(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2010-0138954
(43) 공개일자 2010년12월31일

(51) Int. Cl.
G06F 9/50 (2006.01) *G06F 1/32* (2006.01)
G06F 9/46 (2006.01) *G06F 9/28* (2006.01)
(21) 출원번호 10-2010-7021186
(22) 출원일자(국제출원일자) 2009년02월16일
심사청구일자 없음
(85) 번역문제출일자 2010년09월20일
(86) 국제출원번호 PCT/US2009/034209
(87) 국제공개번호 WO 2009/120427
국제공개일자 2009년10월01일
(30) 우선권주장
12/057,716 2008년03월28일 미국(US)

(71) 출원인
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
마샬, 알렌
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 내
뎅, 이민
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 내
(뒷면에 계속)
(74) 대리인
양영준, 백만기

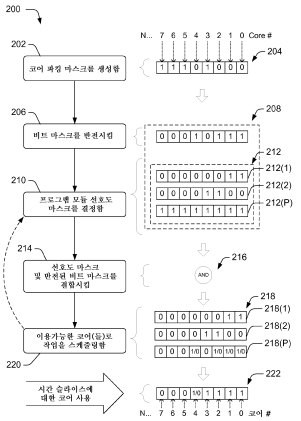
전체 청구항 수 : 총 20 항

(54) 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용

(57) 요 약

전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용을 제공하는 기술 및 장치가 개시되어 있다. 일부 양태들에서, 코어 활동을 확인하기 위하여 다중-코어 시스템이 모니터링된다. 코어 활동이 전력 절감 계획과 성능 계획을 균형을 이루게 하는 전원 정책과 비교될 수 있다. 다중-코어 시스템에 의한 전력 소모를 감소시키기 위해 상기 비교에 따라 코어들 중 하나 이상이 파킹될 수 있다. 부가의 양태들에서, 코어들을 동적으로 파킹 또는 파킹 해제하기 위해 미리 정해진 구간 동안 전력-인식 스케줄링이 수행될 수 있다. 추가적인 양태들이 코어 활동 및 전원 정책의 비교에 따라 파킹 해제된 코어들의 전원 상태를 조정하는 것을 포함한다.

대 표 도 - 도2



(72) 발명자

저지, 니콜라스 에스.

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 내

키샨, 아룬 유.

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 내

리츠, 앤드류 제이.

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 내

특허청구의 범위

청구항 1

복수의 코어들을 갖는 컴퓨팅 장치의 성능과 전력 절감을 균형을 이루게 하는 방법으로서,

어느 코어들이 작업을 처리하는 데 사용되는지를 정의하는 단계(202, 204);

상기 정의된 코어들 및 정의되지 않은 코어들 둘다에 대한 성능 및 전력 절감 계획을 시작하기 위해 전원 정책을 결정하는 단계(210, 214, 406); 및

상기 전원 정책에 기초하여 상기 정의되지 않은 코어들 중 적어도 하나를 파킹하는 단계(222)

를 포함하는 방법.

청구항 2

제1항에 있어서,

상기 전원 정책을 결정하는 단계는,

코어 파킹 마스크를 생성하는 단계(202);

스레드 프로세서 선호도(thread processor affinities)를 결정하는 단계(210);

이용가능한 프로세서 세트를 생성(220)하기 위하여 상기 코어 파킹 마스크와 스레드 프로세서 선호도 마스크들을 결합(214)함으로써 상기 코어들에 대한 성능과 전력 절감 중 적어도 일부분을 제공하는 단계; 및

상기 이용가능한 프로세서 세트 및 작업을 능동적으로 처리하는 코어들에 기초하여 어느 코어들이 파킹되거나 파킹 해제되는지를 산출하는 단계(222)

를 더 포함하는 방법.

청구항 3

제2항에 있어서,

상기 전원 정책을 결정하는 단계는 반복 프로세스(210, 214, 220)로서 동적으로 행해지는 방법.

청구항 4

제1항에 있어서,

상기 전원 정책에 기초하여 상기 정의된 코어들 중 적어도 하나를 스케일링하는 단계(512)를 더 포함하는 방법.

청구항 5

제4항에 있어서,

상기 정의된 코어들 중 적어도 하나를 스케일링하는 단계는 전력 절감을 증가시키기 위해 코어의 코어 활용도 및 전원 상태(p-상태) 중 적어도 하나를 조정하는 단계(512)를 포함하는 방법.

청구항 6

제1항에 있어서,

상기 정의되지 않은 코어들 중 적어도 하나를 파킹하는 단계는,

파킹 해제된 코어 블럭이 파킹된 코어(316(2))를 포함하는지의 여부를 판정하는 단계; 및

상기 파킹된 코어를 갖는 파킹 해제된 코어 블럭이 확인된 경우, 적어도 하나의 파킹 해제된 코어를 상기 파킹 해제된 코어 블록(320)에 파킹하는 단계

를 포함하는 방법.

청구항 7

제1항에 있어서,

코어 및 시스템의 발견적 학습법(heuristics)(708);

프로세서 의존성 관계(710, 714, 718); 및

코어 파킹을 위한 코어 정책 파라미터들(706)

중 적어도 하나를 사용하여 상기 전원 정책을 수정하는 단계를 더 포함하는 방법.

청구항 8

다중-코어 시스템에서 코어 활동을 모니터링하는 단계(402);

다중-코어 시스템에 대한 전원 정책을 검색하는 단계(406) - 상기 전원 정책은 상기 다중-코어 시스템 내의 각각의 코어의 전력 절감과 처리 성능을 균형을 이루게 함 -; 및

상기 전원 정책에 기초하여 상기 코어 활동에 응답하여 적어도 하나의 코어를 파킹하는 단계(412)

를 포함하는 방법.

청구항 9

제8항에 있어서,

상기 코어를 파킹하는 단계는 상기 코어 전원 상태를 최소 전력 프로세서 아이들(idle) 전원 상태(C-상태)로 감소시키는 단계(524)를 포함하는 방법.

청구항 10

제8항에 있어서,

상기 다중-코어 시스템에서 적어도 하나의 파킹 해제된 코어의 전원 상태를 조정하는 단계(520)를 더 포함하는 방법.

청구항 11

제8항에 있어서,

상기 코어 활동에 응답하여 적어도 하나의 코어를 파킹하는 단계는 미리 정해진 빈도수로 동적으로 시작되는(502, 524) 방법.

청구항 12

제8항에 있어서,

다중-코어 시스템에서 코어 활동을 모니터링하는 단계는 미리 정해진 기간 동안 행해지고(402), 상기 코어 활동에 응답하여 적어도 하나의 코어를 파킹하는 단계는 상기 모니터링된 코어 활동에 의해 검출된 전력 절감 기회에 응답하여 동적으로 행해지는(210, 214, 220) 방법.

청구항 13

제8항에 있어서,

상기 코어 활동에 응답하여 적어도 하나의 코어를 파킹하는 단계는 프로그램 스케줄과 연관된 선호도 마스크에 의해 허용된 바대로 작업이 파킹 해제된 코어로 이동하는 단계(712, 714)를 포함하는 방법.

청구항 14

제8항에 있어서,

수정된 코어 활동을 생성하기 위해 상기 모니터링된 코어 활동을 수정하는 단계(508);

상기 전원 정책에 기초하여 상기 수정된 코어 활동에 응답하여 적어도 하나의 코어를 파킹 해제하는 단계(512); 및

적어도 하나의 스레드를 상기 적어도 하나의 파킹 해제된 코어로 재할당하는 단계(514)

를 더 포함하는 방법.

청구항 15

다중 논리 프로세서 시스템으로서,

복수의 프로세서들(104); 및

상기 복수의 프로세서들에 결합된 제어기(106)

를 포함하며,

상기 제어기는,

성능 스케줄을 구현하고;

전력 절감 정책을 구현하며;

상기 복수의 프로세서들 중 하나 이상을 파킹함으로써 상기 성능 스케줄 및 전력 절감 정책을 균형을 이루게 하는 시스템.

청구항 16

제15항에 있어서,

상기 제어기는 상기 전력 절감 정책을 구현하기 위해 시스템 메모리에 존재하는 커널 전원 관리자(114)로부터 명령어들을 수신하는 시스템.

청구항 17

제15항에 있어서,

상기 전력 절감 정책(702)은 코어 파킹 마스크(202)를 포함하고, 상기 코어 파킹 마스크는 프로그램 모듈 스케줄들과 결합되어 작업을 스케줄링할 결과 프로세서 세트(116, 220)를 결정하는 시스템.

청구항 18

제15항에 있어서,

상기 복수의 프로세서들은 복수의 코어들(300)을 갖는 적어도 하나의 코어 블럭을 포함하는 시스템.

청구항 19

제18항에 있어서,

상기 성능 스케줄 및 전력 절감 정책을 균형을 이루게 하는 것은 적어도 하나의 파킹된 코어(318, 714)를 가지고 있는 코어 블럭 내의 파킹 해제된 코어에 대한 코어 파킹 우선순위를 생성하는 것을 포함하는 시스템.

청구항 20

제15항에 있어서,

적어도 하나의 파킹 해제된 코어의 전원 상태를 조정하는 것(520)을 더 포함하는 시스템.

명 세 서

배 경 기 술

[0001] 컴퓨터 시스템의 전력 소모 관리는 배터리의 수명을 연장하고 전체적인 전력 소모를 줄이는 데 있어서 중요하며, 이는 재정적으로도 환경적으로도 도움이 될 수 있다. 심지어 비모바일 컴퓨터의 경우에도, 전력 요

건을 감소시키는 것이 중요한 글로벌 자원을 절감하고, 정전 동안과 같이, 베터리 백업 시스템에 의존할 때 작동을 연장시키는 데 유익하다.

[0002] 비록 컴퓨팅 시스템의 대부분의 구성요소들이 시스템 동작 중에 전력을 사용하지만, 프로세서는 시스템 전력의 불균형적인 몫을 사용한다. 소비자-기반 시스템을 포함하여 많은 컴퓨터 시스템들이 다중 프로세서 및/또는 다중-코어 프로세서를 포함한다. 다중 프로세서로 컴퓨터가 점점 더 많은 작업 레벨들을 병렬로 수행할 수 있지만, 부가의 프로세서들은 또한 전력 소모도 증가시킬 수 있다. 대부분의 최신 프로세서들은 매우 낮은 전력의 아이들 전원 상태(idle power state)를 특징으로 하며, 이는 다중-코어 시스템 상에서 코어 단위로 적용될 수 있고 운영 체제에 의해 제어될 수 있다. 또한, 시스템에 의한 전력 사용을 줄이기 위하여, 프로세서 주파수가 코어 단위로 또는 코어 그룹 단위로 스케일링될 수 있다.

발명의 내용

[0003] 이 요약은 이하에서 상세한 설명에 더 기술되는 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용의 개념을 간략화하여 소개하기 위해 제공된 것이다. 이 요약은 청구된 발명 대상의 필수적인 특징들을 확인하기 위한 것이 아니며, 청구된 발명 대상의 범위를 정하는 데 사용하기 위한 것도 아니다.

[0004] 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용을 제공하는 예시적인 기술 및 장치가 개시되어 있다. 하나 이상의 실시예들에 따르면, 코어 활동을 확인하기 위하여 다중-코어 시스템이 모니터링된다. 코어들에 대한 성능 및 전력 절감 계획을 시작하기 위해 전원 정책이 검색된다. 다중-코어 시스템의 코어들 중 하나 이상이 전원 정책 및 코어 활동에 기초하여 파킹된다(시스템-지정된 저전력 상태에 놓인다). 하나 이상의 코어들이 파킹될 때, 파킹 해제된 코어들이 나머지 시스템활동 전부를 처리하게 된다. 일부 실시예들에서, 전원 정책은 전력 절감 또는 시스템 성능에 영향을 미치는 추가적인 요인들을 포함하도록 수정될 수 있다. 적어도 하나의 다른 실시예에서, 다중-코어 시스템은, 하나 이상의 코어들을 파킹하는 것에 부가하여, 하나 이상의 파킹 해제된 코어들의 전원 상태를 동적으로 조정할 수 있다.

도면의 간단한 설명

[0005] 상세한 설명이 첨부 도면들을 참조하여 기술된다. 도면들서, 참조 번호의 가장 왼쪽의 숫자들은 그 참조 번호가 처음으로 나타나는 도면을 알려준다. 서로 다른 도면들에서의 동일한 참조 번호는 유사한 또는 동일한 항목들을 가리킨다.

도 1은 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용의 적어도 하나의 실시예를 구현하는 데 사용될 수 있는 예시적인 시스템을 나타낸 도면이다.

도 2는 프로세서들에게 작업을 할당하는 것을 가능하게 하도록 코어 파킹 마스크를 생성하고 스레드 스케줄러로 이 마스크를 구현하는 프로세스의 적어도 하나의 실시예의 흐름도이다.

도 3a 및 도 3b는 본 발명의 적어도 하나의 실시예에 따른 예시적인 코어 활용을 나타낸 도면이다. 보다 구체적으로는, 도 3b는 도 3a에 도시된 것과 같은 예시적인 시스템 코어 활용에 응답한 코어 활용의 예시적인 수정을 나타낸 것이다.

도 4는 스레드 스케줄링 및 프로세서들의 동적 사용을 평가하고 프로세서들의 수정된 구성을 결정하는 예시적인 프로세스의 적어도 하나의 실시예의 흐름도이다.

도 5는 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용을 제공하는 예시적인 프로세스의 적어도 하나의 실시예의 흐름도이다.

도 6은 도메인 아이들 어카운팅(domain idle accounting)을 더 포함하는, 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용을 제공하는 예시적인 프로세스의 적어도 하나의 실시예의 다른 흐름도이다.

도 7은 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용에 대한 추가적인 설정을 제공하는 적어도 하나의 실시예의 예시적인 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0006] 프로세서들은 전력을 소비하지 않는(0 와트를 소비하는) 아이들 상태를 포함해 저전력 아이들 전원 상태들(low power idle power states)을 가능하게 할 수 있다. 운영 체제는, 수행할 유용한 작업이 없을 때, 하나 이상의

프로세서들(또는 간단히 "코어들")을 아이들 전원 상태(즉, 프로세스 절전 모드)로 전환시킬 수 있다. 이러한 저전력 상태들에서 사용되는 시간을 최대로 하는 것은 시스템 에너지 효율을 증대시키고 및/또는 배터리 성능을 연장시킬 수 있다. 이러한 프로세서 아이들 전원 상태들에 부가하여, 프로세서들은 또한, 독자적으로 또는 프로세서 코어 전압의 동시적인 감소와 함께, 프로세서의 주파수를 스케일링하는 제어들도 제공할 수 있다. 이러한 제어들을 모두 합하여 프로세서 전원 관리(processor power management, PPM) 기능들이라고 말할 수 있다.

[0007] 프로세서들은 10억 EPS(executions per second)의 실행을 용이하게 할 수 있다. 프로세서들은, 이러한 높은 컴퓨터 명령어 실행 능력을 가지고 있는 반면, 단기간에 걸쳐 상당한 작업량 변동을 겪을 수 있다. 예를 들어, 타이피스트의 키스트로크들 사이의 지연과 같은 짧은 기간에 의해 PPM은 프로세서 전력을 순간적으로 감소시킬 수 있거나 심지어 짧은 절전 모드에 들어갈 수 있다. 비록 몇분의 1초의 전력은 논의의 여지가 있는 것처럼 보일 수도 있지만, 보다 오랜 기간 동안에 걸쳐 누적 전력 절감은 상당할 수 있다.

[0008] 따라서, 프로세서들이 높은 전원 상태를 뒷받침할 적절한 작업량을 가지고 있지 않을 때, PPM은 사용되지 않는 프로세서들을 저전력 상태, 즉 절전 모드("파킹된" 상태)로 전환시킴으로써 전력 수요를 감소시킬 수 있다. 파킹된 코어들은 최소한의 전력을 사용하거나 전력을 전혀 사용하지 않는 프로세서 아이들 전원 상태(ACPI C-상태)에 놓여질 수 있다. 시스템에서 수행될 활성 작업은 파킹 해제된 프로세서들에 대해 시간 멀티플렉싱될 것이다.

[0009] 따라서, 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용의 제공을 용이하게 하는 기술과 장치가 본 명세서에서 이하의 다양한 섹션들에 개시되어 있다.

[0010] 예시적인 환경

[0011] 도 1은 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용의 적어도 하나의 실시예를 구현하는 데 사용될 수 있는 예시적인 시스템(100)이다. 시스템(100)은 컴퓨팅 장치(102)를 포함하고 있다. 예를 들어, 컴퓨팅 장치는, 가능한 컴퓨팅 장치들 중에서도 특히, 모바일 컴퓨터(102(1)), 데스크톱 컴퓨터(102(2)), 및/또는 서버(102(N))일 수 있다. 아주 기본적인 구성에서, 컴퓨팅 장치(102)는 통상적으로 하나 이상의 프로세서들("프로세서들")(104)을 포함한다. 예를 들어, 프로세서들(104)은 병렬로 또는 직렬로 구성된 복수의 독립적인 프로세서들 및 다중-코어 처리 장치 중 적어도 하나가 단독으로 있는 것 또는 다양한 조합으로 있는 것일 수 있다. 멀티-코어 프로세서는 동일한 칩 또는 집적 회로에 포함된 2개 이상의 프로세서들("코어들")을 가질 수 있다. 특정의 요소와 관련하여 달리 특별히 언급되지 않는 한, "프로세서", "코어" 및 "논리 프로세서"라는 용어들은 본 명세서에서 서로 바꾸어 사용될 수 있다

[0012] 또한, 컴퓨팅 장치(102)는 시스템 메모리(106)를 포함하고 있다. 컴퓨팅 장치의 정확한 구성 및 유형에 따라, 시스템 메모리(106)는 휘발성(예컨대, RAM), 비휘발성(예컨대, ROM, 플래쉬 메모리 등), 또는 이 둘의 일부 조합일 수 있다. 시스템 메모리(106)는 통상적으로 운영 체제(108), 하나 이상의 프로그램 모듈(110)을 포함하고, 프로그램 데이터(112)를 포함할 수 있다.

[0013] 운영 체제(108)는 프로세서 전원 관리(PPM) 기능들의 사용을 지시하는 일을 맡고 있는 커널 전원 관리자(kernel power manager)(114)를 포함할 수 있다. 커널 전원 관리자(114)는 성능 상태(p-상태) 혹은 선형 스로틀 상태(t-상태)를 이용하여 프로세서들(104)의 성능(예를 들어, 속도)를 조정할 수 있다. 예를 들어, 커널 전원 관리자(114)는 가능한 경우 에너지를 절감하기 위해 프로세서들(104)의 전력 소모를 현재의 작업량과 균형을 이루게 할 수 있다. 부가하여 또는 대안으로서, 커널 전원 관리자(114)는 프로세서들(104)이 작업량 수요에 응답하여 최대 처리 용량을 제공하게 할 수 있다. 또한, 커널 전원 관리자(114)는 프로그램 모듈(110)의 스레드와 같이 실행할 준비가 된 활성 스레드가 없을 때, 프로세서들(104) 중 하나 이상을 저전력 절전 모드로 들어가게 할 수 있다.

[0014] 운영 체제(108)는 시스템(100) 내의 모든 이용가능한 프로세서들(104)에 걸쳐 작업의 단위들(스레드들)을 큐잉(queuing), 스케줄링(scheduling), 우선순위 지정(prioritizing), 디스패치(dispatching)하는 모듈들을 제공할 수 있으며, 이 모듈들은 모두 합하여 커널 스레드 스케줄러(kernel thread scheduler)(116)라고 하는 모듈들의 모음으로서 표현될 수 있다. 활성 스레드가 실행될 준비가 될 때, 커널 스케줄러(116)는, 하나 이상의 모듈들을 통해, 처리를 위해 스레드를 임의의 이용가능한 코어로 디스패치한다.

[0015] 일반적으로 말하면, 커널 전원 관리자(114) 및 커널 스케줄러(116)는 프로세서들(104)의 제어와 동작에 있어서 상충적 이해관계가 있다. 커널 전원 관리자는 프로세서들(104)의 전력 소모를 최소화하도록 구성되어 있고, 따라서 프로세서들(104) 중 하나 이상의 주파수 및/또는 전원 상태를 감소시키려고 시도한다. 반대로, 커널 스케

줄러(116)는 처리량(processing throughput)을 극대화하도록 구성되어 있으며, 따라서 작업을 모든 프로세서들(104)에게 분배하려고 한다.

[0016] 컴퓨팅 장치(102)는 또한 부가의 특징들 및 기능들을 가질 수 있다. 예를 들어, 컴퓨팅 장치(102)는 또한, 예를 들어, 자기 디스크, 광 디스크 또는 테이프와 같은 부가의 데이터 저장 장치(이동식 및/또는 비이동식)를 포함할 수 있다. 이러한 부가의 저장 장치가 도 1에서 이동식 저장 장치(118) 및 비이동식 저장 장치(120)로 나타내어져 있다. 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터와 같은 정보를 저장하는 임의의 방법 또는 기술로 구현되는 휘발성 및 비휘발성, 이동식 및 비이동식 매체를 포함할 수 있다. 시스템 메모리(106), 이동식 저장 장치(118), 및 비이동식 저장 장치(120)가 모두 컴퓨터 저장 매체의 예이다. 따라서, 컴퓨터 저장 매체는 RAM, ROM, EEPROM, 플래쉬 메모리 또는 기타 메모리 기술, CD-ROM, DVD(digital versatile disk) 또는 기타 광 저장 장치, 자기 카세트, 자기 테이프, 자기 디스크 저장 장치 또는 기타 자기 저장 장치, 또는 컴퓨팅 장치(102)에 의해 액세스될 수 있고 원하는 정보를 저장하는 데 사용될 수 있는 임의의 기타 매체를 포함하지만, 이들로 제한되는 것은 아니다. 임의의 이러한 컴퓨터 저장 매체가 컴퓨터 장치(102)의 일부일 수 있다.

[0017] 컴퓨팅 장치(102)는 또한 키보드, 마우스, 펜, 음성 입력 장치, 터치 입력 장치 등과 같은 하나 이상의 입력 장치(122)도 가질 수 있다. 디스플레이, 스피커, 프린터 등과 같은 하나 이상의 출력 장치(124)도 역시 직접 또는 컴퓨팅 장치(102)에의 접속을 통해 포함되어 있을 수 있다.

[0018] 컴퓨팅 장치(100)는 또한 장치가 네트워크를 통한 것과 같이 다른 컴퓨팅 장치들과 통신할 수 있게 하는 통신 접속(126)도 포함할 수 있다. 통신 접속(126)은 통신 매체의 일례이다. 통신 매체는 통상적으로 컴퓨터 판독가능 명령어, 데이터 구조, 또는 프로그램 모듈에 의해 구현될 수 있다. "피변조 데이터 신호(modulated data signal)"라는 용어는, 신호 내에 정보를 인코딩하도록 그 신호의 특성들 중 하나 이상을 설정 또는 변경시킨 신호를 의미한다. 예로서, 통신 매체는 유선 네트워크 또는 직접 배선 접속(direct-wired connection)과 같은 유선 매체, 그리고 음향, RF, 적외선, 기타 무선 매체와 같은 무선 매체를 포함하지만 이에 제한되는 것은 아니다. 컴퓨터 판독가능 매체는 컴퓨팅 장치(102)에 의해 액세스될 수 있는 이용가능한 매체라면 어느 것이라도 될 수 있다. 예로서, 컴퓨터-판독가능 매체는 "컴퓨터 저장 매체" 및 "통신 매체"를 포함할 수 있지만 이에 제한되는 것은 아니다.

[0019] 다양한 모듈 및 기술이 본 명세서에서 일반적으로 하나 이상의 컴퓨터 또는 기타 장치에 의해 실행되는, 프로그램 모듈과 같은, 컴퓨터-실행가능 명령어와 관련하여 기술될 수 있다. 일반적으로, 프로그램 모듈은 특정 태스크를 수행하거나 특정 추상 데이터 유형을 구현하는 루틴, 프로그램, 개체, 구성요소, 데이터 구조 등을 포함한다. 이들 프로그램 모듈 등은 네이티브 코드(native code)로서 실행될 수 있거나, 가상 기계 또는 기타 JIT(just-in-time) 컴파일 실행 환경에서와 같이 다운로드되어 실행될 수 있다. 통상적으로, 프로그램 모듈의 기능이 다양한 실시예들에서 원하는 바에 따라 결합되거나 분산되어 있을 수 있다. 이 모듈 및 기술의 구현은 일부 형태의 컴퓨터-판독가능 매체 상에 저장되거나 그 매체를 통해 전송될 수 있다.

[0020] 예시적인 프로세서 구성

[0021] 도 2는 프로세서들에게 작업을 할당하는 것을 가능하게 하도록 코어 파킹 마스크를 생성하고 스레드 스케줄러로 이 마스크를 구현하는 프로세스(200)의 적어도 하나의 실시예의 흐름도를 나타낸 것이다. 프로세스(200)는 하드웨어, 소프트웨어, 또는 이들의 조합으로 구현될 수 있는 동작들의 순서를 나타내는 논리 흐름도에서 블록들의 모음으로 나타내어져 있다. 소프트웨어와 관련하여, 블록들은, 하나 이상의 프로세서들에 의해 실행될 때, 언급된 동작들을 수행하는 컴퓨터-실행가능 명령어들을 나타낸다. 일반적으로, 컴퓨터-실행가능 명령어들은 특정 기능을 수행하거나 특정 추상 데이터 유형을 구현하는 루틴, 프로그램, 개체, 구성요소, 데이터 구조 등을 포함한다. 동작들이 기술되어 있는 순서가 제한으로서 해석되어서는 안 되며, 임의의 수의 기술된 블럭들이 프로세스를 구현하기 위해 임의의 순서로 및/또는 병렬로 조합될 수 있다. 프로세스(200)에 부가하여, 본 명세서에 기술된 다른 프로세스들이 그에 따라 해석되어야 한다. 설명을 위해, 도 1의 시스템(100)을 참조하여 프로세스(200)가 기술되어 있다.

[0022] 도 2에 도시된 바와 같이, 단계(202)에서 코어 파킹 마스크가 시스템 메모리에 생성된다. 예를 들어, 단계(202)에서 운영 체제(108)에 존재하는 커널 전원 관리자(114)가 코어 파킹 마스크를 생성할 수 있다. 예시적 코어 파킹 마스크("비트 마스크" 또는 단순히 "마스크")(204)는 대응하는 코어를 나타내는 셀(cell)을 제공할 수 있다. 도 2에 도시된 바와 같이, 예시적인 시스템은 8개의 코어를 포함하지만, 더 많거나 더 적은 수의 코어들이 사용될 수도 있다. 비트 마스크(204)는 각각의 셀 내에 비트값을 포함하는 데, 여기서 "1"은 파킹된 코

어를 나타내고 "0"은 파킹 해제된 코어를 나타낸다. 파킹된 코어는 저전력 대기 모드에 들어간 코어이다. 몇몇 실시예들에서, 파킹된 코어는 전력 소모가 없으며, 따라서 0 와트를 사용한다. 몇몇 실시예들에서, 코어는 공유 하드웨어 회로와 같은 의존관계를 가진다. 양쪽 코어가 모두 저전력 상태에 들어가면, 의존관계도 역시 암시적으로 저전력 상태로 들어갈 수 있다. 따라서, 다른 마스크와 비교하여 전력 절감을 최대로 하는 코어 파킹 마스크가 선택될 수 있다. 예를 들어, 하나의 프로세서 소켓 내의 모든 코어들을 끄느 것이 2개의 프로세서 소켓 내의 코어들의 절반을 끄는 것보다 더 많은 전력을 절감할 수 있다. 비트 마스크(204)(오른쪽에서 왼쪽으로 0부터 7까지 번호가 매겨져 있음)는 4개의 파킹 코어들, 즉 3, 5, 6, 및 7을 포함한다. 당연히 코어들(0, 1, 2, 및 4)은 파킹 해제된 코어들이다.

[0023] 하나 이상의 실시예들에 따르면, 단계(206)에서 비트 마스크(204)가 반전된 비트 마스크(208)를 생성하기 위해 반전될 수 있다. 예를 들어, 커널 전원 관리자(114)가 반전된 비트 마스크(208)를 생성할 수 있다. 반전된 비트 마스크는 각각의 셀(즉, 코어)에 대한 반전된 비트값을 포함한다. 따라서, "1"로 지정된 코어들은 데이타를 처리하는 데 선호될 수 있는 반면, "0"으로 지정된 코어들은 응용 프로그램 스레드를 처리하는 데 선호되지 않을 수 있다.

[0024] 단계(210)에서, 응용 프로그램 스케줄이 커널 스케줄러(116) 등에 의해 결정된다. 예를 들어, 컴퓨팅 장치(102)는 프로그램 모듈들(110) 중 하나 이상을 실행시킬 수 있다. 프로그램 모듈들(110) 중 일부는 단일 스레드 프로그램을 포함할 수 있는 반면, 다른 프로그램 모듈들은 다중-스레드 모듈들을 포함할 수 있다. 통상적으로, 커널 스케줄러(116)는 우선순위, 코어 이용가능성, 선호도(스케줄링 제한), 및 기타 인자들과 같은 복수의 인자들에 기초하여 각각의 스레드를 이용가능한 코어로 스케줄링한다. 스레드들의 수가 이용가능한 코어들보다 많을 때, 커널 스케줄러(116)는 스레드를 교대로 하거나, 스레드들이 궁극적으로 진행하도록(즉, 프로세서에 의해 실행되도록) 스레드들을 다른 방식으로 스케줄링한다. 프로세서들(104)은 초당 여러번 단일 코어 상에서 스레드를 교대로 할 수 있으며, 따라서 커널 스케줄러(116)가 스레드들을 이용가능한 코어들로 효과적으로 스케줄링할 수 있는 기회를 제공한다.

[0025] 많은 프로그램 모듈이 특정 프로세서들에 스레드들을 할당하지 않는 반면, 일부 복잡한 프로그램 모듈들에서는 스레드가 특정 프로세서에 의해 실행되어야만 할 수도 있으며, 이를 스레드의 프로세서 선호도를 구성하는 것이라고 한다. 프로그램 모듈 스레드 선호도 마스크(212)는, 프로그램 모듈들(110)에 의해 판정되는 바와 같이, 스레드들을 처리하도록 요청받은 코어들을 나타낸다. 예를 들어, 제1 프로그램 모듈은 스레드들이 커널 스케줄러(116)에 의해 코어 0 및 코어 1에 스케줄링되어야 한다는 것을 나타내는 제1 선호도 마스크(212(1))를 가질 수 있다. 제2 프로그램 모듈과 연관된 제2 선호도 마스크(212(2))는 스레드들이 코어 2 및 코어 3에 스케줄링될 수 있다는 것을 나타내는 반면, 다른 프로그램 모듈은 스레드들이 모든 이용가능한 코어들에 스케줄링될 수 있다는 것을 나타내는(모든 코어들이 선택된 것으로 도시되어 있음) 선호도 마스크(212(P))를 포함할 수 있다. 유의할 점은, 유연성 마스크(212(P))가 코어들에 스레드들을 할당하는 데 사실상 어떠한 제약도 없는 특별한 경우라는 것이다.

[0026] 몇몇 실시예들에서, 단계(214)에서, 프로그램 모듈 선호도 마스크들(212)은 "AND" 연산자(216)를 사용하여 한번에 하나씩 반전된 비트 마스크(208)과 결합되어 이용가능한 프로세서 세트(218)에 적격인 일련의 프로세서들을 결정한다. 단계(220)에서, 제1 선호도 마스크(212(1))는 제1 이용가능한 프로세서 세트(218(1))를 생성하는 데 사용된다. 프로세스(200)는 각각의 프로그램 모듈에 대한(즉, 반전된 비트 마스크(208) 및 선호도 마스크(212)의 각각의 조합에 대한) 동작들(210, 214, 및 220)의 반복 프로세스를 포함할 수 있다. 따라서, 제2 선호도 마스크(212(2))는 동작들(210, 214, 및 220) 또는 기타의 두번째 반복 동안에 제2 이용가능한 프로세서 세트(218(2))를 생성하는 데 사용된다.

[0027] 앞서 기술한 바와 같이, 각각의 코어의 비트값(예를 들어, 코어 0, …, 코어 7)은 스레드들을 스케줄링하는 데 이용가능한 프로세서 세트(218)를 결정하는 데 사용된다. "AND" 연산자(216)는 피연산자들(즉, 하나의 코어에 대한 결합되는 비트값들) 둘다가 특정의 코어로 우선 할당(affinitize)되는 스레드를 나타내는 "1"을 포함하는 경우 "1"의 코어 비트값을 반환한다. 예를 들어, 제1 선호도 마스크(212(1))가 반전된 비트 마스크(208)와 결합될 때, 코어 0 및 코어 1 둘다가 활성 코어이고 "1"의 코어 비트값을 반환할 것인 반면, 나머지 코어들(2-7)은, 제1 이용가능한 프로세서 세트(218(1))에 예시로서 나타낸 바와 같이, "0"의 코어 비트값을 포함한다.

[0028] 제2 선호도 마스크(212(2))는 코어 3에 "1"의 코어값을 포함하는 반면, 반전된 비트 마스크(208)는 코어 3이 파킹되어 있는 것을 나타낸다. 커널 스케줄러(116)는 코어 3이 "1"의 코어값을 포함하는 제2 이용가능한 프로세서 세트(218(2))에 나타내어져 있는 제2 선호도 마스크(212(2))를 수용하기 위해 반전된 비트 마스크를 무시하

기로 할 수 있으며, 따라서 반전된 비트 마스크(208)에 파킹되는 것으로 지정된 코어(이 코어가 그 후에 파킹 해제될 수 있음)로 작업을 이동시킨다. 몇몇 실시예들에서, 스레드는 임의의 수의 발견적 학습법들을 사용하여 스케줄링될 수 있다. 반전된 비트 마스크(208)를 무시하면서 스레드의 선호도에서의 최적의 코어가 사용될 수 있다. 최적의 코어가 파킹되어 있는 경우, 대안은 동일한 NUMA(non-uniform memory access) 노드 내의 프로세서를 선호된 코어로서 선택하는 것을 포함할 수 있다. 스케줄러는 코어 파킹 반전된 마스크를, 스레드를 실행할 선호된 장소들의 힌트로서 취급하지만, 가장 성능이 좋은 옵션이라고 생각되는 것의 엄격한 제한(엄격한 선호도) 중에서 선택될 것이다

[0029] 선호도 마스크(212(P))에서와 같이, 모든 코어들이 이용가능한 것으로 지정된 경우, 비트값이 무시될 수 있는데, 그 이유는 프로그램 모듈에게 스레드들이 어느 코어에 의해서도 실행될 수 있다는 것을 알려주기 때문이다. 선호도 마스크(212(P))는 코어값들이 "1/0"("1" 또는 "0", 반면에 선호도 마스크(212(P))의 작업을 스케줄링하는 것을 가능하게 하도록 적어도 하나의 코어가 "1"의 코어값을 가지고 있어야만 함)을 나타내는 것에 의해 이용가능한 프로세서 세트(218(P))가 나타내는 바와 같이, 코어 0, 코어 1, 코어 2 및 코어 4와 같이 반전된 비트 마스크(208)에 의해 표시된 코어들 중 임의의 코어에 스케줄링된 코어들을 포함할 수 있다. 몇몇 실시예들서, 이용가능한 프로세서 세트(218(P))는, 이용가능한 코어들에 대한 작업의 배정을 결정할 때, 파킹 해제되고 아이들 상태에 있는 코어들을 선택할 수 있다. 이상적으로는, 작업 이동은 작업을 코어 0, 코어 1, 코어 2, 및 코어 3에 배정할 수 있으며, 따라서 코어 4를 후속하는 동작에서 사용되지 않고 어쩌면 파킹된 채로 둘 수 있다. 이하에서 기술하는 바와 같이, 기타 고려사항들은 선호되는 장소를 생성하기 위해 선호도 마스크(212(P))에 따라 어느 코어들이 작업을 배정받는지를 결정할 수 있다. 예를 들어, 선호되는 장소는 선택적인 메모리 액세스 성능과 같은 인자들에 기초할 수 있다.

[0030] 동작들(210, 214, 및 220)은, 도 2를 참조하여 이상에서 기술한 바와 같이, 스레드들을 스케줄링하는 데 사용될 수 있다. 또한, DPC(deferred procedure call, 지연된 프로시저 호출), 타이머, 처리 인터럽트, 또는 기타 프로세서 작업과 같은 기타 작업이 동작들(210, 214, 및 220)에 의해 프로세스(200)에서 수행될 수 있다.

[0031] 하나 이상의 실시예들에 따르면, 도 2는 시간 슬라이스(time slice)에 대한 예시적인 코어 사용(222)을 나타낸 것이다. 예를 들어, 이용가능한 프로세서 세트(218)로부터의 활성인 코어들을 조합하면 주어진 시간 체크 구간(예컨대 100 밀리초, 이에 제한되지 않음) 동안의 코어 사용(222)이 얻어질 수 있다. 앞서 기술한 바와 같이, 이용가능한 프로세서 세트(218(P)) 내의 코어 4에 작업이 스케줄링되어 있는지 여부에 따라 코어 사용(222) 내의 코어 4가 사용됨으로 표시되거나 그렇지 않을 수 있다. 커널 전원 관리자(114)의 관점에서 볼 때, 코어 사용(222)은 이상적으로 코어 4에 대해 "0"의 코어값을 포함하게 되며, 이로써 파킹 해제된 코어들의 수를 최소로 하고 그 결과 전력 소모가 감소된다. 코어 사용(222)과 관계없이, 그 다음 시간 슬라이스에 대해 새로운 코어 파킹 마스크가 생성될 수 있는데, 이는 새로운 코어 파킹 마스크를 결정하기 위해 코어 사용(222)으로부터의 정보를 사용할 수 있다. 일부 경우들에서, 코어 2가 제2 선호도 마스크(212(2))를 충족시킬 정도로 충분한 처리 활용도를 갖지 못할 수 없고 여전히 파킹 해제된 코어이기 때문에, 코어 3이 필요하지 않을 수도 있다.

[0032] 도 3a 및 도 3b는 본 발명의 적어도 하나의 실시예에 따른 코어 활용을 나타낸 것이다. 보다 구체적으로는, 도 3b는 도 3a에 도시된 것과 같은 예시적인시스템 코어 활용에 응답한 코어 활용의 예시적인 수정을 나타낸 것이다.

[0033] 하나 이상의 실시예들에 따르면, 시스템(300)의 대체 실시예들에서 더 많거나 더 적은 수의 코어들이 포함될 수 있지만, 도 3a는 복수의 코어들(302, 304, 306 및 308)을 포함하는 시스템(300)을 나타낸 것이다. 커널 전원 관리자(114)는 전원 정책을 사용하여 코어들을 제어할 수 있다. 전원 정책은 활성 코어의 수를 결정할 수 있고 커널 전원 관리자(114)가 이용가능한 코어들의 세트를 어떻게 산출할 수 있는지에 영향을 줄 수 있다. 전원 정책은 이용가능한 코어들의 수를 제한하거나, 커널 전원 관리자(114)가 이용가능한 코어들의 수를 스케일링할 수 있게 하는 데 사용될 수 있다. 플랫폼에 의해 지원되는 코어들의 수는 서로 다른 시스템들에 걸쳐 다를 수 있으며, 따라서 몇개의 코어들이 이용가능한지를 미리 알고 있지 않고서도 커널 전원 관리자에 의해 코어들의 수가 지정될 수 있게 하는 유연한 방식이 필요하다. 따라서, 몇몇 실시예들에서, 사용될 코어들의 수는 최대 코어 활용의 퍼센트로서 표현될 수 있다. 따라서, 코어들을 구현하는 데 그 퍼센트를 코어들의 수를 나타내는 그 다음의 이용가능한 수로 반올림할 필요가 있을 수 있다. 예를 들어, 코어 사용 퍼센트가 60%로 산출될 수 있다. 시스템이 4개의 코어들을 포함하는 경우, 퍼센트는 75%로 반올림될 수 있고, 3개의 코어가 파킹 해제될 것인 반면 1개의 코어가 파킹되어 있다.

[0034] 각각의 코어는, 코어의 성능 상태와 독립적으로, 총 시간 중 코어의 실행시간(run time)의 퍼센트로 표현되는,

그 코어의 작업량을 나타내는 코어 활용("코어 활용도")(310)을 포함한다. 예를 들어, 코어 0(302)는 코어 "0"의 작업 최대 용량보다 20% 더 적은 작업을 하고 있다는 것을 나타내는 80%의 코어 활용도를 가질 수 있다. 따라서, 100%의 코어 활용도는 코어가 최대 용량으로 작업하고 있다는 것을 나타내는 반면, 0% 코어 활용도를 갖는 코어는 사용되지 않는 코어를 나타낸다. 몇몇 실시예들에서, 커널 전원 관리자(114)는 코어 활용도(310)를 모니터링할 수 있다.

[0035] 또한, 각각의 코어는 성능 상태(고급 구성 및 전원 인터 페이스(ACPI) p-상태)(312)를 포함할 수 있다. p-상태(312)는 코어 주파수 및 전압 설정이며 커널 전원 관리자(114)에 의해 제어된다. P-상태(312)는 모터의 스로틀 제어와 유사하다. 100%의 p-상태(312)는 코어의 최대 성능 상태를 나타내는 반면, 50%의 p-상태는 대응하는 감소된 코어 전압 레벨을 갖는, 최대 주파수의 1/2에 있는 코어를 나타낸다. 유의해야 할 점은, 코어의 실제 전력 소모가 p-상태(312)와 일치하지 않거나 비례하지 않을 수 있다는 것이다. 예를 들어, 코어 전력 누설, 코어-기반 전력 소모, 및/또는 기타 인자들과 같은 기타 인자들로 인해, 코어의 p-상태(312)를 두배로 하는 것이 코어의 전력 소모를 두배로 하지 않을 수 있다. 몇몇 실시예들에서, 커널 전원 관리자(114)는, 전원 정책을 참조하는 것에 의한 것과 같이, p-상태(312)를 결정 및/또는 제어할 수 있다.

[0036] 각각의 코어는 전체 작업 용량에 대한 코어의 작업량을 나타내는 출력 활용값("출력 활용도")(314)을 포함한다. 예를 들어, 출력 활용도(314)는 0-10,000의 스케일을 가질 수 있으며, 0은 활용 없음을 나타내는 반면, 10,000은 최대 활용을 나타낸다. 출력 활용도(314)는 코어 활용도(310)를 p-상태(312)와 곱하여 산출될 수 있다. 예를 들어, 코어 0(302)는 80%의 코어 활용도 및 80%의 p-상태를 포함하며, 따라서 출력 활용도는 6,400이다. 일부 실시예들에서, 출력 활용도(314)는 커널 전원 관리자(114)에서, 전원 정책을 참조하는 것에 의한 같이, 코어 파킹 여부를 결정하고 및/또는 p-상태(312) 설정을 결정하는 데 사용된다.

[0037] 몇몇 실시예들에서, 시스템(300)은 제1 코어 블럭(316(1)) 및 제2 코어 블럭(316(2))과 같은 코어 블럭(316)을 포함하지만, 더 많거나 더 적은 수의 코어 블럭들이 시스템(300)에서 구현될 수 있다. 코어 블럭(316)은, 듀얼-코어 또는 다중-코어 프로세서와 같은, 하나의 회로에 복수의 코어를 갖는 플랫폼을 나타낸다. 각각의 코어 블럭(316)은 독자적인 전력 소모 특성들을 가질 수 있다. 예를 들어, 코어는 활성 레벨 전력 소모, 코어 누설, 또는 블록 내의 코어들 중 어느 하나가 파킹 해제될 때 일어나는 기타 전력 손실들을 포함할 수 있다. 예를 들어, 제1 코어 블럭(316(1)) 내의 코어들 둘다가 파킹 해제되고 10,000의 출력 활용도를 가지고 있는 경우, 결합된 전력 소모는 두배(2x)(와트)일 수 있다. 제1 코어 블럭(316(1)) 내의 코어 0(302)가 그 후에 파킹되고(예를 들어, 출력 활용이 0이고) 코어 1(304)가 그대로 있는 경우, 전력 누설, 활성 전력 소모, 및/또는 기타 인자들과 같은 코어 블럭(316)과 연관된 인자들로 인해, 결합된 전력 소모는 x 와트보다 클 수 있다. 코어 1이 그 후에 파킹될 때, 그 결과에 의한 전력 소모가 0 와트일 수 있다. 따라서, 다른 코어들이 그 후에 파킹되기 전에 전체 코어 블록들이 파킹될 수 있도록 코어들을 파킹함으로써, 전력 절감을 최대화하는 것이 유익할 수 있다.

[0038] 도 3a에 도시된 바와 같이, 시스템(300)의 예시적인 구현은 다른 코어들이 파킹 해제된(즉, 활성인) 동안 파킹된 코어 3(308)를 포함한다. 비록 코어 3이 100%의 p-상태를 포함하고 있지만, 이것이 코어 3에 전력이 공급된다는 것을 나타내는 것은 아닐 수 있다. 달리 말하면, 커널 전원 관리자(114)는 p-상태를 0%보다 큰 레벨에 두면서 코어를 파킹할 수도 있다.

[0039] 예시적인 활용 시나리오에서, 커널 전원 관리자(114)는 30,000의 최대 총 시스템 활용도(즉, 3개의 파킹 해제된 코어 x 10,000 = 30,000) 중에서 12,600의 총 시스템 활용도(즉, 6,400 + 3,200 + 3,000 = 12,600)를 결정하기 위해 시스템(300)의 출력을 산출할 수 있다. 이상에서 언급한 활용 숫자들은 총 시스템 활용도를 사용하여 수행될 수 있는 산출의 성질상 설명적인 것으로 보아야 하며, 따라서 본 발명을 제한하는 것이 아니다.

[0040] 도 3b는 도 3a에 도시된 총 시스템 활용에 응답한 수정된 시스템(318)에서의 코어 활용의 예시적인 수정을 나타낸 것이다. 수정 시스템(318)은 코어들(302, 304, 306, 및 308)의 파킹된/파킹 해제된 상태의 수정을 포함한다. 시스템(300)을 참조하여 이상에서 언급한 바와 같이, 총 시스템 활용도가 일례에서 12,600인 것으로 산출되었다. 따라서, 커널 전원 관리자(114)는 현재의 작업 수요를 충족시키기 위해 시스템의 능력을 축소하지 않으면서 부가의 코어를 파킹할 수 있는데, 그 이유는 총 시스템 활용도가 2개의 코어의 최대 활용 용량보다 작기 때문이다(즉, 12,600 < 20,000). 일부 실시예들에 따르면, 커널 전원 관리자(114)는, 어쩌면 시스템의 사용자에게 완전히 투명할 수 있는 전원 정책 또는 기타 인자들에 기초하여, 파킹할 코어를 선택할 수 있다. 사용자 투명성은 더 나은 전원 효율이라는 주된 목적을 제외한 사용자 감지가능 시스템 또는 응용 프로그램 성능 영향을 포함하지 않는다. 코어 파킹을 구현하는 데 필요한 어떤 변경들도 아주 낮을 레벨의 운영 체제 내부 구성 요소들로 제약될 수 있고, 따라서 코어 파킹과 연관된 최종 사용자에 대한 어떤 행동적 또는 경험적 변경도 없

을 수 있다.

[0041] 제2 코어가 파킹될 때, 수정된 시스템의 최대 총 시스템 활용도가 20,000으로 떨어질 것이다(2개의 코어 x 10,000). 커널 전원 관리자(114)는 (도 3a에 도시된 바와 같이) 이전 상태에서 활성이었던 모든 코어들을 파킹할 수 있다. 앞서 기술한 바와 같이, 제2 코어 블럭(316(2))을 완전히 파킹하도록 코어 2(306)를 파킹하는 것으로 인해 파킹된 코어 블럭(320)이 생기는 것이 유익할 수 있다. 따라서, 파킹된 코어 블럭(320)은 코어 2(306) 대신에 코어 0(302) 또는 코어 1(304)를 파킹하는 것과 비교하여 전력 절감을 증가시킬 수 있다.

[0042] 시스템(318)의 모든 사용자에게 투명한 변경에 대응하기 위해, 파킹된 코어(코어 2)의 출력 활용도가 나머지 파킹 해제된 코어들(즉, 코어 0 와 코어 1)에 홉수 또는 재분배되어야 한다. 도 3a에서, 코어 2 의 출력 활용도는 3,000이었다. 따라서, 한 경우에, 각각의 코어가 각각의 코어마다 1,500의 부가의 출력 활용도를 가정하는 수정된 출력 활용도(322)를 갖게 함으로써, 파킹 해제된 코어들이 부담을 균등하게 공유할 수 있다. 몇몇 경우들에, 파킹된 코어의 총 활용도의 다른 구분들은 활용도를 파킹 해제된 코어들에 배정할 때 사용될 수 있다. 예를 들어, 파킹 해제된 코어는 용량(최대 활용도인 10,000에 가까운 총 활용도) 근처에서 실행될 수 있다. 이러한 경우에, 더 큰 대역폭을 갖는 코어가 파킹된 코어의 총 활용도를 더 많이 흡수할 수 있다.

[0043] 하나 이상의 실시예들에서, 커널 전원 관리자(114)는 코어 0(302) 및 코어 1(304)의 코어 활용도(310)를 각각 79% 및 47%로 조정할 수 있다. 또한, 커널 전원 관리자(114)는 파킹 해제된 코어들 둘다에 대해 p-상태(312)를 100%로 증가시킬 수 있다. 따라서, 수정된 시스템(318)의 총 시스템 활용도가 12,600에 있는 시스템(300)의 총 시스템 활용도와 같게 유지된다.

[0044] 수정된 시스템(318)은 수정된 시스템(318)에 대해 전력 소모가 감소되면서 사용자들에게 투명한 코어 수정을 대응하기 위해, 파킹된/파킹 해제된 상태, 코어 활용도(310), 및/또는 p-상태(312)의 한가지 가능한 수정을 나타낸 것이다. 그러나, 전력 소모의 절감을 가져오고 사용자들에게 투명한 많은 다른 수정들이 시스템(318)에서 행해질 수 있다. 예를 들어, p-상태가 50%로 감소되어 4700의 총 활용도가 얻어지면서 코어 1(304)의 코어 활용도(310)가 94%로 증가될 수 있다. 커널 전원 관리자(114)와 커널 스케줄러(116)의 상충하는 이해관계를 고려할 수 있는 전원 정책에 의해 수정 전략이 결정될 수 있다.

[0045] 수정된 시스템(318)에 영향을 줄 수 있는 기타 고려사항들이 전원 정책에 구현될 수 있다. 예를 들어, 시간에 민감하지 않은 스레드들(예를 들어, 백그라운드 스레드들)은 변경들을 사용자들에 투명하게 유지하면서 코어 파킹 결정에 영향을 덜 줄 수 있다. 보다 구체적으로는, 주어진 코어에서 실행되는 스레드 우선순위들에 걸쳐 평균 실행시간(runtime)을 분배하는 것을 비롯하여, 커널 전원 관리자(114)는 낮은 우선순위 스레드들 및 작업들이 높은 우선순위 스레드 및 작업만큼 중요하지 않도록 코어의 산출된 활용도를 스케일링할 수 있다.

[0046] 도 4는 스레드 스케줄링 및 프로세서들의 동적 사용을 평가하고 프로세서들의 수정된 구성을 결정하는 예시적인 프로세스(400)의 적어도 하나의 실시예의 흐름도를 나타낸 것이다. 프로세스(400)는, 운영 체제(108) 내의 및/또는 시스템 메모리(106)에 존재하는 기타 모듈들에 부가하여, 커널 전원 관리자(114)에 의해 구현된다.

[0047] 하나 이상의 실시예들에 따르면, 커널 전원 관리자(114)는 (코어 활용도(310), p-상태(312), 기타를 비롯한 인자들 중에서도 특히) 출력 활용도(314)를 모니터링하는 기간을 평가한다. 각각의 스레드를 처리하는 데 이용가능한 시간 슬라이스와 같거나 그보다 긴 기간이 선택될 수 있다. 주어진 빈도수에서, 커널 전원 관리자(114)는 코어들의 모니터링을 시작할 수 있다.

[0048] 단계(404)에서, 커널 전원 관리자(114)는 총 시스템 활용도 및 최대 총 시스템 활용도를 산출할 수 있다. 커널 전원 관리자(114)는, 커널 전원 관리자(114)의 전력 절감 요구와 커널 스케줄러(116)의 코어 이용가능성(처리 성능) 요구를 균형을 이루게 하도록 코어 사용을 어떻게 조정할지를 결정하기 위해, 406에서 전원 정책을 검토할 수 있다. 단계(408)에서, 코어 사용을 조정할지 여부 및 조정하는 경우, 사용자들에 대한 투명한 변경들, 온도 요구사항들에 대응하는 것, 및/또는 다른 제약조건들에 대응하는 것과 같은 시스템 목적들을 달성하기 위해 사용을 어떻게 조정할지를 결정하는 데 다른 인자들이 사용될 수 있다.

[0049] 단계(410)에서, 커널 전원 관리자(114)는 코어들에 대한 새로운 구성들을 산출할 수 있다. 예를 들어, 도 2에 도시된 바와 같이, 커널 전원 관리자는 새로운 비트 마스크(202)를 산출할 수 있다. 하나 이상의 실시예들에서, 단계(410)에서 이용가능한 프로세서 세트(218)를 생성하기 위해 비트 마스크가 조정될 수 있다. 단계(412)에서, 커널 전원 관리자(114)는 단계(410)으로부터의 구성들을 구현할 수 있다. 몇몇 실시예들에서, 미리 정해진 빈도수로 반복하는 것 등에 의해, 동적 코어 할당을 생성하기 위해 프로세스(400)가 반복할 수 있다. 대안으로서, 프로세스(400)는 주어진 기간에 걸쳐 정적 코어 할당을 생성할 수 있다.

[0050] 예시적인 동작

[0051] 도 5는 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용을 제공하는 예시적인 프로세스(500)의 적어도 하나의 실시예의 흐름도를 나타낸 것이다. 프로세스(500)가 프로세스의 순서 및 구현과 관련해서는 도 2의 프로세스(200)와 유사하게 해석되어야 한다. 예를 들어, 프로세스(500)에서의 동작들이 기술되어 있는 순서가 제한으로서 해석되어서는 안되며, 임의의 수의 기술된 블럭들이 프로세스를 구현하기 위해 임의의 순서로 및/또는 병렬로 조합될 수 있다. 설명을 위해, 도 1의 시스템(100)을 참조하여 프로세스(500)가 기술되어 있다.

[0052] 하나 이상의 실시예들에 따르면, 단계(502)에서 주기적인 평가 루틴인 "TimeCheck"가 시작될 수 있다. 예를 들어, 단계(502)에서 DPC(deferred procedure call)가 시작할 수 있다. 일부 실시예들에서, 100 ms, 50 ms와 같은 시간값 또는 다른 시간값에 대해 전원 정책 파라미터 "TimeCheck"에 의해 구성된 일정한 주기적인 비율로 실행되는 DPC를 통해 각각의 코어 상에서 상태 시스템에 들어간다. 단계(504)에서, 커널 전원 관리자(114)는 코어들에 대한 메트릭들을 수집할 수 있다. 예를 들어, DPC는 활성 코어들에 대한 메트릭들을 스냅핑하기 위해 각각의 현재 활성인 코어로 큐잉된다. 메트릭들은, 가능한 메트릭들 중에서도 특히, 코어 활용도, 스레드 우선순위 분포, 각각의 코어에 대한 준비된 스레드들의 평균 대기 시간, 및/또는 아이들 상태 레지던시에 대한 성공 및 실패 메트릭들을 포함할 수 있다.

[0053] 단계(506)에서, 커널 전원 관리자(114)는 코어 파킹 마스크(204)와 같은, 비트 마스크를 산출할 수 있다. 예를 들어, 활용도 임계값, 전원 정책, 및/또는 어떤 종속성 관계들에 기초하여 활성 코어들의 목표 수에 대한 새로운 값이 산출될 수 있다. 단계(508)에서, 이용가능한 프로세서 세트(218)를 생성하도록 커널 전원 관리자(114)에 의해 감독되는 프로세스(200)의 일부를 구현하는 것 등에 의해 활성 세트가 업데이트될 수 있다.

[0054] 단계(510)에서, 활성 세트가 운영 체제(108)에 의해 구현될 수 있다. 단계(512)에서, 커널 전원 관리자(114)는 코어들이 추가되었는지(파킹 해제되어 있는지)를 판정할 수 있다. 코어들이 파킹 해제된 경우, 단계(514)에서 커널 스케줄러(116)에게 통보될 수 있고 스레드들을 스케줄링하기 위해 파킹 해제된 코어들을 사용하기 시작할 수 있다. 이러한 파킹 해제된 코어들은 (즉, 다른 프로세서로부터의) 원격 스레드 스케줄링의 대상일 수 있거나, 다른 프로세서들로부터의 스레드들을 선제적으로 선택할 수 있다. 몇몇 실시예들에서, 스레드들이 파킹 해제된 코어들 상에서 실행되도록 재할당될 수 있고, 그로써 다른 파킹 해제된 코어들의 작업을 경감시킬 수 있다.

[0055] 단계(516)에서, 예상된 코어 출력 활용도가 커널 전원 관리자(114)에 의해 산출될 수 있고, 단계(506)으로부터의 파킹 해제된 코어들을 포함할 수 있다. 따라서, 단계(506)에서 결정된 활성 코어들의 수에 대한 새로운 값이 예상된 프로세서 활용도를 산출하기 위해 사용된다. 단계(518)에서, 커널 전원 관리자(114)는 p-상태에 대한 새로운 값을 산출할 수 있다. 일부 실시예들에서, DPC는 그의 목표 p-상태를 업데이트하도록 각 코어 상에서 스케줄링된다. 예를 들어, 예상된 활용도가 증가하면, 단계(512)에서 코어들이 파킹 해제된 경우, p-상태 값도 증가할 수 있다. 그러나, 단계(512)에서 코어들이 파킹 해제된 경우, p-상태는 커널 전원 관리자(114)의 전력 절감 요구와 커널 스케줄러(116)의 처리 요구를 균형을 이루도록 증가 또는 감소될 수 있다.

[0056] 단계(520)에서, 커널 전원 관리자(114)는 단계(518)에서의 계산으로부터의 결과에 기초하여 p-상태 및/또는 t-상태 (선형 스로틀 상태)의 현재 값들이 수정되어야 하는지를 결정한다. 단계(520)에서 p-상태 및/또는 t-상태가 수정되는 경우, 단계(522)에서 커널 전원 관리자(114)는 영향받은 코어들로 전환 DPC들을 큐잉할 수 있다. 따라서, 커널 스케줄러(116)에 의해 스케줄링되는 스레드들은 단계(520)에서 구현되는 바와 같이 새로운 p-상태 및/또는 t-상태에서 실행되는 코어들로 스케줄링될 수 있다.

[0057] 단계(524)에서, 커널 전원 관리자(114)는 코어가 단계(506)에서 파킹되었는지를 판정한다. 예를 들어, 단계(516)으로부터 예상되는 코어 출력 활용도가 현재의 활용도보다 작을 수 있다. 단계(524)에서 코어가 파킹된 경우, 커널 전원 관리자(114)는 새로 파킹된 코어들로 스레드들을 스케줄링하는 것을 종료하라고 단계(526)에서 커널 스케줄러(116)에 통지할 수 있다. 활성 코어 마스크에 추가되거나 그로부터 제거되는 모든 코어들에 대해서, 그 코어에 대해 DPC가 스케줄링된다. 도 3a 및 도 3b를 참조하여 기술한 바와 같이, 파킹된 코어로부터의 작업이 하나 이상의 파킹 해제된 코어들로 재배정될 수 있다. 몇몇 실시예들에서, 새로 파킹된 코어들이 이용할 수 있는 가장 깊은 c-상태로 될 수 있다. 마지막으로, 단계(528)에서, 프로세스(500)가 반복될 수 있다.

[0058] 도 6은 도메인 아이들 어카운팅(domain idle accounting)을 더 포함하는, 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용을 제공하는 예시적인 프로세스(600)의 적어도 하나의 실시예의 다른 흐름도를 나타낸 것이다. 프로세스(600)는 도 5에 기술된 하위-프로세스들(sub-processes) 중 복수를 포함하며, 따라서 도 5로부터의 그

하위-프로세스들에 대해 다시 설명하지 않을 것이다.

[0059] 단계(602)에서, 커널 전원 관리자(114)는 도메인 아이들 어카운팅(domain idle accounting)이 사용될 수 있는지를 결정할 수 있다. 도메인 아이들 어카운팅이 사용될 수 있는 경우, 도메인 마스터(domain master)는 단계(604)에서 도메인(예를 들어, 코어 블럭(316) 또는 모든 코어들)에 대한 메트릭들(metrics)을 스냅핑한다.

[0060] 몇몇 실시예들에서, 도메인 아이들 어카운팅은 단계(606)에서 다른 결정을 시작할 수 있다. 도메인 아이들 어카운팅이 사용가능하게 되면, 커널 전원 관리자(114)는 단계(608)에서 도메인 목표 p-상태를 산출할 수 있다. 마지막으로, 단계(610)에서, 프로세스(600)가 반복될 수 있다.

[0061] 도 7은 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용을 위한 부가의 전원 정책 설정들 및 기타 입력들을 제공하는 프로세스(700)의 적어도 하나의 실시예의 흐름도를 나타낸 것이다. 몇몇 실시예들에서, 코어 파킹은 프로세서 성능 상태들에 대해 사용될 목표 상태를 산출하는 기존의 상태 시스템에 대한 개선으로서 구현될 수 있다. 어떤 주어진 시간에 사용될 코어들의 정확한 수는 프로세스(700)를 참조하여 이하에서 기술한 바와 같은 복수의 인자들에 기초하여 결정될 것이다.

[0062] 하나 이상의 실시예들에 따르면, 현재의 전원 정책은 단계(702)에서 이용될 코어들의 수를 설정하는데 사용될 수 있다. 단계(704)에서, 코어들의 최소 또는 최대 수가 설정될 수 있다. 커널 전원 관리자(114)는 에너지 효율적인 방식으로 주어진 작업을 완료하는 데 요구되는 활성 코어들의 수를 산출할 수 있다. 몇몇 경우들에서, 최소 수의 코어들을 실행하는 것이 전력 절감에 유리할 수 있고 커널 전원 관리자(114)에 유익할 수 있다. 반대로, 최대 수의 코어들을 실행하는 것이 최고 레벨의 성능을 제공할 수 있으며, 따라서 커널 스케줄러(116)에 유익할 수 있다.

[0063] 단계(706)에서, 코어 파킹에 대한 부가의 전원 정책 파라미터들은 하위-프로세스들(706(1),… 706(4)) 중 하나 이상을 사용하여 커널 전원 관리자(114)에 의해 구현될 수 있다. 단계(706(1))에서, 코어들을 파킹하는 및/또는 파킹 해제하는 데 요구되는 시간 간격이 조정될 수 있다. 예를 들어, 코어들을 파킹 및/또는 파킹 해제하는 빈도수가 간격을 변경함으로써 조작될 수 있다. 몇몇 실시예들에서, 코어들은 제1 간격으로 파킹되고 제2 간격으로 파킹 해제될 수 있다. 예를 들어, 전력 절감에 유리한 정책에서 코어들을 파킹하는 것은 매 100 ms마다와 같이 빈번하게 행해질 수 있지만, 코어들을 파킹 해제하는 것은 단지 매 500 ms마다 행해질 수 있다.

[0064] 단계(706(2))에서, 커널 전원 관리자(114)는 증가 및/또는 감소 정책을 구현할 수 있다. 예를 들어, 제1 정책 옵션은, 한번에 하나의 코어와 같이, 한번에 설정된 수의 코어들만을 파킹할 수 있다. 제2 정책 옵션은 이상적 코어 활용도를 달성하기 위해 코어들을 파킹 또는 파킹 해제할 수 있고, 따라서, 한번에 복수의 코어들을 파킹 및/또는 파킹 해제할 수 있다. 제3 정책은 한쪽 극단 또는 다른쪽 극단으로 갈 수 있다(가능한 한 많이 파킹하거나 가능한 한 많이 파킹 해제할 수 있다).

[0065] 단계(706(3))에서, 요구되는 활용도 임계값이 프로세서들의 사용중 상태(busyness)에 기초하여 증가되거나 감소될 수 있다. 예를 들어, 프로세서(또는 다른 프로세서들)가 주어진 기간 동안 사용중 상태를 포함할 때까지 프로세서는 파킹됨 또는 파킹 해제됨의 상태의 변경을 겪지 않을 수 있다. 이것은 프로세서가 연속하여 파킹된 상태와 파킹 해제된 상태 사이를 왔다갔다하는 것을 감소시킬 수 있다.

[0066] 마지막으로, 단계(706(4))에서, 파킹 해제된 코어들을 스케일링하는 정책이 구현될 수 있다. 예를 들어, 커널 전원 관리자(114)는 현재 활성인 세트 내의 프로세서 코어들의 수에 기초하여 이상적인 목표 프로세서 성능 상태를 산출할 수 있다. 절력 절감, 성능 및 특정의 작업에 대한 응답성 간의 최선의 트레이드오프를 제공하기 위해, 커널 전원 관리자(114)는 유익하게도 높은 성능 상태에서 더 적은 수의 프로세서들을 실행할 수 있거나, 이와 반대로 낮은 성능 상태에서 더 많은 수의 코어들을 실행할 수 있다.

[0067] 단계(708)에서, 코어들 및/또는 시스템 발견적 학습법은 하위-프로세스들(708(1) 및 708(2)) 중 하나 이상을 사용하여 커널 전원 관리자(114)에 의해 구현될 수 있다. 단계(708(1))에서, 커널 전원 관리자(114)는 깊은 프로세서 아이들 전원 상태(절전 모드)의 성공적 사용에 기초하여 요구되는 활성 코어들의 수 및 활성 사용중인 코어들의 최적의 수행 상태를 산출할 수 있다. 이것에 의해, 깊은 아이들 상태가 활성(파킹 해제된) 코어들의 세트에 걸쳐 효율적으로 사용되지 않을 때를 커널 전원 관리자가 감지할 수 있다. 전력을 절감하면서도 여전히 성능을 제공하기 위하여, 더 많은 코어들을 파킹 상태에 들어가게 하고 나머지 활성 코어들의 성능 상태를 향상시켜 작업이 효율적으로 실행되게 하는 것이 유익할 수 있다.

[0068] 단계(708(2))에서, 코어에 배정(할당)될 준비가 된 스레드들에 대해 평균 대기 시간이 사용될 수 있다. 예를 들어, 준비 상태에 있는 스레드들에 대한 평균 대기 상태의 분포를 사용하여 커널 전원 관리자(114)는 요구된

프로세서 코어들의 수를 계산할 수 있으며, 이로써 커널 전원 관리자가 준비 상태의 스레드들이 실행될 수 있기 전에 지연 시간을 줄이기 위해 사용 중인 코어들의 수를 스케일링할 수 있으며, 이로써 성능 및 응답성을 증가시킨다. 일례로서, 많은 수의 스레드들이 실행될 필요가 있을 때, 코어들의 p-상태를 감소키면서 코어들을 파킹 해제시키는 것이 유리할 수 있는데, 그 이유는 각각의 코어가 한번에 하나씩의 스레드만을 처리할 수 있기 때문이다. 따라서, 일부 경우들에, 코어 이용가능성이 높을수록 많은 수의 스레드들을 처리하는 것이 가능하게 된다.

[0069] 단계(710)에서, 아이들 상태 의존성은 커널 전원 관리자(114)에서 코어 파킹 구현을 조정하는 데 사용될 수 있다. 커널 전원 관리자(114)는, 어느 특정의 코어들이 파킹되거나 파킹 해제되어야 하는지를 선택할 때, 어느 코어들이 전원 또는 클럭 리소스들을 공유할 수 있는가를 결정하고 공유된 제어들에 기초하여 가장 전원 효율적인 방식으로 코어들을 파킹하거나 파킹 해제하기로 하기 위해 먼저 코어들의 아이들 상태 종속성 관계를 조사하게 된다.

[0070] 단계(712)에서, 하나 이상의 코어들을 파킹할지 파킹 해제할지를 결정할 때 성능 및 스로틀 상태 관계가 고려될 수 있다. 성능 상태 또는 스로틀링 제어를 공유하는 코어들은 보다 높은 전원 효율성을 실현하기 위하여 전부 파킹되거나 파킹 해제될 수 있다.

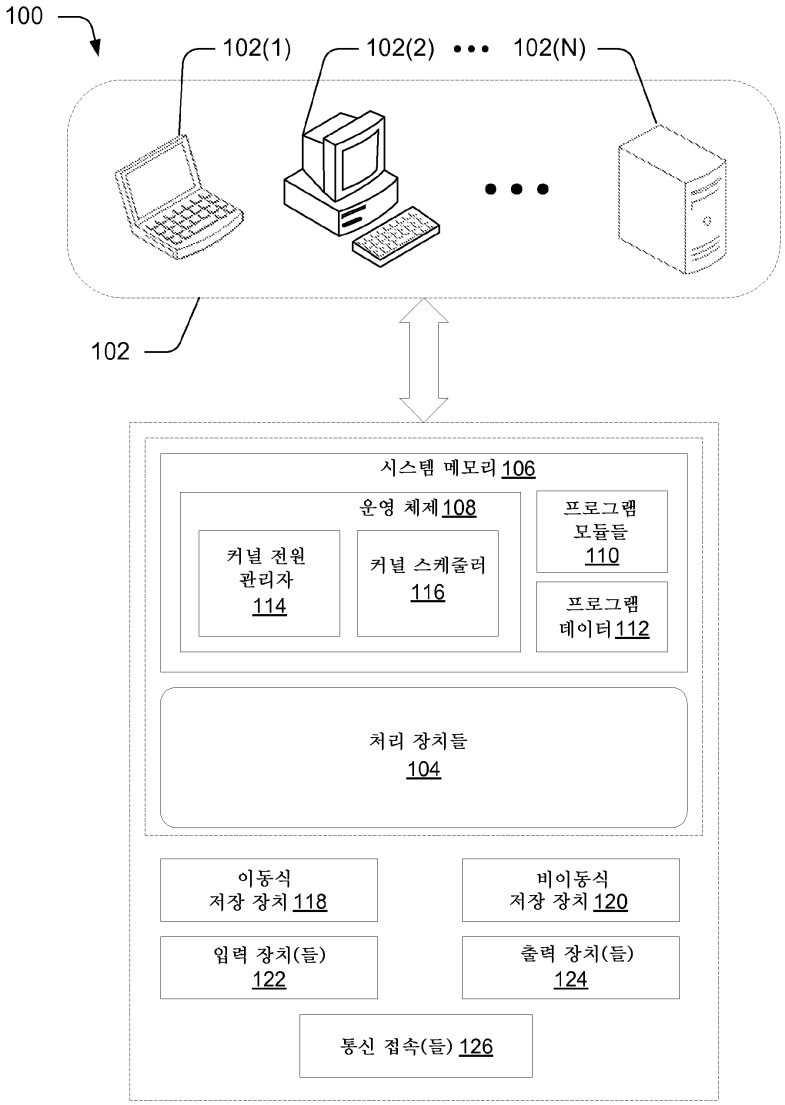
[0071] 단계(714)에서, 코어 블럭의 효율에 관한 도 3a 및 도 3b에 기술된 의존성, 보다 구체적으로는 새로운 코어 블럭에 다른 코어를 파킹하기 전에 코어 블럭을 파킹하는 것 등의 코어 패키지(블럭) 관계들이 고려될 수 있다. 마지막으로, 단계(716)에서, 코어 파킹 고려사항들을 구현할 때, 커널 전원 관리자(114)에 의해 메모리 지역성(memory locality)이 사용될 수 있다. 예를 들어, 2개 이상의 코어들은 공유된 메모리 뱅크(예를 들어, NUMA(non-uniform memory access) 노드)를 가지고 있는 물리적 프로세서 패키지를 공유하는 것과 같은 패키지 관계를 가질 수 있다. 공유된 메모리 뱅크에 의해, 코어들은 공유된 메모리 뱅크를 공유하지 않는 코어들에 비해 감소된 메모리 액세스 시간을 가질 수 있다.

[0072] **결론**

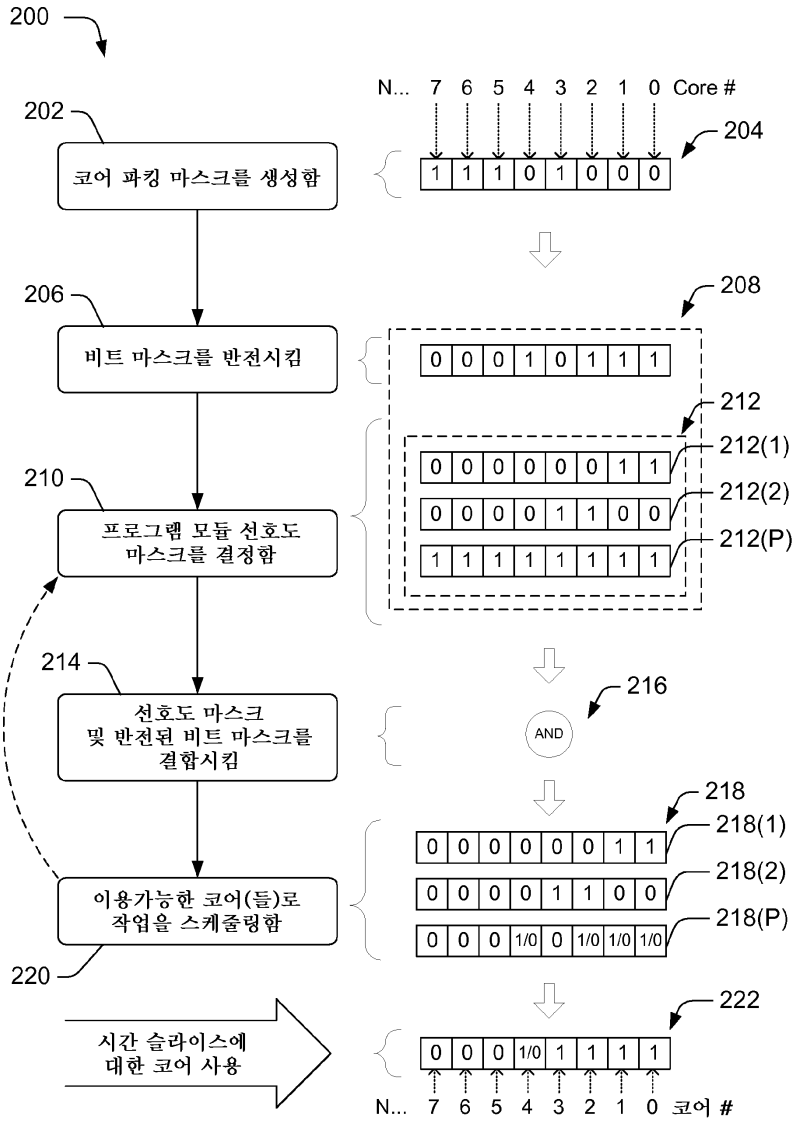
[0073] 이상에서 기술한 기술, 시스템 및 장치는 전력-인식 스레드 스케줄링 및 프로세서들의 동적 사용에 관한 것이다. 기술들, 시스템들 및 장치들이 구조 특징들 및/또는 방법 동작들과 관련하여 기술되어 있지만, 첨부된 특허청구범위가 꼭 기술된 특정의 특징들 또는 동작들로 제한되는 것은 아니라는 것을 잘 알 것이다. 오히려, 특정의 특징들 및 동작들은 이러한 기술들 및 장치들을 구현하는 예시적인 형태로서 개시되어 있다.

도면

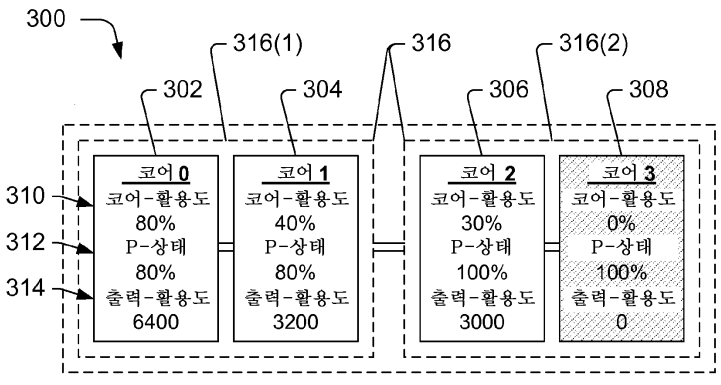
도면1


100
102(1)
102(2)
102(N)
102
시스템 메모리 106
운영 체제 108
커널 전원 관리자 114
커널 스케줄러 116
프로그램 모듈들 110
프로그램 데이터 112
처리 장치들 104
이동식 저장 장치 118
비이동식 저장 장치 120
입력 장치(들) 122
출력 장치(들) 124
통신 접속(들) 126

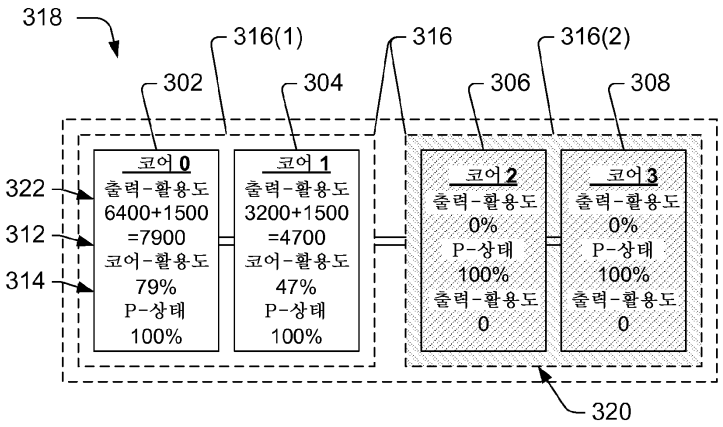
도면2


200
202
코어 파킹 마스크를 생성함
N... 7 6 5 4 3 2 1 0 Core #
204
1 1 1 0 1 0 0 0
206
비트 마스크를 반전시킴
208
0 0 0 1 0 1 1 1
212
212(1)
0 0 0 0 0 0 1 1
212(2)
0 0 0 0 1 1 0 0
212(P)
1 1 1 1 1 1 1 1
210
프로그램 모듈 선호도
마스크를 결정함
214
선호도 마스크
및 반전된 비트 마스크를
결합시킴
216
AND
218
218(1)
0 0 0 0 0 0 1 1
218(2)
0 0 0 0 1 1 0 0
218(P)
0 0 0 1/0 0 1/0 1/0 1/0
220
이용가능한 코어(들)로
작업을 스케줄링함
222
시간 슬라이스에
대한 코어 사용
0 0 0 1/0 1 1 1 1
N... 7 6 5 4 3 2 1 0 코어 #

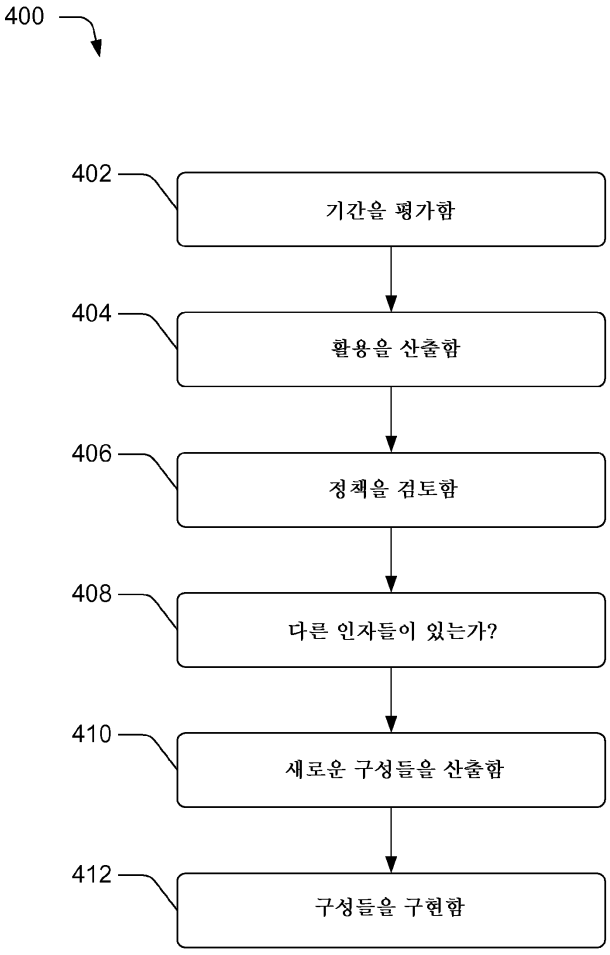
도면3a


300
316(1)
316
316(2)
302
304
306
308
310
312
314
코어 0
코어-활용도
80%
P-상태
80%
출력-활용도
6400
코어 1
코어-활용도
40%
P-상태
80%
출력-활용도
3200
코어 2
코어-활용도
30%
P-상태
100%
출력-활용도
3000
코어 3
코어-활용도
0%
P-상태
100%
출력-활용도
0

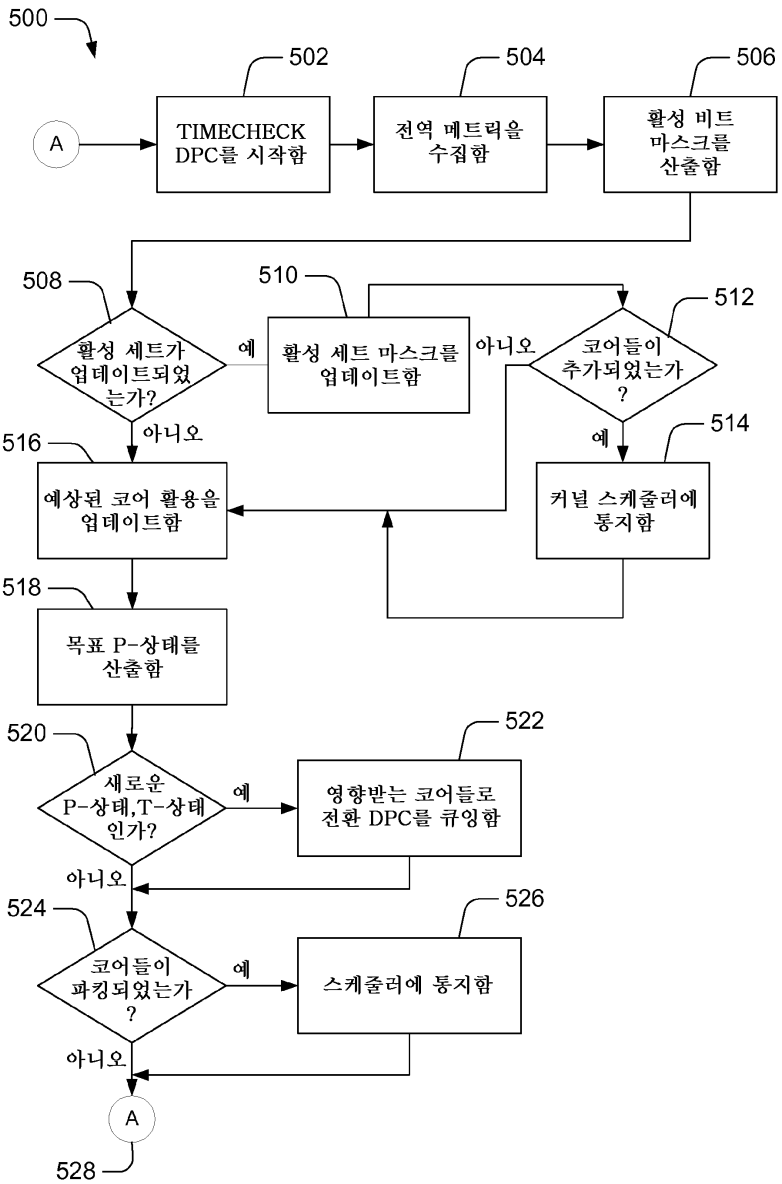
도면3b


318
316(1)
316
316(2)
302
304
306
308
322
312
314
코어 0
출력-활용도
6400+1500
=7900
코어-활용도
79%
P-상태
100%
코어 1
출력-활용도
3200+1500
=4700
코어-활용도
47%
P-상태
100%
코어 2
출력-활용도
0%
P-상태
100%
출력-활용도
0
코어 3
출력-활용도
0%
P-상태
100%
출력-활용도
0
320

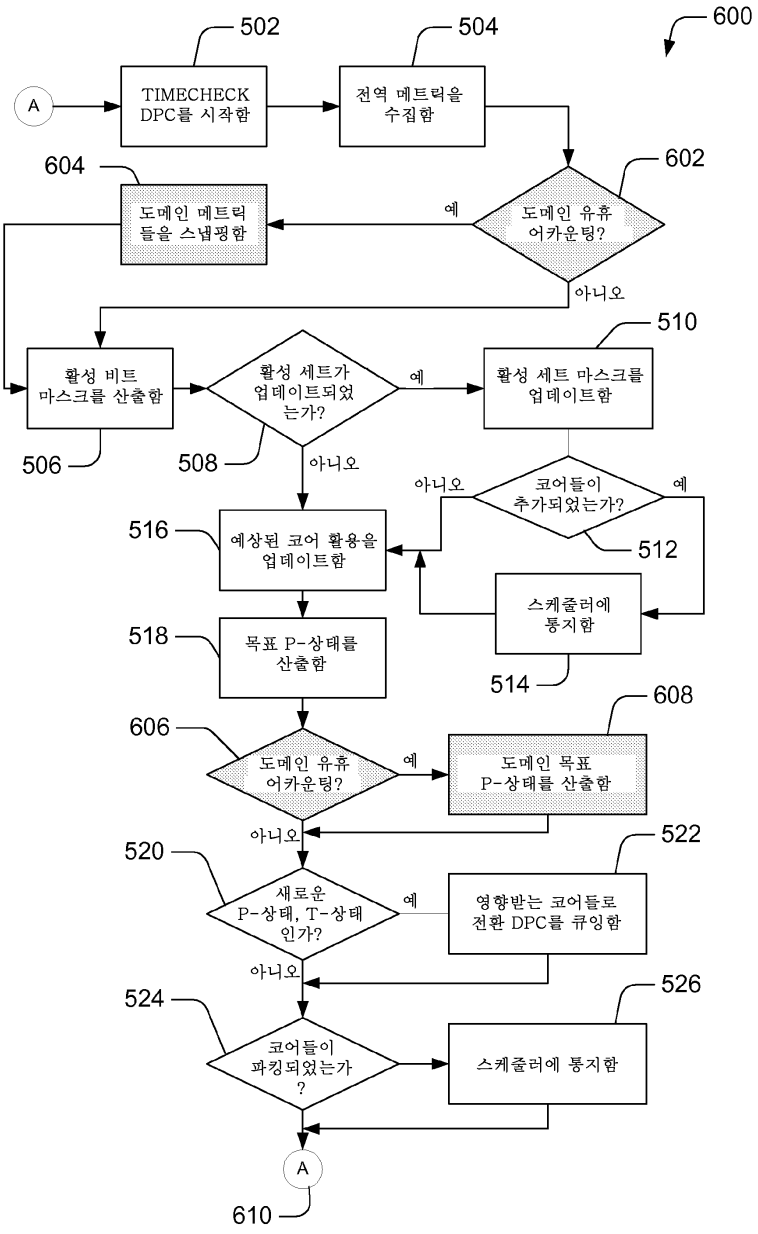
도면4


400
402
기간을 평가함
404
활용을 산출함
406
정책을 검토함
408
다른 인자들이 있는가?
410
새로운 구성들을 산출함
412
구성들을 구현함

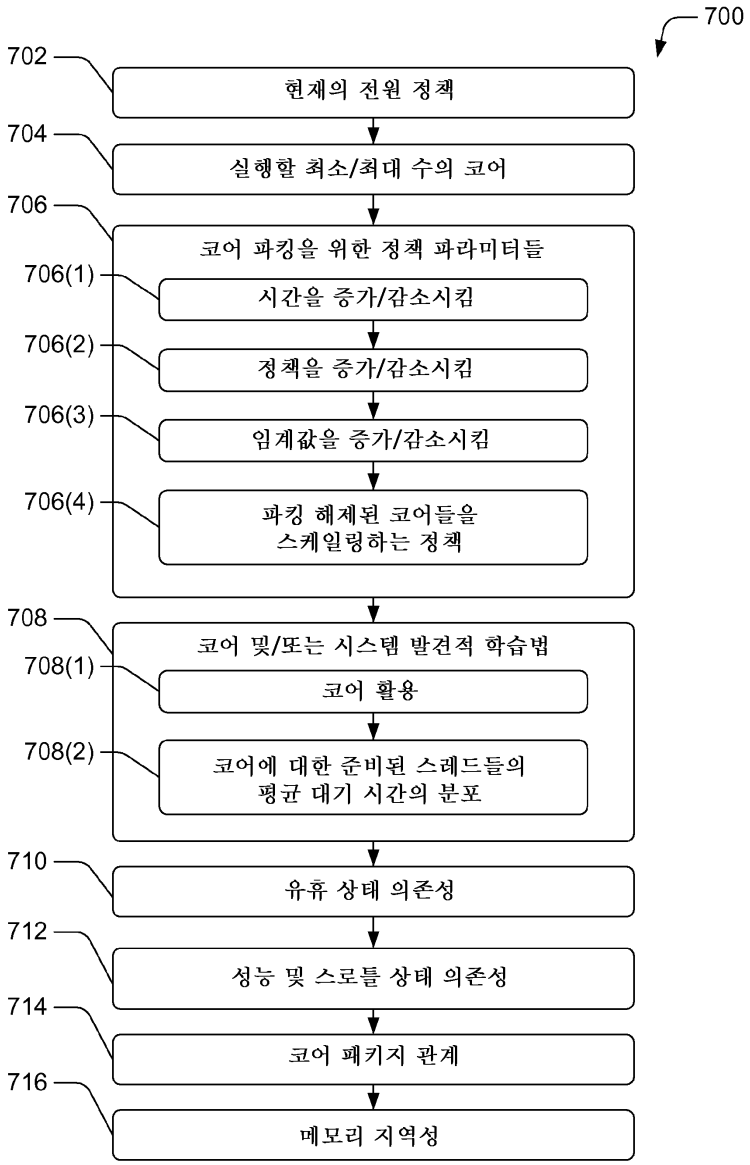
도면5


500
502
TIMECHECK DPC를 시작함
504
전역 메트릭을 수집함
506
활성 비트 마스크를 산출함
508
활성 세트가 업데이트되었는가?
예
510
활성 세트 마스크를 업데이트함
아니오
512
코어들이 추가되었는가 ?
예
514
커널 스케줄러에 통지함
아니오
516
예상된 코어 활용을 업데이트함
518
목표 P-상태를 산출함
520
새로운 P-상태,T-상태인가?
예
522
영향받는 코어들로 전환 DPC를 큐잉함
아니오
524
코어들이 파킹되었는가 ?
예
526
스케줄러에 통지함
아니오
A
528
A

도면6


600
502
504
A
TIMECHECK DPC를 시작함
전역 메트릭을 수집함
602
604
도메인 메트릭들을 스냅핑함
예
도메인 유휴 어카운팅?
아니오
510
활성 비트 마스크를 산출함
활성 세트가 업데이트되었는가?
예
활성 세트 마스크를 업데이트함
506
508
아니오
아니오
코어들이 추가되었는가?
예
512
516
예상된 코어 활용을 업데이트함
스케줄러에 통지함
518
목표 P-상태를 산출함
514
608
606
도메인 유휴 어카운팅?
예
도메인 목표 P-상태를 산출함
아니오
522
520
새로운 P-상태, T-상태인가?
예
영향받는 코어들로 전환 DPC를 큐잉함
아니오
526
524
코어들이 파킹되었는가?
스케줄러에 통지함
A
610

도면7


700
702
현재의 전원 정책
704
실행할 최소/최대 수의 코어
706
코어 파킹을 위한 정책 파라미터들
706(1)
시간을 증가/감소시킴
706(2)
정책을 증가/감소시킴
706(3)
임계값을 증가/감소시킴
706(4)
파킹 해제된 코어들을
스케일링하는 정책
708
코어 및/또는 시스템 발견적 학습법
708(1)
코어 활용
708(2)
코어에 대한 준비된 스레드들의
평균 대기 시간의 분포
710
유휴 상태 의존성
712
성능 및 스로틀 상태 의존성
714
코어 패키지 관계
716
메모리 지역성