데이터 구조, 프로그램 모듈 또는 반송파 또는 기타 전송 메커니즘과 같은 변조 데이터 신호(modulated data signal)의 기타 데이터를 구현한다. "변조 데이터 신호"라는 용어는, 신호 내의 정보를 인코딩하도록 그 신호의 특성들 중 하나 이상을 설정 또는 변경시킨 신호를 의미한다. 예를 들어, 그러나 제한 없이, 통신 매체는 유선 네트워크 또는 직접 배선 연결과 같은 유선 매체, 그리고 음향, RF, 적외선, 기타 무선 매체와 같은 무선 매체를 포함한다. 또한, 상기 매체들의 모든 조합이 컴퓨터 판독 가능 매체의 범위 안에 포함된다.

[0113] 메모리(712)는 휘발성 및/또는 비휘발성 메모리의 형태로 된 컴퓨터-저장 매체를 포함한다. 상기 메모리는 이동식, 비이동식, 또는 이들의 조합일 수 있다. 예시적인 하드웨어 장치는 고체 상태 메모리, 하드 드라이브, 광-

디스크 드라이브 등을 포함한다. 컴퓨팅 장치(700)는 메모리(712) 또는 I/O 컴포넌트(720)와 같은 다양한 엔티티들로부터 데이터를 판독하는 하나 이상의 프로세서를 포함한다. 프레젠테이션 컴포넌트(들)(716)는 사용자 또는 다른 장치에게 데이터 표시를 제공한다. 예시적인 프레젠테이션 컴포넌트는 디스플레이 장치, 스피커, 인쇄 컴포넌트, 진동 컴포넌트 등을 포함한다.

[0114] I/O 포트(718)는 컴퓨팅 장치(700)가 I/O 컴포넌트(720)를 포함하는 다른 장치들에 논리적으로 연결되도록 하며, 이들 중 일부는 내장될 수 있다. 예시적인 컴포넌트는 마이크로폰, 조이스틱, 게임 패드, 위성 접시, 스캐너, 인쇄기, 무선 장치 등을 포함한다.

[0115] 도 9는 본 발명에 따른 방법의 일례를 도시한다. 도 9에서, 하나 이상의 애플리케이션-정의 파티셔닝 시스템 인터페이스들이 애플리케이션 또는 사용자로부터 수신된다(910). 예를 들어, 애플리케이션 또는 사용자로부터의 요청에 기반하여, 수신된 애플리케이션-정의 파티셔닝 시스템 인터페이스들을 포함하는 복수의 마스터 롤 인스턴스들이 생성된다(920). 복수의 마스터 롤 인스턴스들은 마스터 스토리지 객체에 대응한다. 마스터 스토리지 객체(930)에 대한 리스가 마스터 롤 인스턴스들 중 어느 하나에 할당된다. 복수의 마스터 롤 인스턴스들이 리스 할당을 위해 경쟁한다. 리스가 할당된 마스터 롤 인스턴스가 딕테이터 마스터 롤 인스턴스가 된다. 딕테이터 마스터 롤 인스턴스는 복수의 파티션 서버들에 파티션 그룹을 할당한다(940). 애플리케이션에 대응하는 잡들이 복수의 파티션 서버들을 사용하여 실행된다(950).

[0116] 도 10은 본 발명에 따르는 방법의 다른 예를 도시한다. 도 10에서, 계산 요청이 수신된다(1010). 계산 요청은 복수의 네임스페이스와 서로 다른 적어도 두 개의 마스터 롤 인스턴스를 포함한다. 적어도 두 개의 마스터 롤 인스턴스가 생성된다(1020). 생성된 복수의 마스터 롤 인스턴스에 대한 장애 조치 서비스를 제공하는 적어도 하나의 머신이 할당된다(1030). 장애 조치 서비스는 계획에 없던 장애 조치, 예정된 업데이트, 계획된 보수 이벤트, 또는 기타 사유와 같이 마스터 롤 인스턴스가 동작을 중단하는 임의의 편리한 사유에 해당할 수 있다. 생성된 마스터 롤 인스턴스들 중 어느 하나에 대한 장애 조치 이벤트가 검출된다(1040). 할당된 머신에서 장애 조치 이벤트에 대응하는 마스터 롤의 추가 인스턴스가 생성된다(1050).

[0117] 본 발명은 특정 실시예들에 관해 설명되었으며, 이 실시예들은 제한하기보다는 예시를 들기 위한 것이다. 본 발명이 그 범위를 벗어나지 않는 한, 본 발명에 관련된 다른 실시예들도 당업자들에게 자명할 것이다.

[0118] 일 실시예에서, 분산 컴퓨팅 환경에서 계산을 실행하는 방법이 제공된다. 상기 방법은 하나 이상의 애플리케이션-정의 파티셔닝 시스템 인터페이스를 수신하는 단계, 하나 이상의 애플리케이션-정의 파티셔닝 시스템을 포함하는 복수의 마스터 롤 인스턴스를 생성하는 단계 - 마스터 롤 인스턴스들은 마스터 스토리지 객체에 대응함 -, 마스터 스토리지 객체에 대한 리스를 할당하는 단계 - 각각의 마스터 롤 인스턴스는 리스를 위해 경쟁하고, 리스가 할당된 마스터 롤 인스턴스가 딕테이터 마스터 롤 인스턴스임 -, 딕테이터 마스터 롤 인스턴스에 의해, 복수의 파티셔닝 서버에 파티션 그룹을 할당하는 단계, 및 복수의 파티셔닝 서버를 사용하여 애플리케이션에 대응하는 하나 이상의 계산을 실행하는 단계를 포함한다.

[0119] 다른 실시예에서, 분산 컴퓨팅 환경에서 계산을 실행하는 방법이 제공된다. 상기 방법은 적어도 두 개의 네임스페이스 및 적어도 두 개의 마스터 롤 인스턴스를 포함하는 계산에 대한 요청을 수신하는 단계, 적어도 두 개의 마스터 롤 인스턴스를 생성하는 단계, 생성된 복수의 딕테이터 마스터 롤 인스턴스에 대해 장애 조치 서비스를 제공하는 적어도 하나의 머신을 할당하는 단계, 생성된 딕테이터 마스터 롤 인스턴스들 중 어느 하나에 대한 장애 조치 이벤트를 검출하는 단계, 및 할당된 머신에, 검출된 장애 조치 이벤트에 대응하는 마스터 롤의 추가 인스턴스를 생성하는 단계를 포함한다.

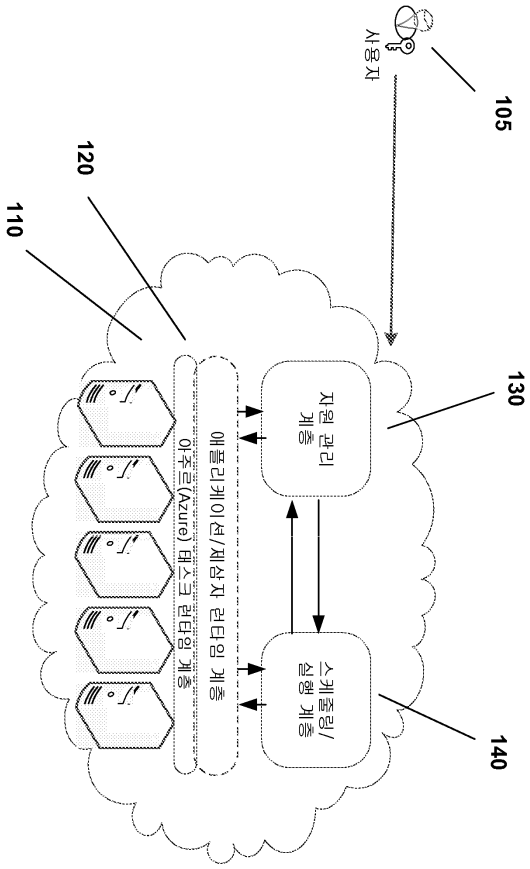
[0120] 또 다른 실시예에서, 분산 컴퓨팅 환경에서 태스크를 계산을 실행하는 시스템이 제공된다. 상기 시스템은 복수의 파티션 서버 - 파티션 서버들은 파티션 서버들에 관련된 스토리지 객체들을 관리하는 적어도 하나의 파티셔닝 인터페이스 및 하나 이상의 애플리케이션-정의 파티셔닝 시스템 인터페이스를 갖고 있고, 파티션 서버는 할당된 파티션들에 관한 정보를 저장하는 관련 스토리지 객체를 갖고 있음 -, 애플리케이션-정의 네임스페이스에 기반한 파티션들을 포함하는 파티션 테이블 - 파티션들은 네임스페이스를 포괄하는 애플리케이션-정의 네임스페이스로부터의 키 범위에 대응하고, 파티셔닝 시스템이 파티션 테이블에 액세스 가능함 -, 파티션 서버 가상 머신들에 대한 파티션들의 할당을 관리하고 파티션 서버 가상 머신들에 대한 파티션들의 파티션 테이블 할당을 유지하는 고정 파티셔닝 시스템 인터페이스들을 포함하는 제 1 마스터 롤 인스턴스, 및 네임스페이스로부터 키 값을 포함하는 클라이언트 요청들을 수신하고 키 값에 대응하는 파티션 서버의 주소들을 반환하는 적어도 하나의 고정 파티셔닝 시스템 인터페이스를 갖고 있는 클라이언트 컴포넌트를 포함하는 시스템을, 실행 시에, 제공하는 컴퓨터 사용-가능 명령어를 실행하는 복수의 프로세서를 포함한다.

[0121] 전술한 바로부터, 본 발명은 상기 시스템 및 방법에 자명하고 고유한 다른 효과와 함께, 앞서 설명한 모든 목적 및 목표를 이루도록 조정된 것임을 알 수 있다.

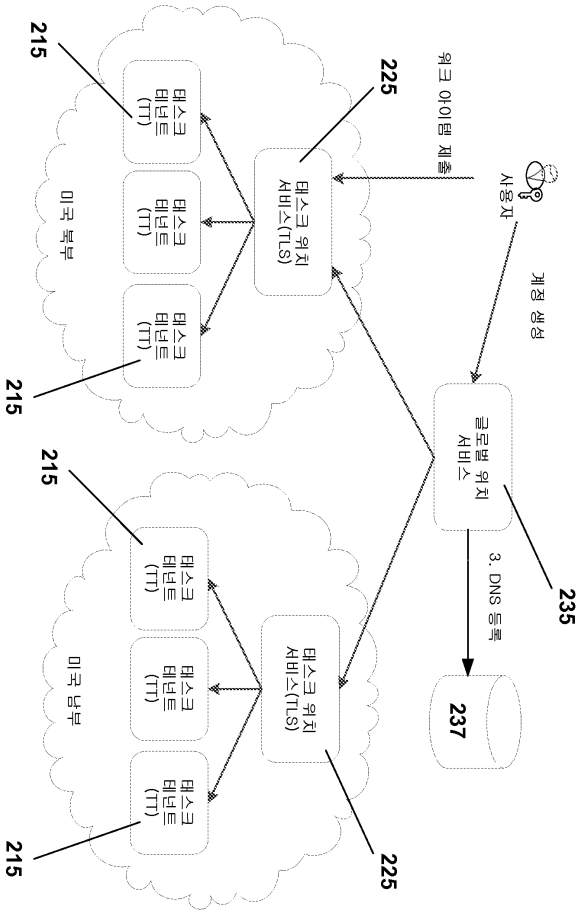
[0122] 특정 특징들 및 서브컴비네이션들이 유용하고, 이들은 다른 특징들 및 서브컴비네이션들과 관계없이 이용될 수 있음을 이해할 것이다. 이는 특허청구범위에 의해 그리고 그 범위 내에서 고려된다.

도면

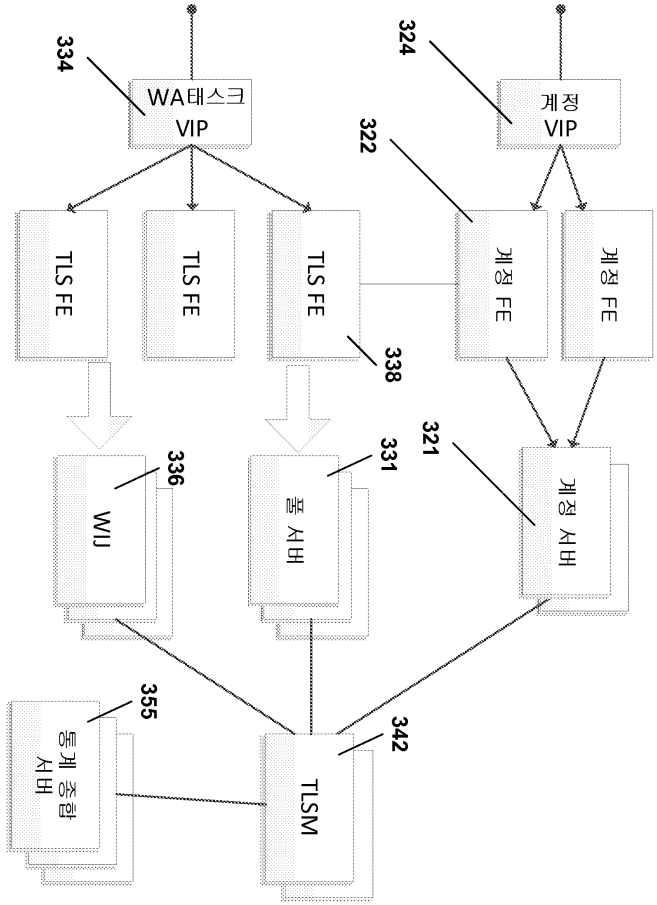
도면1



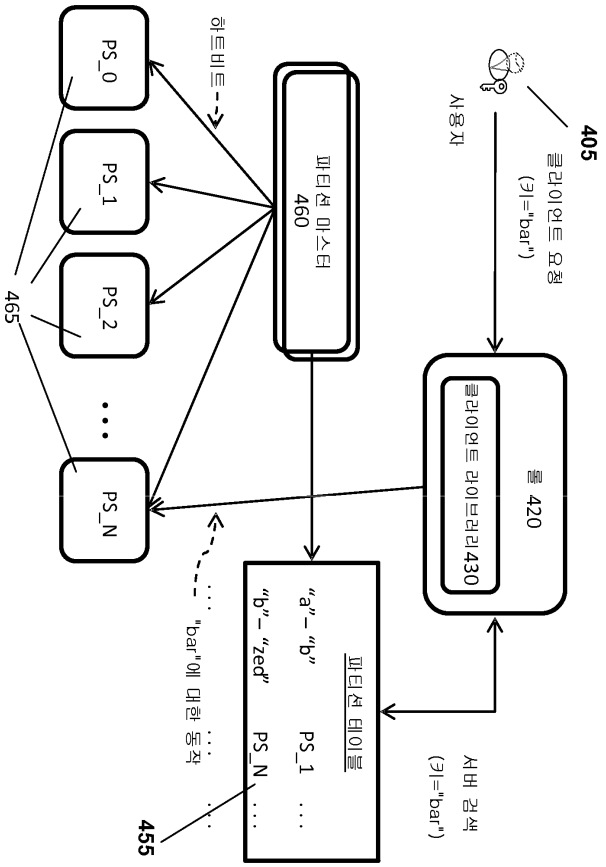
도면2


215
215
215
215
225
225
235
237
테스크 터넌트 (TT)
테스크 터넌트 (TT)
테스크 터넌트 (TT)
테스크 터넌트 (TT)
테스크 터넌트 (TT)
테스크 터넌트 (TT)
테스크 워치 서비스(TLS)
테스크 워치 서비스(TLS)
글로벌 워치 서비스
미국 북부
미국 남부
워크 아이템 제출
계정 생성
사용자
3. DNS 등록

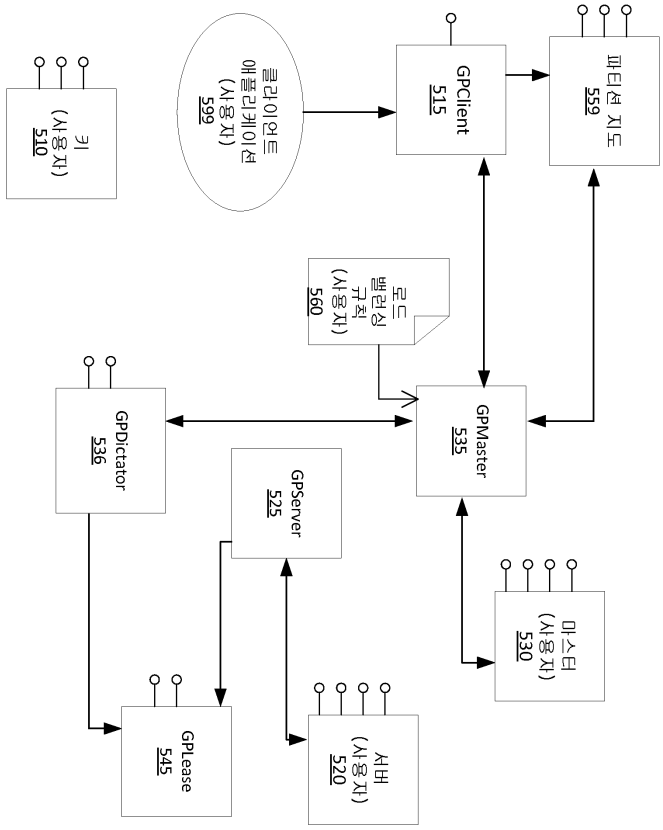
도면3


334
WA태스크
VIP
324
계정
VIP
322
TLS FE
TLS FE
TLS FE
338
계정 FE
계정 FE
321
336
WIJ
331
풀 서버
계정 서버
355
통계 종합
서버
342
TLSM

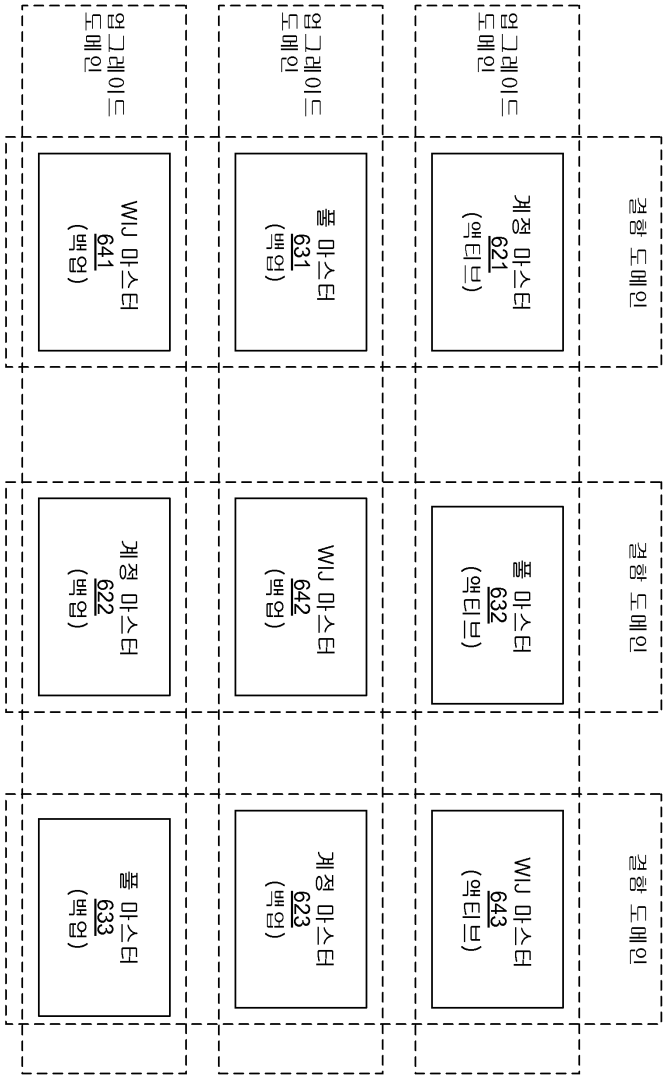
도면4


405
클라이언트 요청
(키="bar")
사용자
하트비트
PS_0
PS_1
PS_2
PS_N
465
파티션 마스터
460
클라이언트 라이브러리430
룰 420
서버 검색
(키="bar")
파티션 테이블
"a" – "b" PS_1 ...
"b" – "zed" PS_N ...
... "bar"에 대한 동작 ...
455

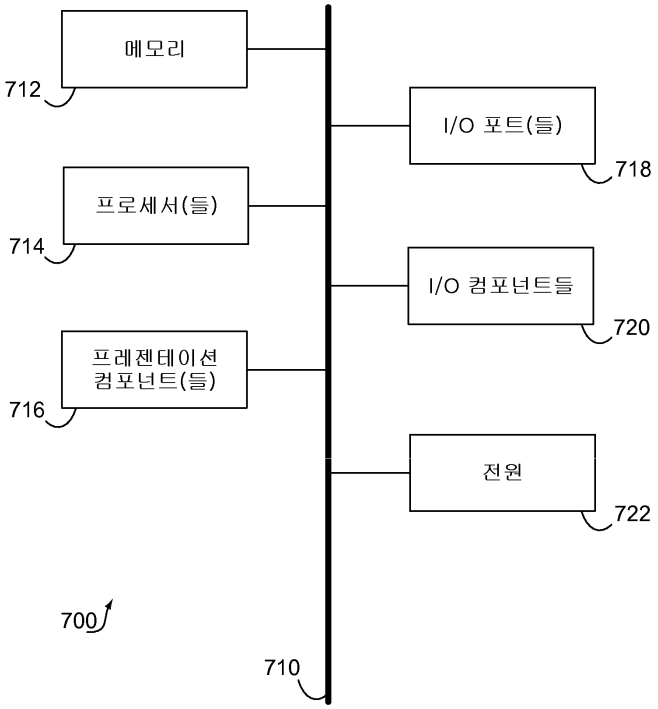
도면5


키
(사용자)
510
클라이언트
애플리케이션
(사용자)
599
GPClient
515
파티션 지도
559
로드
밸런싱
규칙
(사용자)
560
GPMaster
535
GPDictator
536
GPServer
525
마스터
(사용자)
530
GPLease
545
서버
(사용자)
520

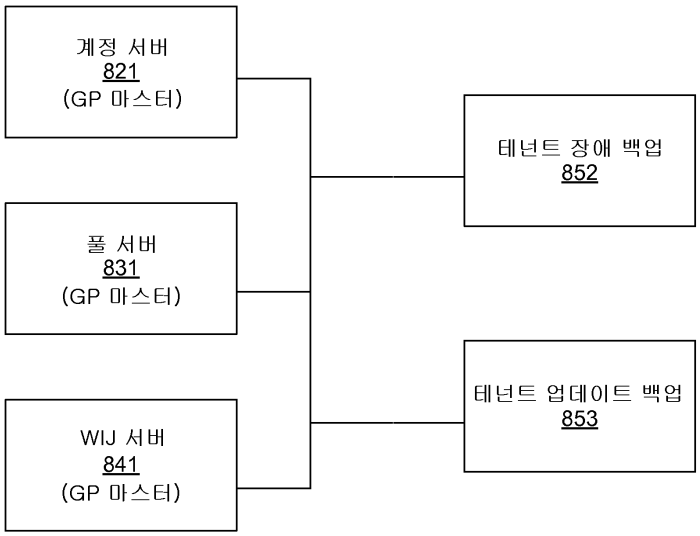
도면6


업그레이드 도메인
업그레이드 도메인
업그레이드 도메인
결합 도메인
결합 도메인
결합 도메인
계정 마스터 621 (액티브)
폴 마스터 631 (백업)
WIJ 마스터 641 (백업)
폴 마스터 632 (액티브)
WIJ 마스터 642 (백업)
계정 마스터 622 (백업)
WIJ 마스터 643 (액티브)
계정 마스터 623 (백업)
폴 마스터 633 (백업)

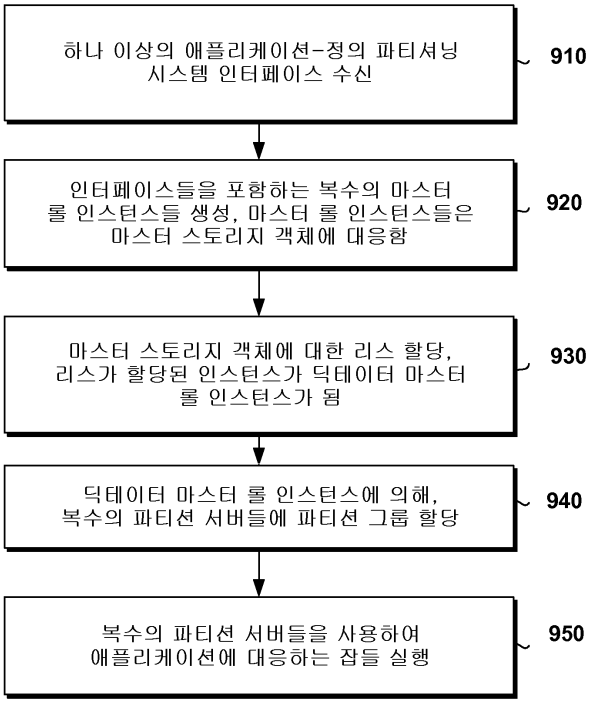
도면7


메모리
712
I/O 포트(들)
718
프로세서(들)
714
I/O 컴포넌트들
720
프레젠테이션
컴포넌트(들)
716
전원
722
700
710

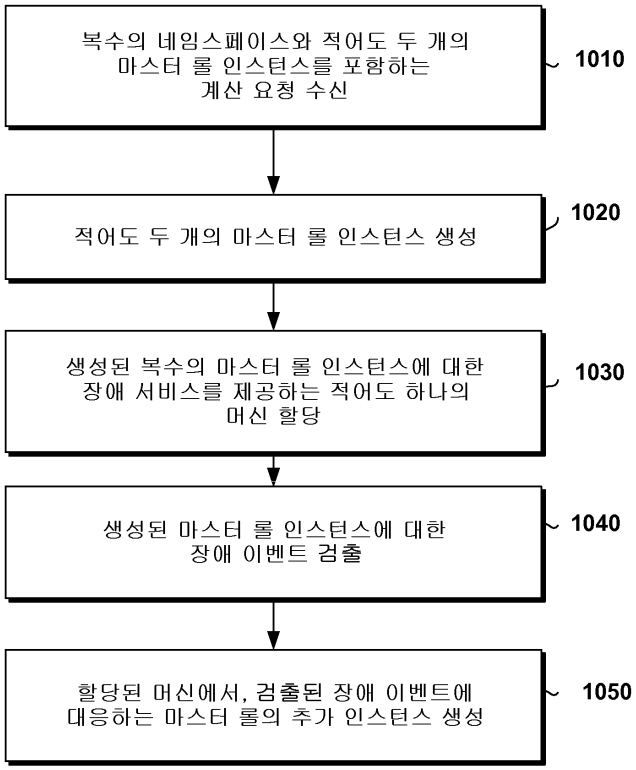
도면8


계정 서버
821
(GP 마스터)
테넌트 장애 백업
852
풀 서버
831
(GP 마스터)
테넌트 업데이트 백업
853
WIJ 서버
841
(GP 마스터)

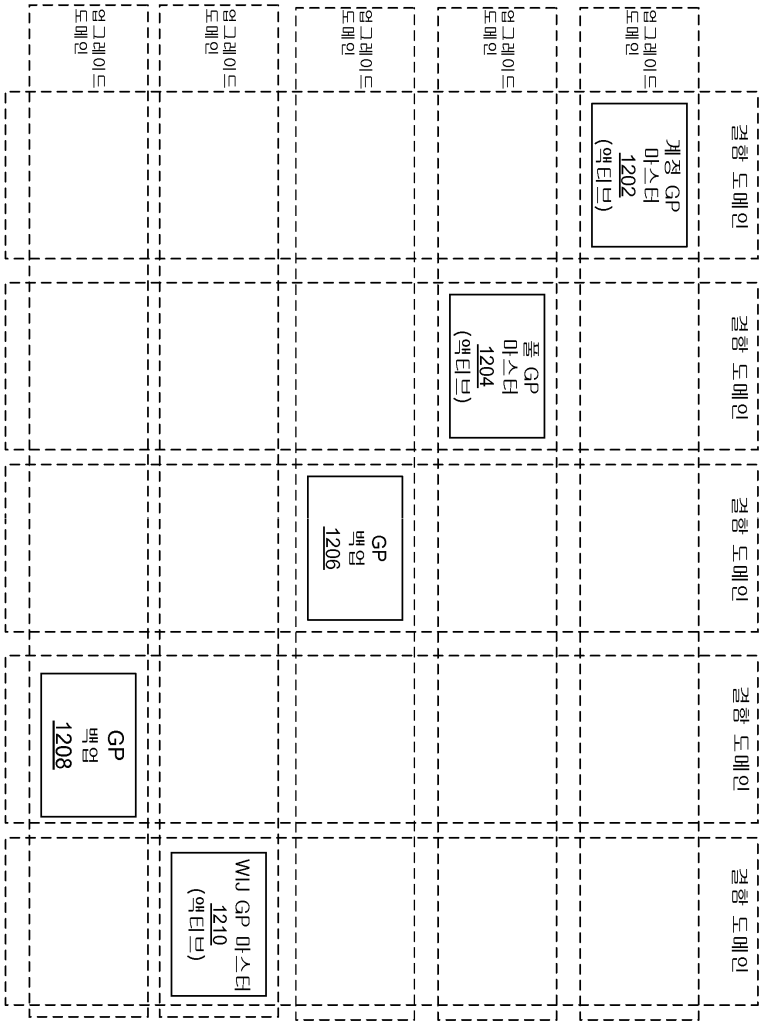
도면9


하나 이상의 애플리케이션-정의 파티셔닝 시스템 인터페이스 수신
910
인터페이스들을 포함하는 복수의 마스터 롤 인스턴스들 생성, 마스터 롤 인스턴스들은 마스터 스토리지 객체에 대응함
920
마스터 스토리지 객체에 대한 리스 할당, 리스가 할당된 인스턴스가 딕테이터 마스터 롤 인스턴스가 됨
930
딕테이터 마스터 롤 인스턴스에 의해, 복수의 파티션 서버들에 파티션 그룹 할당
940
복수의 파티션 서버들을 사용하여 애플리케이션에 대응하는 잡들 실행
950

도면10


복수의 네임스페이스와 적어도 두 개의 마스터 롤 인스턴스를 포함하는 계산 요청 수신
1010
적어도 두 개의 마스터 롤 인스턴스 생성
1020
생성된 복수의 마스터 롤 인스턴스에 대한 장애 서비스를 제공하는 적어도 하나의 머신 할당
1030
생성된 마스터 롤 인스턴스에 대한 장애 이벤트 검출
1040
할당된 머신에서, 검출된 장애 이벤트에 대응하는 마스터 롤의 추가 인스턴스 생성
1050

도면11


결합 도메인
결합 도메인
결합 도메인
결합 도메인
결합 도메인
언그레이드 도메인
언그레이드 도메인
언그레이드 도메인
언그레이드 도메인
언그레이드 도메인
계정 GP 마스터 1202 (액티브)
풀 GP 마스터 1204 (액티브)
GP 백업 1206
GP 백업 1208
WJ GP 마스터 1210 (액티브)