(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0077605
(43) 공개일자 2020년06월30일

(51) 국제특허분류(Int. Cl.)
G06F 11/34 (2006.01) *G06F 11/30* (2006.01)
G06F 9/445 (2018.01)
(52) CPC특허분류
G06F 11/3419 (2013.01)
G06F 11/302 (2013.01)
(21) 출원번호 10-2020-7017831(분할)
(22) 출원일자(국제) 2017년03월23일
심사청구일자 없음
(62) 원출원 특허 10-2018-7031432
원출원일자(국제) 2017년03월23일
심사청구일자 2018년10월30일
(85) 번역문제출일자 2020년06월19일
(86) 국제출원번호 PCT/US2017/023771
(87) 국제공개번호 WO 2017/172474
국제공개일자 2017년10월05일
(30) 우선권주장
62/315,315 2016년03월30일 미국(US)
(뒷면에 계속)

(71) 출원인
주식회사 소니 인터랙티브 엔터테인먼트
일본국 도쿄도 미나토쿠 코난 1쵸메 7반 1고
(72) 발명자
심슨, 데이비드
미국 캘리포니아 94404, 산 마테오, 브릿지포인트 파크웨이 2207, 소니 인터랙티브 엔터테인먼트 아메리가 엘엘씨 내
세르니, 마크 에반
미국 캘리포니아 94404, 산 마테오, 브릿지포인트 파크웨이 2207, 소니 인터랙티브 엔터테인먼트 아메리가 엘엘씨 내
(74) 대리인
윤앤리특허법인(유한)

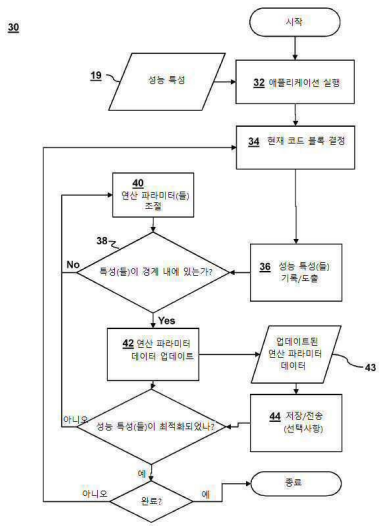
전체 청구항 수 : 총 17 항

(54) 발명의 명칭 **하위 호환성을 위한 애플리케이션-특정 연산 파라미터 도출**

(57) 요 약

레거시 애플리케이션을 새로운 시스템 상에서 실행시킬 때 연산 파라미터의 차후 조절을 위해 레거시 애플리케이션의 성능이 특징화될 수 있다. 레거시 애플리케이션을 레거시 시스템 상에서 실행시킬 때 레거시 애플리케이션에 대한 성능 정보가 기록되거나 도출된다. 레거시 시스템 상에서 실행되는 레거시 애플리케이션에 대한 하나 이상의 성능 특성이 성능 정보를 분석함으로써 결정되며, 상기 하나 이상의 성능 특성은 하나 이상의 핵심 성능 메트릭과 기타 성능 정보를 포함한다. 하나 이상의 핵심 성능 메트릭은 레거시 애플리케이션이 새로운 시스템 상에서 실행될 때 충족되어야 한다. 기타 성능 정보는 레거시 애플리케이션을 새로운 시스템 상에서 실행시킬 때 새로운 시스템의 연산 파라미터의 차후 조절을 위해 유용하다.

대 표 도 - 도1b



(52) CPC특허분류
G06F 11/3452 (2013.01)
G06F 9/44505 (2013.01)
G06F 2201/805 (2013.01)
G06F 2201/865 (2013.01)
G06F 2201/88 (2013.01)

(30) 우선권주장
62/315,345 2016년03월30일 미국(US)
15/466,759 2017년03월22일 미국(US)
15/466,769 2017년03월22일 미국(US)

명 세 서

청구범위

청구항 1

레거시 애플리케이션을 새로운 시스템 상에서 실행시킬 때 연산 파라미터의 후속 조절을 위해 상기 레거시 애플리케이션의 성능을 특징화하는 방법으로서, 상기 방법은:

상기 레거시 애플리케이션이 상기 새로운 시스템 상에서 실행될 때 충족되어야 하는 하나 이상의 성능 메트릭, 및 상기 레거시 애플리케이션을 상기 새로운 시스템 상에서 실행시킬 때 상기 새로운 시스템의 연산 파라미터의 차후 조절에 유용한 기타 성능 정보를 결정하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 2

제1항에 있어서, 상기 하나 이상의 성능 메트릭은 프로그램 실행 동안 이벤트들 간 클록 사이클을 카운팅함으로써 결정되는 것을 특징으로 하는 방법.

청구항 3

제1항에 있어서, 상기 하나 이상의 성능 메트릭은 상기 레거시 애플리케이션의 실행에 관련된 정보를 추적하는 하나 이상의 전용 프로세서 레지스터에 저장된 값으로부터 결정되는 것을 특징으로 하는 방법.

청구항 4

제3항에 있어서, 상기 하나 이상의 전용 프로세서 레지스터에 저장된 상기 정보는 카운터 값을 포함하는 것을 특징으로 하는 방법.

청구항 5

제4항에 있어서, 상기 카운터 값은 프로그램 카운터 값인 것을 특징으로 하는 방법.

청구항 6

제4항에 있어서, 상기 카운터 값은 메모리 사이클용, 산술 논리 장치(ALU: arithmetic logic unit) 사이클용, 또는 픽셀용 카운터의 값인 것을 특징으로 하는 방법.

청구항 7

제1항에 있어서, 상기 하나 이상의 성능 메트릭은 검출된 비지 대기(busy wait)로부터 결정되는 것을 특징으로 하는 방법.

청구항 8

제1항에 있어서, 상기 하나 이상의 핵심 성능 정보 메트릭은 초당 프레임을 포함하는 것을 특징으로 하는 방법.

청구항 9

제1항에 있어서, 상기 하나 이상의 핵심 성능 정보 메트릭은 프로그램 카운터(PC) 범위에 비닝된(binned) 사이클당 명령(IPC: instructions per cycle)을 포함하는 것을 특징으로 하는 방법.

청구항 10

제1항에 있어서, 상기 기타 성능 정보는 간접적으로 도출되는 것을 특징으로 하는 방법.

청구항 11

제9항에 있어서, 상기 기타 성능 정보는 단위 시간당 평균적인 병렬 처리 하드웨어 스케줄링 유닛 점유, 평균적인 병렬 처리 하드웨어 스케줄링 유닛 수명, 메모리 연산을 위한 평균 대기시간(latency), 또는 단위 시간당 타

깃을 렌더링하기 위해 출력되는 픽셀의 카운트를 포함하는 것을 특징으로 하는 방법.

청구항 12

제1항에 있어서, 상기 새로운 시스템의 연산 파라미터를 튜닝하기에 유용한 하나 이상의 성능 정보 값을 결정하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 13

제12항에 있어서, 상기 새로운 시스템의 연산 파라미터를 튜닝하기에 유용한 상기 하나 이상의 성능 정보 값을 결정하는 단계는 핵심 성능 정보 값 및 연산 파라미터에 있어서의 변경들 간에 하나 이상의 상관관계를 결정하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 14

제12항에 있어서, 상기 새로운 시스템의 연산 파라미터를 튜닝하기에 유용한 상기 하나 이상의 성능 정보 값을 결정하는 단계는 다변량 분석을 통해 핵심 성능 정보 값 및 연산 파라미터에 있어서의 변경들 간에 하나 이상의 상관관계를 결정하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 15

제1항에 있어서, 상기 레거시 시스템 및 상기 새로운 시스템은 비디오 게임 시스템인 것을 특징으로 하는 방법.

청구항 16

시스템으로서, 상기 시스템은,

프로세서;

메모리; 및

상기 메모리 내에 구현되는, 프로세서 실행 가능한 명령을 포함하고,

상기 명령은, 레거시 애플리케이션이 새로운 시스템 상에서 실행될 때, 연산 파라미터의 후속 조절을 위해 상기 레거시 애플리케이션의 성능을 특징화하기 위한 방법을 실행하도록 구성되며,

상기 방법은:

레거시 시스템 상에서 상기 레거시 애플리케이션을 실행하는 단계;

상기 레거시 애플리케이션이 상기 새로운 시스템 상에서 실행될 때 충족되어야 하는 하나 이상의 성능 메트릭, 및 상기 레거시 애플리케이션을 상기 새로운 시스템 상에서 실행시킬 때 상기 새로운 시스템의 연산 파라미터의 차후 조절에 유용한 기타 성능 정보를 결정하는 단계를 포함하는 것을 특징으로 하는 시스템.

청구항 17

컴퓨터 판독 가능한 매체 내에 구현되는 컴퓨터 판독 가능한 명령을 갖는 비-일시적인 컴퓨터 판독 가능한 매체로서,

상기 명령은, 레거시 애플리케이션을 새로운 시스템 상에서 실행시킬 때, 연산 파라미터의 후속 조절을 위해 상기 레거시 애플리케이션의 성능을 특징화하는 방법을 구현하도록 구성되고,

상기 방법은:

레거시 시스템 상에서 상기 레거시 애플리케이션을 실행하는 단계;

상기 레거시 애플리케이션이 상기 새로운 시스템 상에서 실행될 때 충족되어야 하는 하나 이상의 성능 메트릭, 및 상기 레거시 애플리케이션을 상기 새로운 시스템 상에서 실행시킬 때 상기 새로운 시스템의 연산 파라미터의 차후 조절에 유용한 기타 성능 정보를 결정하는 단계를 포함하는 것을 특징으로 하는 비-일시적인 컴퓨터 판독 가능한 매체.

발명의 설명

기 술 분 야

[0001] 우선권 주장

[0002] 본 출원은 2016년 03월 30일에 출원된 미국 가특허출원 번호 62/315,315의 이익을 주장하며, 이의 전체 내용이 본 명세서에 참조로서 포함된다. 이 출원은 또한 2016년 03월 30일에 출원된 미국 가특허출원 번호 62/315,345의 이익을 주장하며, 이의 전체 내용이 본 명세서에 참조로서 포함된다. 본 출원은 또한 2017년 03월 22일에 출원된 미국 특허 출원 번호 15/466,759의 이익을 주장하며, 이의 전체 내용이 본 명세서에 참조로서 포함된다. 이 출원은 2017년 03월 22일에 출원된 미국 특허출원 번호 15/466,769의 이익을 또한 주장하며, 이의 전체 내용이 본 명세서에 참조로서 포함된다.

[0003] 기술분야

[0004] 본 발명의 양태는 컴퓨터 시스템 상에서의 컴퓨터 애플리케이션의 실행과 관련된다. 구체적으로, 본 발명의 양태는 컴퓨터 시스템의 이전 버전에 대해 설계된 애플리케이션/타이틀을 위한 하위 호환성을 제공하는 시스템 또는 방법과 관련된다.

배 경 기 술

[0005] 새로운 컴퓨터 아키텍처가 배포될 때 아키텍처의 이전 버전에 대해 써진 애플리케이션이 새로운 아키텍처에서 흠 없이 실행될 수 있는 것이 바람직할 수 있다. 이 능력이 "하위 호환성(backward compatibility)"이라고 지칭된다. 하위 호환성을 구현하는 것은 새로운 호스트 아키텍처 상에서 타깃 레거시 디바이스를 에뮬레이션하여, 새로운 아키텍처가 레거시 디바이스에 대해 써진 프로그램의 명령을 실행시킬 수 있도록 하는 것을 포함한다. 버스, 클록 속도, 프로세서 아키텍처, 캐싱, 표준 등에서의 기술적 진보를 이용하기 위해 컴퓨터 아키텍처가 시간에 따라 변한다. 하나의 컴퓨터 아키텍처가 더 새로운 아키텍처로 교체될 때, 오래된 아키텍처가 레거시 아키텍처(legacy architecture)라고 지칭되는 것이 된다. 이의 개발 과정에서 소프트웨어 애플리케이션, 가령, 네트워크 프로토콜, 사용자 인터페이스, 오디오 프로세싱, 디바이스 드라이버, 그래픽 프로세싱, 메시징, 워드 프로세서, 스프레드시트, 데이터베이스 프로그램, 게임 및 그 밖의 다른 애플리케이션이 레거시 아키텍처에 대해 써진다. 이들이 새로운 아키텍처로 업그레이드되는 경우에도 이러한 레거시 소프트웨어는 사용자에게 여전히 가치가 있다. 따라서 새로운 아키텍처 상에서 레거시 소프트웨어를 실행시킬 필요가 있다.

[0006] 새로운 디바이스와 레거시 디바이스의 하드웨어 구성요소의 성능의 차이가 새로운 디바이스 상에서 동기화 에러를 초래할 수 있고, 이로 인해 새로운 디바이스 아키텍처 상에서 실행될 때 레거시 애플리케이션이 충돌하거나 잘못된 출력을 생성할 수 있다. 이러한 성능 차이는 가령 새로운 디바이스와 레거시 디바이스 간 하드웨어 아키텍처의 차이로부터 발생할 수 있다. 이러한 맥락에서 본 발명의 양태가 발생한다.

발명의 내용

도면의 간단한 설명

[0007] 도 1a는 본 발명의 양태에 따라 애플리케이션-특정 연산 파라미터의 도출을 도시하는 흐름도이다.

도 1b는 본 발명의 양태에 따라 애플리케이션-특정 연산 파라미터의 실시간 조절을 도시하는 흐름도이다.

도 2a는 본 발명의 양태에 따라 하위 호환성 모드로 동작하도록 구성될 수 있는 중앙 처리 장치(CPU) 코어의 예시를 도시하는 블록도이다.

도 2b는 본 발명의 양태에 따르는 CPU에 대한 가능한 멀티-코어 아키텍처의 일례를 도시하는 블록도이다.

도 3은 본 발명의 양태에 따라 하위 호환성 모드로 동작하도록 구성된 CPU를 갖는 디바이스의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0008] 도입

[0009] 새로운 디바이스 상에서 레거시 애플리케이션을 실행시킬 때 하드웨어 거동의 차이로 인해 발생하는 문제를 해결하기 위해, 새로운 하드웨어가 레거시 애플리케이션을 실행시키도록 튜닝될 수 있다.

[0010] 시험 스테이지 동안 레거시 애플리케이션이 레거시 아키텍처를 갖는 레거시 디바이스 상에서 실행되고 성능 정

보가 수집된다. 성능 정보의 예시는 단위 시간당 ALU 명령 또는 메모리 연산의 수, 및 평균적인 병렬 처리 하드웨어 스케줄링 유닛(가령, 웨이브프론트(wavefront)) 점유시간 또는 수명을 포함한다. 레거시 디바이스 상에서 게임과 애플리케이션을 실행시키고 카운터를 판독함으로써 성능 정보(ALU 및 메모리 연산)가 직접 측정될 수 있다. 대안으로, 성능 정보가 측정 프로세스의 일부로서 이러한 카운터 또는 그 밖의 다른 데이터 출력을 읽는 것으로부터 도출될 수 있다. 이러한 도출의 예를 들면, 평균 웨이브프론트 점유 시간 및 수명이 웨이브프론트가 시작하고 종료할 때의 측정으로부터 도출될 수 있다. 특정 애플리케이션, 가령, 특정 비디오 게임에 대한 조합된 성능 데이터가 본 명세서에서 상기 애플리케이션에 대한 성능 특성이라고 지칭된다. 시험 스테이지에서 애플리케이션에 대해 결정된 성능 특성이 새로운 시스템 상에서 동일한 애플리케이션을 실행시키기 위해 사용되는 기준으로서 사용되어, 하위 호환성을 보장할 수 있다.

[0011] 새로운 디바이스의 연산 파라미터를 튜닝함으로써 새로운 디바이스 상에서의 애플리케이션의 성능이 레거시 디바이스 상에서의 동일한 애플리케이션의 성능에 비슷하게 매칭될 수 있다. 연산 파라미터의 예시로는, 새로운 디바이스의 클록 주파수, 가용한 범용 레지스터(GPR)의 수, 명령 개시율(instruction launch rate) 등이 있다. 애플리케이션은 애플리케이션-특정 성능 특성을 조절하기 위해 이의 연산 파라미터를 튜닝하면서 새로운 시스템 상에서 반복적으로 실행될 수 있다. 새로운 시스템에 대한 충분한 횟수의 시험 후에, 연산 파라미터가 변함에 따라 새로운 시스템 상의 애플리케이션의 성능 특성이 어떻게 수렴하는지를 분석할 수 있다. 수렴 분석을 기초로 새로운 연산 파라미터 세트가 생성될 수 있다. 이 프로세스는 연산 파라미터가 새로운 시스템 상의 애플리케이션에 대해 최적으로 설정된다. 더 최적화하기 위해, 오류를 일으키지 않고 애플리케이션이 새로운 하드웨어 상에서 더 빠르게 실행될 수 있는지 여부를 알기 위해 새로운 하드웨어의 실행을 조절할 수 있다.

[0012] 상세한 설명

[0013] 애플리케이션-특정 성능 특성 결정

[0014] 도 1a는 애플리케이션을 레거시 시스템 상에서 실행시키는 시험 단계 동안 애플리케이션-특정 연산 파라미터를 도출하기 위한 방법(10)을 도시한다. 애플리케이션이 레거시 시스템 상에서 실행되고(12) 각각의 코드 블록에 대해(14) 성능 정보가 기록되거나 도출된다(16). 애플리케이션을 실행시키는 것은, 가령, 캡처를 로딩하고 입력 없이 이를 실행시키거나, 게임의 특정 영역을 통해 플레이하는 것을 포함할 수 있다. 성능 정보는 핵심 성능 메트릭(Key Performance Metrics) 및 기타 성능 정보를 포함한다. 핵심 성능 메트릭은 애플리케이션이 새로운 시스템 상에서 실행될 때 가장 중요한 성능 정보의 서브세트를 일컫는다. 핵심 성능 메트릭은 애플리케이션이 새로운 하드웨어 상에서 실행될 때 충족되어야 한다. 핵심 성능 메트릭의 비제한적 예시로는 초당 프레임(가령, 비디오 집중 애플리케이션, 가령, 비디오 게임의 경우) 및 프로그램 카운터(PC) 범위에 비닝된 사이클당 명령(IPC: instructions per cycle)이 있다.

[0015] 기타 성능 정보의 비제한적 예시로는 PC 블록 레지던스, 단위 시간(CPU 및 GPU) 당 발행되는 산술 논리 장치(ALU: arithmetic logic unit) 명령의 개수, 단위 시간(CPU 및 GPU) 당 발행되는 메모리 연산의 수, 단위 시간당 평균 병렬 처리 하드웨어 스케줄링 유닛(가령, 웨이브프론트, 와프(warp), 또는 벡터 폭) 점유시간, 평균 병렬 처리 하드웨어 스케줄링 유닛 수명, 메모리 연산을 위한 평균 대기시간(latency), 단위 시간당 타깃을 렌더링하기 위해 출력되는 픽셀의 카운트, 및 한 프레임 동안 활성인 총 사이클(ALU 카운트가 이의 특정 예시임)이 있다.

[0016] 성능 정보가 카운터로부터 직접 판독되거나 이러한 값 및 그 밖의 다른 정보로부터 도출된 값, 가령, 프로그램 실행 동안 이벤트들 간 클록 사이클을 카운팅함으로써 도출된 값을 포함할 수 있다. 성능 정보는 (18)에서 더 분석되고 선택된 성능 정보는 조합되어 성능 특성의 세트(19)를 결정하고, 이는 저장되거나 전송될 수 있다(20).

[0017] 특정 성능 정보 값은 애플리케이션의 실행과 관련된 정보를 계속 주지하는 전용 프로세서 레지스터에 저장될 수 있다. 이러한 값의 비제한적 예시로는, 카운터 값, 가령, 프로그램 카운터 및 메모리 사이클, 산술 논리 유닛(ALU) 사이클을 위한 카운터, 및 픽셀 등이 있다. Intel x86 및 Itanium 마이크로프로세서에서 명령 포인터(IP)라고도 불리고, 때때로 명령 어드레스 레지스터(IAR), 또는 명령 카운터라고도 불리는 프로그램 카운터(PC)가 컴퓨터가 이의 프로그램 시퀀스에서 있는 곳을 가리키는 프로세서 레지스터이다.

[0018] 앞서 언급된 바와 같이, 특정 기타 성능 정보, 가령, 단위 시간 당 평균 병렬 처리 하드웨어 스케줄링 유닛(가령, 웨이브프론트, 와프, 또는 벡터 폭) 점유시간, 평균 병렬 처리 하드웨어 스케줄링 유닛 수명, 메모리 연산에 대한 평균 대기시간, 단위 시간당 타깃을 렌더링하기 위해 출력되는 픽셀의 카운트가 간접적으로 도출될 수

있다. 비제한적 예를 들어, 초기 프로그램 카운터 값과 최종 프로그램 카운터 값 간 차이를 초기 프로그램 카운터 값과 최종 프로그램 카운터 값 사이의 클록 사이클의 수로 나눔으로써, IPC(instructions per cycle)의 수가 도출될 수 있다. 또한 평균 병렬 처리 하드웨어 스케줄링 유닛 수명을 결정하는 것은 이러한 스케줄링 유닛의 개시(launch)와 완료를 검출하고, 이들 사이의 클록 사이클을 카운팅하는 것을 포함할 수 있다. 마찬가지로, 병렬 처리 하드웨어 스케줄링 유닛의 단위 시간당 평균 점유시간을 결정하는 것은 특정 시간 윈도 동안 개시 및 완료를 기록하는 문제이고, 상기 시간 윈도 내 임의의 특정 시점에서 평균적으로 얼마나 많이 실행될 수 있는지를 결정하는 것을 포함한다.

[0019] 본 명세서에서 용어 "병렬 처리 스케줄링 유닛"이 상이한 프로세서 하드웨어의 제조자에 의해 사용되는 몇 가지 상이한 용어를 포함하는 일반적인 용어로서 사용되어, 코드의 병렬 처리를 위한 가장 작은 실행 단위의 개념을 기술할 수 있다. 예를 들어, GPU의 맥락에서, 병렬 처리 스레드가 가장 기본적인 스케줄링 유닛으로서 (NVIDIA 하드웨어의 경우) "와프(warp)" 또는 (AMD 하드웨어의 경우) "웨이브프론트(wavefront)"라고 일컬어지는 것으로 묶이며, 차이는 함께 그룹지어지는 스레드의 수이다. 또 다른 동등한 정의는, "최소 실행 단위 코드가 연산될 수 있음" 또는 "이의 모든 스레드에 걸쳐 단일 명령에 의해 동시에 이뤄지는 처리의 단위" 또는 "SIMD 방식으로 처리되는 데이터의 최소 크기"를 포함한다. CPU 하드웨어의 경우, 병렬처리의 가장 기본적인 레벨의 개념이 종종 (가령, Intel 및 AMD 프로세서 상에서 SSE 명령을 이용할 때) "벡터 폭(vector width)"이라고 일컬어진다. 단순성을 위해, 본 명세서에서 용어 "웨이브프론트"가 "병렬 처리 스케줄링 유닛"을 대체하여 사용될 것이다. 웨이브프론트 내 모든 스레드가 락-스텝(lock-step)에서 동일한 명령을 실행하며, 유일한 차이는 상기 명령에 의해 연산되는 데이터이다.

[0020] 그 밖의 다른 연산 정보가 연산 레지스터 값으로부터 복수의 상이한 방식으로 도출될 수 있다. 예를 들어, 프로그램이 실행될 때 실행된 명령의 총 수를 포함하는 카운터를 샘플링함으로써 IPC가 도출될 수 있다. 예를 들어, 이 카운터는 N회 사이클마다 샘플링될 수 있다. 초기 총 명령 실행 값(TIE_i) 및 N회 사이클 후의 다음 값(TIE_{i+N})에서 (TIE_{i+N}-TIE_i)/N으로부터로부터 IPC 값이 도출될 수 있다. 실시할 때, 애플리케이션의 특정 섹션(가령, 코드의 블록)에 대한 IPC 값이 특정 섹션에 대해 PC 범위에 의해 비닝될 수 있다. 덧붙여, 애플리케이션 내 각각의 PC 범위가 상이한 가능한 거동 및 이에 따라 상이한 IPC 값을 가질 수 있다. 따라서 예를 들어 코드 블록 수에 의해, IPC 값을 프로그램 코드의 식별된 섹션과 연관시키는 것이 유용하다.

[0021] 일반적으로 GPU가 복수의 코드 조각을 동시에 실행시키기 때문에, 현재 실행 중인 애플리케이션 코드의 블록을 지칭하는 PC 블록 레지던스가 GPU보다 CPU에 대해 더 관련성 있을 수 있다. N회 사이클마다 PC를 샘플링하고 샘플이 동일한 코드 블록 내에 속하는 횟수를 카운팅함으로써 PC 블록 레지던스가 도출될 수 있다.

[0022] ALU 또는 메모리 연산을 발행하는 주파수가, 이러한 연산의 발행을 검출하고 특정 시간 윈도에 걸쳐 발행된 이러한 연산의 수를 카운팅함으로써 도출될 수 있다. 마찬가지로, 단위 시간당 타깃을 렌더링하기 위해 출력되는 픽셀의 카운트가 특정 시간 윈도에 걸쳐 출력되는 픽셀을 카운팅함으로써 도출될 수 있다. 캐시 읽기/쓰기 및/또는 메모리 액세스 명령의 발행 및 완료를 검출하고 발행과 완료 사이의 클록 사이클을 카운팅함으로써, 대기시간, 가령, 캐시 대기시간 또는 메모리 연산 대기시간이 도출될 수 있다.

[0023] (16)에서의 성능 정보 기록/도출이 비지 대기(busy wait)를 검출하는 것을 포함할 수 있다. 일반적으로 비지 대기는 짧은 루프로 구현된다. 카운터 관점에서, 이는 PC가 매우 작은 범위 내에 머무르는 것(그리고 반복하는 것)처럼 보일 것이며 루프를 통과할 때 매번 발생하는 일종의 메모리 읽기 또는 IO 읽기 연산이 존재할 것이다. IPC가 루프 때문에 더 높을 수 있지만, 실제로는 루프 내 시간이 메모리 또는 IO 연산의 결과가 반환되도록 대기하는 것에 의해 점유될 것이기 때문에 IPC가 낮을 가능성이 높을 것이다. 비지 대기는 PC가 매우 작은 범위 내에 머무르고 시간이 메모리 또는 IO 연산이 완료되기를 대기하는 것에 의해 점유되는 때를 찾음으로써 검출될 수 있다. 비지 대기는 IPC 및 기타 성능 정보 측정치를 왜곡시키는 경향이 있다. 비지 대기에 걸리는 시간이 예측될 수 없기 때문에, 비지 대기 동안 이뤄진 측정치는 (18)의 성능 특성 결정 프로세스의 일부로서 성능 정보에서 제거될 수 있다. 이로써, 연산 파라미터를 조정하는 다음 프로세스가 비지 대기의 존재에 의해 영향받지 않을 것이다.

[0024] 일반적으로 (18)에서 기록된 또는 도출된 성능 정보를 분석하는 것은 성능 정보를 일반적으로 실행 동안의 애플리케이션의 거동을 특징화하는 성능 특성(19)의 유용 세트로 좁히는 것을 포함한다. 성능 특성(19)의 비제한적 예를 들면, 하나 이상의 핵심 성능 메트릭 및 이하에서 설명될 차후 연산 파라미터를 결정할 때 유용한 기타 성능 정보가 있다.

[0025] 성능 특성 결정 스테이지(18)가, 가령, 특정 연산 파라미터의 변경에 응답하여 많은 상이한 성능 정보 값이 변할 수 있음에 따라 다변량 분석을 통해 핵심 성능 정보 값 및 연산 파라미터의 변경들 간 상관관계를 결정함으로써, 연산 파라미터를 튜닝하는 데 어느 성능 정보 값이 유용한지를 결정할 수 있다.

[0026] 파라미터 조절 프로세스

[0027] 도 1b는 애플리케이션이 새로운 시스템 상에서 실행될 때 연산 파라미터를 최적화하기 위해 레거시 애플리케이션의 실행을 조절하기 위한 방법(30)을 기재한다. 새로운 시스템은 레거시 애플리케이션(32)을 실행시킬 때 실시간으로 하나 이상의 연산 파라미터를 조절하도록 성능 특성(19)을 이용할 수 있다. 앞서 언급된 바와 같이, 각각의 코드 블록(34)에 대해 성능 정보가 도출된다(36). 하나 이상의 핵심 성능 메트릭이 경계 내에 있지 않는 경우(38), 경계 내에 있을 때까지 하나 이상의 연산 파라미터가 반복적으로 조절될 수 있다(40). (38)에서 핵심 성능 메트릭이 경계 내에 있으면, 연산 파라미터가 업데이트될 수 있고(42) 추가 조절(40)에 의해 최적화될 수 있다. 업데이트/최적화된 연산 파라미터 데이터(43)가 저장되거나 전송될 수 있다(44).

[0028] 일반적으로 용어 "연산 파라미터"는 핵심 성능 메트릭을 포함하는 성능 정보에 영향을 미치도록 조절될 수 있는 새로운 시스템 상에서의 애플리케이션의 실행의 양태를 지칭한다. 연산 파라미터의 예시로는, 비제한적으로: 클록 주파수, 가령, CPU, GPU 또는 메모리에 대한 클록 주파수, 명령의 개시율(launch rate), ALU 및/또는 메모리 연산의 개시율, 자원, 가령, 범용 레지스터(GPR), 웨이브프론트 슬롯, 읽기 및 저장 큐 크기 등, 기능 비활성화(feature disablement), 캐시 파라미터(가령, 캐시 크기, 웨이의 수, 뱅크의 수 등), 웨이브프론트 개시율, 렌더 백엔드(render backend)로부터의 픽셀 출력율, 메모리 연산 정지(memory operation stalling)가 있을 수 있다.

[0029] 알고리즘 매칭이 새로운 시스템 아키텍처에 대해 써진 새롭고 개선된 알고리즘 대신 레거시 시스템 아키텍처로부터의 알고리즘을 이용해 새로운 시스템 상에 특정 연산을 수행하는 것을 지칭한다. 이러한 알고리즘 매칭의 예로는 레거시 시스템이 새로운 시스템 상에서 브랜치 예측을 수행하기 위해 브랜치 예측자(branch predictor)를 이용하는 것이 있을 것이다. 이 예시에서, 알고리즘 매칭 파라미터가 레거시 알고리즘에서 사용되는 파라미터를 이용할 것이다.

[0030] 그 밖의 다른 연산 파라미터가 또한 자원 제한과 관련된 파라미터(가령, 본 명세서에서 참조로서 포함되는 2015년 07월 27일에 출원된 미국 특허 출원 번호 14/810,361에 기재된 바 있음) 및 알고리즘 매칭, 기능 비활성화, 및 매칭 대기시간 또는 처리율과 관련된 파라미터(가령, 본 명세서에서 참조로서 포함되는 2015년 07월 27일에 출원된 미국 특허 출원 번호 14/810,334에 기재된 바 있음)를 포함할 수 있다.

[0031] (40)에서의 연산 파라미터의 조절은 단순하게 가령, 새로운 하드웨어 상에서 레거시 하드웨어와 동일한 개수의 범용 레지스터(GPR)를 설정하는 것일 수 있다. 대안으로, 새로운 하드웨어가 특정 연산에 대해 레거시 알고리즘을 이용하거나 새로운 하드웨어의 기능이 레거시 애플리케이션의 동작에 대해 비활성화될 수 있다. 새로운 시스템 상에서의 대기시간을 레거시 하드웨어 대기시간에 매칭시키도록 실행이 조절될 수 있다.

[0032] 연산 파라미터의 조절은 레거시 하드웨어와 새로운 하드웨어 간 아키텍처 차이 때문에 더 복잡할 수 있다. 일부 경우, 가령, 약간 더 많은 GPR을 설정함으로써 본래의 하드웨어보다 새로운 하드웨어에 더 많은 자원이 할당될 수 있다.

[0033] 이하의 표 1은 애플리케이션-특정 연산 파라미터, 이들을 도출하는 법, 및 이들을 조절하는 법의 비제한적 일부 예시를 나열한다.

표 1

성능 정보	측정/도출 방법	연산 파라미터를 조절함으로써 변화시키는 방법
사이클당 명령(IPC: Instructions per Cycle)	프레임 동안 카운터로 발행되는 모든 명령 측정 (CPU 및 GPU)	명령 개시율을 조절(하드웨어에서 할 필요 있음)
ALU 연산 발행 주파수	프레임 동안 카운터로 ALU 사이클 측정 (CPU 및 GPU)	매 N회 사이클마다 ALU 연산을 허용하지 않음(하드웨어에서 할 필요 있음)
메모리 연산 발행 주파수	프레임 동안 카운터로 메모리 사이클 측정 (CPU 및 GPU)	매 N회 사이클마다 메모리 연산을 허용하지 않음(하드웨어에서 할 필요 있음)
단위 시간당 평균 웨이브프론트 점유시간	카운터로 이를 샘플링하거나 캡처를 리플레이하고 웨이브프론트가 시작되고 종료될 때를 검토. (GPU)	선택적으로 GPR을 할당 (소프트웨어에서 할 수 있음) 또는 웨이브프론트 개시율을 스로틀링(하드웨어에서 할 필요 있음)
평균 웨이브프론트 수명	캡처를 리플레이하고 웨이브프론트가 시작하고 종료될 때를 검토. (GPU)	선택적으로 GPR을 할당(소프트웨어에서 할 수 있음) 또는 웨이브프론트 개시율을 스로틀링(하드웨어에서 할 필요 있음)
단위 시간당 타깃을 렌더링하기 위해 출력되는 픽셀	기존 카운터로 단위 시간당 픽셀 카운터를 검토. (GPU)	렌더 백엔드로부터 출력율(# 픽셀)을 스로틀링 (이들은 그래픽 파이프라인의 하단에 타깃을 렌더링하도록 출력되는 픽셀을 씀) (하드웨어에서 할 필요 있음)
평균 메모리 연산 대기시간	메모리 명령이 발행될 때를 결정하고 실행될 때 이들 사이의 클록 사이클을 카운트 (CPU 및 GPU)	메모리 연산이 완료되지 않게 중단 (하드웨어에서 할 필요 있음) 또는 상이한 율로 클록을 실행 (소프트웨어에서 할 수 있음)
PC 블록 레지던스	프로그램 카운터 판독(CPU)	연산 파라미터 값이 블록 레지던스와 강력하게 상관되는 경우 연산 파라미터를 조절할 때 유용한 정보

[0034]

[0035] 최종 업데이트된 연산 파라미터(43)가 애플리케이션-특정 성능 정보의 각각의 아이템에 대한 하한을 포함하는데, 상기 하한을 초과할 때 레거시 하드웨어 상에서 실행되는 레거시 애플리케이션의 성능 메트릭이 일관되게 충족된다. 새로운 하드웨어에 대한 추가 시험에 의해, 애플리케이션-특정 성능 정보의 각각의 아이템이 상한을 더 포함할 수 있으며, 상기 상한을 초과할 때 레거시 애플리케이션은 적절하게 작동하지 않거나 레거시 애플리케이션의 핵심 성능 메트릭이 새로운 시스템 상에서 더는 충족되지 않는다. 애플리케이션-특정 성능 정보가 이하의 표 2의 정보에 대응할 수 있다.

표 2

코드 블록	성능 정보	최소	최대
CB1	사이클당 명령(IPC)	IPC1최소	IPC1최대
CB1	ALU 연산 발행 주파수	AOF1최소	AOF1최대
CB1	메모리 연산 발행 주파수	MOF1최소	MOF1최대
CB1	단위 시간당 평균 웨이브프론트 점유시간	AWO1최소	AWO1최대
CB1	단위 시간당 타깃을 렌더링하기 위해 출력되는 픽셀	PORT1최소	PORT1최대
CB1	평균 메모리 연산 대기시간	AML1최소	AML1최대
CB1	PC 블록 레지던스	PB1최소	PB1최대
CB2	사이클당 명령(IPC)	IPC2최소	IPC2최대
CB2	ALU 연산 발행 주파수	AOF2최소	AOF2최대
CB2	메모리 연산 발행 주파수	MOF2최소	MOF2최대
CB2	단위 시간당 평균 웨이브프론트 점유시간	AWO2최소	AWO2최대
CB2	단위 시간당 타깃을 렌더링하기 위해 출력되는 픽셀	PORT2최소	PORT2최대
CB2	평균 메모리 연산 대기시간	AML2최소	AML2최대
CB2	PC 블록 레지던스	PB2최소	PB2최대

[0036]

[0037] 표 2에 나타난 예시에서, 레거시 프로그램에서의 각각의 코드 블록에 대한 성능 정보에 대한 상한 및 하한의 세

트가 있다. 이 정보는 새로운 하드웨어 상에서의 레거시 게임의 후속 동작에서 사용될 수 있다. 이러한 후속 동작이 도 1b에 기재된 바와 같이 진행될 수 있으며, 이때 새로운 하드웨어가 성능 정보를 최솟값과 최댓값 사이로 유지하도록 실행을 조절한다.

[0038] 연산 파라미터 도출 및 조절이 도 2a-2b 및 도 3에 나타난 하드웨어의 기능과 관련될 수 있다. 도 2a는 CPU 코어(100)의 일반화된 아키텍처를 도시한다. 일반적으로 CPU 코어(100)는 브랜치가 취해질지 여부를 예측하려 시도하고 또한 (브랜치가 취해지는 경우) 상기 브랜치의 도착지 주소를 예측하려 시도하는 브랜치 예측 유닛(102)을 포함한다. 이들 예측이 올바른 범위까지, 추측에 근거하여 실행된 코드의 효율이 증가될 것이며, 따라서 정확도가 높은 브랜치 예측이 매우 바람직할 수 있다. 브랜치 예측 유닛(102)은, 고도로 특수화된 서브-유닛, 가령, 서브루틴으로부터의 복귀 주소를 추적하는 복귀 주소 스택(104), 간접 브랜치의 도착지를 추적하는 간접 타깃 어레이(106), 및 브랜치의 과거 히스토리를 추적하여 최종 주소를 더 정확히 예측하는 브랜치 타깃 버퍼(108) 및 이의 연관된 예측 로직을 포함할 수 있다.

[0039] 일반적으로 CPU 코어(100)는, 명령 인출 유닛(112), 명령 바이트 버퍼(114) 및 명령 디코딩 유닛(116)을 포함하는 명령 인출 및 디코딩 유닛(110)을 포함한다. 상기 CPU 코어(100)는 일반적으로, 복수의 명령 관련 캐시 및 명령 변환 색인 버퍼(ITLB)(120)를 더 포함한다. 이들은 가상 주소를 물리 주소 변환 정보, 가령, 페이지 테이블 항목, 페이지 디렉토리 항목 등에 캐싱하는 ITLB 캐시 계층구조(124)를 포함할 수 있다. 이 정보는 명령의 가상 주소를 물리 주소로 변환하여, 명령 인출 유닛(112)이 캐시 계층구조로부터 명령을 로딩할 수 있도록 하는 데 사용된다. 비제한적 예를 들면, 코어 내에 상주하는 레벨 1 명령 캐시(L1 I-Cache)(122)와 CPU 코어(100) 외부의 다른 캐시 레벨(176)을 포함하는 캐시 계층구조에 따라 프로그램 명령이 캐싱될 수 있고, 명령의 물리 주소를 이용해, 이들 캐시가 프로그램 명령에 대해 우선 검색된다. 명령이 발견되지 않는 경우, 이들은 시스템 메모리(101)로부터 로딩된다. 아키텍처에 따라, 아래에서 기재된 바와 같이 디코딩된 명령을 포함하는 마이크로-연산 캐시(micro-op cache)(126)가 존재할 수 있다.

[0040] 프로그램 명령이 인출되면, 이들은 일반적으로 명령 인출 및 디코딩 유닛(110)에 의한 처리를 대기하는 명령 바이트 버퍼(114) 내에 위치한다. 디코딩이 매우 복잡한 프로세스일 수 있으며, 각각의 사이클마다 복수의 명령을 디코딩하는 것이 어렵고 하나의 사이클에서 디코딩될 수 있는 명령의 수를 제한하는 제한을 명령의 명령 정렬 또는 유형에 적용할 수 있다. 디코딩된 명령은, 아키텍처에 따라 (새로운 CPU 상에 존재하는 경우) 마이크로-연산 캐시(126) 내에 위치하여, 프로그램 명령의 후속 사용에서 디코딩 스테이지가 우회될 수 있다.

[0041] 일반적으로 디코딩된 명령은 분배 및 스케줄링을 위한 다른 유닛(130)으로 전달된다. 이들 유닛은 CPU 파이프라인의 나머지 부분을 통해 명령의 상태를 추적하기 위해 은퇴 큐(132)를 이용할 수 있다. 또한 다수의 CPU 아키텍처 상에서 이용 가능한 범용 및 SIMD 레지스터의 제한된 수 때문에, 논리적 레지스터(아키텍처 레지스터라고도 알려짐)가 명령 실행 스트림에서 발생할 때 물리 레지스터(140)가 이들을 나타내도록 할당되는 레지스터 리네이밍(register renaming)이 수행될 수 있다. 물리 레지스터(140)는 SIMD(Single Instruction Multiple Data) 레지스터 뱅크(142) 및 GP(General Purpose) 레지스터 뱅크(144)를 포함할 수 있고, 이들의 크기는 특정 CPU 아키텍처 상에서 이용 가능한 논리 레지스터의 수보다 훨씬 크고 따라서 성능이 상당히 증가될 수 있다. 레지스터 리네이밍(134)이 수행된 후, 일반적으로 명령이 스케줄링 큐(136) 내에 위치하며, 이로부터 실행 유닛(150)에 의해 실행되기 위해 각각의 사이클마다 (종속성을 기초로) 복수의 명령이 선택될 수 있다.

[0042] 일반적으로 실행 유닛(150)은 SIMD 레지스터 뱅크(142)에 포함되는 128-비트 이상의 SIMD 레지스터에 포함되는 복수의 데이터 필드에 대한 복수의 병렬 연산을 수행하는 SIMD 파이프(152), GP 레지스터 뱅크(144)에 포함된 GPR에 대한 복수의 논리, 산술 및 다양한 연산을 수행하는 산술 논리 유닛(ALU)(154), 및 메모리가 저장 또는 로딩되어야 하는 주소를 계산하는 주소 생성 유닛(AGU)(156)을 포함한다. 각각의 유형의 실행 유닛의 복수의 인스턴스가 존재할 수 있으며, 인스턴스는 상이한 기능을 가질 수 있는데, 가령, 특정 SIMD 파이프(152)가 부동 소수점 곱셈 연산은 수행할 수 있지만 부동 소수점 덧셈 연산은 수행할 수 없을 수 있다.

[0043] 일반적으로 저장 및 로딩이 저장 큐(162) 및 로드 큐(164)에 버퍼링되어 많은 메모리 연산이 병렬로 수행되게 할 수 있다. 메모리 연산을 보조하기 위해, 일반적으로 CPU 코어(100)가 복수의 데이터 관련 캐시 및 데이터 변환 색인 버퍼(DTLB)(170)를 포함한다. DTLB 캐시 계층구조(172)가 가상 주소를 물리 주소 변환, 가령, 페이지 테이블 항목, 페이지 디렉토리 항목 등에 캐싱하고, 이 정보는 메모리 연산의 가상 주소를 물리 주소로 변환하여, 데이터가 시스템 메모리에 저장 또는 시스템 메모리로부터 로딩될 수 있게 하는 데 사용된다. 일반적으로 상기 데이터는 코어 내에 상주하는 레벨 1 데이터 캐시(L1 D-캐시)(174)뿐 아니라 코어(100) 외부에 있는 다른 캐시 레벨(176)에 캐싱된다.

[0044] 본 발명의 특정 양태에 따라서, CPU는 복수의 코어를 포함할 수 있다. 비제한적 예를 들면, 도 2b는 본 발명의 양태와 함께 사용될 수 있는 가능한 멀티-코어 CPU(200)의 일례를 도시한다. 구체적으로, CPU(200)의 아키텍처가 M개의 클러스터(201-1…201-M)를 포함할 수 있고, 여기서 M은 0보다 큰 정수이다. 각각의 클러스터는 N개의 코어(202-1, 202-2…202-N)를 가질 수 있으며, 여기서 N은 1보다 큰 정수이다. 본 발명의 양태는 상이한 클러스터가 상이한 개수의 코어를 갖는 구현예를 포함하다. 각각의 코어는 하나 이상의 대응하는 전용 로컬 캐시(가령, L1 명령, L1 데이터, 또는 L2 캐시)를 포함할 수 있다. 로컬 캐시 각각은 임의의 타 코어와 공유되지 않는다는 점에서 특정 대응하는 코어에 특화될 수 있다. 각각의 클러스터는 대응하는 클러스터 내 코어들 간에 공유될 수 있는 클러스터-레벨 캐시(203-1…203-M)를 더 포함할 수 있다. 일부 구현예에서, 상이한 캐시와 연관되는 코어에 의해 클러스터-레벨 캐시가 공유되지 않는다. 덧붙여, CPU(200)는 클러스터들 간에 공유될 수 있는 하나 이상의 상위 레벨 캐시(204)를 포함할 수 있다. 클러스터 내 코어들 간 통신을 촉진시키기 위해, 클러스터(201-1…202-M)는 각각의 코어 및 클러스터에 대한 클러스터-레벨 캐시에 연결되는 대응하는 로컬 버스(205-1…205-M)를 포함할 수 있다. 마찬가지로, 클러스터들 간 통신을 촉진시키기 위해, CPU(200)는 클러스터(201-1…201-M) 및 상위 레벨 캐시(204)에 연결된 하나 이상의 상위 레벨 버스(206)를 포함할 수 있다. 일부 구현예에서, 상위 레벨 버스 또는 버스들(206)이 또한 다른 디바이스, 가령, GPU, 메모리, 또는 메모리 제어기로 연결될 수 있다. 또 다른 구현예에서, 상위 레벨 버스 또는 버스들(206)이 시스템 내 상이한 디바이스로 연결되는 디바이스-레벨 버스로 연결될 수 있다. 또 다른 구현예에서, 상위 레벨 버스 또는 버스들(206)은 클러스터(201-1…201-M)를 상위 레벨 캐시(204)로 연결할 수 있고 디바이스-레벨 버스(208)는 상위 레벨 캐시(204)를 다른 디바이스, 가령, GPU, 메모리, 또는 메모리 제어기로 연결할 수 있다. 비제한적 예를 들면, 이러한 디바이스-레벨 버스(208)에 의한 구현은, 가령, 상위 레벨 캐시(204)가 모든 CPU 코어에 대해 L3이지만, GPU 용도에 대해서는 아닌 경우 발생할 수 있다.

[0045] CPU(200)에서 OS 처리가 특정 코어 또는 특정 코어 서브세트 상에서 주로 발생할 수 있다. 마찬가지로, 애플리케이션-레벨 처리가 특정 코어 또는 코어 서브세트 상에서 주로 발생할 수 있다. 애플리케이션에 의해 개별 애플리케이션 스레드가 특정 코어 또는 특정 코어 서브세트 상에서 실행되도록 지정될 수 있다. 캐시 및 버스가 공유되기 때문에, 특정 애플리케이션 스레드에 의한 처리 속도가 특정 애플리케이션 스레드와 동일한 클러스터에서 실행되는 다른 스레드(가령, 애플리케이션 스레드 또는 OS 스레드)에 의해 발생하는 처리에 따라 달라질 수 있다. CPU(200)의 세부사항에 따라서, 하나의 코어가 한 번에 단 하나의 스레드만 실행시키거나 복수의 스레드를 동시에 실행시킬 수 있다("하이퍼스레딩(hyperthreading)"). 하이퍼스레딩된 CPU의 경우, 애플리케이션은 어느 스레드가 어느 다른 스레드와 동시에 실행될 수 있는지를 지정할 수 있다. 스레드의 성능은 동일한 코어에 의해 실행되는 그 밖의 다른 임의의 스레드에 의해 수행되는 특정 처리에 의해 영향 받는다.

[0046] 이제 도 3을 참조하면, 본 발명의 양태에 따라 동작하도록 구성된 디바이스(300)의 예시가 도시된다. 본 발명의 양태에 따라, 디바이스(300)가 임베디드 시스템, 모바일 전화기, 개인 컴퓨터, 태블릿 컴퓨터, 휴대용 게임 디바이스, 워크스테이션, 게임 콘솔 등일 수 있다.

[0047] 일반적으로 디바이스(300)는 도 1에 도시되고 앞서 설명된 바 있는 유형의 하나 이상의 CPU 코어(323)를 포함할 수 있는 중앙 처리 장치(CPU)(320)를 포함한다. CPU(320)는 도 2의 CPU(200)에서 나타난 것과 유사한 구성으로 복수의 이러한 코어(323) 및 하나 이상의 캐시(325)를 포함할 수 있다. 비제한적 예를 들면, CPU(320)가 단일 칩 상에 CPU(320)와 그래픽 처리 유닛(GPU)(330)를 포함하는 가속 처리 유닛(APU: accelerated processing unit)(310)의 일부일 수 있다. 대안적 구현예에서, CPU(320) 및 GPU(330)는 별개 칩 상의 별개 하드웨어 구성요소로서 구현될 수 있다. GPU(330)는 또한 둘 이상의 코어(332) 및 둘 이상의 캐시(334) 및 (일부 구현예에서) 하나 이상의 버스를 더 포함하여, 코어와 캐시 및 그 밖의 다른 시스템 구성요소 간 통신을 촉진시킬 수 있다. 버스는 APU(310)용 내부 버스 또는 버스들(317) 및 외부 데이터 버스(390)를 포함할 수 있다.

[0048] 디바이스(300)는 메모리(340)를 더 포함할 수 있다. 선택사항으로서, 상기 메모리(340)는 CPU(320) 및 GPU(330)에 의해 액세스 가능한 메인 메모리 유닛을 포함할 수 있다. CPU(320) 및 GPU(330)는 각각 하나 이상의 프로세서 코어, 가령, 단일 코어, 두 개의 코어, 네 개의 코어, 여덟 개의 코어 또는 그 이상을 포함할 수 있다. CPU(320) 및 GPU(330)는 외부 데이터 버스(390)를 이용해 하나 이상의 메모리 유닛을 액세스하도록 구성될 수 있으며 일부 구현에서, 디바이스(300)가 둘 이상의 상이한 버스를 포함하는 것이 유용할 수 있다.

[0049] 메모리(340)는 주소 지정 가능 메모리(addressable memory)를 제공하는 집적 회로, 가령, RAM, DRAM 등의 형태로 된 하나 이상의 메모리 유닛을 포함할 수 있다. 메모리는 방법, 예컨대, 도 5의 방법이 결정을 위해 실행될 때, 레거시 CPU 상에서 실행되도록 본래 생성된 애플리케이션을 실행시킬 때 디바이스(300)를 타이밍 시험 모드에서 동작하도록 하는 실행 명령을 포함할 수 있다. 덧붙여, 메모리(340)는 그래픽 자원을 일시적으로 저장하기

위한 전용 그래픽 메모리, 그래픽 버퍼, 및 그래픽 렌더링 파이프라인을 위한 그 밖의 다른 그래픽 데이터를 포함할 수 있다.

[0050] CPU(320)는 운영 체제(OS)(321) 또는 애플리케이션(322)(가령, 비디오 게임)을 포함할 수 있는 CPU 코드를 실행하도록 설정될 수 있다. 운영 체제는 소프트웨어(가령, 애플리케이션(322))으로부터 입/출력(I/O) 요청을 관리하고 이를 CPU(320), GPU(330) 또는 디바이스(300)의 그 밖의 다른 구성요소에 대한 데이터 처리 명령으로 변환하는 커넬(kernel)을 포함할 수 있다. OS(321)는 또한 비휘발성 메모리에 저장될 수 있는 펌웨어를 포함할 수 있다. 상기 OS(321)는, 이하에서 상세히 설명될 바와 같이, 타이밍 시험 모드에서 CPU(320)를 동작시키는 특정 기능을 구현하도록 구성될 수 있다. CPU 코드는 애플리케이션(322)의 상태를 기초로 GPU(330)에 의해 구현되는 그리기 명령어 또는 그리기 콜을 발행하기 위한 그래픽 애플리케이션 프로그래밍 인터페이스(API)(324)를 포함할 수 있다. 또한 CPU 코드는 물리적 시뮬레이션 및 그 밖의 다른 기능을 구현할 수 있다. OS(321), 애플리케이션(322) 또는 API(324) 중 하나 이상에 대한 코드의 일부분이 메모리(340), CPU의 내부 또는 외부 캐시 또는 CPU(320)에 의해 액세스 가능한 대용량 저장 디바이스에 저장될 수 있다.

[0051] 디바이스(300)는 메모리 제어기(315)를 포함할 수 있다. 메모리 제어기(315)는 메모리(340) 안팎으로 오고 가는 데이터의 흐름을 관리하는 디지털 회로일 수 있다. 비제한적 예를 들면, 메모리 제어기가 도 3에 도시된 예시에서와 같이 APU(310)의 일체 부분이거나, 별개 하드웨어 구성요소일 수 있다.

[0052] 디바이스(300)는 또한, 가령 버스(390)를 통해 시스템의 타 구성요소와 통신할 수 있는 공지된 지원 기능(350)을 포함할 수 있다. 이러한 지원 기능의 비제한적 예를 들면, 입/출력(I/O) 요소(352), CPU(320), GPU(330) 및 메모리(340) 각자에 대해 개별 클록을 포함할 수 있는 하나 이상의 클록(356), 및 CPU(320) 및 GPU(330)의 외부에 위치할 수 있는 하나 이상의 레벨의 캐시(358)가 있을 수 있다. 디바이스(300)는 선택사항으로서 프로그램 및/또는 데이터를 저장하기 위해 대용량 저장 디바이스(360), 가령, 디스크 드라이브, CD-ROM 드라이브, 플래시 메모리, 테이프 드라이브, 블루-레이 드라이브 등을 포함할 수 있다. 하나의 예를 들면, 대용량 저장 디바이스(360)는 본래 레거시 CPU를 갖는 시스템 상에서 실행되도록 설계된 레거시 애플리케이션을 담는 컴퓨터 판독 매체(362)를 수용할 수 있다. 대안으로, 레거시 애플리케이션(362)(또는 이의 일부)은 메모리(340)에 저장되거나 캐시(358)에 부분적으로 저장될 수 있다.

[0053] 또한 디바이스(300)는 GPU(330)에 의해 제공된 렌더링된 그래픽(382)을 사용자에게 표시하기 위한 디스플레이 유닛(380)을 더 포함할 수 있다. 디바이스(300)는 시스템(100)과 사용자 간 대화를 촉진시키기 위한 사용자 인터페이스 유닛(370)을 더 포함할 수 있다. 디스플레이 유닛(380)은 평면 패널 디스플레이, 캐소드 레이 튜브(CRT) 스크린, 터치 스크린, 두부 장착형 디스플레이(HMD), 또는 텍스트, 숫자, 그래픽 심볼 또는 이미지를 디스플레이할 수 있는 그 밖의 다른 디바이스의 형태를 가질 수 있다. 디스플레이(380)는 본 명세서에 기재된 다양한 기법에 따라 렌더링된 그래픽(382)을 디스플레이할 수 있다. 사용자 인터페이스(370)는 하나 이상의 주변 장치, 가령, 키보드, 마우스, 조이스틱, 광 펜(light pen), 게임 제어기, 터치 스크린, 및/또는 그래픽 사용자 인터페이스(GUI)와 함께 사용될 수 있는 그 밖의 다른 디바이스를 포함할 수 있다. 특정 구현예에서, 가령, 애플리케이션(322)이 비디오 게임 또는 그 밖의 다른 그래픽 집중형 애플리케이션을 포함하는 경우 애플리케이션(322)의 상태 및 그래픽의 기저 콘텐츠가 적어도 부분적으로 사용자 인터페이스(370)를 통한 사용자 입력에 의해 결정될 수 있다.

[0054] 디바이스(300)는 디바이스로 하여금 네트워크를 통해 타 디바이스와 통신하게 할 수 있는 네트워크 인터페이스(372)를 더 포함할 수 있다. 네트워크는, 가령, 로컬 영역 네트워크(LAN), 광역 네트워크, 가령, 인터넷, 개인 영역 네트워크, 가령, 블루투쓰 네트워크(Bluetooth network) 또는 그 밖의 다른 유형의 네트워크일 수 있다. 도시되고 기재된 구성요소들 중 다양한 구성요소가 하드웨어, 소프트웨어, 또는 펌웨어, 또는 이들 중 둘 이상의 일부 조합으로 구현될 수 있다.

[0055] 이하의 표 3은 앞서 도 2a, 도 2b, 및 도 3과 관련하여 기재된 특정 하드웨어 요소가 성능 정보 및 조절될 대응하는 연산 파라미터를 결정하는 데 사용될 수 있는 방법의 비제한적 예시를 나열한다.

표 3

성능 정보	기록/도출 방법	조절할 연산 파라미터
사이클당 명령 (IPC)	실행된 명령의 카운터 판독(도 2a, 은퇴 큐(132)의 일부분)	명령 개시율 조절(도 2a, CPU에 대해 스케줄링 큐(136) 및 도 3, GPU에 대해 GPU 코어(332)의 일부)
ALU 연산 발행 주파수	ALU 카운터 판독 (도 2a, CPU에 대해 ALU(154)의 일부 및 SIMD 파이프(152) 및 도 3, GPU에 대해 GPU 코어(332)의 일부)	매 N회 사이클마다 ALU 연산을 허용하지 않음(도 2a, CPU에 대한 스케줄링 큐(136)의 일부 및 도 3, GPU에 대한 GPU 코어(332)의 일부)
메모리 연산 발행 주파수	메모리 연산 카운터(들) 판독(도 2a, CPU에 대해 AGU(156)의 일부, GPU에 대해 GPU 코어(332)의 일부)	매 N회 사이클마다 메모리 연산을 허용하지 않음(도 2a, CPU에 대해 스케줄링 큐(136)의 일부 및 도 3, GPU에 대해 GPU 코어(332)의 일부)
단위 시간당 평균 웨이브프론트 점유시간	GPU 코어에 의해 발생된 웨이브프론트 시작 및 완료 이벤트를 기록 (도 3, GPU 코어(332)의 일부)	GPR(도 3, GPR(336))을 선택적으로 할당 또는 웨이브프론트 개시율 스로틀링(도 3, GPU 코어(332)의 일부)
평균 웨이브프론트 수명	GPU 코어에 의해 발생된 웨이브프론트 시작 및 완료 이벤트를 기록(도 3, GPU 코어(332)의 일부) 또는 웨이브프론트 수명 카운터 코어 판독(도 3, GPU 코어(332)의 일부)	선택적으로 GPR 할당(도 3, GPR(336) 또는 웨이브프론트 개시율 스로틀링(도 3, GPU 코어(332)의 일부)
단위 시간당 타깃을 렌더링하도록 출력되는 픽셀	픽셀 카운터 판독(도 3, GPU 코어(332)의 일부)	렌더 백엔드로부터의 출력율(# 픽셀) 스로틀링(도 3, GPU 코어(332)의 일부)
평균 메모리 연산 대기시간	미결 메모리 연산의 길이 추적(도 2a, CPU에 대해 저장 큐(162) 및 로드 큐(164)의 일부 및 도 3 GPU에 대해 GPU 코어(332)의 일부)	메모리 연산이 완료되지 못하게 정지시킴 (도 2a, CPU에 대해 저장 큐(162) 및 로드 큐(164)의 일부 및 도 3 GPU에 대해 GPU 코어(332)의 일부) 또는 상이한 율로 클록 실행(도 3, CLK(356))
브랜치 예측 적중 및 부적중	브랜치 예측 적중 및 부적중의 카운터 판독(도 2a, 브랜치 예측(102)의 일부)	레거시 브랜치 예측 알고리즘 매칭(도 2a, 브랜치 예측(102)의 일부) 또는 상이한 율로 클록 실행(도 3, CLK(356))
PC 블록 레지던스	판독 프로그램 카운터(도 2a, 인출 및 디코딩 유닛(110)의 일부)	직접 조정할 연산 파라미터가 없으나, 연산 파라미터 값이 블록 레지던스와 더 연관되는 경우 유용한 정보

[0056]

[0057] 새로운 디바이스 상에서 레거시 애플리케이션을 실행시킬 때 기록 또는 도출되는 성능 정보가 레거시 디바이스 상에서 레거시 애플리케이션을 실행시키기 위한 대응하는 성능 정보를 충족 또는 초과하는 것이 가능하지 않을 수 있다. 예를 들어, 연산 파라미터가 조절되는 방식과 무관하게 새로운 디바이스 상에서의 평균 메모리 대기시간이 레거시 디바이스 상에서 측정된 평균 메모리 대기시간보다 높을 수 있다. 새로운 시스템에 대해 이러한 유형의 성능 정보를 아는 것이 연산 파라미터를 조절할 때 유용할 수 있지만, 레거시 디바이스 상에서 실행되는 애플리케이션과 새로운 디바이스 상에서 실행되는 애플리케이션의 성능 특성을 비교할 때 사용되지 않아야 한다. 핵심 성능 메트릭, 비제한적 예를 들면, 초당 프레임(FPS: frames per second) 및 사이클당 명령(IPC: instructions per cycle)이 실제로 사용되어야 한다.

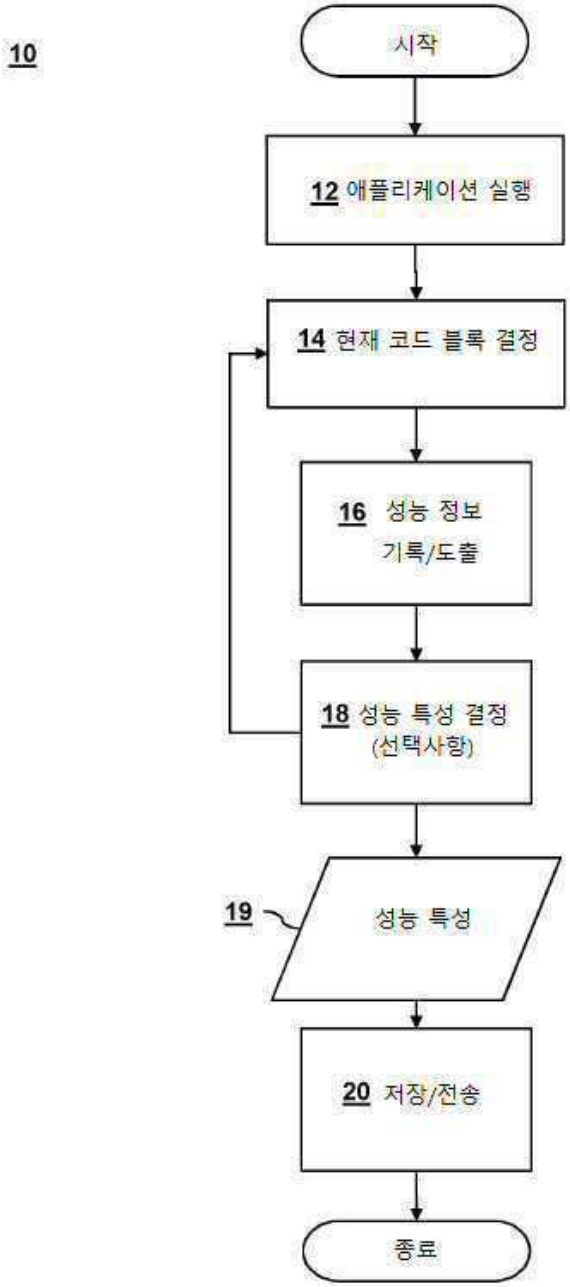
[0058] 애플리케이션이 새로운 디바이스 상에서 실행될 때 레거시 디바이스 상에서 실행되는 동일한 애플리케이션의 핵심 성능 메트릭을 충족시키기 위해, 연산 파라미터를 조절하는 프로세스가 비디오 게임과 관련된 다음의 예시로부터 이해될 수 있다. 먼저, 레거시 디바이스 상에서 실행되는 게임에 대해 성능 데이터가 수집되어 이의 핵심 성능 메트릭을 결정할 수 있다. 그 후 새로운 디바이스의 연산 파라미터를 조절하면서 게임이 새로운 디바이스 상에서 실행된다. 새로운 디바이스의 성능은 게임이 레거시 디바이스 상에서 실행됐을 때와 동일한 성능 정보를 새로운 디바이스 상에서 수집하고, 이들 두 디바이스 상에서 실행되는 게임의 핵심 성능 메트릭들을 비교함으로써 측정될 수 있다. 새로운 디바이스에 대한 성능 정보가 레거시 디바이스로부터의 성능 데이터와 완벽하게 매칭되는 것이 바람직할 수 있지만, 현실적으로 가능하지 않을 수 있다. 새로운 디바이스 상의 성능 정보가 레거시 디바이스 상의 성능 정보와 가능한 비슷하게 매칭되는 것이 충분하다. 그러나 새로운 디바이스 상의 핵심 성

능 메트릭이 레거시 디바이스 상의 핵심 성능 메트릭보다 열악한 것은 용납될 수 없는데, 이는 애플리케이션 또는 게임이 (일반적으로 동기화 문제 때문에)충돌하거나 (동일한 이유로) 올바르지 않은 결과를 생성하기 때문이다.

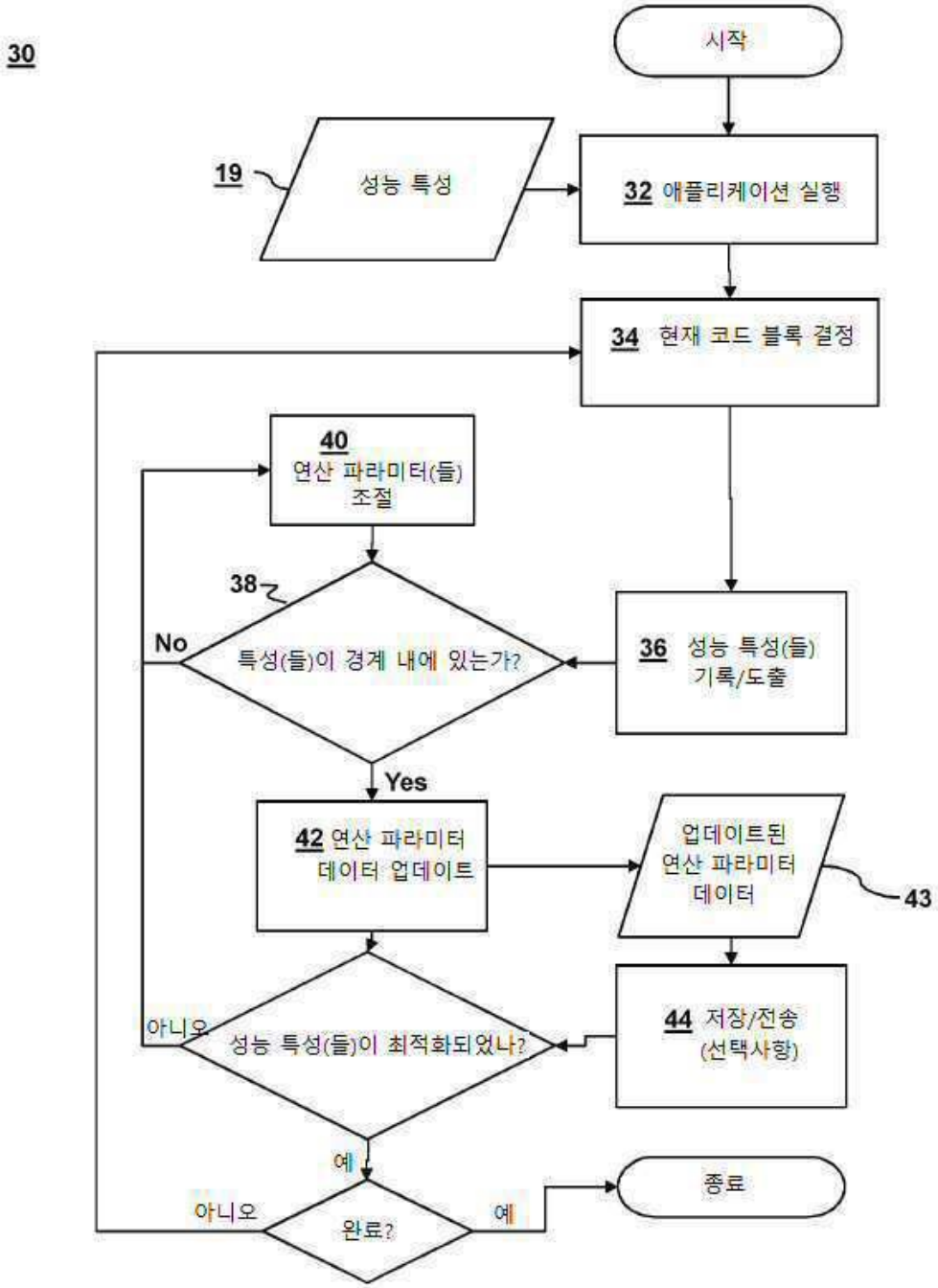
[0059] 현실적으로 말하자면, 게임이 새로운 디바이스 상에서 실행되는 처음 몇 번은 연산 파라미터가 레거시 디바이스 상에서와 동일하도록 설정될 가능성이 높을 것이다. 새로운 디바이스 상에서 충분한 게임이 실행되고 이들의 연산 파라미터가 튜닝되면, 이러한 경험 및 데이터가 추가 게임을 위해 사용될 수 있는 휴리스틱(heuristic)을 구축하는 데 사용될 수 있다. 상기 휴리스틱은 게임의 성능 특성을 기초로 새로운 디바이스 상의 연산 파라미터의 초기 값을 설정하는 데 사용될 수 있다. 그 후 게임은 새로운 디바이스 상에서 실행될 것이고 연산 파라미터는 핵심 성능 메트릭에 더 잘 매칭되도록 수정될 수 있다. 새로운 디바이스 상에서 측정된 모든 성능 데이터가 핵심 성능 메트릭이 아니라 연산 파라미터를 조절하는 것을 보조하기 위해 사용될 수 있다. 연산 파라미터의 임의의 조절이 또한 휴리스틱을 더 정제(refine)하는 데 사용될 수 있다.

도면

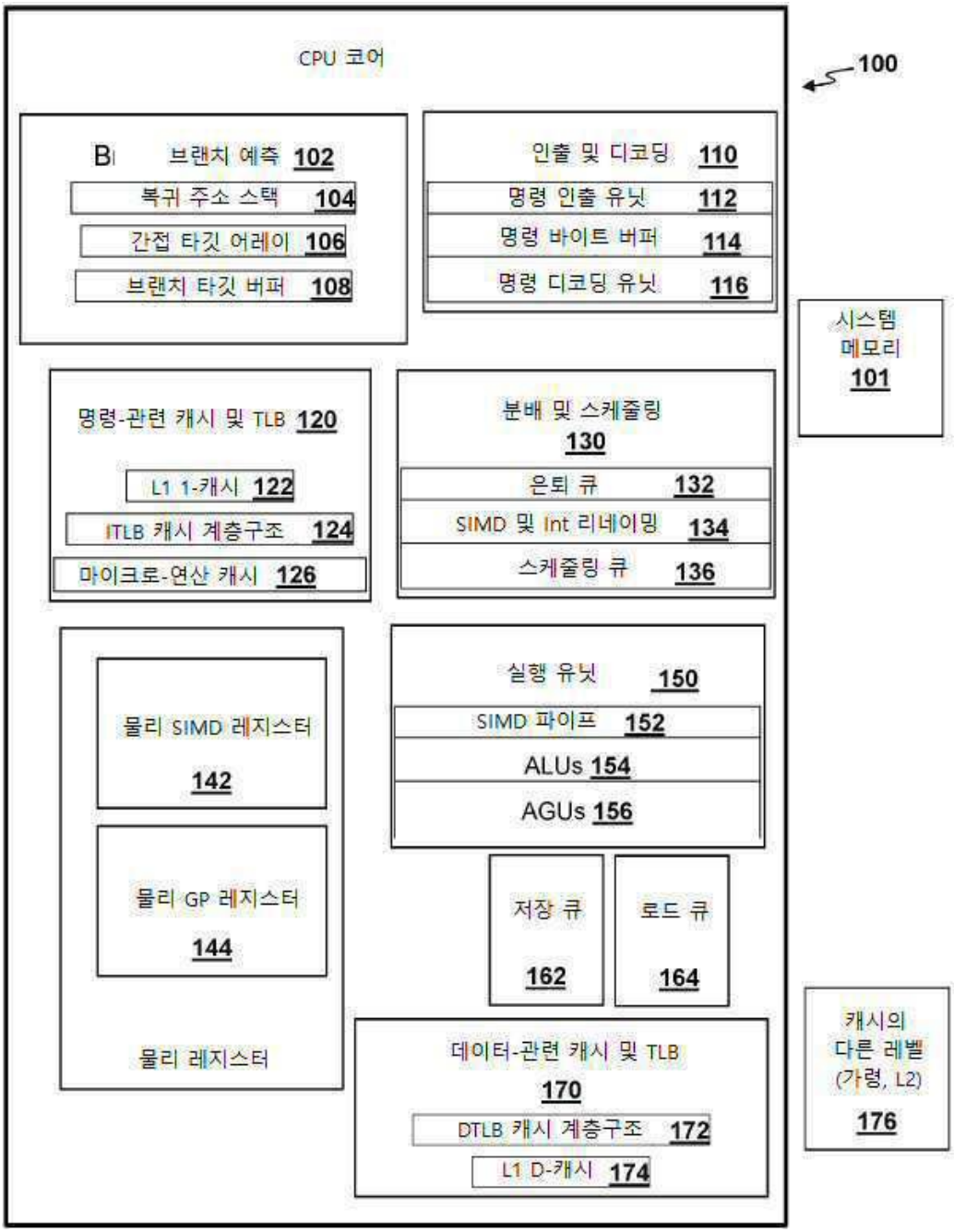
도면1a



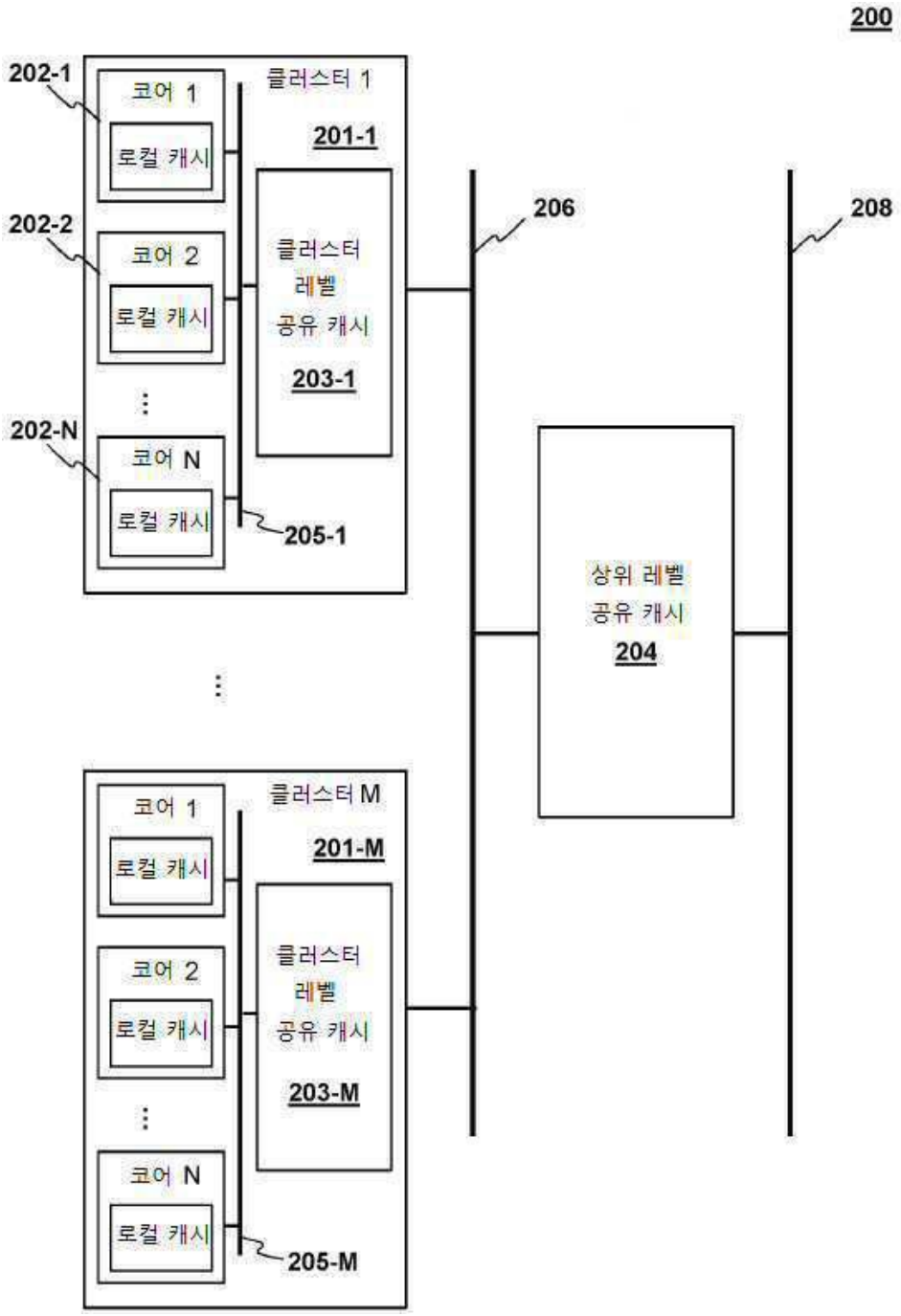
도면1b


30
시작
19
성능 특성
32 애플리케이션 실행
34 현재 코드 블록 결정
40
연산 파라미터(들)
조절
38
No
특성(들)이 경계 내에 있는가?
36 성능 특성(들)
기록/도출
Yes
42 연산 파라미터
데이터 업데이트
업데이트된
연산 파라미터
데이터
43
아니오
성능 특성(들)이 최적화되었나?
44 저장/전송
(선택사항)
예
아니오
완료?
예
종료

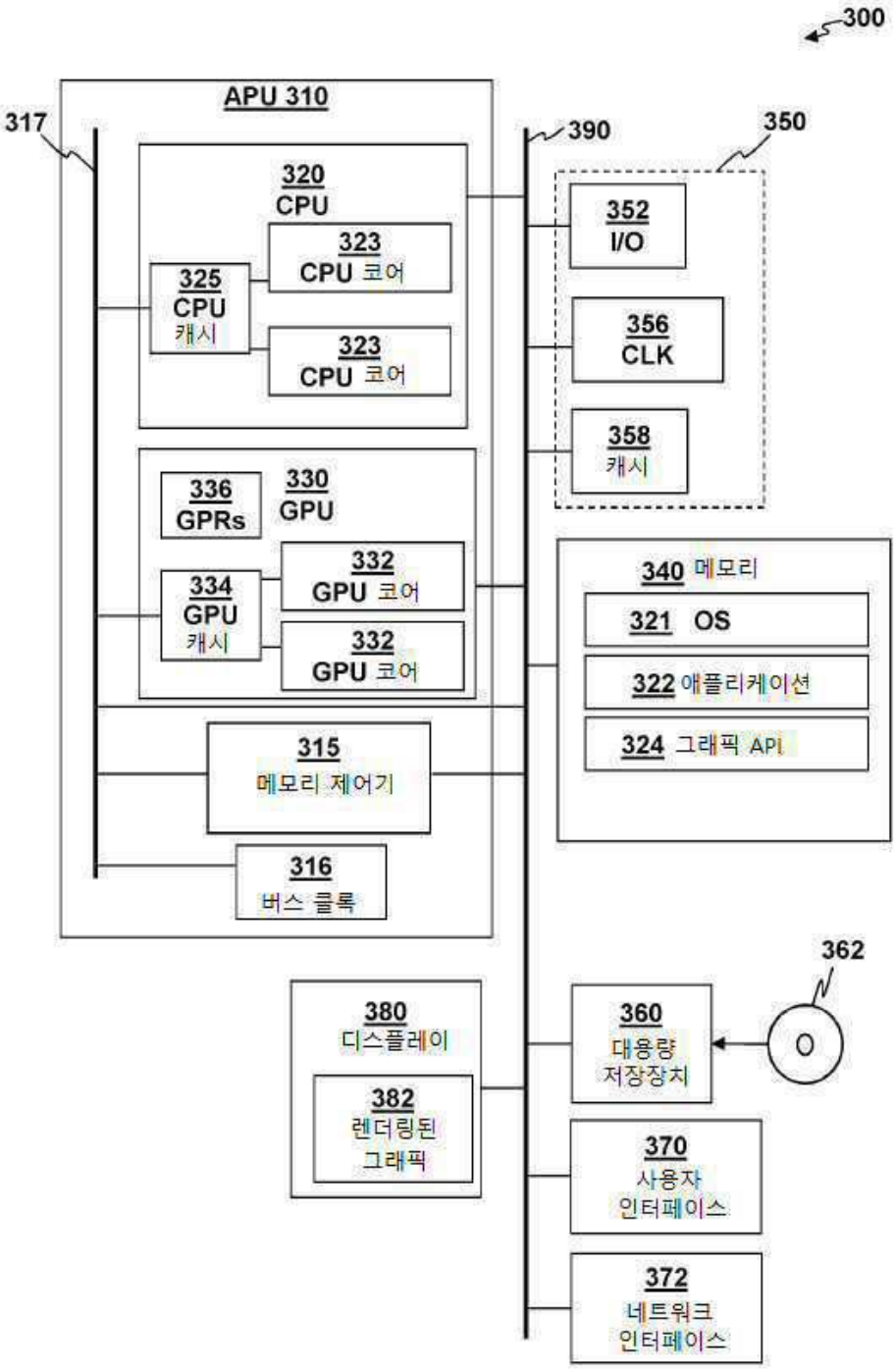
도면2a


CPU 코어
100
BI 브랜치 예측 102
복귀 주소 스택 104
간접 타깃 어레이 106
브랜치 타깃 버퍼 108
인출 및 디코딩 110
명령 인출 유닛 112
명령 바이트 버퍼 114
명령 디코딩 유닛 116
시스템 메모리 101
명령-관련 캐시 및 TLB 120
L1 1-캐시 122
ITLB 캐시 계층구조 124
마이크로-연산 캐시 126
분배 및 스케줄링 130
은퇴 큐 132
SIMD 및 Int 리네이밍 134
스케줄링 큐 136
물리 SIMD 레지스터 142
물리 GP 레지스터 144
물리 레지스터
실행 유닛 150
SIMD 파이프 152
ALUs 154
AGUs 156
저장 큐 162
로드 큐 164
데이터-관련 캐시 및 TLB 170
DTLB 캐시 계층구조 172
L1 D-캐시 174
캐시의 다른 레벨 (가령, L2) 176

도면2b


200
202-1
코어 1
로컬 캐시
클러스터 1
201-1
202-2
코어 2
로컬 캐시
클러스터
레벨
공유 캐시
203-1
206
208
202-N
코어 N
로컬 캐시
205-1
상위 레벨
공유 캐시
204
코어 1
로컬 캐시
클러스터 M
201-M
코어 2
로컬 캐시
클러스터
레벨
공유 캐시
203-M
코어 N
로컬 캐시
205-M

도면3


300
APU 310
317
390
350
320
CPU
323
CPU 코어
325
CPU
캐시
323
CPU 코어
352
I/O
356
CLK
358
캐시
336
GPRs
330
GPU
334
GPU
캐시
332
GPU 코어
332
GPU 코어
340 메모리
321 OS
322 애플리케이션
324 그래픽 API
315
메모리 제어기
316
버스 클록
362
380
디스플레이
382
렌더링된
그래픽
360
대용량
저장장치
370
사용자
인터페이스
372
네트워크
인터페이스