(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2013-0075377
(43) 공개일자 2013년07월05일

(51) 국제특허분류(Int. Cl.)
G06F 9/50 (2006.01)
(21) 출원번호 10-2011-0143728
(22) 출원일자 2011년12월27일
심사청구일자 2011년12월27일

(71) 출원인
서강대학교산학협력단
서울특별시 마포구 백범로 35 (신수동, 서강대학교)
(72) 발명자
박성용
서울특별시 송파구 문정2동 훼밀리아파트229동1103호
고영록
서울특별시 강남구 학동로4길 29, 202호 (논현동)
윤현준
경기도 수원시 장안구 화산로125번길 9, 120동 2403호 (천천동, 천천 푸르지오)
(74) 대리인
특허법인충현

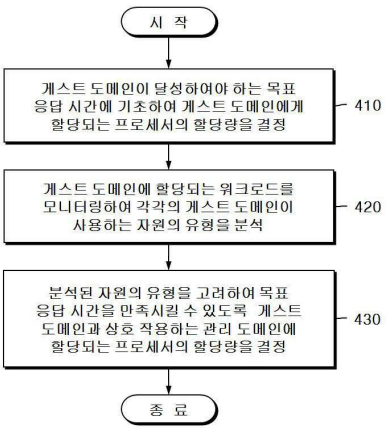
전체 청구항 수 : 총 17 항

(54) 발명의 명칭 **가상화 환경에서 프로세서를 할당하는 장치 및 방법**

(57) 요 약

본 발명은 가상화 환경에서 프로세서를 할당하는 장치, 방법 및 그 방법을 기록한 기록매체에 관한 것으로, 본 발명의 일 실시예에 따른 가상화 환경에서 프로세서를 할당하는 방법은, 게스트 도메인이 달성하여야 하는 목표 응답 시간에 기초하여 게스트 도메인에게 할당되는 프로세서의 할당량을 결정하고, 게스트 도메인에 할당되는 워크로드(workload)를 모니터링(monitoring)하여 각각의 게스트 도메인이 사용하는 자원의 유형을 분석하며, 분석된 자원의 유형을 고려하여 목표 응답 시간을 만족시킬 수 있도록 게스트 도메인과 상호 작용하는 관리 도메인에 할당되는 프로세서의 할당량을 결정한다.

대 표 도 - 도4



이 발명을 지원한 국가연구개발사업

과제고유번호	2010A0150036
부처명	지식경제부
연구사업명	대학 IT 연구센터 육성지원사업
연구과제명	(U-1) 소셜미디어 서비스를 위한 클라우드 플랫폼 및 응용 서비스 기술 개발
주관기관	건국대학교 산학협력단
연구기간	2011.01.01 ~ 2011.12.31

특허청구의 범위

청구항 1

적어도 하나 이상의 가상 머신을 제어하는 게스트 도메인(guest domain)과 상기 게스트 도메인을 제어하는 하나의 관리 도메인으로 구성된 가상화 환경에서 프로세서(processor)를 할당하는 방법에 있어서,

상기 게스트 도메인이 달성하여야 하는 목표 응답 시간에 기초하여 상기 게스트 도메인에게 할당되는 프로세서의 할당량을 결정하는 단계;

상기 게스트 도메인에 할당되는 워크로드(workload)를 모니터링(monitoring)하여 각각의 게스트 도메인이 사용하는 자원의 유형을 분석하는 단계; 및

상기 분석된 자원의 유형을 고려하여 상기 목표 응답 시간을 만족시킬 수 있도록 상기 게스트 도메인과 상호 작용하는 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정하는 단계;를 포함하는 방법.

청구항 2

제 1 항에 있어서,

상기 관리 도메인에 할당되는 프로세서의 할당량은,

상기 관리 도메인에게 할당되어야 하는 최소의 프로세서 할당량과 상기 분석된 자원의 유형을 고려한 가중치를 각각의 게스트 도메인에 반영한 프로세서 할당량의 합산한 값인 것을 특징으로 하는 방법.

청구항 3

제 2 항에 있어서,

상기 게스트 도메인이 사용하는 자원 중, CPU 사용량이 I/O 사용량에 비해 상대적으로 많을 경우, 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정함에 있어서 상기 게스트 도메인에 대한 가중치를 미리 설정된 평균 할당량보다 크게 하는 것을 특징으로 하는 방법.

청구항 4

제 2 항에 있어서,

상기 게스트 도메인이 사용하는 자원 중, I/O 사용량이 CPU 사용량에 비해 상대적으로 많을 경우, 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정함에 있어서 상기 게스트 도메인에 대한 가중치를 미리 설정된 평균 할당량보다 작게 하는 것을 특징으로 하는 방법.

청구항 5

제 1 항에 있어서,

상기 게스트 도메인이 사용하는 자원의 유형을 분석하는 단계는,

상기 게스트 도메인으로부터 상기 관리 도메인에 송수신되는 패킷의 수를 이용하여 상기 게스트 도메인에 할당되는 워크로드를 모니터링하는 것을 특징으로 하는 방법.

청구항 6

제 1 항에 있어서,

상기 가상화 환경은 Xen이고,

상기 게스트 도메인은 domain-u이며,

상기 관리 도메인은 domain-0인 것을 특징으로 하는 방법.

청구항 7

적어도 하나 이상의 가상 머신을 제어하는 게스트 도메인과 상기 게스트 도메인을 제어하는 하나의 관리 도메인으로 구성된 가상화 환경에서 프로세서를 할당하는 방법에 있어서,

상기 게스트 도메인이 달성하여야 하는 목표 응답 시간에 기초하여 상기 게스트 도메인에게 할당되는 프로세서의 할당량을 결정하는 단계;

상기 게스트 도메인에 할당되는 워크로드를 모니터링하여 각각의 게스트 도메인이 사용하는 자원의 유형을 분석하는 단계;

상기 분석된 자원의 유형을 고려하여 상기 목표 응답 시간을 만족시킬 수 있도록 상기 게스트 도메인과 상호 작용하는 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정하는 단계;

상기 게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 상기 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 상기 가상화 환경에 따른 시스템의 물리적인 용량을 초과하는지 여부를 검사하는 단계; 및

상기 검사 결과에 따라 상기 결정된 프로세서의 할당량들을 조절하는 단계;를 포함하는 방법.

청구항 8

제 7 항에 있어서,

상기 게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 상기 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 상기 가상화 환경에 따른 시스템의 물리적인 용량을 초과할 경우, 상기 결정된 프로세서의 할당량들을 조절하는 단계는,

상기 시스템의 물리적인 용량을 초과하는 프로세서의 할당량을 산출하는 단계; 및

상기 산출된 할당량을 상기 게스트 도메인별의 요청 할당량에 비례하여 감소시키는 단계;를 포함하는 방법.

청구항 9

제 7 항에 있어서,

상기 관리 도메인에 할당되는 프로세서의 할당량은,

상기 관리 도메인에게 할당되어야 하는 최소의 프로세서 할당량과 상기 분석된 자원의 유형을 고려한 가중치를 각각의 게스트 도메인에 반영한 프로세서 할당량의 합산한 값인 것을 특징으로 하는 방법.

청구항 10

제 1 항 내지 제 9 항 중에 어느 한 항의 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

청구항 11

적어도 하나 이상의 가상 머신을 제어하는 게스트 도메인과 상기 게스트 도메인을 제어하는 하나의 관리 도메인으로 구성된 가상화 환경에서 프로세서를 할당하는 장치에 있어서,

사용자의 요청을 처리하기 위해 상기 게스트 도메인과 상기 관리 도메인에 할당되는 프로세서;

상기 게스트 도메인에 할당되는 워크로드를 모니터링하여 각각의 게스트 도메인이 사용하는 자원의 유형을 분석하는 제어부; 및

상기 게스트 도메인이 달성하여야 하는 목표 응답 시간에 기초하여 상기 게스트 도메인에게 할당되는 프로세서의 할당량을 결정하고, 상기 분석된 자원의 유형을 고려하여 상기 목표 응답 시간을 만족시킬 수 있도록 상기 게스트 도메인과 상호 작용하는 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정하는 프로세서 할당부;를 포함하는 장치.

청구항 12

제 11 항에 있어서,

상기 관리 도메인에 할당되는 프로세서의 할당량은,

상기 관리 도메인에게 할당되어야 하는 최소의 프로세서 할당량과 상기 분석된 자원의 유형을 고려한 가중치를 각각의 게스트 도메인에 반영한 프로세서 할당량의 합산한 값인 것을 특징으로 하는 장치.

청구항 13

제 12 항에 있어서,

상기 게스트 도메인이 사용하는 자원 중, CPU 사용량이 I/O 사용량에 비해 상대적으로 많을 경우, 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정함에 있어서 상기 게스트 도메인에 대한 가중치를 미리 설정된 평균 할당량보다 크게 하는 것을 특징으로 하는 장치.

청구항 14

제 12 항에 있어서,

상기 게스트 도메인이 사용하는 자원 중, I/O 사용량이 CPU 사용량에 비해 상대적으로 많을 경우, 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정함에 있어서 상기 게스트 도메인에 대한 가중치를 미리 설정된 평균 할당량보다 작게 하는 것을 특징으로 하는 장치.

청구항 15

제 11 항에 있어서,

상기 제어부는,

상기 게스트 도메인으로부터 상기 관리 도메인에 송수신되는 패킷의 수를 이용하여 상기 게스트 도메인에 할당되는 워크로드를 모니터링하는 것을 특징으로 하는 장치.

청구항 16

제 11 항에 있어서,

상기 프로세서 할당부는,

상기 게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 상기 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 상기 가상화 환경에 따른 시스템의 물리적인 용량을 초과하는지 여부를 검사하고,

상기 검사 결과에 따라 상기 결정된 프로세서의 할당량들을 조절하는 것을 특징으로 하는 장치.

청구항 17

제 16 항에 있어서,

상기 게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 상기 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 상기 가상화 환경에 따른 시스템의 물리적인 용량을 초과할 경우, 상기 프로세서 할당부는,

상기 시스템의 물리적인 용량을 초과하는 프로세서의 할당량을 산출하고,

상기 산출된 할당량을 상기 게스트 도메인별의 요청 할당량에 비례하여 감소시키는 것을 특징으로 하는 장치.

명 세 서

기 술 분 야

[0001] 본 발명은 가상화 환경에서 프로세서를 할당하는 기술에 관한 것으로, 특히 클라우드(cloud) 기반의 가상화 환경에서 가상 머신을 관리하는 관리 도메인의 오버헤드(overhead)를 고려하여 각각의 가상 머신 및 관리 도메인에 프로세서를 할당하는 장치, 방법 및 그 방법을 기록한 기록매체에 관한 것이다.

배 경 기 술

[0002] 가상화(virtualization)는 컴퓨터의 자원을 추상화(abstraction)해 실제 물리적인 자원과 분리시키는 기술이다. 종래에는 중앙처리장치(CPU)나 메모리 같은 물리적인 컴퓨팅 자원은 운영체제(operating system: OS)와 연결되어 하나의 컴퓨팅 시스템을 구성하였다. 즉, 하나의 하드웨어 세트(set)에는 하나의 컴퓨팅 시스템만이 구성 가능하였다. 그러나, 가상화는 이러한 종래의 개념을 깨고 하나의 물리적인 하드웨어 상에 다수의 운영체제 및 이에 기반한 컴퓨팅 시스템을 구축하는 것을 가능하게 한다. 특히, 다양한 가상화 기술 중 서버 가상화 기술은 사용자나 기업의 입장에서는 하나의 PC나 서버를 가지고서도 여러 대의 PC를 가지고 있는 것처럼 사용할 수 있게 해 준다.

[0003] 통합된 서버 시스템은 가상화를 사용하여 그 영역을 자유롭게 확장할 수 있다. 특히 대규모 컴퓨팅 또는 클러스터 시스템에서 이러한 가상화는 서버 호스팅 센터 또는 서버 마켓에 대한 가장 중요한 고려 요소들 중의 하나인 운영 비용을 낮추고자 하는 노력으로부터 비롯하였다. 서버 통합으로 인해 동일한 서버의 수를 동작시키기 위해 더 적은 수의 물리적 장치가 필요하게 되었다. 이러한 고려 요소들은 통합된 서버 시스템의 전체 운영 비용에 직접적으로 관련되어 있다. 나아가, 가상화 시스템은 서버의 가용성(availability) 및 운용성(manageability)에 있어서도 장점을 가진다.

[0004] 하드웨어의 빠른 발전에 기인한 서버 가상화 분야는 손쉬운 가상 기계 통합(virtual machine consolidation)이 가능하고 보안(security), 격리(isolation), 이주(migration), 오류 감내(fault tolerance), 복제(cloning), 전력 소비 감소 및 관리의 편리함으로 가상화된 PC의 사용은 폭발적으로 증가할 것으로 전망된다. 이러한 전망을 바탕으로 많은 종류의 가상화 솔루션이 개발되고 있으며, 리눅스 기반의 Xen 역시 그 중의 하나이다.

[0005] 가상화 솔루션은 서버들을 보다 편리하게 관리할 수 있는 기반은 제공하지만 컴퓨팅 파워, 네트워크 대역폭 및 전력 소모 등의 하드웨어 자원을 효율적인 이용하기 위해서는 자동화된 관리가 필수적이다. 이를 위해 이하에서 인용되는 비특허문헌에서는 지속적으로 연구되고 있는 자율 컴퓨팅(Autonomous Computing)을 비롯하여 자동화된 자원 관리 프레임워크가 가상화 환경에 적합한 형태로 제안되고 있다. 대표적인 예로 Sandpiper 프레임워크는 가상화 환경에서 시스템 상태 정보 수집하여 과부하를 탐지함으로써, 가상 기계의 service level agreement(SLA)의 만족 여부를 가늠한다. 만약 현재 시스템의 상태가 SLA를 만족시킬 수 없다고 판단될 경우 가상 기계에 할당된 CPU의 weight를 조정하여 자원을 재분배하거나, 필요한 경우 가상 기계를 다른 물리 기계로 이주시킴으로써 과부하를 해결한다.

[0006] 그러나 기존의 자동화된 자원 관리 프레임워크는 사용자의 요청 응답 시간 만족이라는 측면에서 불완전한 부분이 존재하는바, 이하에서는 이러한 가상화 환경에서의 자동화된 자원 관리 프레임워크의 문제점을 해결할 수 있는 기술적 수단이 요구된다.

선행기술문헌

비특허문헌

[0007] (비특허문헌 0001) Manish Parashar and Salim Hariri, "Autonomic Computing: Concepts, Infrastructure, and Applications", CRC Press, 2007.

(비특허문헌 0002) Simon Dobson, Roy Sterritt, Paddy Nixon and Mike Hinchey, "Fulfilling the Vision of Autonomic Computing", IEEE Computer, vol.43, no.1, pp.35-41, Jan. 2010.

(비특허문헌 0003) Shenin Hassan, Dhiya Al-Jumeily and Abir Jaafar Hussain, "Autonomic Computing Paradigm to Support System's Development", DESE, 2009 Second International Conference, pp.273-278, Dec. 2009.

발명의 내용

해결하려는 과제

[0008] 본 발명의 실시예들이 해결하고자 하는 기술적 과제는, 가상화 환경에서 자동화된 자원을 관리하는 종래의 프레임워크가 CPU와 같은 프로세서 자원을 할당함에 있어서, 관리 도메인의 오버헤드를 고려하지 않고 있다는 기술적 약점을 극복하고, 이러한 약점으로 인해 응답 시간과 같은 사용자 요구 사항을 만족시키지 못하는 문제점을

해결하고자 한다.

과제의 해결 수단

[0009] 상기 기술적 과제를 해결하기 위하여, 본 발명의 일 실시예에 따라 적어도 하나 이상의 가상 머신을 제어하는 게스트 도메인(guest domain)과 상기 게스트 도메인을 제어하는 하나의 관리 도메인으로 구성된 가상화 환경에서 프로세서(processor)를 할당하는 방법은, 상기 게스트 도메인이 달성하여야 하는 목표 응답 시간에 기초하여 상기 게스트 도메인에게 할당되는 프로세서의 할당량을 결정하는 단계; 상기 게스트 도메인에 할당되는 워크로드(workload)를 모니터링(monitoring)하여 각각의 게스트 도메인이 사용하는 자원의 유형을 분석하는 단계; 및 상기 분석된 자원의 유형을 고려하여 상기 목표 응답 시간을 만족시킬 수 있도록 상기 게스트 도메인과 상호 작용하는 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정하는 단계;를 포함한다.

[0010] 일 실시예에 따른 상기된 가상화 환경에서 프로세서를 할당하는 방법에서, 상기 관리 도메인에 할당되는 프로세서의 할당량은, 상기 관리 도메인에게 할당되어야 하는 최소의 프로세서 할당량과 상기 분석된 자원의 유형을 고려한 가중치를 각각의 게스트 도메인에 반영한 프로세서 할당량의 합산한 값이다.

[0011] 또한, 일 실시예에 따른 상기된 가상화 환경에서 프로세서를 할당하는 방법에서, 상기 게스트 도메인이 사용하는 자원의 유형을 분석하는 단계는, 상기 게스트 도메인으로부터 상기 관리 도메인에 송수신되는 패킷의 수를 이용하여 상기 게스트 도메인에 할당되는 워크로드를 모니터링한다.

[0012] 상기 기술적 과제를 해결하기 위하여, 본 발명의 일 실시예에 따라 적어도 하나 이상의 가상 머신을 제어하는 게스트 도메인과 상기 게스트 도메인을 제어하는 하나의 관리 도메인으로 구성된 가상화 환경에서 프로세서를 할당하는 방법은, 상기 게스트 도메인이 달성하여야 하는 목표 응답 시간에 기초하여 상기 게스트 도메인에게 할당되는 프로세서의 할당량을 결정하는 단계; 상기 게스트 도메인에 할당되는 워크로드를 모니터링하여 각각의 게스트 도메인이 사용하는 자원의 유형을 분석하는 단계; 상기 분석된 자원의 유형을 고려하여 상기 목표 응답 시간을 만족시킬 수 있도록 상기 게스트 도메인과 상호 작용하는 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정하는 단계; 상기 게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 상기 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 상기 가상화 환경에 따른 시스템의 물리적인 용량을 초과하는지 여부를 검사하는 단계; 및 상기 검사 결과에 따라 상기 결정된 프로세서의 할당량들을 조절하는 단계;를 포함한다.

[0013] 일 실시예에 따른 상기된 가상화 환경에서 프로세서를 할당하는 방법에서, 상기 게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 상기 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 상기 가상화 환경에 따른 시스템의 물리적인 용량을 초과할 경우, 상기 결정된 프로세서의 할당량들을 조절하는 단계는, 상기 시스템의 물리적인 용량을 초과하는 프로세서의 할당량을 산출하는 단계; 및 상기 산출된 할당량을 상기 게스트 도메인별의 요청 할당량에 비례하여 감소시키는 단계;를 포함한다.

[0014] 한편, 이하에서는 상기 기재된 가상화 환경에서 프로세서를 할당하는 방법들을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체를 제공한다.

[0015] 상기 기술적 과제를 해결하기 위하여, 본 발명의 일 실시예에 따라 적어도 하나 이상의 가상 머신을 제어하는 게스트 도메인과 상기 게스트 도메인을 제어하는 하나의 관리 도메인으로 구성된 가상화 환경에서 프로세서를 할당하는 장치는, 사용자의 요청을 처리하기 위해 상기 게스트 도메인과 상기 관리 도메인에 할당되는 프로세서; 상기 게스트 도메인에 할당되는 워크로드를 모니터링하여 각각의 게스트 도메인이 사용하는 자원의 유형을 분석하는 제어부; 및 상기 게스트 도메인이 달성하여야 하는 목표 응답 시간에 기초하여 상기 게스트 도메인에게 할당되는 프로세서의 할당량을 결정하고, 상기 분석된 자원의 유형을 고려하여 상기 목표 응답 시간을 만족시킬 수 있도록 상기 게스트 도메인과 상호 작용하는 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정하는 프로세서 할당부;를 포함한다.

[0016] 일 실시예에 따른 상기된 가상화 환경에서 프로세서를 할당하는 장치에서, 상기 관리 도메인에 할당되는 프로세서의 할당량은, 상기 관리 도메인에게 할당되어야 하는 최소의 프로세서 할당량과 상기 분석된 자원의 유형을 고려한 가중치를 각각의 게스트 도메인에 반영한 프로세서 할당량의 합산한 값이다.

[0017] 또한, 일 실시예에 따른 상기된 가상화 환경에서 프로세서를 할당하는 장치에서, 상기 제어부는, 상기 게스트 도메인으로부터 상기 관리 도메인에 송수신되는 패킷의 수를 이용하여 상기 게스트 도메인에 할당되는 워크로드를 모니터링한다.

[0018] 나아가, 일 실시예에 따른 상기된 가상화 환경에서 프로세서를 할당하는 장치에서, 상기 프로세서 할당부는, 상기 게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 상기 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 상기 가상화 환경에 따른 시스템의 물리적인 용량을 초과하는지 여부를 검사하고, 상기 검사 결과에 따라 상기 결정된 프로세서의 할당량들을 조절한다.

발명의 효과

[0019] 본 발명의 실시예들은 게스트 도메인이 사용하는 자원의 유형을 고려하여 목표 응답 시간을 만족시킬 수 있도록 관리 도메인에 할당되는 프로세서의 할당량을 결정함으로써, 관리 도메인의 오버헤드에 의한 병목을 방지하고, 그 결과 목표 응답 시간과 같은 사용자 요구 사항을 만족시킬 수 있다. 나아가, 본 발명의 실시예들은 가상화 환경에 따른 시스템의 물리적인 용량을 고려하여 결정된 프로세서의 할당량을 조절함으로써 전체 가상화 시스템의 과부하를 방지할 수 있다.

도면의 간단한 설명

[0020] 도 1은 가상화 환경에서의 물리적인 자원과 가상 머신의 구조를 예시한 도면이다.

도 2는 가상화 환경에서의 워크로드 모니터링 방법을 예시한 도면이다.

도 3은 본 발명의 실시예들이 구현되는 가상화 환경에서 요청을 처리하기 위한 관리 도메인(도메인-0)의 프로세서 할당을 설명하기 위한 도면이다.

도 4는 본 발명의 일 실시예에 따른 가상화 환경에서 프로세서를 할당하는 방법을 도시한 흐름도이다.

도 5a 및 도 5b는 프로세서 할당량 조절 방법에 따라 프로세서 할당량이 조율되는 과정을 설명하기 위한 예시도이다.

도 6은 본 발명의 일 실시예에 따른 도 4의 프로세서 할당 방법에서 전체 가상화 시스템의 자원이 부족한 경우 프로세서 할당량을 조절하는 방법을 추가적으로 도시한 흐름도이다.

도 7은 본 발명의 일 실시예에 따른 가상화 환경에서 프로세서를 할당하는 장치를 도시한 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0021] 본 발명의 실시예들을 설명하기에 앞서, 가상화 환경에서 자동화된 자원 관리 프레임워크가 갖는 문제점을 검토한 후, 이들 문제점을 해결하기 위해 본 발명의 실시예들이 채택하고 있는 기술적 배경에 대해 개괄적으로 검토하도록 하겠다.

[0022] 기존의 자동화된 자원 관리 프레임워크는 사용자의 요청 응답 시간 만족이라는 측면에서 불완전한 부분이 존재한다. Xen은 도메인-0이라는 가장 먼저 생성되는 가상 기계를 통해서 하이퍼바이저에 접근함으로써 가상 시스템을 관리하고 하드웨어에 접근하도록 한다. 또한 Xen은 분리 드라이버 모델(split driver model)을 채택하여, 도메인-0의 백엔드(Back-end) 장치 드라이버와 도메인-U의 프론트엔드(Front-end) 장치 드라이버가 통신함으로써 도메인-U가 하드웨어 자원을 이용할 수 있도록 한다. 이러한 Xen의 구조는 기존 디바이스 드라이버를 수정 없이 도메인-0에서 그대로 사용할 수 있기 때문에 호환성과 안정성 면에서 강점을 가진다.

[0023] 만약 도메인-U에서 I/O가 발생되면 프론트엔드 드라이버를 통해 도메인-0의 백엔드 드라이버로 요청이 들어가고 최종적으로 도메인-0에서 I/O가 수행된 뒤 다시 백엔드 드라이버와 프론트엔드 드라이버를 거쳐 도메인-U로 결과가 전달된다. 이로 인해 도메인-U의 I/O는 도메인-0에 CPU 오버헤드를 주는 원인이 된다. 때문에 Xen 가상화 환경에서 사용자의 요청 응답 시간은 도메인-0의 자원 이용률과 도메인-U의 I/O 양에 많은 영향을 받는다. 만약 도메인-U가 I/O가 많은 네트워크 자원 위주의 요청을 처리해야 할 경우, CPU 할당 시 도메인-0을 고려하지 않는 기존의 자동화된 자원 관리 프레임워크로는 요청에 대한 응답 시간을 보장할 수 없다. 따라서 이 문제를 해결하기 위해서는 도메인-U의 네트워크 자원 활용 경향을 고려한 도메인-0의 CPU 할당 방법이 필요하다.

[0024] 이하에서 기술될 본 발명의 실시예들은 사용자의 서비스 요청 응답 시간을 최대한 만족시키기 위해, 도메인-0이 도메인-U와 자신의 CPU를 할당하는 방법을 제안한다. 이 할당 기법은 도메인-0에 부과되는 오버헤드를 고려하였고 송수신 네트워크 바이트 수보다 송수신 패킷의 수가 도메인-0의 CPU 사용률에 영향을 준다는 점을 알고리즘에 반영하여 도메인의 cap을 조절함으로써 효율적인 자원 할당을 이끌어내도록 한다.

[0025] 일반적으로 서버는 안정적인 서비스를 제공하기 위해 응용 프로그램이 요구하는 QoS를 만족시키기 위한 리소스

를 보장해야 한다. 한 대의 물리 기계 상에 다수의 가상머신이 동작하는 가상화 환경은 자원을 공유하여 사용하기 때문에, 비가상화 환경에 비하여 QoS를 만족시키는 것이 복잡하다. 이하에서는 가상화 환경에서 QoS 만족을 위한 자원 관리 프레임워크인 Sandpiper의 동작과 그 문제점에 대해서 살펴본다.

[0026] Sandpiper는 가상화 환경에서 시스템 상태 정보 수집, 수집된 정보의 프로파일링, 과열 지점 탐지 그리고 이주를 수행하는 자동화된 프레임워크이다. Sandpiper는 SLA를 만족시키기 위해 CPU의 weight를 조정하기 위해 노력하며, 이 배경에는 G/G/1 대기 행렬 모델이 자리 잡고 있다.

[0027] Sandpiper는 다음과 같이 구성된다. 물리 기계 위에 Xen 하이퍼바이저가 있고, 그 위에 시스템의 상태 정보를 수집하는 모니터링 엔진을 포함한 도메인-0과 도메인-U가 있다. 모니터링 엔진은 도메인-0뿐만 아니라 도메인-U에서 제공하는 상세한 정보를 모아서 프로파일링 엔진에 전달한다. 프로파일링 엔진은 모니터링 엔진으로부터 전달된 정보들을 모아서 시계열 프로파일과 활용 프로파일을 생성한다. 시계열 프로파일은 정해진 시간의 윈도우 W만큼 데이터를 유지해주고, 활용 프로파일은 윈도우 W 동안 특정 수준의 데이터가 얼마나 높은 빈도를 갖는지 관리해준다. 과열 지점 탐지기는 프로파일 데이터를 기반으로 각 CPU, 메모리, 네트워크에 대해 최근 n개의 데이터 중 k개 이상이 미리 정의해 둔 임계값을 넘어서면 해당 항목에 대해 과열 지점이 발견된 것으로 판단한다. n의 값이 k보다 커질 경우, 과열 지점을 보수적으로 판단한다. n과 k의 값이 비슷할수록 과열 지점을 공격적으로 판단한다. 과열 지점의 해소 방법으로는 1차적으로 예측된 다음 단계의 워크로드를 기반으로 CPU, 메모리, 네트워크 자원을 재분배한다. 만약 자원의 재분배를 통해 해결할 수 없을 정도로 과부하 상태라고 판단될 경우 이주 관리자는 선택적으로 도메인-U를 다른 물리 기계로 이주시킨다.

[0028] 그러나 Sandpiper는 자원의 재분배에 있어서 문제점을 가지고 있는데, 이는 각 대기 행렬 즉 각 도메인-U의 워크로드만을 관찰하여 자원을 재분배한다는 것이다. 도메인-U는 요청을 처리하기 위해서 일정량의 CPU를 소모하고 I/O를 처리해야만 하며, Xen 드라이버 모델의 특성으로 인해 도메인-U의 I/O는 도메인-0의 CPU를 소모하게 된다. 따라서 도착하는 요청이 여러 개의 작은 패킷으로 이루어져 있거나 디스크 접근이 많은 경우 요청을 처리하기 위해서는 도메인-0의 CPU 자원이 추가적으로 필요하게 된다. 만약 도메인-0에게 충분한 양의 CPU가 할당되지 못할 경우, 도메인-U의 요청들이 정해진 목표 응답시간 내에 처리하지 못할 확률이 커진다.

[0029] 또한 Sandpiper는 각 가상머신에 할당되는 가중치(weight)를 설정함으로써 CPU를 가상머신에 분배하고 있는데, 이는 cap 설정을 통한 분배방식에 비해서 효율적이지 못하다.

[0030] 비가상화 환경에서 온라인 측정 방법과 워크로드를 통한 응답 시간 예측을 통해, 응답 시간에 관한 사용자의 불만(discontent) 지수를 최소화하는 자원 할당 방법은 다음과 같다. 다수의 서비스가 하나의 시스템에서 동작하는 구조에서는 서비스간의 공유자원의 할당 문제가 QoS 만족에 영향을 주게 된다. 특히 서비스의 워크로드가 시간에 따라 빠르게 변화하는 특성을 가지는 경우 자원 할당의 문제는 더욱 중요해진다. 이 경우, 상기된 자원 할당 방법은 이러한 환경에서의 워크로드와 응답 시간 예측을 위한 프레임워크를 제시하고, 이에 기반을 두어 사용자의 불만을 최소한으로 유지하는 자원 할당 비율을 비선형 최적화 문제로 변환하여 해결할 수 있다.

[0031] 상기된 자원 할당 방법을 통해 제안된 프레임워크의 리소스 모델은 다음과 같다. 서버는 n개의 큐가 있으며, 각 큐는 서버 상에서 동작하는 특정 애플리케이션으로의 요청이 저장된다. 큐의 요청은 선입선출의 형태로 저장되며 항상 큐의 끝에 존재하는 요청만 처리된다. 각 큐는 요청을 처리하기 위해 서버의 자원을 공유하여 사용하며 공유되는 자원의 총량은 C로, i번째 큐가 할당 받은 자원의 량은 ω_i로 표현한다. 이 때, i번째 큐가 할당 받은 공유 자원의 비율(ϕ_i)은 다음의 수학식 1과 같이 계산될 수 있다.

수학식 1

[0032]
$$\phi_i = \frac{w_i}{\sum_j w_j}$$

[0033] 제안된 프레임워크는 일정한 주기 I마다 요청의 도착률 λ_i와 서비스 큐에 대기 중인 요청의 수 q_i, 큐의 요청의 처리율 μ_i 등의 정보를 반복적으로 수집하여 축적한다. 축적된 데이터는 일정시간 동안 유지되며, 축적된 데이

터를 바탕으로 i번째 큐의 응답 시간 $\overline{T_i}$는 다음의 수학식 2와 같이 예측된다.

수학식 2

$$\overline{T_i} = \left(\frac{\hat{s}_i}{\phi_i \cdot C} \right) \cdot (\overline{q_i} + 1)$$

[0034]

[0035] 이 때 $\hat{s}_i$는 프레임워크에 의해 예측된 i번째 큐에 대기 중인 요청의 평균 자원 요구량이며, $\overline{q_i}$는 예측된 큐의 길이이다. 즉 i번째 큐에 대기 중인 요청을 처리하는데 걸리는 예상 응답 시간은 큐에 할당된 자원의 량($\phi_i \cdot C$)에 반비례하며, 대기 중인 요청의 수($\overline{q_i}$)와 처리시간($\hat{s}_i$)에 비례한다. 이하의 표 1은 응답 시간 $\overline{T_i}$를 예측하는 데 필요한 정보를 정리한 것이다.

표 1

[0036]

변수명	설명
C	서버가 제공하는 공유 자원의 총량
w_i	i번째 큐에 할당된 공유 자원의 량
ϕ_i	i번째 큐에 할당된 자원의 비율
λ_i	i번째 큐의 요청의 도착률
q_i	i번째 큐에 대기 중인 요청 수
μ_i	i번째 큐의 요청 처리율
$\hat{s}_i$	i번째 큐의 요청이 처리되는데 필요한 자원의 평균에 대한 예측값
$\overline{q_i}$	i번째 큐에 대기 중인 요청 수의 예측값
$\overline{T_i}$	i번째 큐에 대기 중인 요청의 평균 응답 시간에 대한 예측값

[0037] 이상에서는 비가상화 환경 하에서 워크로드를 통해 요청의 응답 시간을 예측하는 방법에 대하여 살펴보았다. 이상의 응답 시간 예측을 위한 프레임워크는 본 발명의 실시예들이 해결하고자 하는 하나의 물리 기계 상에서 다수의 가상 기계가 동작하는 가상화 환경에서의 CPU의 효율적인 분배 문제에도 적용이 가능하다. 이는 본 발명의 실시예들에 따른 가상화 환경이 이상의 자원 모델과 같은 방식으로 모델링이 가능하기 때문이다. 따라서 가상화 환경에서도 이상의 수학식 2를 통해서 요청이 처리되는 데까지 걸리는 시간을 예측할 수 있다.

[0038] 도 1은 가상화 환경에서의 물리적인 자원(120)과 가상 머신(110)의 구조를 예시한 도면으로서, 도 1은 예측을 위해 필요한 표 1의 변수들이 가상화 환경에서 어떻게 수집될 수 있는 지를 설명하고 있다.

[0039] 앞서 제시한 응답 시간 예측 모델에서 큐는 요청을 저장하고 처리하는 애플리케이션을 의미하는데, 가상화 환경에서는 큐를 서비스를 제공하는 가상 기계(110)로 볼 수 있다. 또한 서버(120)는 물리 기계로, 공유자원의 물리 기계가 제공하는 CPU의 총 크레딧으로 볼 수 있고, 각 가상 기계는 cap과 weight 설정에 따라서 크레딧을 할당받아 대기 중인 요청을 처리한다.

[0040] 수학식 2를 통해 응답 시간을 예측하기 위해서는 식의 계산에 필요한 변수 중 측정을 통해서 얻을 수 있는 데이

터, 즉 C, ω_i, λ_i, q_i, μ_i을 모니터링을 통해서 구해야만 한다. 이 데이터 중 공유 자원의 총량(C)은 상수값으로 고정되어 있으며, 할당된 크레딧(ω_i)과 비율(ϕ_i)은 가상 기계의 cap과 weight 설정 값을 통해서 구할 수 있다. 그리고 λ_i, q_i, μ_i에 해당하는 값은 가상 기계에서 작동하고 있는 서비스를 모니터링 함으로써 얻을 수 있다.

[0041] 도 2는 가상화 환경에서의 워크로드 모니터링 방법을 예시한 도면으로서, 본 발명의 실시예들은 도 2와 같은 실험을 위해 가상 기계의 서비스로 아파치 웹서버를 사용했으며 아파치의 로그(access_log)를 분석하는 모니터를 작성하여 요청의 도착률(λ_i)과 요청의 처리율(μ_i)을 얻을 수 있었다. 또한 대기 중인 요청의 수(q_i)는 도착률과 처리율의 차를 통해서 얻을 수 있었다. 이 때 도착률과 처리율의 정확한 측정을 위해서 기존의 로그의 시간 정확성을 높일 필요가 있었고, 이를 위해 아파치 웹서버의 로그 모듈을 수정하였다. 각 가상 기계의 모니터는 요청의 도착률과 처리율, 대기 중인 요청의 수를 주기적으로 수집하여 도메인-0에서 동작하는 자율적 관리자에게 일정 시간마다 전송한다. 자율적 관리자가 수집된 각 가상 기계의 워크로드 특성을 반영하여 각 가상 기계의 cap과 weight를 조절함으로써 공유자원의 분배는 완료된다.

[0042] 이상에서는 관리 프레임워크의 일례인 Sandpiper가 가상머신으로 유입되는 요청의 응답시간을 보장하기 위해 취한 CPU 재분배 방법과 그 문제점에 대해서 살펴보았다. Sandpiper 방식은 다음 두 가지 이유로 인해 응답시간을 보장하는 데 있어 어려움이 따른다. 첫째는 Sandpiper가 응답시간 예측 및 할당량 계산 시에 워크로드에 포함된 요청의 I/O 요구량을 파악하지 않기 때문이며, 두 번째는 wieght를 조절하여 CPU를 할당하기 때문이다.

[0043] 도 3은 본 발명의 실시예들이 구현되는 가상화 환경에서 요청을 처리하기 위한 관리 도메인(도메인-0)의 프로세서 할당을 설명하기 위한 도면으로서, 도 3은 도메인-U(321, 322)에서 실행되는 요청이 CPU 위주인 경우(도메인-U2)(322)와 I/O를 포함한 요청인 경우(도메인-U1)(321)의 차이점을 도메인-0(310)의 CPU 소모라는 관점에서 설명하고 있다. 도메인-U는 요청에 포함된 I/O를 처리하기 위해서는 도메인-U가 도메인-0(310)에게 처리할 I/O를 전송하고, 도메인-0(310)이 실제 I/O를 수행하여 다시 도메인-U에게 결과를 전송하는 과정을 거쳐야 한다. 다시 말해 도메인-U(321, 322)의 I/O를 처리하는 데는 도메인-U(321, 322)의 CPU 뿐만 아니라, 도메인-0(310)의 CPU 자원 역시 소모하게 된다. 따라서 도메인-U(321, 322)에서 처리하는 요청이 I/O가 많은 경우, 응답 시간을 보장하기 위해서는 도메인-0(310)에 I/O를 처리해낼 수 있을 정도의 CPU가 할당되어야만 한다. 즉 CPU의 재 할당 시에는 도메인-U(321, 322)에 할당되는 CPU의 비율(ϕ_1^{domU}, ϕ_2^{domU}) 뿐만 아니라, 도메인-0(310)에 할당되는 CPU의 비율(ϕ^{dom0})역시 고려해야만 한다. 이하에서는 요청의 특성을 CPU 위주와 네트워크 위주로 나누어서 각 요청에 따라서 관리 도메인을 고려한 CPU 할당에 대한 방법을 제안한다.

[0044] (1) 도메인-U의 CPU 할당

[0045] 이상에서는 도메인-U 가상 기계에 대해서 정보들을 수집하고, $\phi^{domU}C$의 CPU 할당에 대해 평균적으로 어느 정도의 응답 시간을 얻을 수 있는지 계산했다. 사용자로부터 각 가상 기계가 달성해야 하는 목표 응답 시간(R_{domU})을 입력 받으므로, R_{domU}을 만족하는 $\phi^{domU}C$를 계산해낼 수 있다. CPU를 $\Delta Alloc$ (예를 들어, 10%, 20% 등이 될 수 있다.) 단위로 할당할 수 있을 때, 다음의 수학식 3을 만족하는 최소의 n_{domU}을 결정한다(단, $n_{domU}>0$).

수학식 3

$$n_{domU} \bullet \Delta Alloc = \phi^{domU}C \geq \left(\frac{\hat{s}_{domU}}{R_{domU}}\right)\left(\overline{q_{domU}}+1\right)$$

[0046]

[0047] (2) 도메인-0의 CPU 할당

[0048] 도메인-U가 목표 응답 시간을 만족시키기 위해서는 도메인-0의 CPU 할당량 역시 계산되어야 한다. 이는 도메인-U들과 I/O 처리를 주로 담당하는 도메인-0은 하나의 물리 기계에서 동작하므로 도메인-U의 동작에 영향을 끼치기 때문이다. 도메인-0의 CPU 사용량은 도메인-U에서 I/O를 위해 사용하는 모든 부분이 영향을 미친다. 예를 들면, 네트워크 전송 및 수신을 위한 패킷의 수, 바이트 수 혹은 디스크에 접근하기 위한 읽기와 쓰기 횟수 등이 포함된다. 이 중에서 도메인-0의 CPU 사용량에 가장 많은 영향을 미치는 것은 전송 및 수신을 위한 패킷의 수로 알려져 있다. 따라서 도메인-0의 CPU를 할당률 결정 시, 패킷의 수와 같은 요인을 적극적으로 고려해야 한다.

[0049] 도메인-0의 CPU 할당량 $\phi^{dom0}C$를 결정하기에 앞서, 각각의 도메인-U가 CPU 혹은 I/O 자원을 많이 사용하는지 결정할 필요가 있다. $\phi^{dom0}C$는 주로 $\phi^{domU}C$의 일정 비율로 할당하는 것이 좋은 효율을 낸다고 알려져 있다. 따라서 절대적인 값으로 $\phi^{dom0}C$를 결정하기보다는 $\phi^{domU}C$를 고려한 상대적인 값으로 결정하는 것이 바람직하다.

[0050] $\phi^{domU}C$가 CPU 위주라면, $\phi^{domU}C$의 값이 평균적인 할당보다 큰 값을 가질 것이다. 이것은 $\phi^{dom0}C$의 값을 $\phi^{domU}C$에 대해서 임의의 비율로 할당할 때, 할당 비율이 보다 낮아야 함을 의미한다. 만일, $\phi^{domU}C$가 I/O 위주라면 평균적인 할당이거나 작은 값을 가질 것이고 역으로 $\phi^{dom0}C$의 값이 증가해야 한다. 도메인-U가 CPU와 I/O 위주 중 어떤 특성을 갖는가는 상호 배타적이라고 할 수 없다. CPU 위주와 I/O 위주의 속성을 모두 가질 수도 있고, 모두 갖지 않을 수도 있다. CPU 위주 속성 CI와 I/O 위주 속성 NI는 다음의 수학식 4 및 수학식 5와 같이 정의된다.

수학식 4

[0051]
$$CI = \begin{cases} 1, & CI\text{지수} \geq CPUIntensive_{threshold} \\ 0, & otherwise \end{cases}$$

수학식 5

[0052]
$$NI = \begin{cases} 1, & NI\text{지수} \geq NETIntensive_{threshold} \\ 0, & otherwise \end{cases}$$

[0053] 이때, CI 지수와 NI 지수는 다음의 수학식 6 및 수학식 7과 같이 정의된다.

수학식 6

[0054]
$$CI\text{지수} = \frac{\phi^{domU}C}{packet\#_{domU}}$$

수학식 7

[0055] $$NI\ 지수 = \frac{TX^{domU} + RX^{domU}}{packet\#^{domU}}$$

[0056] $CPUIntensive_{threshold}$ 와 $NETIntensive_{threshold}$ 는 물리 기계의 CPU 성능이나 I/O 채널 혹은 NIC의 성능에 의해 쉽게 변할 수 있는 값이므로, 효과적인 값을 찾기 위해 참조 모델을 세우거나 실험에 의한 조절이 필요하다.

[0057] $\phi^{dom0}C$ 는 최소 할당량 $\phi^{dom0}_{min}C$ 에 각각 도메인-U의 CI 및 NI를 고려한 할당량을 모두 합산함으로써 다음의 수학식 8과 같이 계산할 수 있다.

수학식 8

[0058] $$\phi^{dom0}C = \phi^{dom0}_{min}C + \sum_{U}\phi^{domU}C\ [\ \alpha CI + \beta NI\ ]^{+}$$

[0059] 최소 할당량 $\phi^{dom0}_{min}C$ 는 도메인-0이 최소한으로 보장받아야 할 할당량을 의미한다. 도메인-U의 I/O 처리를 위한 할당량 이외에도 물리 기계의 정보를 수집하거나, 도메인-0의 관리 및 운영을 위한 부분이다. α와 β는 각각 CI 및 NI를 위한 가중치이다. α가 커질수록 CI일 때 반영되는 $\phi^{dom0}C$ 가 증가하며, β가 커질수록 CI일 때 반영되는 $\phi^{dom0}C$ 가 증가한다. 앞서 기술한 바에 따라 대체적으로 α는 음의 값으로 결정되며, β는 양의 값으로 결정된다. 본 실시예를 구현하는 실험에서는 α와 β를 각각 -0.1과 0.3으로 설정하였다.

[0060] 도 4는 본 발명의 일 실시예에 따른 가상화 환경에서 프로세서를 할당하는 방법을 도시한 흐름도로서, 이러한 가상화 환경은 적어도 하나 이상의 가상 머신을 제어하는 게스트 도메인(guest domain)과 상기 게스트 도메인을 제어하는 하나의 관리 도메인으로 구성된다. 예컨대, 상기 가상화 환경은 Xen이고, 상기 게스트 도메인은 domain-u이며, 상기 관리 도메인은 domain-0일 수 있으나, 단지 이러한 구현예에 한정되지 않으며, 이하에서 기술될 특징들이 나타나는 다양한 구현예에 널리 적용될 수 있을 것이다.

[0061] 410 단계에서는 상기 게스트 도메인이 달성하여야 하는 목표 응답 시간에 기초하여 상기 게스트 도메인에게 할당되는 프로세서의 할당량을 결정한다. 이러한 과정은 앞서 기술한 수학식 3과 같은 방법을 통해 결정될 수 있을 것이다.

[0062] 420 단계에서는 상기 게스트 도메인에 할당되는 워크로드(workload)를 모니터링(monitoring)하여 각각의 게스트 도메인이 사용하는 자원의 유형을 분석한다. 이 때, 상기 게스트 도메인이 사용하는 자원의 유형을 분석하는 단계는, 상기 게스트 도메인으로부터 상기 관리 도메인에 송수신되는 패킷의 수를 이용하여 상기 게스트 도메인에 할당되는 워크로드를 모니터링할 수 있다.

[0063] 430 단계에서는 상기 분석된 자원의 유형을 고려하여 상기 목표 응답 시간을 만족시킬 수 있도록 상기 게스트 도메인과 상호 작용하는 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정한다. 앞서 수학식 8을 통해 기술한 바와 같이, 상기 관리 도메인에 할당되는 프로세서의 할당량은, 상기 관리 도메인에게 할당되어야 하는 최소의 프로세서 할당량과 상기 분석된 자원의 유형을 고려한 가중치를 각각의 게스트 도메인에 반영한 프로세서 할당량의 합산한 값이 될 수 있다.

[0064] 특히, 상기 게스트 도메인이 사용하는 자원 중, CPU 사용량이 I/O 사용량에 비해 상대적으로 많을 경우, 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정함에 있어서 상기 게스트 도메인에 대한 가중치를 미리 설정

된 평균 할당량보다 크게 하는 것이 바람직하다. 반면, 상기 게스트 도메인이 사용하는 자원 중, I/O 사용량이 CPU 사용량에 비해 상대적으로 많을 경우, 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정함에 있어서 상기 게스트 도메인에 대한 가중치를 미리 설정된 평균 할당량보다 작게 하는 것이 바람직하다.

[0065] 이상에서는, 예상되는 응답 시간을 예측하고, 사용자의 응답 시간인 R을 만족하는 도메인-0과 도메인-U의 할당량을 결정하였다. 하지만 실제로 할당량을 물리 기계에 적용하기 전에 요청된 할당량이 물리 기계의 용량(C)을 초과하지 않는지 검토해야 한다. 현재 Xen 하이퍼바이저에서 기본적으로 동작하고 있는 크레딧 스케줄러는 모든 할당량의 합이 시스템 전체의 용량을 초과하는 경우에는 작업 보장성(work conserving) 스케줄링 방법으로 동작하도록 설계되어 있다. 작업 보장성 스케줄링 방법은 사용되는 CPU의 양을 명확하게 구분하지 않기 때문에 목표 응답 시간을 갖는 할당 정책에서는 배제되어야 한다. 할당량의 조율은 다음과 같이 세 가지로 분류되어 수행된다.

[0066] i) CASE 1: $\phi^{dom0} + \sum_{U} \phi^{domU} < \phi_{lowerbound}$ 인 경우,

[0067] 할당은 도메인-0과 도메인-U의 요청 할당량이 시스템에 과부하를 주지 않는 범위 내에 속하므로 조율의 필요가 없다.

[0068] ii) CASE 2: $\phi_{lowerbound} \leqq \phi^{dom0} + \sum_{U} \phi^{domU} < \phi_{upperbound}$ 인 경우,

[0069] 할당은 요청 할당량이 적정한 수준에 머물러 있으나, 할당량에 근접하도록 CPU가 활용될 때 목표 응답 시간을 만족하지 못할 가능성이 있다고 판단한다. 이 때, 비용이 높은 이주를 바로 수행하기보다는 가상 기계의 cap을 조정함으로써 목표 응답 시간을 지키기 위한 노력을 선행한다. 이를 위해 물리 기계의 용량을 넘어서는 할당 비율을 다음의 수학식 9와 같이 정의할 수 있다.

수학식 9

[0070] $$\gamma = \left[\phi^{dom0} + \sum_{U} \phi^{domU} - \phi_{upperbound} \right]$$

[0071] 또한 도메인-U의 각 $\phi^{domU} C$ 에 대해, 조율된 ϕ_{new}^{domU} 는 다음의 수학식 10과 같다.

수학식 10

[0072] $$\phi_{new}^{domU} = \phi^{domU} - \gamma \frac{\phi^{domU}}{\sum_{U} \phi^{domU}}$$

[0073] 즉, 물리 기계의 용량을 넘어서는 부분에 대해서 가상 기계의 할당량을 요청 할당량에 비례하여 감소시킨다. 다만, 도메인-0의 할당량은 계산에서 제외되는 것이 바람직하다.

[0074] 예를 들면 도 5a와 같이 하나의 물리 기계 위에 세 개의 가상 기계가 있을 때, ϕ^{domU} 가 각각 0.2, 0.4, 0.3 이고 ϕ^{dom0} 은 0.2이다. $\phi_{lowerbound}$ =1.0, $\phi_{upperbound}$ =1.1 이라 가정한다. 즉, 도 5a는 조율 전의 할당량을 나타낸다.

[0075] 이제, 이상에서 제시한 할당량 조정 방법에 따라 조율된 ϕ_{new}^{domU} 는 도 5b와 같이 계산된다. 즉, 도 5b는 조율 후의 할당량을 나타낸다.

[0076] iii) CASE 3: $\phi^{dom0} + \sum_{U} \phi^{domU} \geqq \phi_{upperbound}$ 인 경우,

[0077] 할당은 요청 할당량이 물리 기계의 용량을 크게 벗어난다. 따라서 가상 기계 이주 정책 관리자에게 현재의 물리 기계는 과부하의 상태에 놓일 가능성이 크므로, 가상 기계 중 하나를 이주 정책에 따라 다른 물리 기계로 이주할 것을 권고하는 것이 바람직하다.

[0078] 도 6은 본 발명의 일 실시예에 따른 도 4의 프로세서 할당 방법에서 전체 가상화 시스템의 자원이 부족한 경우 프로세서 할당량을 조절하는 방법을 추가적으로 도시한 흐름도로서, 앞서 도 4를 통해 설명한 430 단계의 이후의 과정을 도시하였다.

[0079] 440 단계에서는 410 단계를 통해 게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 430 단계를 통해 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 가상화 환경에 따른 시스템의 물리적인 용량을 초과하는지 여부를 검사한다. 이 경우, 합산값이 용량을 초과할 경우, 할당량의 조절이 이루어진다. 이러한 구체적인 조절 과정은 이상에서 설명한 3 가지 경우에 따를 것이나, 여기서는 그 개요만을 개괄적으로 소개한다. (특히, CASE 2의 경우에 집중하여 소개하도록 한다.)

[0080] 합산값이 용량을 초과하는 경우, 450 단계에서는 시스템의 물리적인 용량을 초과하는 프로세서의 할당량을 산출한다. 그런 다음, 460 단계에서는 산출된 할당량을 게스트 도메인별의 요청 할당량에 비례하여 감소시킨다.

[0081] 도 7은 본 발명의 일 실시예에 따른 가상화 환경에서 프로세서를 할당하는 장치(700)를 도시한 블록도로서, 각각의 구성요소들이 수행하는 동작은 앞서 소개한 도 4 및 도 6의 과정들에 대응되므로, 여기서는 중복을 피하기 위해 그 장치적 특징에 집중하여 개요만을 소개하도록 한다.

[0082] 본 실시예에 따른 가상화 환경은 적어도 하나 이상의 가상 머신을 제어하는 게스트 도메인(10)과 상기 게스트 도메인을 제어하는 하나의 관리 도메인(20)으로 구성된다. 또한, 프로세서(30)는 사용자의 요청을 처리하기 위해 상기 게스트 도메인(10)과 상기 관리 도메인(20)에 할당되는 자원의 일종이다.

[0083] 제어부(40)는 상기 게스트 도메인에 할당되는 워크로드를 모니터링하여 각각의 게스트 도메인이 사용하는 자원의 유형을 분석한다. 이 때, 상기 제어부는 상기 게스트 도메인으로부터 상기 관리 도메인에 송수신되는 패킷의 수를 이용하여 상기 게스트 도메인에 할당되는 워크로드를 모니터링할 수 있다.

[0084] 프로세서 할당부(50)는 상기 게스트 도메인이 달성하여야 하는 목표 응답 시간에 기초하여 상기 게스트 도메인에게 할당되는 프로세서의 할당량을 결정하고, 상기 분석된 자원의 유형을 고려하여 상기 목표 응답 시간을 만족시킬 수 있도록 상기 게스트 도메인과 상호 작용하는 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정한다. 이 때, 상기 관리 도메인에 할당되는 프로세서의 할당량은, 상기 관리 도메인에게 할당되어야 하는 최소의 프로세서 할당량과 상기 분석된 자원의 유형을 고려한 가중치를 각각의 게스트 도메인에 반영한 프로세서 할당량의 합산한 값이 될 수 있다.

[0085] 특히, 상기 게스트 도메인이 사용하는 자원 중, CPU 사용량이 I/O 사용량에 비해 상대적으로 많을 경우, 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정함에 있어서 상기 게스트 도메인에 대한 가중치를 미리 설정된 평균 할당량보다 크게 하는 것이 바람직하다. 반면, 상기 게스트 도메인이 사용하는 자원 중, I/O 사용량이 CPU 사용량에 비해 상대적으로 많을 경우, 상기 관리 도메인에 할당되는 프로세서의 할당량을 결정함에 있어서

상기 게스트 도메인에 대한 가중치를 미리 설정된 평균 할당량보다 작게 하는 것이 바람직하다.

[0086] 나아가, 상기 프로세서 할당부는, 상기 게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 상기 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 상기 가상화 환경에 따른 시스템의 물리적인 용량을 초과하는지 여부를 검사하고, 상기 검사 결과에 따라 상기 결정된 프로세서의 할당량들을 조절할 수 있다. 만약 상기 게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 상기 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 상기 가상화 환경에 따른 시스템의 물리적인 용량을 초과할 경우, 상기 프로세서 할당부는, 상기 시스템의 물리적인 용량을 초과하는 프로세서의 할당량을 산출하고, 상기 산출된 할당량을 상기 게스트 도메인별의 요청 할당량에 비례하여 감소시키는 것이 바람직하다.

[0087] 상기된 본 발명의 실시예들에 따르면, 게스트 도메인이 사용하는 자원의 유형을 고려하여 목표 응답 시간을 만족시킬 수 있도록 관리 도메인에 할당되는 프로세서의 할당량을 결정함으로써, 관리 도메인의 오버헤드에 의한 병목을 방지하고, 그 결과 목표 응답 시간과 같은 사용자 요구 사항을 만족시킬 수 있다. 나아가, 본 발명의 실시예들은 가상화 환경에 따른 시스템의 물리적인 용량을 고려하여 결정된 프로세서의 할당량을 조절함으로써 전체 가상화 시스템의 과부하를 방지할 수 있다.

[0088] 한편, 본 발명의 실시예들은 컴퓨터로 읽을 수 있는 기록 매체에 컴퓨터가 읽을 수 있는 코드로 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록 매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다.

[0089] 컴퓨터가 읽을 수 있는 기록 매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광 데이터 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어 인터넷을 통한 전송)의 형태로 구현하는 것을 포함한다. 또한, 컴퓨터가 읽을 수 있는 기록 매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산 방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다. 그리고 본 발명을 구현하기 위한 기능적인(functional) 프로그램, 코드 및 코드 세그먼트들은 본 발명이 속하는 기술 분야의 프로그래머들에 의하여 용이하게 추론될 수 있다.

[0090] 이상에서 본 발명에 대하여 그 다양한 실시예들을 중심으로 살펴보았다. 본 발명에 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

부호의 설명

[0091] 110 : 가상 머신 120 : 물리 기계/자원(서버)

210 : 관리 도메인(도메인-0) 220 : 게스트 도메인(도메인-U)

310 : 관리 도메인 321, 322 : 게스트 도메인

700 : 가상화 환경에서 프로세서를 할당하는 장치

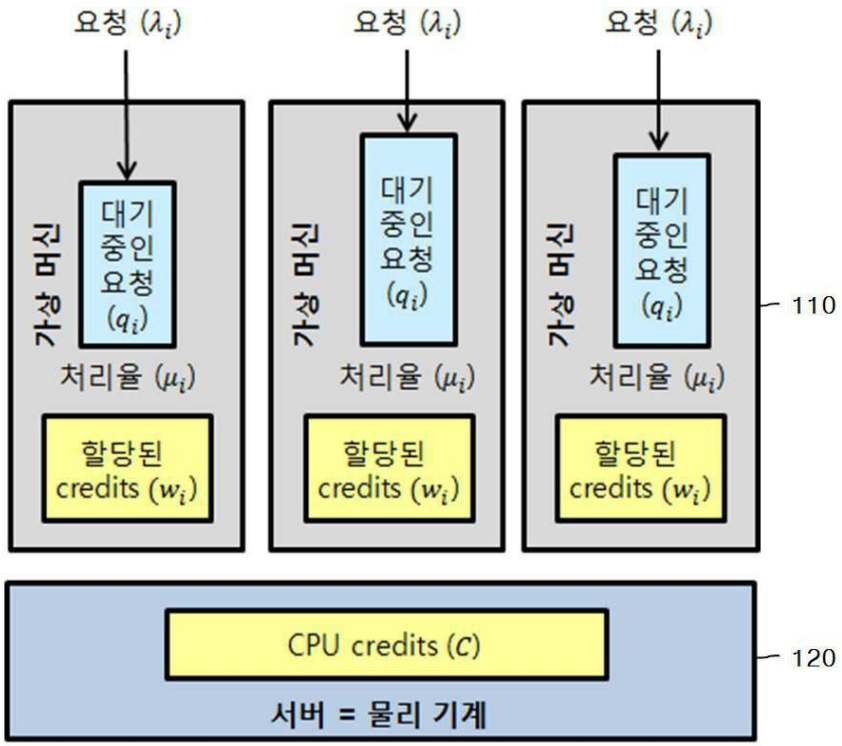
10 : 게스트 도메인 20 : 관리 도메인

30 : 프로세서 40 : 제어부

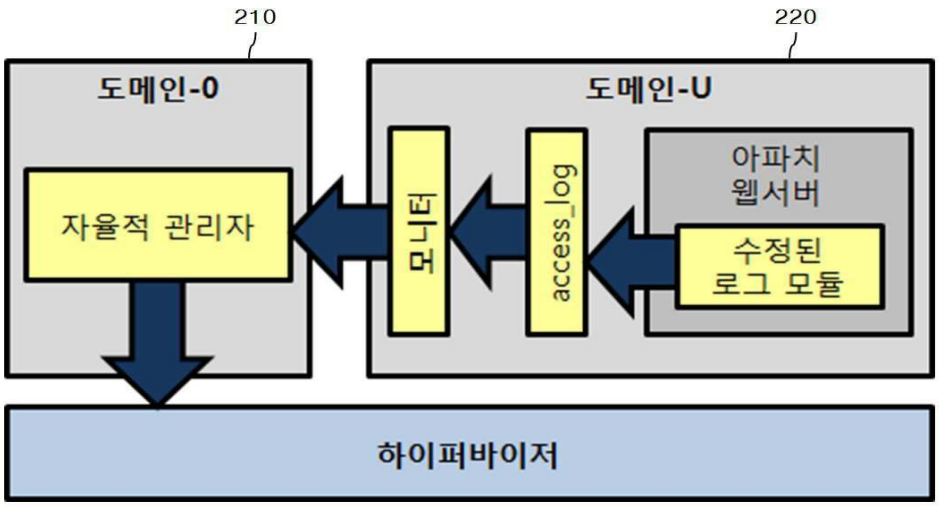
50 : 프로세서 할당부

도면

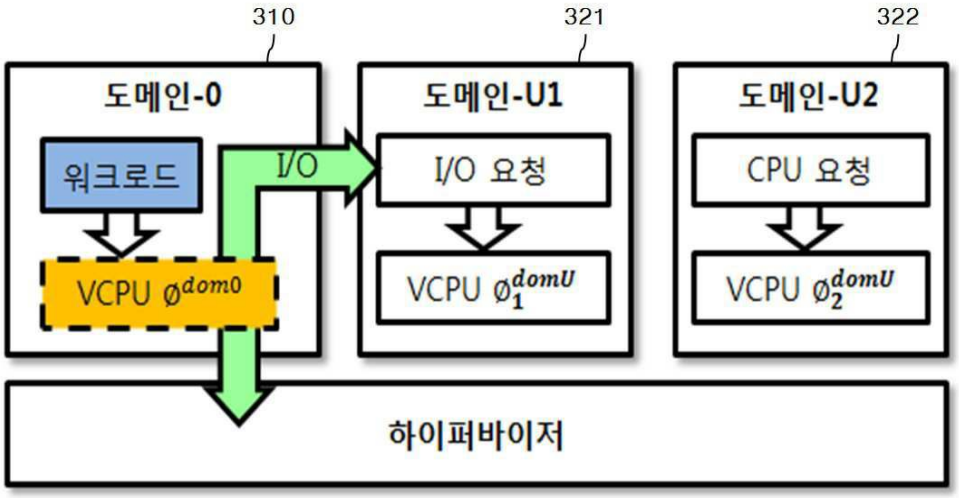
도면1


요청 (λ_i)
요청 (λ_i)
요청 (λ_i)
가상 머신
대기 중인 요청 (q_i)
처리율 (μ_i)
할당된 credits (w_i)
가상 머신
대기 중인 요청 (q_i)
처리율 (μ_i)
할당된 credits (w_i)
가상 머신
대기 중인 요청 (q_i)
처리율 (μ_i)
할당된 credits (w_i)
110
CPU credits (C)
서버 = 물리 기계
120

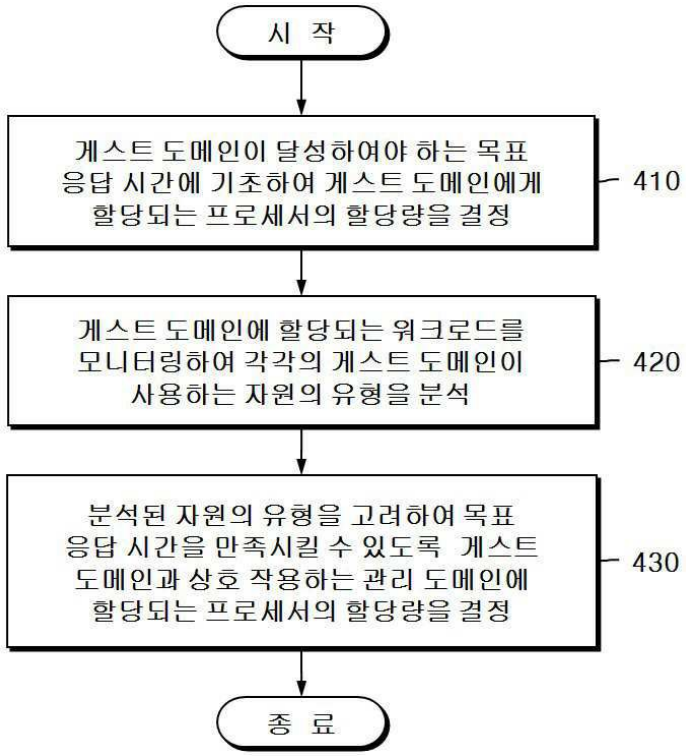
도면2


210
220
도메인-0
도메인-U
자율적 관리자
모니터
access_log
아파치 웹서버
수정된 로그 모듈
하이퍼바이저

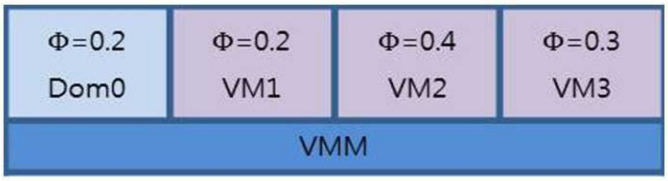
도면3


310
321
322
도메인-0
워크로드
I/O
VCPU $\emptyset^{dom0}$
도메인-U1
I/O 요청
VCPU $\emptyset_1^{domU}$
도메인-U2
CPU 요청
VCPU $\emptyset_2^{domU}$
하이퍼바이저

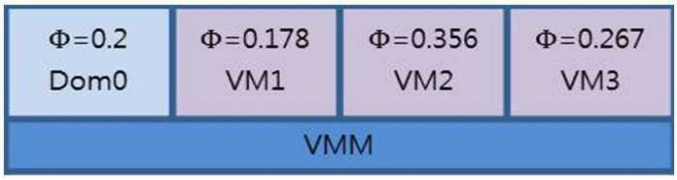
도면4


시 작
게스트 도메인이 달성하여야 하는 목표 응답 시간에 기초하여 게스트 도메인에게 할당되는 프로세서의 할당량을 결정
410
게스트 도메인에 할당되는 워크로드를 모니터링하여 각각의 게스트 도메인이 사용하는 자원의 유형을 분석
420
분석된 자원의 유형을 고려하여 목표 응답 시간을 만족시킬 수 있도록 게스트 도메인과 상호 작용하는 관리 도메인에 할당되는 프로세서의 할당량을 결정
430
종 료

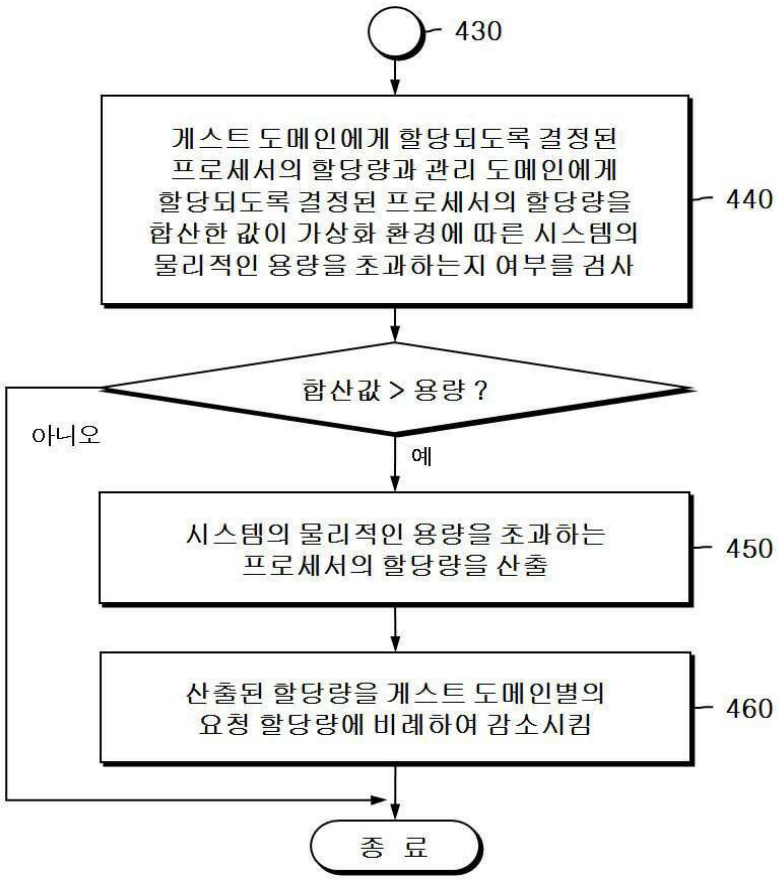
도면5a


Φ=0.2
Dom0
Φ=0.2
VM1
Φ=0.4
VM2
Φ=0.3
VM3
VMM

도면5b


Φ=0.2
Dom0
Φ=0.178
VM1
Φ=0.356
VM2
Φ=0.267
VM3
VMM

도면6


430
게스트 도메인에게 할당되도록 결정된 프로세서의 할당량과 관리 도메인에게 할당되도록 결정된 프로세서의 할당량을 합산한 값이 가상화 환경에 따른 시스템의 물리적인 용량을 초과하는지 여부를 검사
440
합산값 > 용량 ?
아니오
예
시스템의 물리적인 용량을 초과하는 프로세서의 할당량을 산출
450
산출된 할당량을 게스트 도메인별의 요청 할당량에 비례하여 감소시킴
460
종 료

도면7

