(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년10월10일
(11) 등록번호 10-1661783
(24) 등록일자 2016년09월26일

(51) 국제특허분류(Int. Cl.)
G06F 12/02 (2006.01) *G06F 12/06* (2006.01)
G06F 9/44 (2006.01)
(21) 출원번호 10-2011-7022356
(22) 출원일자(국제) 2010년03월19일
심사청구일자 2015년02월17일
(85) 번역문제출일자 2011년09월23일
(65) 공개번호 10-2012-0000066
(43) 공개일자 2012년01월03일
(86) 국제출원번호 PCT/US2010/028034
(87) 국제공개번호 WO 2010/111149
국제공개일자 2010년09월30일
(30) 우선권주장
12/412,272 2009년03월26일 미국(US)
(56) 선행기술조사문헌
US20060184938 A1
US20090307686 A1

(73) 특허권자
마이크로소프트 테크놀로지 라이센싱, 엘엘씨
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
오쉰스 제이콥
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 인터내셔널 페이턴츠 마이크로소프트 코포레이션
(74) 대리인
제일특허법인

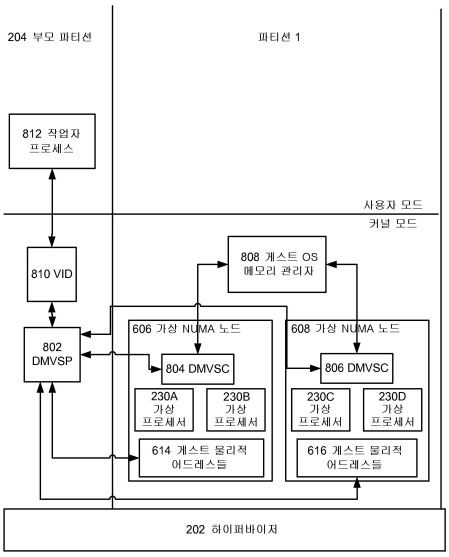
전체 청구항 수 : 총 20 항 심사관 : 임정묵

(54) 발명의 명칭 **가상 머신들을 위한 가상 비균일 메모리 아키텍처**

(57) 요 약

가상 머신을 위한 가상 NUMA 아키텍처를 실시하는 방법과 가상 NUMA 노드들 안의 메모리를 조절하는 방법에 대한 기술들이 서술된다.

대 표 도



명 세 서

청구범위

청구항 1

가상 머신을 인스턴트화(instantiate)시키는 요청을 수신하는 단계 - 상기 요청은 상기 가상 머신에 대한 특성을 포함함 - 와

상기 특성을 기초로 상기 가상 머신에 대한 복수의 가상 NUMA(Non-Uniform Memory Architecture) 노드들을 포함하는 가상 NUMA 노드 토폴로지(topology)를 선택하는 단계와,

컴퓨터 시스템상에서 상기 복수의 가상 NUMA 노드들을 포함하는 상기 가상 머신을 인스턴트화하는 단계와,

상기 복수의 가상 NUMA 노드 중 특정 가상 NUMA 노드 내의 메모리 압력에 기초해서 상기 특정 가상 NUMA 노드에 배정된 게스트 메모리의 용량을 조절하는 단계를 포함하는

방법.

청구항 2

제 1 항에 있어서,

상기 복수의 가상 NUMA 노드들 중 제 2 가상 NUMA 노드 내의 메모리 압력이 사전 결정된 값보다 크다는 것을 판단하는 단계와,

상기 제 2 가상 NUMA 노드를 상기 컴퓨터 시스템의 제 2 NUMA 노드로 이동시키는 단계를 더 포함하는

방법.

청구항 3

제 1 항에 있어서,

상기 게스트 메모리 용량을 조절하는 단계는,

상기 특정 가상 NUMA 노드로부터 게스트 메모리의 적어도 하나의 메모리 블록을 디커밋(de-committing)하는 단계와,

상기 게스트 메모리의 디커밋된 적어도 하나의 메모리 블록을 제2 가상 NUMA 노드에 커밋(committing)하는 단계를 더 포함하는

방법.

청구항 4

제 1 항에 있어서,

상기 게스트 메모리 용량을 조절하는 단계는,

상기 특정 가상 NUMA 노드의 게스트 메모리의 적어도 한 블록이 시스템 메모리와 분리됨을 판단하는 단계와,

상기 게스트 메모리의 적어도 하나의 메모리 블록을 시스템 메모리의 적어도 하나의 메모리 블록에 맵핑하는 단계를 더 포함하는

방법.

청구항 5

제 1 항에 있어서,

상기 특정 가상 NUMA 노드를 상기 컴퓨터 시스템의 제 1 NUMA 노드에 맵핑하는 단계와,

상기 특정 가상 NUMA 노드를 상기 컴퓨터 시스템의 제 2 NUMA 노드로 이동시키는 단계를 더 포함하는

방법.

청구항 6

제 1 항에 있어서,

가상 프로세서를 상기 특정 가상 NUMA 노드에 추가하는 단계를 더 포함하는

방법.

청구항 7

제 1 항에 있어서,

상기 가상 머신의 가상 프로세서의 실행 요청을 수신하는 단계 - 상기 가상 프로세서는 논리 프로세서에 배정되고, 상기 논리 프로세서가 NUMA 노드에 배정되며, 상기 가상 프로세서는 상기 복수의 가상 NUMA 노드들 중 하나에 배정됨 - 와,

상기 논리 프로세서가 상기 가상 프로세서를 실행하지 못함을 판단하는 단계와,

상기 가상 프로세서를 실행할 제 2 NUMA 노드로부터의 제 2 논리 프로세서를 선택하는 단계를 더 포함하는,

방법.

청구항 8

컴퓨터 시스템으로서,

복수의 가상 NUMA 노드들을 포함하는 토폴로지를 가지는 가상 머신을 실행하는 회로 - 상기 가상 머신의 상기 토폴로지는 상기 컴퓨터 시스템의 물리적 토폴로지로부터 독립적으로 생성됨 - 와,

상기 복수의 가상 NUMA 노드들 중 각각의 가상 NUMA 노드 내의 메모리 압력을 판단하는 회로와,

상기 복수의 가상 NUMA 노드들 중 각각의 가상 NUMA 노드 내의 상기 메모리 압력에 기초해서 상기 복수의 가상 NUMA 노드 중 적어도 하나의 가상 NUMA 노드에 배정된 게스트 메모리를 조절하는 회로를 포함하는,

컴퓨터 시스템.

청구항 9

제 8 항에 있어서,

상기 가상 머신을 제 2 컴퓨터 시스템으로 보내는 회로를 더 포함하는

컴퓨터 시스템.

청구항 10

제 8 항에 있어서,

상기 복수의 가상 NUMA 노드들 중 제 1 가상 NUMA 노드를 상기 컴퓨터 시스템의 제 1 NUMA 노드에 맵핑하는 회로와,

상기 복수의 가상 NUMA 노드들 중 제 2 가상 NUMA 노드를 상기 컴퓨터 시스템의 상기 제 1 NUMA 노드에 맵핑하는 회로를 더 포함하는

컴퓨터 시스템.

청구항 11

제 8 항에 있어서,

상기 복수의 가상 NUMA 노드들 중 제 1 가상 NUMA 노드를 상기 컴퓨터 시스템의 제 1 NUMA 노드에 맵핑하는 회로와,

상기 복수의 가상 NUMA 노드들 중 제 2 가상 NUMA 노드를 상기 컴퓨터 시스템의 제 2 NUMA 노드에 맵핑하는 회로를 더 포함하는

컴퓨터 시스템.

청구항 12

제 10 항에 있어서,

상기 제 2 가상 NUMA 노드 내의 메모리 압력이 사전 결정된 값보다 크다는 것을 판단하는 회로와,

상기 제 2 가상 NUMA 노드를 상기 컴퓨터 시스템의 제 2 NUMA 노드로 이동시키는 회로를 더 포함하는

컴퓨터 시스템.

청구항 13

제 11 항에 있어서,

상기 제 2 가상 NUMA 노드의 메모리 압력이 사전 결정된 값보다 크다는 것을 판단하는 회로와,

상기 가상 머신의 상기 제 2 가상 NUMA 노드를 상기 컴퓨터 시스템의 상기 제 1 NUMA 노드로 이동시키는 회로를 더 포함하는

컴퓨터 시스템.

청구항 14

프로세서 실행 가능한 명령어들을 포함하는 컴퓨터 판독 가능 저장 매체에 있어서,

복수의 가상 NUMA 노드들을 포함하는 토폴로지를 가지는 제 1 가상 머신을 실행하는 명령어와,

상기 복수의 가상 NUMA 노드들 중 하나의 가상 NUMA 노드에게 추가적인 가상 프로세서를 추가하는 명령어를 포함하되,

상기 복수의 가상 NUMA 노드들 각각이 가상 프로세서와 게스트 물리적 어드레스들을 포함하고, 상기 가상 머신의 상기 토폴로지는 컴퓨터 시스템의 물리적 토폴로지로부터 독립적으로 생성되는

컴퓨터 판독 가능한 저장 매체.

청구항 15

제 14 항에 있어서,

상기 복수의 가상 NUMA 노드들 각각에서의 메모리 압력을 판단하는 명령어와,

상기 복수의 가상 NUMA 노드들 각각에서의 메모리 압력에 기초해서 복수의 가상 NUMA 노드들 중 적어도 하나의 가상 NUMA 노드에 배정된 게스트 메모리를 조절하는 명령어를 더 포함하는

컴퓨터 판독 가능한 저장 매체.

청구항 16

제 14 항에 있어서,

상기 복수의 가상 NUMA 노드들로부터 가상 프로세서를 제거하는 명령어를 더 포함하는

컴퓨터 판독 가능한 저장 매체.

청구항 17

제 14 항에 있어서,

상기 복수의 가상 NUMA 노드들에 대한 NUMA 비율을 게스트 운영 체제에 보고하는 명령어를 더 포함하는

컴퓨터 판독 가능한 저장 매체.

청구항 18

제 14 항에 있어서,

상기 가상 머신을 제 1 가상 NUMA 노드로부터 복수의 가상 NUMA 노드들로 이동시키는 명령어를 더 포함하는

컴퓨터 판독 가능한 저장 매체.

청구항 19

제 15 항에 있어서,

상기 게스트 메모리를 조절하는 상기 명령어는,

상기 가상 NUMA 노드의 현재 메모리 압력이 목표 임계치보다 낮다는 판단에 기초하여 메모리를 상기 복수의 가상 NUMA 노드 중 제 1 가상 NUMA 노드로부터 디커밋(de-committing)하는 명령어를 더 포함하는

컴퓨터 판독 가능한 저장 매체.

청구항 20

제 15 항에 있어서,

상기 게스트 메모리를 조절하는 상기 명령어는,

게스트 운영 체제의 상기 메모리 압력이 목표 임계치보다 크다는 판단에 기초하여 메모리를 상기 복수의 가상 NUMA 노드 중 제 1 가상 NUMA 노드에 커밋(committing)하는 명령어를 더 포함하는

컴퓨터 판독 가능한 저장 매체.

발명의 설명

배 경 기 술

[0001] 가상화 기술은 복수의 파티션들(partitions)로 하여금 하드웨어 리소스를 공유할 수 있게 하며, 각각의 파티션들은 시스템을 운용하는 게스트를 호스트할 수 있다. 일반적으로, 가상 머신 기술(virtual machine technology)은 서버들을 통합시키고 그들의 이식성(portability)을 증가시킨다. 가상 머신이 커질수록, 그들의 작업량도 늘어나고, 쉽게 통합 및/또는 그들을 한 시스템에서 다른 시스템으로 이동시키는 것이 좀 더 어려워진다. 이에 따라서, 통합 및/또는 더 큰 가상 머신들을 이동시키는 능력을 증가시키는 기술들이 요구된다.

발명의 내용

과제의 해결 수단

[0002] 여기에 개시된 예시적 실시예는 방법을 서술한다. 이 예시에서, 방법은 가상 머신을 인스턴트화(instantiate)시키는 가상 머신에 대한 특성을 포함하는 요청을 수신하는 단계; 특성들을 기초로 가상 머신에 대한 복수의 가상 NUMA 노드들을 포함하는 가상 NUMA 노드 토폴로지(topology)를 선택하는 단계; 컴퓨터 시스템상에서 복수의 가상 NUMA 노드들을 포함하는 가상 머신을 인스턴트화하는 단계; 그리고 복수의 가상 NUMA 노드 중 특정한 것 내의 메모리 압력에 기초해서 특정 가상 NUMA 노드에 배정된 게스트 메모리의 용량을 조절하는 단계를 포함하나 이에 한정되지는 않는다. 상술된 내용에 추가하여, 다른 측면들이 청구항들, 도면들, 그리고 개시된 내용을 일부를 구성하는 텍스트 안에서 서술되어 있다.

[0003] 여기에 개시된 예시적 실시예는 방법을 서술한다. 이 예시에서, 방법은 가상 머신을 실행하는 단계를 포함하나 이에 한정되지는 않고, 이 가상장치는 복수의 가상 NUMA 노드들을 포함하는 토폴로지를 가지고, 가상 머신의 토폴로지가 컴퓨터 시스템의 물리적 토폴로지로부터 독립적으로 생성되고; 복수의 가상 NUMA 노드들 중 각각의 가상 NUMA 노드 내의 메모리 압력을 판단하는 단계; 그리고 복수의 가상 NUMA 노드들 중 각각의 가상 NUMA노드 내의 메모리 압력에 기초해서 복수의 가상 NUMA 노드 중 적어도 하나에 배정된 게스트 메모리를 조절단계를 포함한다. 상술된 내용에 추가하여, 다른 측면들이 청구항들, 도면들, 그리고 개시된 내용을 일부를 구성하는 텍스트 안에서 서술되어 있다.

[0004] 여기에 개시된 예시적 실시예는 방법을 서술한다. 이 예시에서, 방법은 가상 머신을 실행하는 단계를 포함하나 이에 한정되지는 않고, 이 가상장치는 복수의 가상 NUMA 노드들을 포함하는 토폴로지를 가지고, 복수의 가상 NUMA 노드들 각각의 가상 프로세서와 게스트 물리적 어드레스들을 포함하고, 가상 머신의 토폴로지는 컴퓨터 시스템의 물리적 토폴로지로부터 독립적으로 생성되며; 그리고 추가적 가상 프로세서를 복수의 가상 NUMA 노드에 추가하는 단계를 포함한다. 다른 측면들이 청구항들, 도면들, 그리고 개시된 내용을 일부를 구성하는 텍스트 안에서 서술되어 있다.

[0005] 본 기술 분야의 당업자들은 개시된 내용의 하나 또는 그 이상의 다양한 측면들이 전기 회로망 및/또는 여기 개시된 내용의 참조된 측면에 영향을 미치는 프로그래밍을 포함할 수 있으나 그에 한정되지 않는다는 것을 인지할 것이고; 전기 회로망 및/또는 프로그래밍은 시스템 디자이너의 디자인적인 선택에 따라 여기서 참조된 측면들에 영향을 미치도록 설정된 하드웨어, 소프트웨어, 및/또는 펌웨어의 임의의 결합일 수 있음을 인지할 것이다.

[0006] 상술된 내용은 요약에 해당하고 따라서 필요적으로, 개략적인 내용, 일반화된 내용을 포함하고 상세한 내용은 생략된다. 본 기술의 당업자들은 요약은 단지 예시적인 목적에 해당할 뿐 한정적인 의도가 없음을 인지할 것이다.

도면의 간단한 설명

[0007] 도 1은 개시된 내용의 측면들이 구현되는 예시적 컴퓨터 시스템을 도시한다.

도 2는 개시된 내용의 측면들을 실시하기 위한 작업 환경을 도시한다.

도 3은 개시된 내용의 측면들을 실시하기 위한 작업 환경을 도시한다.

도 4는 개시된 내용의 실시예들 안에서 어떻게 메모리가 배열될 수 있는지를 도시한다.

도 5는 개시된 내용의 측면들을 실시하는 예시적 작업 환경을 도시한다.

도 6은 개시된 내용의 측면들을 실시하는 예시적 작업 환경을 도시한다.

도 7은 개시된 내용의 측면들을 실시하는 예시적 작업 환경을 도시한다.

도 8은 개시된 내용의 측면들을 나타내는 예시적 블록 다이어그램을 도시한다.

도 9는 개시된 내용의 측면들을 실시하는 작업 과정을 도시한다.

도 10은 도 9의 과정의 대체적 실시예를 도시한다.

도 11은 개시된 내용의 측면들을 실시하는 작업 과정을 도시한다.

도 12는 도 11의 과정의 대체적 실시예를 도시한다.

도 13은 도 12의 과정의 대체적 실시예를 도시한다.

도 14는 도 12의 과정의 대체적 실시예를 도시한다.

도 15는 개시된 내용의 측면들을 실시하는 작업 과정을 도시한다.

도 16은 도 15의 과정의 대체적 실시예를 도시한다.

도 17은 도 16의 과정의 대체적 실시예를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0008] 실시예들은 하나 또는 그 이상의 컴퓨터들에서 실행될 수 있다. 도 1 과 뒤따르는 논의들은 개시된 내용이 구현될 수 있는 적절한 컴퓨터 환경의 간결하고 일반적인 설명을 제공하기 위한 것이다. 본 기술의 당업자는 도 1의 컴퓨터 시스템이 일부 실시예들 내에서 컴퓨터 시스템들(200, 300, 600, 그리고 700)을 달성할 수 있음을 인지할 것이다. 이들 예시적 실시예들에서, 컴퓨터 시스템은 도 1에서 도시되어 있는 요소들 중 일부 또는 전부 그리고 개시된 발명의 측면들을 인스턴트화(instantiate)하도록 설정된 전기 회로망을 포함할 수 있다.

[0009] 개시된 내용에서 사용된 회로망(circuitry) 이라는 용어는 하드웨어 중단 제어부(hardware interrupt controller), 하드 드라이브, 네트워크 어댑터, 그래픽 프로세서들, 하드웨어 기반 비디오/오디오 코덱들, 그리고 하드웨어 운용을 위해 사용되는 펌웨어/소프트웨어와 같은 하드웨어 요소들을 포함할 수 있다. 동일한 또는 다른 실시예들에서 용어 회로망은 펌웨어들 또는 특정하게 설정된 스위치들에 의해 함수(들)를 수행하도록 설정된 마이크로프로세서들을 포함할 수 있다. 동일한 또는 다른 실시예들에서 용어 회로망은 하나 또는 그 이상의 논리 프로세서들을 포함할 수 있고, 그 예로는 하나 또는 그 이상의 코어들을 가진 멀티코어 일반적 프로세싱 유닛(multi-core general processing unit)이 있다. 예시 내의 논리적 프로세서(들)는 RAM, ROM, 펌웨어, 및/또는 가상 메모리 등에서 로딩 되는 함수(들)를 수행하기 위해 운용되는 로직으로 구현되는 소프트웨어 명령들에 의해 설정될 수 있다. 회로망이 소프트웨어와 하드웨어의 결합을 포함하는 예시적 실시예들에서, 구현자는 로직을 구현하는 소스 코드를 작성할 수 있고, 그 뒤에 소스코드는 논리적 프로세서에 의해 프로세스 될 수 있는 장치 판독 가능 코드로 컴파일된다. 본 기술의 당업자 시선에서는 하드웨어, 소프트웨어, 또는 하드웨어/소프트웨어의 결합은 큰 차이가 없기 때문에, 기능들을 달성하기 위한 하드웨어와 소프트웨어의 선택은 단지 디자인적인 선택에 해당한다. 따라서, 본 기술의 당업자는 소프트웨어 프로세스가 그와 등가의 하드웨어 아키텍처로 변환할 수 있고, 하드웨어 아키텍처 그 자신도 등가의 소프트웨어 프로세스로 변환될 수 있고, 하드웨어적 구현과 소프트웨어적 구현의 선택은 구현자에게 맡긴 디자인적인 선택임을 인지할 것이다.

[0010] 도 1에 있어서, 예시적 일반 목적 컴퓨터 시스템(general purpose computing system)이 도시된다. 일반 목적 컴퓨터 시스템은 통상적 컴퓨터(20) 또는 이와 유사한 것을 포함하고, 통상적 컴퓨터는 논리 프로세서(21), 시스템 메모리(22), 그리고 시스템 메모리를 포함하는 다양한 시스템 요소들을 논리 프로세서(21)에 결합시키는 시스템 버스(23)를 포함한다. 시스템 버스(23)는 메모리 버스 또는 메모리 제어부, 주변부 버스, 그리고 다양한 버스 아키텍처들 중 임의의 것을 사용하는 로컬 버스를 포함하는 여러 종류의 버스 아키텍처들 중 임의의 것일 수 있다. 시스템 메모리는 ROM(24) 그리고 RAM(25)을 포함할 수 있다. 스타트 업(start-up) 과정 도중과 같이 컴퓨터(20) 안에서 요소들 사이에 정보 이동을 도와주는 기본 루틴(basic routines)을 포함하는 BIOS(basic input/output system)(26)는 ROM(24) 안에 저장된다. 컴퓨터(20)는 더 나아가 하드 디스크(미도시)에 기록하고 하드 디스크로부터 읽기 위한 하드 디스크 드라이브(27), 이동식 자기 디스크(29)로부터 읽고 쓰기 위한 자기 디스크 드라이브(28), CD-ROM 또는 다른 광학 매체와 같은 이동식 광학 디스크(31)로부터 읽고 쓰기 위한 광학

디스크 드라이브(30)들을 포함할 수 있다. 하드 디스크 드라이브(27), 자기 디스크 드라이브(28), 그리고 광학 디스크 드라이브(30)는 각각 하드 디스크 드라이브 인터페이스(32), 자기 디스크 드라이브 인터페이스(33), 그리고 광학 디스크 인터페이스(34)와 연결된 상태로 나타난다. 드라이브들과 그들과 결부된 컴퓨터 판독 가능 저장 매체는 컴퓨터 판독 가능 명령들의 비휘발성 저장부, 데이터 아키텍처들, 프로그램 모듈들 그리고 컴퓨터(20)에 대한 다른 데이터를 제공한다. 여기서 서술된 예시적 실시예는 하드 디스크, 이동식 자기 디스크(29) 그리고 이동식 광학 디스크(31)에 구현되지만, 본 기술의 당업자 기준에서는 자기 카세트, 플래시 메모리 카드, DVD, Bernoulli 카트리지, RAM, ROM 그리고 유사한 것들과 같은 컴퓨터에 의해 접속 가능한 데이터들을 저장할 수 있는 다른 종류의 컴퓨터 판독 가능 저장 매체들도 예시적 운용 환경에서 사용될 수 있다. 일반적으로, 특정 실시예들에서는, 이런 컴퓨터 판독 가능 저장 매체는 개시된 발명의 측면들에서 구현되는 프로세서 실행가능 명령들을 저장하는데 사용될 수 있다.

[0011] 운영 체제(35), 하나 또는 그 이상의 애플리케이션 프로그램들(36), 다른 프로그램 모듈들(37) 그리고 프로그램 데이터(38) 등을 포함하는 프로그램 모듈들이 하드 디스크, 자기 디스크(29), 광학 디스크(31), ROM(24) 또는 RAM(25)에 저장될 수 있다. 사용자는 키보드(40)와 포인팅 장치(pointing device)(42)와 같은 입력 장치들을 통해서 컴퓨터(20)에 명령들과 정보를 입력할 수 있다. 다른 입력 장치들(미도시)은 마이크로폰, 조이스틱, 게임 패드, 위성 디스크, 스캐너 또는 그와 유사한 것을 포함할 수 있다. 이들과 다른 입력 장치들은 종종 시스템 버스와 결합된 직렬 포트 인터페이스(serial port interface)(46)를 통해서 논리 프로세서(21)가 연결되나, 병렬 포트, 게임 포트 또는 USB 등의 다른 인터페이스들과 연결될 수도 있다. 디스플레이(47) 또는 다른 종류의 디스플레이 장치도 비디오 어댑터(48)와 같은 인터페이스를 통해서 시스템 버스(23)와 연결될 수 있다. 디스플레이(47)에 추가로, 컴퓨터들은 통상 스피커들과 프린터들과 같은 다른 주변부 출력 장치들(미도시)을 포함한다. 도 1 의 예시적 시스템 또한 호스트 어댑터(55), SCSI(Small Computer System Interface) 버스(56), 그리고 SCSI 버스(56)에 연결된 외부 저장 장치(62)를 포함한다.

[0012] 컴퓨터(20)는 원격 컴퓨터(49)와 같은 하나 또는 그 이상의 원격 컴퓨터들과의 논리적 연결들을 사용하는 네트워크 환경 안에서 운용될 수 있다. 원격 컴퓨터(49)는 다른 컴퓨터, 서버, 라우터, 네트워크 PC, 대등 장치 또는 다른 일반적인 네트워크 노드(common network node) 등을 포함하고, 도 1에서는 메모리 저장 장치(50)만 도시되어 있지만 일반적으로 컴퓨터(20)과 관련되는 상술 된 요소들 전부 또는 다수를 포함할 수 있다. 도 1 에 도시된 논리 연결부들은 LAN(51)과 WAN(52)을 포함할 수 있다. 이런 네트워크 환경은 사무실들, 사내 전산망, 인트라넷들 그리고 인터넷에서 흔하다.

[0013] LAN 네트워크 환경에서 사용되는 경우에, 컴퓨터(20)는 LAN(51)과 네트워크 인터페이스 또는 어댑터(53)를 통해서 연결될 수 있다. WAN 네트워크 환경에서 사용되는 경우에, 컴퓨터(20)는 일반적으로 모뎀(54) 또는 인터넷과 같이 WAN(52) 전역에서 통신을 설정하기 위한 다른 수단들을 포함할 수 있다. 모뎀(54)은, 내장형 또는 외장형일 수 있고, 직렬 포트 인터페이스(46)를 통해서 시스템 버스(23)와 연결될 수 있다. 네트워크 환경 안에서, 프로그램 모듈들은 컴퓨터(20) 또는 그 일부와 관련해서 도시되어 있고, 원격 메모리 저장 장치 내에 저장될 수 있다. 도시된 네트워크 환경은 예시적일 뿐이고, 컴퓨터들 사이의 통신 링크를 설정하기 위해서 다른 수단들이 사용될 수도 있다. 더 나아가, 개시된 발명의 다수의 실시예들이 특히 컴퓨터화된 시스템들에 부합되지만, 이에 한정되지는 않는다.

[0014] 도 2와 도 3을 참조하면, 그들은 컴퓨터 시스템들의 하이 레벨 블록 다이어그램(high level block diagram)을 도시한다. 도면을 통해서 볼 수 있듯이, 컴퓨터 시스템(200)은 저장 장치(208), 예를 들면 하드 드라이브, 네트워크 인터페이스 제어부(210), 그래픽 카드(234), 적어도 하나의 논리 프로세서(212), RAM(214)와 같은 물리적 하드웨어 장치들을 포함할 수 있다. 컴퓨터 시스템(200)은 또한 도 1의 컴퓨터(20)과 같은 유사한 요소들을 포함할 수도 있다. 하나의 논리 프로세서가 도시되어 있지만, 컴퓨터 시스템(200)의 다른 실시예들은 복수의 논리 프로세서들을 가질 수 있고, 예를 들면 복수 실행 코어 퍼 프로세서(multiple execution cores per processor) 및/또는 각각이 복수 실행 코어를 가지는 복수 프로세서들을 가질 수 있다. 계속해서 도 2에 대하여 서술하면, 도시된 것은 가상 머신 모니터라고도 불리는 하이퍼바이저(hypervisor)(202)이다. 도시된 실시예에서 하이퍼바이저(202)는 컴퓨터 시스템(200)의 하드웨어로의 접속을 제어하고 중재하는 실행가능한 명령들을 포함한다. 대략적으로, 하이퍼바이저(202)는 자식 파티션(child partition) 1부터 자녀 파티션 N(여기서 N은 1보다 큰 정수)과 같은 파티션이라고 지칭되는 실행 환경들을 생성할 수 있다. 실시예들에서 자식 파티션은 하이퍼바이저(202)에 의해 지지되는 분리된 기본 단위로 여겨지고, 즉 각각의 자식 파티션은 하이퍼바이저(202) 및/또는 부모 파티션의 감독 아래 메모리, 장치들, 논리 프로세서 사이클들 등의 하드웨어 리소스들의 집합에 맵핑(map)될 수 있다. 실시예들 안에서 하이퍼바이저(202)는 독립된 소프트웨어 제품, 운영 체제의 일부, 마더보드 펌웨어에

내장, 특수 통합 회로들, 또는 그들의 조합일 수 있다.

[0015] 도시된 예시인 컴퓨터 시스템(200)은 오픈 소스 공동체(open source community) 안에서 도메인 0으로 취급될 수도 있는 부모 파티션(204)을 포함한다. 부모 파티션(204)은 VSPs(virtualization service providers)(228)를 사용해서 자식 파티션들 1-N 안에서 실행되는 게스트 운영 체제들에게 리소스를 제공하도록 설정될 수 있고, VSPs는 오픈 소스 공동체 안에서 백 앤드 드라이버들(back-end drivers)로 알려져 있기도 하다. 이 예시적 아키텍처 안에서, 부모 파티션(204)은 근원적 하드웨어(underlying hardware)에 게이트 접속을 할 수 있다. 대략적으로, VSPs(228)은 오픈 소스 공동체 안에서는 프런트 앤드 드라이버(front-end drivers)로 알려져 있기도 한 VSCs(virtualization service clients)를 통해서 인터페이스들을 하드웨어 리소스들로 다중 송신(multiplex)하는데 사용될 수 있다. 각각의 자식 파티션은 가상 프로세서들(230-232)과 같은 하나 또는 그 이상의 가상 프로세서들을 포함할 수 있고, 이때 게스트 운영 체제들(220-222)이 그들에게서 실행되는 스레드들을 관리하고 스케줄 할 수 있다. 일반적으로, 가상 프로세서들(230-232)은 실행가능한 명령들과 물리적 프로세서의 표현에 구체적인 아키텍처를 제공해주는 결부된 상태 정보이다. 예를 들면, 하나의 가상 머신은 인텔 x86 프로세서의 특성들을 가지는 가상 프로세서를 가질 수 있고, 다른 가상 프로세서는 PowerPC 프로세서의 특성들을 가질 수 있다. 이 예시 내의 가상 프로세서들은 컴퓨터 시스템의 논리 프로세서들과 맵핑되어 가상 프로세서들을 실행하는 명령들이 논리 프로세서들에 의해서 뒷받침될 수 있다. 따라서, 이들 예시적 실시예들에서, 복수의 가상 프로세서들은 다른 논리 프로세서가 하이퍼바이저 명령들을 실행하는 동안에 동시 병행적으로 실행될 수 있다. 일반적으로, 그리고 도면에 도시된 바와 같이, 가상 프로세서들, 다양한 VSCs, 그리고 파티션 내 메모리들의 조합은 가상 머신(240 또는 242)와 같은 가상 머신로 간주될 수 있다.

[0016] 일반적으로, 게스트 운영 체제들(220-222)은 Microsoft, Apple, 오픈 소스 공동체 등의 운용시스템을 포함하는 어떤 운용시스템도 포함할 수 있다. 게스트 운영 체제들은 작업의 사용자/커널 모드들을 포함할 수 있고 스케줄러들, 메모리 관리자들 등을 포함할 수 있는 커널 등을 가질 수 있다. 각각의 게스트 운영 체제(220-222)는 애플리케이션들이 저장된 e-커머스 서버들, 이메일 서버들 등의 결부된 파일 시스템들과 게스트 운영 체제들 그 자세들을 가질 수 있다. 게스트 운영 체제들(220-222)은 가상 프로세서들(230-232)에서 실행되는 스레드들의 일정을 잡을 수 있고 그런 애플리케이션들의 인스턴트화가 실시될 수 있다.

[0017] 도 3을 참조하면, 도 3은 사용 가능한 대체적 아키텍처를 도시한다. 도 3은 도 2와 유사한 요소들을 도시하나, 이 예시에서 하이퍼바이저(202)는 VSPs(228) 및 장치 드라이버들(224), 그리고 설정 유틸리티들(236)을 포함할 수 있는 부모 파티션을 포함할 수 있다. 이 아키텍처의 하이퍼바이저(202)는 도 2의 하이퍼바이저(202)와 유사 또는 동일한 기능들을 수행할 수 있다. 도 3의 하이퍼바이저(202)는 독립된 소프트웨어 제품, 운영 체제의 일부, 마더보드의 펌웨어 내에 내장 또는 특정 통합 회로에 의해 유발되는 하이퍼바이저의 일부일 수 있다. 이 예시적 부모 파티션(204)은 하이퍼바이저(202)를 설정하는데 사용될 수 있는 명령들을 가질 수 있으나, 하드웨어 접속 요청들은 부모 파티션(204)에 넘겨지지 않고 하이퍼바이저(202)에 의해 처리될 수도 있다.

[0018] 도 4를 참조하면, 도 4는 가상 머신들을 포함하는 실시예들 내에서 메모리가 정렬되는 방법을 도시한다. 예를 들면, 컴퓨터 시스템(200)과 같은 컴퓨터 시스템은 메모리 어드레스들을 포함하는 RAM(214)을 가질 수 있다. 시스템 물리적 메모리 어드레스들을 가상 머신들에 보고하는 대신에, 하이퍼바이저(202)는 GPAs(guest physical addresses) 등의 시스템 물리적 어드레스들을 위한 다른 어드레스들을 게스트 운영 체제들의 메모리 관리자들에게 나타낼 수 있다. 그리고 나서 게스트 운영 체제들은 GPAs를 제어하고 하이퍼바이저(202)는 GPAs와 SPAs에 의한 관계들을 유지한다. 도면에 나타난 바와 같이, 실시예들에서 GPAs와 SPAs는 메모리 블록들로 배열될 수 있다. 대략적으로, 메모리 블록은 하나 또는 그 이상의 메모리 페이지를 포함할 수 있다. GPAs와 SPAs 사이의 관계는 "Enhanced Shadow Page Table Algorithms"로 명명된 미국 특허 출원서 No. 11/128,665에서 도시되어 있는 것과 같은 그림자 페이지 표(shadow page table)에 의해 유지될 수 있고, 그 내용 전부가 여기서 참조로 포함된다. 작업 중에, 게스트 운영 체제가 블록 1의 GPA에 데이터를 저장하는 경우에, 데이터는 시스템상의 블록 6과 같이 실제로는 다른 SPA에 저장될 수 있다.

[0019] 개략적으로 도 5는 개시된 발명의 측면들을 실시하기 위한 운용 환경을 도시한다. 예를 들면, 일부 컴퓨터 시스템들(504-510)은 데이터 센터(500) 안에서 결합될 수 있다(네 개의 컴퓨터들이 도시되어 있지만, 본 기술의 당업자는 더 적은 또는 더 많은 수의 컴퓨터 시스템들을 포함할 수 있음을 인지할 것이다). 도시된 컴퓨터 시스템들은 다른 토폴로지들(topologies)을 가질 수 있고 더 나아가, 다른 양의 RAM, 다른 RAM 속도 다른 양의 논리 프로세서들, 및/또는 다른 속도를 가진 논리 프로세서들과 같은 다른 특성들을 가질 수 있다.

[0020] 관리 시스템(502)은 도 1의 컴퓨터 시스템(20)과 유사한 요소들, 및/또는 컴퓨터 시스템들(200, 300, 600 또는

700)을 포함할 수 있다. 이것은, 이 실시예 내에서 관리 시스템(502)은 도 6 또는 도 7에 대하여 후술 되는 내용을 포함하는 컴퓨터 시스템일 수 있다.

[0021] 도면들과 관련해서 개략적인 개요를 계속하자면, 도 6은 SMP(symmetric multiprocessing topology) 또는 "flat" 토폴로지를 가지는 컴퓨터 시스템(600)을 도시한다. 일반적으로, SMP는 단일 공유 메모리에 연결된 복수의 프로세서들을 포함하는 컴퓨터 아키텍처이다. 이런 배열 내에서, 메모리 제어부(602)는 메모리에 송/수신되는 데이터의 흐름을 관리할 수 있다. 메모리 접속은 각각의 논리 프로세서(212A-F)에 대하여 균일할 수 있고, 각각의 논리 프로세서는 메모리의 전 범위를, 즉 시스템 물리적 어드레스들 622-632, 접속할 수 있다. 이 토폴로지는 상대적으로 적은 수의 프로세서들을 가지는 컴퓨터 시스템 내에서는 주효하나, 모두가 공유된 메모리 버스에 접속하기 위해 경쟁하는 많은 수의 프로세서들을 포함하는 컴퓨터 시스템의 경우에는 시스템의 퍼포먼스가 감소할 수 있다. 더 나아가, 컴퓨터 시스템의 복잡성의 큰 증가는 결국 프로세서 당 가격을 상승시킨다.

[0022] 간략하게, 컴퓨터 시스템(600)은 컴퓨터(200 또는 300)와 동일 또는 유사한 요소들을 포함할 수 있다. 도면에서 볼 수 있듯이, 컴퓨터 시스템(600)은 RAM(214)으로의 접속을 개시해주는 메모리 제어부(602)를 통해서 결합된 복수의 논리 프로세서들(212A-212F)을 가질 수 있다. 상술 된 것과 유사하게, 각각의 논리 프로세서들(212A-212F)은 클록 속도들, 캐시 크기 등의 다른 특성들을 가질 수 있다. 이런 배열에서 메모리 제어부(602)는 RAM(214)이 송수신하는 데이터의 흐름을 관리할 수 있다.

[0023] 하이퍼바이저(202)가 예시적으로 제시될 수 있고, 하이퍼바이저(202)는 컴퓨터 시스템(600)의 하드웨어를 제어할 수 있다. 하이퍼바이저(202)는 하나 또는 그 이상의 가상 머신들(240-242)을 관리할 수 있고, 각각의 가상 머신들은 가상 NUMA 노드들(606-612)과 같은 가상 NUMA 노드들을 가질 수 있다. 가상 NUMA 노드들(606-612)은 가상 토폴리지들을 게스트 애플리케이션 또는 게스트 운영 체제들(220, 222)과 같은 게스트 운영 체제에 보고하는 방식으로 가상 머신의 리소스들을 체계화하는데 사용될 수 있다. 도면에서 볼 수 있듯이, 각각의 가상 NUMA 노드(606-612)는 하나 또는 그 이상의 가상 프로세서들(230A-D, 232A-D)과 게스트 물리적 어드레스들(614-616, 그리고 618-620)을 가질 수 있다. 일반적으로, 하이퍼바이저(202)는 각각의 가상 NUMA 노드(606-612)를 하나 또는 그 이상의 논리 프로세서들과 RAM(214)으로부터의 시스템 물리적 주소들로 뒷받침해줄 수 있다. 이것은 하이퍼바이저(202)가 하나 또는 그 이상의 논리 프로세서들을 가상 프로세서 스레드들을 실행하는데 사용될 수 있는 이상적 프로세서들로 설정할 수 있다는 의미이다.

[0024] 간략하게, 도 7은 NUMA 노드들(702-706)을 포함하는 토폴로지를 가지는 컴퓨터 시스템(700)을 도시한다. NUMA노드들을 가지는 컴퓨터 시스템들은 일반적으로 작은 컴퓨터 시스템들로 이루어진 컴퓨터들로 취급된다. 이 예시에서, 각각의 NUMA노드(606-612)는 하나 또는 그 이상의 논리 프로세서들과 로컬 메모리(local memory)를 포함할 수 있다. NUMA 노드의 내부 메모리는 로컬 메모리로 취급되고 NUMA 노드의 그 외의 메모리는 원격 메모리로 취급되며, 이는 오직 노드 내부의 프로세서들만이 동일한 메모리 버스와 연결되기 때문이다. NUMA 노드들은 캐시 상관성 도메인 상호연결부(cache coherency domain interconnetcts)에 의해 상호 연결되며, 캐시 상관성 도메인 상호 연결부는 하나의 NUMA 노드 내의 프로세서들로 하여금 다른 NUMA 노드들에 있는 메모리들을 상관성 방법(coherent way)에 의해 접속가능하게 한다. 따라서, 시스템 물리적 어드레스들(622-632)은 각각의 프로세서들에 대하여 균일하다. 또는 다른 말로 표현하면, 시스템 물리적 어드레스 20,000은 컴퓨터 시스템 내의 모든 프로세서들에 있어서 동일하다. 차이점은 일부 프로세서 메모리 어드레스 20,000은 로컬 메모리 어드레스이지만, 예를 들면 그들의 NUMA 노드 내부, 다른 프로세서 메모리 어드레스 20,000은 원격, 예를 들면 그들의 NUMA 노드 밖, 이라는 점이다. 일반적으로, 로컬 메모리는 원격 메모리에 비해 빠르게 접속될 수 있고 로컬 메모리와 원격 메모리의 접속 시간의 관계를 NUMA 비율이라고 부른다. NUMA 비율이 1 에서 2인 것은 로컬 시스템 물리적 어드레스에 비해 특정 원격 시스템 물리적 어드레스에 접속하는 경우에 두 배의 프로세서 사이클들이 요구된다는 것을 의미한다. NUMA 는 SMP 시스템에 의해 유발된 병목 현상을 어떤 하나의 메모리 버스 상의 프로세서 수를 제한하는 방식에 의해 완화시키고, 동일한 수량의 논리 프로세서들을 가지는 SMP 컴퓨터 시스템에 비해서 저렴하다.

[0025] 컴퓨터 시스템(700)은 컴퓨터(200 또는 300)와 동일 또는 유사한 요소들을 포함할 수 있다. 도면에서 볼 수 있듯이, 이 운용 환경 컴퓨터 시스템(700)은 상호 연결부(708)에 의해 연결된 세 개의 NUMA 노드들(702-706)(컴퓨터는 더 많은 또는 더 적은 수의 노드들을 가질 수도 있다)을 포함한다. 도면에서 볼 수 있듯이, 각각의 NUMA 노드들 내의 프로세서들의 수는 가변적이고 각각의 노드는 그들 고유의 RAM을 가질 수 있다.

[0026] 도 7과 유사하게, 하이퍼바이저(202)는 컴퓨터 시스템(700)의 하드웨어를 제어할 수 있다. 게스트 운영 체제들 또는 단일체 어플리케이션(monolithic application)을 부팅할 때, 그들은 상술 된 것과 유사한 가상 머신들(240

그리고 242)의 토폴로지들을 감지할 수 있다. 각각의 가상 NUMA 노드(606-612)는 하나 또는 그 이상의 이상적 프로세서들(ideal processors)과 가상 프로세서들의 스레드들을 실행하는데 사용될 수 있는 동일한 NUMA 노드로부터의 시스템 물리적 어드레스들에 배정될 수 있다.

[0027] 컴퓨터 시스템(600 그리고 700)은 두 개의 가상 머신들(240 그리고 242)을 포함하는 것으로 도시되어 있지만, 다른 실시예들에서 그들은 더 많거나 더 적은 가상 머신들을 실행할 수 있다. 더 나아가, 각각의 가상 머신은 두 개의 가상 NUMA 노드들을 가지는 것으로 도시되어 있지만, 다른 실시예들에서 가상 머신들은 더 많거나 더 적은 NUMA 노드들을 가질 수 있다. 또한, 가상 NUMA 노드들이 두 개의 가상 프로세서들을 가지는 것으로 도시되어 있지만, 다른 실시예들에서 가상 NUMA 노드들은 더 많거나 더 적은 가상 프로세서들을 가질 수 있다. 더 나아가, 각각의 가상 NUMA 노드는 다른 가상 NUMA 노드들과는 다른 토폴로지를 가질 수 있고, 예를 들면 한 가상 NUMA 노드는 네 개의 가상 프로세서들과 8기가바이트의 RAM을 가질 수 있고 다른 가상 NUMA 노드는 두 개의 가상 프로세서들과 4기가바이트의 RAM을 가질 수 있다.

[0028] 도 8은 개시된 발명의 측면들에서 사용될 수 있는 블록 다이어그램 환경을 도시한다. 도면에서 볼 수 있듯이, DMVSP(dynamic memory virtualization service provider)(802)로 알려진 가상 머신에 배정되는 메모리를 관리하는 요소가 도시되고 이것은 가상 NUMA 노드에 접속 가능한 메모리의 용량을 조절하는데 사용될 수 있다. 도면에서 볼 수 있듯이, DMVSP(802)는 하나 또는 그 이상의 벌루닝 드라이버들(ballooning drivers)과 결부될 수 있고, 이들은 가상화 서비스 클라이언트들, 즉 DMVSCs(dynamic memory virtualization service clients)(804 및/또는 806)로 알려져 있기도 하다(가상 NUMA 노드 당 하나의 DMVSC가 도시되어 있으나 다른 실시예들에서는 파티션 당 하나의 DMVSC가 사용될 수도 있다). 개략적으로, DMVSCs(804 및/또는 86)는 DMVSP(802)에 의해 가상 NUMA 노드의 메모리를 조절하는데 사용될 수 있는 정보를 제공할 수 있고 각각의 DMVSC는 결부된 가상 NUMA 노드에 대해서 메모리를 커밋(commit) 또는 디커밋(de-commit)하는 것을 도울 수 있다. DMVSCs(804, 806) 그리고 DMVSP(802)는 'Partition Bus"로 명명된 미국 특허 출원 번호 11/128,647에서 개시된 가상화 버스를 통해서 통신할 수 있고, 이 미국 특허 출원의 내용은 그 내용 전부가 참조로 포함된다. 더 나아가, DMVSCs 그리고 DMVSPs의 추가적 측면들이 "Dynamic Virtual Machine Memory"로 명명된 미국 특허 출원 번호 12/345,469에서 설명되어 있고, 이 미국 특허 출원의 내용은 그 내용 전부가 참조로 포함된다.

[0029] 계속해서 도 8에 대하여 서술하면, 시스템은 자식 파티션(들)을 관리할 수 있는 작업자 프로세스(worker process)(812)를 포함할 수 있다. 작업자 프로세스(812)는 VID(virtualization infrastructure driver)(810)와 함께 구동될 수 있고, VID(810)는 메모리를 자식 파티션에 할당할 수 있다. 예를 들면, VID(810)는 게스트 물리적 어드레스들과 시스템 물리적 어드레스들 사이의 관계들을 설정하거나 제거할 수 있다. 도 8은 또한 메모리 관리자(808)를 포함할 수 있는 게스트 운영 체제(220)와 같은 게스트 운영 체제를 포함할 수 있는 파티션을 도시한다. 일반적으로, 메모리 관리자(808)는 그들의 요청에 의해 메모리를 애플리케이션들에 할당할 수 있고 더 이상 애플리케이션들에게 필요가 없는 경우에는 메모리를 할당 제거할 수 있다.

[0030] 뒤따르는 내용들은 프로세스들의 구현들을 도시하는 플로우 차트들의 시리즈이다. 이해 목적을 위해서, 플로우 차트들은 초기 플로우 차트들은 전반적 "큰 그림" 관점에서 구현예들을 나타내고, 뒤따르는 플로우 차트들은 추가적이 상세 사항 등을 제공한다. 더 나아가, 본 기술의 당업자는 일점 쇄선으로 도시된 운용 방법은 선택적인 것으로 간주한다.

[0031] 도 9는 작업들(900-910)을 포함하는 개시된 내용의 측면들을 실시하는 운용 방법을 도시한다. 운용 방법은 작업 900에서 시작하고 작업 902는 가상 머신을 인스턴트화하는 요청을 수신하는 단계를 도시하며, 요청은 가상 머신에 대한 특성을 포함한다. 예를 들면, 도 6 또는 도 7을 참조하면, 하이퍼바이저(202)는 가상 머신(240)와 같은 가상 머신을 생성하는 요청을 수신할 수 있다. 예를 들면, 요청은 관리 시스템(502), 도 2 또는 3의 부모 파티션(204)으로부터 수신될 수 있다. 요청은 새로운 가상 머신에 대한 것 또는 이전에 저장된 가상 머신을 인스턴트화 하는것 일 수 있다. 가상 머신(240)가 새로운 가상 머신인 경우에, 가상 머신의 특성들, 예를 들면 가상 머신에 배정된 RAM의 크기, 가상 프로세서들의 수, 또는 가상 머신에 적합한 입력 장치들의 종류 등이 예를 들면 관리자에 의해 설정될 수 있다.

[0032] 계속해서 도 9에 대해서 서술하면, 작업 904는 특성을 기초로 가상 머신을 위한 가상 NUMA 노드 토폴로지를 선택하는 과정을 보여주고, 가상 NUMA 노드 토폴로지는 복수의 가상 NUMA 노드들을 포함한다. 예를 들면, 부모 파티션(204)(및/또는 하이퍼바이저(202)) 내의 과정(process)은 수신된 특성을 기초로 하여 가상 머신(240)를 위한 토폴로지를 판단할 수 있다. 예를 들면, 부모 파티션(204)은 가상 NUMA 노드(606)와 같은 가상 NUMA 노드에 대한 디폴트 크기(default 크기)를 식별하는 정보를 포함할 수 있다. 부모 파티션(204) 내의 과정은 디폴트 크

기와 요구되는 특성을 서술하는 정보를 사용하여 가상 머신(240)를 위한 가상 NUMA의 수를 판단할 수 있다. 특정 예시에서, 요구되는 특성은 10기가바이트의 RAM을 가지는 여섯 개의 프로세서 가상 머신일 수 있다. 만약 가상 NUMA 노드의 디폴트 크기가 두 개의 가상 프로세서들과 4기가바이트의 RAM이라면, 관리 시스템(502)은 가상 머신(240)가 세 개의 NUMA 노드들을 포함할 것을 나타내는 설정 파일을 생성할 수 있다.

[0033] 실시예에서 디폴트 가상 NUMA 노드 크기는 관리자 또는 관리 시스템(502)에 의해 설정될 수 있다. 도 5로 넘어가서, 관리 시스템(502)은 데이터센터(500) 내의 컴퓨터 시스템들(504-510)의 물리적 토폴로지를 식별할 수 있는 정보를 얻을 수 있는 하나 또는 그 이상의 프로그램들을 실행할 수 있고, 그 정보에는 각각의 컴퓨터 시스템(504-510)이 가지고 있는 RAM의 크기, RAM의 속도, RAM의 배열, 프로세서 속도, 각각의 프로세서가 몇 개의 코어가 있는지 여부 등이 포함된다.

[0034] 일반적으로, 가상 NUMA 노드의 크기는 데이터센터(500) 안의 가상 머신의 운용에 영향을 미친다. 예를 들면, 가상 NUMA 노드의 크기가 증가할수록, 예를 들면 메모리 및/또는 프로세서들 안에서, 가상 NUMA 노드의 이동성은 감소한다. 달리 말하면, 커다란 가상 NUMA 노드들은 가상 머신로 이동하기가 어렵다. 이것은 가상 NUMA 노드는 NUMA 노드 또는 가상 NUMA 노드를 실시하기 위해 충분한 'flat' 리소스들을 가지는 컴퓨터 시스템에 배정되어야 한다. 만약에, 예를 들어, 가상 NUMA 노드가 너무 큰 경우, 예를 들면 RAM이 너무 크거나 가상 프로세서들이 너무 많은 경우, 데이터센터(500) 안의 작은 NUMA 노드들에 맞지 않게 되고, 따라서 가상 머신을 이동시키는 능력이 제한된다. 더 나아가, 만약에 커다란 가상 NUMA 노드가 단순히 복수의 작은 NUMA 노드들에 배정될 경우에, 가상 머신의 퍼포먼스는 로컬 메모리와 원격 메모리 접속 시간 사이에 존재하는 차이로 인해서 감소할 것이다.

[0035] 다른 한편으로, 가상 NUMA 노드의 크기가 감소할수록, 게스트 운영 체제의 퍼포먼스는 불리하게 영향을 받을 수 있다. 이런 비효율성은 게스트 운영 체제가 애플리케이션들의 분리를 시도하고 단일 가상 NUMA 노드들에서 실행되기 때문에 발생할 수 있다. 게스트 운영 체제는 이 경우에 제한될 것이고, 퍼포먼스는 감소할 것이다.

[0036] 그에 맞춰서, 실시예에서 관리 시스템(502)은 데이터센터(500)를 위한 최적의 가상 NUMA 노드 크기를 판단함으로써 이동성과 효율성 사이의 균형을 찾을 수 있다. 예를 들면, 실시예에서 관리 시스템(502)의 논리 프로세서는 프로그램을 실행할 수 있고 데이터센터 내의 NUMA 노드의 평균 크기를 판단할 수 있고, 예를 들면 논리 프로세서들의 평균 수, RAM의 평균 용량 등, 가상 NUMA 노드의 크기가 시스템 내의 NUMA의 평균과 동일하거나 더욱 작도록 설정할 수 있다. 다른 실시예에서 프로그램은 가상 NUMA노드의 크기가 데이터센터(500) 내의 가장 작은 NUMA 노드의 크기보다 약간 작도록 세팅되도록 설정될 수 있다. 실시예 안에서, 가상 NUMA 노드의 크기는 평균 크기 또는 최소 크기보다 약간 작도록 설정될 수 있고, 이에 따라 만약 컴퓨터 시스템이 과도하게 커밋된 경우에, 복수의 가상 NUMA 노드가 단일 NUMA 노드에 배정될 수 있다. 특정 예시에서, 최소 NUMA 노드가 네 개의 논리 프로세서들과 8기가바이트의 RAM을 가질 경우에, 예를 들면 가상 NUMA 노드의 크기는 두 개의 가상 프로세서들과 4기가바이트의 RAM을 가지도록 설정될 수 있다.

[0037] 작업 906은 컴퓨터 시스템상에서 가상 머신의 인스턴트화(instantiating)를 보여주며, 가상 머신은 복수의 가상 NUMA 노드들을 포함한다. 실시예 안에서, 하이퍼바이저(202)는 논리 프로세서에 의해서 실행될 수 있으며 복수의 가상 NUMA 노드들을 가지는 가상 머신이 인스턴트화 될 수 있다. 예를 들면, 도 6 및/또는 도 7을 참조하면, 가상 NUMA노드들(606-608)을 가지는 가상 머신(240)가 컴퓨터 시스템(600 또는 700)에 의해 실시될 수 있다. 즉, VID(810)는 가상 머신(240)의 게스트 물리적 어드레스들을 RAM으로부터의 시스템 물리적 어드레스들로 뒷받침하고 가상 프로세서들을 하나 또는 그 이상의 논리 프로세서들로 뒷받침할 수 있다. 예를 들면, 게스트 물리적 어드레스들(614)의 블록들은 시스템 물리적 어드레스들(622)의 블록들로 뒷받침될 수 있고, 게스트 물리적 어드레스들(616)의 블록들은 시스템 물리적 어드레스들(624)의 블록들로 뒷받침될 수 있다. 그리고 나서 하이퍼바이저 스레드들은 가상 프로세서들을 뒷받침하는 논리 프로세서들 상에 스케줄 될 수 있고 가상 프로세서들을 나타내는 명령들이 실행될 수 있다. 도 6과 도 7에서 볼 수 있듯이, 각각의 가상 머신들의 토폴로지는 근원적인 하드웨어의 토폴로지로부터 독립적으로 생성될 수 있다. 즉, 각각의 가상 머신의 토폴로지는 그것을 실행하는 컴퓨터 시스템의 근원적 하드웨어 토폴로지로부터 분리되어 있다.

[0038] 실시예에서 가상 머신 BIOS 또는 부트 펌웨어는 가상 머신의 토폴로지를 묘사할 수 있고, 예를 들면 가상 NUMA 노드들을 가지는지 여부, 가상 NUMA 노드들의 크기, 그리고 가상 NUMA노드들에 대한 NUMA 비율들을 단일 애플리케이션의 게스트 운영 체제에 묘사할 수 있다. 데이터 구조(structure)는 게스트 OS(220) 또는 애플리케이션에 의해 프로세스 될 수 있고, 이것은 가상 NUMA노드들의 존재의 이점을 누리기 위해서 OS 또는 애플리케이션에 의해 사용될 수 있다. 예를 들면, 게스트 운영 체제(220)는 애플리케이션의 실행이 로컬에 남아있도록 NUMA가 인지하지 못하는 애플리케이션의 스레드들을 가상 NUMA 노드에 화합되도록 시도할 수 있다. 다른 예시에서, SQL

서버와 같은 데이터베이스 관리 프로그램은 가상 NUMA 노드들에 로컬에 고정하는 장치들(locks local)을 할당할 수 있고 데이터베이스는 가상 NUMA 노드들 상에서 일어나는 읽기/쓰기 요청들을 분할할 수 있다. 또 다른 실시예에서, 게스트 운영 체제(220)는 가상 머신 내의 각각의 가상 NUMA노드에 대하여 페이지 풀들(page pools)을 생성할 수 있다.

[0039] 계속해서 도 9에 대하여 서술하면, 작업 908은 조정 단계(adjusting)를 보여주고, 이는 복수의 가상 NUMA 노드들 중 특정한 가상 NUMA노드 안의 메모리 압력(memory pressure), 특정 가상 NUMA노드에 배정된 게스트 메모리의 용량에 기초한다. 예를 들면, 논리 프로세서, 예를 들면 도 6 또는 도 7의 논리 프로세서(212A),는 DMVSP(802)를 나타내는 명령들을 실행할 수 있고 가상 NUMA노드(606)와 같은 가상 NUMA 노드에 사용 가능한 게스트 물리적 어드레스들의 용량을 조절할 수 있다. 즉, DMVSP(802)는 실행될 수 있고 메모리는 가상 NUMA노드들이 경험하는 압력을 기초로 커밋 또는 디커밋될 수 있다.

[0040] 실시예에서, 메모리 압력은 게스트 운영 체제(220)의 퍼포먼스가 각각의 가상 NUMA 노드(606-608)에 사용 가능한 메모리 용량에 의해 어떤 영향을 받는지를 식별할 수 있다. 이 정보는 게스트 운영 체제(220)의 실시 시간(runtime) 동안에 계산될 수 있고, 예를 들면 DMVSC(804 및/또는 806)와 같은 하나 또는 그 이상의 DMVSCs에 의해 계산될 수 있고 DMVSP(802)로 보내질 수 있다. 예를 들면, 메모리 압력은 가상 NUMA 노드 내의 다른 레벨의 메모리 압력을 식별할 수 있는 수치들의 시리즈로 보여질 수 있다. 가상 NUMA 노드 내의 리소스들이 더 압박받을수록, 즉 가상 NUMA 노드 상에서 현재 작업량을 효율적으로 실행하기 위해서 요구되는 메모리 용량이 증가할수록, DMVSC(804)는 수치를 수정할 수 있고 DMVSP(802)와 이 정보를 주고 받을 수 있다.

[0041] 실시예에서, 메모리 압력 정보는 게스트 운영 체제(220)로부터 수신된 정보로부터 DMVSC(804)에 의해 계산될 수 있다. 예를 들면, DMVSC(804)는 가상 NUMA 노드(606)에 대한 메모리 관리자(808)로부터 운영 체제 페이징 정보(operating system paging information)를 수신하도록 설정될 수 있다. 게스트 운영 체제 페이징 레이트(paging rate)는 메모리 관리자(808)와 캐시 관리자에 의해 노출된 두 개의 카운터를 통해서 모니터 될 수 있고, 즉 페이징 레이트와 캐시 로테이션 레이트(rotation rate)가 이들이다.

[0042] 동일한 또는 다른 실시예에서, DMVSC(804)는 메모리 관리자(808)로부터 가상 NUMA 노드(606)와 결부된 물리적 메모리 통지들을 수신할 수 있고 이 정보를 사용하여 가상 NUMA 노드(606)의 메모리 압력을 컴퓨팅할 수 있다. 예를 들면, 메모리 관리자(808)는 가상 NUMA 노드(606)와 결부된 게스트 운영 체제(220) 내의 활동에 기초해서 고-메모리(high-memory) 통지들과 저-메모리(low-memory) 통지들을 출력할 수 있다. 메모리 관리자(808)는 이들 통지들을 LMT(low-memory threshold)와 HMT(high-memory threshold)에 기초하여 개시할 수 있다. 특정 예시적 실시예에서, 저-메모리 리소스 통지 이벤트를 나타내는 사용 가능한 메모리의 디폴트 레벨(default level)은 대략 4 GB 당 32 MB일 수 있고, 이때 최대값은 64MB이다. 고-메모리 리소스 통지 이벤트를 나타내는 사용 가능한 메모리의 디폴트 레벨은 예를 들면 저-메모리 디폴트 레벨의 세 배일수 있다. 그 둘 사이의 중간 메모리 사용 가능 레벨은 LMT와 HMT 레벨 사이의 간격을 나누는 방식으로 결정될 수 있다. 본 기술의 당업자들은 이들 수치들은 예시적 수치에 불과하고 개시 내용의 본질에서 벗어나지 않는 변경들이 이루어질 수 있음을 인지할 것이다.

[0043] 이들 통지들은, 다른 것들과 함께, DMVSC(804)에 의해 가상 NUMA 노드(606)의 메모리 압력을 계산하는데 사용될 수 있다. 예를 들면, 각각의 레벨은 수치, 예를 들면 0-4, 와 결부될 수 있고 만약에 다른 퍼포먼스 카운터들도 고려되었다면 그들 역시 수치들과 결부될 수 있다. 그 뒤 각각의 퍼포먼스 카운터의 수치들은 가상 NUMA 노드(606)의 현재 메모리 압력을 계산하는데 사용될 수 있다. 특정 예시에서, 메모리 압력은 퍼포먼스 카운터 수치들의 높은 값 또는 낮은 값을 선택함으로써 계산될 수 있다. 다른 예시에서, 퍼포먼스 카운터들의 평균 수치들이 메모리 압력으로 사용될 수 있다. 또 다른 실시예에서, 과거의 퍼포먼스 카운터 수치들을 고려하고 계산 과정에서 그들의 상대적 가중치에 영향을 미치기 위해서 각각의 퍼포먼스 카운터에 스칼라(scalar)를 배정하는 더 복잡한 알고리즘을 사용해서 메모리 압력을 계산할 수도 있다.

[0044] 메모리 커밋의 판단이 이루어진 후에, DMVSP(802)는 다양한 기술들을 사용할 수 있고, 그들 중에 하나는 핫 애드(hot-add) 작업이다. 예를 들면, 특정 운영 체제들은 시스템 재부팅 필요없이 실행 중인 운영 체제에 물리적 메모리의 범위들이 추가되도록 하는 핫 애드를 지원할 수 있다. 즉, 메모리 관리자(808)는 실행 중인 시스템에 동적 메모리의 추가를 지원하도록 설정될 수 있다. 핫 애드 실시에에서, DMVSC(804)는 메모리 관리자(808)의 핫 애드 인터페이스를 접속하도록 설정될 수 있고 DMVSC(804)는 핫 애드된 GPAs와 어떤 가상 NUMA 노드와 결부되어 있는지를 나타내는 메시지를 게스트 운영 체제(220)에 전송할 수 있다. 그 뒤에 메모리 관리자(808)는 게스트 운영 체제(220), 드라이버들, 애플리케이션들, 또는 가상 NUMA 노드(606) 상에서 실행중인 어떤 다른 프로세스

들에게 사용 가능한 새로운 메모리를 만들 수 있다. 예를 들면, DMVSC(804)는 VID(810)이 GPAs와 SPAs 사이의 관계들을 형성하고 난 후에 DMVSP(802)로부터 핫 애드된 메모리 어드레스들을 수신할 수 있다.

[0045] 유사하게, 핫 리무브(hot-remove) 작업이 가상 NUMA 노드(606)와 같은 가상 NUMA 노드로부터 메모리 주소들의 디커밋(de-commit)을 하는데 사용될 수 있다. 예를 들면, DMVSC(804)는 메모리가 핫 리무브 되었음을 나타내는 메시지를 게스트 운영 체제(220)에 전송할 수 있다. DMVSC(804)는 제거를 위하여 메모리 관리자(808)에게 가상 NUMA 노드(606)로부터 GPAs의 블록을 제공 요청할 수 있다. 그 다음에 이 실시예 안에서 DMVSC(804)는 메모리 관리자(808)의 제거 API를 호출할 수 있고 게스트 운영 체제(220)로부터 GPAs를 제거할 수 있다. 핫 리무브가 사용되는 실시예 안에서, 제거된 메모리는 게스트 현재 커밋에 반해서 카운트 되지 않고, 메모리 관리자(808)는 운영 체제에 의해서 마더보드로부터 물리적으로 제거된 메모리를 제거하기 위해서 사용되는 기술과 유사한 기술을 사용하여 그것의 내부 카운터들을 조절할 수 있다.

[0046] 다른 실시예에서 메모리는 벌루닝(ballooning) 기술을 사용하여 가상 NUMA 노드로 디커밋될 수 있다. 즉, 메모리는 가상 NUMA 노드 내의 게스트 물리적 어드레스들을 그들을 뒷받침하는 물리적 어드레스들로부터 분리시키는 방식에 의해 디커밋 될 수 있다. 예를 들면, 논리 프로세서(212B)는 DMVSC를 나타내는 명령들을 실행할 수 있고 하나 또는 그 이상의 메모리 블록과 같이 DMVSC(804)에 의해 사용하기 위한 특정 메모리 용량을 유보하도록 메모리 관리자(808)에 요청하는 메시지를 메모리 관리자(808)에게 보낼 수 있다. 메모리 관리자(808)는 DMVSC(804)에서의 독점적 사용을 위해서 메모리를 잠금 할 수 있고 DMVSC(804)는 메모리의 GPAs를 DMVSP(802)에 보낼 수 있다. 이 실시예에서, DMVSP(802)는 GPAs를 VID(810)에 보낼 수 있고 VID(810)은 그림자 페이지 표 안에서 이들 GPAs를 위한 입력들을 제거할 수 있다. 이 예시에서, 메모리 관리자(808)는 GPAs가 여전히 유효함을 나타내는 정보를 포함할 수 있으나, 실제로 GPAs는 시스템 물리적 어드레스들에 의해 더는 뒷받침되지 않는다. 이 예시에서, 메모리 관리자(808)는 잠긴 GPAs를 사용하지 않을 것이며 그들을 뒷받침하는 SPAs는 재할당될 수 있다.

[0047] 분리된 게스트 물리적 어드레스들은 물리적 어드레스들과 다시 결부될 수 있다. 이 예시에서 메모리 커밋 요청은 VID(810)에 의해 수신될 수 있고 VID(810)은 요청을 만족시키기 위해 SPAs를 얻을 수 있고 어드레스들의 범위를 DMVSP(802)에 보낼 수 있다. 실시예에서, VID(810)은 시스템 효율성을 증가시키기 위해서 연속적 범위의 SPAs를 얻도록 설정될 수 있다. 이 예시에서, VID(910)는 가상 NUMA 노드(606)와 결부된 DMVSC(804)가 독점적으로 사용하기 위해 잠겨진(locked) GPAs를 게스트 운영 체제(22)가 가지고 있는지를 판단할 수 있다. VID(810)은 잠긴 GPAs와 SPAs 사이의 관계를 생성할 수 있고, DMVSP(802)에 메시지를 전송할 수 있다. 그 뒤에 DMVSP(802)는 DMVSC(804)에 메시지를 보낼 수 있고 DMVSC(804)는 GPAs가 잠근 해제될 수 있고 가상 NUMA 노드(606)와 결부된 메모리 관리자(808)의 메모리 풀로 돌아올 수 있다는 것을 나타내는 메시지를 메모리 관리자(808)에게 보낼 수 있다.

[0048] 실시예에서, VID(810)는 GPAs가 벌루닝되는지 여부에 따라 핫 애드 기술 또는 벌루닝 기술을 사용할지 여부를 판단할 수 있다. 예를 들면, VID(810)이 SPAs를 수신하여 NUMA 노드(606)에 커밋하는 경우에, 이것은 임의의 GPAs가 DMVSC(804)에 의해 잠기는지 여부를 판단할 수 있다. 잠긴 GPAs가 있는 예시에서, VID(810)은 메모리가 핫 애드되기 전에 SPAs로 그들을 뒷받침할 수 있다. 메모리가 가상 NUMA 노드(606)에 커밋되기 전에, 이들은 제로잉(zeroing)될 수 있고 이들과 결부된 캐시 라인들은 보안상의 이유로 플러쉬(flush) 될 수 있다. 메모리를 제로잉함으로서, 하나의 파티션과 이전에 결부되어 있었던 메모리의 내용이 다른 파티션으로 누설되지 않는다.

[0049] 도 10으로 넘어가서, 도 10은 추가적인 작업들(1010-1020)을 포함하는 도 9의 작업 과정의 대체적 실시예를 도시한다. 작업 1010은 복수의 가상 NUMA 노드들 중에 제 2 가상 NUMA 노드의 메모리 압력이 사전 결정된 수치보다 큰지 여부를 판단하는 단계; 제 2 가상 NUMA 노드를 컴퓨터 시스템의 제 2 NUMA 노드로 이동시키는 단계를 보여준다. 예를 들면, 그리고 도 7로 돌아가서 보면, 실시예에서 제 2 가상 NUMA 노드(608)의 메모리 압력은 증가할 수 있다. 즉, 메모리 압력을 나타내는 수치가 DMVSP(802)에 의해 수신될 수 있고 이것은 가상 NUMA 노드(608)가 압박받고 있음을 나타낸다. 이 예시에서, 가상 머신(240) 또는 개개의 가상 NUMA 노드(608)는 목표 압력 수치를 가질 수 있고, 현재 압력 수치는 관리자에 의해 설정되는 목표 수치보다 작을 수 있다. 목표 압력 수치는 데이터 아키텍처 내에 저장될 수 있고, DMVSP(802)에 의해 접속될 수 있다. 그 뒤에 실행 중인 가상 머신 또는 가상 NUMA 노드들의 현재 압력 수치가 수신될 수 있다. DMVSP(802)는 실행 중인 가상 머신들 또는 가상 NUMA 노드들의 목록들 사이를 연속해서 관통해서 이동할 수 있고 메모리 압력 수치가 목표 수치로 감소되도록 메모리를 커밋할 수 있고, 목표 수치까지 압력을 증가시키기 위해 메모리 디커밋을 할 수 있다.

[0050] 예시에서, DMVSP(802)는 현재 가상 노드들(606 및 608), 예를 들면 NUMA 노드(702), 을 호스팅하는 NUMA 노드

가 양쪽의 가상 NUMA 노드들에 대한 목표 메모리 압력 수치를 얻는데 충분한 메모리를 할당할 수 없음을 판단하도록 설정될 수 있다. 예시에서, DMVSP(802)는 하이퍼바이저(202)에게 신호를 보내도록 설정될 수 있고 하이퍼바이저(202)는 NUMA 노드(702)에서 가상 NUMA 노드들 중 하나의 이동을 시도하도록 설정될 수 있다. 하이퍼바이저(202)는 NUMA 노드들(702-706)의 현재 작업량을 점검할 수 있고 예를 들면 NUMA 노드(704)가 가상 NUMA 노드를 호스트 할 수 있고 메모리 압력을 목표 수치까지 감소시키도록 하기 위해 필요한 리소스들을 할당할 수 있음을 판단할 수 있다. 예시에서, 하이퍼바이저(202)는 가상 NUMA 노드(608)를 NUMA 노드(704)에 재배정하도록 설정될 수 있다. 즉, 하이퍼바이저(202)는 VID(810)와 함께 게스트 물리적 어드레스들(616)을 시스템 물리적 어드레스들(714)로 다시 맵핑할 수 있고, 논리 프로세서들(212E-F)을 가상 프로세서들(230C, 230D)을 위한 이상적 프로세서들로 설정할 수 있다.

[0051] 계속해서 도 10을 서술하면, 작업 1012는 특정 가상 NUMA 노드로부터 게스트 메모리의 적어도 하나의 메모리 블록을 디커밋하는 단계와; 그리고 디커밋된 게스트 메모리의 적어도 하나의 메모리 블록을 제 2 가상 NUMA 노드에 커밋하는 단계를 도시한다. 예를 들면, DMVSP(802)는, 예를 들면 가상 NUMA 노드(606)로부터 메모리를 디커밋하고 메모리를 가상 NUMA 노드(608)에 커밋하도록 설정될 수 있다. 이 예시에서, 가상 NUMA 노드들 (606 및 608)은 단일 NUMA 노드, 또는 'flat" 아키텍처에 의해 뒷받침될 수 있다. 이 예시적 실시예에서, DMVSP(802)는 예를 들면 가상 NUMA 노드(608)에 커밋될 수 있는 메모리가 없는 경우에, 가상 NUMA 노드(606)로부터 메모리를 디커밋할 수 있다. 다른 예시에서, DMVSP(802)는, 예를 들면 가상 NUMA 노드(610)로부터 메모리 디커밋하여, 가상 NUMA 노드 (608)에 커밋할 수 있다. 즉 메모리는 한 가상 머신로부터 할당 해제돼서 다른 장치에 주어진다.

[0052] 특정 예시에서, 도 6을 참조하면, 가상 NUMA 노드들 (606 및 608)은 컴퓨터 시스템(600)의 자원들에 맵핑될 수 있다. 이 예시에서, DMVSP(802)는 예를 들어 메모리 우선도의 순서가 예를 들면 가상 머신(240)의 낮은 우선도(low priority) 가상 NUMA 노드들 또는 최저 우선도의 가상 머신로부터 시작하는지를 다른 가상 NUMA 노드들에서 점검할 수 있다. 만약에, 예를 들면, 가상 NUMA 노드(606)와 같은 가상 NUMA 노드가 감지되고 그것이 목표 임계치보다 낮은 메모리 압력 수치를 가진다면, DMVSP(802)는 메모리 디커밋을 개시할 수 있고, 가상 NUMA 노드(606)로부터 메모리를 제거한다. 디커밋 완료 후에, 커밋 작업이 개시될 수 있고 메모리는 가상 NUMA 노드(608)에 핫 애드될 수 있고, 또는 벌룬 게스트 물리적 어드레스들(ballooned guest physical addresses)이 시스템 물리적 어드레스들과 다시 결부될 수 있다.

[0053] 특정 실시예에서, 도 7을 참조하면, DMVSP(802)는 동일한 NUMA 노드(702)에 의해 뒷받침되는 다른 가상 NUMA 노드들의, 예를 들면, 메모리 우선도의 순서를 점검할 수 있다. 만약에, 예를 들어, 가상 NUMA 노드(608)와 동일한 NUMA 노드 상에서 가상 NUMA 노드가 감지되고 그것이 목표 임계치 미만의 메모리 압력 수치를 가진다면, DMVSP(802)는 메모리 디커밋을 개시할 수 있다. 디커밋이 완료된 이후, 커밋 작업이 개시될 수 있고 메모리는 가상 NUMA 노드(608)에 핫 애드되거나 또는 벌룬 게스트 물리적 어드레스들이 시스템 물리적 어드레스들과 다시 결부될 수 있다.

[0054] 계속해서 도 10에 대해서 서술하면, 작업 1014는 특정 가상 NUMA 노드에서 적어도 하나의 게스트 메모리 블록이 시스템 메모리에서 분리된 것을 판단하는 단계; 게스트 메모리의 적어도 하나의 메모리 블록을 시스템 메모리의 적어도 하나의 메모리 블록에 맵핑하는 단계를 도시한다. 예를 들면, 실시예에서, DMVSP(802)는 논리 프로세서에 의해 실행될 수 있고 가상 NUMA 노드(606)안의 GPAs를 SPAs(624)로 뒷받침하기 위한 판단이 이루어질 수 있다. 예를 들면, GPAs는 DMVSC(804)에 의해 유보될 수 있고 SPAs는 다른 가상 NUMA 노드 또는 부모 파티션(204)에 재할당될 수 있다. 이 실시예에서, 메모리 페이지 커밋 요청은 VID(810)이 수신할 수 있고 VID(81)은 요청을 만족시키기 위해 SPAs를 얻을 수 있고 어드레스들의 범위를 DMVSP(802)에 보낼 수 있다. 이 실시예에서, VID(810)은 시스템 효율성을 증가시키기 위해서 연속적 범위의 SPAs를 얻도록 설정될 수 있다. NUMA 실시예 안에서, VID(810)은 가상 NUMA 노드(606)를 실행하는 NUMA 노드와 동일한 NUMA 노드로부터 연속적 범위의 SPAs를 얻도록 설정될 수 있다. VID(810)은 잠긴 GPAs와 SPAs 사이의 관계들을 생성할 수 있고 메시지를 DMVSP(802)로 보낼 수 있다. DMVSP(802)는 그리고 나서 메시지를 DMVSC(804)로 보낼 수 있고 DMVSC(804)는 GPAs가 잠금 해제될 수 있고 가상 NUMA 노드(606)와 결부된 메모리 풀로 되돌아갈 수 있다는 것을 나타내는 메시지를 메모리 관리자(808)에 보낼 수 있다.

[0055] 계속해서 도 10에 관해서 서술하면, 작업 1016은 특정 가상 NUMA 노드를 컴퓨터 시스템의 제 1 NUMA 노드로 맵핑하는 단계; 그리고 특정 가상 NUMA 노드를 컴퓨터 시스템의 제 2 NUMA 노드로 이동시키는 단계를 도시한다. 예를 들면, 도 7을 참조하면, 게스트 운영 체제(220)는 NUMA 노드들(702, 704)과 같은 적어도 두 개의 NUMA 노드들을 가로질러서 분산될 수 있다. 예를 들면, 도 7을 참조하면, 하이퍼바이저(202)는 가상 NUMA 노드들(606, 608)이 NUMA 노드(702)에서 실행되도록 일정을 잡을 수 있다. 이 예시에서, 하이퍼바이저(202)는 NUMA 노드

(702)가 압박을 받고 있음을 나타내는 신호를 수신할 수 있다. 예를 들면, 게스트 운영 체제(220)는 가상 NUMA 노드들(606, 608)에서 메모리가 부족함을 나타내는 신호를 생성할 수 있다. 이 예시에서, 하이퍼바이저(202)는 가상 NUMA 노드(608)를 NUMA 노드(702)로부터 이동시킴으로써 압박받는 NUMA 노드의 작업량을 감소시키도록 설정될 수 있다.

[0056] 계속해서 도 10과 관련해서 서술하면, 작업 1018은 가상 프로세서를 특정 가상 NUMA 노드에 추가시키는 단계를 도시한다. 예를 들면, 실시예에서, 가상 프로세서(230B)와 같은 가상 프로세서는 예를 들면 핫 애드 작업을 사용하여 가상 머신(240)의 실행 실시 시간 동안에 추가될 수 있다. 즉, 가상 NUMA 노드(606)는 한 시점에서 오직 하나의 단일 가상 프로세서(230A)만을 가질 수 있고, 그리고 나서 또 하나가 추가될 수 있다. 실시예에서, 새롭게 추가된 프로세서는 가상 프로세서(230A)를 뒷받침하는 프로세서에 배정될 수 있고, 또는 다른 논리 프로세서가 가상 프로세서(230B) 스레드들을 실행하기 위해 할당될 수 있다. NUMA 실시예 안에서, 만약에 다른 논리 프로세서가 가상 프로세서(230B)를 뒷받침하기 위해서 사용된다면, 이것은 가상 NUMA 노드(606) 안에서 다른 가상 프로세서들을 뒷받침하는 것과 동일한 NUMA 노드(702)로부터 할당될 수 있다.

[0057] 계속해서 도 10과 관련해서 서술하면, 작업 1020은 가상 프로세서는 논리 프로세서에 배정되고, 논리 프로세서는 NUMA 노드에 배정되고 가상 프로세서는 가상 NUMA 노드에 배정되는 가상 머신의 가상 프로세서의 실행 요청을 수신하는 단계; 논리 프로세서가 가상 프로세서를 실행할 수 없음을 판단하는 단계; 그리고 가상 프로세서를 실행하는 제 2 NUMA 노드로부터의 제 2 논리 프로세서를 선택하는 단계를 도시한다. 예를 들면, 그리고 도 7을 참조하면, 실시예에서 하이퍼바이저(202)는 가상 프로세서(230A)로부터의 가상 프로세서 스레드의 실행 요청을 수신할 수 있고 이상적 프로세서(212A) 상에서 스레드의 일정을 잡기 위한 시도를 할 수 있다. 이 예시에서, 하이퍼바이저(202)는 논리 프로세서(212A)가 초과 커밋되었고 가상 프로세서 스레드를 실행할 수 없음을 감지할 수 있다. 이 경우에, 하이퍼바이저(202)는 실행될 수 있고 이것은 다른 논리 프로세서를 선택하여 가상 프로세서 스레드를 실행하도록 할 수 있다. 예를 들면, 하이퍼바이저(202)는 동일한 NUMA 노드 상의 다른 논리 프로세서의 선택을 시도할 수 있다. 만약에, 예를 들어, NUMA 노드가 초과 커밋된 상태라면, 하이퍼바이저(202)는 논리 프로세서(212E)와 같은 원격 프로세서를 선택해서 가상 프로세서(230A)를 실행하도록 설정할 수 있다. 이 예시에서, 원격 노드 상에 일정을 잡을지 기다릴지 여부의 결정은 NUMA 노드(704)와 결부된 NUMA 비율(ratio)을 사용해서 이루어질 수 있다. 만약에 NUMA 비가 낮고 이상적 프로세서의 예상 대기 시간이 긴 경우에, NUMA 노드(704) 상에 일정을 잡는 판단이 이루어질 수 있다. 반면에, 만약에 NUMA 비가 높고 예상 대기 시간이 적은 경우에, 대기 판단이 이루어질 수 있다.

[0058] 도 11로 넘어가서, 도 11은 작업 1100, 1102, 1104, 그리고 1106을 포함하는 개시 내용의 측면들을 실행하는 작업 과정을 도시한다. 작업 1100은 작업 과정을 시작하고 작업 1102는 가상 머신을 실행하는 단계를 보여주며, 가상 머신은 복수의 가상 NUMA 노드들을 포함하는 토폴로지를 가지고, 가상 머신의 토폴로지는 컴퓨터 시스템의 물리적 토폴로지와는 독립적으로 생성된다. 예를 들면, 하이퍼바이저(202)는 복수의 가상 NUMA 노드들을 가지는 가상 머신을 실행할 수 있다. 도 6에서 볼 수 있듯이, 가상 머신(240)는 가상 NUMA 노드들(606, 608)을 포함하도록 생성될 수 있다. 가상 NUMA 노드들은 각각 하나 또는 그 이상의 가상 프로세서들(230A-D)과 게스트 물리적 어드레스들(614, 616)을 가진다. 이 실시예에서, 가상 NUMA 노드들(606, 608)은 근원적 하드웨어의 토폴로지로부터 독립적으로 생성될 수 있다. 즉, 가상 머신의 토폴로지는 도 6 및 도 7에서 도시된 바와 같이 근원적 하드웨어와는 관련이 없다. 따라서, 이 실시예에서, 각각의 가상 머신의 토폴로지는 그것을 실행하는 컴퓨터 시스템의 근원적 물리적 토폴로지로부터 분리된다.

[0059] 계속해서 도 11과 관련해서 서술하면, 작업 1104는 각각의 복수인 가상 NUMA 노드 안의 메모리 압력을 판단하는 단계를 도시한다. 예를 들면, 그리고 도 8을 참조하면, 각각의 가상 NUMA노드(606, 608)에 대하여 메모리 압력이 얻어질 수 있고, 예를 들면 생성되거나 및/또는 수신될 수 있다. 메모리 압력 정보는 각각의 가상 NUMA 노드(606-608)에서의 사용 가능한 메모리 용량에 의해 게스트의 퍼포먼스가 어떤 영향을 받는지를 식별할 수 있다. 이 정보는 게스트 운영 체제(220)의 실행 시간 동안에, 예를 들면, DMVSC(804)와 같은 하나 또는 그 이상의 DMVSCs에 의해 계산될 수 있고, DMVSP(802)로 보내질 수 있다. 즉, 특정 실시예에서, 논리 프로세서는 DMVSC(804 또는 806)를 나타내는 명령들을 실행할 수 있고 각각의 가상 NUMA 노드에 대한 메모리 압력 정보를 생성할 수 있다. 그 뒤에 이 정보는, 예를 들면, DMVSP(802)로 보내질 수 있다.

[0060] 예시적 실시예에서, 메모리 압력 정보는 0-4의 범위를 가지는 수치들의 시리즈를 포함할 수 있고, 각각의 수치는 가상 NUMA 노드들(606-608)로 인해서 게스트 OS가 경험하는 메모리 압력의 다른 레벨을 식별할 수 있다. 게스트 운영 체제가 더 압박받을수록, 즉 현재 작업량을 효율적으로 실행하는데 요구되는 메모리 용량이 증가할수

록, DMVSCs(804, 806)은 그들의 수치를 수정하고 이 정보를 DMVSP(802)에게 전달할 수 있다.

[0061] 계속해서 도 11에 관해서 서술하면, 작업 1106은 다수의 가상 NUMA 노드들 각각의 메모리 압력에 기초하여 다수의 가상 노드 중 적어도 하나에 배정된 게스트 메모리를 조절하는 단계를 도시한다. 작업 1206을 포함하는 실시예에서, 논리 프로세서(212A)는 DMVSP(802)를 나타내는 명령들을 실행할 수 있고 예를 들면 가상 NUMA 노드(606) 내의 게스트 물리적 어드레스들의 용량을 조절할 수 있다. 즉, DMVSP(802)는 가상 NUMA 노드(606) 내의 메모리 압력을 기초로 메모리 커밋 또는 디커밋을 할 수 있고, 예를 들면 만약에 가상 NUMA 노드(606)에 할당된 프로세스들이 압박받으면, 메모리는 커밋될 수 있다.

[0062] 실시예에서, DMVSP(802)에 의해 설정된 논리 프로세서(212)가 메모리 커밋 또는 디커밋을 판단하는 경우에, 이것은 각 메모리 블록 기준으로 이를 수행할 수 있다. 예를 들면, DMVSP(802)는 메모리 블록의 커밋/디커밋을 할 수 있고 메모리 상태가 어떻게 변하는지 점검할 수 있다. 만약에 메모리 상태가 변하지 않는다면, DMVSP(802)는 다른 메모리 블록에 커밋/디커밋 할 수 있다.

[0063] 도 12로 넘어와서, 도 12는 작업들(1208-1216)을 포함하는 도 11의 작업 과정(1100)의 대체적 실시예를 도시한다. 도시된 바와 같이, 작업 1208은 가상 머신을 제 2 컴퓨터 시스템에 보내는 단계를 보여준다. 예를 들면, 그리고 도 5를 참조하면, 실시예에서 가상 머신의 상태는 하나 또는 그 이상의 설정 파일들에 저장될 수 있고 다른 컴퓨터 시스템으로 보내질 수 있고, 예를 들면 컴퓨터(504)에서 컴퓨터(506)로 보내질 수 있다. 컴퓨터 시스템(506)의 하이퍼바이저는 파일 또는 파일들을 읽거나 가상 머신을 인스턴트화할 수 있다.

[0064] 가상 머신 토폴로지는 가장 장치들을 복원하고 이동하는 능력에 영향을 미친다. 구체적으로, 근원적 하드웨어의 토폴로지의 감지를 허용하는 판단과 가상 NUMA 노드들의 크기는 가상 머신이 얼마나 잘 실행되고 쉽게 이동될 수 있는지에 영향을 미친다. 예를 들면, 가상 NUMA 노드들의 크기는 이동시키는 능력에 영향을 미치고, 예를 들면 가상 NUMA 노드의 크기가 증가할수록 가상 NUMA노드의 이동성은 감소하고 가상 NUMA 노드의 크기가 감소할수록 가상 머신의 퍼포먼스 역시 감소한다. 추가로, 근원적 컴퓨터의 토폴로지를 감지할 수 있는 가상 머신들은 쉽게 이동될 수 없고, 이는 NUMA 인지 운영 체제와 애플리케이션이 부팅시에 그들이 감지하는 제 1 토폴로지에 근거해서 스스로를 최적화하나, 이러한 최적화는 가상 머신이 장래에 이동될 컴퓨터상에서는 효력이 없을 수 있기 때문이다. 따라서, 게스트 운영 체제의 부팅시에 가상 NUMA 노드들을 게스트 운영 체제에 드러냄으로써 운영 체제는 NUMA노드 사용을 위해 최적화될 수 있다. 가상 NUMA노드들의 크기를 알맞게 맞춤으로써, 가상 머신은 데이터센터(500) 안의 많은 다양한 컴퓨터 시스템들에 맞게 최적화될 수 있다.

[0065] 예를 들면, 그리고 도 6을 참조하면, 가상 머신(240)는 하나 또는 그 이상의 가상 NUMA 노드들(606, 608)을 포함할 수 있다. 하이퍼바이저(240)는 가상 프로세서들(230A-D)을 논리 프로세서들(212A-D)로 뒷받침할 수 있다. 게스트 운영 체제(220)가 부팅 될 때, 이것은 가상 NUMA 노드들(606, 608)을 감지할 수 있고 가상 NUMA노드들(606, 608)을 사용하기 위하여 프로세스들을 스케줄링과 실행을 최적화하도록 설정될 수 있다. 약간의 시간이 흐르고나서, 가상 머신(240)는 도 7에 도시된 것과 유사한 물리적 토폴로지를 가지는 컴퓨터 시스템으로 이동될 수 있다. 도 7의 하이퍼바이저(202)는 가상 프로세서들(230A, 230B)을 논리적 프로세서들(212A, 212B)로 뒷받침할 수 있고 가상 프로세서들(230C, 230D)을 논리적 프로세서들(212E, 212F)로 뒷받침할 수 있다. 게스트 운영 체제(220)는 비록 근원적 컴퓨터 토폴로지가 SMP에서 NUMA로 변화했을지라도 도 6의 컴퓨터 시스템에서 실행되었을 때와 동일한 방식으로 계속 운용될 수 있다.

[0066] 계속해서 도 12에 관해서 서술하면, 작업 1210은 복수의 가상 NUMA 노드 중 제 1 가상 노드를 컴퓨터 시스템의 제 1 NUMA 노드로 맵핑하는 단계; 그리고 복수의 가상 NUMA 노드 중 제 2 가상 노드를 컴퓨터 시스템의 제 2 NUMA 노드로 맵핑하는 단계를 보여준다. 예를 들면, 그리고 도 7을 참조하면, 논리 프로세서(212A)와 같은 논리 프로세서는 하이퍼바이저 명령들을 실행할 수 있고 가상 NUMA 노드(606, 608)와 같은 가상 NUMA 노드들을 NUMA 노드(702)와 같은 NUMA 노드에 밀접화(affinitize)할 수 있다. 더 구체적으로, 논리 프로세서는 게스트 물리적 어드레스들(614, 616)을 NUMA노드(702)의 RAM(214)으로부터의 시스템 물리적 어드레스들로 뒷받침할 수 있고 가상 프로세서들(230A-230D)을 논리 프로세서들(212A-212D)로 뒷받침할 수 있다.

[0067] 계속해서 도 12에 관해서 서술하면, 작업 1212는 복수의 가상 NUMA 노드들 중 제 1 가상 NUMA 노드를 컴퓨터 시스템의 제 1 NUMA 노드에 맵핑하는 단계; 그리고 복수의 가상 NUMA 노드들 중 제 2 가상 NUMA 노드를 컴퓨터 시스템의 제 2 NUMA 노드에 맵핑하는 단계를 보여준다. 예를 들면, 그리고 도 7을 참조하면, 논리프로세서(212A)와 같은 논리 프로세서는 하이퍼바이저 명령들을 실행할 수 있고 가상 NUMA 노드(606)를 NUMA노드(702)에 배정할 수 있고 가상 NUMA노드(608)를 NUMA 노드(704)에 배정할 수 있다. 이 예시에서, 하이퍼바이저(202)가 실행될 때, 하이퍼바이저 스케줄러는 가상 프로세서들(230A-B)로부터의 스레드를 논리 프로세서들(212A-212D) 상에 스

케줄 할 수 있고 가상 프로세서(230C-230D)로부터의 스레드를 논리 프로세서들(212E-212G) 상에 스케줄 할 수 있다.

[0068] 도 13으로 넘어가면, 이것은 작업 1314를 포함하는 도 12의 작업 과정의 대체적 실시예를 도시하고, 작업 1314는 제 2 가상 NUMA 노드의 메모리 압력이 사전 결정된 수치보다 크다는 것을 결정하는 단계; 그리고 제 2 가상 NUMA 노드를 컴퓨터 시스템의 제 2 NUMA 노드로 이동시키는 단계를 보여준다. 예를 들면, 그리고 도 7로 넘어가서, 실시예 내에서 제 2 가상 NUMA 노드(608) 내의 메모리 압력은 증가할 수 있다. 즉, 메모리 압력을 나타내는 수치가 DMVSP(802)에 의해 수신될 수 있고 이것은 가상 NUMA 노드가 압박받고 있음을 나타낸다. 이 예시에서, 각각의 가상 NUMA노드(606-612), 및/또는 가상 머신(240-242)는 목표 압력 수치를 가질 수 있고 가상 NUMA 노드(608)에 대한 현재 압력 수치는 관리자에 의해 설정된 목표 압력 수치보다 클 수 있다. 목표 압력 수치는 DMVSP(802)에 의해 접속 가능한 데이터 아키텍처 내에 저장될 수 있다. 그 다음에 실행 중인 가상 머신들 또는 가상 NUMA 노드들의 현재 압력 수치들이 수신될 수 있다. DMVSP(802)는 실행 중인 가상 머신들 또는 가상 NUMA 노드들의 목록을 연속적으로 지나갈 수 있고 메모리 압력 수치들을 목표 수치까지 낮추기 위해서 메모리를 커밋할 수 있고, 압력을 목표 수치들까지 높이기 위해서 메모리를 디커밋할 수 있다.

[0069] 도 14로 넘어가면, 이것은 작업 1416을 포함하는 도 12의 작업 과정의 대체적 실시예를 도시하고, 작업 1416은 제 2 가상 머신의 메모리 압력이 사전 결정된 수치보다 큰 것을 결정하는 단계; 그리고 가상 머신의 제 2 가상 NUMA 노드를 컴퓨터 시스템의 제 1 NUMA 노드로 이동시키는 단계를 보여준다. 실시예에서 적어도 두 개의 가상 머신들이 실행 중일 수 있고, 예를 들면 가상 머신들(240, 242)이 실행 중일 수 있다. 이 예시에서 가상 머신(240)의 가상 NUMA 노드들은, 예를 들면, 도 7의 NUMA 노드들(702, 704)에 맵핑될 수 있고 가상 머신(242)의 가상 NUMA 노드들은, 예를 들면 NUMA 노드(706)에 맵핑될 수 있다. 이 실시예에서 각각의 가상 머신(240, 242), 및/또는 각각의 가상 NUMA 노드(606-612)는 DMVSP(802)에 의해 접속될 수 있는 데이터 아키텍처 내에 저장될 수 있는 목표 압력 수치를 가질 수 있다. 이 예시에서, 제 2 가상 머신(242)의 메모리 압력은 가상 머신 내의 활동, 예를 들면 많은 읽기/쓰기 요청들을 받은 가상 머신(242), 으로 인해서 증가할 수 있고, 수치는 DMVSP(802)에 의해 수신될 수 있다. DMVSP(802)는 실행 중인 가상 머신 또는 가상 NUMA 노드들의 현재 압력 수치들을 수신할 수 있고, 메모리 압력을 감소시키기 위해서 메모리가 가상 머신(242)에 커밋될 수 있는지 여부를 판단하기 위해서 실행 중인 가상 머신들 또는 가상 NUMA노드들의 목록을 연속적으로 전체적으로 살펴볼 수 있다.

[0070] 압력이 메모리의 커밋 또는 디커밋에 의해서 완화될 수 없는 상황에서는, DMVSP(802)는 하이퍼바이저(202)에 신호를 보내도록 설정될 수 있고, 하이퍼바이저(202)는 메모리 압력을 완화하기 위해서 컴퓨터 시스템의 리소스들의 재할당을 시도할 수 있다. 예를 들면, 하이퍼바이저(202)는 NUMA 노드들(702-706)의 현재 작업량을 점검하고, 예를 들면 NUMA 노드(702)는 가상 머신(240)로부터 가상 NUMA 노드들을 호스트하고 가상 NUMA 노드(608)를 NUMA 노드(702)로 재배정할 수 있다는 판단을 할 수 있다. 즉, 하이퍼바이저(202)는, VID(810)와 함께, 게스트 물리적 어드레스들(616)을 시스템 물리적 어드레스들(712)로 다시 맵핑(remap)하고 논리 프로세서들(212A, 212D)을 가상 프로세서들(230C, 230D)을 위한 이상적 프로세서들로 설정할 수 있다. 그 뒤에 하이퍼바이저는 가상 NUMA 노드(610)를 NUMA 노드(704)로 다시 맵핑하고 메모리 압력을 감소시키기 위해서 가상 머신(242)의 각각의 가상 NUMA 노드(610-612) 안의 메모리를 조절할 수 있다.

[0071] 도 15로 넘어가면, 이것은 작업들 1500, 1502, 그리고 1504를 포함하는 개시된 내용의 측면을 실시하는 작업 과정을 도시한다. 작업 1500은 작업 과정을 시작하고 작업 1502는 제 1 가상 머신을 실행하는 단계를 보여주고, 가상 머신은 복수의 가상 NUMA 노드들을 포함하는 토폴로지를 가지고, 각각의 복수의 가상 NUMA 노드는 가상 프로세서와 게스트 물리적 어드레스들을 포함하고, 가상 머신의 토폴로지는 컴퓨터 시스템의 물리적 토폴로지로부터 독립적으로 생성된다. 예를 들면, 도 7의 하이퍼바이저(202)는 복수의 가상 NUMA 노드들(608-610)을 가지는 가상 머신(240)를 실행할 수 있다. 가상 NUMA 노드(606, 608)는 각각 하나 또는 그 이상의 가상 프로세서들(230A-230D)과 게스트 물리적 어드레스들(614-616)을 가질 수 있다. 이 실시예에서 가상 NUMA 노드들(606, 608)은 컴퓨터 시스템(700)의 리소스들에 맵핑될 수 있다. 예를 들면, 논리 프로세서들(212A, 212B)은 가상 프로세서(230A, 230B)의 이상적 프로세서들로 설정될 수 있고 게스트 물리적 어드레스들(614)은 시스템 물리적 어드레스들(710)로 뒷받침될 수 있다. 마찬가지로, 논리 프로세서들(212E, 212F)은 가상 프로세서(230C, 230D)의 이상적 프로세서로 설정될 수 있고 게스트 물리적 어드레스들(616)은 시스템 물리적 어드레스들(714)에 맵핑될 수 있다. 이 실시예에서 가상 NUMA 노드들(606, 608)은 근원적 하드웨어의 토폴로지로부터 독립적이다. 즉, 가상 머신의 토폴로지는 도 6 및 도 7에 도시된 것과 같이 근원적 하드웨어와는 무관하다. 따라서, 이 실시예에서, 가상 머신의 토폴로지는 그것을 실시하는 컴퓨터 시스템의 근원적 물리적 토폴로지로부터 분리되어

있다.

[0072] 계속해서 도 15에 대하여 서술하면, 작업 1504는 복수의 가상 NUMA 노드에 추가적 가상 프로세서를 추가하는 단계를 보여준다. 예를 들면, 실시예에서, 추가적 가상 프로세서는 예를 들면 가상 NUMA 노드(606)와 같은 가상 NUMA 노드에 추가될 수 있다. 이 예시에서, 가상 프로세서와 같은 가상 프로세서는 예를 들면 프로세서 핫 애드 작업을 사용하는 가상 머신(240)의 실시 시간 실행 도중에 추가될 수 있다. 실시예에서 새로 추가된 프로세서는 가상 프로세서(230A)를 뒷받침하는 프로세서 또는 이상적 프로세서로 설정될 수 있는 다른 논리적 프로세서에 배정되어 새로 추가된 가상 프로세서의 스레드들을 실행할 수 있다. NUMA 실시예에서, 논리 프로세서들은 가상 NUMA 노드(606)를 뒷받침하는 것과 동일한 NUMA 노드(702)로부터 할당될 수 있다.

[0073] 도 16으로 넘어가면, 이것은 추가적 작업들(1606-1612)을 포함하는 도 15에 도시된 작업 과정의 대체적 실시예를 도시한다. 작업 1606은 각각의 복수의 가상 NUMA 노드 내의 메모리 압력을 판단하는 단계; 그리고 각각의 복수의 가상 NUMA 노드 내의 메모리 압력에 기초해서 조절하는 단계, 복수의 가상 NUMA 노드 중 적어도 하나에 배정된 게스트 메모리를 보여준다. 도 8을 참조하면, 각각의 가상 NUMA 노드(606, 608)에 대한 메모리 압력은, 예를 들면 생성되거나 및/또는 수신되는 것을 통해 얻어질 수 있다. 메모리 압력 정보는 각각의 가상 NUMA 노드에 대하여 사용 가능한 메모리 용량에 의해 게스트의 퍼포먼스가 어떻게 영향을 받는지를 식별할 수 있다. DMVSCs(804, 806)은 수신하도록 설정될 수 있고, 예를 들면 물리적 메모리 통지들 및/또는 메모리 관리자(808)로 부터 게스트 운영 체제 페이징 정보를 수신할 수 있고, 이를 사용해서 각각의 가상 NUMA 노드(606, 608)의 메모리 압력을 산출할 수 있다.

[0074] 계속해서 이 예시에 대하여 서술하면, 예를 들어 논리 프로세서(212A)는 DMVSP(802)를 나타내는 명령들을 실행하고 예를 들면 가상 NUMA 노드(606) 내의 게스트 물리적 어드레스들의 용량을 조절할 수 있다. 즉, DMVSP(802)는 가상 NUMA 노드(606)의 리소스들로 인해서 게스트 OS가 경험하는 메모리 압력에 기초해서 메모리의 커밋 또는 디커밋을 한다.

[0075] 계속해서 도 16에 대하여 서술하면, 작업 1608은 복수의 가상 NUMA 노드로부터 가상 프로세서를 제거하는 단계를 도시한다. 1706을 포함하는 실시예에서, 하이퍼바이저(202)는 논리 프로세서에 의해서 실행될 수 있고 가상 프로세서는 가상 NUMA 노드(606)로부터 제거될 수 있다. 예를 들면, 하이퍼바이저(202)는 게스트 운영 체제(220)의 핫 리무브(hot-remove) API를 접속할 수 있고 예를 들면 가상 NUMA 노드(608)로부터 가상 프로세서(230B)를 제거할 수 있다.

[0076] 작업 1610으로 넘어가면, 이것은 복수의 가상 NUMA 노드들에 대한 NUMA 비율을 게스트 운용시스템에 보고하는 단계를 보여준다. 예를 들면, 실시예에서 하이퍼바이저(202)는 가상 NUMA 노드들(606-608)에 대한 NUMA 비율을 생성할 수 있고 이 정보는 도 6 또는 도 7의 게스트 운영 체제(22)에 보고될 수 있다. 실시예에서 하이퍼바이저(202)는 가상 NUMA 노드들에 대한 NUMA 비율을 나타내는 가상 머신의 펌웨어 표 내의 데이터 아키텍처를 생성할 수 있고 게스트 운영 체제(22)가 부팅 될 때, 게스트는 표를 읽고 스케줄링 결정에 있어서 그 정보를 사용할 수 있다. 예를 들면, 게스트 운영 체제, 또는 NUMA 인지 애플리케이션, NUMA 비율을 사용하여 원격 NUMA 노드로부터의 리소스들을 사용할지 여부를 판단할 수 있다. 예를 들면, 운영 체제는 실행 준비 단계인 계속 중인 스레드들을 가질 수 있다. 이 예시의 운영 체제는 또한 이상적 프로세서가 자유로워질 때까지 일정 시간을 기다리도록 설정될 수 있고 안 그랬다면 사전 결정된 NUMA 비율보다 낮은 수치를 가지는 원격 프로세서상에 스레드를 스케줄 할 것이다. 이 경우에 스케줄러가 허용하는 대기 시간은 NUMA 비에 따라 달라진다.

[0077] 작업 1612로 넘어가 보면, 이것은 가상 머신로부터 복수의 NUMA 노드들 중 제 1 NUMA 노드로 이동하는 단계를 보여준다. 예를 들면, 실시예에서 하이퍼바이저 명령들은 논리 프로세서에 의해 실행될 수 있고 가상 머신(240)는 복수의 NUMA 노드들(704, 706)에 맵핑될 수 있다. 이 실시예에서 컴퓨터 시스템(700)은 과도한 사용 상태일 수 있고, 예를 들면 가상 머신(242)가 대부분의 리소스들을 사용 중일 수 있고, 따라서 가상 머신(242)가 컴퓨터(700)로부터 이동될 수 있다. 이런 상황에서 하이퍼바이저(202)는 컴퓨터 시스템(700)의 리소스들을 재할당할 수 있고 가상 머신(240)를 NUMA 노드들(704, 706)로 맵핑할 수 있다.

[0078] 도 17로 넘어가 보면, 이것은 작업들 1714와 1716을 포함하는 도 16의 작업 과정의 대체적 실시예를 도시한다. 실시예에서 작업 1606은 작업 1714를 포함하고, 작업 1714는 가상 NUMA 노드의 현재 메모리 압력이 목표 임계치(threshold)보다 낮다는 판단에 기초하여 복수의 가상 NUMA 노드 중 제 1 가상 NUMA 노드에서 메모리를 디커밋하는 단계를 보여준다. 예를 들면, 실시예에서 DMVSP(802)는 가상 NUMA 노드(608)의 메모리 압력이 목표 임계치보다 낮은 경우에 가상 NUMA 노드(606)로부터 메모리를 디커밋하도록 설정될 수 있다. 예를 들면, 실시예 안에서 DMVSP(802)는 예를 들면 부모 파티션에 커밋 가능한 메모리가 없고 가상 노드(608)가 수용 불가능한 압력을

경험하고 있다면 가상 NUMA 노드(606)로부터 메모리를 풀어주는 시도를 할 수 있다. 만약에 메모리가 가상 NUMA 노드(606)로부터 디커밋된다면, 비동기식 메시지가 DMVSC(804)로 보내질 수 있고 이 메시지는 DMVSC(804)에게 메모리 디커밋할 것을 지시한다. 게스트 OS와 결부된 DMVSC(804)가 응답할 때, 이것은 가상 NUMA 노드(606) 내의 새로운 메모리 압력을 나타낼 수 있다. 일부 경우에서, 메모리 압력은 메모리 제거 과정에 대한 응답 과정에서 증가될 수 있다.

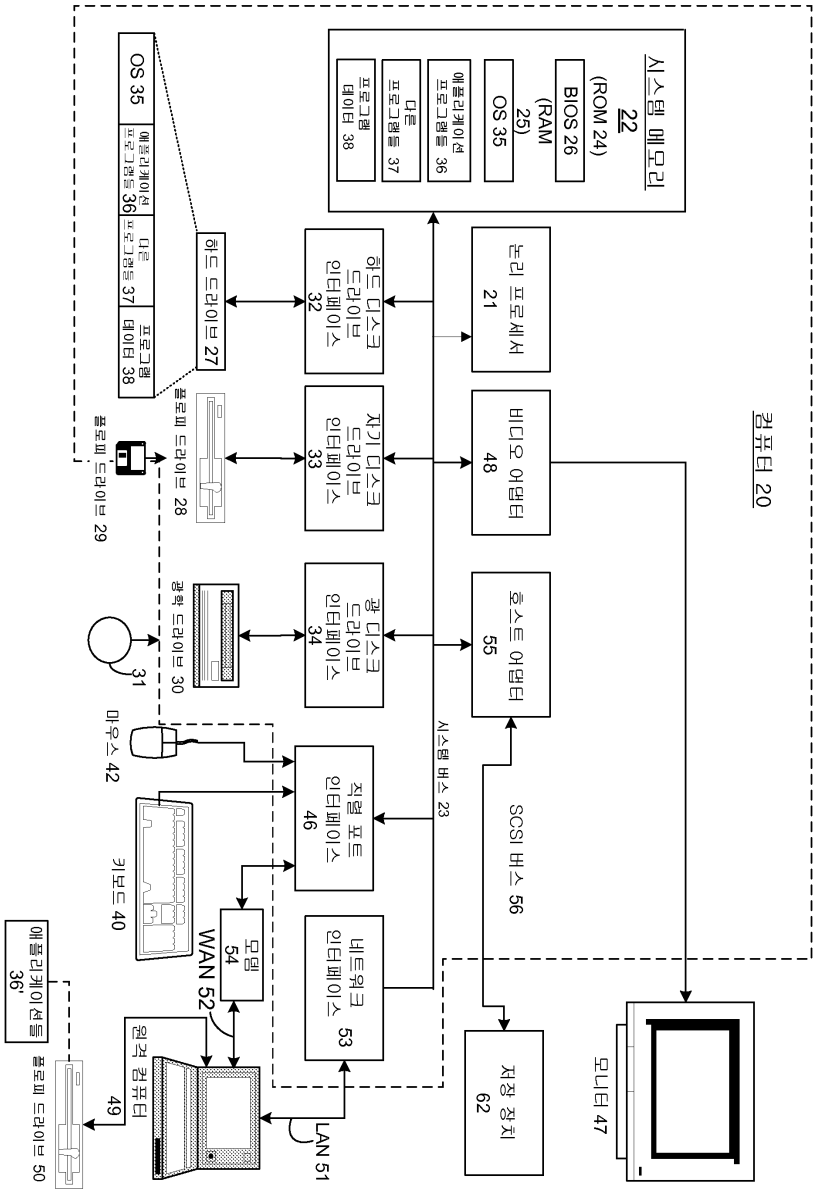
[0079] 계속해서 도 17에 대하여 서술하면, 작업 1716은 게스트 운영 체제의 현재 메모리 압력이 목표 임계치보다 높다는 판단에 기초하여 복수의 가상 NUMA 노드들 중 제 1 가상 NUMA 노드에 메모리를 커밋하는 단계를 보여준다. 예를 들면, 실시예에서 DMVSP(802)는 가상 NUMA 노드(606)의 메모리 압력이 목표 임계치보다 높은 경우에 가상 NUMA 노드(606)에 메모리를 커밋하도록 설정될 수 있다. 이 경우에, 만약에 메모리가 사용 가능하다면 메모리는 가상 NUMA 노드(606)에 할당될 수 있다. 즉, DMVSP(802)는 사용 가능한 메모리 용량에 의해 가상 NUMA 노드(606)의 퍼포먼스가 어떤 영향을 받는지 여부를 식별하는 메모리 압력 정보를 얻을 수 있고 가상 NUMA 노드(606)에 메모리를 추가할 수 있다. 특정 예시에서 메모리 압력 정보는 수치일 수 있다. 이 예시 안에서 DMVSP(802)는 현재 메모리 압력 수치를 가상 NUMA 노드(606)의 최소 수치를 나타내는 정보를 포함하는 표와 비교할 수 있고 가상 NUMA 노드(606)의 메모리 압력이 최소치와 같아질 때까지 메모리를 조절할 수 있다. 예를 들면 관리자는 크리터컬 애플리케이션(critical application)을 실행하는 게스트 운영 체제로 하여금 가상 NUMA 노드들(606, 608)에 대한 최저 수치들을 가지도록 설정할 수 있다.

[0080] 앞서 논의된 내용은 예시들 및/또는 작업 다이어그램들을 통해서 시스템들 및/또는 프로세스의 다양한 실시예를 서술한다. 그들 블록 다이어그램, 및/또는 하나 또는 그 이상의 기능들 및/또는 작업들을 포함하는 예시들에 한해서는, 본 기술의 당업자 입장에서는 이들 블록 다이어그램들, 또는 예시들 내의 각각의 기능 및/또는 작업들이 독립적으로 및/또는 함께 구현 가능하고 이는 광범위한 하드웨어, 소프트웨어, 펌웨어, 또는 그들 간의 어떠한 조합으로도 구현 가능하다.

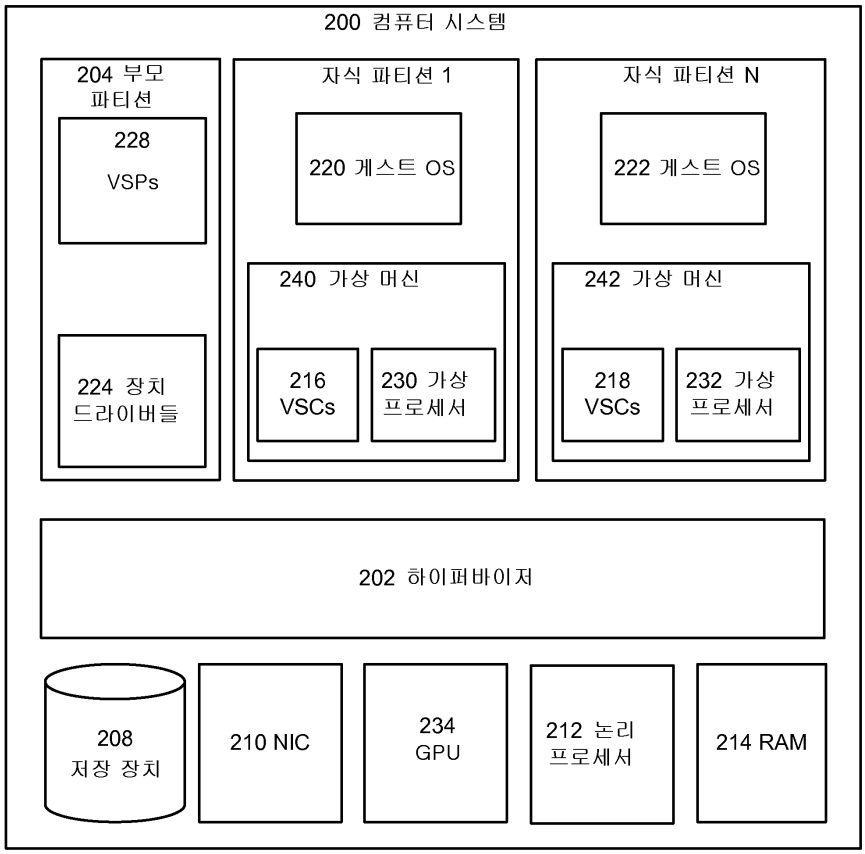
[0081] 본 발명이 속한 분야의 당업자라면 상술 된 설명 및 첨부된 도면 내에 제시된 개시 내용의 이점을 가지면서 본 명세서에 제시된 본 발명의 여러 수정 및 다른 실시예를 생각해 낼 수 있을 것이다. 그러므로 본 발명은 개시되어 있는 특정한 실시예로 한정되지 않고, 수정 및 다른 실시예는 첨부된 청구항의 범주에 속하도록 의도되었다는 것을 이해할 것이다.

도면

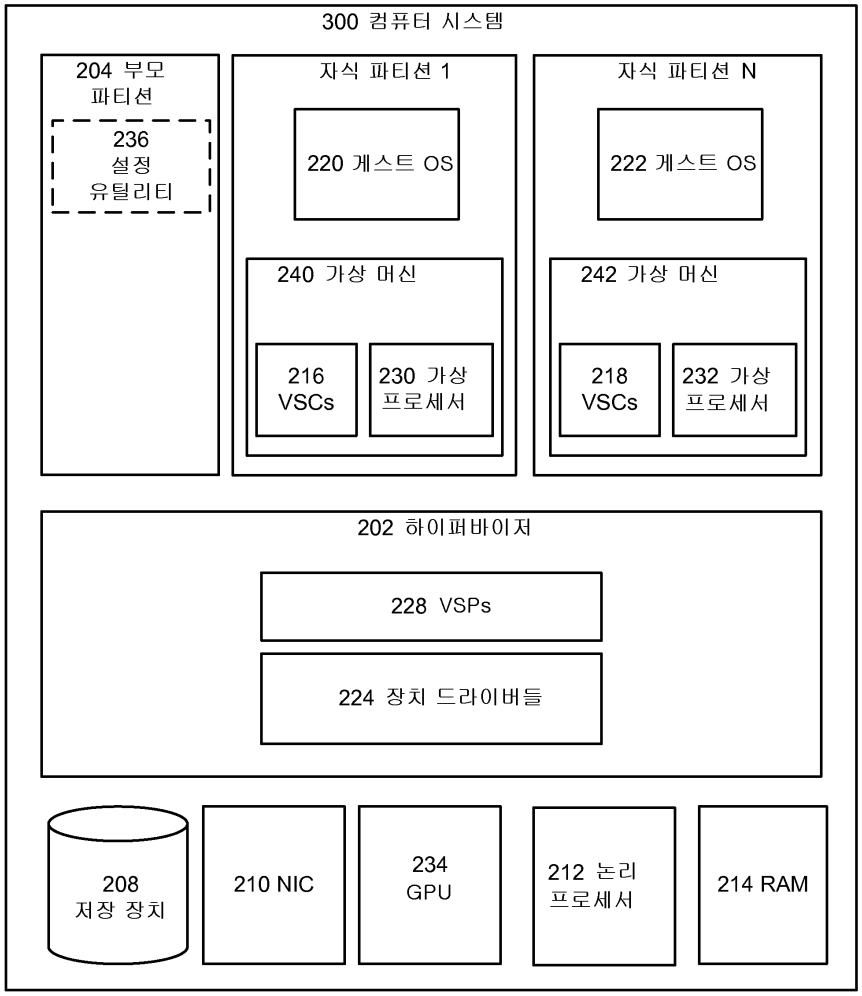
도면1


컴퓨터 20
시스템 메모리 22
(ROM 24)
BIOS 26
(RAM 25)
OS 35
에플리케이션 프로그램들 36
다른 프로그램들 37
프로그램 데이터 38
논리 프로세서 21
비디오 어댑터 48
호스트 어댑터 55
하드 디스크 드라이브 인터페이스 32
자기 디스크 드라이브 인터페이스 33
광 디스크 드라이브 인터페이스 34
직렬 포트 인터페이스 46
네트워크 인터페이스 53
시스템 버스 23
SCSI 버스 56
하드 드라이브 27
OS 35
에플리케이션 프로그램들 36
다른 프로그램들 37
프로그램 데이터 38
플로피 드라이브 28
플로피 드라이브 29
광학 드라이브 30
31
마우스 42
키보드 40
모뎀 54
WAN 52
LAN 51
원격 컴퓨터 49
에플리케이션들 36'
플로피 드라이브 50
저장 장치 62
모니터 47

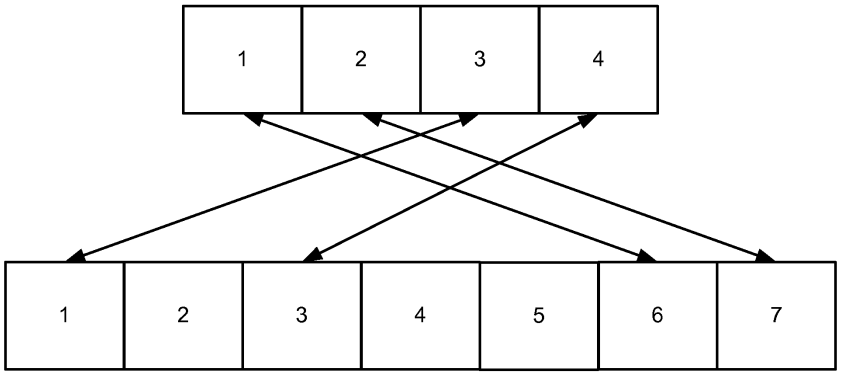
도면2


200 컴퓨터 시스템
204 부모 파티션
228 VSPs
224 장치 드라이버들
자식 파티션 1
220 게스트 OS
240 가상 머신
216 VSCs
230 가상 프로세서
자식 파티션 N
222 게스트 OS
242 가상 머신
218 VSCs
232 가상 프로세서
202 하이퍼바이저
208 저장 장치
210 NIC
234 GPU
212 논리 프로세서
214 RAM

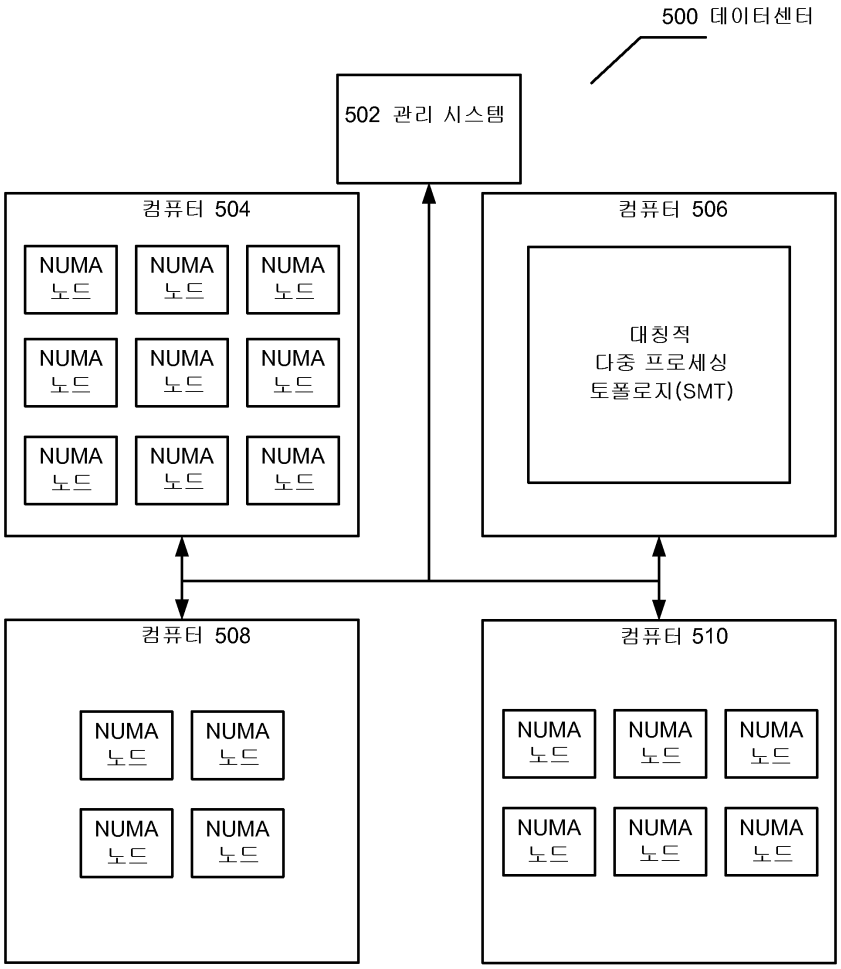
도면3


300 컴퓨터 시스템
204 부모 파티션
236 설정 유틸리티
자식 파티션 1
220 게스트 OS
240 가상 머신
216 VSCs
230 가상 프로세서
자식 파티션 N
222 게스트 OS
242 가상 머신
218 VSCs
232 가상 프로세서
202 하이퍼바이저
228 VSPs
224 장치 드라이버들
208 저장 장치
210 NIC
234 GPU
212 논리 프로세서
214 RAM

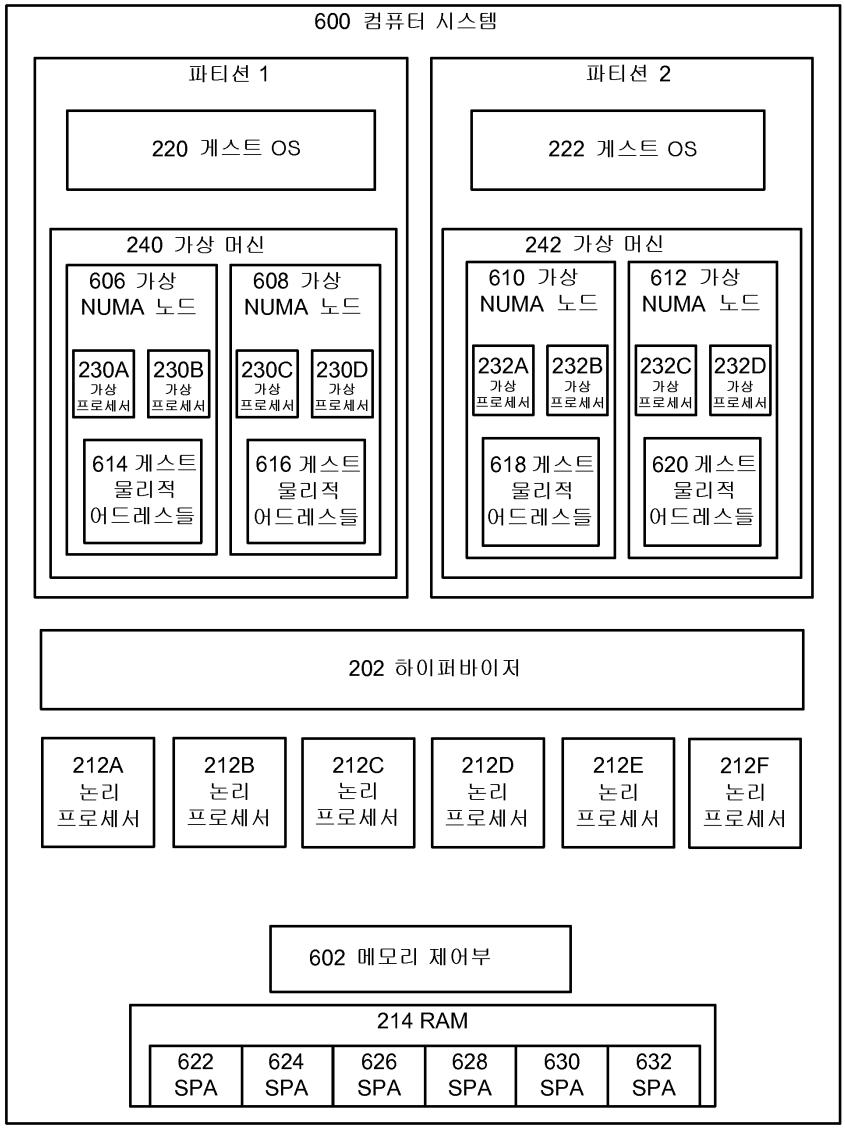
도면4


게스트 물리적 어드레스
1
2
3
4
1
2
3
4
5
6
7
시스템 물리적 어드레스

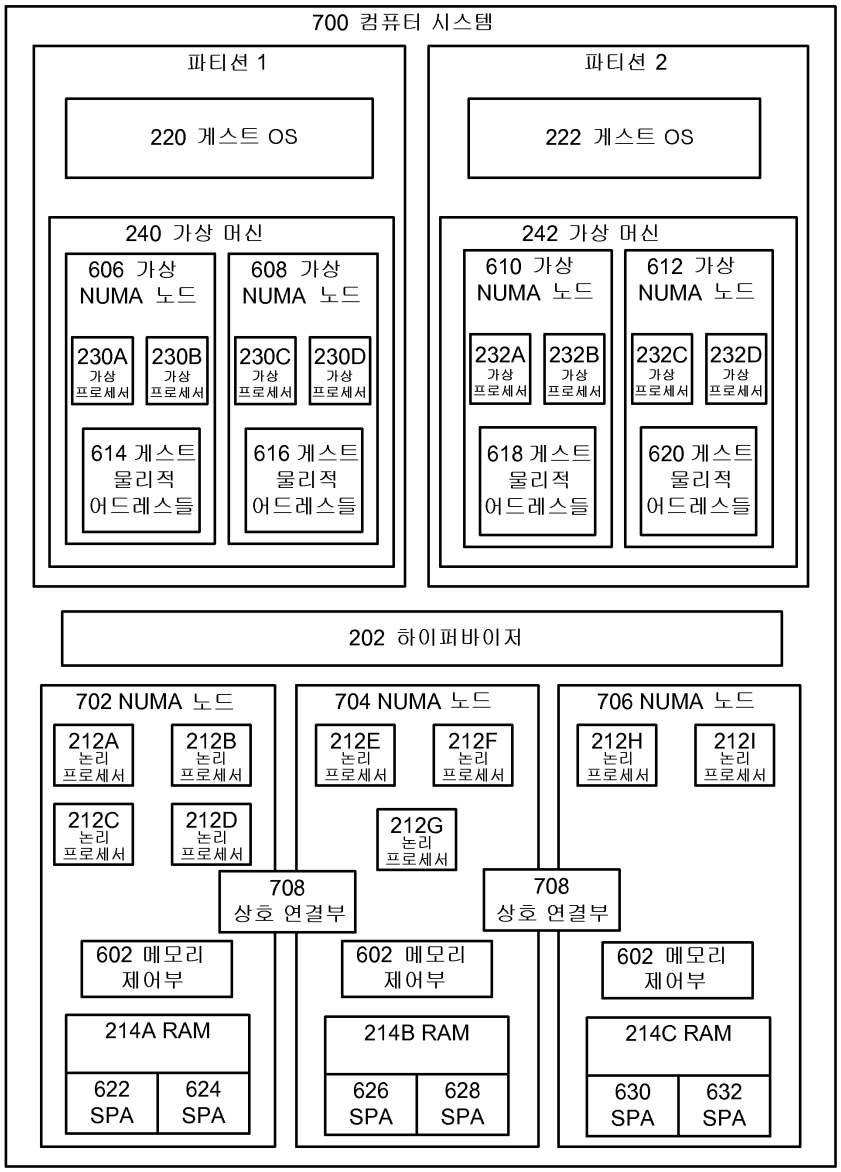
도면5


500 데이터센터
502 관리 시스템
컴퓨터 504
NUMA 노드
NUMA 노드
NUMA 노드
NUMA 노드
NUMA 노드
NUMA 노드
NUMA 노드
NUMA 노드
NUMA 노드
컴퓨터 506
대칭적
다중 프로세싱
토폴로지(SMT)
컴퓨터 508
NUMA 노드
NUMA 노드
NUMA 노드
NUMA 노드
컴퓨터 510
NUMA 노드
NUMA 노드
NUMA 노드
NUMA 노드
NUMA 노드
NUMA 노드

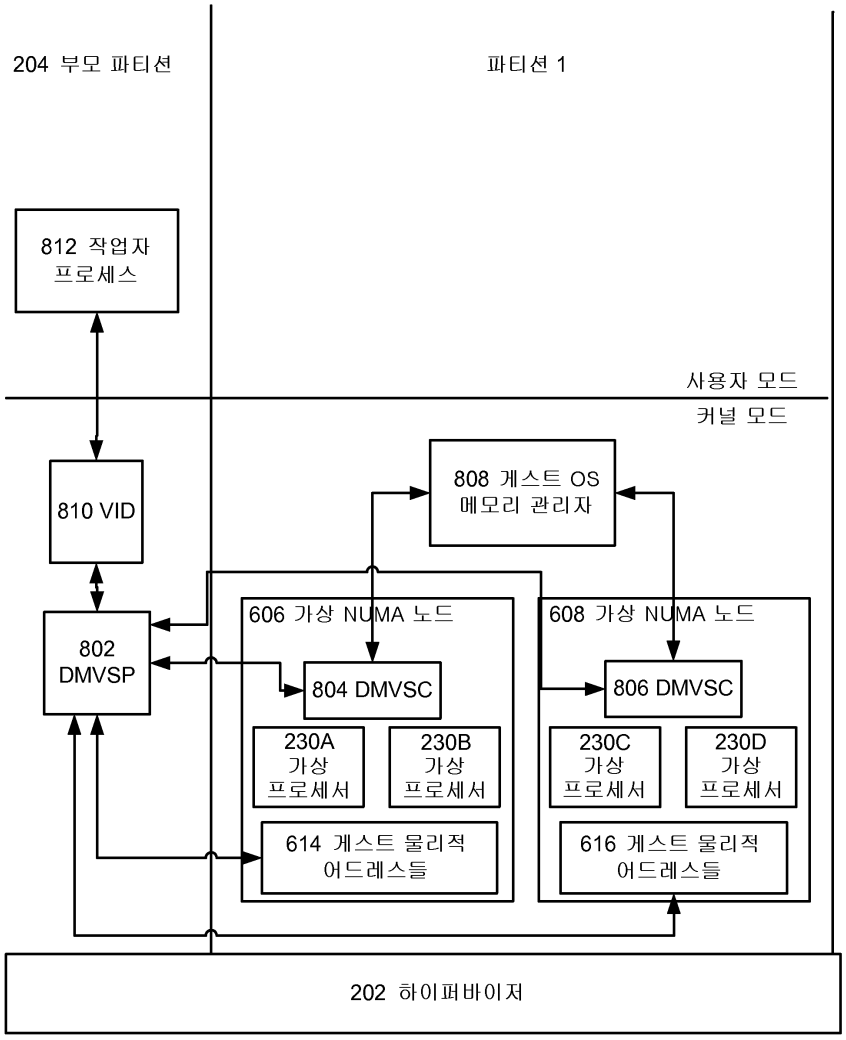
도면6


600 컴퓨터 시스템
파티션 1
220 게스트 OS
240 가상 머신
606 가상 NUMA 노드
608 가상 NUMA 노드
230A 가상 프로세서
230B 가상 프로세서
230C 가상 프로세서
230D 가상 프로세서
614 게스트 물리적 어드레스들
616 게스트 물리적 어드레스들
파티션 2
222 게스트 OS
242 가상 머신
610 가상 NUMA 노드
612 가상 NUMA 노드
232A 가상 프로세서
232B 가상 프로세서
232C 가상 프로세서
232D 가상 프로세서
618 게스트 물리적 어드레스들
620 게스트 물리적 어드레스들
202 하이퍼바이저
212A 논리 프로세서
212B 논리 프로세서
212C 논리 프로세서
212D 논리 프로세서
212E 논리 프로세서
212F 논리 프로세서
602 메모리 제어부
214 RAM
622 SPA
624 SPA
626 SPA
628 SPA
630 SPA
632 SPA

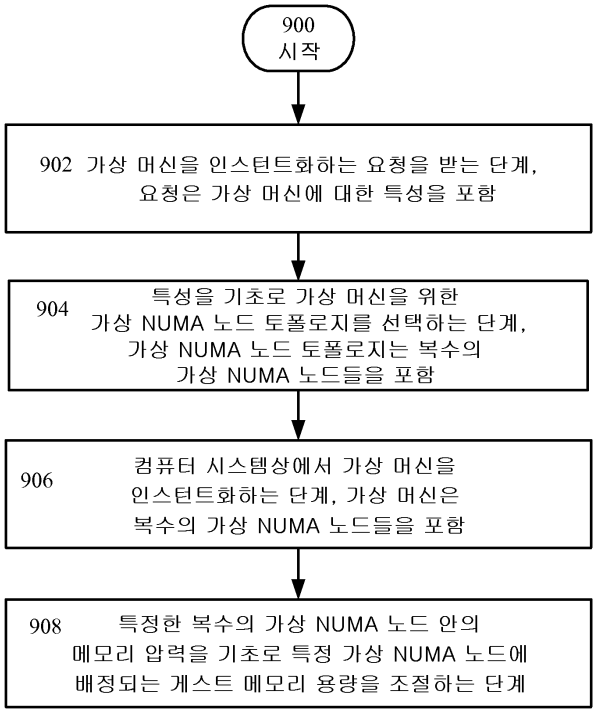
도면7


700 컴퓨터 시스템
파티션 1
220 게스트 OS
240 가상 머신
606 가상 NUMA 노드
608 가상 NUMA 노드
230A 가상 프로세서
230B 가상 프로세서
230C 가상 프로세서
230D 가상 프로세서
614 게스트 물리적 어드레스들
616 게스트 물리적 어드레스들
파티션 2
222 게스트 OS
242 가상 머신
610 가상 NUMA 노드
612 가상 NUMA 노드
232A 가상 프로세서
232B 가상 프로세서
232C 가상 프로세서
232D 가상 프로세서
618 게스트 물리적 어드레스들
620 게스트 물리적 어드레스들
202 하이퍼바이저
702 NUMA 노드
704 NUMA 노드
706 NUMA 노드
212A 논리 프로세서
212B 논리 프로세서
212C 논리 프로세서
212D 논리 프로세서
212E 논리 프로세서
212F 논리 프로세서
212G 논리 프로세서
212H 논리 프로세서
212I 논리 프로세서
708 상호 연결부
708 상호 연결부
602 메모리 제어부
602 메모리 제어부
602 메모리 제어부
214A RAM
622 SPA
624 SPA
214B RAM
626 SPA
628 SPA
214C RAM
630 SPA
632 SPA

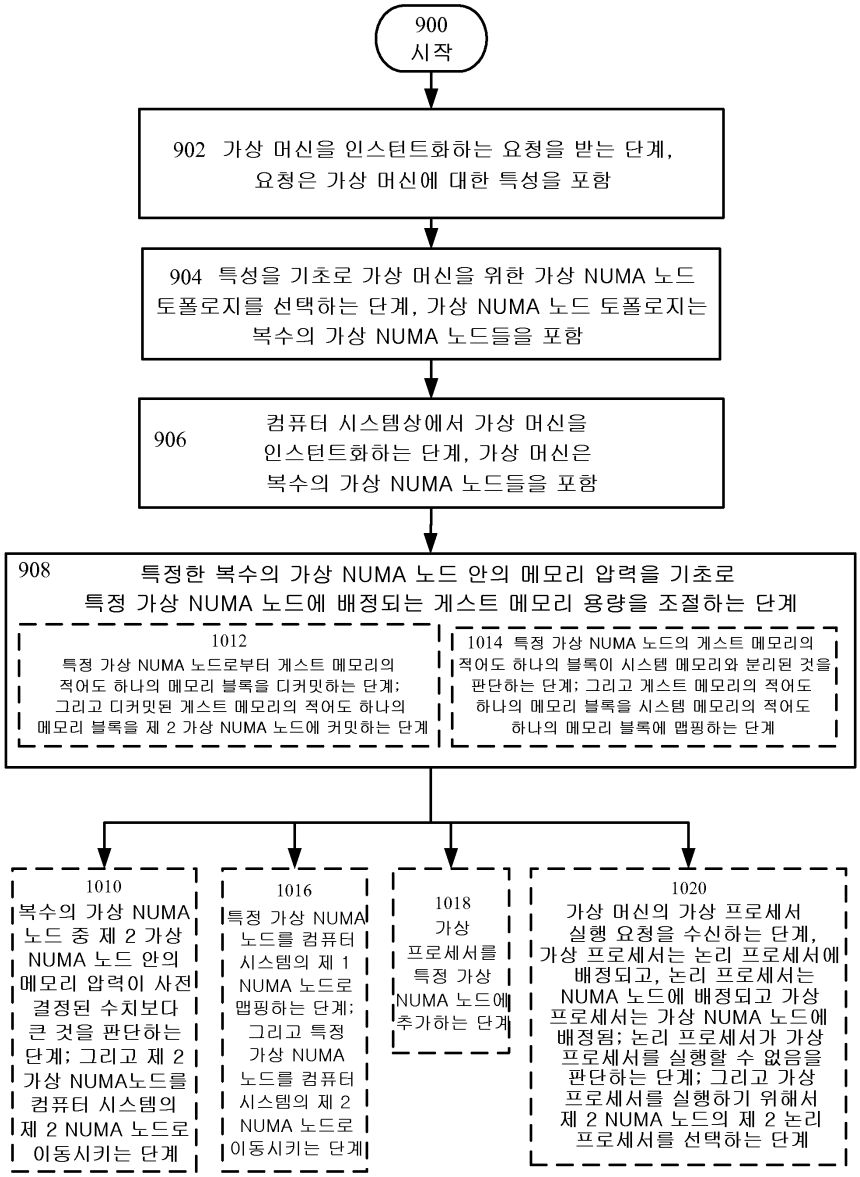
도면8


204 부모 파티션
파티션 1
812 작업자 프로세스
사용자 모드
커널 모드
808 게스트 OS 메모리 관리자
810 VID
802 DMVSP
606 가상 NUMA 노드
608 가상 NUMA 노드
804 DMVSC
806 DMVSC
230A 가상 프로세서
230B 가상 프로세서
230C 가상 프로세서
230D 가상 프로세서
614 게스트 물리적 어드레스들
616 게스트 물리적 어드레스들
202 하이퍼바이저

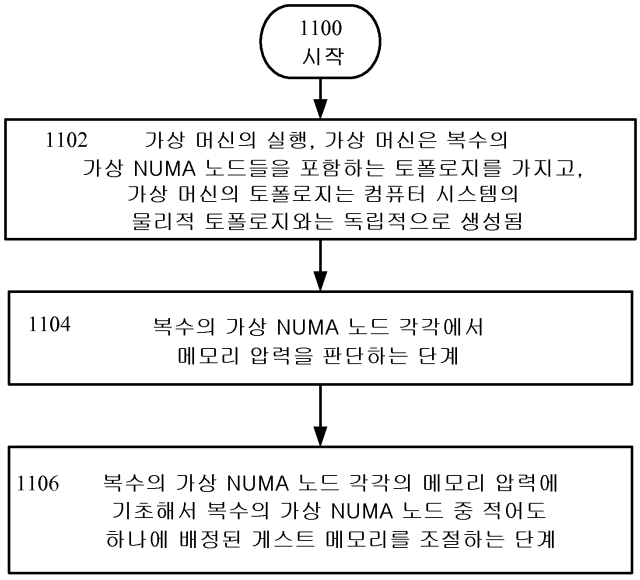
도면9


900
시작
902 가상 머신을 인스턴트화하는 요청을 받는 단계, 요청은 가상 머신에 대한 특성을 포함
904 특성을 기초로 가상 머신을 위한 가상 NUMA 노드 토폴로지를 선택하는 단계, 가상 NUMA 노드 토폴로지는 복수의 가상 NUMA 노드들을 포함
906 컴퓨터 시스템상에서 가상 머신을 인스턴트화하는 단계, 가상 머신은 복수의 가상 NUMA 노드들을 포함
908 특정한 복수의 가상 NUMA 노드 안의 메모리 압력을 기초로 특정 가상 NUMA 노드에 배정되는 게스트 메모리 용량을 조절하는 단계

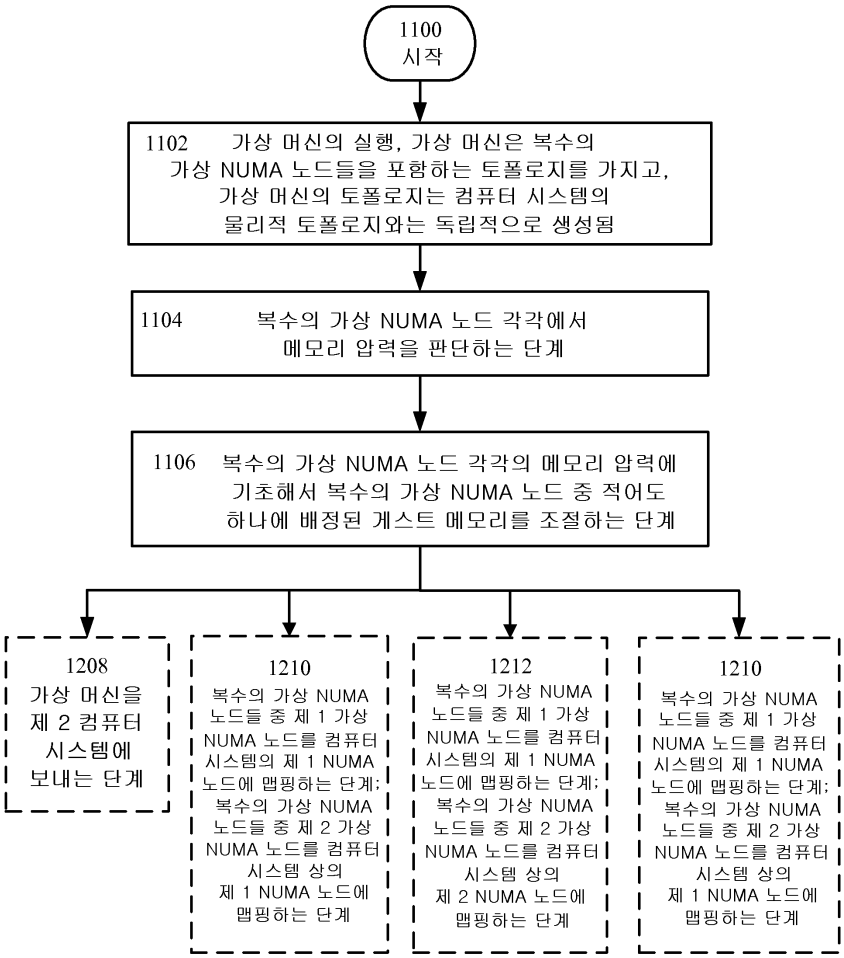
도면10


900
시작
902 가상 머신을 인스턴트화하는 요청을 받는 단계, 요청은 가상 머신에 대한 특성을 포함
904 특성을 기초로 가상 머신을 위한 가상 NUMA 노드 토폴로지를 선택하는 단계, 가상 NUMA 노드 토폴로지는 복수의 가상 NUMA 노드들을 포함
906 컴퓨터 시스템상에서 가상 머신을 인스턴트화하는 단계, 가상 머신은 복수의 가상 NUMA 노드들을 포함
908 특정한 복수의 가상 NUMA 노드 안의 메모리 압력을 기초로 특정 가상 NUMA 노드에 배정되는 게스트 메모리 용량을 조절하는 단계
1012 특정 가상 NUMA 노드로부터 게스트 메모리의 적어도 하나의 메모리 블록을 디커밋하는 단계; 그리고 디커밋된 게스트 메모리의 적어도 하나의 메모리 블록을 제 2 가상 NUMA 노드에 커밋하는 단계
1014 특정 가상 NUMA 노드의 게스트 메모리의 적어도 하나의 블록이 시스템 메모리와 분리된 것을 판단하는 단계; 그리고 게스트 메모리의 적어도 하나의 메모리 블록을 시스템 메모리의 적어도 하나의 메모리 블록에 맵핑하는 단계
1010 복수의 가상 NUMA 노드 중 제 2 가상 NUMA 노드 안의 메모리 압력이 사전 결정된 수치보다 큰 것을 판단하는 단계; 그리고 제 2 가상 NUMA노드를 컴퓨터 시스템의 제 2 NUMA 노드로 이동시키는 단계
1016 특정 가상 NUMA 노드를 컴퓨터 시스템의 제 1 NUMA 노드로 맵핑하는 단계; 그리고 특정 가상 NUMA 노드를 컴퓨터 시스템의 제 2 NUMA 노드로 이동시키는 단계
1018 가상 프로세서를 특정 가상 NUMA 노드에 추가하는 단계
1020 가상 머신의 가상 프로세서 실행 요청을 수신하는 단계, 가상 프로세서는 논리 프로세서에 배정되고, 논리 프로세서는 NUMA 노드에 배정되고 가상 프로세서는 가상 NUMA 노드에 배정됨; 논리 프로세서가 가상 프로세서를 실행할 수 없음을 판단하는 단계; 그리고 가상 프로세서를 실행하기 위해서 제 2 NUMA 노드의 제 2 논리 프로세서를 선택하는 단계

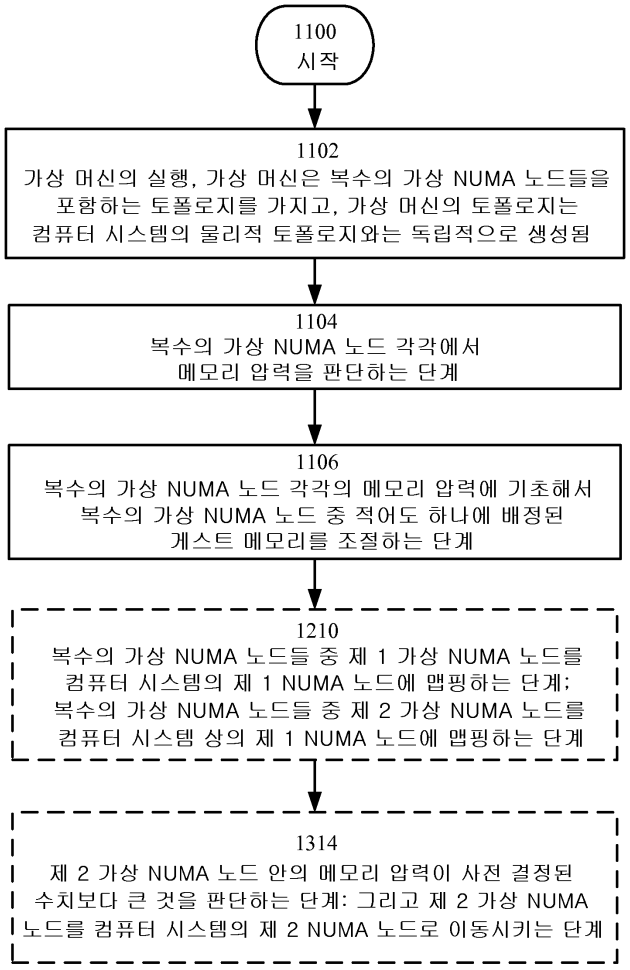
도면11


1100
시작
1102 가상 머신의 실행, 가상 머신은 복수의 가상 NUMA 노드들을 포함하는 토폴로지를 가지고, 가상 머신의 토폴로지는 컴퓨터 시스템의 물리적 토폴로지와는 독립적으로 생성됨
1104 복수의 가상 NUMA 노드 각각에서 메모리 압력을 판단하는 단계
1106 복수의 가상 NUMA 노드 각각의 메모리 압력에 기초해서 복수의 가상 NUMA 노드 중 적어도 하나에 배정된 게스트 메모리를 조절하는 단계

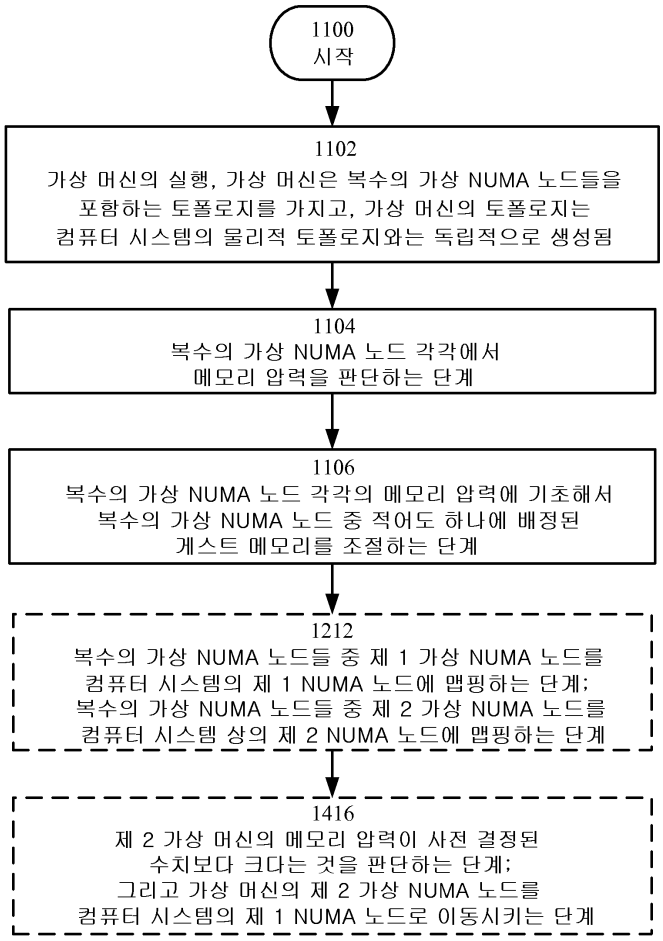
도면12


1100
시작
1102 가상 머신의 실행, 가상 머신은 복수의 가상 NUMA 노드들을 포함하는 토폴로지를 가지고, 가상 머신의 토폴로지는 컴퓨터 시스템의 물리적 토폴로지와는 독립적으로 생성됨
1104 복수의 가상 NUMA 노드 각각에서 메모리 압력을 판단하는 단계
1106 복수의 가상 NUMA 노드 각각의 메모리 압력에 기초해서 복수의 가상 NUMA 노드 중 적어도 하나에 배정된 게스트 메모리를 조절하는 단계
1208 가상 머신을 제 2 컴퓨터 시스템에 보내는 단계
1210 복수의 가상 NUMA 노드들 중 제 1 가상 NUMA 노드를 컴퓨터 시스템의 제 1 NUMA 노드에 맵핑하는 단계; 복수의 가상 NUMA 노드들 중 제 2 가상 NUMA 노드를 컴퓨터 시스템 상의 제 1 NUMA 노드에 맵핑하는 단계
1212 복수의 가상 NUMA 노드들 중 제 1 가상 NUMA 노드를 컴퓨터 시스템의 제 1 NUMA 노드에 맵핑하는 단계; 복수의 가상 NUMA 노드들 중 제 2 가상 NUMA 노드를 컴퓨터 시스템 상의 제 2 NUMA 노드에 맵핑하는 단계
1210 복수의 가상 NUMA 노드들 중 제 1 가상 NUMA 노드를 컴퓨터 시스템의 제 1 NUMA 노드에 맵핑하는 단계; 복수의 가상 NUMA 노드들 중 제 2 가상 NUMA 노드를 컴퓨터 시스템 상의 제 1 NUMA 노드에 맵핑하는 단계

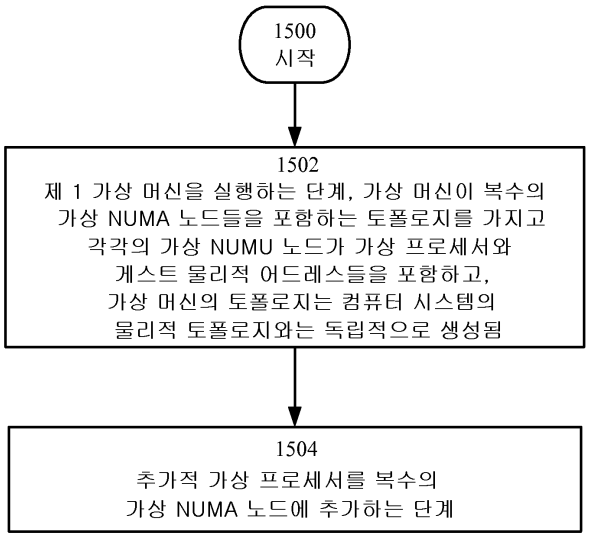
도면13


1100
시작
1102
가상 머신의 실행, 가상 머신은 복수의 가상 NUMA 노드들을 포함하는 토폴로지를 가지고, 가상 머신의 토폴로지는 컴퓨터 시스템의 물리적 토폴로지와는 독립적으로 생성됨
1104
복수의 가상 NUMA 노드 각각에서 메모리 압력을 판단하는 단계
1106
복수의 가상 NUMA 노드 각각의 메모리 압력에 기초해서 복수의 가상 NUMA 노드 중 적어도 하나에 배정된 게스트 메모리를 조절하는 단계
1210
복수의 가상 NUMA 노드들 중 제 1 가상 NUMA 노드를 컴퓨터 시스템의 제 1 NUMA 노드에 맵핑하는 단계; 복수의 가상 NUMA 노드들 중 제 2 가상 NUMA 노드를 컴퓨터 시스템 상의 제 1 NUMA 노드에 맵핑하는 단계
1314
제 2 가상 NUMA 노드 안의 메모리 압력이 사전 결정된 수치보다 큰 것을 판단하는 단계: 그리고 제 2 가상 NUMA 노드를 컴퓨터 시스템의 제 2 NUMA 노드로 이동시키는 단계

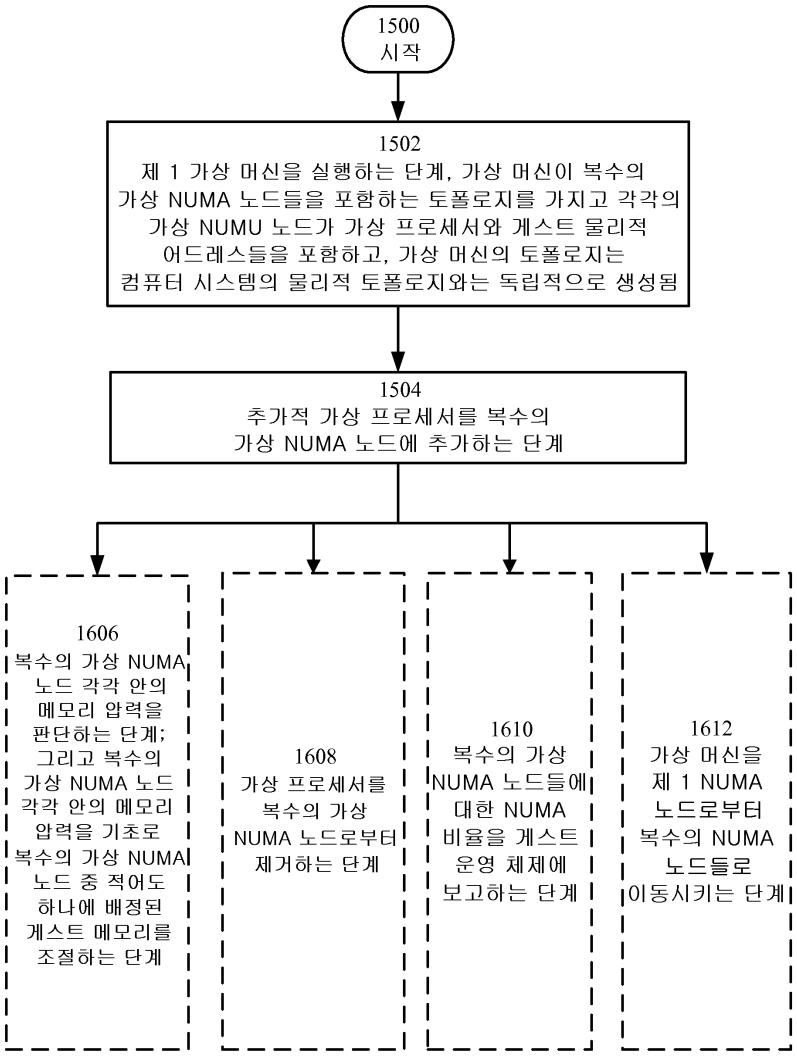
도면14


1100
시작
1102
가상 머신의 실행, 가상 머신은 복수의 가상 NUMA 노드들을
포함하는 토폴로지를 가지고, 가상 머신의 토폴로지는
컴퓨터 시스템의 물리적 토폴로지와는 독립적으로 생성됨
1104
복수의 가상 NUMA 노드 각각에서
메모리 압력을 판단하는 단계
1106
복수의 가상 NUMA 노드 각각의 메모리 압력에 기초해서
복수의 가상 NUMA 노드 중 적어도 하나에 배정된
게스트 메모리를 조절하는 단계
1212
복수의 가상 NUMA 노드들 중 제 1 가상 NUMA 노드를
컴퓨터 시스템의 제 1 NUMA 노드에 맵핑하는 단계;
복수의 가상 NUMA 노드들 중 제 2 가상 NUMA 노드를
컴퓨터 시스템 상의 제 2 NUMA 노드에 맵핑하는 단계
1416
제 2 가상 머신의 메모리 압력이 사전 결정된
수치보다 크다는 것을 판단하는 단계;
그리고 가상 머신의 제 2 가상 NUMA 노드를
컴퓨터 시스템의 제 1 NUMA 노드로 이동시키는 단계

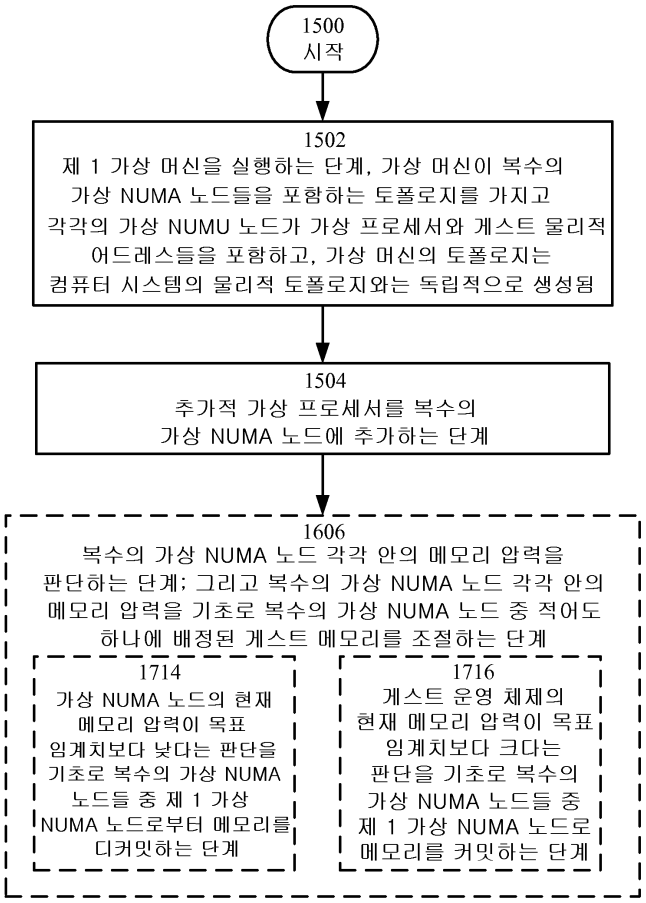
도면15


1500
시작
1502
제 1 가상 머신을 실행하는 단계, 가상 머신이 복수의
가상 NUMA 노드들을 포함하는 토폴로지를 가지고
각각의 가상 NUMU 노드가 가상 프로세서와
게스트 물리적 어드레스들을 포함하고,
가상 머신의 토폴로지는 컴퓨터 시스템의
물리적 토폴로지와는 독립적으로 생성됨
1504
추가적 가상 프로세서를 복수의
가상 NUMA 노드에 추가하는 단계

도면16


1500
시작
1502
제 1 가상 머신을 실행하는 단계, 가상 머신이 복수의 가상 NUMA 노드들을 포함하는 토폴로지를 가지고 각각의 가상 NUMU 노드가 가상 프로세서와 게스트 물리적 어드레스들을 포함하고, 가상 머신의 토폴로지는 컴퓨터 시스템의 물리적 토폴로지와는 독립적으로 생성됨
1504
추가적 가상 프로세서를 복수의 가상 NUMA 노드에 추가하는 단계
1606
복수의 가상 NUMA 노드 각각 안의 메모리 압력을 판단하는 단계; 그리고 복수의 가상 NUMA 노드 각각 안의 메모리 압력을 기초로 복수의 가상 NUMA 노드 중 적어도 하나에 배정된 게스트 메모리를 조절하는 단계
1608
가상 프로세서를 복수의 가상 NUMA 노드로부터 제거하는 단계
1610
복수의 가상 NUMA 노드들에 대한 NUMA 비율을 게스트 운영 체제에 보고하는 단계
1612
가상 머신을 제 1 NUMA 노드로부터 복수의 NUMA 노드들로 이동시키는 단계

도면17



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 청구범위

【보정세부항목】 제14항

【변경전】

상기 컴퓨터 시스템

【변경후】

컴퓨터 시스템