(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년10월12일
(11) 등록번호 10-1907564
(24) 등록일자 2018년10월05일

(51) 국제특허분류(Int. Cl.)
G06F 9/46 (2006.01) *G06F 9/22* (2018.01)
(21) 출원번호 10-2013-7032527
(22) 출원일자(국제) 2012년06월07일
심사청구일자 2017년05월04일
(85) 번역문제출일자 2013년12월06일
(65) 공개번호 10-2014-0033393
(43) 공개일자 2014년03월18일
(86) 국제출원번호 PCT/US2012/041434
(87) 국제공개번호 WO 2012/170746
국제공개일자 2012년12월13일
(30) 우선권주장
13/155,387 2011년06월08일 미국(US)
(56) 선행기술조사문헌
WO2011047906 A1
(뒷면에 계속)

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

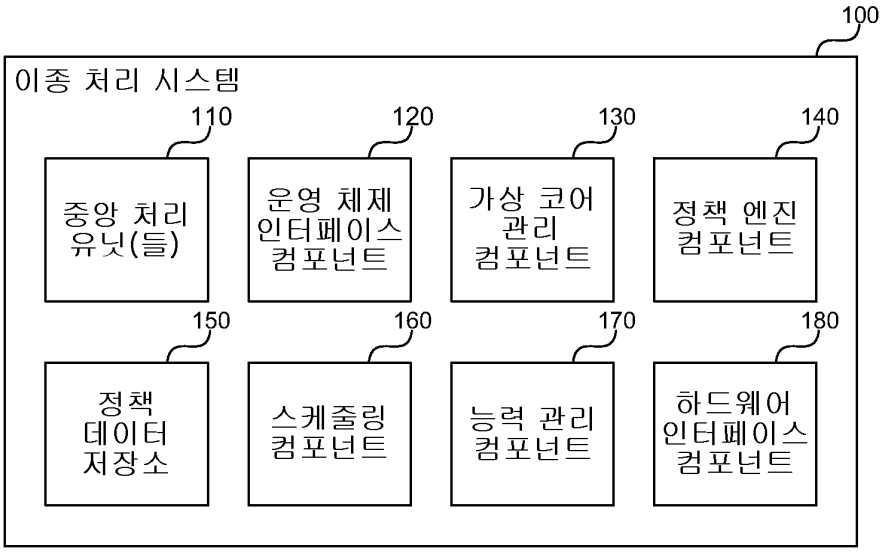
전체 청구항 수 : 총 20 항 심사관 : 유진태

(54) 발명의 명칭 **운영 체제 분리형 이종 컴퓨팅 기법**

(57) 요 약

본 명세서에서는 에너지 효율을 향상시키거나 다른 처리 목표들을 충족시키기 위하여 운영 체제의 인지 또는 관여 없이 크고 작은 코어들에 걸치는 운영 체제 스레드 스케줄링을 자율적으로 제어하기 위한 하이퍼바이저를 제공하는 이종 처리 시스템이 설명된다. 시스템은 가상화된 컴퓨팅 코어들의 유한 세트를 운영 체제에 제공하며, 시스템은 실행할 스레드들을 유한 세트에 스케줄링한다. 이어서, 표면 아래에서, 하이퍼바이저는 에너지 사용 또는 다른 처리 요구들을 관리하기 위해 각각의 스레드를 실행할 크거나 작은 코어(들)의 물리적 할당 및 선택을 지능적으로 제어한다. 소프트웨어 하이퍼바이저를 이용하여 기본적인 크고 작은 컴퓨터 아키텍처를 추상화함으로써, 코어들 간의 성능 및 전력 운영 차이들이 운영 체제에 대해 불투명하게 유지된다. 고유의 간접 지정(indirection)은 또한 운영 체제 출시 스케줄로부터 새로운 능력들을 갖는 하드웨어의 출시를 분리한다.

대 표 도 - 도1



(56) 선행기술조사문헌
JP2004192612 A
US20090037911 A1
US20090055826 A1
Youngjin Kwon et al. 'Virtualizing performance asymmetric multi-core systems'. 2011 38th Annual International Symposium on Computer Architecture, 2011.6, pp.45-56.

명 세 서

청구범위

청구항 1

이종 처리 코어를 관리하는 하이퍼바이저를 통해 하나 이상의 운영 체제 스레드를 스케줄링하는 컴퓨터 구현 방법으로서,

운영 체제에 가상 코어들의 세트를 제공하는 단계 - 상기 가상 코어들은 컴퓨팅 장치에 액세스할 수 있는 물리적 처리 코어들 사이의 하나 이상의 능력 차이를 알지 못하게 상기 운영 체제를 격리시킴 - 과,

상기 운영 체제로부터, 상기 하이퍼바이저에 의해 제공된 상기 가상 코어들의 세트 중에서 식별된 가상 코어에서 스레드의 명령어를 실행하라는 스레드 스케줄링 요청을 수신하는 단계와,

상기 스레드를 실행하기 전에, 상기 수신된 스케줄링 요청의 하나 이상의 처리 요구를 결정하는 단계와,

상기 컴퓨팅 장치를 동작시키기 위한 하나 이상의 목표를 지정하는 스케줄링 정책에 액세스하는 단계 - 상기 목표 중 적어도 하나는 전력 사용과 관련됨 - 와,

상기 수신된 스케줄링 요청과 관련된 상기 스레드를 실행할 물리적 처리 코어를 선택하는 단계 - 상기 선택은 상기 액세스된 스케줄링 정책 및 이용가능한 시스템 설비에 기초하여 이루어짐 - 와,

상기 선택된 물리적 처리 코어에서 실행하도록 상기 스레드를 스케줄링하는 단계

를 포함하는 방법.

청구항 2

제1항에 있어서,

상기 스레드 스케줄링 요청을 수신하는 단계는, 상기 운영 체제가 스레드의 실행을 위해 선택하는 특정한 가상 코어에 관계없이, 하이퍼바이저가 하나 이상의 하이퍼바이저 정책에 따라 상기 스레드를 실행할 임의의 특정한 물리적 처리 코어를 선택하는 단계를 포함하는 방법.

청구항 3

제1항에 있어서,

상기 처리 요구를 결정하는 단계는 상기 스케줄링된 스레드에 의해 사용되는 특정 명령어 세트, 이용 가능 코프로세서(co-processor), 명령어 세트 확장 또는 아키텍처 확장을 결정하는 단계를 포함하는 방법.

청구항 4

제1항에 있어서,

상기 처리 요구를 결정하는 단계는 상기 스레드의 하나 이상의 성능 요건을 결정하는 단계를 포함하는 방법.

청구항 5

제1항에 있어서,

상기 처리 요구를 결정하는 단계는 상기 스레드가 더 적은 전력 사용에서 더 느린 실행에 적합한지를 결정하는 단계를 포함하는 방법.

청구항 6

제1항에 있어서,

상기 처리 요구를 결정하는 단계는 상기 스레드가 추가적인 처리 자원이 이용 가능할 때까지 지연될 수 있는지를 결정하는 단계를 포함하는 방법.

청구항 7

제1항에 있어서,

상기 처리 요구를 결정하는 단계는 특정 운영 체제에 대한 특정 지식에 액세스하여 특정 스레드의 처리 요구를 결정하는 단계를 포함하는 방법.

청구항 8

제1항에 있어서,

상기 스케줄링 정책에 액세스하는 단계는, 상기 정책이 전력 사용의 최적화를 요청하고 스레드의 스케줄링을 이용 가능한 저전력 처리 코어로 지정하도록 결정하는 단계를 포함하는 방법.

청구항 9

제1항에 있어서,

상기 스케줄링 정책에 액세스하는 단계는, 상기 정책이 성능의 최적화를 요청하고 스레드의 스케줄링을 이용 가능한 고성능 처리 코어로 지정하도록 결정하는 단계를 포함하는 방법.

청구항 10

제1항에 있어서,

상기 스케줄링 정책에 액세스하는 단계는, 상기 컴퓨팅 장치가 이용 가능한 전력이 제한된 이동 장치 또는 열적 한계를 갖는 장치이고 스레드를 스케줄링할 코어의 선택을 통한 전력 사용의 최적화 또는 열 관리를 지정하도록 결정하는 단계를 포함하는 방법.

청구항 11

제1항에 있어서,

상기 물리적 처리 코어를 선택하는 단계는 상기 시스템이 상기 스레드를 스케줄링할 수 있는 상이한 능력 및 성능/전력 특성을 가진 다수의 이용 가능 코어 중 하나를 선택하는 단계를 포함하는 방법.

청구항 12

제1항에 있어서,

상기 물리적 처리 코어를 선택하는 단계는 상기 컴퓨팅 장치의 성능 및 전력 사용을 관리하기 위한 하나 이상의 목표를 촉진하는 방식으로 상기 선택을 수행하는 단계를 포함하는 방법.

청구항 13

제1항에 있어서,

상기 스레드와 상기 선택된 물리적 처리 코어 사이의 하나 이상의 능력 차이를 처리하는 단계를 더 포함하며,

상기 하나 이상의 능력 차이를 처리하는 단계는 상기 스레드가 상기 선택된 물리적 처리 코어 상에서 이용할 수 없는 명령어를 포함하는 경우, 상기 명령어를 에뮬레이션하거나 상기 명령어를 상기 선택된 물리적 처리 코어에 의해 지원되는 하나 이상의 등가 명령어로 대체하는 것을 포함하는 방법.

청구항 14

제1항에 있어서,

실행할 상기 스레드를 스케줄링하는 단계는, 출력을 처리하고 상기 출력을 상기 운영 체제에 제공하여, 상기 운영 체제에게 상기 출력이 상기 운영 체제가 상기 스레드를 할당한 상기 가상 코어로부터 오는 것으로 보이게 하는 단계를 포함하는 방법.

청구항 15

하이퍼바이저를 통해 운영 체제 분리형 이종 컴퓨팅을 제공하기 위한 컴퓨터 시스템으로서,

이종 처리 능력 및 전력 프로필을 갖는 하나 이상의 처리 코어를 포함하는 하나 이상의 처리 콤플렉스와,

하이퍼바이저와 운영 체제 사이에서 통신하여, 하드웨어 자원에 전달할 명령어를 수신하고, 상이한 코어가 상기 운영 체제에게 동일하게 보이게 하는 상기 하드웨어 자원으로부터의 출력을 수신하는 운영 체제 인터페이스 컴포넌트와,

상기 하이퍼바이저가 상기 운영 체제에 제공하는 하나 이상의 가상 코어를 관리하는 가상 코어 관리 컴포넌트 - 상기 하나 이상의 가상 코어는 상기 처리 코어 사이의 하나 이상의 능력 차이를 알지 못하게 상기 운영 체제를 격리시킴 - 와,

이용 가능한 상기 하나 이상의 처리 콤플렉스에 기초하여 운영 체제 스레드를 스케줄링하고 가상 코어를 상기 운영 체제에 제공하기 위한 하나 이상의 정책을 관리하는 정책 엔진 컴포넌트 - 상기 정책 중 적어도 하나는 전력 사용에 기초함 - 와,

선택 정책 및 이용가능한 시스템 설비에 기초하여, 상기 운영 체제로부터 스레드로서 수신된 하나 이상의 명령어 스트림을 상기 컴퓨터 시스템에 설치된 상기 처리 콤플렉스 중 하나 이상에 대해 스케줄링하는 스케줄링 컴포넌트와,

상기 하이퍼바이저와 처리 콤플렉스 사이에서 통신하여, 소프트웨어 명령어가 이용 가능한 물리 코어에서 실행되도록 스케줄링하는 하드웨어 인터페이스 컴포넌트

를 포함하는 컴퓨터 시스템.

청구항 16

제15항에 있어서,

상기 가상 코어 관리 컴포넌트는, 상기 운영 체제가 함께 동작하도록 설계되지 않은 하드웨어와 상기 운영 체제가 함께 동작할 수 있도록 하기 위해, 상기 운영 체제에게 CPU 코어로 보이지만 이용가능한 물리적 하드웨어와는 특성이 다른 가상 코어를 제공하는

컴퓨터 시스템.

청구항 17

제15항에 있어서,

상기 스케줄링 컴포넌트는 상기 운영 체제로부터, 상기 운영 체제가 상기 스레드를 스케줄링하도록 요청하는 가상 코어를 식별하는 가상 코어 식별자를 수신하는

컴퓨터 시스템.

청구항 18

제15항에 있어서,

상기 스케줄링 컴포넌트는 상기 운영 체제로부터 수신된 스케줄 요청을 검사하고, 실행할 상기 스레드를 스케줄링할 물리 코어를 결정하는

컴퓨터 시스템.

청구항 19

제15항에 있어서,

처리 코어들 사이의 하나 이상의 차이를 관리하는 능력 관리 컴포넌트(capability management component)를 더 포함하는

컴퓨터 시스템.

청구항 20

프로세서에 의해 실행가능한 명령어를 포함하는 컴퓨터 판독가능 저장 장치로서,

상기 명령어는 실행시에,

운영 체제에 가상 코어들의 세트를 제공하는 동작 - 상기 가상 코어들의 세트는 하이퍼바이저에 의해 구성되고, 상기 가상 코어들은 컴퓨팅 장치에 액세스할 수 있는 물리적 처리 코어들 사이의 하나 이상의 능력 차이를 알지 못하게 상기 운영 체제를 격리시킴 - 과,

상기 운영 체제로부터, 상기 하이퍼바이저에 의해 제공되는 상기 가상 코어들의 세트 중에서 식별된 가상 코어에서 스레드의 명령어를 실행하기 위한 스레드 스케줄링 요청을 수신하는 동작과,

상기 스레드를 실행하기 전에, 상기 수신된 스케줄링 요청의 하나 이상의 처리 요구를 결정하는 동작과,

상기 컴퓨팅 장치를 동작시키기 위한 하나 이상의 목표를 지정하는 스케줄링 정책에 액세스하는 동작 - 상기 목표 중 적어도 하나는 전력 사용과 관련됨 - 과,

상기 수신된 스케줄링 요청과 관련된 상기 스레드를 실행할 물리적 처리 코어를 선택하는 동작 - 상기 선택은 상기 액세스된 스케줄링 정책 및 이용가능한 시스템 설비에 기초하여 이루어짐 - 과,

상기 선택된 물리적 처리 코어에서 실행하도록 상기 스레드를 스케줄링하는 동작

을 수행하는 컴퓨터 판독가능 저장 장치.

발명의 설명

기 술 분 야

배 경 기 술

[0001] 에너지 효율이 점점 더 이동 전화와 데이터 센터를 구별하는 중요한 구별 인자가 되고 있다. 고객들은 더 오래 가는 이동 전화 경험들을 위해 기꺼이 프리미엄을 지불할 것이지만, 또한 이러한 동일 장치들로부터 향상되는 성능을 얻을 것을 갈망한다. 한편 규모 측면에서, 데이터 센터들은 컴퓨팅 능력을 계속 확장시키지만, 효율적으로 냉각할 수 있는 것에 대한 열적 한계에 직면하고 있다. 게다가, 대중들은 에너지 사용 및 에너지 사용의 환경적 영향을 점점 더 의식하고 있다. 따라서, 에너지를 효율적으로 사용하는 것은 많은 타입의 컴퓨팅 시스템에서의 더 높은 우선 순위의 설계 목표이다.

[0002] 이러한 기술적으로 상반되는 과제들 - 더 양호한 성능 제공, 그러나 더 적은 전력 사용 - 은 산업으로 하여금 단일 시스템 또는 실리콘 칩 내에 본 명세서에서 이종 코어들 또는 처리로서 지칭되는 "큰" 컴퓨팅 코어들과 "작은" 컴퓨팅 코어들을 밀접하게 결합하는 이종 설계들을 경험하게 하였다. 큰 코어들은 더 큰 전력 엔벨로프(envelope)에서 높은 성능을 제공하도록 설계되는 반면, 작은 코어들은 더 작은 전력 엔벨로프에서 더 낮은 성능을 제공하도록 설계된다. 전통적인 지식은 운영 체제의 스케줄러가 작업 부하(들)에 따라 큰 또는 작은 코어들 상에 스레드들을 선택적으로 스케줄링할 것이라는 것이다. 하루 중 적어도 일부 시간들 동안, 운영 체제는 큰 코어(들)를 완전히 턴 오프시키고 전력을 적게 소비하는 작은 코어들에 의존할 수 있다.

[0003] 크고 작은 코어들은 동일 명령어 세트 또는 특징들을 공유하거나 공유하지 않을 수 있다. 예를 들어, 작은 코어들은 호환성 있는 코어 상에 프로세스들을 스케줄링하기 위해 운영 체제에 의한 추가적인 결정을 필요로 하는 축소 명령어 세트 또는 다른 차이들을 포함할 수 있다. 하나의 전통적인 예는 중앙 처리 유닛(CPU) 및 그래픽 처리 유닛(GPU)을 포함하는 시스템이다.

[0004] 기존 및 현재의 솔루션들은 크고 작은 코어들의 존재, 이들 각각의 성능 및 전력 특성들, 및 운영 체제가 특정 스레드를 어느 코어(들) 상에 스케줄링할지를 결정하기 위해 시스템 내의 어떤 설비(예로서, CPU 성능 카운터, 캐시 미스/히트 카운터, 버스 활동 카운터 등)를 모니터링할 수 있는지에 대해 운영 체제에게 "알려주기(enlighten)" 위해 운영 체제의 커널을 변경하는 것에 의존한다. 이러한 접근 방법은 1) 모든 지원되는 운영 체제들에 대한 커널 변경을 필요로 하고, 2) (예를 들어, N개의 상이한 구현을 지원하는) 잠재적으로 상이한 아키텍처들에 걸치는 큰/작은 코어들의 차이들을 이해할 것을 변경된 커널에 요구하며, 3) 운영 체제 커널의 출시(release) 스케줄과 기본 컴퓨터 아키텍처를 밀접하게 결합시키는 여러 가지 단점을 갖는다. 게다가, 컴퓨터 아키텍처에 대한 변경들은 커널이 새로운 코어들을 상업적으로 지원할 수 있으려면 다음 스케줄의 운영 체제 출시(즉, 잠재적으로 수년 이상)를 기다려야 하는 것을 필요로 한다(또는 그 반대도 마찬가지이다).

발명의 내용

[0005] 본 명세서에서는 에너지 효율을 향상시키거나 다른 처리 목표들을 충족시키기 위하여 운영 체제의 인지 또는 관여 없이 크고 작은 코어들에 걸쳐 운영 체제 스레드 스케줄링을 자율적으로 제어하기 위한 하이퍼바이저를 제공하는 이종 처리 시스템이 설명된다. 시스템은 가상화된 컴퓨팅 코어들의 유한 세트를 운영 체제에 제공하며, 시스템은 실행할 스레드들을 유한 세트에 스케줄링한다. 이어서, 표면 아래에서, 하이퍼바이저는 에너지 사용 또는 다른 처리 요구들을 관리하기 위해 각각의 스레드를 실행할 크거나 작은 코어(들)의 물리적 할당 및 선택을 지능적으로 제어한다. 소프트웨어 하이퍼바이저를 이용하여 기본적인 크고 작은 컴퓨터 아키텍처를 추상화함으로써, 코어들 간의 성능 및 전력 운영 차이들이 운영 체제에 대해 불투명하게 유지된다. 고유의 간접 지정(indirection)은 또한 운영 체제 출시 스케줄로부터 새로운 능력들을 갖는 하드웨어의 출시를 분리한다. 하드웨어 판매자는 갱신된 하이퍼바이저를 출시하고, 새로운 하드웨어로 하여금 판매자가 선택하는 임의의 운영 체제 버전과 함께 동작하는 것을 가능하게 할 수 있다.

[0006] 본 요약은 상세한 설명에서 더 후술하는 개념들의 발췌를 간단한 형태로 소개하기 위해 제공된다. 본 요약은 청구 발명 대상의 중요한 특징들 또는 본질적인 특징들을 식별하는 것을 의도하지 않으며, 청구 발명 대상의 범위를 한정하는 데 사용되는 것도 의도하지 않는다.

도면의 간단한 설명

[0007] 도 1은 일 실시예에서, 이종 처리 시스템의 컴포넌트들을 도시하는 블록도이다.

도 2는 일 실시예에서, 코어들과 운영 체제 사이의 하이퍼바이저를 이용하여 이종 처리 코어들을 갖는 컴퓨팅 장치를 초기화하기 위한 이종 처리 시스템의 처리를 도시하는 흐름도이다.

도 3은 일 실시예에서, 이종 처리 코어들을 관리하는 하이퍼바이저를 통해 하나 이상의 운영 체제 스레드를 스케줄링하기 위한 이종 처리 시스템의 처리를 도시하는 흐름도이다.

도 4는 일 실시예에서, 이종 처리 시스템의 운영 환경을 도시하는 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0008] 본 명세서에서는 에너지 효율을 향상시키거나 다른 처리 목표들을 충족시키기 위하여 운영 체제의 인지 또는 관여 없이 크고 작은 코어들에 걸쳐 운영 체제 스레드 스케줄링을 자율적으로 제어하기 위한 하이퍼바이저를 제공하는 이종 처리 시스템이 설명된다. 시스템은 가상화된 컴퓨팅 코어들의 유한 세트를 운영 체제에 제공하고, 시스템은 실행할 스레드들을 유한 세트에 스케줄링한다. 이어서, 표면 아래에서, 하이퍼바이저는 에너지 사용 또는 다른 처리 요구들을 관리하기 위해 각각의 스레드를 실행할 크거나 작은 코어(들)의 물리적 할당 및 선택을 지능적으로 제어한다. 소프트웨어 하이퍼바이저를 이용하여 기본적인 크고 작은 컴퓨터 아키텍처를 추상화함으로써, 코어들 간의 성능 및 전력 운영 차이들이 운영 체제에 대해 불투명하게 유지된다. 고유의 간접 지정(indirection)은 또한 운영 체제 출시 스케줄로부터 새로운 능력들을 갖는 하드웨어의 출시를 분리한다. 하드웨어 판매자는 갱신된 하이퍼바이저를 출시하고, 새로운 하드웨어로 하여금 판매자가 선택하는 임의의 운영 체제 버전과 함께 동작하는 것을 가능하게 할 수 있다.

[0009] 하이퍼바이저 구현은 기본 컴퓨터 아키텍처와 밀접하게 결합되며, 이용 가능한 시스템 피드백(예를 들어, CPU 사용, 버스/캐시 활동 등)을 이용하여, 요청된 작업 부하들에 대해 적절한 코어들을 자율적으로 할당한다. 이러한 접근 방법은 기본 컴퓨터 아키텍처로 하여금 소프트웨어 하이퍼바이저와 협력하여 자주 변하고 이러한 진화를 위의 운영 체제(들)로부터 분리하는 것을 가능하게 한다. 이종 처리 시스템은 운영 체제 커널 자체의 변경 없이 간단한 대략적인 전력 관리를 제공한다. 따라서, 이종 처리 시스템은 더 빠른 하드웨어 혁신을 가능하게 하고, 기존의 데이터 센터 및 다른 설비들로 하여금 이용 가능한 이종 처리 하드웨어로부터 현재 이익을 얻을 수 있게 해준다.

[0010] 이종 컴퓨팅은 시스템에서 이용 가능한 상이한 타입의 컴퓨팅 코어들(예를 들어, CPU, GPU, 가속기 등)에 기초하여 작업 부하들의 실행을 최적화하는 목표를 갖는 산업 내의 새로운 분야이다. 최적화는 성능, 전력, 지연 또는 다른 목표들을 위한 것일 수 있다. 이종 처리 시스템은 이러한 더 일반적인 사례들에 적용 가능하지만, 동일한 기능적 등가성, 그러나 상이한 성능/전력 운영 특성들을 갖는 코어들을 구비한 시스템들도 목표로 할 수 있다. 통상적으로, 이러한 시스템들은 하나 이상의 큰 코어 및 하나 이상의 작은 코어를 갖는다. 통상적으로, 큰 코어들은 깊은 파이프라인들, 무질서 실행, 큰 캐시들, 높은 클럭 속도들을 가지며, 더 높은 누설 프로세스들(예를 들어, 40G)을 이용하여 제조된다. 통상적으로, 작은 코어들은 더 짧은 파이프라인들, 더 작은 캐시들, 더 낮은 클럭 속도들, 다양한 전력 레벨들을 가지며, 낮은 누설 프로세스들(예로서, 40LP)을 이용하여 제조된다.

[0011] 일부 실시예들에서, 크고 작은 코어들은 아키텍처 등가성, 마이크로 아키텍처 등가성, 글로벌 인터럽트 제어기, 일관성 및 가상화를 가질 수 있다. 아키텍처 등가성은 동일한 명령어 세트 아키텍처(ISA), 단일 명령어 다중 데이터(SIMD), 부동 소수점(FP), 코프로세서 가용성 및 ISA 확장성을 포함할 수 있다. 마이크로 아키텍처 등가성은 성능 차이, 그러나 동일한 구성 가능 특징들(예를 들어, 캐시 라인 길이)을 포함할 수 있다. 글로벌 인터럽트 제어기는 인터럽트들을 관리하고, 처리하고, 모든 코어들로 전달하기 위한 능력을 제공한다. 일관성은 모든 코어들이 필요에 따라 전달과 더불어 다른 코어들로부터의 데이터에 액세스(캐시)할 수 있다는 것을 의미한다. 가상화는 코어들로부터/코어들로 작업 부하들을 전환/이동시키기 위한 것이다.

[0012] 일부 실시예들에서, 이종 처리 시스템은 코어들의 사소한 차이들을 처리하는 것이 가능할 수 있다. 예를 들어, (현재 4개의 반복 SSE1, SSE2, SSE3 및 SSE4에 존재하는) 스트리밍 단일 명령어 다중 데이터(SIMD) 확장성(SSE)을 지원하지 않는 작은 코어가 다른 인텔 x86 기반 소프트웨어 코드를 여전히 처리할 수 있다. 하이퍼바이저는 명령어 스트림 내에서 지원되지 않는 명령어들을 검출하고, 그러한 스트림들을 할당할 적절한 코어를 깨울 수 있다. 다른 명령어 스트림들은 임의의 코어 상에서 충실하게 동작할 수 있다. 소수의 지원되지 않는 명령어들만이 사용되는 일부 예들에서, 하이퍼바이저는 이용 가능한 명령어 세트 상의 지원되지 않는 명령어들을 에뮬레이션하기 위한 소정 레벨의 에뮬레이션을 포함할 수 있다. 예를 들어, 벡터 수학과 같은 연산들이 종종 중단되고, 표준 수학 명령어들을 이용하여 더 낮은 효율로 구현될 수 있다.

[0013] 소프트웨어 하이퍼바이저는 그 자신을 운영 체제(OS) 초기화 전에 장치 부트 프로세스 동안 설치한다. 지정된 하드웨어 구성(즉, 메모리 구성, 가상화 설비들의 초기화 등)을 완료한 후, 하이퍼바이저는 정책을 통해 컴퓨팅 장치 내에 설치된 크고 작은 처리 코어들을 구성한다. 예를 들어, 장치가 이동 전화인 경우, 정책은 하이퍼바이저가 이용 가능한 최소 성능으로 운영 체제를 시동할 것을 지시하여 배터리 수명을 최적화할 수 있으며, 이어서 하이퍼바이저는 하나 이상의 작은 코어에 대해 운영 체제 스레드들을 스케줄링할 것이다. 대안으로서, 장치가 데이터 센터 블레이드인 경우, 정책은 하이퍼바이저가 이용 가능한 최대 성능으로 운영 체제를 시동할 것을 지시하여 에너지 효율을 희생시킬 수 있으며, 이어서 하이퍼바이저는 이용 가능한 큰 코어들에 대해 - 또한 아마도 이용 가능한 열 예산에 따라서는 작은 코어들에 대해서도 - 운영 체제 스레드들을 스케줄링할 것이다. 초기화를 완료한 후, 소프트웨어 하이퍼바이저는 운영 체제 부트 관리자를 로딩하며, 이 관리자는 운영 체제를 로딩한다.

[0014] 실행 시간 동안, 이종 처리 시스템은 가상화된 코어 세트를 운영 체제에 제공한다. 코어들의 운영 특성들 및 이들 간의 차이들은 운영 체제에 대해 불투명하며, 정의된 운영 정책에 기초하여 소프트웨어 하이퍼바이저에 의해 비밀리에 관리된다. 운영 정책은 시스템 초기화 동안 설정되거나 실행 시간 동안 동적으로 설정될 수 있다.

[0015] 하이퍼바이저는 이용 가능한 시스템 설비들(예로서, CPU 성능 카운터, 캐시 미스/히트 카운터, 버스 활동 카운터 등)과 협력하여 운영 정책을 이용하여 운영 체제 스레드들을 어느 코어들에 대해 스케줄링할지를 결정한다. 하이퍼바이저는 이러한 정보를 이용하여, CPU 코어 사용, 시간 경과에 따른 경향, 정보의 위치 및 입출력(I/O) 패턴을 이해할 것이다. 이러한 정보로부터, 하이퍼바이저는 적절한 경우에 크고 작은 코어들에 걸쳐 운영 체제 스레드들을 동적으로, 추론적으로 이동시킬 수 있다. 게다가, 하이퍼바이저는 시스템 구현에 따라 운영 체제를 대신하여 동적 주파수 및 전압 스케일링(DFVS)을 제어할 수도 있다.

[0016] 하이퍼바이저가 제어할 수 있는 이용 가능한 운영 정책들의 샘플링: 최소 전력(MiPo), 최대 성능(MaPe), MiPoD(Minimal Power, Performance on Demand) 및 MaPel(Maximum Performance, Power Down on Idle)이 존재한다. 이들 각각은 아래의 단락들에서 설명된다. 그러나, 임의의 특정 구현에 의해 선택되는 바와 같이 추가적인 더 진보된 운영 정책들이 구현될 수 있다.

[0017] 최소 전력(MiPo)은 코어들의 최소 세트에 대해 스레드들을 스케줄링한다. 이것은 통상적으로 하이퍼바이저가 작은 코어들에 대해 스레드들을 스케줄링하고 필요에 따라 DVFS를 이용하여 코어에 대한 전력 및 성능 운영 포인트를 제어한다는 것을 의미할 것이다. 필요에 따라 추가적인 작은 코어들이 급전되고 스케줄링될 수 있다.

[0018] 최대 성능(MaPe)은 코어들의 최대 세트에 대해 스레드들을 스케줄링한다. 이것은 통상적으로 하이퍼바이저가 모든 이용 가능한 코어들에 대해 - 큰 코어들로부터 시작하여 - 스레드들을 스케줄링하고 필요에 따라 DVFS를 이용하여 코어에 대한 전력 및 성능 운영 포인트를 제어한다는 것을 의미할 것이다. 이용 가능한 열 예산에 의해 허용되는 정도로, 작은 코어들도 급전되고 스케줄링된다.

[0019] MiPoD(Minimal Power, Performance on Demand)는 통상적으로 가장 낮은 이용 가능한 전력 상태에서 (예를 들어, 하나 이상의 작은 코어 상에서) 동작하지만, 작업 부하들이 요구할 때 성능을 증대시킨다. 이것은 통상적으로 "터보" 또는 "증대" 동작 모드로서 지칭되며, 큰 코어들을 동적으로 할당하고 그들에 대해 스케줄링함으로써 가능해진다. 작업 부하가 완료되면, 시스템은 최소 전력 상태로 (예를 들어, 작은 코어 상으로) 복귀한다.

[0020] MaPel(Maximum Performance, Power Down on Idle)은 통상적으로 최대 이용 가능 성능 상태에서 (예를 들어, 하나 이상의 큰 코어 상에서) 동작하지만, 유휴 임계치에 도달하면 더 낮은 전력 상태에 들어간다. 이 경우에 유휴 임계치는 통상적인 0에 가까운 CPU 사용이 아니라, 정책에 의해 정의되는 바와 같은 소정의 DMIPS(Dhrystone Million Instructions per Second) 또는 CPU 사용 백분율로 임의로 정의될 수 있다. 유휴 상태에 들어갈 때, 하이퍼바이저는 작은 코어들을 동적으로 할당하고 그들에 대해 스케줄링하며, 사용되지 않는 큰 코어들을 대기/파킹 상태가 되게 한다. 정책 및/또는 장래의 작업 부하들은 시스템이 언제 최대 이용 가능 성능 상태로 (예를 들어, 큰 코어들 상으로) 복귀할지를 결정한다.

[0021] 도 1은 일 실시예에서, 이종 처리 시스템의 컴포넌트들을 도시하는 블록도이다. 시스템(100)은 하나 이상의 중앙 처리 유닛(100), 운영 체제 인터페이스 컴포넌트(120), 가상 코어 관리 컴포넌트(130), 정책 엔진 컴포넌트(140), 정책 데이터 저장소(150), 스케줄링 컴포넌트(160), 능력 관리 컴포넌트(170) 및 하드웨어 인터페이스 컴포넌트(180)를 포함한다. 이들 컴포넌트 각각은 본 명세서에서 더 상세히 설명된다. 아래의 컴포넌트들은 컴퓨팅 장치의 운영 체제와 하드웨어 자원들 사이에 위치하는 소프트웨어 하이퍼바이저 내에 구현될 수 있다.

[0022] 하나 이상의 중앙 처리 유닛(110)은 이종 처리 능력들 및 전력 프로필들을 갖는 하나 이상의 처리 코어를 포함한다. 통상적으로, 각각의 CPU 콤플렉스는 단일 실리콘 다이 상에 위치하며, CPU 콤플렉스의 각각의 코어는 단일 다이를 공유한다. 하드웨어는 다양한 타입의 장치들에 대한 다양한 패키지들 내에 구현될 수 있다. 예를 들어, 더 새로운 이동 장치들 및 심지어 일부 최신 데스크탑 프로세서들은 CPU와 GPU 사이의 효율적인 통신 및 더 낮은 전력 사용을 위해 동일 칩 상에 CPU 및 GPU를 포함한다. 각각의 CPU 콤플렉스는 하나 이상의 크고 작은 코어를 포함할 수 있다. 대안으로서 또는 추가로, 하나의 CPU 콤플렉스가 모든 큰 코어를 포함할 수 있고, 다른 CPU 콤플렉스가 모든 작은 코어들을 포함할 수 있다. 여기서 사용되는 바와 같은 CPU 콤플렉스들은 GPU들 및 소프트웨어 명령어들을 실행할 수 있는 다른 하드웨어에 적용된다.

[0023] 운영 체제 인터페이스 컴포넌트(120)는 하드웨어 자원들로 전달할 명령어들을 수신하고, 하드웨어 자원들로부터 출력을 수신하기 위해 하이퍼바이저와 운영 체제 사이에서 통신한다. 운영 체제는 스레드들을 스케줄링하고, 명령어 스트림에 대한 포인터(예를 들어, 프로그램 카운터(PC))를 제공하고, 하드웨어에 명령어들을 전달하는 메모리 영역들에 기록하는 것 등을 행할 수 있다. 운영 체제는 통상적으로 컴퓨팅 장치 상의 하드웨어와 직접 상호작용한다. 그러나, 하이퍼바이저는 다양한 목적들을 위해 운영 체제와 하드웨어 사이에 간접 지정 계층을 삽입한다. 종종, 하이퍼바이저들은 다수의 운영 체제가 동일 하드웨어 상에서 동시에 실행될 수 있도록 가상화를 제공하는 데 사용된다. 하이퍼바이저는 컴퓨팅 장치에 설치된 실제 하드웨어와 다른 가상 하드웨어를 운영 체제에 제공하는 데에도 사용될 수 있다. 이종 처리 시스템(100)의 경우에, 이것은 크고 작은 코어들이 운영 체제에게 동일하게 보이게 하는 것을 포함할 수 있다. 시스템(100)은 장치 내에 실제로 존재하는 것과 다른 수의 코어를 운영 체제에 제공할 수도 있다.

[0024] 가상 코어 관리 컴포넌트(130)는 하이퍼바이저가 운영 체제에 제공하는 하나 이상의 가상 코어를 관리한다. 가상 코어는 운영 체제에게 CPU 코어로 보이지만, 컴퓨팅 장치 내의 이용 가능한 물리 하드웨어와 특성들이 다를 수 있다. 예를 들어, 가상 코어들은 운영 체제로부터 처리 또는 전력 능력들의 차이들을 숨길 수 있으며, 따라서 이종의 크고 작은 코어들과 함께 동작하도록 설계되지 않은 소정의 운영 체제는 상기 운영 체제가 설계된 방식으로 동작할 수 있다. 그러한 경우에, 하이퍼바이저는 이종 컴퓨팅 환경을 향상시키는 데 필요한 임의의 특수화된 프로그래밍을 제공하며, 따라서 운영 체제는 변경될 필요가 없다.

[0025] 정책 엔진 컴포넌트(140)는 이용 가능한 하나 이상의 중앙 처리 유닛에 기초하여 운영 체제 스레드들을 스케줄링하고 가상 코어들을 운영 체제에 제공하기 위한 하나 이상의 정책을 관리한다. 정책 엔진 컴포넌트(140)는 특정 하이퍼바이저 구현에 고유한 하드코딩된 정책들을 포함할 수 있거나, 특정 설치 목표들에 맞도록 변경될 수 있는 관리자 구성 가능 정책들을 포함할 수 있다. 정책들은 어느 코어들을 먼저 스케줄링할 것인지, 전력 사용과 처리 목표들 간의 균형, 절전을 위해 어떻게 코어들을 끄고 깨울 것인지, 가상 코어들을 어떻게 운영 체제에 제공할 것인지 등을 결정할 수 있다.

[0026] 정책 데이터 저장소(150)는 부팅 및 실행 시간에 하이퍼바이저가 액세스할 수 있는 저장 설비에 하나 이상의 정책을 저장한다. 정책 데이터 저장소(150)는 시스템(100)의 실행 세션들에 걸쳐 데이터를 유지하기 위한 하나 이상의 파일, 파일 시스템, 하드 드라이브, 데이터베이스 또는 기타 저장 설비들을 포함할 수 있다. 일부 실시예들에서, 관리자는 구성 단계를 통해 시스템(100)으로 하여금 하이퍼바이저에 의한 사용을 위해 초기 정책 세트를 저장하게 하는 셋업 단계를 수행한다.

[0027] 스케줄링 컴포넌트(160)는 운영 체제로부터 스레드들로서 수신된 하나 이상의 명령어 스트림을 컴퓨팅 장치에 설치된 중앙 처리 유닛들 중 하나 이상에 대해 스케줄링한다. 스케줄링 컴포넌트는 운영 체제가 스레드를 스케줄링하도록 요청하는 가상 코어를 식별하는 가상 코어 식별자를 운영 체제로부터 수신한다. 스케줄링 컴포넌트(160)는 스케줄 요청을 검사하고, 실행할 스레드를 스케줄링할 물리 코어를 결정한다. 예를 들어, 컴포넌트(160)는 스레드에 대해 전력 또는 처리가 더 관련 있는지를 결정하고, 이에 응답하여 적절한 작은 또는 큰 코어에 대해 스케줄링할 수 있다. 일부 예들에서, 컴포넌트(160)는 소정 코어들에 대한 스레드들의 스케줄링을 회피하여, 그러한 코어들로 하여금 절전을 위해 파워 다운되게 할 수 있다.

[0028] 능력 관리 컴포넌트(170)는 옵션으로서 크고 작은 처리 코어들 사이의 하나 이상의 차이를 관리한다. 일부 예들에서, 시스템(100)은 크고 작은 코어들이 동일한 능력들을 공유하는 처리 유닛들 상에서만 동작할 수 있으며, 능력 관리 컴포넌트(170)는 필요하지 않다. 다른 예들에서, 시스템(100)은 이용 가능한 처리 코어들 사이의 사소하거나 중요한 차이들을 처리한다. 예를 들어, 시스템(100)은 일부 코어들에 의해 지원되지 않는 명령어들을 관찰하고, 이러한 명령어들을 지원하는 코어들 상에 대응하는 스레드들을 스케줄링할 수 있다. 더 정교한 구현들에서, 컴포넌트(170)는 작은 코어들 상에 큰 코어의 능력들을 가상화 또는 에뮬레이션하여(또는 그 반대도 가

능), 전력 또는 다른 프로필 목표를 충족시킬 수 있다.

[0029] 하드웨어 인터페이스 컴포넌트(180)는 이용 가능한 물리 코어들 상에서 실행할 소프트웨어 명령어들을 스케줄링하기 위해 하이퍼바이저와 중앙 처리 유닛들 사이에서 통신한다. 하드웨어 인터페이스 컴포넌트(180)는 다른 컴포넌트들로부터 그리고 특히 하이퍼바이저에 의해 관리되는 게스트 운영 체제(들)로부터 숨겨진 실제 하드웨어에 액세스하기 위한 실제 메모리 어드레스들 또는 다른 설비를 포함할 수 있다.

[0030] 이종 처리 시스템을 구현하는 컴퓨팅 장치는 중앙 처리 유닛, 메모리, 입력 장치들(예를 들어, 키보드 및 포인팅 장치들), 출력 장치들(예를 들어, 디스플레이 장치들) 및 저장 장치들(예를 들어, 디스크 드라이브들 또는 다른 비휘발성 저장 매체들)을 포함할 수 있다. 메모리 및 저장 장치들은 시스템을 구현하거나 가능화하는 컴퓨터 실행 가능 명령어들(예로서, 소프트웨어)로 인코딩될 수 있는 컴퓨터 판독 가능 저장 매체들이다. 게다가, 데이터 구조들 및 메시지 구조들은 저장되거나, 통신 링크 상의 신호와 같은 데이터 전송 매체를 통해 전송될 수 있다. 인터넷, 근거리 네트워크, 광역 네트워크, 점대점 다이얼 업 접속, 셀폰 네트워크 등과 같은 다양한 통신 링크들이 사용될 수 있다.

[0031] 시스템의 실시예들은 개인용 컴퓨터, 서버 컴퓨터, 핸드헬드 또는 랩탑 장치, 멀티프로세서 시스템, 마이크로프로세서 기반 시스템, 프로그래밍 가능한 소비자 전자 장치, 디지털 카메라, 네트워크 PC, 미니 컴퓨터, 메인 프레임 컴퓨터, 임의의 상기 시스템들 또는 장치들을 포함하는 분산 컴퓨팅 환경, 셋톱 박스, 시스템 온 칩(SOC) 등을 포함하는 다양한 운영 환경들에서 구현될 수 있다. 컴퓨터 시스템들은 셀 폰, 개인용 휴대 단말기, 스마트폰, 개인용 컴퓨터, 프로그래밍 가능한 소비자 전자 장치, 디지털 카메라 등일 수 있다.

[0032] 시스템은 하나 이상의 컴퓨터 또는 다른 장치에 의해 실행되는 프로그램 모듈들과 같은 컴퓨터 실행 가능 명령어들과 일반적으로 관련하여 설명될 수 있다. 일반적으로, 프로그램 모듈들은 특정 작업들을 수행하거나 특정 추상 데이터 타입들을 구현하는 루틴, 프로그램, 객체, 컴포넌트, 데이터 구조 등을 포함한다. 통상적으로, 프로그램 모듈들의 기능은 다양한 실시예들에서 필요에 따라 결합되거나 분산될 수 있다.

[0033] 도 2는 일 실시예에서, 코어들과 운영 체제 사이의 하이퍼바이저를 이용하여 이종 처리 코어들을 갖는 컴퓨팅 장치를 초기화하기 위한 이종 처리 시스템의 처리를 도시하는 흐름도이다.

[0034] 블록 210에서 시작하여, 시스템은 컴퓨팅 장치를 초기화하기 위한 시동 요청을 수신한다. 예를 들어, 기본 입출력 시스템(BIOS), 확장형 펌웨어 인터페이스(EFI), 부트 로더 또는 다른 초기 장치 소프트웨어가 이종 처리 시스템을 구현하는 하이퍼바이저를 로딩 및 호출할 수 있다. 일부 예들에서, 관리자는 컴퓨팅 장치 상에 하이퍼바이저를 설치하기 위한 설치 단계를 이전에 수행했을 것이지만, 시스템은 네트워크 부트, 및 컴퓨팅 장치들에 대해 일반적으로 제공되는 다른 비설치 시나리오들도 지원할 수 있다.

[0035] 블록 220에서 계속하여, 시스템은 컴퓨팅 장치의 둘 이상의 물리적 처리 코어를 열거한다. 일부 실시예들에서, 적어도 2개의 코어가 상이한 성능 및 전력 사용 특성들을 제공한다. 그러나, 시스템은 비대칭이 존재하지 않는 경우에도 사용될 수 있다. 예를 들어, 전력 관리를 위해 소프트웨어 하이퍼바이저를 사용하는 것은 다이 상에 N개의 물리 CPU를 갖지만, K개만이 주변 온도, 폼 팩터 인클로저(form factor enclosure), 이용 가능 전력의 비용 등과 같은 외관들(externalities)에 기초하여 동작할 수 있는 시나리오들에도 적용 가능할 수 있다. 부트 시에, 하이퍼바이저는 이러한 "정책" 정보를 이용하여, 가상화된 K개의 코어의 세트를 운영 체제에 보고할 수 있으며, 이것은 각각의 부트 사이클 시에 변할 수 있다. 하이퍼바이저는 대칭 코어들에 대한 이러한 시나리오에서 동일한 작업을 수행할 것이다. 시스템은 BIOS 또는 다른 기본 계층을 호출하여, 컴퓨팅 장치가 얼마나 많은 그리고 어떤 종류의 프로세서를 설치하였는지를 결정할 수 있고, CPUID 또는 다른 유사한 명령어를 실행하여, 프로세서들의 처리 능력들에 대한 정보를 결정할 수 있다. 일부 실시예들에서, 시스템은 하이퍼바이저 자체를 갱신할 필요 없이 하이퍼바이저에 새로운 처리 하드웨어에 대한 지원을 추가하기 위해 하이퍼바이저 제조자 또는 제삼자가 드라이버들 또는 다른 하이퍼바이저 확장들을 구현하고 추가할 수 있는 확장성 인터페이스를 포함할 수 있다.

[0036] 블록 230에서 계속하여, 시스템은 각각의 열거된 처리 코어의 능력들을 결정한다. 능력들은 각각의 코어에 의해 제공되는 하나 이상의 전력 프로필, 각각의 코어에 의해 지원되는 하나 이상의 명령어 세트, 각각의 코어의 성능 특성들 등을 포함할 수 있다. 시스템은 코어 자체의 정보 인터페이스들(전술한 CPUID 명령어 등) 또는 드라이버 또는 다른 확장에 의해 하이퍼바이저에 제공된 정보를 이용하여, 각각의 코어의 능력들을 결정할 수 있다. 시스템은 결정된 능력들을 이용하여, 각각의 코어와 호환되는 스레드들을 각각의 코어에 할당하고, 수신된 정책들 및 처리 목표들과 일치하는 방식으로 스케줄링을 수행한다.

[0037] 블록 240에서 계속하여, 시스템은 하이퍼바이저가 나열된 물리 코어들에 대한 액세스 및 스케줄링을 관리할 하나 이상의 운영 체제를 식별한다. 시스템은 컴퓨팅 장치의 하드 드라이브, 플래시 드라이브 또는 다른 저장 장치에 액세스하여, 하이퍼바이저가 초기화된 후에 어느 운영 체제를 호출할지를 결정할 수 있다. 하이퍼바이저는 다양한 운영 체제들에 대한 정보를 갖도록 설계될 수 있으며, 하이퍼바이저 자체의 갱신 없이 새로운 운영 체제들을 지원할 수 있는 확장성을 포함할 수 있다. 각각의 운영 체제 및 운영 체제 버전은 운영 체제로 하여금 가상화된 처리 자원들 상에서 정확하게 실행되게 하기 위해 하이퍼바이저가 처리하는 상이한 스케줄링 시맨틱들 또는 다른 뉘앙스들을 가질 수 있다. 일부 예들에서, 하이퍼바이저는 다수의 운영 체제가 열거된 물리적 처리 코어들을 공유하는 것을 허가하도록 요청될 수 있으며, 정책은 그러한 공유를 어떻게 처리할지를 지시할 수 있다.

[0038] 블록 250에서 계속하여, 시스템은 열거된 물리적 처리 코어들 상에 운영 체제 스레드들을 스케줄링하기 위한 하나 이상의 목표를 지정하는 하이퍼바이저 정책 정보에 액세스한다. 목표들은 성능 목표들, 전력 사용 목표들, 또는 운영 체제 스레드들을 어느 코어 또는 코어들 상에서 실행할지를 결정하기 위한 다른 지시들을 포함할 수 있다. 정책은 컴퓨팅 장치와 관련된 저장 장치에 저장되고, 하이퍼바이저 구현 내에 하드코딩되고, 기타 등등일 수 있다. 하이퍼바이저는 관리자들에 의해 제공되는 관리 인터페이스를 통해 정책에 대한 갱신들을 수신할 수 있다.

[0039] 블록 260에서 계속하여, 시스템은 식별된 운영 체제에 노출시킬 하나 이상의 가상 코어를 생성하며, 각각의 가상 코어는 물리적 처리 코어들 사이의 결정된 능력 차이들로부터 운영 체제를 격리시킨다. 예를 들어, 이종 처리 시스템은 둘 이상의 크고 작은 코어를 단일 타입의 균일한 가상 코어로서 운영 체제에 제공할 수 있다. 가상 코어 상에서 스레드를 실행하기 위한 운영 체제로부터의 스케줄링 요청의 수신시에, 시스템은 액세스된 하이퍼바이저 정책에 기초하여 작업을 위해 어느 물리 코어를 선택할지를 결정한다. 하이퍼바이저 정책은 예를 들어 더 낮은 전력을 요구하는 코어들을 이용하기 위해 적어도 일부 고전력 요구 코어들을 중단 없이 파워 다운시킬 수 있는 것이 목표일 때 하이퍼바이저가 물리 코어들의 수와 다른 수의 가상 코어들을 제공하도록 지정할 수 있다. 대안으로서, 시스템은 운영 체제가 인식하는 코어들을 여전히 파워 다운시키지만, 운영 체제가 코어들을 사용하기로 또는 더 적게 급전받는 코어들만으로는 충족될 수 없는 양의 코어들을 사용하기로 결정하는 경우에는 그러한 코어들을 깨울 수 있다.

[0040] 블록 270에서 계속하여, 시스템은 식별된 운영 체제를 호출하며, 생성된 가상 코어들을 운영 체제에 제공하는 한편, 열거된 물리적 처리 코어들로부터 식별된 운영 체제를 격리시킨다. 운영 체제의 호출은 운영 체제 로더를 호출하고, 운영 체제의 기초가 되는 통상의 BIOS 또는 다른 계층 대신에 하이퍼바이저를 제공하는 것을 포함할 수 있다. 운영 체제는 물리 하드웨어 상에서 직접 실행되고 있는 것처럼 동작하지만, 하이퍼바이저는 운영 체제와 물리 하드웨어 사이에 위치하여, 본 명세서에서 설명되는 바와 같은 운영 체제의 지식 없이 스케줄링 논리를 수행한다. 블록 270에서, 이러한 단계들이 종료된다.

[0041] 도 3은 일 실시예에서, 이종 처리 코어들을 관리하는 하이퍼바이저를 통해 하나 이상의 운영 체제 스레드를 스케줄링하기 위한 이종 처리 시스템의 처리를 도시하는 흐름도이다.

[0042] 블록 310에서 시작하여, 시스템은 하이퍼바이저에 의해 제공되는 식별된 가상 코어 상에서 스레드의 명령어들을 실행하기 위한 운영 체제로부터의 스레드 스케줄링 요청을 수신하며, 가상 코어는 컴퓨팅 장치가 액세스할 수 있는 둘 이상의 물리적 처리 코어 간의 하나 이상의 능력 차이로부터 운영 체제를 격리한다. 통상적으로 운영 체제는 각각의 검출된 처리 코어에 대한 유휴 루프를 포함하며, 그 코어에서 운영 체제는 그 코어 상에서 실행하기를 원하는 임의의 명령어들을 스케줄링하고 배치할 수 있다. 운영 체제는 특정 처리 코어 상에서 실행할 다수의 애플리케이션 스레드를 시간 슬라이싱(time slicing)할 수 있다. 운영 체제가 스레드를 실행하기 위해 선택하는 특정 가상 코어에 관계없이, 하이퍼바이저는 하나 이상의 하이퍼바이저 정책에 따라 스레드를 실행할 임의의 특정 물리적 처리 코어를 선택할 수 있다.

[0043] 블록 320에서 계속하여, 시스템은 수신된 스케줄링 요청의 처리 요구를 결정한다. 예를 들어, 시스템은 스케줄링된 스레드에 의해 사용되는 특정 명령어 세트(예를 들어, 하나 이상의 명령어 세트 확장, 코프로세서 또는 다른 능력이 요청되고 있는지), 스레드의 성능 요구들, 스레드가 더 낮은 전력 사용에서의 더 느린 실행에 적합한지, 추가적인 처리 자원들이 이용 가능할 때까지 스레드가 지연될 수 있는지 등을 결정할 수 있다. 시스템은 특정 운영 체제들에 대한 특정 지식 또는 정책을 통해 수신된 명령어들을 이용하여 특정 스레드의 처리 요구를 결정할 수 있다. 예를 들어, 시스템은 운영 체제의 내부 동작과 관련된 스레드들, 애플리케이션 스레드들 등을 식별하고, 이들 각각을 정책에 따라 처리할 수 있다.

[0044] 블록 330에서 계속하여, 시스템은 장치를 동작시키기 위한 하나 이상의 목표를 지정하는 스케줄링 정책에 액세스한다. 예를 들어, 정책은 전력 사용, 성능 또는 이 둘의 혼합의 최적화를 요청할 수 있다. 정책은 장치와 관련된 데이터 저장소에 저장되거나, 하이퍼바이저의 특정 구현들 내에 하드코딩될 수 있다. 예를 들어, 시스템은 고성능 작업을 수행하는 스레드가 운영 체제에 의해 스케줄링될 때까지 작은 처리 코어들을 선호하는 하이퍼바이저의 저전력 사용 버전을 제공할 수 있다. 그 포인트에서, 시스템은 큰 코어 상에 고성능 작업을 스케줄링하고, 이어서 작업이 완료된 후에 큰 코어에게 자라고 지시할 수 있다.

[0045] 블록 340에서 계속하여, 시스템은 수신된 스케줄링 요청과 관련된 스레드를 실행할 물리적 처리 코어를 선택하며, 이러한 선택은 액세스된 스케줄링 정책에 기초하여 행해진다. 시스템은 시스템이 스레드를 스케줄링할 수 있는 상이한 능력들 및 성능/전력 특성들의 다수의 이용 가능 코어를 가질 수 있다. 시스템의 코어 선택에 기초하여, 컴퓨팅 장치는 더 많거나 적은 전력을 사용할 것이며, 더 많거나 적은 시간 내에 스레드의 실행을 완료할 것이다. 작업 또는 스케줄링 정책은 시스템으로 하여금 컴퓨팅 장치의 성능, 전력 또는 다른 특성들을 관리하기 위한 하나 이상의 목표를 촉진하는 방식으로 선택을 행하는 것을 가능하게 한다. 이동 장치는 저전력 사용을 선호할 수 있는 반면, 고성능 서버는 고성능을 선호할 수 있다. 일부 예들에서, 정책은 하루의 시각(예를 들어, 피크 대 비피크(non-peak) 전기 비용) 또는 다른 사항들에 기초하여 다를 수 있으며, 따라서 정책은 시간 경과에 따라 또는 소정 조건들에 기초하여 변한다.

[0046] 블록 350에서 계속하여, 시스템은 옵션으로서 스레드와 선택된 물리적 처리 코어 사이의 임의의 능력 차이들을 처리한다. 예를 들어, 스레드가 선택된 코어 상에서 이용할 수 없는 명령어를 포함하는 경우, 시스템은 명령어를 에뮬레이션하거나, 명령어를 선택된 코어에 의해 지원되는 하나 이상의 등가 명령어로 대체할 수 있다. 능력 차이들을 관리하는 것은 상당한 양의 복잡성을 시스템에 더하며, 하이퍼바이저 구현자는 임의의 특정 구현이 (존재할 경우) 얼마나 많은 또는 적은 처리 코어들 사이의 능력 차이들을 지원할지를 결정할 수 있다.

[0047] 블록 360에서 계속하여, 시스템은 선택된 물리적 처리 코어 상에 실행할 스레드를 스케줄링한다. 시스템은 또한 임의의 출력을 처리하고, 출력을 운영 체제에 제공하여, 운영 체제가 스레드를 할당한 가상 코어로부터 출력이 나오는 것처럼 보이게 한다. 따라서, 운영 체제는 하이퍼바이저에 의해 관리되는 코어들의 타입 및 수를 모르는 상태로 유지되고, 운영 체제가 하이퍼바이저 및 이종 처리 코어를 갖지 않는 시스템 내에 있는 것처럼 가상 코어들의 세트를 사용한다. 블록 360 후에 이들 단계가 종료된다.

[0048] 도 4는 일 실시예에서, 이종 처리 시스템의 운영 환경을 도시하는 블록도이다. 게스트 운영 체제(410)는 본 명세서에서 설명되는 이종 처리 시스템을 구현하는 소프트웨어 하이퍼바이저(430)에 의해 제공되는 하나 이상의 가상 처리 코어(420)를 본다. 소프트웨어 하이퍼바이저(430)는 글로벌 인터럽트 제어기(450), 하나 이상의 큰 코어(460) 및 하나 이상의 작은 코어(470)를 포함하는 이종 처리 하드웨어(440)를 관리한다. 도시의 편의를 위해 (크고 작은) 2개의 코어 타입만이 도시되지만, 시스템은 임의 수의 상이한 코어들과 함께 동작할 수 있다는 점에 유의한다. 예를 들어, 일부 프로세서 패키지들은 전력 사용 및 성능을 점차 낮추는 여러 개의 처리 저장소를 포함할 수 있다. 소프트웨어 하이퍼바이저(430)는 이종 처리 하드웨어(440)를 관리하여, 하드웨어(440)를 효과적으로 사용하는 데 필요한 임의의 특별한 지식 또는 처리로부터 게스트 운영 체제(410)를 격리시킨다. 따라서, 소프트웨어 하이퍼바이저(430)는 게스트 운영 체제(410)와 같은 변경되지 않은 레거시 운영 체제로 하여금 더 새로운 이종 처리 하드웨어(440)를 이용하게 할 수 있다. 하이퍼바이저(430)는 하드웨어(440) 변경들을 따르도록 변경될 수 있는 반면, 운영 체제(410)는 (더욱더 양호한 전력/성능 특성들을 갖지만) 통상적으로 계속 동작한다.

[0049] 일부 실시예들에서, 이종 처리 시스템은 스레드가 이미 실행된 후에 스레드를 하나의 물리적 처리 코어로부터 다른 물리적 처리 코어로 이동시킨다. 일부 예들에서, 하이퍼바이저가 전력 소비를 줄이거나, 성능을 증가시키거나, 다른 정책 목표들을 실행하기로 결정할 때, 하나 이상의 스레드가 이미 실행되고 있을 수 있다. 코어들은 캐시 저장 장치 또는 다른 설비들을 공유할 수 있으며, 따라서 하이퍼바이저는 데이터에 대한 스레드의 액세스에 영향을 미치지 않고 스레드를 다른 코어로 이동시킬 수 있다. 따라서, 하이퍼바이저는 스레드의 실행을 인터럽트하고, 스레드의 명령어 스트림을 상이한 물리적 처리 코어로 이동시키고, 타겟 코어 상에서 실행을 재개할 수 있다.

[0050] 일부 실시예들에서, 이종 처리 시스템은 프로세서 전압 및 주파수 변경들을 이용하여, 상이한 코어를 선택하기 전에 전력을 줄이거나 성능을 임계치까지 증가시킨다. 예를 들어, 시스템은 큰 코어 상에서의 특정 스레드의 실행을 시작하고, 이어서 코어의 동작 전압을 줄임으로써 큰 코어의 전력 사용을 줄이고, 마지막으로 큰 코어의 작업을 작은 코어로 이동시킬 수 있다. 이것은 시스템으로 하여금 전력 사용을 점차 낮추어 열 엔벨로프를 관

리하거나 정책에 의해 지정된 다른 컴퓨팅 목표들을 충족시키게 할 수 있다.

[0051] 일부 실시예들에서, 이종 처리 시스템은 일부 처리 작업들이 클라우드 컴퓨팅 설비로 이동되게 할 수 있다. 시스템은 클라우드 컴퓨팅 설비를 작업들이 스케줄링될 수 있는 단지 또 하나의 처리 코어로서 제공할 수 있다. 적절한 작업들에 대해, 시스템은 컴퓨팅 장치로부터 작업들을 완전히 오프로딩하고, 나중에 작업의 출력을 게스트 운영 체제로 반환하는 것이 가능할 수 있다. 이것은 시스템이 컴퓨팅 장치 상에서 저전력 상태에 들어가거나, 작업을 피크 전기 비용의 데이터 센터로부터 더 적은 전기 비용의 데이터 센터로 이동시키는 것을 가능하게 할 수 있다.

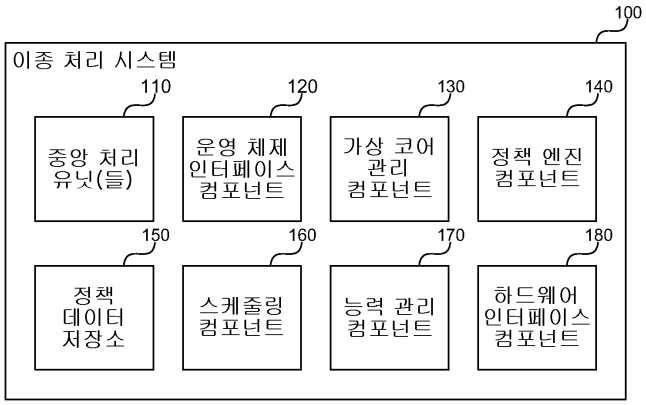
[0052] 일부 실시예들에서, 이종 처리 시스템은 경합 조건들을 처리하고, 소프트웨어 록킹 패러다임들을 이용하여 운영 체제 기대들을 관리한다. 많은 예에서, 운영 체제들은 특정 스레드들 사이의 상호 의존성, 또는 의존성의 결여에 기초하여 스레드들을 스케줄링한다. 소프트웨어는 운영 체제에 의해 제공되는 록(lock), 뮤텍스(mutex), 세마포어(semaphore) 또는 다른 동기화 프리미티브(primitive)를 이용하여, 소프트웨어 코드로 하여금 다수의 동시 실행 스레드의 환경에서 정확히 동작하는 것을 가능하게 할 수 있다. 이종 처리 시스템은 스레드 안전 및 다른 동기화에 대한 운영 체제의 보증들이 충족되는 것을 보장하며, 새로운 경합 조건 또는 다른 문제가 발생하지 않는 것을 보장하는 방식으로 추가적인 록들을 도입하거나 스레드 스케줄링을 결정할 수 있다.

[0053] 일부 실시예들에서, 이종 처리 시스템은 하드웨어 하이퍼바이저를 포함한다. 소프트웨어 하이퍼바이저가 본 명세서의 예들에서 사용되었지만, 당업자들은 컴퓨팅 작업들의 구현을 위한 하드웨어 또는 소프트웨어의 선택이 종종 성능 또는 다른 목표들을 충족시키기 위해 변화될 수 있는 구현 상세라는 것을 인식할 것이다. 따라서, 시스템은 하드웨어 하이퍼바이저를 사용하여 구현될 수 있으며, 일부 처리 유닛들은 처리 유닛 자체에 시스템을 포함하도록 제조될 수 있다.

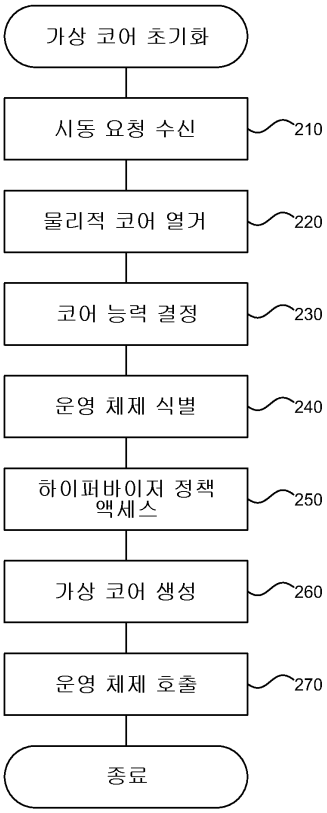
[0054] 이상으로부터, 본 명세서에서 이종 처리 시스템의 특정 실시예들이 예시의 목적으로 설명되었지만, 본 발명의 사상 및 범위로부터 벗어나지 않으면서 다양한 변경들이 이루어질 수 있다는 것을 알 것이다. 따라서, 본 발명은 첨부된 청구범위에 의해서만 한정된다.

도면

도면1



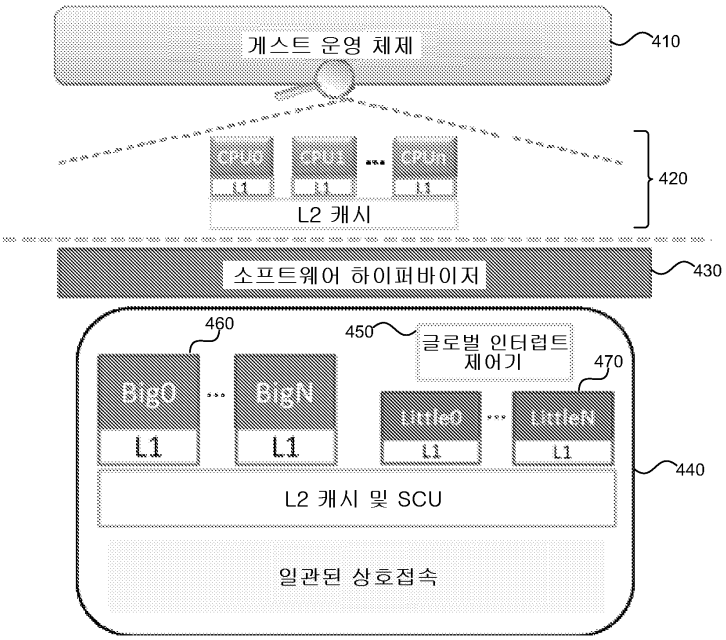
도면2


가상 코어 초기화
시동 요청 수신
210
물리적 코어 열거
220
코어 능력 결정
230
운영 체제 식별
240
하이퍼바이저 정책 액세스
250
가상 코어 생성
260
운영 체제 호출
270
종료

도면3


스레드 스케줄링
스레드 스케줄링 요청 수신
310
스레드 처리 요구 결정
320
스케줄링 정책 액세스
330
물리적 처리 코어 선택
340
코어 능력 차이 관리
350
스레드가 선택된 코어 상에 실행하도록 스케줄링
360
종료

도면4


게스트 운영 체제
410
CPU0
L1
CPU1
L1
CPUn
L1
L2 캐시
420
소프트웨어 하이퍼바이저
430
460
450
글로벌 인터럽트 제어기
470
Big0
L1
BigN
L1
Little0
L1
LittleN
L1
440
L2 캐시 및 SCU
일관된 상호접속