(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년07월15일
(11) 등록번호 10-2421672
(24) 등록일자 2022년07월12일

(51) 국제특허분류(Int. Cl.)
G06F 9/50 (2018.01) *H04L 47/00* (2022.01)
(52) CPC특허분류
G06F 9/5077 (2013.01)
H04L 47/70 (2022.05)
(21) 출원번호 10-2017-7003705
(22) 출원일자(국제) 2015년07월10일
심사청구일자 2020년07월10일
(85) 번역문제출일자 2017년02월09일
(65) 공개번호 10-2017-0032360
(43) 공개일자 2017년03월22일
(86) 국제출원번호 PCT/US2015/040048
(87) 국제공개번호 WO 2016/007922
국제공개일자 2016년01월14일
(30) 우선권주장
62/023,076 2014년07월10일 미국(US)
(뒷면에 계속)
(56) 선행기술조사문헌
US08621178 B1*
US20100185963 A1*
US20130304788 A1*
US20140075029 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
오라클 인터내셔날 코포레이션
미국, 캘리포니아 94065, 레드우드 쇼어스 엠에스 5오피7, 오라클 파크웨이 500
(72) 발명자
탸가라잔 시바쿠마르
인도 카르나타카 560075 방갈로르 지반 비마 나가르 씸마 레디 코로니 라 레지던시 플랫 207
라무 자가디쉬
인도 카르나타카 560 037 방갈로르 친나파나할리 넥스트 투 에이.이.씨.에스 레이아웃 로한 미히라 아파트먼츠 비-103
(뒷면에 계속)
(74) 대리인
박장원

전체 청구항 수 : 총 20 항 심사관 : 장재우

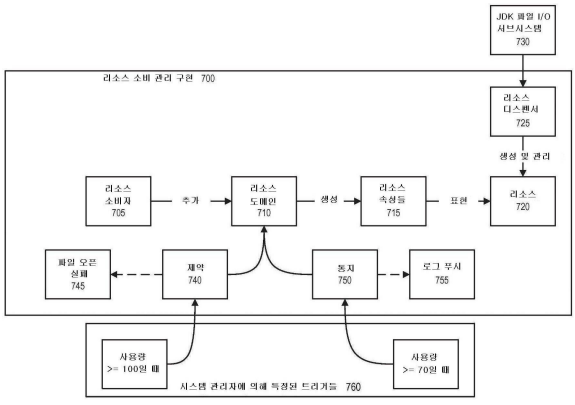
(54) 발명의 명칭 **멀티테넌트 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 시스템 및 방법**

(57) 요 약

본 발명의 실시예에 따르면, 본 명세서에서 설명되는 것은, 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 시스템 및 방법이다. 본 시스템은 하나 이상의 컴퓨터들에서, 복수의 리소스들 및 하나 이상의 파티션들을 제공할 수 있고, 여기서 하나 이상의 컴퓨터들은 이러한 하나 이상의 컴퓨터들 상에서 실행되는 애플리케이션 서

(뒷면에 계속)

대 표 도



버 환경을 포함하고, 복수의 리소스들은 애플리케이션 서버 환경 내에서 사용될 수 있는 것이고, 각각의 파티션은 임의의 도메인의 관리 및 런타임 하위분할구역을 제공한다. 본 시스템은 또한, 각각의 파티션에 의한 복수의 리소스들의 사용을 모니터링하도록 리소스 소비 관리 모듈을 구성할 수 있다. 리소스 소비 관리 모듈은, 리소스 예약들, 리소스 제약들, 및 리소스 통지들로 이루어진 그룹의 적어도 하나의 요소를 포함할 수 있다.

(52) CPC특허분류
G06F 2209/5014 (2013.01)
G06F 2209/508 (2013.01)
(30) 우선권주장
62/054,901 2014년09월24일 미국(US)
14/795,427 2015년07월09일 미국(US)

(72) 발명자
삭세나 크쉬티즈
인도 카르나타카 560 037 방갈로르 브룩필즈 에이.이.씨.에스 레이아웃 쉬리람 스푸르티 아파트먼츠 씨-008
스리바스타바 라훌
인도 카르나타카 560 029 방갈로르 2엔디 크로스 로드 치크카 아우두고디 엔오. 18 프레스티지 에스티. 존’스 우드즈 프레스티지 렉싱턴
페이겐 로렌스
미국 뉴저지 07069 와청 레드몬트 로드 104
메타 나만
인도 카르나타카 560 029 방갈로르 2엔디 크로스 로드 치크카 아우두고디 엔오. 18 프레스티지 에스티. 존’스 우드즈 프레스티지 렉싱턴
수브라마니안 프라사드
인도 카르나타카 560 029 방갈로르 2엔디 크로스 로드 치크카 아우두고디 엔오. 18 프레스티지 에스티. 존’스 우드즈 프레스티지 렉싱턴

명 세 서

청구범위

청구항 1

애플리케이션 서버 환경(application server environment)에서 리소스 격리 및 소비(resource isolation and consumption)를 위한 시스템으로서,

상기 시스템은,

하나 이상의 컴퓨터(computer)들과; 그리고

구성가능한 리소스 소비 관리 모듈(configurable resource consumption management module)을 포함하고,

상기 하나 이상의 컴퓨터들은,

소프트웨어 애플리케이션(software application)들의 배치(deployment) 및 실행(execution)을 인에이블(enable)시키는 애플리케이션 서버와, 여기서 상기 애플리케이션 서버는 상기 소프트웨어 애플리케이션들의 실행을 위한 도메인(domain)을 정의하기 위해 런타임(runtime)시에 사용되는 도메인 구성(domain configuration)과 관련되고,

상기 애플리케이션 서버 환경 내에서 사용될 수 있는 복수의 리소스들과, 그리고

복수의 파티션(partition)들을 포함하고,

상기 복수의 파티션들 각각은 파티션 구성(partition configuration)과 관련되고, 각각의 파티션은 상기 도메인의 하위분할구역(subdivision)을 제공하고, 각각의 파티션은 각각의 파티션 내에 배치되는 하나 이상의 리소스 그룹(resource group)들 및 하나 이상의 소프트웨어 애플리케이션들을 포함하고, 상기 하나 이상 리소스 그룹들은 하나 이상의 리소스 그룹 템플릿(resource group template)들에 의해 정의되고,

상기 구성가능한 리소스 소비 관리 모듈은, 리소스 예약(resource reservation)들, 리소스 제약(resource constraint)들, 및 리소스 통지(resource notification)들로 이루어진 그룹의 적어도 하나의 요소(member)를 포함하고,

상기 리소스 소비 관리 모듈은 각각의 파티션에 의한 상기 복수의 리소스들의 사용을 모니터링하도록 구성되고, 여기서 각각의 파티션에 의한 상기 복수의 리소스들의 사용은 각각의 파티션 내에 배치되는 상기 하나 이상의 소프트웨어 애플리케이션들과 관련되는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 시스템.

청구항 2

제1항에 있어서,

상기 복수의 리소스들은 공유된 리소스(shared resource)들을 포함하고,

상기 리소스 소비 관리 모듈은 또한, 상기 리소스 예약들, 상기 리소스 제약들, 및 상기 리소스 통지들로 이루어진 상기 그룹의 적어도 하나의 요소를 구성하기 위한 명령들을 수신하도록 구성되는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 시스템.

청구항 3

제2항에 있어서,

상기 리소스 제약들은, 상기 하나 이상의 파티션들 중 임의의 파티션이 상기 공유된 리소스들의 미리-정의된 양(pre-defined amount)보다 더 많이 사용할 때, 제약 동작(constraint action)을 수행하도록 구성되는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 시스템.

청구항 4

제3항에 있어서,

상기 제약 동작은 저속화(slow), 불이행(fail), 및 멈춤(shutdown)으로 이루어진 그룹으로부터 선택되는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 시스템.

청구항 5

제2항에 있어서,

상기 리소스 통지들은, 상기 하나 이상의 파티션들 중 임의의 파티션이 상기 공유된 리소스들의 미리-정의된 양보다 더 많이 사용할 때, 통지 동작(notification action)을 제공하도록 구성되는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 시스템.

청구항 6

제2항에 있어서,

상기 리소스 제약들은, 상기 하나 이상의 파티션들 중 임의의 파티션이 상기 공유된 리소스들의 공정한 배당 값(fair share value)보다 더 많이 사용하는 경우, 제약 동작을 수행하도록 구성되고,

상기 제약 동작은 둔화(slow-down)를 포함하고,

상기 둔화는 상기 하나 이상의 파티션들 중 상기 임의의 파티션의 상기 공정한 배당 값을 감소시키고,

상기 공정한 배당 값은 상기 하나 이상의 파티션들 중 상기 임의의 파티션에 할당된 작업 인스턴스(work instance)들의 개수와 관련되어 있는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 시스템.

청구항 7

제1항에 있어서,

상기 애플리케이션 서버 환경은 멀티-테넌트 애플리케이션 서버 환경(multi-tenant application server environment)을 포함하고,

상기 하나 이상의 파티션들 각각은 복수의 미리-정의된 서비스 레벨(pre-defined service level)들 중 하나와 관련되고,

상기 복수의 미리-정의된 서비스 레벨들 각각은 상기 파티션들에 의한 리소스들의 관리를 결정하기 위해서 상기 리소스 소비 관리 모듈에 의해 사용되도록 구성되는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 시스템.

청구항 8

애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 방법으로서,

상기 방법은,

하나 이상의 컴퓨터들에서, 복수의 리소스들 및 복수의 파티션들을 제공하는 단계와; 그리고

각각의 파티션에 의한 상기 복수의 리소스들의 사용을 모니터링하도록 리소스 소비 관리 모듈을 구성하는 단계를 포함하고,

상기 하나 이상의 컴퓨터들은, 소프트웨어 애플리케이션들의 배치 및 실행을 인에이블시키는 애플리케이션 서버를 포함하고, 여기서 상기 애플리케이션 서버는 상기 소프트웨어 애플리케이션들의 실행을 위한 도메인을 정의하기 위해 런타임시에 사용되는 도메인 구성과 관련되고,

상기 복수의 리소스들은 상기 애플리케이션 서버 환경 내에서 사용될 수 있는 것이며,

상기 복수의 파티션들 각각은 파티션 구성과 관련되고, 각각의 파티션은 상기 도메인의 하위분할구역을 제공하고, 각각의 파티션은 각각의 파티션 내에 배치되는 하나 이상의 리소스 그룹들 및 하나 이상의 소프트웨어 애플리케이션들을 포함하고, 상기 하나 이상 리소스 그룹들은 하나 이상의 리소스 그룹 템플릿들에 의해 정의되고,

상기 리소스 소비 관리 모듈은, 리소스 예약들, 리소스 제약들, 및 리소스 통지들로 이루어진 그룹의 적어도 하나의 요소를 포함하고,

각각의 파티션에 의한 상기 복수의 리소스들의 사용은 각각의 파티션 내에 배치되는 상기 하나 이상의 소프트웨어 애플리케이션들과 관련되는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 방법.

청구항 9

제8항에 있어서,

상기 복수의 리소스들은 공유된 리소스들을 포함하고,

상기 리소스 소비 관리 모듈은 또한, 상기 리소스 예약들, 상기 리소스 제약들, 및 상기 리소스 통지들로 이루어진 상기 그룹의 적어도 하나의 요소를 구성하기 위한 명령들을 수신하도록 구성되는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 방법.

청구항 10

제9항에 있어서,

상기 방법은 또한, 상기 하나 이상의 파티션들 중 임의의 파티션이 상기 공유된 리소스들의 미리-정의된 양보다 더 많이 사용할 때, 제약 동작을 수행하도록 상기 리소스 제약들을 구성하는 단계를 포함하는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 방법.

청구항 11

제10항에 있어서,

상기 제약 동작은 저속화, 불이행, 및 멈춤으로 이루어진 그룹으로부터 선택되는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 방법.

청구항 12

제9항에 있어서,

상기 방법은 또한, 상기 하나 이상의 파티션들 중 임의의 파티션이 상기 공유된 리소스들의 미리-정의된 양보다 더 많이 사용할 때, 통지 동작을 제공하도록 상기 리소스 통지들을 구성하는 단계를 포함하는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 방법.

청구항 13

제12항에 있어서,

상기 통지 동작은 로그 이벤트 파일(log event file)을 푸시(push)하고,

상기 로그 이벤트 파일은 시스템 관리자(system administrator)에 의해 액세스가능한 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 방법.

청구항 14

제8항에 있어서,

상기 하나 이상의 파티션들 각각은 복수의 미리-정의된 서비스 레벨들 중 하나와 관련되고,

상기 복수의 미리-정의된 서비스 레벨들 각각은 상기 파티션들에 의한 리소스들의 관리를 결정하기 위해서 상기 리소스 소비 관리 모듈에 의해 사용되도록 구성되는 것을 특징으로 하는 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 방법.

청구항 15

삭제

청구항 16

삭제

청구항 17

비-일시적 컴퓨터 판독가능 저장 매체(non-transitory computer readable storage medium)로서, 상기 비-일시적 컴퓨터 판독가능 저장 매체는 상기 비-일시적 컴퓨터 판독가능 저장 매체 상에 저장되는 명령들을 포함하고, 상기 명령들은 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 명령들이고, 상기 명령들은 하나 이상의 컴퓨터들에 의해 판독 및 실행될 때 상기 하나 이상의 컴퓨터들로 하여금 단계들을 수행하도록 하며, 상기 단계들은,

하나 이상의 컴퓨터들에서, 복수의 리소스들 및 복수의 파티션들을 제공하는 단계와; 그리고

각각의 파티션에 의한 상기 복수의 리소스들의 사용을 모니터링하도록 리소스 소비 관리 모듈을 구성하는 단계를 포함하고,

상기 하나 이상의 컴퓨터들은, 소프트웨어 애플리케이션들의 배치 및 실행을 인에이블시키는 애플리케이션 서버를 포함하고, 여기서 상기 애플리케이션 서버는 상기 소프트웨어 애플리케이션들의 실행을 위한 도메인을 정의하기 위해 런타임시에 사용되는 도메인 구성과 관련되고,

상기 복수의 리소스들은 상기 애플리케이션 서버 환경 내에서 사용될 수 있는 것이며,

상기 복수의 파티션들 각각은 파티션 구성과 관련되고, 각각의 파티션은 상기 도메인의 하위분할구역을 제공하고, 각각의 파티션은 각각의 파티션 내에 배치되는 하나 이상의 리소스 그룹들 및 하나 이상의 소프트웨어 애플리케이션들을 포함하고, 상기 하나 이상 리소스 그룹들은 하나 이상의 리소스 그룹 템플릿들에 의해 정의되고,

상기 리소스 소비 관리 모듈은, 리소스 예약들, 리소스 제약들, 및 리소스 통지들로 이루어진 그룹의 적어도 하나의 요소를 포함하고,

각각의 파티션에 의한 상기 복수의 리소스들의 사용은 각각의 파티션 내에 배치되는 상기 하나 이상의 소프트웨어 애플리케이션들과 관련되는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 저장 매체.

청구항 18

제17항에 있어서,

상기 복수의 리소스들은 공유된 리소스들을 포함하고,

상기 리소스 소비 관리 모듈은 또한, 상기 리소스 예약들, 상기 리소스 제약들, 및 상기 리소스 통지들로 이루어진 상기 그룹의 적어도 하나의 요소를 구성하기 위한 명령들을 수신하도록 구성되는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 저장 매체.

청구항 19

제18항에 있어서,

상기 단계들은, 상기 하나 이상의 파티션들 중 임의의 파티션이 상기 공유된 리소스들의 미리-정의된 양보다 더 많이 사용할 때, 제약 동작을 수행하도록 상기 리소스 제약들을 구성하는 단계를 포함하는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 저장 매체.

청구항 20

제19항에 있어서,

상기 제약 동작은 저속화, 불이행, 및 멈춤으로 이루어진 그룹으로부터 선택되는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 저장 매체.

청구항 21

제18항에 있어서,

상기 단계들은 또한, 상기 하나 이상의 파티션들 중 임의의 파티션이 상기 공유된 리소스들의 미리-정의된 양보다 더 많이 사용할 때, 통지 동작을 제공하도록 상기 리소스 통지들을 구성하는 단계를 포함하는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 저장 매체.

청구항 22

제21항에 있어서,

상기 통지 동작은 로그 이벤트 파일을 푸시하고,

상기 로그 이벤트 파일은 시스템 관리자에 의해 액세스가능한 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 저장 매체.

발명의 설명

기 술 분 야

[0001] 저작권 고지(COPYRIGHT NOTICE)

[0002] 본 특허 문서의 개시내용의 일부분은 저작권 보호를 받는 자료를 포함한다. 저작권 소유자는 특허 및 상표 관리국의 특허 파일 또는 기록에 나타나 있는 바와 같이 본 특허 문서 또는 특허 개시내용을 임의의 사람이 복제(facsimile reproduction)하는 것에 대해 반대하지 않지만, 한편으로는 어떤 경우라도 모든 저작권 권리들을 보유한다.

[0003] 본 발명의 실시예들은 일반적으로 애플리케이션 서버(application server)들 및 클라우드(cloud) 환경들에 관한 것이고, 특히 멀티테넌트 애플리케이션 서버 환경(multitenant application server environment)에서 리소스 격리 및 소비(resource isolation and consumption)를 위한 시스템 및 방법에 관한 것이다.

배 경 기 술

[0004] 소프트웨어 애플리케이션 서버들(이러한 것의 예들은 오라클 웹로직 서버(Oracle WebLogic Server, WLS) 및 글래스피시(Glassfish)를 포함함)은, 일반적으로 엔터프라이즈 소프트웨어 애플리케이션(enterprise software application)들을 실행하기 위한 관리되는 환경(managed environment)을 제공한다. 최근, 클라우드 환경들에서의 사용을 위한 기술들이 또한 개발되어 오고 있으며, 이러한 클라우드 환경들은 사용자들 혹은 테넌트(tenant)들이 클라우드 환경 내에서 자신들의 애플리케이션들을 개발 및 실행할 수 있도록 하며, 아울러 해당 환경에 의해 제공되는 배분된 리소스들을 이용할 수 있게 한다.

발명의 내용

[0005] 본 발명의 실시예에 따르면, 본 명세서에서 설명되는 것은, 애플리케이션 서버 환경(application server environment)에서 리소스 격리 및 소비(resource isolation and consumption)를 위한 시스템 및 방법이다. 본 시스템은 하나 이상의 컴퓨터(computer)들에서, 복수의 리소스들 및 하나 이상의 파티션(partition)들을 제공할 수 있고, 여기서 하나 이상의 컴퓨터들은 이러한 하나 이상의 컴퓨터들 상에서 실행되는 애플리케이션 서버 환경을 포함하고, 복수의 리소스들은 애플리케이션 서버 환경 내에서 사용될 수 있는 것이고, 각각의 파티션은 임의의 도메인(domain)의 관리 및 런타임 하위분할구역(administrative and runtime subdivision)을 제공한다. 본 시스템은 또한, 각각의 파티션에 의한 복수의 리소스들의 사용을 모니터링하도록 리소스 소비 관리 모듈(resource consumption management module)을 구성할 수 있다. 리소스 소비 관리 모듈은, 리소스 예약(resource reservation)들, 리소스 제약(resource constraint)들, 및 리소스 통지(resource notification)들로 이루어진 그룹의 적어도 하나의 요소(member)를 포함할 수 있다.

도면의 간단한 설명

[0006] 도 1은 본 발명의 일 실시예에 따른, 애플리케이션 서버, 클라우드, 혹은 다른 환경에서 멀티-테넌시(multi-tenancy)를 지원하기 위한 시스템을 예시한다.

도 2는 또한, 본 발명의 일 실시예에 따른, 애플리케이션 서버, 클라우드, 혹은 다른 환경에서 멀티-테넌시를 지원하기 위한 시스템을 예시한다.

도 3은 또한, 본 발명의 일 실시예에 따른, 애플리케이션 서버, 클라우드, 혹은 다른 환경에서 멀티-테넌시를 지원하기 위한 시스템을 예시한다.

도 4는 본 발명의 일 실시예에 따른, 예시적인 멀티-테넌트 환경(multi-tenant environment)과 함께 사용하기 위한 도메인 구성(domain configuration)을 예시한다.

도 5는 또한, 본 발명의 일 실시예에 따른, 예시적인 멀티-테넌트 환경을 예시한다.

도 6은 본 발명의 일 실시예에 따른, 애플리케이션 서버 환경에서 리소스 격리 및 소비를 예시한다.

도 7은 본 발명의 일 실시예에 따른, 리소스 소비 관리 구현을 예시한다.

도 8은 본 발명의 일 실시예에 따른, 리소스 소비 관리 통합에서 상호작용들을 보여주는 시퀀스 도면을 예시한다.

도 9는 본 발명의 일 실시예에 따른, 리소스 소비 관리 구현을 예시한다.

도 10은 본 발명의 일 실시예에 따른, 애플리케이션 서버 환경에서 리소스 격리 및 소비를 예시한다.

도 11은 본 발명의 일 실시예에 따른, 애플리케이션 서버 환경에서의 리소스 격리 및 소비를 위한 방법에 대한 흐름도를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0007] 본 발명의 실시예에 따르면, 본 명세서에서 설명되는 것은, 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 시스템 및 방법이다. 본 시스템은 하나 이상의 컴퓨터들에서, 복수의 리소스들 및 하나 이상의 파티션들을 제공할 수 있고, 여기서 하나 이상의 컴퓨터들은 이러한 하나 이상의 컴퓨터들 상에서 실행되는 애플리케이션 서버 환경을 포함하고, 복수의 리소스들은 애플리케이션 서버 환경 내에서 사용될 수 있는 것이고, 각각의 파티션은 임의의 도메인의 관리 및 런타임 하위분할구역을 제공한다. 본 시스템은 또한, 각각의 파티션에 의한 복수의 리소스들의 사용을 모니터링하도록 리소스 소비 관리 모듈을 구성할 수 있다. 리소스 소비 관리 모듈은, 리소스 예약들, 리소스 제약들, 및 리소스 통지들로 이루어진 그룹의 적어도 하나의 요소를 포함할 수 있다.

[0008] **애플리케이션 서버(Application Server)(예를 들어, 멀티-테넌트(Multi-Tenant, MT)) 환경**

[0009] 도 1은 본 발명의 일 실시예에 따른, 애플리케이션 서버, 클라우드, 혹은 다른 환경에서 멀티-테넌시(multi-tenancy)를 지원하기 위한 시스템을 예시한다.

[0010] 도 1에서 예시되는 바와 같이, 본 발명의 일 실시예에 따르면, 애플리케이션 서버(예를 들어, 멀티-테넌트, MT) 환경(100), 또는 소프트웨어 애플리케이션들의 배치(deployment) 및 실행(execution)을 인에이블(enable)시키는 다른 컴퓨팅 환경이, 애플리케이션 서버 도메인을 정의하기 위해 런타임(runtime)시에 사용되는 도메인(102) 구성을 포함하도록 그리고 이러한 도메인(102) 구성에 따라 동작하도록 구성될 수 있다.

[0011] 본 발명의 일 실시예에 따르면, 애플리케이션 서버는 런타임시에서의 사용을 위해 정의되는 하나 이상의 파티션들(104)을 포함할 수 있다. 각각의 파티션은 전역적으로 고유한 파티션 식별자(globally unique partition identifier)(ID) 및 파티션 구성과 관련될 수 있고, 그리고 또한 하나 이상의 리소스 그룹(resource group)들(124)을 포함하되, 리소스 그룹 템플릿(resource group template)에 대한 참조(reference)(126) 및/또는 파티션-특정(partition-specific) 애플리케이션들 혹은 리소스들(128)을 함께, 포함할 수 있다. 도메인-레벨(domain-level) 리소스 그룹들, 애플리케이션들 및/또는 리소스들(140)이 또한 도메인 레벨에서 정의될 수 있고, 선택에 따라서는 리소스 그룹 템플릿을 참조하여 정의될 수 있다.

[0012] 각각의 리소스 그룹 템플릿(160)은 하나 이상의 애플리케이션들(애플리케이션 A(162), 애플리케이션 B(164)), 리소스들(리소스 A(166), 리소스 B(168)), 및/또는 다른 배치가능한 애플리케이션들 혹은 리소스들(170)을 정의할 수 있고, 그리고 리소스 그룹에 의해 참조될 수 있다. 예를 들어, 도 1에서 예시되는 바와 같이, 파티션(104) 내의 리소스 그룹(124)은 리소스 그룹 템플릿(160)을 참조(190)할 수 있다.

[0013] 일반적으로, 시스템 관리자는 파티션들, 도메인-레벨 리소스 그룹들 및 리소스 그룹 템플릿들, 그리고 보안 영역(security realm)들을 정의할 수 있고, 반면, 파티션 관리자는 자기들 자신의 파티션의 애스팩트(aspect)들을 정의할 수 있는데, 예를 들어, 파티션-레벨 리소스 그룹을 생성하는 것, 애플리케이션을 파티션에 배치하는 것, 또는 파티션에 대한 특정 영역들을 참조하는 것을 수행함으로써 정의할 수 있다.

[0014] 도 2는 또한, 본 발명의 일 실시예에 따른, 애플리케이션 서버, 클라우드, 혹은 다른 환경에서 멀티-테넌시를 지원하기 위한 시스템을 예시한다.

[0015] 도 2에서 예시되는 바와 같이, 본 발명의 일 실시예에 따르면, 파티션(202)은 예를 들어, 리소스 그룹(205)(이것은 리소스 그룹 템플릿(210)에 대한 참조(206)를 포함함), 가상 타겟(virtual target)(예를 들어, 가상 호스트(virtual host)) 정보(207), 그리고 플러거블 데이터베이스(Pluggable DataBase, PDB) 정보(208)를 포함할 수 있다. 리소스 그룹 템플릿(예를 들어, 210)은 예를 들어, 복수의 애플리케이션들(애플리케이션 A(211) 및 애플리케이션 B(212))을 정의하되, 리소스들을 함께 정의할 수 있는 데, 여기서 리소스들은 예컨대, 자바 메시지 서버(Java Message Server, JMS) 서버(213), 스토어-앤-포워드(Store-And-Forward, SAF) 에이전트(agent)(215), 메일 세션 컴포넌트(mail session component)(216), 혹은 자바 데이터베이스 접속성(Java DataBase Connectivity, JDBC) 리소스(217)와 같은 것이다.

[0016] 예컨대, 도 2에서 예시되는 리소스 그룹 템플릿이 제공되는 데, 본 발명의 다른 실시예들에 따르면, 리소스 그룹 템플릿들 및 엘리먼트(element)들의 상이한 타입들이 제공될 수 있다.

[0017] 본 발명의 일 실시예에 따르면, 파티션(예를 들어, 202) 내의 리소스 그룹이 특정 리소스 그룹 템플릿(예를 들어, 210)을 참조(220)하는 경우, 파티션-특정 정보(230)(예를 들어, 파티션-특정 PDB 정보)를 표시하기 위해, 특정 파티션과 관련된 정보가 그 참조되는 리소스 그룹 템플릿과 결합하여 사용될 수 있다. 그 다음에, 파티션-특정 정보는 파티션에 의한 사용을 위해서, 리소스들(예를 들어, PDB 리소스)을 구성하도록 애플리케이션 서버에 의해 사용될 수 있다. 예를 들어, 파티션(202)과 관련된 파티션-특정 PDB 정보는, 해당하는 그 파티션에 의한 사용을 위해서, 임의의 적절한 PDB(238)를 갖는 컨테이너 데이터베이스(Container DataBase, CDB)(236)를 구성(232)하도록 애플리케이션 서버에 의해 사용될 수 있다.

[0018] 유사하게, 본 발명의 일 실시예에 따르면, 특정 파티션과 관련된 가상 타겟 정보는 파티션에 의한 사용을 위해서 파티션-특정 가상 타겟(240)을 정의(239)하는데 사용될 수 있는데, 예를 들어, baylandurgentcare.com을 정의하는데 사용될 수 있고, 그 다음에 이것은 유니폼 리소스 로케이터(Uniform Resource Locator, URL), 예를 들어, http://baylandurgentcare.com을 통해 액세스가능하게 될 수 있다.

[0019] 도 3은 또한, 본 발명의 일 실시예에 따른, 애플리케이션 서버, 클라우드, 혹은 다른 환경에서 멀티-테넌시를 지원하기 위한 시스템을 예시한다.

[0020] 본 발명의 일 실시예에 따르면, config.xml 구성 파일(configuration file)과 같은 시스템 구성이, 파티션을 정의하는데 사용되며, 시스템 구성에는 해당하는 그 파티션과 관련된 리소스 그룹들 및/또는 다른 파티션 특성들에 대한 구성 엘리먼트(configuration element)들이 포함된다. 값(value)들은 특성 명칭/값 쌍(property name/value pair)들을 사용하여 파티션-별로 특정될 수 있다.

[0021] 본 발명의 일 실시예에 따르면, 복수의 파티션들이 임의의 관리되는 서버/클러스터(managed server/cluster)(242) 내에서 실행될 수 있고, 또는 CDB(243)에 대한 액세스를 제공할 수 있는 아울러 웹 티어(web tier)(244)를 통해 액세스가능한 유사한 환경 내에서 실행될 수 있다. 이것은 예를 들어, 도메인 혹은 파티션이 (CDB의) PDB들 중 하나 이상의 PDB와 관련될 수 있게 한다.

[0022] 본 발명의 일 실시예에 따르면, 복수의 파티션들(본 예에서는 파티션 A(250) 및 파티션 B(260))은 해당하는 그 파티션과 관련된 복수의 리소스들을 포함하도록 구성될 수 있다. 예를 들어, 파티션 A는 리소스 그룹(251)을 포함하도록 구성될 수 있고, 리소스 그룹(251)은 애플리케이션 A1(252), 애플리케이션 A2(254), 및 JMS A(256)를 포함하되, PDB A(259)와 관련된 데이터소스 A(datasource A)(257)를 함께 포함하고, 여기서 파티션은 가상 타겟 A(258)를 통해 액세스가능하다. 유사하게, 파티션 B(260)는 리소스 그룹(261)을 포함하도록 구성될 수 있고, 리소스 그룹(261)은 애플리케이션 B1(262), 애플리케이션 B2(264), 및 JMS B(266)를 포함하되, PDB B(269)와 관련된 데이터소스 B(267)를 함께 포함하고, 여기서 파티션은 가상 타겟 B(268)를 통해 액세스가능하다.

[0023] 앞서의 예들 중 몇 가지 예는 CDB 및 PDB들의 사용을 예시하고 있지만, 다른 실시예들을 따르는 경우, 멀티-테넌트(multi-tenant) 혹은 비-멀티-테넌트(non-multi-tenant) 데이터베이스들의 다른 타입들이 지원될 수 있으며, 여기서 각각의 파티션에 대해 특정 구성이 제공될 수 있는바, 예를 들어, 스키마(schema)들의 사용을 통해 제공될 수 있거나 혹은 상이한 데이터베이스들의 사용을 통해 제공될 수 있다.

[0024] **리소스들(Resources)**

[0025] 본 발명의 일 실시예에 따르면, 리소스는 해당 환경의 임의의 도메인에 배치될 수 있는 시스템 리소스, 애플리

케이션, 또는 다른 리소스 혹은 객체(object)이다. 예를 들어, 본 발명의 일 실시예에 따르면, 리소스는 애플리케이션, JMS, JDBC, 자바메일(JavaMail), WLDF, 데이터 소스, 또는 다른 시스템 리소스 혹은 다른 타입의 객체(서버, 클러스터, 혹은 다른 애플리케이션 서버 타겟에 배치될 수 있는 것)일 수 있다.

[0026] **파티션들(Partitions)**

[0027] 본 발명의 일 실시예에 따르면, 파티션은 임의의 도메인의 런타임 및 관리 하위분할구역 혹은 슬라이스(slice)이며, 이것은 파티션 식별자(ID) 및 구성과 관련될 수 있고, 그리고 애플리케이션들을 포함할 수 있고, 그리고/또는 리소스 그룹들 및 리소스 그룹 템플릿들의 사용을 통해 도메인-전체 리소스(domain-wide resource)들을 참조할 수 있다.

[0028] 일반적으로, 파티션은 자기 자신의 애플리케이션들을 포함할 수 있고, 그리고 리소스 그룹 템플릿들을 통해 도메인 전체 애플리케이션들을 참조할 수 있고, 그리고 자기 자신의 구성을 가질 수 있다. 파티션화가 가능한 엔티티들(partitionable entities)은 리소스들을 포함할 수 있는데, 예를 들어, JMS, JDBC, 자바메일, WLDF 리소스들, 및 다른 컴포넌트들(예컨대, JNDI 네임스페이스(JNDI namespace), 네트워크 트랙픽(network traffic), 작업 매니저들(work managers), 및 보안 정책들 및 영역들(security policies and realms))을 포함할 수 있다. 멀티-테넌트 환경의 상황에서, 시스템은 테넌트와 관련된 파티션들의 관리 및 런타임 애스팩트들에 대한 테넌트 액세스(tenant access)를 제공하도록 구성될 수 있다.

[0029] 본 발명의 일 실시예에 따르면, 파티션 내의 각각의 리소스 그룹은 선택에 따라서는 리소스 그룹 템플릿을 참조할 수 있다. 파티션은 다수의 리소스 그룹들을 가질 수 있고, 그리고 이들 각각은 리소스 그룹 템플릿을 참조할 수 있다. 각각의 파티션은 파티션의 리소스 그룹들이 참조하는 리소스 그룹 템플릿들에서 특정되지 않은 구성 데이터에 대한 특성들을 정의할 수 있다. 이것은 파티션으로 하여금 리소스 그룹 템플릿에서 정의된 배치가능한 리소스들을 해당하는 그 파티션과의 사용을 위한 특정 값들에 결합시킨 결합체(binding)로서 동작할 수 있게 한다. 일부 경우들에, 파티션은 리소스 그룹 템플릿에 의해 특정된 구성 정보를 오버라이드(override)할 수 있다.

[0030] 본 발명의 일 실시예에 따르면, 파티션 구성, 예컨대 config.xml 구성 파일에 의해 정의된 바와 같은 파티션 구성은, 복수의 구성 엘리먼트들을 포함할 수 있는데, 예를 들어, "파티션(partition)"; "리소스 그룹(resource-group)"; "리소스-그룹-템플릿(resource-group-template)"; "jdbc-시스템-리소스-오버라이드(jdbc-system-resource-override)"; 및 "파티션-특성들(partition-properties)"을 포함할 수있으며, 여기서 "파티션(partition)"은 파티션을 정의하는 속성(attribute)들 및 차일드 엘리먼트(child element)들을 포함하고; "리소스 그룹(resource-group)"은 파티션에 배치되는 애플리케이션들 및 리소스들을 포함하고; "리소스-그룹-템플릿(resource-group-template)"은 템플릿에 의해 정의되는 애플리케이션들 및 리소스들을 포함하고; "jdbc-시스템-리소스-오버라이드(jdbc-system-resource-override)"는 데이터베이스-특정 서비스 명칭, 사용자 명칭, 및 패스워드(password)를 포함하고; 그리고 "파티션-특성들(partition-properties)"은 리소스 그룹 템플릿들에서 매크로 대체(macro replacement)를 위해 사용될 수 있는 특성 키 값(property key value)들을 포함한다.

[0031] 시동(startup)될 때, 시스템은, 리소스 그룹 템플릿으로부터 각각의 리소스에 대한 파티션-특정 구성 엘리먼트들을 발생시키기 위해서, 구성 파일에 의해 제공되는 정보를 사용할 수 있다.

[0032] **리소스 그룹들(Resource Groups)**

[0033] 본 발명의 일 실시예에 따르면, 리소스 그룹은 (도메인 레벨 혹은 파티션 레벨에서 정의될 수 있는 아울러 리소스 그룹 템플릿을 참조할 수 있는) 배치가능한 리소스들의 임의의 명명된 정규화된 집합체(named, fully-qualified collection)이다. 리소스 그룹 내의 리소스들은, 관리자가 이러한 리소스들의 개시(start) 혹은 이러한 리소스들에 대한 연결(connect)을 위해 필요한 정보(예를 들어, 데이터 소스에 연결되기 위한 증명서(credential)들, 혹은 애플리케이션에 대한 타겟팅 정보(targeting information))를 모두 제공한다는 점에서 정규화된 것(fully-qualified)으로 고려된다.

[0034] 시스템 관리자는 도메인 레벨에서 혹은 파티션 레벨에서 리소스 그룹들을 선언(declare)할 수 있다. 도메인 레벨에서, 리소스 그룹은 관련된 리소스들을 그룹화하는 편리한 방법을 제공한다. 시스템은 도메인-레벨 리소스 그룹에서 선언된 리소스들을 그룹화되지 않은 리소스들과 동일하게 관리할 수 있고, 이에 따라 리소스들은 시스템 시동(system start-up) 동안 개시될 수 있고, 그리고 시스템 멈춤(system shut-down) 동안 정지될 수 있다. 관리자는 또한 그룹 내의 리소스를 개별적으로 정지시킬 수 있거나, 개시시킬 수 있거나, 또는 제거할 수 있고, 그리고 그룹에 관해 동작함으로써 그룹 내의 모든 리소스들에 관해 암묵적으로(implicitly) 동작할 수 있다. 예를 들어, 리소스 그룹을 정지시키는 것은 그룹 내에서 이미 정지 상태가 아닌 리소스들을 모두 정지시키고; 리

소스 그룹을 개시시키는 것은 그룹 내에서 이미 개시 상태가 아닌 임의의 리소스들을 개시시키고; 그리고 리소스 그룹을 제거하는 것은 그룹 내에 포함된 모든 리소스들을 제거한다.

[0035] 파티션 레벨에서, 시스템 혹은 파티션 관리자는 파티션 내에서 임의의 보안 제한(security restriction)들의 영향을 받는 0개 혹은 1개 이상의 리소스 그룹들을 구성할 수 있다. 예를 들어, SaaS 사용의 경우에, 다양한 파티션-레벨 리소스 그룹들은 도메인-레벨 리소스 그룹 템플릿들을 참조할 수 있고; 반면 PaaS 사용의 경우에는, 리소스 그룹 템플릿들을 참조하지는 않지만, 대신에 오로지 해당하는 그 파티션 내에서만 이용가능하도록 되어 있는 애플리케이션들 및 이들의 관련 리소스들을 나타내는, 그러한 파티션-레벨 리소스 그룹들이 생성될 수 있다.

[0036] 본 발명의 일 실시예에 따르면, 리소스 그룹화는 애플리케이션들과 (이들이 도메인 내에서 개별 관리 유닛(administrative unit)으로서 사용하는) 리소스들을 함께 그룹화하기 위해 사용될 수 있다. 예를 들어, 아래에서 설명되는 의료 기록(Medical Records)(MedRec) 애플리케이션에서, 리소스 그룹화는 MedRec 애플리케이션 및 그 리소스들을 정의한다. 다수의 파티션들이 동일한 MedRec 리소스 그룹을 임의의 파티션-특정 구성 정보를 각각 사용하여 실행시킬 수 있고, 이에 따라 각각의 MedRec 인스턴스(MedRec instance)의 일부분인 애플리케이션들이 각각의 파티션에 특정되게 된다.

[0037] **리소스 그룹 템플릿들(Resource Group Templates)**

[0038] 본 발명의 일 실시예에 따르면, 리소스 그룹 템플릿은 도메인 레벨에서 정의되는 배치가능한 리소스들의 집합체이며, 이것은 리소스 그룹으로부터 참조될 수 있고, 그리고 그 리소스들을 활성화시키기 위해 요구되는 정보의 일부는 템플릿 자체의 일부분으로서 저장되지 않을 수 있고, 이에 따라 파티션 레벨 구성의 특정(specification)을 지원하게 된다. 도메인은 임의 개수의 리소스 그룹 템플릿들을 포함할 수 있고, 이러한 리소스 그룹 템플릿들 각각은 예를 들어, 하나 이상의 관련된 자바 애플리케이션들 및 (이러한 애플리케이션들이 의존하는) 리소스들을 포함할 수 있다. 이러한 리소스들에 대한 정보의 일부는 모든 파티션들에 걸쳐 동일할 수 있고, 반면 다른 정보는 파티션마다 달라질 수 있다. 모든 구성이 도메인 레벨에서 특정될 필요는 없고, 대신에 파티션 레벨 구성은 매크로들, 혹은 특성 명칭/값 쌍들의 사용을 통해 리소스 그룹 템플릿 내에서 특정될 수 있다.

[0039] 본 발명의 일 실시예에 따르면, 특정 리소스 그룹 템플릿은 하나 이상의 리소스 그룹들에 의해 참조될 수 있다. 일반적으로, 임의의 주어진 파티션 내에서, 리소스 그룹 템플릿은 단지 한번에 하나의 리소스 그룹에 의해 참조될 수 있는데, 즉, 리소스 그룹 템플릿은 동일한 파티션 내에서 다수의 리소스 그룹들에 의해 동시에 참조될 수 없으며, 하지만 리소스 그룹 템플릿은 상이한 파티션 내의 다른 리소스 그룹에 의해서는 동시에 참조될 수 있다. 리소스 그룹을 포함하는 객체, 예를 들어, 도메인 혹은 파티션은, 리소스 그룹 템플릿 내에서 임의의 토큰(token)들의 값을 설정하기 위해, 특성 명칭/값 할당(property name/value assignment)들을 사용할 수 있다. 시스템이 임의의 (참조를 행하는) 리소스 그룹을 사용하여 리소스 그룹 템플릿을 활성화시키는 경우, 시스템은 리소스 그룹이 포함하는 객체 내에서 설정된 값들로 이러한 토크들을 대체할 수 있다. 일부 경우들에서, 시스템은 또한, 각각의 파티션/템플릿 결합에 대한 런타임 구성을 발생시키기 위해서, 정적으로-구성되는 리소스 그룹 템플릿들 및 파티션들을 사용할 수 있다.

[0040] 예를 들어, SaaS 사용의 경우에, 시스템은 동일한 애플리케이션들 및 리소스들을 여러 번 활성화시킬 수 있다 (여기에는 이들을 사용할 각각의 파티션에 대한 한 번의 활성화가 포함됨). 관리자가 리소스 그룹 템플릿을 정의할 때, 이들은 다른 곳에서 공급될 정보를 나타내기 위해 토콘들을 사용할 수 있다. 예를 들어, CRM-관련 데이터 리소스에 대한 연결시 사용하기 위한 사용자명칭은 리소스 그룹 템플릿 내에서 \${CRMDataUsername}으로서 표시될 수 있다.

[0041] **테넌트들(Tenants)**

[0042] 본 발명의 일 실시예에 따르면, 멀티-테넌트(MT) 애플리케이션 서버 환경과 같은 멀티-테넌트 환경에서, 테넌트는 하나 이상의 파티션들 및/또는 하나 이상의 테넌트-인식 애플리케이션(tenant-aware application)들에 의해 표현될 수 있는 엔티티이거나, 혹은 그렇지 않으면, 하나 이상의 파티션들 및/또는 하나 이상의 테넌트-인식 애플리케이션들과 관련될 수 있는 엔티티이다.

[0043] 예를 들어, 테넌트들은 개별 사용자 조직체들을 나타낼 수 있는데, 예컨대, 상이한 외부 회사들, 혹은 특정 기업체 내의 상이한 부서들(예를 들어, HR 및 재무 부서들)을 나타낼 수 있는바, 이들 각각은 상이한 파티션과 관련될 수 있다. 테넌트의 전역적으로 고유한 식별자(테넌트 ID)는 특정 사용자를 특정 순간에 특정 테넌트와 관련시킨다. 시스템은 특정 사용자가 어떤 테넌트에 속해 있는지를 사용자 아이덴티티(user identity)로부터 도출

할 수 있는바, 예를 들어, 사용자 아이덴티티 저장소(user identity store)를 참조함으로써 도출할 수 있다. 사용자 아이덴티티는 시스템으로 하여금 사용자가 수행할 권한을 받은 그러한 동작들을 실시(enforce)하게 할 수 있다(여기에는 사용자가 속할 수 있는 테넌트가 어떤 테넌트인지가 포함되지만, 이러한 것으로만 한정되는 것은 아님).

[0044] 본 발명의 일 실시예에 따르면, 시스템은 상이한 테넌트들의 관리 및 런타임을 서로 격리시킬 수 있다. 예를 들어, 테넌트들은 자신들의 애플리케이션들의 일부 행태(behavior)들, 그리고 이들이 액세스하게 되는 리소스들을 구성할 수 있다. 시스템은, 특정 테넌트가 또 하나의 다른 테넌트에 속하는 아티팩트(artifact)들을 관리할 수 없도록 보장할 수 있고; 그리고 런타임시, 특정 테넌트를 위해서 작업을 행하는 애플리케이션들이 다른 테넌트들과 관련된 리소스들을 참조함이 없이 오로지 해당하는 그 테넌트와 관련된 리소스들만을 참조하도록 보장할 수 있다.

[0045] 본 발명의 일 실시예에 따르면, 테넌트-비인식 애플리케이션(tenant-unaware application)은 테넌트들을 명시적으로(explicitly) 처리하는 어떠한 로직(logic)도 포함하지 않는 애플리케이션이고, 이에 따라 이러한 애플리케이션이 사용하는 임의의 리소스들은 애플리케이션이 응답하게 되는 요청을 어떤 사용자가 제출했는지에 상관없이 액세스가능할 수 있다. 이와는 대조적으로, 테넌트-인식 애플리케이션은 테넌트들을 명시적으로 처리하는 로직을 포함한다. 예를 들어, 사용자의 아이덴티티에 근거하여 애플리케이션은 사용자가 속하는 테넌트를 도출할 수 있고 그리고 테넌트-특정 리소스들에 액세스하기 위해 이러한 정보를 사용할 수 있다.

[0046] 본 발명의 일 실시예에 따르면, 시스템은 사용자들로 하여금 (테넌트-인식이 가능하도록 명시적으로 기입된) 애플리케이션들을 배치할 수 있게 하고, 이에 따라 애플리케이션 개발자들은 현행 테넌트의 테넌트 ID를 획득할 수 있게 된다. 그 다음에, 테넌트-인식 애플리케이션은 애플리케이션의 단일 인스턴스를 사용하고 있는 다수의 테넌트들을 핸들링(handling)하기 위해 테넌트 ID를 사용할 수 있다.

[0047] 예를 들어, (한 명의 의사의 사무실 혹은 병원을 지원하는) MedRec 애플리케이션은 두 개의 상이한 파티션들 혹은 테넌트들에 노출될 수 있는데, 예를 들어, 베이랜드 응급 진료(Bayland Urgent Care) 테넌트, 및 밸리 헬스(Valley Health) 테넌트에 노출될 수 있으며, 이들 각각은 기반이 되는 애플리케이션 코드(underlying application code)를 변경함이 없이 개별 테넌트-특정 리소스들(예컨대, 개별 PDB들)에 액세스할 수 있다.

[0048] **예시적 도메인 구성 및 멀티-테넌트 환경(Exemplary Domain Configuration and Multi-Tenant Environment)**

[0049] 본 발명의 일 실시예에 따르면, 애플리케이션들은 도메인 레벨에서 리소스 그룹 템플릿에 배치될 수 있거나, 또는 임의의 파티션에 국한된 범위를 갖는 리소스 그룹에 배치될 수 있거나 혹은 해당 도메인에 국한된 범위를 갖는 리소스 그룹에 배치될 수 있다. 애플리케이션 구성은 애플리케이션-별로 특정된 배치 계획(deployment plan)들 혹은 파티션-별로 특정된 배치 계획들을 사용하여 오버라이드될 수 있다. 배치 계획들은 또한 리소스 그룹의 일부분으로서 특정될 수 있다.

[0050] 도 4는 본 발명의 일 실시예에 따른, 예시적인 멀티-테넌트 환경과 함께 사용하기 위한 도메인 구성을 예시한다.

[0051] 본 발명의 일 실시예에 따르면, 시스템이 파티션을 개시시킬 때, 시스템은 제공된 구성에 따라 각각의 데이터베이스 인스턴스들에 대한 가상 타겟들(예를 들어, 가상 호스트들) 및 연결 풀(connection pool)들을 생성한다(여기에는 각각의 파티션에 대한 하나의 가상 타겟(예를 들어, 가상 호스트) 및 연결 풀이 포함됨).

[0052] 전형적으로, 각각의 리소스 그룹 템플릿은 하나 이상의 관련된 애플리케이션들, 그리고 이러한 애플리케이션들이 의존하는 리소스들을 포함할 수 있다. 각각의 파티션은 자신이 참조하고 있는 리소스 그룹 템플릿들에서 특정되지 않은 구성 데이터를 제공할 수 있는데, 이러한 제공은 리소스 그룹 템플릿 내의 배치가능한 리소스들을 파티션과 관련된 특정 값들에 결합시킨 결합체를 제공함으로써 이루어지는바, 일부 경우들에서, 여기에는 리소스 그룹 템플릿에 의해 특정된 특정 구성 정보를 오버라이드하는 것이 포함된다. 이것은 시스템으로 하여금 리소스 그룹 템플릿에 의해 표현된 애플리케이션을 각각의 파티션이 정의한 특성 값들을 사용하여 각각의 파티션에 대해 서로 다르게 활성화시킬 수 있게 한다.

[0053] 일부 인스턴스들에서, 파티션은 리소스 그룹 템플릿을 참조하지 않는 리소스 그룹들을 포함할 수 있거나, 혹은 자기 자신들의 파티션에 국한된 범위를 갖는 배치가능한 리소스들을 직접적으로 정의하는 리소스 그룹들을 포함할 수 있다. 파티션 내에서 정의된 애플리케이션들 및 데이터 소스들은 일반적으로 해당하는 그 파티션에 대해서만 이용가능하다. 리소스들은 partition:<partitionName>/<resource JNDI name>, 또는 domain:<resource

JNDI name>을 사용하여 이들이 파티션들에 걸쳐 파티션들로부터 액세스될 수 있도록 배치될 수 있다.

[0054] 예를 들어, MedRec 애플리케이션은 복수의 자바 애플리케이션들, 데이터 소스, JMS 서버, 및 메일 세션을 포함할 수 있다. 다수의 테넌트들에 대해 MedRec 애플리케이션을 실행시키기 위해, 시스템 관리자는 단일의 MedRec 리소스 그룹 템플릿(286)을 정의할 수 있고, 템플릿 내에서 이러한 배치가능한 리소스들을 선언할 수 있다.

[0055] 도메인-레벨 배치가능한 리소스들과는 대조적으로, 리소스 그룹 템플릿 내에서 선언된 배치가능한 리소스들은 템플릿 내에서 완벽하게 구성되지 않을 수 있거나, 혹은 있는-그대로(as-is) 활성화될 수 없는데, 왜냐하면, 이들은 일부 구성 정보를 갖고 있지 않기 때문이다.

[0056] 예를 들어, MedRec 리소스 그룹 템플릿은 애플리케이션들에 의해 사용되는 데이터 소스를 선언할 수 있지만, 데이터베이스에 연결되기 위한 URL을 특정할 수 없다. 상이한 테넌트들과 관련된 파티션들(예를 들어, 파티션 BUC-A(290)(베이랜드 응급 진료(Bayland Urgent Care), BUC) 및 파티션 VH-A(292)(밸리 헬스(Valley Health), VH))가 하나 이상의 리소스 그룹 템플릿들을 참조할 수 있는데, 이러한 참조는 파티션들 각각이 (MedRec 리소스 그룹 템플릿을 참조(296, 297)하는) MedRec 리소스 그룹(293, 294)을 포함함으로써 이루어진다. 그 다음에, 이러한 참조는 각각의 테넌트에 대한 가상 타겟들/가상 호스트들을 생성(302, 306)하는데 사용될 수 있는바, 이러한 가상 타겟들/가상 호스트들에는 베이랜드 응급 진료 테넌트에 의한 사용을 위해서 BUC-A 파티션과 관련되는 가상 호스트 baylandurgentcare.com(304); 그리고 밸리 헬스 테넌트에 의한 사용을 위해서 VH-A 파티션과 관련되는 가상 호스트 valleyhealth.com (308)이 포함된다.

[0057] 도 5는 또한, 본 발명의 일 실시예에 따른, 예시적인 멀티-테넌트 환경을 예시한다. 도 5에서 예시되는 바와 같이, 그리고 본 발명의 일 실시예에 따른, 두 개의 파티션들이 MedRec 리소스 그룹 템플릿을 참조하는 앞선 예로부터 예시를 계속하면, 복수의 테넌트 환경들(본 예에서는, 베이랜드 응급 진료 의사 테넌트 환경(320) 및 밸리 헬스 의사 테넌트 환경(330))을 지원하기 위해 서블릿 엔진(servlet engine)(310)이 사용될 수 있다.

[0058] 본 발명의 일 실시예에 따르면, 각각의 파티션(321, 331)은, 해당하는 테넌트 환경에 대한 들어오는 트래픽(incoming traffic)을 수용하게 되는 상이한 가상 타겟을 정의할 수 있고, 아울러 파티션에 연결되기 위한 그리고 그 리소스들(324, 334)에 연결되기 위한 상이한 URL(322, 332)을 정의할 수 있는바, 여기서 리소스들(324, 334)은 본 예에서 베이랜드 응급 진료 데이터베이스 혹은 밸리 헬스 데이터베이스를 각각 포함한다. 데이터베이스 인스턴스들은 호환가능한 스키마(compatible schema)들을 사용할 수 있는데, 왜냐하면 동일한 애플리케이션 코드가 양쪽 데이터베이스들에 대해 실행될 것이기 때문이다. 시스템이 파티션들을 개시시킬 때, 시스템은 각각의 데이터베이스 인스턴스들에 대한 가상 타겟들 및 연결 풀들을 생성할 수 있다.

[0059] **리소스 격리 및 소비(Resource Isolation and Consumption)**

[0060] 본 발명의 일 실시예에 따르면, 다수의 테넌트들이 단일 도메인을 공유할 수 있게 함으로써 소비자의 컴퓨팅 인프라구조(computing infrastructure)의 향상된 밀도(density) 및 증진된 이용(utilization)이 달성된다. 그러나, 복수의 테넌트들 혹은 파티션들이 동일한 도메인 내에 상주하는 경우, 도메인 런타임에서 리소스들에 대한 다양한 파티션들의 액세스 및 사용을 (이러한 리소스들을 할당할 때의 공정성(fairness)을 증진시키기 위해서, 그리고 공유된 리소스들에 대한 액세스의 경합(contention) 및/또는 간섭(interference)을 막기 위해서, 그리고 다수의 파티션들 및/또는 관련된 테넌트들에 대한 일관된 성능을 제공하기 위해서) 결정, 관리, 격리 및 모니터링하는 것이 필요할 수 있다.

[0061] 본 발명의 일 실시예에 따르면, 본 시스템은 자바 개발 키트(Java Development Kit)(예를 들어, JDK 8u40)에 의해 제공되는 리소스 관리 지원을 (시스템 관리자들로 하여금 JDK에 의해 관리되는 리소스들에 관해 리소스 소비 관리 정책들을 특정할 수 있도록 하기 위해서(예를 들어, 제약들, 의지 동작(recourse action)들, 및 통지들을 특정할 수 있도록 하기 위해서)) 이용한다. 이러한 리소스들은 예를 들어, CPU, 힙(Heap), 파일(File), 및 네트워크(Network)를 포함할 수 있다. 본 명세서에서 설명되는 방법들 및 시스템들은, 경량(lightweight)의, 그리고 표준(standards)에-기반을 둔, 그리고 포괄적인, 그리고 확장가능한 리소스 소비 관리(Resource Consumption Management, RCM) 프레임워크(framework)를 전달할 수 있는데, 이러한 리소스 소비 관리(RCM) 프레임워크는 공유된 리소스들의 소비를 관리하기 위해 애플리케이션 서버 환경 내에서 컨테이너들 및 컴포넌트들에 의해 사용될 수 있는 것이다.

[0062] 본 발명의 일 실시예에 따르면, 애플리케이션들이 애플리케이션 서버 환경 내에서 상이한 파티션들 및/또는 상이한 테넌트들에 의해 배치되는 경우, 공유된 리소스들(예를 들어, CPU, 네트워크, 및 저장소와 같은 낮은 레벨의 리소스들, 또는 데이터소스들, JMS 연결들, 로깅(logging), 등과 같은 더 높은 레벨의 리소스들)을 애플리케

이션들이 사용하려고 할 때 일어날 수 있는 일반적으로 두 가지 문제들이 존재한다. 이러한 두 개의 문제들은, 할당 동안의 경합/불공정, 그리고 가변적인 성능, 그리고 잠재적인 서비스 레벨 협약(Service Level Agreement, SLA) 위반들을 포함한다. 할당 동안의 경합 및 불공정은 공유된 리소스에 대한 다수의 요청들이 존재하는 경우 발생하는 결과일 수 있고, 이것은 결과적으로 경합 및 간섭이 일어나게 한다. 비정상적인 리소스 소비 요청들은 또한, 비치명적인 원인들(예를 들어, 높은-트래픽, DDoS 공격들), 오작동(misbehaving)하는 또는 버그(bug)가 있는 애플리케이션들, 악성 코드(malicious code) 등으로 인해 일어날 수 있다. 이와 같은 비정상적인 요청들은 결과적으로, 공유된 리소스의 용량(capacity)의 오버로딩(overloading)이 일어나게 할 수 있고, 따라서 해당 리소스에 대한 다른 애플리케이션의 액세스를 막을 수 있다. 가변적 성능은 잠재적인 SLA 위반들로 이어질 수 있다. 경합 및 불공정은 결과적으로 상이한 애플리케이션들/소비자들에 대한 성능이 달라지게 할 수 있고, 그리고 결과적으로 SLA 위반들이 일어나게 할 수 있다.

[0063] 본 명세서에서 설명되는 다양한 시스템들 및 방법들이 비록 리소스 격리 및 소비를 지원하는 것과 관련된 문제들을 논의할 때 멀티-파티션/멀티-테넌트 애플리케이션 서버 환경을 사용하고 있지만, 이러한 시스템들 및 방법들은, 도메인 내에서 다수의 함께 상주하는 애플리케이션들(이것은 도메인 내의 파티션들 내에 있는 애플리케이션들과는 대조적인 것임)이, 공유된 리소스들에 대해 서로 경쟁하며 액세스함으로써 결과적으로 경합 및 예측불가능한 성능이 일어나게 하는 그러한 종래의 배치들에도 적용가능하다.

[0064] 본 발명의 일 실시예에 따르면, 시스템 관리자와 같은 관리자는 공유된 리소스들(예를 들어, CPU, 힙, 파일 및 네트워크와 같은 JDK 리소스들)에 대한 액세스를 모니터링하는 메커니즘(mechanism)을 제공받고, 그리고 소비자들에 의한 이러한 리소스들의 소비에 관한 정책들을 실시한다. 예를 들어, 멀티-파티션 시스템에서, 시스템 관리자는 소비자가 임의의 공유된 리소스(들)를 과도하게-소비(over-consume)하여 다른 소비자로 하여금 그 공유된 리소스에 대한 액세스를 하지 못하게 되는 일이 일어나지 않도록 시스템을 구성할 수 있다.

[0065] 본 발명의 일 실시예에 따르면, 시스템 관리자와 같은 관리자는 상이한 소비자들/고객들에게 계층화된/차별화된 서비스 레벨들을 제공할 수 있다. 추가적으로, 관리자는, 선택에 따라서는, 사안-별 정책들(business-specific policies)을 구성할 수 있고(예를 들어, 파티션 내에서 실행되게 되는 특정된 혹은 예측된 작업부하(workload)들을 수용하기 위해 하루-중-시간(time-of-day)에 근거하여 특정 파티션에 대해 상이한 정책을 적용하는 것), 그리고 해당 정책들이 임의의 공유된 리소스의 소비 시점에 실시되게 할 수 있다.

[0066] 본 발명의 일 실시예에 따르면, 관리자는 적응성 있는(flexible) 그리고 확장가능한(extensible) 그리고 범용(general purpose)의 리소스 소비 프레임워크를 제공받으며, 이러한 리소스 소비 프레임워크는 애플리케이션 서버 환경의 컨테이너들 및 컴포넌트들이 상이한 소비자들에 의해 액세스되는 그리고/또는 이용되는 공유된 리소스들을 관리하는 것을 보조하는데 사용될 수 있다. 추가적으로, 이러한 프레임워크는 RCM 프레임워크의 상이한 사용자들에 대해 상이한 레벨들에서, 세밀화된 정책들(fine-grained policies)의 특정 및 실시를 가능하게 할 수 있다.

[0067] 본 발명의 일 실시예에 따르면, 리소스 격리 및 소비를 위한 방법들 및 시스템들을 기술하거나 혹은 설명하는 경우 다음과 같은 정의들이 사용될 수 있다.

[0068] 본 발명의 일 실시예에 따르면, 그리고 리소스 소비 관리의 상황에서, 용어 "리소스(resource)"는 애플리케이션 서버 환경과 같은 시스템 내의 엔티티를 의미하는바, 이것은 소비자들, 파티션들 및/또는 테넌트들 간에 공유되는 것을 나타내고, 그리고 한정된 양으로 존재하는 것을 나타내며, 이들이 부족하면 결과적으로 리소스 소비자에 대한 성능 변화가 일어날 수 있다.

[0069] 본 발명의 일 실시예에 따르면, 용어 "리소스 소비자(resource consumer)"는 실행가능한 엔티티로서 이러한 엔티티의 리소스 사용이 리소스 소비 관리(Resource Consumption Management)(RCM) API에 의해 제어되는 그러한 실행가능 엔티티일 수 있다. 예를 들어, 멀티-테넌트 애플리케이션 서버 환경 내의 리소스 소비자는 파티션을 포함할 수 있다.

[0070] 본 발명의 일 실시예에 따르면, 리소스 도메인은 리소스 소비자들의 그룹을 임의의 리소스에 결합시킬 수 있고, 이러한 리소스에 대해서 리소스 소비자 그룹에 공통 리소스 관리 정책을 적용한다.

[0071] 본 발명의 일 실시예에 따르면, 리소스 관리 정책은 예약들, 제약들 및 통지들을 정의할 수 있고, 그리고 리소스 소비자에 의해 소비되는 리소스들의 개수를 제한할 수 있거나, 혹은 소비의 속도를 떨어뜨릴 수 있다.

[0072] 본 발명의 일 실시예에 따르면, 제약은 임의의 리소스에 대한 리소스 소비 요청이 발생할 때 호출되게 되는 콜백(callback)일 수 있다. 콜백은 요청에 의해 소비될 수 있는 리소스 유닛들의 개수를 결정하는 해당하는 그 리

소스에 대한 정책을 구현할 수 있다.

[0073] 본 발명의 일 실시예에 따르면, 통지는 리소스 소비 요청이 핸들링된 직후 호출되는 콜백일 수 있다. 통지는 다음과 같은 리소스 소비, 예를 들어, 어카운팅(accounting), 로깅(logging) 등을 수행하기 위한 동작들을 특정하는 리소스 관리 정책을 구현하는데 사용될 수 있다.

[0074] 본 발명의 일 실시예에 따르면, 리소스 속성은 리소스의 특성들 혹은 속성들을 설명한다. 추가적으로, 리소스 디스펜서(resource dispenser)는 리소스를 생성하는 코드(code)/컨테이너(container)/로직(logic)을 의미한다. 더욱이, 무엇인가를 RM-인에이블(RM-enable)시키는 것은 RCM 프레임워크가 해당하는 그 리소스의 소비를 관리할 수 있도록 리소스 디스펜서를 수정하는 동작을 의미할 수 있다.

[0075] 본 발명의 일 실시예에 따르면, 의지 동작은 리소스 소비자에 의한 리소스 소비가 특정 상태에 도달할 때 수행되는 동작일 수 있다. 의지 동작들은 통지들일 수 있거나, 혹은 리소스의 과사용을 막기 위한 예방적 동작(precautionary action)을 행할 수 있거나, 혹은 과사용을 핸들링하려고 시도할 수 있다. 의지 동작은 리소스 소비 요청이 발생된 쓰레드(thread)와는 다른 쓰레드에서 수행될 수 있다.

[0076] 도 6은 본 발명의 일 실시예에 따른, 애플리케이션 서버 환경에서 리소스 격리 및 소비를 예시한다. 도 6은 도메인(610)을 포함하는 애플리케이션 서버 환경(600)을 도시한다. 시스템 관리자(620)는 도메인 내에서 리소스 관리 정책(630)을 구성할 수 있다. 리소스 관리 정책(630)은 적어도 리소스 예약들(631), 리소스 제약들(632), 및 리소스 통지들(633)을 포함하도록 구성될 수 있다. 도메인은 추가적으로 파티션 A(640) 및 파티션 B(650)를 각각 포함한다. 파티션 A 및 파티션 B는 리소스 그룹(642) 및 리소스 그룹(652)을 각각 포함하는데, 이러한 리소스 그룹들은 가상 타겟 A(645) 및 가상 타겟 B(655)와 각각 관련된다. 도 6에서 제시되는 바와 같이, 파티션 A 혹은 파티션 B는 또한 공유된 리소스들(660)에 액세스할 수 있는데, 예를 들어, 파티션들이 임의의 공유된 리소스의 사용을 요구하는 애플리케이션를 실행할 때 이러한 공유된 리소스들(660)에 액세스할 수 있다. 공유된 리소스들(660)은 CPU, 힙, 네트워크 및 파일과 같은 JDK 리소스들을 포함할 수 있지만, 이러한 것으로만 한정되는 것은 아니다.

[0077] 본 발명의 일 실시예에 따르면, 리소스 관리 정책은, 리소스 관리 정책에 의해 설정되는 구성된 조건들이 그 공유된 리소스들에 관해 파티션 A 혹은 파티션 B에 의해 충족될 때 동작들 혹은 태스크(task)들을 수행하도록, 시스템 관리자 또는 적절한 허가(permission)를 갖는 또 하나의 다른 것에 의해 구성될 수 있다. 이러한 동작들 혹은 태스크들은 제약들 혹은 통지들을 포함할 수 있지만, 이러한 것으로만 한정되는 것은 아니다. 제약들의 예들은 파티션에 의한 공유된 리소스들의 사용을 느리게 하는 저속화 혹은 정지시키는 것을 포함한다. 통지들의 예들은 로깅, 또는 관리자에게 통지들을 제공하는 것을 포함하지만, 이러한 것으로만 한정되는 것은 아니다.

[0078] 본 발명의 일 실시예에 따르면, 도 6에 도시된 애플리케이션 서버 환경은 오라클 웹로직 서버(Oracle WebLogic server)일 수 있다. 웹로직 서버는 클래스로더-기반의 격리(ClassLoader-based isolation)(애플리케이션들의 공유된 라이브러리들(shared libraries)에 속하지 않는 클래스들은 서로 격리됨) 및 웹로직 작업 매니저(WebLogic Work Manager) 특징들을 포함할 수 있는데, 이것은 관리자로 하여금 애플리케이션이 일 세트의 스케줄링 가이드라인들(scheduling guidelines)을 구성함으로써 그 작업의 실행에 대한 우선순위를 어떻게 정할지를 구성하게 할 수 있다. 그러나, 이러한 두 개의 특징들은 애플리케이션들에게 제공되는 격리에 관해 공정하게 제한되며, 그리고 WLS-MT 배치에서, 모든 JDK 리소스들과 같은 모든 리소스들이 파티션화되는 것, 그리고 파티션들에 대한 성능 격리를 보장하기 위해 이들의 소비가 관리되는 것이 바람직하다.

[0079] **소프트 및 하드 격리(Soft and Hard Isolation)**

[0080] 본 발명의 일 실시예에 따르면, 격리된 그리고 공유된 실행 환경들 간의 구분은 이원적(binary)이지 않고, 공유된-인프라구조-및-완전히-격리된 실행 환경들(shared-infrastructure-and-completely-isolated execution environments)과 비-격리된 실행 환경들(non-isolated execution environments) 간의 선택들의 연속(continuum)이 존재할 수 있다.

[0081] 본 발명의 일 실시예에 따르면, 시스템 관리자가, 동일한 애플리케이션 서버 환경 내의 상이한 파티션들 내에서 비-신뢰 테넌트(non-trusting tenant)들로부터의 코드를 제공하기 위해서, 시스템 관리자는 격리된 실행 환경에 대한 지원을 제공할 수 있고(예를 들어, MVM/격리 보호(MVM/Isolates protection), 격리된 VM 런타임(isolated VM runtimes) - [GC, 힙(Heap), 쓰레드(Threads), 시스템 클래스(system classes)], 디버깅 및 진단(debugging and diagnostics) 등), 이에 따라 시스템은 파티션들 내에서 일어나는 동작들을 서로 샌드박스(sandbox)화할 수 있는바, 이것은 실행 격리로서 지칭될 수 있다. 이것은 하드 격리(hard isolation)의 형태인 것으로 고려될 수

있다.

[0082] JDK 8u40에 의해 노출되는 RM API들로부터의 지원을 통해, 시스템은 또한 더 소프트한 형태의 격리를 제공할 수 있는바, 이 경우 특정의 공유된 JDK 리소스들에 대한 액세스는 격리되고, 이들의 격리는 의지 동작들에 걸쳐 유지되는데, 이것은 성능 격리로서 지칭될 수 있다.

[0083] **작업 관리자(Work Manager, WM)에 대한 관계(Relationship)**

[0084] 본 발명의 일 실시예에 따르면, 애플리케이션 서버(예를 들어, 웹로직 서버) 작업 매니저(WM)는 관리자로 하여금 애플리케이션이 일 세트의 스케줄링 가이드라인들(최대/최소 및 용량 제약들(Max/Min and Capacity Constraints), 응답시간 요청 클래스들(ResponseTime Request Classes))을 구성함으로써 그 작업의 실행에 대한 우선순위를 어떻게 정할지를 구성하게 할 수 있다. 이것은 WM으로 하여금 애플리케이션에 할당된 쓰레드들을 관리하게 할 수 있고, 그리고 이러한 쓰레드들에 대한 스케줄링 작업 인스턴스들을 관리하게 할 수 있고, 그리고 협약들을 유지하는 것을 돕게 할 수 있다.

[0085] 본 발명의 일 실시예에 따르면, WM은 그 스케줄링 정책들에 대해서 작업 인스턴스의 실행을 위해 할당된 시간을 사용한다. 작업 인스턴스에 대한 경과된 시간은 CPU 시간(CPU time)과 비 CPU 시간(non CPU time)(I/O 상에서의 대기(waiting), 시스템 호출 네이티브 호출(system calls native calls) 등)의 결합일 수 있고, 특정 사용의 경우(예를 들어, I/O 경계 작업부하(I/O bound workloads))에 대한 정책들을 특정하기 위한 더 좋은 측정치이다.

[0086] **리소스 소비 관리(Resource Consumption Management, RCM)를 통한 증진들(Enhancements)**

[0087] 본 발명의 일 실시예에 따르면, WM 파티션 공정한 배당(WM partition fair share)은, 공정한-배당 계산(fair-share computation)에서 파티션의 작업 인스턴스들 실행하기 위해, 경과된 시간을 사용할 수 있다. 웹로직 서버와 같은 애플리케이션 서버 환경에서 파티션들이 함께 위치하는 경우, CPU 레벨에서 성능 격리를 보장하는 것이 또한 중요하다.

[0088] 예를 들어, 파티션 P1이 I/O가-대부분인 작업부하(I/O-mostly workload)를 갖고 파티션 P2가 CPU-집약적인 작업부하(CPU-intensive workload)를 가지며, P1과 P2가 동일한 WM 공정한-배당을 갖는 경우를 고려한다. 작업 인스턴스들이 P1에 의해 제출되고 완료를 위해 동일한 시간을 가정하면, P1과 P2는 WM에 의해 유사한 방식으로 스케줄링되게 된다(예를 들어, 동일한 개수의 작업 인스턴스들이 P1 및 P2에 대해 스케줄링되고 실행되게 됨). 하지만, P2의 작업부하는 CPU 집약적이기 때문에, 공유된 CPU의 P2의 사용은 P1보다 더 크다. 이것은 바람직하지 않을 수 있다.

[0089] 본 발명의 일 실시예에 따르면, JDK RM API와 같은 API에 의해 제공되는 CPU 시간 사용 통지들은 리소스 상황에 누적되는 쓰레드당 CPU 시간을 고려한다. 이것은 시스템 관리자가 CPU 레벨에서 정책들을 특정하고 함께 위치하는 파티션들 간에 성능 격리를 달성하는 상이한 직교 방식(orthogonal way)을 제공한다. CPU 시간 리소스에 대해 이용가능한 공정한-배당 정책 및 WM 파티션 공정한-배당은 모두 상이한 결과들을 달성하기 위해 시스템 관리자에 의해 유효하게 이용될 수 있다. I/O 관련 성능 격리를 위해, 파일 및 네트워크 리소스 레벨에서의 공정한-배당 정책들이 또한 RCM 특징을 통해 이용가능하다.

[0090] 본 발명의 일 실시예에 따르면, WM은 자신이 관리하는 쓰레드들에 관해서만 자신의 정책들을 실시할 수 있다. WM에 의해 관리되지 않은 쓰레드들로서 리소스 상황의 일부인 쓰레드들이 존재할 수 있고(예를 들어, 쓰레드들에 의해 생성(spawn)된 (파티션 내에 상주하는) 애플리케이션), 이들을 관리하기 위해 RCM 정책들이 사용될 수 있다. 교착화된 쓰레드 형국(stuck thread situation)들에 추가하여 JDK RM API 지원을 통해, 시스템은 런어웨이(runaway) 및 데드락된(deadlocked) 쓰레드들을 결정할 수 있고 마크(mark)할 수 있으며, 시스템 관리자로 하여금 이들에 관한 동작들을 행하게 할 수 있다.

[0091] **규칙-기반 탄력성(Rule-based Elasticity), 및 감시/통지(Watch/Notification)에 대한 관계(Relationship)**

[0092] 본 발명의 일 실시예에 따르면, 웹로직 진단 프레임워크(WebLogic Diagnostics Framework, WLDF)와 같은 진단 프레임워크의 감시 및 통지 컴포넌트는 시스템 관리자로 하여금 서버 및 애플리케이션 상태들을 모니터링할 수 있게 하며 기준들에 근거하여 통지들을 전송할 수 있게 한다. 이러한 감시들은 리소스가 소비된 이후 시스템의 상태에 관한 메트릭(metric)들의 모니터링에 대해 작동한다. RCM 정책들은 소비 요청 동안 정책들의 실시를 허용한다. RCM 정책들은 관리자가 리소스 소비자에 의한 그 공유된 리소스들의 사용을 정밀하게 성형하는 것을 돕는다.

[0093] 본 발명의 일 실시예에 따르면, 규칙-기반 탄력성 특징(rules-based elasticity feature)은 시스템 관리자들로 하여금 스케일링 관리 동작(scaling administrative action)들을 행하기 위해 클러스터에 걸쳐 리소스 사용을 모니터링하는 복합 규칙들을 구성할 수 있도록 WLDF 감시/통지 특징을 기반으로 구축될 수 있다. 규칙-기반 탄력성은, 마이크로 레벨(micro level) 상의 다수의 작업들에 영향을 미치는 동작들을 모니터링 및 수행하기 위해, '매크로(macro)' 레벨(예를 들어, 인스턴스에 대해 WLS 클러스터-전체 혹은 전반적 서비스들) 상의 이력적 경향(historical trend)들을 사용하는 것에 초점을 맞출 수 있는바, 공유된 컴포넌트들에 대한 개개의 소비자들에 의한 개별적 리소스 소비 요청들에 초점이 맞추어 지며, 전형적으로는 결과적으로 해당하는 그 리소스의 특정된 추가 사용에서의 변화들이 일어나게 하는 정책들(의지 동작들)에 초점이 맞추어 진다(예를 들어, 단지 현재의 VM에게만 영향을 미치는 동작).

[0094] 본 발명의 일 실시예에 따르면, JDK RM API는 JDK로-관리되는 리소스들에 대한 특정 리소스 소비 메트릭들에 관한 정보를 리소스-별 상황(예를 들어, 파티션)에 근거하여 제공할 수 있다. 이러한 것들은 이들이 임의의 대응하는 PartitionRuntimeMBean 상에서 속성들로서 표면화될 수 있도록 임의의 파티션에 국한된 범위를 갖는 모니터링 특징으로 전달될 수 있다.

[0095] **파티션 라이프싸이클 이벤트들(Partition Lifecycle Events), 및 개시/정지 지원(Start/Stop Support)에 대한 관계(Relationship)**

[0096] 본 발명의 일 실시예에 따르면, RCM 프레임워크는 쓰레드 내에 올바른 리소스 상황을 확립할 수 있고, 이에 따라 해당하는 그 쓰레드의 상황 내에서 소비되는 모든 리소스들은 구현에 의해 해당하는 그 리소스 상황에 대해 적절하게 고려되게 된다. RCM 프레임워크는, 새로운 파티션이 언제 개시 혹은 정지되는지를 알아내기 위해 그리고 새로운 리소스상황(resourcecontext)들을 생성하기 위해, 혹은 이들을 삭제하기 위해, 웹로직 서버 이벤트 특징(WebLogic Server events feature)과 같은 애플리케이션 서버 특징을 통해 제공되는 파티션 개시/정지(및 만약 이용가능한 경우, 인에이블/디스에이블(enable/disable)) 이벤트들을 사용할 수 있다.

[0097] 본 발명의 일 실시예에 따르면, 파티션 파일 시스템은 가상 파일 시스템을 구현할 수 없기 때문에, 파티션의 파일 시스템에 대한 액세스는 java.[n]io을 통해 존재할 수 있고, 애플리케이션 서버는 이에 관한 파일 I/O 동작들을 인터셉트(intercept)할 수 없다. 그러나, RCM 프레임워크는 JDK 파일 기술자(file descriptor) 및 파일 관련 리소스들을 시스템 관리자에게 직접적으로 노출할 수 있다.

[0098] **협력적 메모리 관리(Co-operative Memory Management, CMM)에 대한 관계(Relationship)**

[0099] 본 발명의 일 실시예에 따르면, 협력적 메모리 관리(Co-operative Memory Management, CMM) 특징은, 시스템의 상이한 부분들이 메모리 압력에 반응하도록 함으로써 낮은-메모리 형국(low-memory situation)들을 핸들링하는 데 사용될 수 있다. 메모리 압력은 외부적으로 결정될 수 있다. 이 경우, 시스템 내에 상주하는 애플리케이션들, 미들웨어(Middleware), 및 JVM은 메모리 압력 레벨에 반응할 수 있다. 예를 들어, 높은 메모리 압력 형국들에서, 최소로-사용된 애플리케이션들은 아이들(idle)/동면(hibernate) 상태에 들어갈 수 있으며, JDBC/JMS 및 코히어런스 서브시스템(Coherence subsystem)들은 자신들의 캐시 크기(cache sizes)를 감소시킬 수 있고, JVM은 자신의 힙 크기(heap size) 등을 감소시킬 수 있다.

[0100] 본 발명의 일 실시예에 따르면, CMM 특징은 매크로(기계(machine)) 레벨에서 메모리 압력 시나리오들을 핸들링하기 위해 사용될 수 있고, 그리고 높은 메모리 형국들이 일어난 이후 반응할 수 있다. 임의의 기계에서의 높은 메모리 압력은 그 기계에서의 외부의 오류가 있는 JVM 혹은 프로세스의 결과일 수 있고, 메모리 압력의 "소스(source)"를 제어하기 위해서, 그리고 (할당된 그리고 보유된 메모리 크기)의 해당 지점에서의 비정상적인 메모리 사용을 제어하기 위해서, 어떠한 동작도 행해질 필요가 없다. 정책들은 '마이크로(micro)'(WLS 서버 JVM) 레벨에서 동작할 수 있고, 그리고 오류가 있는 "리소스 소비자"가 힙을 사용하는 것을 능동적으로 제어할 수 있다. 따라서, CMM 특징, 그리고 시스템 관리자에 위한 RCM 정책들의 적용은 성질상 보완적인 것이다.

[0101] **리소스 관리 지원(Resource Management Support)**

[0102] 본 발명의 일 실시예에 따르면, JDK 8u40과 같은 개발 키트에서 제공되는 리소스 매니저 API는 리소스 계측(resource instrumentation) 및 리소스들의 소비의 통지로부터 리소스 정책 실시를 분리시킬 수 있다. RM API는 리소스 소비 매니저와 같은 외부 매니저로 하여금 애플리케이션 도메인 내의 리소스 소비자 쓰레드들을 리소스 상황과 관련시키도록 할 수 있다. 리소스상황은 리소스 소비 요청을 대응하여 고려하기 위해 개발 키트에 대한 상황을 제공할 수 있다. RM API는 또한 리소스 소비 매니저와 같은 외부 매니저로 하여금 리소스미터(resourcemeter)들을 통해서 리소스상황에 의해 (리소스Id(resourceId)들에 의해 식별되는) 특정 리소스들의 소

비에 관한 정보를 획득하기 위한 관심(interest)을 등록하게 할 수 있다.

[0103] 본 발명의 일 실시예에 따르면, JDK와 같은 개발 키트에 의해 제공되는 표준 리소스미터 구현들은, 심플미터(simplemeter)(이것은 리소스 상황에 대한 성공적인 리소스 할당들을 단순히 카운팅(counting)함); 바운디드미터(boundedmeter)(이것은 상한(upper bound)을 실시하여 카운팅함); 통지미터(notifyingmeter)(리소스들의 구성된 세밀한 영역(configured granular county)에 대한 카운트(count)들 및 요청(request)들의 승인(approval)이 리소스 할당들 이전에 리소스 매니저를 형성함); 그리고 쓰로틀미터(throttlemeter)(이것은 소비를 초당 특정 속도로 한정함)이다.

[0104] 본 발명의 일 실시예에 따르면, 심플 미터 및 바운디드 미터와 같은 특정 미터들은 풀 방식(pull manner)으로 정보를 획득한다. 통지미터와 같은 다른 미터들은 리소스 매니저가 리소스 소비 요청을 수락(accept), 저속화(slow), 혹은 거부(deny)하기 위한 정책을 실시하도록 하기 위해 리소스 매니저에게 사전-소비 통지 호출(pre-consumption notification call)들을 다시 전송할 수 있다.

[0105] **RM-인에이블된 JDK 리소스들(RM-Enabled JDK Resources)**

[0106] 본 발명의 일 실시예에 따르면, JDK에 의해 관리되는 리소스들(이들은 RM-인에이블될 수 있으며, 따라서 시스템 관리자로 하여금 이들에 관해 리소스 관리 정책들을 특정하게 할 수 있음)의 예들은, 오픈 파일 기술들(Open File Descriptions), 힙(Heap), 쓰레드들(Threads), CPU 시간, 등을 포함한다. 추가적으로, 리소스 소비 메트릭들로서 이용가능하게 될 수 있는 리소스들로는 다음과 같은 것: 파일 기술자들(file descriptors), 파일들(files)(예를 들어, 오픈 파일 카운트(open file count), 전송된 바이트들(bytes sent), 수신된 바이트들(bytes received), 총계(totals)), 소켓들(sockets) 및 데이터그램들(datagrams), 힙(heap), 쓰레드들(threads), 활성 쓰레드 카운트(active thread count), CPU 시간이 있다.

[0107] 본 발명의 일 실시예에 따르면, CPU 시간 측정들은 (각각의 리소스상황(ResourceContext)에서 활성 상태의 쓰레드들을 샘플링(sampling)할 수 있고 업데이트(update)들을 미터(meter)들에게 전송할 수 있는 쓰레드를 모니터링하는) JDK와 같은 개발 키트에 의해 주기적으로 수행될 수 있다.

[0108] 본 발명의 일 실시예에 따르면, G1 가비지 수집기(garbage collector)는 영역-기반의 할당기(region-based allocator)일 수 있다. 가비지 수집이 수행되고, 객체(object)들이 카피(copy)됨에 따라, 가비지 수집기는 동일한 리소스 상황을 갖는 영역으로 객체들이 카피되는 것을 보장할 수 있다.

[0109] **리소스 소비 관리(Resource Consumption Management) - 리소스(Resource), 트리거(Trigger)**

[0110] 본 발명의 일 실시예에 따르면, 시스템 관리자는 RM-인에이블된 리소스들에 관한 RCM 정책들을 리소스-소비자-별 기반으로 특정할 수 있다. 임의의 리소스에 대해, 시스템 관리자는 하나 이상의 트리거(trigger)들을 특정할 수 있고(예를 들어, 리소스의 최대 사용은 400개의 유닛들로 한정됨), 뿐만 아니라 트리거가 해당하는 값에 도달할 때 수행돼야만 하는 동작을 특정할 수 있다. 이러한 동작들은 결과적으로 리소스 소비 요청이 발생된 쓰레드와 동일한 쓰레드에서 임의의 활동(activity)이 일어나게 할 수 있고(동시성(synchronous)), 또는 리소스 소비 요청이 발생된 쓰레드와는 다른 쓰레드에서 수행될 수 있다(비동시성(asynchronous)). 트리거/동작 결합은 시스템 관리자로 하여금 리소스-소비자에 의한 리소스의 사용을 성형, 제어 및 한정할 수 있게 한다.

[0111] 본 발명의 일 실시예에 따르면, 시스템 관리자가 지정할 수 있는 수 개의 동작들이 존재한다. 이러한 것들은 통지(notify), 저속화(slow), 불이행(fail), 및 멈춤(shutdown)을 포함하지만, 이러한 것으로만 한정되는 것은 아니다. 통지(notify) 동작은 정보를 제공하는 업데이트로서 시스템 관리자에게 통지를 제공한다. 저속화(slow) 동작은 리소스가 소비되는 속도를 떨어뜨릴 수 있다(예를 들어, 둔화(slow down)시킬 수 있음). 불이행(fail) 동작은 하나 이상의 리소스 소비 요청들에 관해 불이행하는 것일 수 있다. 불이행 동작은 또한 리소스의 사용이 그 원하는 한계치 아래로 떨어지는 경우 종결(termina)될 수 있고, 따라서 리소스 소비가 다시 허용가능하다. 멈춤(shutdown) 동작은 SIGTERM-등가(SIGTERM-equivalent)일 수 있으며(예를 들어, 그 종결을 요청하기 위해 프로세스로 전송되는 신호), 그리고 리소스 소비를 정지시켜려고 시도하는 것(아울러 소비자로 하여금 클린업(cleanup)하는 능력을 허용하는 것)일 수 있다.

[0112] 본 발명의 일 실시예에 따르면, RCM은 JSR-284 API 기반의 웹로직 서버 RCM 프레임워크이고, 그리고 웹로직 컨테이너들, 컴포넌트들, 및 툴(tool)들에 의한 사용을 위해서 경량의 RCM SPI(Service Provider Interface; 서비스 제공자 인터페이스) 및 API를 제공한다.

[0113] 이제 도 7를 참조하면, 도 7은 본 발명의 일 실시예에 따른, 리소스 소비 관리 구현을 예시한다. 도시된 바와

같이, 리소소 소비 관리 구현(700)은, 리소스 소비자(705), 리소스 도메인(710), 리소스 속성들(715), 리소스(720), 리소스 디스펜서(725), 제약(740), 통지(750), 트리거들(760), 및 JDK 파일 I/O 서브시스템(730)을 포함할 수 있다. 도 7에서 예시되는 바와 같이, 트리거들(760)은 시스템 관리자에 의해 설정될 수 있는바, 예를 들어, 도 6에서 제시되는 바와 같이 리소스 소비 관리 정책(630) 내에서 시스템 관리자에 의해 설정될 수 있다.

[0114] 리소스 소비자(705)(예를 들어, 파티션)가 리소스들을 소비하기 시작할 때, 이러한 리소스들의 사용은 모니터링된다. 예를 들어, 소비되는 리소스들이 CPU, 힙, 파일, 및 네트워크를 포함할 수 있음을 상기하면, 리소스 소비 관리 구현은 시스템 관리자에 의해 설정된/구성된/특정된 트리거들(760)에 대비하여 그 소비되는 리소스들을 모니터링 및 비교한다.

[0115] 예를 들어, 도 7에서 제시되는 바와 같이, 시스템 관리자에 의해 설정된 트리거는, 리소스의 사용이 70 이상일 때(예를 들어, CPU 시간의 70% 이상일 때), 통지가 시스템 관리자에게 로깅/디스플레이되는 것이다. 리소스 소비자에 대한 CPU 시간이 70%를 넘어가는 경우, 리소스 소비 관리 정책은 임의의 파일에 로그를 푸시(push)(755)할 수 있는데, 여기서 로그 파일(log file)은 리소스 소비자가 트리거 내의 시스템 관리자에 의해 설정된 바와 같은 임계 값을 넘었음을 표시한다.

[0116] 본 발명의 일 실시예에 따르면, 시스템 관리자가 트리거 내의 의지 동작 타입으로서 "통지(Notify)"를 특정할 때, WLS RCM 프레임워크는 도달되는 트리거 내에서 특정된 사용에 대한 표준 메시지를 로깅할 수 있다. 이것은 메시지를 (메시지의 일부분으로서) 요구된 보충 속성들(예를 들어, 현재 사용(Current Usage), 이전의 사용(Previous Usage), 피티션에 대해 도달된 할당치 통지(Notifying Quota Reached For Partition))과 함께 로깅한다. 시스템 관리자는 표준 로그 메시지(standard log message)를 청취할 감시 규칙(watch rule)을 구성하도록 WLDF 시스템 모듈을 배치할 수 있고, 그리고 통지들을 전송하기 위해 프레임워크 내의 통지 기능을 사용할 수 있다.

[0117] 추가적으로, 도 7에 도시된 바와 같이, 리소스의 소비자에 의한 리소스의 사용이 100 이상일 때 제약이 발생(throw)된다(예를 들어, 오픈 파일 기술자들(Open File Descriptors)). 예를 들어, 리소스 소비자(예를 들어, 파티션)가 한계치(limit) 100을 초과한다고 가정한다. 그러면, 시스템 관리자에 의해 설정된 트리거에 근거하여, 제약이 호출되게 되고, 제외(exception)가 발생될 수 있다(예를 들어, 요청된 파일을 오픈하는 데 실패함(745)).

[0118] 본 발명의 일 실시예에 따르면, 제약의 또 하나의 다른 예는 둔화(slow-down)이다. 둔화는 파티션 작업 매니저의 공정한-배당을 감소시키는 것을 포함할 수 있고, 그럼으로써 작업 매니저는 해당 파티션에 대한 작업 인스턴스들의 개수를 더 적게 스케줄링하게 될 것이다. 추가적으로, 작업 매니저는 또한 파티션의 동시발생하는 활성 작업 인스턴스들의 최대 개수를 감소시키기 위해 파티션의 최대-쓰레드-제약(max-threads-constraint)을 감소시킬 수 있다. 둔화는 또한 애플리케이션에 의해 생성된 쓰레드들의 우선순위를 감소시킬 수 있다. 둔화 동작/제약은 파티션을 더 둔화시키기 위해서 여러 번 호출될 수 있다. 이것은 결과적으로 파티션 작업-매니저의 공정한-배당, 최대-쓰레드-제약 및 애플리케이션 쓰레드들의 우선순위의 값들이 더 감소되게 할 수 있다. 이것은 예를 들어, 공정한-배당 정책 지원의 경우에 행해질 수 있다. 유사하게, 저속화 조건은 취소(revoke)될 수 있고, 이것은 결과적으로 파티션 작업 매니저의 공정한-배당 및 최대-쓰레드-제약을 본래 구성된 값들로 되돌리게 되고, 그리고 또한 애플리케이션 쓰레드들에 대한 정상적인 우선순위가 복원되게 한다.

[0119] 본 발명의 일 실시예에 따르면, 트리거가 충족된 이후 의지 동작으로서 파티션 멈춤 동작(shutdown partition action)이 특정될 수 있다. 시스템 관리자가 의지 동작들 중 하나로서 멈춤을 특정하는 경우, 파티션의 리소스 소비가, 특정된 할당치를 위반(breach)하는 경우, 해당하는 그 파티션의 멈춤이 일어날 것이다.

[0120] 본 발명의 일 실시예에 따르면, 파티션 멈춤이 일어나는 경우, 적절한 클린업(cleanup)이 행해지게 되며, 대응하는 JDK 리소스상황(JDK ResourceContext)들이 클로즈(close)될 수 있다. 파티션 및 그 애플리케이션들은 파티션의 멈춤이 일어나면 액세스가능하지 않게 된다.

[0121] **리소스 플러그어빌리티(Resource Pluggability)**

[0122] 본 발명의 일 실시예에 따르면, 리소스 구현자(resource implementer)는 리소스 소비 관리 프레임워크 내에서 그들의 리소스 속성들을 동록하기 위해 SPI를 사용할 수 있다. 관리자는 이러한 리소스 속성들에 관한 리소스 관리 정책들을 정의할 수 있다. 리소스 속성들을 통해, 리소스 구현자는 리소스의 다양한 특징들(처분가능성(disposable), 한계(bounded), 예약가능성(reservable), 세밀화도 (granularity), 측정의 유닛(unit of measurement), 등)을 기술할 수 있다. 리소스 구현자는 또한 리소스의 세밀화도 및 디폴트 최대 측정 지연

(default maximum measurement delay) 및 속도 관리 기간(rate management period)을 결정할 수 있다. 리소스 구현자는 측정의 성능 영향을 제어하기 위해 이러한 값들을 선택할 수 있다.

[0123] 본 발명의 일 실시예에 따르면, 리소스 구현자는 리소스의 임의의 분량(quantity)이 소비될 수 있는지를 점검하기 위해서 RCM 프레임워크에 대해 소비() 호출(consume() call)들을 삽입할 수 있다. 이것은 RCM 프레임워크로 하여금 리소스에 대한 액세스를 요청하는 그러한 리소스 소비자에 대한 리소스에 관해서 리소스 관리 정책들을 점검 및 실시할 수 있게 한다. 처분가능한 리소스들에 대해, 리소스 구현자는, 리소스가 이제 더 이상 리소스 컨테이너에 의해 사용되지 않으며 이에 따라 재사용(reuse)을 위해 이용하가능하게 된 것을 표시하기 위해서, RCM 프레임워크에 대해 철회() 호출(relinquish() call)을 삽입할 수 있다.

[0124] 본 발명의 일 실시예에 따르면, RCM 프레임워크는 특정된 분량이 리소스 소비자에게 제공될 수 있는지를 점검하기 위해 임의의 적절한 리소스 도메인 및 그 관련된 정책들을 참고(consult)할 수 있다.

[0125] **리소스 소비자 플러그어빌리티(Resource Consumer Pluggability)**

[0126] 본 발명의 일 실시예에 따르면, RCM 프레임워크는 새로운 리소스 소비자 타입이 SPI를 통해 플러그인(plug in)되게 할 수 있다. 리소스 소비자는 예를 들어, 파티션일 수 있다. 리소스 디스펜서는 다른 리소스 소비자로부터 하나의 리소스 소비자에 의한 리소스의 사용을 결정, 격리 및 구분할 수 있다. 예를 들어, 만약 리소스 소비자 타입이 파티션이라면, 그리고 액세스되고 있는 리소스가 임의의 공유된 데이터소스에 관한 다수의 오픈 연결(open connection)들이라면, 데이터소스 컨테이너는 임의의 파티션으로부터의 연결 요청(및 그 요청의 후속 만족(satisfaction))을 다른 파티션에 의한 연결 요청과 구분할 수 있다. 리소스의 사용이 임의의 고유한 리소스 소비자에게 명확하게 할당될 수 없다면 격리 및 공정성은 보장되지 않을 수 있다.

[0127] **자바 개발 키트(Java Development Kit)의 리소스 매니저 API(Resource Manager API)**

[0128] 본 발명의 일 실시예에 따르면, 앞서 설명된 RM-인에이블된 JDK 리소스들은 JSR-284 리소스속성(ResourceAttribute)을 통해 나타내어질 수 있다. 리소스를 나타내는 프록시 리소스디스펜서(proxy ResourceDispenser)가 구현될 수 있고 JDK RM API와 인터페이스할 수 있다.

[0129] 본 발명의 일 실시예에 따르면, WLS RCM 구현은 파티션 시동 이벤트(Partition start-up event)들에 대해 청취할 수 있으며, 개시된 파티션에 대한 리소스상황을 생성할 수 있다. 해당하는 그 파티션에 대해 특정된 RCM 정책 구성에 근거하여 모니터링될 필요가 있는 각각의 JDK 리소스에 대해서 JDK 리소스미터(ResourceMeter)들이 생성될 수 있다. 그 다음에, 이러한 JDK 리소스미터들은 파티션에 대한 JDK 리소스상황과 관련될 수 있다.

[0130] 본 발명의 일 실시예에 따르면, WLS RCM 구현은 JDK 리소스승인자(ResourceApprover)(청취자)를 각각의 JDK 리소스미터(예를 들어, 통지미터)에 등록할 수 있다. 이러한 청취자는 매 사전-소비 할당 요청마다 JDK에 의해 다시 호출될 수 있고, 그 다음에 청취자는 이것을 RCM 프레임워크 내에서 적절한 JSR-284 리소스도메인에 위임(delegate)할 수 있다. RCM 프레임워크는 섹션 4.2.8에서 설명된 바와 같은 그러한 리소스 상황에 대한 리소스 소비 관리 정책을 구현한다.

[0131] 이제 도 8을 참조하면, 도 8은 본 발명의 일 실시예에 따른, 리소스 소비 관리 통합에서 상호작용들을 보여주는 시퀀스 도면을 예시한다. 도 8에 도시된 시퀀스는 애플리케이션(805)이 리소스들을 소비하게 될 동작을 수행하는 것으로 시작한다. 그 다음에 JDK(810)는 그 양(amount)을 요청한다. 통지카운터(NotifyingCounter)(815)는 리소스 요청을 수행하고, 그 다음에 리소스승인자(820)는 소비 동작을 수행한다. 리소스도메인(825)은 사전소비(preConsume)를 수행하고, 이것은 또한 파티션들의 현재 및 제안된 사용에 관한 표시자들을 포함할 수 있다. 그 다음에, 시스템 관리자에 의해 구성된 바와 같은 제약(830)이 정책을 실행할 수 있다. 정책(835)은 그 설정들에 따라, 애플리케이션에 의해 요청된 리소스들의 양을 리턴(return)하거나, 혹은 어떠한 리소스도 리턴하지 않음으로써 리소스들에 대한 요청을 거부할 것이다. 이러한 결정은 시스템 관리자에 의해 구성될 수 있는 정책에 근거하고 있다.

[0132] **쓰레드 내에 올바른 상황 확립(Establishing the Correct Context in the Thread)**

[0133] 본 발명의 일 실시예에 따르면, JDK RM 구현은 쓰레드와 관련된 리소스상황에 대해 쓰레드 내에서 일어나는 리소스 소비를 고려하기 때문에, 시스템은 리소스들의 잘못된-어카운팅(mis-accounting)을 방지하기 위해 쓰레드에 관해 올바른 리소스상황을 확립할 필요가 있을 수 있다. RCM(이것은 WLS RCM일 수 있음)은 리소스상황에 대해 사용될 파티션을 결정하기 위해 현재 쓰레드의 컴포넌트 호출 상황(Component Invocation Context, CIC)을 사용할 수 있다. WLS RCM 구현은 쓰레드 내에서의 CIC가 변할 때 통지들을 수신하기 위해 컴포넌트호출상황변화

청취자(ComponentInvocationContextChangeListener)를 등록할 수 있다. 이러한 CIC 변화 통지가 수신되는 경우, 그리고 CIC 변화가 파티션-스위치(partition-switch)로 인한 것일 때(예를 들어, 크로스-파티션(cross-partition), 또는 '파티션에서 글로벌로의 전이들'/'글로벌로부터 파티션으로의 전이들'(partition to/from Global transitions)로 인한 것일 때), WLS RCM 구현은 쓰레드로부터 현재의 리소스상황을 결합해제(unbind)시킬 수 있고, 그리고 새로운 파티션과 관련된 리소스상황을 결합시킬 수 있다.

[0134] 이제 도 9를 참조하면, 도 9는 본 발명의 일 실시예에 따른, 리소스 소비 관리 구현을 예시한다. 도시된 바와 같이, 리소소 소비 관리 구현(700)은, 리소스 소비자(705), 리소스 도메인(710), 리소스 속성들(715), 리소스(720), 리소스 디스펜서(725), 제약(740), 통지(750), 트리거들(760), 및 JDK 파일 I/O 서브시스템(730)을 포함할 수 있다. 예시되는 바와 같이, 트리거들(760)은 시스템 관리자에 의해 설정될 수 있는바, 예를 들어, 도 6에서 제시되는 바와 같이 리소스 소비 관리 정책(630) 내에서 시스템 관리자에 의해 설정될 수 있다. 추가적으로, 리소스 소비 관리 구현은 리소스상황(resourcecontext)(910)과 관련되고, 이러한 리소스상황(910)은 현재 쓰레드의 컴포넌트 호출 상황(component invocation context)(920)을 참조함으로써 설정될 수 있다. 이것은 제약 혹은 통지가 (트리거될 때) 활성 리소스 소비와 관련되는 것을 보장할 수 있다.

[0135] 본 발명의 일 실시예에 따르면, 쓰레드가 새로운 쓰레드를 생성할 때, JDK RM API는 페어런트 쓰레드(parent Thread)에서 확립된 리소스상황을 자동으로 물려받을 수 있다.

[0136] **JDK 리소스 메트릭들(JDK Resource Metrics)**

[0137] 본 발명의 일 실시예에 따르면, 다양한 컴포넌트들이 파티션 특정 리소스 사용 메트릭들을 얻을 수 있다. RCM 구현은 API를 통해 파티션 특정 리소스 소비 메트릭들을 획득하기 위해 모니터링을 위한 API를 노출할 수 있다.

[0138] **CPU 이용(CPU Utilization)**

[0139] 본 발명의 일 실시예에 따르면, JDK 리소스 관리 API는 리소스 상황의 리소스 소비자들에 의해 소비된 CPU 시간을 측정하는 것을 지원한다. 멀티-파티션 환경에서, 각각의 파티션은 임의의 리소스 상황에 맵핑(mapping)될 수 있고, 그리고 쓰레드들은 파티션들을 위해서 작업을 행하기 위해 리소스 상황들에 결합될 것이다. CPU 시간 메트릭들을 갖는 경우, RCM 프레임워크는 파티션에 의해 이용되는 CPU 시간을 측정할 수 있다. 그러나, CPU 사용을 절대적인 수들(예를 들어, 1시간, 30분)에 의해 리소스로서 표현하는 것은 시스템 관리자들에게 유용하지 않을 것인데, 왜냐하면 CPU는 한정되지 않은(제한 없는) 리소스이고 시스템 관리자가 절대적인 CPU 사용 시간에 관해 파티션들에 대한 한계치들/할당치들을 특정하는 것은 실제로 값을 한정시키는 것(혹은 비-직관적(non-intuitive)인 것)이기 때문이다. 이로 인해, (CPU 사용으로부터 도출될 수 있는) CPU 이용 리소스 타입은 퍼센트 값(1-100)을 통해 표현될 수 있다. CPU 이용 퍼센티지는 시스템 프로세스에 대한 이용가능한 CPU에 관해서 파티션에 의해 이용되는 CPU의 퍼센티지를 표시한다. 이러한 값은 CPU 부하 인자(load factor) 및 파티션에 의한 CPU 소비 퍼센티지를 사용하여 계산된다.

[0140] 본 발명의 일 실시예에 따르면, CPU 소비 퍼센티지는 WLS 프로세스에 의한 CPU의 총 소비에 대해서 파티션에 의한 CPU의 소비를 근거로 계산될 수 있다. 그러나, WLS 프로세스의 총 CPU 소비에 대해서 파티션의 CPU 이용의 배당몫을 고려하는 것이, 정책들을 설정할 때 시스템 관리자가 참고 및/또는 참조하기에 항상 유용한 측정치를 산출하지는 않을 것이다.

[0141] 예를 들어, 만약 단지 하나의 파티션만이 WLS 프로세스에서의 활성 참여자라면, 그리고 WLS 프로세스가 실행되고 있는 기계가 경량으로(lightly) 로딩된다면, 전체 WLS 프로세스의 CPU 사용에 대해서 파티션의 CPU 사용의 단순한 비율을 해당하는 그 파티션에 대한 CPU 이용으로서 처리하는 것은 결과적으로 과도한-표현(over-representation)(해당하는 그 파티션에 대해 매우 높은 CPU 이용 값)이 일어나게 하며, 그리고 해당하는 그 파티션을 불필요하게 처벌(punish)하게 된다.

[0142] 추가적인 예로서, 임의의 도메인 내에 두 개의 파티션들이 존재한다고 가정한다. 파티션-1에 대한 CPU 소비 퍼센티지는 95이고, 파티션-2에 대해서는 4이며, WLS 프로세스가 실행되고 있는 기계는 경량으로 로딩된다. 이러한 경우에도, 전체 WLS 프로세스의 CPU 사용에 대해서 파티션의 CPU 사용의 단순한 비율의 처리는 결과적으로 파티션-1을 불필요하게 처벌하게 된다.

[0143] 본 발명의 일 실시예에 따르면, 시스템이 중량적으로(heavily) 로딩되지 않는 앞서의 시나리오들을 (애플리케이션 서버에 의한 CPU의 중량적 사용(heavy usage)으로 인해 혹은 외부 프로세스에 의한 CPU의 중량적 사용으로 인해) 시스템이 중량적으로 로딩될 수 있는 다른 시나리오들과 구분하기 위해서, 추가적인 부하-인자가 CPU 소비 퍼센티지에 적용될 수 있다. 이러한 추가적인 인자는 CPU 부하이고 하나 이상의 API들의 사용으로 계산될 수

있다.

[0144] 본 발명의 일 실시예에 따르면, 서버 내에 단지 하나의 활성 파티션만이 존재하는 형국들에서, 부하-인자는 프로세스CPU부하(ProcessCPULoad) 및 경합_임계치(CONTENTION_THRESHOLD)에 근거할 수 있다. 경합_임계치(CONTENTION_THRESHOLD) 값은 예를 들어, 0.8(혹은 CPU의 80%)일 수 있다.

[0145] 본 발명의 일 실시예에 따르면, 서버 내에 하나보다 많은 활성 파티션이 존재하는 형국들에서, RCM은, 만약 프로세스CPU부하(ProcessCPULoad) 값이 80(경합_임계치(CONTENTION_THRESHOLD))보다 더 크다면, 부하-인자를 계산하기 위해 프로세스CPU부하(ProcessCPULoad) 값을 사용할 수 있다. 이것은 시스템(예를 들어, WLS) 프로세스가 해당 기간 동안 CPU의 80%보다 더 많이 이용하고 있음을 표시한다.

[0146] 본 발명의 일 실시예에 따르면, 서버 내에 하나보다 많은 활성 파티션이 존재하는 형국들에서, 그리고 프로세스CPU부하(ProcessCPULoad) 값이 0.8(혹은 CPU의 80%)보다 작은 경우에, 이것은 시스템(예를 들어, WLS) 프로세스가 CPU 집약적(CPU intensive)이지 않음을 표시할 수 있다. 이와 같은 경우에, RCM은 시스템CPU부하(SystemCPULoad)가 0.8(혹은 CPU의 80%)보다 큰지 여부를 점검할 수 있다. 만약 이것이 0.8(혹은 CPU의 80%)보다 크다면, 부하-인자를 계산하기 위해 시스템CPU부하(SystemCPULoad)가 사용될 수 있다. 이것은 또한 시스템 내에 CPU 집약적인 다른 프로세스들이 존재함, 그리고 JVM 프로세스는 한정된 CPU를 제공받음을 표시한다.

[0147] 본 발명의 일 실시예에 따르면, 서버 내에 하나보다 많은 활성 파티션이 존재하는 앞서의 형국들 중 어느 하나의 형국에 있어서, 만약 CPU 사용이 경합_임계치(CONTENTION_THRESHOLD)보다 더 많다면, 이것은, 시스템이 충분히 로딩됨, 그리고 파티션은 JVM 프로세스에 대한 이용가능한 CPU에 관해서 다른 파티션들을 희생시키고 CPU를 소비하고 있을 수 있음(예를 들어, 리소스 경합이 존재함)을 표시한다.

[0148] 본 발명의 일 실시예에 따르면, 서버 내에 하나보다 많은 활성 파티션이 존재하는 형국들에서, RCM은 프로세스CPU부하(ProcessCPULoad) 혹은 시스템CPU부하(SystemCPULoad)가 이들의 경합_임계치(CONTENTION_THRESHOLD)를 초과했는지 여부에 근거하여 프로세스CPU부하(ProcessCPULoad) 혹은 시스템CPU부하(SystemCPULoad)를 부하-인자로서 고려할 수 있다.

[0149] 본 발명의 일 실시예에 따르면, 서버 내에 하나보다 많은 활성 파티션이 존재하는 형국들에서, 프로세스CPU부하(ProcessCPULoad) 및 시스템CPU부하(SystemCPULoad)가 모두 이들의 경합_임계치(CONTENTION_THRESHOLD)보다 작은 경우, 부하-인자는 프로세스CPU부하(ProcessCPULoad) 및 경합_임계치(CONTENTION_THRESHOLD)에 근거할 것이다.

[0150] 본 발명의 일 실시예에 따르면, CPU 부하 인자는 CPU에 관한 경합을 결정함에 있어서 보조할 수 있고, 그럼으로써 JDK 리소스 관리 API에 근거하여 도출되는 CPU 소비 메트릭들의 값을 또한 정량화(quantify)할 수 있다.

[0151] 본 발명의 일 실시예에 따르면, CPU 이용 값은 통지, 저속화, 멈춤 및 공정한-배당과 같은 다양한 정책들 및 의지방안(recourse)들을 구성하는데 사용될 수 있다. CPU 시간 소비에 관한 JDK로부터의 통지들은 사후 소비 호출(post consumption call)들이기 때문에, CPU 이용 리소스 타입에 대한 요청별 불이행 의지 동작(per request fail re-course action)을 지원하는 것은 가능하지 않다.

[0152] 본 발명의 일 실시예에 따르면, 통지, 저속화, 멈춤 등과 같은 의지 동작들은 CPU 이용 값의 감쇄되는 평균(decayed average)이 임의의 기간 동안 일관적으로 임계치보다 클 때 행해진다. 일반 폴링 메커니즘 기반의 알고리즘(generic polling mechanism based algorithm)이, 보유된 힙(Heap Retained) 및 CPU 이용(CPU Utilization) 리소스 타입들 모두에 의해 사용될 수 있고, 이것은 또한 스파이크들(spikes)(산발적인 소비의 급상승/하강들)을 회피/무시하는데 도움을 준다.

[0153] 본 발명의 일 실시예에 따르면, 공정한-배당 정책은 CPU 이용 리소스 타입에 대해 지원되며, 이것은 서버 내에서 모든 파티션들의 공정한-배당 값들에 근거하여 작동한다.

[0154] **보유된 힙(Heap Retained)**

[0155] 본 발명의 일 실시예에 따르면, API(예를 들어, JDK 리소스 관리 API)는 특정 리소스 상황에서 힙 할당 및 보유된 힙을 측정하는 것을 지원할 수 있다. 이러한 리소스 타입들 모두에 대한 콜백은 사후 소비이다. 이로 인해, 힙 소비를 제어하기 위해 선행하여 어떠한 동작도 수행될 수 없다.

[0156] 본 발명의 일 실시예에 따르면, 힙 할당의 경우에, JDK RM은 특정 리소스 상황에 대한 힙 할당들 전체를 실행시키기 위해 콜백을 제공한다. JDK로부터의 힙 할당 통지들은 단조적으로(monotonically) 증가하고 있을 수 있고,

일반적으로 해당하는 그 리소스 상황에 의한 사용에서 실제 힙을 반영하지 않는다. WLDF 규칙들은 힙이 할당된 파티션에 국한된 범위를 갖는 메트릭들의 사용을 통해 힙 할당을 모니터링하도록 구성될 수 있다.

[0157] 본 발명의 일 실시예에 따르면, 보유된 힙의 경우, 콜백은 가비지 수집기 활동의 결과로서 JDK로부터 발신(originate)될 수 있다. 보유된 힙 통지(heap retained notification)는 관련된 정확도 레벨(accuracy level)(예를 들어, 신뢰도 인자(confidence factor))을 가질 수 있다. 힙 사용의 높은 신뢰도 측정들은 전체 가비지 수집 이후에 수행될 수 있고, 따라서 이것은 결과적으로 높은 정확도 통지가 일어나게 한다. 이후 힙 사용 측정의 신뢰도는 감소한다. 이벤트 완료를 동시에 마크하는 G1 가비지 수집(G1 garbage collection concurrent marking event completion)은 높은 신뢰도 값들을 산출할 수 있고, 따라서 이것은 높은 정확도 통지가 일어나게 한다. 마이너 가비지 수집(minor garbage collection)은 측정된 값들에 관해 낮은 신뢰도를 제공할 수 있고, 따라서 이것은 더 낮은 신뢰도 통지가 일어나게 한다. 다음 가비지 수집 싸이클(garbage collection cycle)까지, 힙 사용에 대해 JDK에 의해 보고되는 값들은 비-선형적으로(non-linearly) 감소하는 신뢰도 인자를 갖게 된다. 보유된 힙 사용 정보의 정확도는, 정확한(accurate) 것(리소스 상황에 대한 가비지 수집 이후 남겨진 모든 객체들을 태깅(tagging)함, 그리고 크기의 총합을 구함) 그리고 값비싼(expensive) 것일 수 있거나, 혹은 더 개략적(coarser)인 것 그리고 더 값싼(cheaper) 것(리소스 소비자에게 할당된 G1 영역들을 카운팅함)일 수 있다. 가비지 수집 싸이클들 간의 신뢰도 인자는 과거 할당 이력(past allocation history)에 근거하고 있을 수 있으며 정상-상태 할당(steady-state allocation)을 추정할 수 있다.

[0158] 본 발명의 일 실시예에 따르면, 멀티-파티션 환경에서, 각각의 파티션은 리소스 상황에 맵핑될 수 있고, 쓰레드들은 파티션을 위해 작업을 실행할 때 적절한 리소스 상황에 결합될 것이다. 이것은 리소스, 판독된 보유된-힙, 소비가, 해당하는 그 리소스 상황에 대해 올바르게 고려되게 할 수 있다. JDK로부터의 보유된 힙 콜백(heap retained callback)을 통해, RCM 프레임워크는 파티션에 의한 보유된 힙의 양을 측정할 수 있다. 앞에서 논의된 바와 같이, 높은 정확도를 갖는 통지들은 보유된 힙 리소스 타입에 대해 RCM 정책을 실시하는데 사용될 수 있다. 통지, 저속화, 멈춤 등과 같은 의지 동작들은 JDK로부터의 단일 통지에 근거하여 호출될 필요가 없다. 하지만, 이러한 동작들은 JDK에 대한 보유된 힙 통지들의 감쇄되는 평균이 임의의 기간 동안 일관적으로 그 구성된 임계치보다 클 때 호출될 수 있다. 디폴트 기간(default time period)을 설정하는 것이 가능하다. 예를 들어, 이러한 메커니즘에 근거하여 동작을 트리거하는데 사용되는 디폴트 기간은 100초이다. 일반 폴링 메커니즘 기반의 알고리즘이, 보유된 힙 및 CPU 이용 리소스 타입들 모두에 의해 사용될 수 있고, 이것은 또한 스파이크들(산발적인 소비의 급상승/하강들)을 회피/무시하는데 도움을 준다.

[0159] 본 발명의 일 실시예에 따르면, 공정한-배당 정책은 보유된 힙 리소스 타입에 대해 지원되며, 이것은 서버 내에서 모든 파티션들의 공정한-배당 값들에 근거하여 작동한다.

[0160] **정책 평가, 의지 동작들/통지들(Policy Evaluation, Recourse Actions/Notifications)**

[0161] 본 발명의 일 실시예에 따르면, 파티션(예를 들어, 멀티-파티션 환경 내의 파티션)은 리소스-매니저와 관련될 수 있다. 임의의 파티션과 어떠한 리소스-매니저도 관련되어 있지 않은 경우, 해당하는 그 파티션은 임의의 리소스 소비 관리 정책들에 영향을 받지 않으며 이러한 파티션에 의한 리소스들의 사용은 제약을 받지 않는다.

[0162] 본 발명의 일 실시예에 따르면, 임의의 파티션에 대해 정의된 각각의 리소스-매니저에 대해서, RCM 프레임워크는 리소스Id(resourceId)를 나타내는 적절한 리소스속성(ResourceAttribute)에 대해 JSR-284 리소스도메인(ResourceDomain)을 인스턴스화(instantiate)할 수 있다. JSR 284 제약들 및 통지들이, 사용자/시스템 관리자에 의해 특정된 트리거에 근거하여 이러한 리소스도메인(ResourceDomain)에 대해 등록된다.

[0163] 본 발명의 일 실시예에 따르면, 시스템 관리자는 리소스의 사용을 통제하기 위해 하나 이상의 트리거들을 특정할 수 있다. 각각의 트리거는 명칭(name), 값 쌍(value pair) 및 동작(action)을 포함할 수 있다. 트리거는 리소스 소비 레벨들이 시스템 관리자에 의해 설정된 값에 도달하는 경우 동작의 과정(course)에 영향을 미치도록 시스템 관리자에 의해 특정될 수 있다. WLS RCM 프레임워크는 사용이 그 주어진 값에 도달하면 관련된 동작을 실행한다.

[0164] 본 발명의 일 실시예에 따르면, 동작 엘리먼트를 통해 특정되는 의지 동작 타입은 다음과 같은 네 개의 타입들, 통지, 저속화, 불이행, 멈춤 중 하나 일 수 있다.

[0165] 본 발명의 일 실시예에 따르면, 시스템 관리자에 의해 특정되는 의지 동작 타입을 실현하도록 WLS RCM 프레임워크에 의해 실행되는 활동들의 특정 세트는, 사용자에 의해 정의될 수 있거나, 또는 사전에-존재하는 매트릭스(matrix)로부터 선택될 수 있다.

[0166] 예를 들어, 본 발명의 일 실시예에 따르면, 시스템 관리자는 파티션 P1이 단지 2000개의 파일-기술자(file-descriptor)들을 사용하기를 원할 수 있다. 사용이 특정화 값(specify value)에 도달하면, 시스템 관리자는 해당하는 그 파티션에 의한 더 이상의 파일 기술자들의 사용을 막고자 한다. 이러한 트리거는 아래와 같이 보일 수 있다.

[0167] <trigger>

[0168] <value>2000</value>

[0169] <action>fail</action>

[0170] </trigger>

[0171] 특정된 불이행 동작이 파일 오픈 요청에 대한 IO제외(IOException)의 발생에 맵핑될 수 있다. WLS RCM 프레임워크는 파티션 P1에 대한 파일_오픈 리소스타입(FILE_OPEN ResourceType)을 나타내는 리소스속성(ResourceAttribute)에 대해 리소스도메인(ResourceDomain)에서 제약을 등록함으로써 파일 오픈 요청을 인터셉트할 수 있다.

[0172] 본 발명의 일 실시예에 따르면, 시스템에 의해 수행되는 의지 동작들 중 일부는 소비의 동작 동안 일어날 수 있다(예를 들어, 오픈 파일 기술자(Open File Descriptor)들에 대해 설정된 불이행 트리거(FAIL trigger)는 결과적으로 새로운 파일의 생성 동안 IO제외(IOException)가 발생되게 함). 추가적으로, 현재의 소비 요청 이후에 일부 동작들이 수행될 수 있다(예를 들어, CPU 및 힙 사용에 관한 통지들은 비동시적(asynchronous)이고, 이른바, 이러한 리소스들에 근거하는 트리거들을 위한 "멈춤(shutdown)" 의지 동작들(결과적으로 관련 파티션을 멈추게 하는 것)이 비동시적으로 수행됨).

[0173] 본 발명의 일 실시예에 따르면, 동시적 동작들은 JSR-284 제약을 통해 실현될 수 있고, 비동시적 동작들은 JSR-284 통지를 통해 실현될 수 있다.

[0174] **공정한-배당 정책 구현(Fair-Share Policy Implementation)**

[0175] 본 발명의 일 실시예에 따르면, 멀티-테넌트 애플리케이션 서버 환경 내에서의 상이한 리소스 소비자들(예를 들어, 멀티-파티션 환경 내에서의 상이한 파티션들)에 대한 RM-인에이블된 공유된 리소스들의 "공정한(fair)" 할당을 시스템 관리자들에게 정성적으로(qualitatively) 보증하고 정량적으로(quantitatively) 보장하는 것이 바람직하다.

[0176] 본 발명의 일 실시예에 따르면, 공정한 배당 값(fair share value)을 동일한 값으로서 갖는 두 개의 리소스 도메인들에게 리소스들을 할당하는 것은, 만약 이러한 리소스 할당이 불평등/불균등하다면, 혹은 달리 말해서 하나의 도메인의 소비자들 중 일부를 향해 다른 것들과 비교하여 편향된다면, 불공정한(unfair) 것으로 고려된다. 두 개의 리소스 도메인들은 변하는 리소스 요청 패턴들을 가질 수 있고, 순간순간 항상 공정한 배당을 보장하는 것은 어려울 수 있다. 하지만, 이들의 구성된 공정한 배당의 비율에서 리소스 할당을 유지하는 것이 여전히 요구되는데, 특히 리소스 도메인들이 모두 임의의 리소스에 대해 경쟁/경합하고 있을 때(해당하는 그 리소스에 대한 시스템의 최대 한계치를 향해 파티션들에 걸쳐 리소스 사용을 푸시함) 이러한 것이 요구된다.

[0177] 본 발명의 일 실시예에 따르면, 리소스 할당에서의 공정성은 개개의 리소스 도메인들로부터의 리소스 소비 요청들에서의 변화를 고려하기 위해 임의의 기간에 걸쳐 계산될 수 있다. 공정성이 시간 경과에 따라 컴퓨팅될 수 있고 실시될 수 있기 때문에, 공정성이 시간의 특정 윈도우(window)에서 배분의 균등(equality)(공정한-배당들이 동일한 경우임)을 반드시 시사(imply)하지는 않는다. 특정 리소스 소비자가 해당 윈도우 내에서 리소스들에 관한 그들의 배당을 사용하지 않았음을 인식하는 경우, 공정한-배당 정책 구현은 해당하는 그 소비자에 대한 특정된 공정한-배당보다 더 큰 리소스들의 배당을 일시적으로 할당할 수 있다. 하지만, 시간 경과에 따라, 할당들은 사용자에 의해 특정된 공정한-배당들에 맞도록 조정될 수 있다.

[0178] 공정한-배당 정책이 일시적으로 과도하게-할당(over-allocate)할 수 있는 정도는 또한 리소스의 타입에 근거할 수 있다. 예를 들어, CPU 이용과 같은 "한정되지 않은(unbounded)" 리소스의 경우에, 공정한-배당 구현은 자신들의 배당에 관해 뒤떨어지고 있는 소비자에게 리소스들을 제공하기 위해서 최근에 매우 능동적이었던 리소스 소비자에 대한 보급을 완전히 차단할 수 있다. 하지만, 힙 할당들/보유된 힙과 같은 "한정된(bounded)" 그리고 "처분가능한(disposable)" 그리고 "비-취소가능한(non-revocable)" 리소스의 경우에, 공정한-배당 정책 구현은 자신들의 배당에 관해 뒤떨어지고 있는 리소스 소비자들로부터의 모든 힙 할당 요청들을 허용할 수 없다(왜냐하

면 임의의 부여된 할당들은 이후에 취소될 수 없기 때문).

[0179] 본 발명의 일 실시예에 따르면, 공정한-배당 정책은 시스템 관리자에게 다음과 같은 보증을 제공할 수 있다. 리소스에 대한 "경합(contention)"이 존재하는 경우, 그리고 "임의의 기간에 걸쳐(over a period of time)" 두 개의 리소스 도메인들에 의한 "균일한 부하(uniform load)"가 존재하는 경우, 두 개의 도메인들에 할당되는 리소스들의 배당은, 이러한 두 개의 도메인들에 대해서 시스템 관리자에 의해 구성된 공정한 배당에 "개략적으로(roughly)" 따른다.

[0180] 본 발명의 일 실시예에 따르면, 공정한 배당 정책은 CPU 이용 및 보유된 힙 리소스 타입들에 대해 지원된다. 이러한 리소스 타입들 모두에 대해, JDK로부터의 통지들은 사후-소비(post-consumption)로 제공될 수 있다. 이러한 것이 일어나는 경우, 공정한 분배 정책은 리소스 소비를 위한 요청 시에 정확하게 적용될 수 없다. 보유된 힙의 경우에, JDK로부터의 통지는 가비지 수집(Garbage Collection, GC) 싸이클들에 링크(link)될 수 있고 그리고 힙 소비 요청이 행해진 시점보다 더 이후에 일어날 수 있다. CPU 이용의 경우에, 그 이용은 시간의 고정된 윈도우 이후 결정될 수 있다.

[0181] 본 발명의 일 실시예에 따르면, 리소스 소비를 위한 요청은 파티션의 작업 매니저에 의해서 파티션에 대해 행해지는 작업에 간접적으로 링크될 수 있다. 따라서, 만약 파티션에 대해 실행되는 작업 인스턴스들이 파티션의 리소스 소비에 근거하여 제어될 수 있다면, 해당하는 그 리소스에 대한 공정한 배당이 여전히 보증될 수 있다.

[0182] 본 발명의 일 실시예에 따르면, 보유된 힙의 경우에, 더 일찍 할당된 어떠한 힙도 시스템에 의해 강제로 취소될 수 없기 때문에, 공정한 배당 정책은 파티션의 작업 매니저의 공정한-배당을 제어함으로써 힙의 할당을 제어한다.

[0183] **RCM 구성(RCM Configuration)**

[0184] 본 발명의 일 실시예에 따르면, 리소스 소비 매니저(Resource Consumption Manager, RCM) 정책들 및 의지 동작들은 시스템 관리자들과 같은 관리자들에 의해 구성될 수 있다. 시스템 관리자들은 또한 재사용가능한 정의(reusable definition)들을 생성할 수 있고, 그리고 임의의 파티션에 대해 커스터마이징(customizing)된 정책들을 생성할 수 있다.

[0185] 본 발명의 일 실시예에 따르면, 시스템 관리자는 리소스 매니저 정의(Resource Manager definition)를 생성할 수 있고, 그 다음에 이것은 다수의 파티션들에 걸쳐 재사용될 수 있다. 예를 들어, SaaS 사용의 경우에, 도메인을 관리하는 시스템 관리자는 그들이 고객의 특정 "클래스(class)"(예를 들어, 브론즈(Bronze), 실버(Silver), 골드(Gold)의 클래스들)에 대해 생성한 모든 새로운 파티션에 그들이 적용하고자 하는 RCM 정책들의 세트를 가질 수 있다. 예컨대, 브론즈 고객의 파티션은 힙 및 CPU 사용에 관해 특정의 관리되는 리소스 정책들(예를 들어, 힙은 2GB로 제한됨)을 가질 수 있다. 새로운 브론즈 고객이 온-보드(on-board)되는 경우, 그리고 파티션 P2가 이들을 위해 생성되는 경우, 시스템 관리자가 도메인 레벨에서 브론즈 리소스 관리 정책을 생성하는 것이 가능하고, 그리고 이것을 브론즈 파티션의 생성 동안 가리키는 것(point)이 가능하다. 그 다음에, 브론즈 리소스 매니저 내에 포함되는 모든 정책 구성은 P2의 리소스 관리 정책에 적용된다. 리소스 매니저가 추가되는 각각의 파티션은 해당하는 그 구성에서 정의된 관리되는 리소스 정책들의 "카피(copy)"를 얻는다. 이에 따라, 예를 들어, 브론즈 리소스 매니저는 P2 파티션 및 P3 파티션에 적용된다. P2와 P3 양쪽 각각에게는 2GB 힙 사용이 허용된다.

[0186] 본 발명의 일 실시예에 따르면, 시스템 관리자는 임의의 파티션에 대해 커스터마이징된 정책들을 생성할 수 있다. 예를 들어, 통합 사용(consolidation use)의 경우에, 시스템 관리자는 특정 부서를 위해 생성된 파티션에 대해 고유한 리소스 소비 관리 정책들을 특정할 수 있다. 해당하는 그 파티션에 대해서, 파티션에 국한된 범위를 갖는 새로운 리소스 관리 정책을 생성하는 것이 가능하다. 예컨대, 시스템 관리자는 회사의 CEO를 위해서 파티션 CEO를 생성한다. 그 다음에, 시스템 관리자는 또한, (시스템 내에서 정의된 다른 파티션들과는 구분되는) 해당하는 그 파티션에게만 적용가능한 특정의 관리되는 리소스 정책들(예를 들어, 해당하는 그 파티션에게 상대적으로 높은 공정한-배당을 주는 것)을 정의할 수 있다.

[0187] 도 10은 본 발명의 일 실시예에 따른, 애플리케이션 서버 환경에서 리소스 격리 및 소비를 예시한다. 도 10은 도메인(610)을 포함하는 애플리케이션 서버 환경(600)을 도시한다. 도메인은 파티션 A(640) 및 파티션 B(650)를 각각 포함할 수 있다. 파티션 A 및 파티션 B는 리소스 그룹(642) 및 리소스 그룹(652)을 각각 포함하는데, 이러한 리소스 그룹들은 가상 타겟 A(645) 및 가상 타겟 B(655)와 각각 관련된다. 도 6에서 제시되는 바와 같이, 파티션 A 혹은 파티션 B는 또한 공유된 리소스들(660)에 액세스할 수 있는데, 예를 들어, 파티션들이 임의의 공유

된 리소스의 사용을 요구하는 애플리케이션를 실행할 때 이러한 공유된 리소스들(660)에 액세스할 수 있다. 공유된 리소스들(660)은 CPU, 힙, 네트워크 및 파일과 같은 JDK 리소스들을 포함할 수 있지만, 이러한 것으로만 한정되는 것은 아니다.

[0188] 앞에서 기술된 바와 같이, 시스템 관리자는 파티션 특정 리소스 소비 관리 정책들(1005 및 1015)을 각각 정의할 수 있다. 이러한 파티션 특정 리소스 소비 관리 정책들은, 파티션 특정 리소스 예약들(1006 및 1016)을 각각 포함하도록 구성될 수 있고, 그리고 파티션 특정 리소스 예약들(1007 및 1017)을 각각 포함하도록 구성될 수 있고, 그리고 파티션 특정 리소스 통지들(1008 및 1018)을 각각 포함하도록 구성될 수 있다.

[0189] 본 발명의 일 실시예에 따르면, 파티션 특정 리소스 소비 관리 정책들은, 리소스 관리 정책에 의해 설정되는 구성된 조건들이 그 공유된 리소스들에 관해 파티션 A 혹은 파티션 B에 의해 각각 충족될 때 동작들 혹은 태스크들을 수행하도록, 시스템 관리자 또는 적절한 허가를 갖는 또 하나의 다른 것에 의해 구성될 수 있다. 이러한 동작들 혹은 태스크들은 제약들 혹은 통지들을 포함할 수 있지만, 이러한 것으로만 한정되는 것은 아니다. 제약들의 예들은 파티션에 의한 공유된 리소스들의 사용을 느리게 하는 저속화 혹은 정지시키는 것을 포함한다. 통지들의 예들은 로깅, 또는 관리자에게 통지들을 제공하는 것을 포함하지만, 이러한 것으로만 한정되는 것은 아니다.

[0190] 아래에 있는 것은 임의의 특정된 파티션에 대한 리소스 관리 정책에 대해서 정의한 예시적 정의의 예이다.

[0191] <domain>

[0192] ...

[0193] <!--Define RCM Configuration -->

[0194] <resource-management>

[0195] <resource-manager>

[0196] <name>Gold</name>

[0197] <file-open>

[0198] <trigger>

[0199] <name>Gold2000</name>

[0200] <value>2000</value><!-- in units-->

[0201] <action>shutdown</action>

[0202] </trigger>

[0203] <trigger>

[0204] <name>Gold1700</name>

[0205] <value>1700</value>

[0206] <action>slow</action>

[0207] </trigger>

[0208] <trigger>

[0209] <name>Gold1500</name>

[0210] <value>1500</value>

[0211] <action>notify</action>

[0212] </trigger>

[0213] </file-open>

[0214] <heap-retained>

[0215] <trigger>
[0216] <name>Gold2GB</name>
[0217] <value>2097152</value>
[0218] <action>shutdown</action>
[0219] </trigger>
[0220] <fair-share-constraint>
[0221] <name>FS-GoldShare</name>
[0222] <value>60</value>
[0223] </fair-share-constraint>
[0224] </heap-retained>
[0225] </resource-manager>
[0226] <resource-manager>
[0227] <name>Silver</name>
[0228] <file-open>
[0229] <trigger>
[0230] <name>Silver1000</name>
[0231] <value>1000</value><!-- in units-->
[0232] <action>shutdown</action>
[0233] </trigger>
[0234] <trigger>
[0235] <name>Silver700</name>
[0236] <value>700</value>
[0237] <action>slow</action>
[0238] </trigger>
[0239] <trigger>
[0240] <name>Silver500</name>
[0241] <value>500</value>
[0242] <action>notify</action>
[0243] </trigger>
[0244] </file-open>
[0245] </resource-manager>
[0246] </resource-management>
[0247] <partition>
[0248] <name>Partition-0</name>
[0249] <resource-group>
[0250] <name>ResourceTemplate-0_group</name>

[0251] <resource-group-template>ResourceTemplate-0</resource-

[0252] group-template>

[0253] </resource-group>

[0254] ...

[0255] <partition-id>1741ad19-8ca7-4339-b6d3-78e56d8a5858</partition-

[0256] id>

[0257] <!-- RCM Managers are then targeted to Partitions during

[0258] partition creation time or later by system administrators -->

[0259] <resource-manager-ref>Gold</resource-manager-ref>

[0260] ...

[0261] </partition>

[0262] ...

[0263] </domain>

[0264] 본 발명의 일 실시예에 따르면, 모든 RCM 엘리먼트들의 동적 재구성이 지원될 수 있다. 특정 상황들에서, 리소스 소비 관리 정책들의 재구성을 위해 파티션 및 서버의 어떠한 재개시도 요구되지 않는다.

[0265] **리소스 격리 및 소비(Resource Isolation and Consumption)**

[0266] 이제 도 11을 참조하면, 도 11은 본 발명의 일 실시예에 따른, 애플리케이션 서버 환경에서 리소스 격리 및 소비를 위한 방법에 대한 흐름도를 도시한다. 예시적인 방법은 단계(1110)에서 시작할 수 있고, 이러한 단계(1110)는, 하나 이상의 컴퓨터들에서, 복수의 리소스들 및 하나 이상의 파티션들을 제공하는 단계이며, 여기서, 하나 이상의 컴퓨터들은 이러한 하나 이상의 컴퓨터들 상에서 실행되는 애플리케이션 서버 환경을 포함하고, 복수의 리소스들은 애플리케이션 서버 환경 내에서 사용될 수 있는 것이고, 각각의 파티션은 임의의 도메인의 관리 및 런타임 하위분할구역을 제공한다. 본 방법은 단계(1120)에서 계속될 수 있으며, 이러한 단계(1120)는, 각각의 파티션에 의한 복수의 리소스들의 사용을 모니터링하도록 리소스 소비 관리 모듈을 구성하는 단계이고, 여기서, 리소스 소비 관리 모듈은, 리소스 예약들, 리소스 제약들, 및 리소스 통지들로 이루어진 그룹의 적어도 하나의 요소를 포함한다.

[0267] 본 발명은, 본 개시내용의 가르침들에 따라 프로그래밍되는 하나 이상의 프로세서들, 메모리, 및/또는 컴퓨터 판독가능 저장 매체들을 포함하는 하나 이상의 종래의 범용 혹은 특화된 디지털 컴퓨터, 컴퓨팅 디바이스, 기계, 혹은 마이크로프로세서를 사용하여 편리하게 구현될 수 있다. 소프트웨어 기술분야에서 숙련된 자들에게 명백하게 될 것으로서, 본 개시내용의 가르침들에 근거하여 적절한 소프트웨어 코딩이 숙련된 프로그래머들에 의해 용이하게 준비될 수 있다.

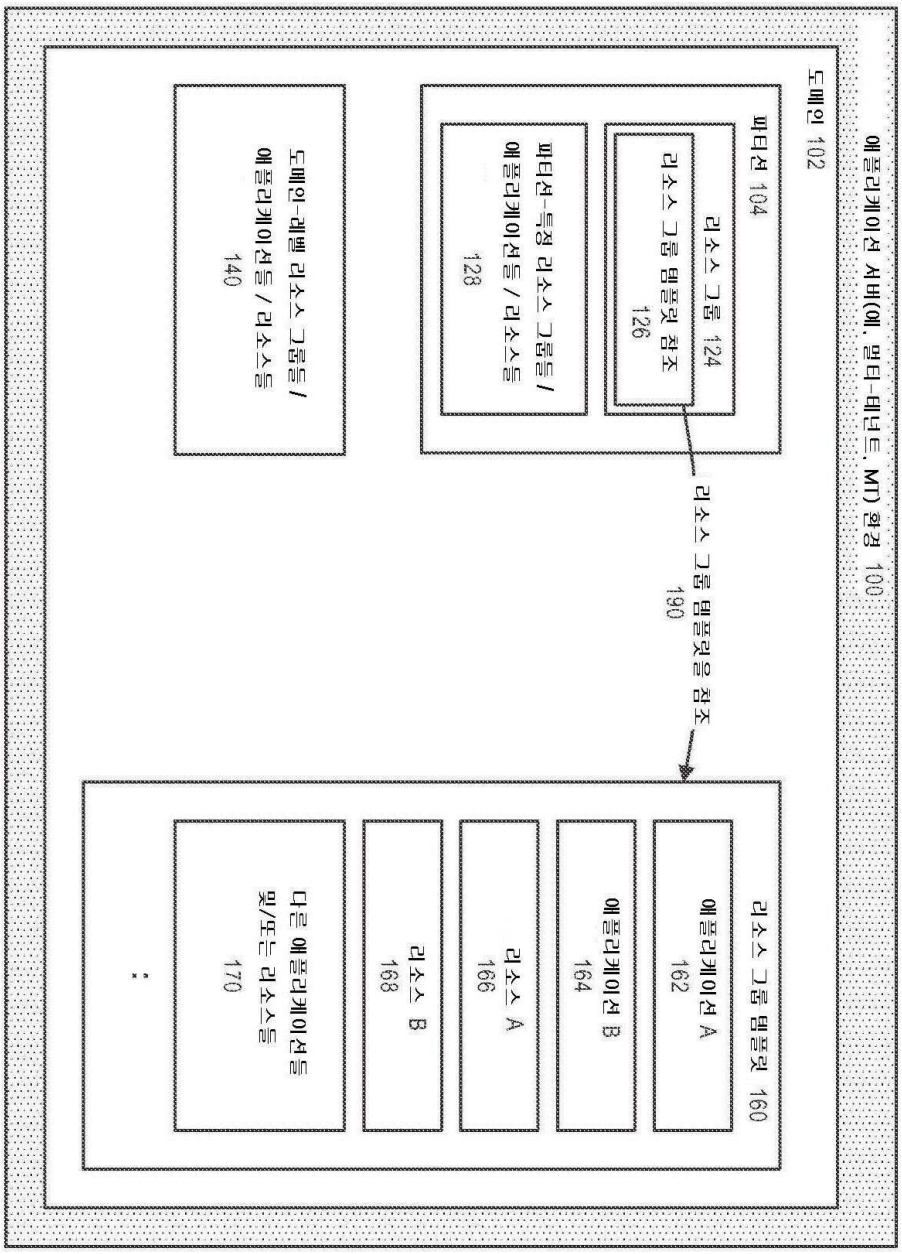
[0268] 일부 실시예들에서, 본 발명은 본 발명의 프로세스들 중 임의의 프로세스를 수행하도록 컴퓨터를 프로그래밍하는데 사용될 수 있는 명령들이 저장되어 있는 비-일시적 저장 매체 혹은 컴퓨터 판독가능 매체(매체들)인 컴퓨터 프로그램 제품을 포함한다. 저장 매체는, 임의 타입의 디스크(여기에는 플로피 디스크들, 광학 디스크들, DVD, CD-ROM들, 마이크로드라이브, 및 광자기 디스크들이 포함됨), ROM들, RAM들, EPROM들, EEPROM들, DRAM들, VRAM들, 플래시 메모리 디바이스들, 자기 혹은 광학 카드들, 나노시스템들(여기에는 분자 메모리 IC들이 포함됨), 또는 명령들 및/또는 데이터를 저장하는데 적합한 임의 타입의 매체들 혹은 디바이스를 포함할 수 있지만, 이러한 것으로만 한정되는 것은 아니다.

[0269] 본 발명의 앞서의 설명은 예시 및 설명 목적으로 제공된 것이다. 이것은 정확히 그 개시되는 형태들로만 본 발명을 한정하려고 의도된 것이 아니며 또한 본 발명의 모든 실시예들을 나타내도록 의도된 것이 아니다. 많은 수정물들 및 변형물들이 본 발명의 기술분야에서 숙련된 실무자들에게는 명백하게 될 것이다. 본 발명의 실시예들은, 본 발명의 원리들 및 그 실제 응용을 가장 잘 설명하기 위해서 선택 및 기술된 것으로, 그럼으로써 본 발명의 기술분야에서 숙련된 다른 사람들로 하여금 다양한 실시예들에 대해 본 발명을 이해할 수 있게 하고 아울러 그 고려되는 특정 용도에 맞는 다양한 수정물들을 갖는 본 발명을 이해할 수 있게 하기 위해 선택 및 기술된 것

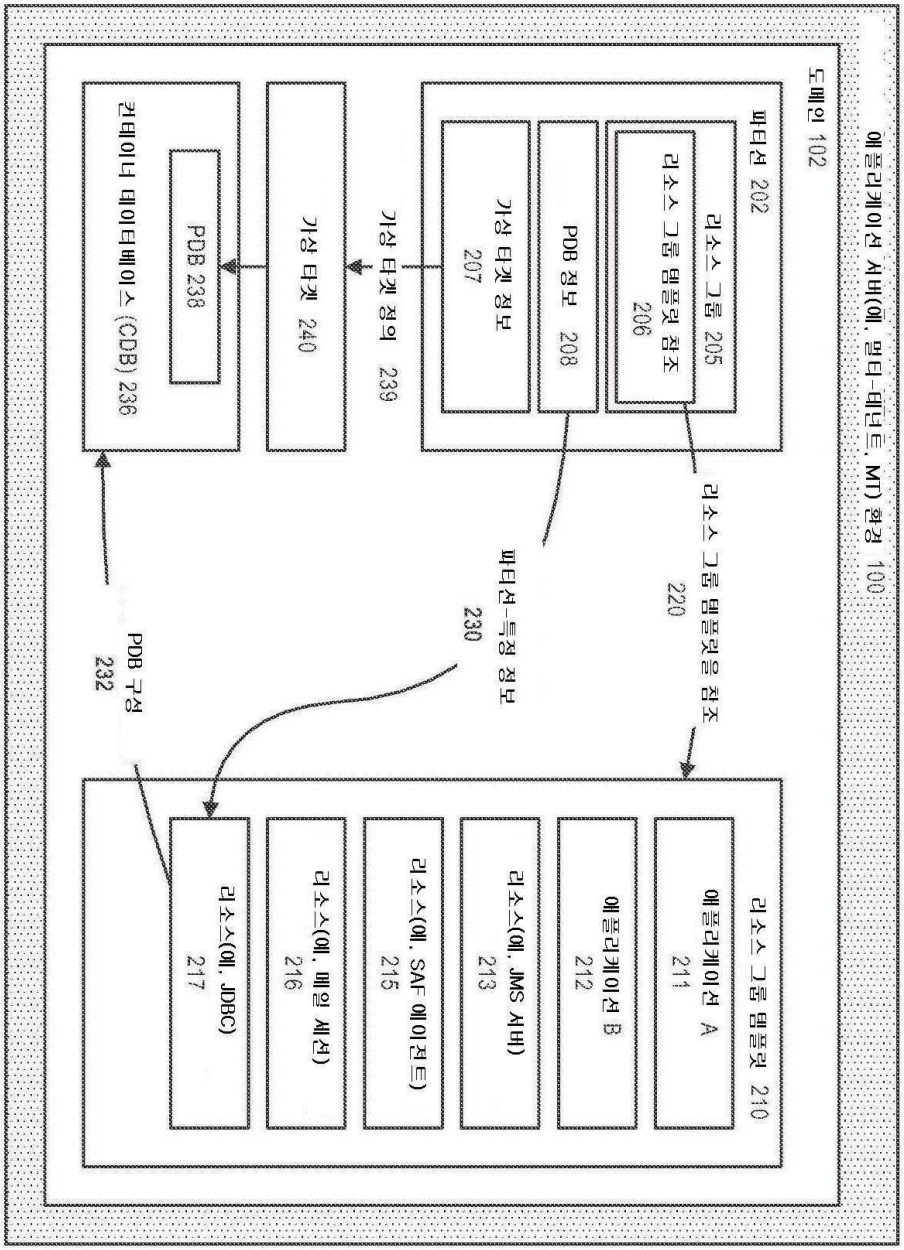
이다. 본 발명의 범위는 다음의 청구항들 및 이들의 등가물들에 의해 정의되도록 의도되었다.

도면

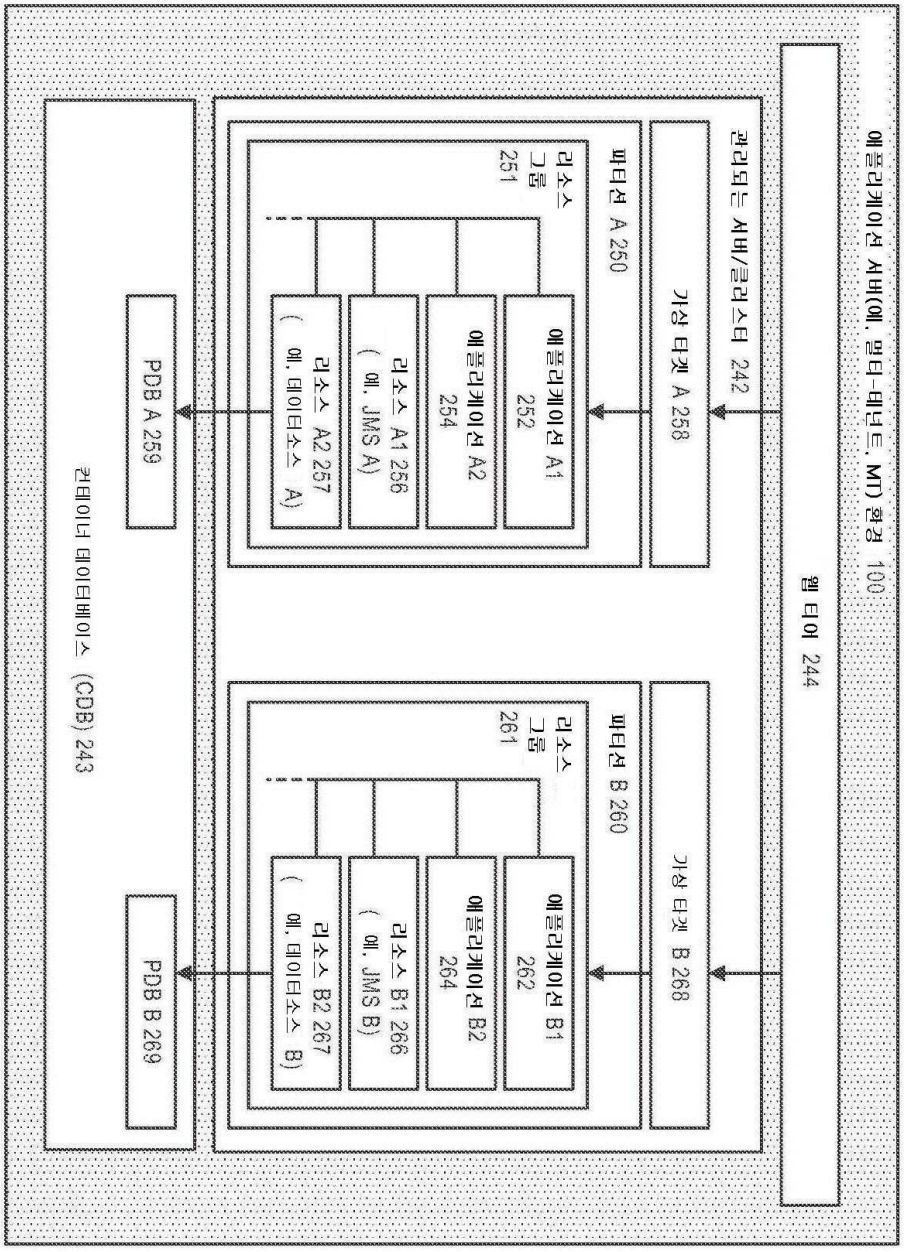
도면1



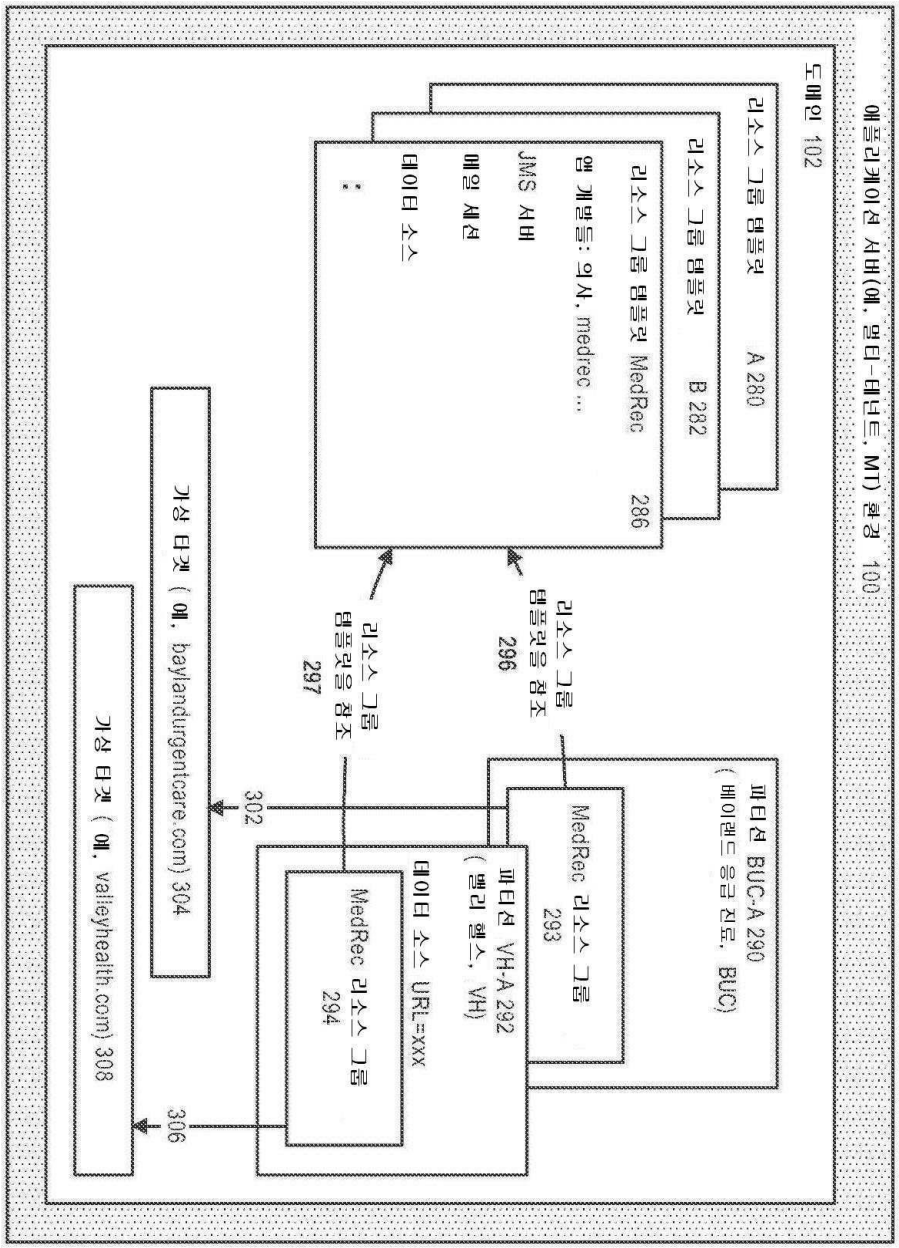
도면2


애플리케이션 서버(예, 멀티-테넌트, MT) 환경 100
도메인 102
파티션 202
리소스 그룹 205
리소스 그룹 템플릿 참조 206
PDB 정보 208
가상 타겟 정보 207
가상 타겟 정의 239
가상 타겟 240
PDB 238
컨테이너 데이터베이스 (CDB) 236
리소스 그룹 템플릿 210
애플리케이션 A 211
애플리케이션 B 212
리소스(예, JMS 서버) 213
리소스(예, SAF 에이전트) 215
리소스(예, 메일 세션) 216
리소스(예, JDBC) 217
리소스 그룹 템플릿을 참조 220
파티션-특정 정보 230
PDB 구성 232

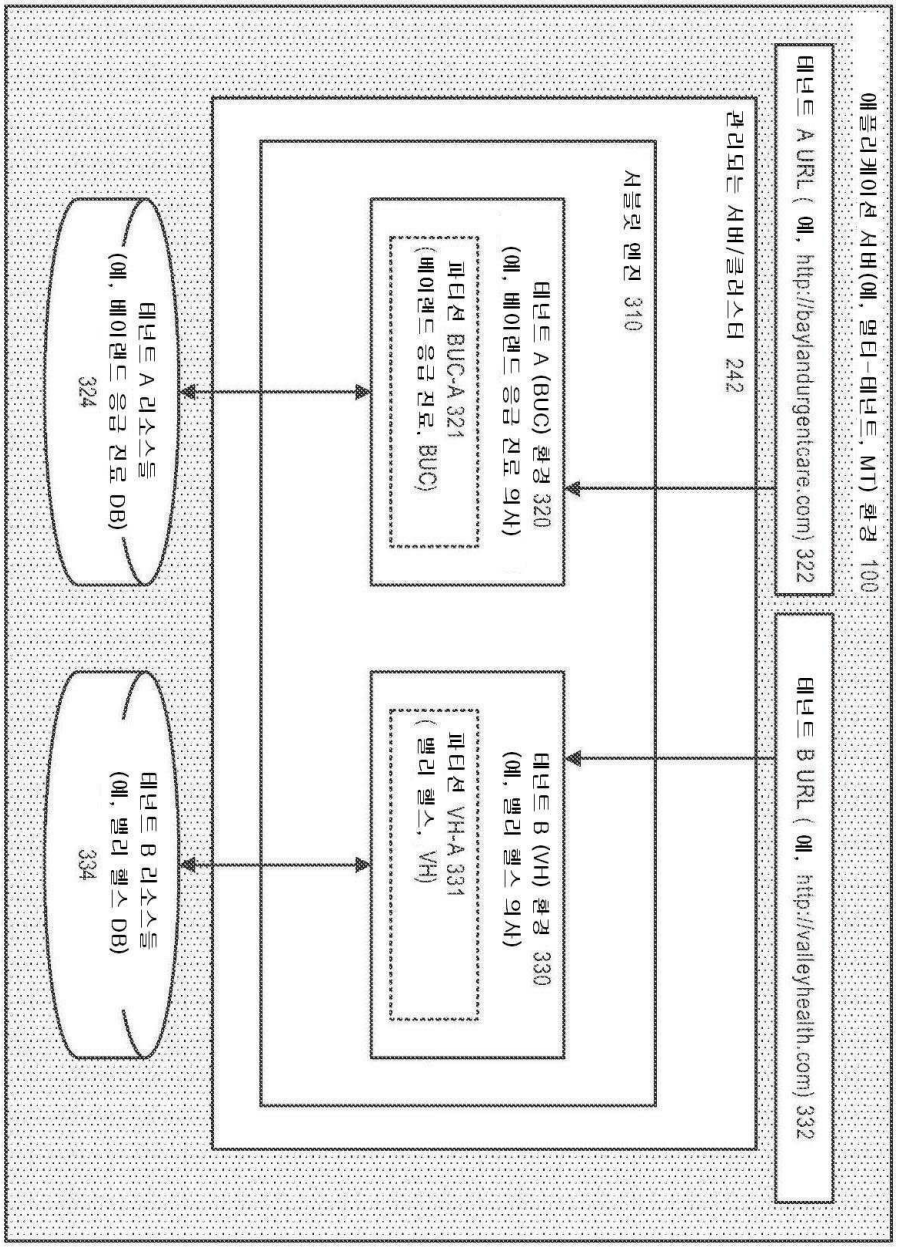
도면3


애플리케이션 서버(예, 멀티-테넌트, MT) 환경 100
웹 티어 244
관리되는 서버/클러스터 242
가상 타겟 A 258
파티션 A 250
리소스 그룹 251
애플리케이션 A1 252
애플리케이션 A2 254
리소스 A1 256 (예, JMS A)
리소스 A2 257 (예, 데이터소스 A)
PDB A 259
가상 타겟 B 268
파티션 B 260
리소스 그룹 261
애플리케이션 B1 262
애플리케이션 B2 264
리소스 B1 266 (예, JMS B)
리소스 B2 267 (예, 데이터소스 B)
PDB B 269
컨테이너 데이터베이스 (CDB) 243

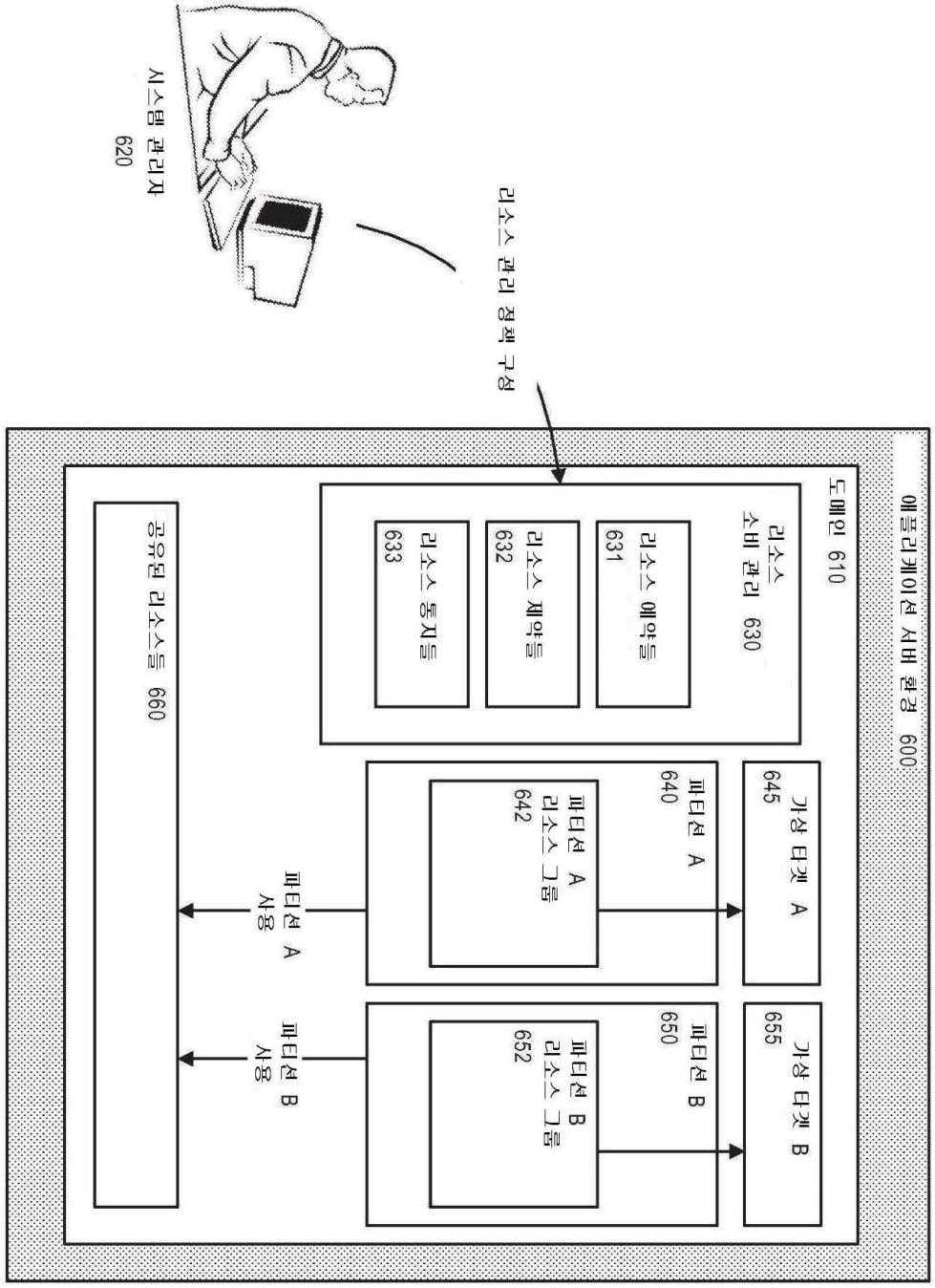
도면4


애플리케이션 서버(예, 멀티-테넌트, MT) 환경 100
도메인 102
리소스 그룹 템플릿 A 280
리소스 그룹 템플릿 B 282
리소스 그룹 템플릿 MedRec 286
앱 개발들: 의사, medrec ...
JMS 서버
메일 세션
데이터 소스
...
리소스 그룹 템플릿을 참조 297
리소스 그룹 템플릿을 참조 296
파티션 BUC-A 290
(베이렌드 응급 진료, BUC)
MedRec 리소스 그룹 293
파티션 VH-A 232
(밸리 헬스, VH)
데이터 소스 URL=xxx
MedRec 리소스 그룹 294
302
306
가상 타겟 (예, baylandurgentcare.com) 304
가상 타겟 (예, valleyhealth.com) 308

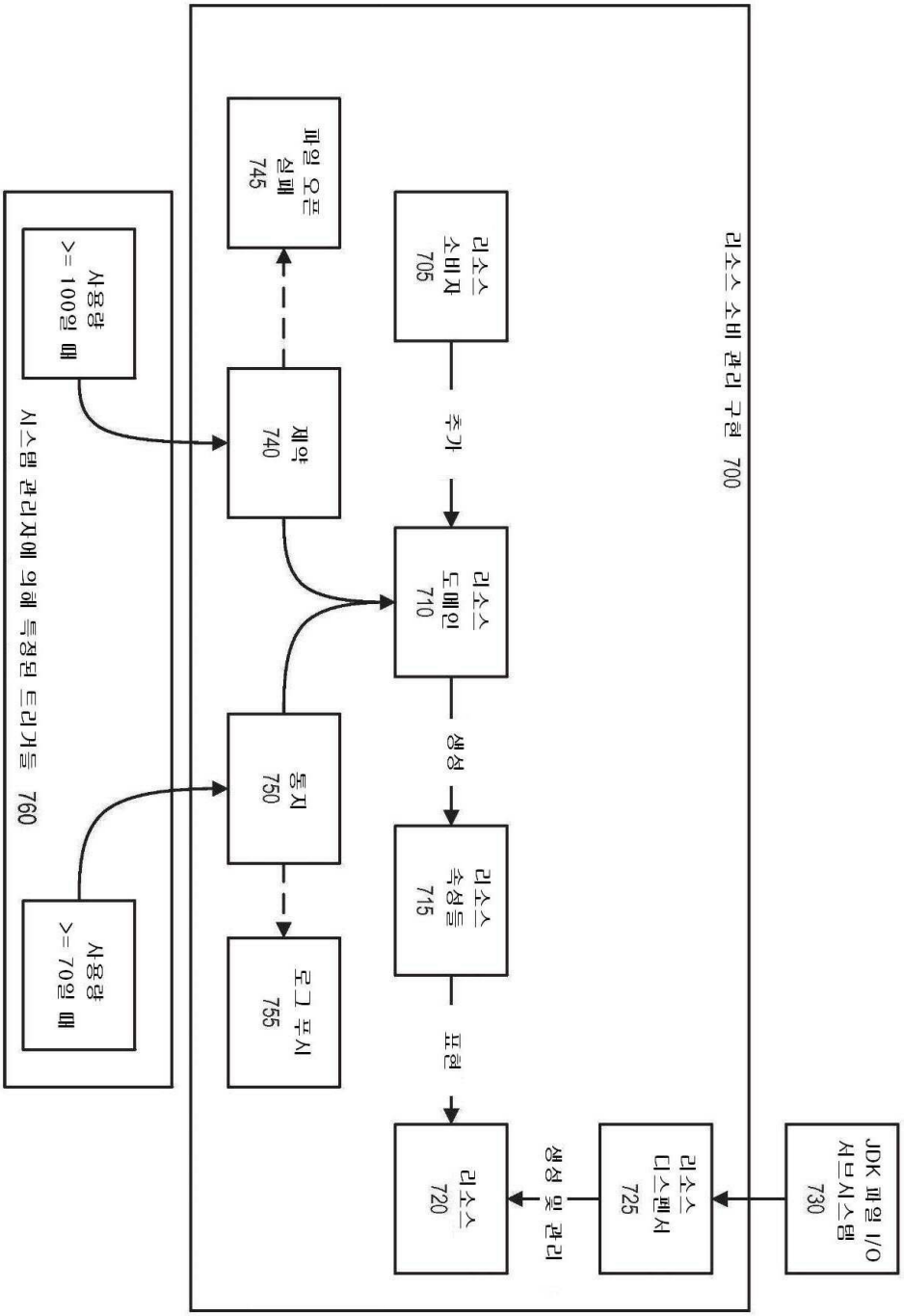
도면5


애플리케이션 서버(예, 멀티-테넌트, MT) 환경 100
테넌트 A URL (예, http://baylandurgentcare.com) 322
테넌트 B URL (예, http://valleyhealth.com) 332
관리되는 서버/클러스터 242
서블릿 엔진 310
테넌트 A (BUC) 환경 320
(예, 베이랜드 응급 진료 의사)
파티션 BUC-A 321
(베이랜드 응급 진료, BUC)
테넌트 B (VH) 환경 330
(예, 밸리 헬스 의사)
파티션 VH-A 331
(밸리 헬스, VH)
테넌트 A 리소스들
(예, 베이랜드 응급 진료 DB)
324
테넌트 B 리소스들
(예, 밸리 헬스 DB)
334

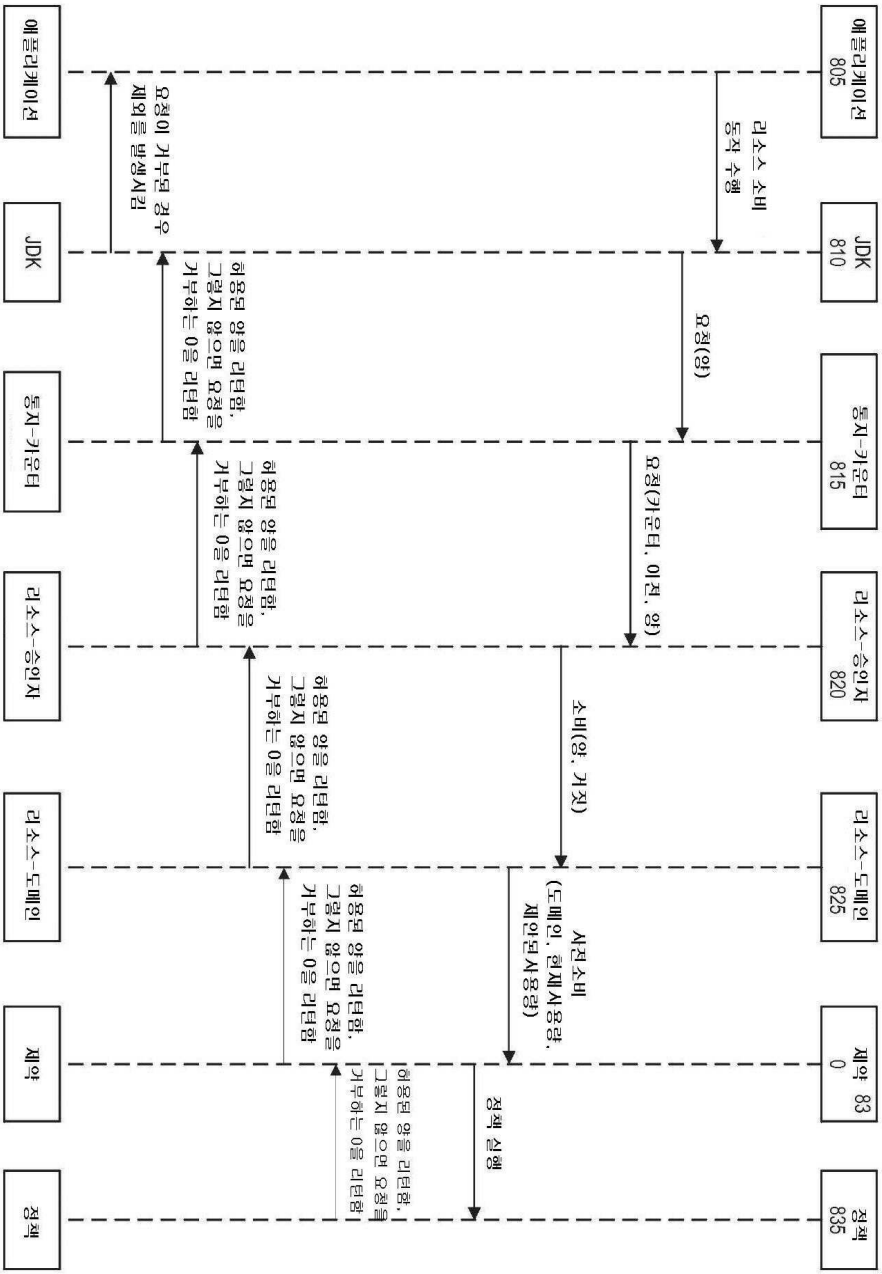
도면6


시스템 관리자
620
리소스 관리 정책 구성
애플리케이션 서버 환경 600
도메인 610
리소스 소비 관리 630
리소스 예약들 631
리소스 제약들 632
리소스 통지들 633
파티션 A 640
파티션 A 리소스 그룹 642
가상 타겟 A 645
파티션 B 650
파티션 B 리소스 그룹 652
가상 타겟 B 655
공유된 리소스들 660
파티션 A 사용
파티션 B 사용

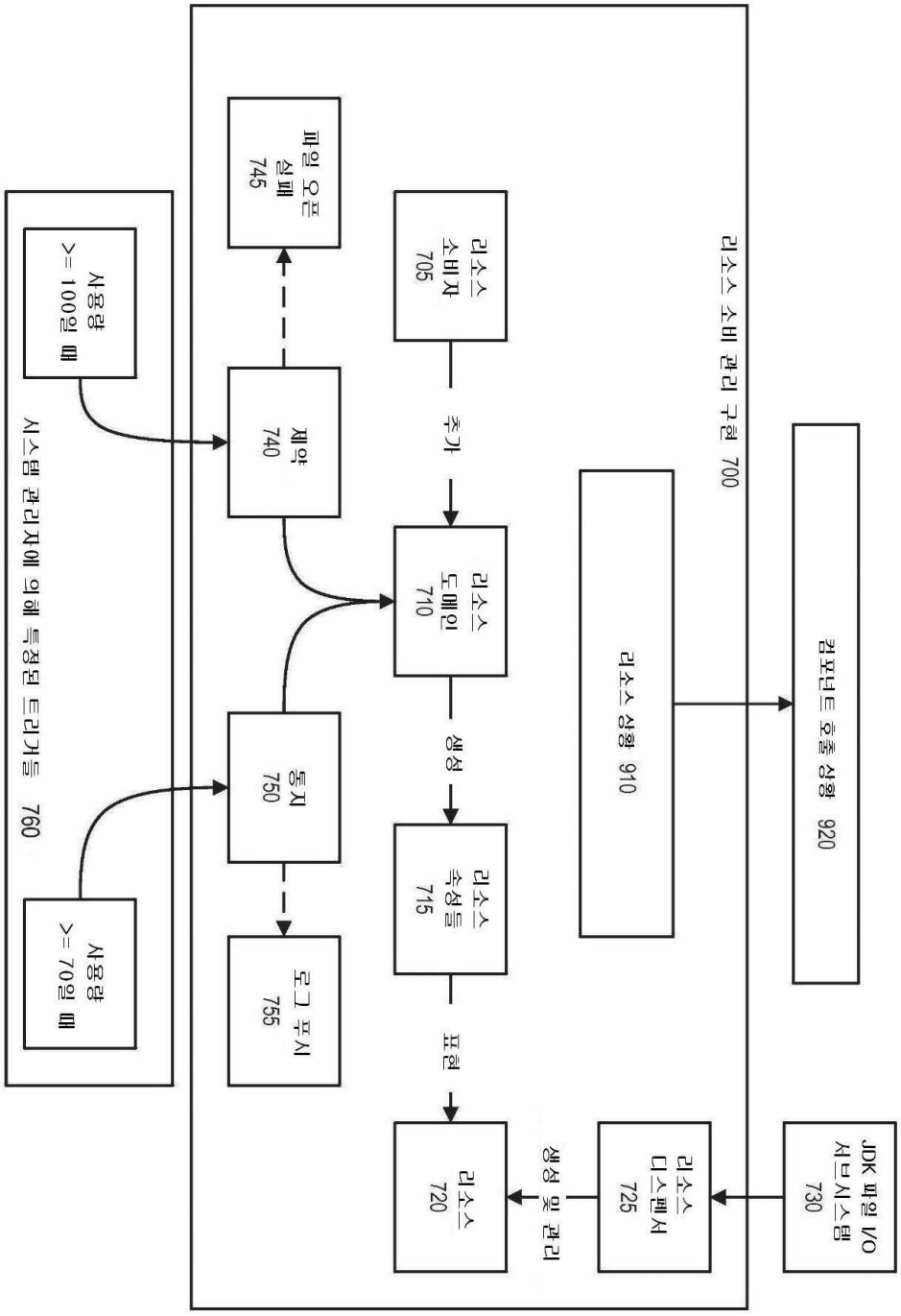
도면7


리소스 소비 관리 구현 700
리소스 소비자 705
추가
리소스 도메인 710
생성
리소스 속성들 715
표현
리소스 720
생성 및 관리
리소스 디스펜서 725
JDK 파일 I/O 서브시스템 730
파일 오픈 실패 745
제약 740
동지 750
로그 푸시 755
시스템 관리자에 의해 특정된 트리거들 760
사용량 >= 100일 때
사용량 >= 70일 때

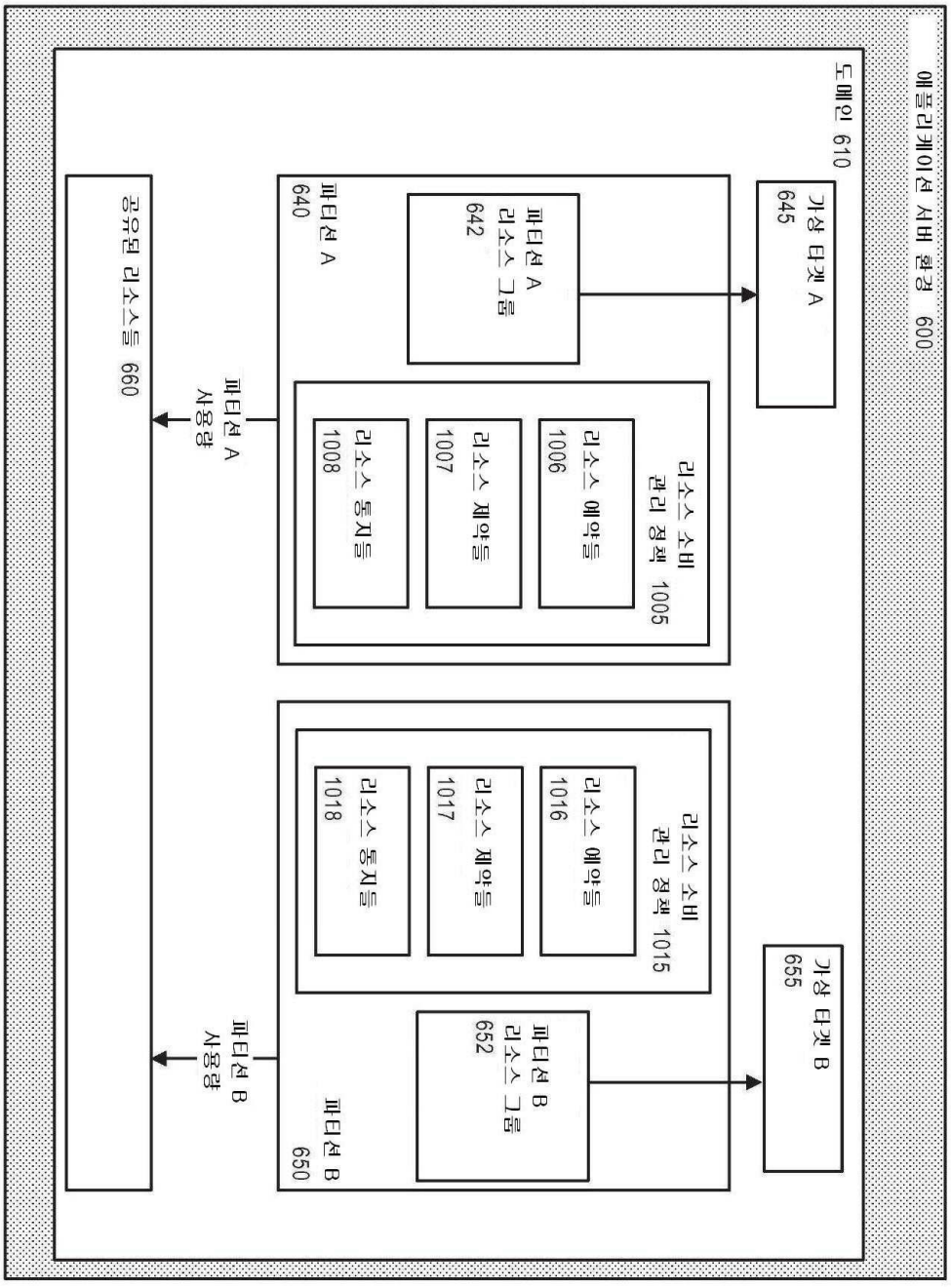
도면8


애플리케이션
805
JDK
810
통지-카운터
815
리소스-승인자
820
리소스-도메인
825
제약
0
정책
835
리소스 소비
동작 수행
요청(양)
요청(카운터, 이전, 양)
소비(양, 거짓)
사전 소비
(도메인, 현재시 용량,
제안된 사용량)
정책 실행
허용된 양을 리턴함,
그렇지 않으면 요청을
거부하는 0을 리턴함
허용된 양을 리턴함,
그렇지 않으면 요청을
거부하는 0을 리턴함
허용된 양을 리턴함,
그렇지 않으면 요청을
거부하는 0을 리턴함
허용된 양을 리턴함,
그렇지 않으면 요청을
거부하는 0을 리턴함
허용된 양을 리턴함,
그렇지 않으면 요청을
거부하는 0을 리턴함
요청이 거부된 경우
제외를 발생시킴
애플리케이션
JDK
통지-카운터
리소스-승인자
리소스-도메인
제약
정책

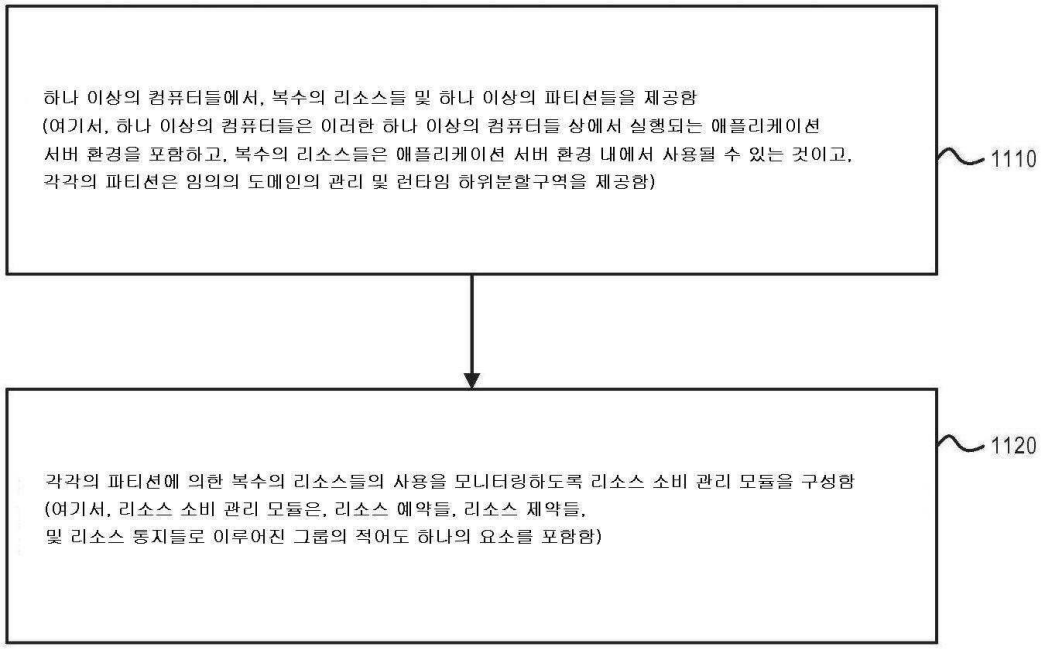
도면9


리소스 소비 관리 구현 700
리소스 소비자 705
추가
리소스 도메인 710
생성
리소스 속성들 715
표현
리소스 720
생성 및 관리
리소스 디스펜서 725
JDK 파일 I/O 서브시스템 730
리소스 상황 910
컴포넌트 호출 상황 920
파일 오픈 실패 745
제약 740
통지 750
로그 퓨시 755
시스템 관리자에 의해 특정된 트리거들 760
사용량 >= 100일 때
사용량 >= 70일 때

도면10


애플리케이션 서버 환경 600
도메인 610
가상 타겟 A
645
파티션 A
640
파티션 A
리소스 그룹
642
리소스 소비
관리 정책 1005
리소스 예약들
1006
리소스 제약들
1007
리소스 통지들
1008
파티션 A
사용량
공유된 리소스들 660
가상 타겟 B
655
파티션 B
650
파티션 B
리소스 그룹
652
리소스 소비
관리 정책 1015
리소스 예약들
1016
리소스 제약들
1017
리소스 통지들
1018
파티션 B
사용량

도면11


하나 이상의 컴퓨터들에서, 복수의 리소스들 및 하나 이상의 파티션들을 제공함
(여기서, 하나 이상의 컴퓨터들은 이러한 하나 이상의 컴퓨터들 상에서 실행되는 애플리케이션
서버 환경을 포함하고, 복수의 리소스들은 애플리케이션 서버 환경 내에서 사용될 수 있는 것이고,
각각의 파티션은 임의의 도메인의 관리 및 런타임 하위분할구역을 제공함)
1110
각각의 파티션에 의한 복수의 리소스들의 사용을 모니터링하도록 리소스 소비 관리 모듈을 구성함
(여기서, 리소스 소비 관리 모듈은, 리소스 예약들, 리소스 제약들,
및 리소스 통지들로 이루어진 그룹의 적어도 하나의 요소를 포함함)
1120