(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2020년04월14일
(11) 등록번호 10-2086019
(24) 등록일자 2020년03월02일

(51) 국제특허분류(Int. Cl.)
G06F 9/50 (2018.01) *G06F 16/00* (2019.01)
G06T 1/00 (2006.01)
(52) CPC특허분류
G06F 9/5033 (2013.01)
G06F 16/24569 (2019.01)
(21) 출원번호 10-2015-7000272
(22) 출원일자(국제) 2013년06월07일
심사청구일자 2018년06월05일
(85) 번역문제출일자 2015년01월06일
(65) 공개번호 10-2015-0024884
(43) 공개일자 2015년03월09일
(86) 국제출원번호 PCT/US2013/044682
(87) 국제공개번호 WO 2013/185015
국제공개일자 2013년12월12일
(30) 우선권주장
61/657,404 2012년06월08일 미국(US)
(56) 선행기술조사문헌
Jeff A. Stuart 외 2명. 'GPU-to-CPU Callback s'. Euro-Par 2010 Workshops, LNCS 6586, pp.365-372, 2011.*
(뒷면에 계속)
전체 청구항 수 : 총 20 항

(73) 특허권자
어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 95054 산타 클라라 어거스틴 드라이브 2485
(72) 발명자
랴세브스키 알렉산더
미국 캘리포니아 95014 쿠퍼티노 세달 스프링 코트 11607
(74) 대리인
박장원

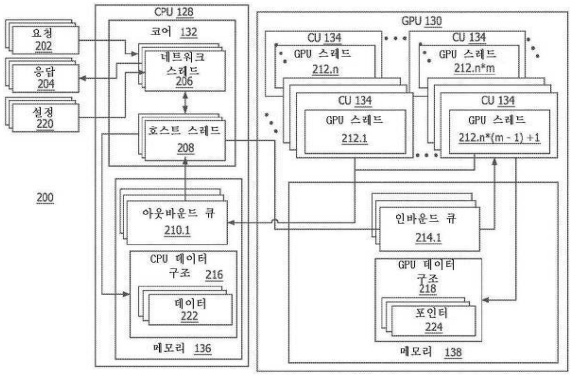
심사관 : 유진태

(54) 발명의 명칭 **이종 프로세서를 사용하여 애플리케이션에 낮은 레이턴시를 제공하는 시스템 및 방법**

(57) 요 약

요청에 응답하는 방법, 장치 및 컴퓨터 판독가능한 매체가 개시된다. 요청에 응답하는 방법은 콜백 함수를 포함하는 요청을 수신하는 단계를 포함할 수 있다. 하나 이상의 요청은 CPU일 수 있는 제1 유형의 프로세서와 연관된 제1 메모리에서 수신될 수 있다. 이 요청은 제2 메모리로 이동될 수 있다. 제2 메모리는 GPU일 수 있는 제2 유형의 프로세서와 연관될 수 있다. GPU 스레드는 요청의 수가 적어도 임계 수치에 이를 때 이 요청을 처리하여 요청에 대한 결과를 결정할 수 있다. 본 방법은 이 결과를 제1 메모리로 이동시키는 단계를 포함할 수 있다. 본 방법은 CPU가 대응하는 결과로 하나 이상의 콜백 함수를 실행하는 단계를 포함할 수 있다. GPU 지속적인 스레드는 요청의 수를 체크하여 요청의 수가 임계값에 이를 때를 결정할 수 있다.

대 표 도 - 도2



(52) CPC특허분류
G06F 16/9014 (2019.01)
G06T 1/00 (2013.01)

(56) 선행기술조사문헌
US20110210982 A1*
US20110057937 A1
US7554959 B1
US6272612 A
*는 심사관에 의하여 인용된 문헌

명 세 서

청구범위

청구항 1

멀티-프로세서 컴퓨터 시스템에서 요청에 응답하는 방법으로서,

제1 유형의 하나 이상의 프로세서에 의해, 하나 이상의 요청을 수신하는 단계 - 각 요청은 키 값을 가지며, 상기 제1 유형의 하나 이상의 프로세서는 제1 메모리와 연관되고, 상기 제1 메모리는 복수의 데이터 값들을 갖는 제1 데이터 구조를 포함하고, 각각의 데이터 값은 상기 제1 메모리 내의 복수의 위치들 중 하나에 대응하며 - 와;

상기 하나 이상의 요청을 상기 제1 유형의 하나 이상의 프로세서와 연관된 상기 제1 메모리로부터 제2 유형의 하나 이상의 프로세서와 연관된 제2 메모리로 이동시키는 단계 -

상기 제2 메모리는 제2 데이터 구조를 포함하고, 그리고

상기 제2 데이터 구조는,

(i) 복수의 쌍의 값들 - 각각의 쌍의 값은 복수의 키 값들 중 하나와 상기 제1 메모리 내의 복수의 위치들 중 하나를 포함하며 - 과, 그리고

(ii) 각각이 상기 복수의 쌍의 값들 중 하나 이상에 대응하는 복수의 인덱스 값들을 포함하고,

각 포인터는 인덱스 값을 포함하며 - 와;

각각의 인덱스 값에서, 상기 복수의 쌍의 값들 중 하나 이상을 상기 제2 데이터 구조에 저장하는 단계 - 각 쌍의 값은 대응하는 데이터 값이 저장되는 상기 제1 메모리의 위치들 중 하나를 가지며 - 와;

상기 하나 이상의 요청 중 적어도 하나에 기초하여 상기 제2 유형의 하나 이상의 프로세서에 의해 상기 제2 메모리 내의 상기 제2 데이터 구조를 수정하는 단계와;

상기 제2 유형의 하나 이상의 프로세서에 의해 상기 하나 이상의 요청 각각에 대한 하나 이상의 포인터를 결정하는 단계 - 상기 하나 이상의 결정된 포인터 각각은 상기 키 값에 그리고 상기 제1 메모리 내의 데이터 값의 대응하는 위치에 대응하고, 상기 하나 이상의 결정된 포인터 각각은 각각의 대응하는 요청에서 상기 키 값으로부터 결정된 포인터 값을 포함하며 - 와;

상기 키 값 및 상기 대응하는 위치를 상기 제1 메모리로 이동시키는 단계와;

상기 하나 이상의 결정된 포인터 각각에 대해, 상기 제1 유형의 하나 이상의 프로세서에 의해, 상기 하나 이상의 결정된 포인터에 의해 지시된 하나 이상의 값에 대한 데이터를 검색하는 단계 - 상기 검색된 데이터는 상기 제1 메모리 내의 대응하는 위치에서 상기 제1 메모리 내의 상기 제1 데이터 구조로부터 검색되며 - 와;

상기 검색된 데이터를 전송하는 상기 제1 유형의 하나 이상의 프로세서에 의한 상기 하나 이상의 요청에 응답하는 단계를 포함하는 방법.

청구항 2

제1항에 있어서, 상기 제1 유형의 하나 이상의 프로세서는 하나 이상의 중앙 프로세서 유닛(CPU) 코어를 포함하고, 상기 제2 유형의 하나 이상의 프로세서는 하나 이상의 그래픽 프로세서 유닛(GPU) 코어를 포함하는 것인 방법.

청구항 3

제2항에 있어서, 상기 하나 이상의 GPU 코어는 상기 하나 이상의 CPU 코어가 상기 제2 메모리에 액세스하는 것보다 빠르게 상기 제2 메모리에 액세스하며; 그리고

상기 하나 이상의 CPU 코어를 위한 상기 제1 메모리의 제1 액세스 시간은 상기 하나 이상의 GPU 코어를 위한 상기 제1 메모리의 제2 액세스 시간보다 적은 것인 방법.

청구항 4

삭제

청구항 5

삭제

청구항 6

제1항에 있어서,

상기 제1 유형의 하나 이상의 프로세서에 의해 하나 이상의 요청을 수신하는 단계는:

상기 제1 유형의 하나 이상의 프로세서에 의해 설정 요청을 수신하는 단계와; 그리고

상기 설정 요청을 수신한 후, 상기 설정 요청에 기초하여 상기 제1 유형의 하나 이상의 프로세서에 의해 상기 제1 데이터 구조를 변경하는 단계를 더 포함하는 방법.

청구항 7

제1항에 있어서,

상기 이동시키는 단계는:

상기 하나 이상의 요청이 임계 수치에 이를 때 상기 하나 이상의 요청을 상기 제1 메모리로부터 상기 제2 메모리로 이동시키는 단계를 더 포함하는 것인 방법.

청구항 8

멀티-프로세서 컴퓨터 시스템에서 요청에 응답하는 방법으로서,

하나 이상의 요청을 수신하는 단계 - 상기 하나 이상의 요청 각각은 콜백 함수(callback function)를 포함하고 그리고 키 값을 가지며, 상기 하나 이상의 요청은 하나 이상의 중앙 프로세서 유닛(CPU) 코어와 연관된 제1 메모리에서 수신되고, 상기 제1 메모리는 복수의 데이터 값들을 갖는 제1 데이터 구조를 포함하고, 각각의 데이터 값은 상기 제1 메모리 내의 복수의 위치들 중 하나에 대응하며 - 와;

상기 하나 이상의 요청을 제2 메모리로 이동시키는 단계 - 상기 제2 메모리는 하나 이상의 그래픽 프로세서 유닛(GPU) 코어와 연관되고, 그리고 상기 제2 메모리는 제2 데이터 구조를 포함하고, 그리고

상기 제2 데이터 구조는,

(i) 복수의 쌍의 값들 - 각각의 쌍의 값은 복수의 키 값들 중 하나와 상기 제1 메모리 내의 복수의 위치들 중 하나를 포함하며 - 과, 그리고

(ii) 각각이 상기 복수의 쌍의 값들 중 하나 이상에 대응하는 복수의 인덱스 값들을 포함하며,

각 포인터는 인덱스 값을 포함하며 - 와;

각각의 인덱스 값에서, 상기 복수의 쌍의 값들 중 하나 이상을 상기 제2 데이터 구조에 저장하는 단계 - 각 쌍의 값은 대응하는 데이터 값이 저장되는 상기 제1 메모리 내의 위치들 중 하나를 가지며 - 와;

상기 하나 이상의 요청 중 적어도 하나에 기초하여 상기 하나 이상의 GPU 코어에 의해 상기 제2 메모리 내의 상기 제2 데이터 구조를 수정하는 단계와;

요청들의 수가 적어도 임계값일 때 상기 하나 이상의 요청 각각에 대한 하나 이상의 포인터를 결정하는 단계 - 상기 하나 이상의 결정된 포인터 각각은 상기 키 값에 그리고 상기 제1 메모리 내의 데이터 값의 대응하는 위치에 대응하고, 상기 하나 이상의 결정된 포인터 각각은 각각의 대응하는 요청에서 상기 키 값으로부터 결정된 포인터 값을 포함하며 - 와;

상기 키 값 및 상기 대응하는 위치를 상기 제1 메모리로 이동시키는 단계와;

상기 하나 이상의 결정된 포인터 각각에 대해, 상기 하나 이상의 CPU 코어에 의해, 상기 하나 이상의 결정된 포인터에 의해 지시된 하나 이상의 값에 대한 데이터를 검색하는 단계 - 상기 검색된 데이터는 상기 제1 메모리

내의 대응하는 위치에서 상기 제1 메모리로부터 검색되며 - 와;

상기 하나 이상의 CPU 코어에 의해, 상기 하나 이상의 요청의 콜백 함수를 사용하여 상기 검색된 데이터로 상기 하나 이상의 요청의 각각을 실행하는 단계를 포함하는 방법.

청구항 9

제8항에 있어서, 상기 하나 이상의 요청을 상기 제2 메모리로 이동시키는 단계는,

상기 하나 이상의 요청을 상기 제2 메모리에 있는 인바운드 큐(inbound queue)로 이동시키는 단계를 더 포함하는 것인 방법.

청구항 10

제9항에 있어서, 상기 요청의 수가 적어도 임계 수치에 이르고 상기 GPU가 하나 이상의 GPU 스레드를 포함할 때, 상기 방법은,

상기 하나 이상의 GPU 스레드 중 하나의 GPU 스레드에 의해, 상기 제2 메모리에 있는 상기 인바운드 큐를 체크하여 상기 요청의 수가 적어도 상기 임계 수치에 이를 때를 결정하는 단계를 더 포함하고, 상기 GPU 스레드는 지속적인 GPU 스레드인 것인 방법.

청구항 11

요청에 응답하는 멀티-프로세서 시스템으로서,

제1 유형의 하나 이상의 프로세서 - 상기 제1 유형의 하나 이상의 프로세서는 제1 메모리와 연관되고, 상기 제1 메모리는 복수의 데이터 값들을 갖는 제1 데이터 구조를 포함하고, 각각의 데이터 값은 상기 제1 메모리 내의 복수의 위치들 중 하나에 대응하며 - 와; 그리고

제2 유형의 하나 이상의 프로세서를 포함하고,

상기 제2 유형의 하나 이상의 프로세서는 제2 메모리와 연관되며,

상기 제2 메모리는 제2 데이터 구조를 포함하고, 그리고 상기 제2 데이터 구조는,

(i) 복수의 쌍의 값들 - 각각의 쌍의 값은 복수의 키 값들 중 하나와 상기 제1 메모리 내의 복수의 위치들 중 하나를 포함하며 - 과, 그리고

(ii) 각각이 상기 복수의 쌍의 값들 중 하나 이상에 대응하는 복수의 인덱스 값들을 포함하며,

각 포인터는 인덱스 값을 포함하며 - 을 포함하며,

각각의 인덱스 값에서, 상기 복수의 쌍의 값들 중 하나 이상은 상기 제2 데이터 구조에 저장되고, 각 쌍의 값은 대응하는 데이터 값이 저장되는 상기 제1 메모리의 위치들 중 하나를 가지며,

상기 제1 유형의 하나 이상의 프로세서는:

콜백 함수를 포함하고 그리고 각각이 키 값을 갖는 하나 이상의 요청을 수신하고,

상기 하나 이상의 요청을 상기 제2 메모리로 이동시키도록 구성되며,

상기 제2 유형의 하나 이상의 프로세서는:

상기 하나 이상의 요청 중 적어도 하나에 기초하여 상기 제2 메모리 내의 상기 제2 데이터 구조를 수정하고,

상기 하나 이상의 요청의 수가 적어도 임계 수치에 이를 때 상기 하나 이상의 요청들 각각에 대해 하나 이상의 포인터를 결정하고, 상기 하나 이상의 결정된 포인터 각각은 상기 키 값에 그리고 상기 제1 메모리 내의 데이터 값의 대응하는 위치에 대응하고, 상기 하나 이상의 결정된 포인터 각각은 각각의 대응하는 요청에서 상기 키 값으로부터 결정된 포인터 값을 포함하며,

상기 결정된 포인터들의 각각에서 유지되는 상기 키 값 및 상기 대응하는 위치를 상기 제1 메모리로 이동시키며,

상기 제1 유형의 하나 이상의 프로세서는,

상기 하나 이상의 결정된 포인터 각각에 대해, 상기 하나 이상의 결정된 포인터에 의해 지시된 하나 이상의 값에 대한 데이터를 검색하고, 상기 검색된 데이터는 상기 제1 메모리 내의 대응하는 위치에서 상기 제1 메모리로부터 검색되며; 그리고

상기 하나 이상의 요청의 콜백 함수를 사용하여 상기 검색된 데이터로 상기 하나 이상의 요청의 각각을 실행하도록 더 구성되는 것인 시스템.

청구항 12

제11항에 있어서, 상기 제2 메모리에 대한 액세스 시간은 상기 제1 유형의 하나 이상의 프로세서보다 상기 제2 유형의 하나 이상의 프로세서에 대해서 더 적은 것인 시스템.

청구항 13

제11항에 있어서, 상기 제1 유형의 하나 이상의 프로세서는 상기 하나 이상의 요청을 상기 제2 메모리에 있는 인바운드 큐로 이동시키도록 더 구성된 것인 시스템.

청구항 14

제13항에 있어서, 상기 제2 유형의 하나 이상의 프로세서는 상기 제2 메모리에 있는 상기 인바운드 큐를 체크하는 상기 하나 이상의 스레드 중 하나의 스레드를 실행하여 상기 하나 이상의 요청의 수가 적어도 상기 임계 수치에 이를 때를 결정하도록 더 구성되고,

상기 스레드는 지속적인 스레드인 것인 시스템.

청구항 15

제11항에 있어서, 상기 제2 유형의 하나 이상의 프로세서는 상기 검색된 데이터를 상기 제1 메모리에 있는 아웃바운드 큐로 이동시키도록 더 구성된 것인 시스템.

청구항 16

제2항에 있어서, 상기 하나 이상의 GPU 코어 각각은 GPU 커널에 기초하여 지속적인 스레드를 실행하는 것인 방법.

청구항 17

제2항에 있어서, 상기 하나 이상의 CPU 코어에 대한 상기 제1 메모리의 제1 액세스 시간은 상기 하나 이상의 GPU 코어에 대한 상기 제1 메모리의 제2 액세스 시간보다 적은 것인 방법.

청구항 18

제1항에 있어서, 상기 복수의 데이터 값들 중 하나의 데이터 값의 크기는 상기 하나 이상의 결정된 포인터 중 하나의 포인터 값의 크기보다 큰 것인 방법.

청구항 19

제8항에 있어서, 상기 제2 메모리에 대한 액세스 시간은 상기 하나 이상의 CPU 코어보다 상기 하나 이상의 GPU 코어에 대해 적은 것인 방법.

청구항 20

제8항에 있어서, 상기 검색된 데이터를 상기 제1 메모리로 이동시키는 단계는,

상기 검색된 데이터를 상기 제1 메모리에 있는 아웃바운드 큐로 이동시키는 단계를 포함하는 것인 방법.

청구항 21

제11항에 있어서,

상기 제1 유형의 하나 이상의 프로세서는 하나 이상의 중앙 프로세서 유닛 (CPU) 코어를 포함하고,

상기 제2 유형의 하나 이상의 프로세서는 하나 이상의 그래픽 프로세서 유닛 (GPU) 코어를 포함하는 것인 시스템.

청구항 22

제21항에 있어서, 상기 하나 이상의 GPU 스레드는 GPU 커널에 기초한 지속적인 스레드인 것인 시스템.

발명의 설명

기 술 분 야

[0001] **관련 출원에 대한 상호 참조**

[0002] 본 출원은, 전체 내용이 본 명세서에 완전히 개시된 것처럼 병합된 미국 가특허 출원 제61/657,404호(출원일: 2012년 6월 8일)의 우선권을 주장한다.

[0003] **기술 분야**

[0004] 본 발명의 실시예는 애플리케이션에 낮은 레이턴시를 제공하는 것에 관한 것으로, 보다 상세하게는 이종 프로세서(heterogeneous processor)를 사용하여 낮은 레이턴시를 제공하는 것에 관한 것이다.

배 경 기 술

[0005] 일부 컴퓨터 시스템은 2개 이상의 프로세서 유형을 포함한다. 예를 들어, 일부 컴퓨터 시스템은 하나 이상의 중앙 프로세서 유닛(central processor unit: CPU)(즉, 제1 프로세서 유형) 및 많은 주변 프로세서 - (즉, 상이한 또는 제2 유형의 프로세서)를 포함한다. 주변 프로세서는 종종 그래픽 프로세서 유닛(graphical processor unit: GPU)이나, 다른 프로세서 유형은 이 기술 분야에 통상의 지식을 가진 자에게 알려져 있다. CPU로부터 별도의 공유 메모리를 구비할 수 있는 많은 GPU들이 있을 수 있다. 일부 애플리케이션은 CPU만을 사용하거나 또는 GPU를 비효율적인 방식으로 사용한다.

[0006] 추가적으로, 일부 애플리케이션은 애플리케이션으로부터 오는 요청에 응답하는데 컴퓨터 시스템으로부터 낮은 레이턴시 또는 지연을 요구한다. 종종, 애플리케이션으로부터 오는 요청에 응답하는데 지연이 너무 길어지지 않는 것을 보장하는데 추가적인 하드웨어를 구매하여야 한다.

[0007] 따라서, 이종 처리를 사용하여 애플리케이션에 낮은 레이턴시를 제공하는 시스템 및 방법이 이 기술 분야에 요구된다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0008] 요청에 응답하는 방법, 장치 및 컴퓨터 판독가능한 매체가 개시된다. 요청에 응답하는 방법은 하나 이상의 중앙 처리 유닛(CPU)이 하나 이상의 요청을 수신하는 단계를 포함할 수 있다. 상기 방법은 상기 하나 이상의 요청을 상기 하나 이상의 CPU와 연관된 제1 메모리로부터 하나 이상의 그래픽 처리 유닛(GPU)과 연관된 제2 메모리로 이동시키는 단계를 포함할 수 있다. 상기 방법은 상기 하나 이상의 GPU가 상기 하나 이상의 요청 각각에 대해 포인터를 결정하는 단계를 포함할 수 있다. 상기 포인터는 상기 요청에 있는 정보에 기초하여 결정될 수 있다. 상기 방법은 상기 결정된 포인터를 상기 제1 메모리로 이동시키는 단계를 포함할 수 있다. 상기 결정된 포인터 각각에 대해, 상기 방법은 상기 결정된 포인터에 의해 지시된 데이터를 검색하는 단계를 포함할 수 있다. 상기 데이터는 상기 제1 메모리에서 제1 데이터 구조로부터 검색될 수 있다. 그리고, 상기 방법은 상기 하나 이상의 CPU가 상기 대응하는 검색된 데이터를 송신하는 것에 의해 상기 수신된 요청에 응답하는 단계를 포함할 수 있다.

[0009] 다른 실시예에서, 요청에 응답하는 방법은 콜백 함수(callback function)를 포함하는 하나 이상의 요청을 수신

하는 단계를 포함할 수 있다. 상기 하나 이상의 요청은 하나 이상의 CPU와 연관된 제1 메모리에서 수신될 수 있다. 상기 방법은 상기 하나 이상의 요청을 제2 메모리로 이동시키는 단계를 포함할 수 있다. 상기 제2 메모리는 하나 이상의 GPU와 연관될 수 있다. 상기 방법은 하나 이상의 GPU 스레드(thread)가 상기 하나 이상의 요청을 처리하여 상기 하나 이상의 요청의 수가 적어도 임계 수치에 이를 때 상기 하나 이상의 요청 각각에 대한 결과를 결정하는 단계를 포함할 수 있다. 상기 방법은 상기 결과를 상기 제1 메모리로 이동시키는 단계를 포함할 수 있다. 그리고, 상기 방법은 상기 하나 이상의 CPU가 상기 대응하는 결과로 상기 하나 이상의 콜백 함수 각각을 실행하는 단계를 포함할 수 있다.

[0010] 요청에 응답하는 시스템이 개시된다. 상기 시스템은 콜백 함수를 포함하는 하나 이상의 요청을 수신하도록 구성된 하나 이상의 CPU를 포함할 수 있다. 상기 하나 이상의 요청은 상기 하나 이상의 CPU와 연관된 제1 메모리에서 수신될 수 있다. 상기 하나 이상의 CPU는 상기 하나 이상의 요청을 제2 메모리로 이동시키도록 구성될 수 있다. 상기 제2 메모리는 하나 이상의 GPU와 연관될 수 있다. 그리고, 상기 하나 이상의 CPU는 대응하는 결과로 상기 하나 이상의 콜백 함수 각각을 실행하도록 구성될 수 있다. 상기 하나 이상의 GPU는 상기 하나 이상의 요청을 처리하는 하나 이상의 GPU 스레드를 실행하여 상기 하나 이상의 요청의 수가 적어도 임계 수치에 이를 때 상기 하나 이상의 요청 각각에 대한 결과를 결정하도록 구성될 수 있다. 그리고, 상기 하나 이상의 GPU는 상기 결정된 결과를 상기 제1 메모리로 이동시키도록 구성될 수 있다.

도면의 간단한 설명

[0011] 본 발명의 실시예는 동일한 부호가 유사한 요소를 나타내는 첨부 도면을 참조하여 비제한 예로서 설명된다.

도 1은 하나 이상의 개시된 실시예를 구현할 수 있는 예시적인 디바이스의 블록도;

도 2는 일부 개시된 실시예에 따라 이종 프로세서를 사용하여 낮은 레이턴시의 애플리케이션을 위한 시스템을 도시한 도면;

도 3 및 도 4는 요청에 응답하는데 낮은 레이턴시를 요구할 수 있는 메모리 캐시 애플리케이션의 동작을 도시한 개략도;

도 5 및 도 6은 일부 개시된 실시예에 따라 메모리 캐시 애플리케이션을 위한 이종 프로세서를 사용하여 낮은 레이턴시의 애플리케이션을 위한 시스템의 동작을 도시한 도면;

도 7은 일부 개시된 실시예에 따라 이종 프로세서를 사용하여 낮은 레이턴시의 애플리케이션을 위한 시스템의 일 실시예를 도시한 개략도;

도 8은 일부 개시된 실시예에 따라 GPU가 실행할 수 있는 커널(kernel)을 도시한 도면;

도 9는 일부 개시된 실시예에 따라 이종 프로세서를 위한 낮은 레이턴시의 애플리케이션을 위한 시스템을 호출(calling)하는 데이터 구조 및 콜(call)을 도시한 도면; 및

도 10은 일부 개시된 실시예에 따라 메모리 캐시 애플리케이션을 위한 이종 프로세서를 사용하여 낮은 레이턴시를 제공하는 시스템 및 방법의 경험적 테스트의 결과 테이블을 도시한 도면.

발명을 실시하기 위한 구체적인 내용

[0012] 도 1은 하나 이상의 개시된 실시예를 구현할 수 있는 예시적인 디바이스(100)의 블록도이다. 디바이스(100)는, 예를 들어, 컴퓨터, 게임 디바이스, 핸드헬드 디바이스, 셋탑 박스, 텔레비전, 모바일 폰 또는 태블릿 컴퓨터를 포함할 수 있다. 디바이스(100)는 프로세서(102), 메모리(104), 저장 매체(106), 하나 이상의 입력 디바이스(108) 및 하나 이상의 출력 디바이스(110)를 포함한다. 디바이스(100)는 입력 드라이버(112) 및 출력 드라이버(114)를 선택적으로 더 포함할 수 있다. 디바이스(100)는 도 1에 도시되지 않은 추가적인 컴포넌트를 포함할 수 있는 것으로 이해된다.

[0013] 프로세서(102)는 하나 이상의 코어(132)를 포함할 수 있는 제1 유형의 하나 이상의 제1 프로세서(예를 들어, 중앙 처리 유닛(CPU))(128) 및 하나 이상의 컴퓨팅 유닛(CU)(134) 또는 GPU 코어를 포함할 수 있는 그래픽 처리 유닛(GPU)(130)과 같은 제2 유형의 하나 이상의 프로세서를 포함할 수 있다. CPU(128) 및 GPU(130)는 동일한 다이(die) 또는 다수의 다이 상에 위치될 수 있다. CU(134)는 CU(134)의 그룹을 제어하는 처리 제어(미도시)를 갖는 그룹으로 구성될 수 있다. 처리 제어는 CU(134)의 그룹이 단일 명령 다수의 데이터(single instruction multiple data: SIMD) 처리 유닛(미도시)으로 수행되도록 CU(134)의 그룹을 제어할 수 있다. CU(134)는 하나

이상의 다른 CU(134)와 공유될 수 있는 메모리(139)를 포함할 수 있다. 예를 들어, 처리 제어는 132개의 CU(134)를 제어할 수 있고, 132개의 CU(134)는 처리 제어로 동일한 메모리(139)를 모두 공유할 수 있다.

[0014] GPU(130) 및 CPU(128)에 더하여 다른 유형의 프로세서 또는 연산 요소, 예를 들어, 디지털 신호 프로세서(digital signal processor: DSP), 애플리케이션 프로세서 등이 있을 수 있다. CPU(128)는 CPU(128)의 코어 중에서 공유된 메모리(136)를 포함할 수 있다. 일부 개시된 실시예에서, 메모리(136)는 L2 캐시이다. GPU(130)는 하나 이상의 GPU(130)의 CU(134) 중에서 공유된 메모리(138)를 포함할 수 있다. 데이터는 메모리(136)와 메모리(138)와 메모리(139) 사이에 (137)을 통해 전송될 수 있다. GPU(130) 및 CPU(128)는 각 코어(132)를 위한 메모리 및 CU(134)의 각 처리 유닛(미도시)을 위한 메모리와 같은 다른 메모리를 포함할 수 있다. 메모리(136, 138 및 104)는 코히어런트 캐시 시스템(coherent cache system)(미도시)의 일부일 수 있다. 일부 실시예에서, 메모리(136, 138 및 104)의 하나 이상은 코히어런트 메모리가 아닐 수 있다. 메모리(104)는 프로세서(102)와 동일한 다이 상에 위치되거나, 또는 프로세서(102)로부터 별개로 위치될 수 있다. 메모리(104)는 휘발성 또는 비-휘발성 메모리, 예를 들어, 랜덤 액세스 메모리(random access memory: RAM), 동적 RAM(DRAM) 또는 캐시를 포함할 수 있다.

[0015] 저장 매체(106)는 고정식 또는 이동식 저장 매체, 예를 들어, 하드 디스크 드라이브, 솔리드 스테이트 드라이브, 광 디스크 또는 플래쉬 드라이브를 포함할 수 있다. 입력 디바이스(108)는 키보드, 키패드, 터치 스크린, 터치 패드, 검출기, 마이크로폰, 가속도계, 자이로스코프, 생체 측정 스캐너 또는 네트워크 연결(예를 들어, 무선 IEEE 802 신호를 송신 및/또는 수신하기 위한 무선 근거리 네트워크 카드)을 포함할 수 있다. 출력 디바이스(110)는 디스플레이, 스피커, 프린터, 햅틱(haptic) 피드백 디바이스, 하나 이상의 조명, 안테나 또는 네트워크 연결(예를 들어, 무선 IEEE 802 신호를 송신 및/또는 수신하기 위한 무선 근거리 네트워크 카드)을 포함할 수 있다.

[0016] 입력 드라이버(112)는 프로세서(102) 및 입력 디바이스(108)와 통신하며, 이를 통해 프로세서(102)는 입력 디바이스(108)로부터 오는 입력을 수신할 수 있다. 출력 드라이버(114)는 프로세서(102) 및 출력 디바이스(110)와 통신하며, 이를 통해 프로세서(102)는 출력을 출력 디바이스(110)로 송신할 수 있다. 입력 드라이버(112) 및 출력 드라이버(114)는 선택적인 컴포넌트이고, 이 디바이스(100)는 입력 드라이버(112)와 출력 드라이버(114)가 존재하지 않는 경우 동일한 방식으로 동작할 수 있는 것으로 이해된다.

[0017] 도 2는 이종 프로세서를 사용하여 낮은 레이턴시 애플리케이션을 위한 시스템을 도시한다. 도 2에는 CPU(128), GPU(130), CU(134), 메모리(138), 요청(202), 응답(204), 설정(220), 네트워크 스레드(206), 호스트 스레드(208), 아웃바운드 큐(outbound queue)(210), CPU 데이터 구조(216), GPU 스레드(212), 인바운드 큐(inbound queue)(214) 및 GPU 데이터 구조(218)가 도시되어 있다. 요청(202)은 네트워크 스레드(206)에 의해 수신되고 호스트 스레드(208)로 전달되며, 이 호스트 스레드는 요청(202)을 인바운드 큐(214)에 놓는다. GPU 스레드(212)는 GPU 데이터 구조(218)를 사용하여 인바운드 큐(214)에 있는 요청(202)을 처리하고 응답(204)을 아웃바운드 큐(210)로 송신하며, 여기서 호스트 스레드(208)는 CPU 데이터 구조(216)를 사용하여 응답(204)을 처리할 수 있다. 응답(204)은 송신을 위해 네트워크 스레드(206)로 송신될 수 있다. 네트워크 스레드(206)는 설정(220)을 수신할 수 있고 이 설정은 CPU 데이터 구조(216) 및 GPU 데이터 구조(218)를 생성하거나 변경하는데 사용될 수 있다.

[0018] 요청(202)은 애플리케이션(미도시)으로부터 수신된 정보나 처리를 위한 요청(202)일 수 있다. 요청(202)은 컴퓨터 네트워크(미도시)를 통해 수신될 수 있다. 요청(202)의 예는 요청(202)에 있는 키에 대응하는 데이터(222)에 대한 요청일 수 있다. 요청(202)은 콜백 함수(call back function)(도 9 참조)를 포함할 수 있다.

[0019] 응답(204)은 요청(202)에 대한 응답(204)일 수 있다. 예시적인 응답(204)은 요청(202)에 있는 키(도 3 참조)에 대응하는 데이터(222)일 수 있다. 설정(220)은 CPU 데이터 구조(216)를 변경하거나 생성하는 명령일 수 있다. 예시적인 설정(220)은 CPU 데이터 구조(216)에 삽입될 새로운 데이터(222)일 수 있다.

[0020] 네트워크 스레드(206)는 입력 디바이스(108)로부터 오는 요청(202)과 설정(220)을 받고 입력 디바이스(108)를 통해 응답(204)을 송신하도록 구성될 수 있다. 예를 들어, 네트워크 스레드(206)는 소켓을 사용하여 요청(202) 및 설정(220)에 대한 하나 이상의 전송 제어 프로토콜(transport control protocol: TCP) 포트를 모니터링하고 TCP를 사용하여 하나 이상의 포트를 통해 응답(204)을 송신하는 멀티태스킹 운영 시스템의 스레드일 수 있다. 네트워크 스레드(206)는 요청(202) 및 설정(220)을 호스트 스레드(208)에 송신하거나 전달하고 호스트 스레드(208)로부터 응답(204)을 수신하도록 구성될 수 있다. CPU(128)는 네트워크 스레드(206)를 실행할 수 있다. 일부 실시예에서, 네트워크 스레드(206)는 메모리(136)(도 1 참조) 및 또는 메모리(104) 또는 코어(132)와 연관된

다른 메모리(미도시)에 존재할 수 있다. 일부 실시예에서, 네트워크 스레드(206)는 애플리케이션 스레드일 수 있다.

[0021] 호스트 스레드(208)는 요청(202)을 수신하고 이를 처리를 위해 GPU(130)용 인바운드 큐(214)에 놓아두도록 구성될 수 있다. 호스트 스레드(208)는 아웃바운드 큐(210)로부터 응답(204)을 수신하도록 구성될 수 있다. 일부 개시된 실시예에서, 호스트 스레드(208)는 아웃바운드 큐(210)를 모니터링할 수 있고, 하나 이상의 응답(204)이 이용가능하게 될 때 호스트 스레드(208)는 응답(204)을 취하고 CPU 데이터 구조(216)에 따라 응답(204)에 추가적인 처리를 수행할 수 있다. 예를 들어, 호스트 스레드(208)는 인바운드 큐(214)로부터 응답(204)을 취하고 응답(204) 내 포인터(224)를 사용하여 CPU 데이터 구조(216)로부터 데이터(222)를 검색하고, 데이터(222)를 포함하도록 응답(204)을 변경할 수 있다. 호스트 스레드(208)는 응답(204)을 네트워크 스레드(206)에 송신할 수 있다. 일부 실시예에서, 호스트 스레드(208)는 응답(204)의 수가 임계 수치 또는 주파수 아래에 있는 경우 응답(204)을 만족시킬 수 있다. 일부 실시예에서, 2개 이상의 호스트 스레드(208)가 있을 수 있다. 일부 실시예에서, 아웃바운드 큐(210)마다 하나의 호스트 스레드(208)가 있을 수 있다. 일부 실시예에서, 호스트 스레드(208)는 CPU(128)와 연관된 메모리에 존재할 수 있다. 일부 실시예에서, 네트워크 스레드(206)는 메모리(136) 및 또는 메모리(104) 또는 코어(132)와 연관된 다른 메모리(미도시)에 존재할 수 있다.

[0022] 아웃바운드 큐(210)는 GPU 스레드(212)에 의해 처리된 요청(202)이 놓인 큐일 수 있다. 일부 실시예에서, 아웃바운드 큐(210)의 수 및 호스트 스레드(208)의 수는 비례할 수 있다. 일부 실시예에서, 호스트 스레드(208)당 하나의 아웃바운드 큐(210)가 있을 수 있다. 일부 실시예에서, 아웃바운드 큐(210)는 메모리(136)에 존재하거나 또는 CPU(128)에 액세스가능한 다른 메모리에 존재할 수 있다.

[0023] GPU 스레드(212)는 요청(202)을 처리하도록 구성될 수 있다. 일부 실시예에서, GPU(130)는 각각이 n개의 GPU 스레드(212)의 m개의 그룹으로 구성될 수 있다. n개의 GPU 스레드(212)의 그룹은 별도의 CU(134)에서 각각 실행될 수 있다. 예를 들어, n은 64일 수 있고 m은 24일 수 있다. 이후 64 * 24 또는 1536개의 GPU 스레드(212)가 있을 수 있다. n개의 GPU 스레드(212)의 그룹 각각에 대해 인바운드 큐(214)가 있을 수 있다. 예를 들어, 인바운드 큐(214.1)는 GPU 스레드(212.1 내지 212.n)에 의해 서비스될 수 있다. n개의 GPU 스레드(212)의 그룹은 단일 명령 다수의 데이터(SIMD) CU(134)일 수 있다. n개의 GPU 스레드(212)의 그룹은 동시에 요청(202)의 그룹을 처리할 수 있다. 예를 들어, GPU 스레드(212.1) 내지 GPU 스레드(212.n)(n=64)와 같은 n개의 GPU 스레드(212)의 그룹은 인바운드 큐(214.1)를 모니터링할 수 있고, 인바운드 큐(214.1)에서 64개의 요청(202)이 이용가능할 때 GPU 스레드(212.1) 내지 GPU 스레드(212.64)의 그룹은 64개의 요청(202)을 동시에 처리할 수 있다. 일부 실시예에서, n개의 GPU 스레드(212)의 그룹의 GPU 스레드(212) 중 하나는 n개의 GPU 스레드(212)의 그룹을 위한 인바운드 큐(214)를 모니터링할 수 있다. GPU 스레드(212)는 동일한 커널 또는 프로그램을 실행하거나, 또는 동일한 방식으로 요청(202)을 처리하도록 구성될 수 있다. GPU 스레드(212)는 응답(204)을 아웃바운드 큐(210)로 송신할 수 있다.

[0024] 인바운드 큐(214)는 요청(202)이 놓인 하나 이상의 큐일 수 있다. 인바운드 큐(214)는 메모리(138) 또는 다른 메모리에 존재할 수 있다. GPU 데이터 구조(218)는 GPU(130)와 연관된 메모리에 존재하는 데이터 구조(218)일 수 있다. GPU 데이터 구조(218)는 하나 이상의 설정(220)에 기초하여 구성될 수 있고 추가적인 정보에 기초할 수 있다. GPU 데이터 구조(218)는 CPU 데이터 구조(216)로부터 데이터(222)를 검색하는데 사용될 수 있는 포인터(224)일 수 있는 포인터(224)를 포함할 수 있다. GPU 데이터 구조(218)는 요청(202)을 처리하기 위해 GPU(130)에 의해 사용될 수 있다. 일부 실시예에서, GPU 데이터 구조(218)는 메모리(138) 및 또는 메모리(104), 또는 GPU(130)와 연관된 다른 메모리(미도시)에 존재할 수 있다.

[0025] CPU 데이터 구조(216)는 CPU(128)와 연관된 메모리에 존재하는 데이터 구조(216)일 수 있다. CPU 데이터 구조(216)는 하나 이상의 설정(220)에 기초하여 구성될 수 있고 추가적인 정보에 기초할 수 있다. CPU 데이터 구조(216)는 포인터(224)에 의해 지시되는 데이터(222)일 수 있는 데이터(222)를 포함할 수 있다. CPU 데이터 구조(216)는 요청(202)을 처리하기 위해 CPU(128)에 의해 사용될 수 있다. 일부 실시예에서, CPU 데이터 구조(216)는 메모리(136) 및 또는 메모리(104)(도 1 참조), 또는 CPU(128)와 연관된 다른 메모리(미도시)에 존재할 수 있다.

[0026] 도 3 및 도 4는 요청에 응답하는데 낮은 레이턴시를 요구할 수 있는 메모리 캐시 애플리케이션의 동작을 도시한 개략도이다. 도 3 및 도 4에는 클라이언트(302), 설정(220), 수신확인(330), 서버(390), 해쉬 테이블(326) 및 요청(202)과 응답(204)이 도시되어 있다. 클라이언트(302)는 서버(390)를 선택하고, 값(338.15)을 키(322.15)와 연관시키는 커맨드(command)일 수 있는 설정(220)을 선택된 서버(390.1)에 송신한다. 메모리 캐시 애플리케

이션(미도시)은 설정(220)을 수신하고 키(322.15)와 연관된 값(338.15)을 해쉬 테이블(326)일 수 있는 데이터 구조에 저장하고 수신확인(330)을 클라이언트(302)에 송신할 수 있다. 클라이언트(302)는 키(322.15)(도 4 참조)를 갖는 요청(202)을 송신할 수 있고, 서버(390.1)는 해쉬 테이블(326)을 검색하는 것에 의해 키(322.15)와 연관된 값(338.15)을 갖는 응답(204)을 송신한다.

[0027] 클라이언트(302)는 LAN 또는 인터넷(미도시)과 같은 통신 네트워크를 통해 서버(390.1)와 통신할 수 있다. 일부 실시예에서, 클라이언트(302)는 서버(390.1)에 존재할 수 있다. 설정(220)은 키(322)와 값(338)의 쌍(324)을 포함하는 커맨드일 수 있다. 키(322)와 값(338)은 데이터일 수 있다. 키(322)는 값(338)을 식별하는 고유한 방식일 수 있다. 수신확인(330)은 설정(220)이 성공적이었는지 아닌지 여부를 나타내는 지시일 수 있다. 해쉬 테이블(326)은 인덱스(328)를 키(322)와 값(338)의 쌍(324)과 연관시키는 테이블일 수 있다.

[0028] 클라이언트(302)는 서버(390)를 선택할 수 있다. 일부 실시예에서, 클라이언트(302)는 키(322)에 기초하여 서버(390)를 선택한다. 예를 들어, 클라이언트(302)는 키(322)의 모듈러스(modulus)와 같은 키(322)의 해쉬값을 결정한 것에 기초하여 서버(390)를 결정할 수 있다. 예를 들어, 서버(390)는 도 3에 도시된 바와 같이 3개의 서버(390)가 있을 때 (키(322) 모듈러스 3) + 1의 값을 결정한 것에 기초하여 선택될 수 있다.

[0029] 클라이언트(302)는 설정(220)을 서버(390.1)에 송신할 수 있다. 메모리 캐시 애플리케이션(미도시)은 키(322)의 인덱스(328)를 결정할 수 있는데, 이것은 일부 실시예에서 해쉬 값 결정이라고 지칭된다. 예를 들어, 해쉬 테이블이 9개의 엔트리인 경우 메모리 캐시 애플리케이션은 30의 값을 갖는 키(322)가 [30 mod 9] +1 = 4의 해쉬 값 또는 인덱스를 구비할 수 있도록 [키(322) 모듈러스 9] + 1이도록 인덱스를 결정할 수 있다. 메모리 캐시 애플리케이션은 키(322.15)와 값(338.15)의 쌍(324.15)을 해쉬 테이블(326)에 저장할 수 있다. 인덱스(328) 각각은 쌍(324)을 검색하기 위해 횡단될 필요가 있는 쌍(324)의 체인을 구비할 수 있다.

[0030] 이런 방식으로 클라이언트(302)는 서버(390.1)가 키(322)와 값(338)의 쌍(324)을 저장하는 해쉬 테이블(326)을 구축할 수 있게 한다. 클라이언트(302)는 전술한 바와 같이 키(322)에 기초하여 서버(390)를 선택하는 것에 의해 키(322)와 연관된 값(338)을 검색하고 나서 키(322.15)(도 4 참조)를 갖는 서버(390.1)에 요청(202)을 송신할 수 있다. 서버(390.1)는 키(322.15)를 받고 키(322.15)의 해쉬 값을 연산하여 인덱스(328.4)를 결정하고 나서 키(322.15)를 갖는 쌍(324.15)에 대해 인덱스(328.4)와 연관된 쌍(324)을 검색한다. 키(322.15)가 발견되면, 메모리 캐시 애플리케이션은 값(338.15)을 검색하고 클라이언트(302)에 값(338.15)을 갖는 응답(204)을 송신한다.

[0031] 따라서, 클라이언트(302)는 키(322)와 값(338)의 220개의 쌍(324)을 해쉬 테이블(326)에 설정할 수 있고 키(322)를 사용하여 해쉬 테이블(326)로부터 202개의 값(338)을 요청할 수 있다. 일부 실시예에서, 해쉬 테이블(326)은 클 수 있고 해쉬 테이블(326)은 104, 136 또는 138(도 1 참조)과 같이 랜덤 액세스 메모리에 저장되어, 요청(202)이 신속히 수행될 수 있다. 일부 실시예에서, 설정(220) 및/또는 요청(202) 커맨드는, 클라이언트(302)가 값(338)을 요청(202)할 때와, 값(338)이 실제 응답(204)으로 리턴될 때 사이에 낮은 레이턴시가 존재하도록 신속히 수행되는 것이 중요할 수 있다. 예를 들어, 해쉬 테이블(326)은 매우 신속한 응답(204)을 요구하는 라우팅을 위한 네트워크 주소를 저장하는데 사용될 수 있다.

[0032] 도 5 및 도 6은 일부 개시된 실시예에 따라 메모리 캐시 애플리케이션을 위한 이종 프로세서를 사용하여 낮은 레이턴시의 애플리케이션을 위한 시스템의 동작을 도시한다. 도 5 및 도 6에는 클라이언트(302), 설정(220), 수신확인(330), 서버(390.1), CPU 데이터 구조(216), GPU 데이터 구조(218) 및 (도 6 참조) 요청(202)과 응답(204)이 도시되어 있다. CPU 데이터 구조(216)는 주소(570)와 값(338)을 갖는 데이터 구조일 수 있다. 이 값(338)은 주소(570)를 사용하여 설정되거나 검색될 수 있다. GPU 데이터 구조(218)는 인덱스(528)에 의해 액세스되는 키(322)와 주소(570)의 쌍(524)을 포함한다. 일부 실시예에서, GPU 데이터 구조(218)는 해쉬 테이블(527)일 수 있다.

[0033] 클라이언트(302)의 관점으로부터 메모리 캐시 애플리케이션의 동작은 도 3 및 도 4와 함께 설명된 것과 동일하다. 도 2 및 도 5를 참조하면, 동작 시, 클라이언트(302)는 설정(220)을 서버(390.1)에 송신한다. 서버(390.1)의 네트워크 스레드(206)는 설정(220)을 호스트 스레드(208)에 송신한다. 호스트 스레드(208)는 CPU 데이터 구조(216)의 주소(570.7)에 값(338.15)을 설정한다. 호스트 스레드(208)는 이후 키(322.15)와 주소(570.7)의 쌍(524.15)을 인바운드 큐(214)에 놓는다. 일부 실시예에서, 호스트 스레드(208)는 인바운드 큐(214)의 내용에 기초하여 쌍(524.15)을 놓아 둘 인바운드 큐(214)를 결정할 수 있다. GPU 스레드(212)는 키(322.15)에 대한 인덱스(528)를 결정하고 키(322.15)와 주소(570.7)의 쌍(524.15)을 해쉬 테이블(527)의 인덱스(528)에 놓는데, 예를 들어, 도 5에 도시된 인덱스(528.4)에 놓는다. 일부 실시예에서, 쌍(524)은 인덱스(528)와 연관된 링크된 리

스트로서 해쉬 테이블(527)에 저장될 수 있다. 설정(220)이 성공적이었다는 수신확인(330)이 클라이언트(302)로 송신될 수 있다.

[0034] 도 2 및 도 6을 참조하면, 클라이언트(302)는 요청(202)을 서버(390.1)에 송신할 수 있다. 네트워크 스레드(206)는 요청(202)을 수신할 수 있다. 네트워크 스레드(206)는 요청(202)을 호스트 스레드(208)에 송신할 수 있다. 호스트 스레드(208)는 요청(202)을 인바운드 큐(214)에 놓을 수 있다. GPU 스레드(212)는 키(322)에 대응하는 인덱스(528)를 결정하는 것에 의해 요청(202)을 처리할 수 있다. 도 6에 도시된 바와 같이, 인덱스(528.4)는 키(322.15)에 대응하고 GPU 스레드(212)는 리스트를 검색하여 키(322.15)와 주소(570.7)의 쌍(524.15)을 찾을 수 있다. GPU 스레드(212)는 키(322.15)와 주소(570.7)의 쌍(524.15)을 아웃바운드 큐(210)로 이동시킬 수 있다. 호스트 스레드(208)는 주소(570.7)를 사용하여 CPU 데이터 구조(216)로부터 값(338.15)을 검색할 수 있다. 호스트 스레드(208)는 값(338.15)을 갖는 응답(204)이 클라이언트(302)로 송신되어야 하는 것을 네트워크 스레드(206)에 나타낼 수 있다. 네트워크 스레드(206)는 값(338.15)을 갖는 응답(204)을 클라이언트(302)에 송신할 수 있다.

[0035] 일부 개시된 실시예에서, CPU 데이터 구조(216)는 메모리(136)에 존재할 수 있다. 일부 개시된 실시예에서, GPU 데이터 구조(218)는 메모리(138)에 존재할 수 있다. 일부 개시된 실시예는 다량의 데이터일 수 있는 값(338)이 메모리(138)와 같은 메모리로 전송될 필요가 없어 시간을 소비하지 않는다는 장점을 구비한다.

[0036] 일부 개시된 실시예에서, CPU(128)의 코어(132)보다 훨씬 더 많은 요청(202)이 있을 수 있다. 일부 개시된 실시예에서, 이 요청(202)의 수가 GPU(130)의 컴퓨팅 유닛(134)의 수 이상이 될 때까지 이 요청(202)의 수는 인바운드 큐(214)에 큐잉(queued)되고, 이후 하나 이상의 요청(202)이 GPU(130)의 컴퓨팅 유닛(134) 각각으로 할당된다.

[0037] 일부 개시된 실시예에서, CPU(128) 및 GPU(130)는 원자 판독/기록 명령을 사용하여 통신한다. 일부 개시된 실시예에서, GPU(130)는 메모리 위치를 폴링(poll)하여 CPU(128)에 의해 기록된 인바운드 큐 포인터를 취득한다. 일부 실시예에서, GPU(130)에서 실행되는 스레드들 중 하나의 스레드는 인바운드 큐(214)에 업데이트할 메모리 위치를 폴링할 수 있다. 일부 개시된 실시예에서, GPU(130)는 CPU(128)가 폴링하는 메모리 위치에 포인터를 기록하는 것에 의해 아웃바운드 큐(210)를 업데이트한다.

[0038] 일부 개시된 실시예에서, GPU 스레드(212)는 커널이 활성으로 유지되는 한, 활성으로 유지되는 지속적인 스레드(persistent thread)일 수 있다. 커널은 셧다운 메시지(shutdown message)에 응답하는 무한 외부 루프(infinite outer loop)를 구비할 수 있다. 일부 개시된 실시예에서, OpenCL은 GPU(130)의 컴퓨팅 유닛(134)마다 2개의 지속적인 스레드를 가지고 사용될 수 있다. 컴퓨팅 유닛(134)마다 2개의 지속적인 스레드는 제1 스레드가 데이터가 도달하기를 기다리는 동안 제2 스레드가 실행될 수 있는 장점을 제공할 수 있다.

[0039] 도 7은 이종 프로세서를 사용하여 낮은 레이턴시의 애플리케이션을 위한 시스템의 일 실시예를 개략적으로 도시한다. 시스템(700)은 CPU(128), 호스트 스레드(708), 아웃바운드 큐(210), GPU(130), GPU 스레드(212), 인바운드 큐(214)를 포함한다. 애플리케이션 스레드(706)는 요청(202)을 송신할 수 있고 시스템(700)은 요청(202)을 처리하고 응답(204)을 애플리케이션 스레드(706)에 송신할 수 있다.

[0040] 애플리케이션 스레드(706)는 CPU(128) 또는 다른 CPU(128)에서 실행되는 애플리케이션일 수 있다. 요청(202)은 처리 작업에 대한 요청일 수 있다. 예를 들어, 설정(220) 및 요청(202)은 도 3, 도 4, 도 5 및 도 6과 함께 메모리 캐시 애플리케이션과 함께 개시된다. 응답(204)은 요청(202)에 대한 응답일 수 있다. 예를 들어, 응답(204)은 메모리 캐시 애플리케이션과 함께 개시된 응답(204)일 수 있다.

[0041] 호스트 스레드(708)는, 요청(202)을 수신하고 응답(204)을 송신하는 스레드일 수 있다. 일부 실시예에서, 애플리케이션 스레드(706)는 암호 애플리케이션, 네트워크 애플리케이션 및 내장된 애플리케이션일 수 있다.

[0042] 도 8은 일부 개시된 실시예에 따라 GPU가 실행할 수 있는 커널을 도시한다. GPU 스레드(212)(도 2 참조)는 제너릭 애플리케이션이 GPU 스레드(212)에 의해 실행될 수 있게 하는 gpu제너릭애플리케이션(gpuGenericApplication) 커널(800)을 실행하는 것일 수 있다. 커널(800)은 OpenCL(등록상표)에 있는 EOS와 같은 차단 신호(breaking signal)에 의해 차단될 수 있는 (804) 내지 (816)의 무한 do 루프 흐름 제어를 구비한다. 커널(800)은 (805)에서 인바운드 큐(214)일 수 있는 _in_control_queue를 판독하고 제1 요청(202)에 대한 read_ptr을 _in_control_queue에 설정한다. 커널(800)은 (806)에서 _in_control_queue에 요청이 있는 동안 while 루프를 수행한다. 커널(800)은 curr_ptr에 의해 지시된 요청(202)에 따라 (808)에서 애플리케이션을 호출한다. 커널(800)은 (810)에서 아웃바운드 큐(210)일 수 있는 _out_control_queue에 응답을 둔다. 커널

(800)은 (812)에서 curr_ptr++을 증분시킨다. 커널(800)은 curr_ptr이 서비스될 필요가 있는 in_control_queue에서 요청(202)을 지시하는 경우 (806)으로 백루프를 수행한다. curr_ptr이 요청(202)을 지시하지 않는 경우, 커널(800)은 (814)로 진행하고 protocol_control을 업데이트한다. 커널(800)은 차단 신호 EOS가 (816)에서 수신되었는지 여부를 (816)에서 체크한다. 차단 신호 EOS가 수신된 경우, 커널(800)이 종료한다. 그렇지 않은 경우, 커널(800)은 (804)로 백루프를 수행된다. 커널(800)은 한번 호출될 수 있고 많은 요청(202)에 응답하여 지속적으로 유지될 수 있다는 장점을 구비한다.

[0043] 도 9는 일부 개시된 실시예에 따라 이종 프로세서를 위한 낮은 레이턴시 애플리케이션을 위한 시스템을 호출하는 데이터 구조 및 호를 도시한다. LOLALY_REQ 라고 명명된 구조(902)는 애플리케이션 id(903) 및 콜백 함수(904)에 식재될 수 있다. 이후 lolalySendRequest(906)를 사용하여 시스템(700)에 요청(202)을 수행할 수 있다.

[0044] 도 10은 메모리 캐시 애플리케이션을 위한 이종 프로세서를 사용하여 낮은 레이턴시를 제공하는 시스템 및 방법의 경험 테스트의 결과 테이블을 도시한다. 테이블(1000)은 요청 사이즈(1002) 및 이후 2개의 데이터 설정(하나는 AMD(등록상표)의 제품인 APU 브라조스(Brazos)(상표명) HD631/430(1004)에 대한 것이고 다른 하나는 AMD(등록상표)의 제품인 AP 트리니티(Trinity)(상표명), HD7660/600에 대한 것)를 포함한다. 예를 들어, 2048개의 요청(1020)에서 APU 브라조스(상표명)는 0.19의 처리량 MR(1006)을 구비하고 3.23의 대역폭 GB(1008)을 구비하는 197μ초의 레이턴시(1010)를 구비한다. 2048개의 요청(1020)에서는, AP 트리니티(상표명)은 0.31의 처리량 MR(1014) 및 5.26의 대역폭 GB(1016)을 구비하는 140μ초의 레이턴시(1018)를 구비한다. 따라서 심지어 다수의 요청이 있는 경우에도 시스템은 허용가능한 레이턴시를 제공한다.

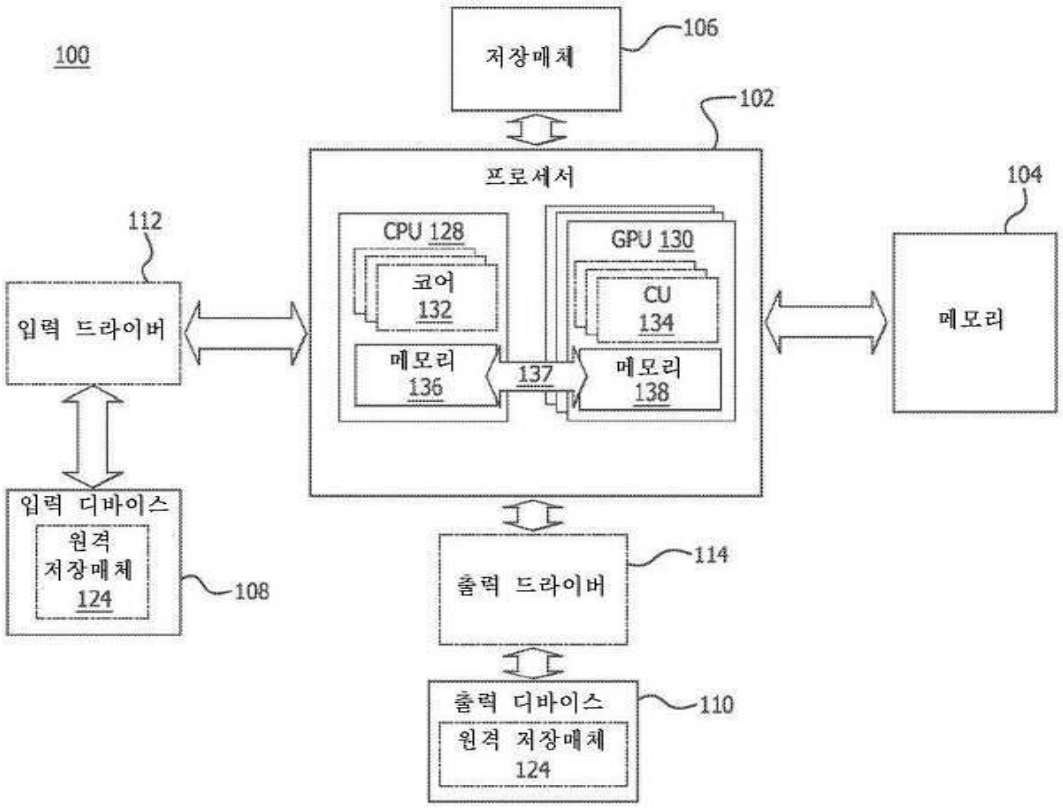
[0045] 많은 변형이 본 명세서에 기초하여 가능한 것으로 이해된다. 특징 및 요소는 특정 조합으로 전술되었으나, 각 특징 또는 요소는 다른 특징 및 요소 없이 단독으로 또는 다른 특징과 요소와 함께 또는 다른 특징과 요소 없이 여러 조합으로 사용될 수 있다.

[0046] 제공된 방법은 범용 컴퓨터, 프로세서 또는 프로세서 코어에서 구현될 수 있다. 적절한 프로세서는, 예로서, 범용 프로세서, 그래픽 처리 유닛(GPU), 특수 목적 프로세서, 종래의 프로세서, 디지털 신호 프로세서(DSP), 복수의 마이크로프로세서, DSP 코어와 연관된 하나 이상의 마이크로프로세서, 제어기, 마이크로제어기, 응용 특정 집적 회로(ASIC), 전계 프로그래밍가능한 게이트 어레이(FPGA) 회로, 임의의 다른 유형의 집적 회로(IC) 및/또는 상태 기계를 포함할 수 있다. 이러한 프로세서는 네트리스트를 포함하는 처리된 하드웨어 설명 언어(HDL) 명령 및 다른 중간 데이터(이러한 명령은 컴퓨터 판독가능한 매체에 저장될 수 있다)의 결과를 사용하여 제조 공정을 구성함으로써 제조될 수 있다. 이러한 처리 결과는 개시된 실시예의 측면을 구현하는 프로세서를 제조하는 반도체 제조 공정에서 사용되는 마스크작업일 수 있다.

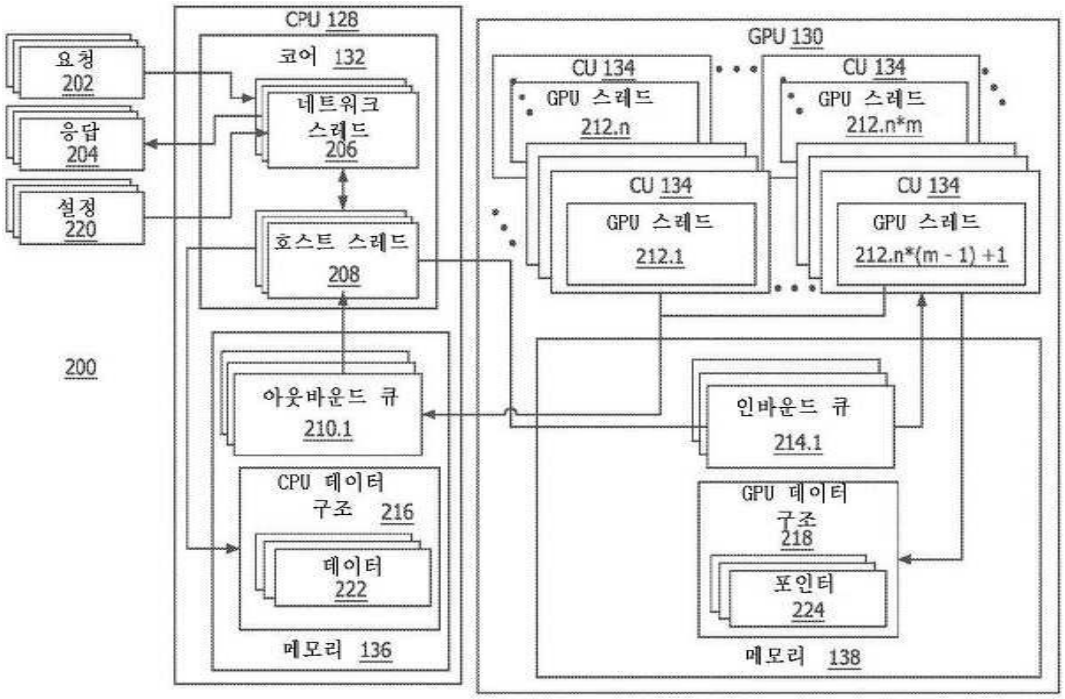
[0047] 본 명세서에 제공된 방법 또는 흐름도는 범용 컴퓨터 또는 프로세서에 의해 실행하기 위해 컴퓨터-판독가능한 저장 매체에 병합된 컴퓨터 프로그램, 소프트웨어 또는 펌웨어로 구현될 수 있다. 일부 실시예에서, 컴퓨터-판독가능한 저장 매체는 비-일시적인 컴퓨터-판독가능한 저장 매체이다. 컴퓨터-판독가능한 저장 매체의 예는 판독 전용 메모리(ROM), 랜덤 액세스 메모리(RAM), 레지스터, 캐시 메모리, 반도체 메모리 디바이스, 자기 매체, 예를 들어, 내부 하드 디스크 및 이동식 디스크, 광자기 매체 및 광 매체, 예를 들어, CD-ROM 디스크 및 DVD(digital versatile disk)이다.

도면

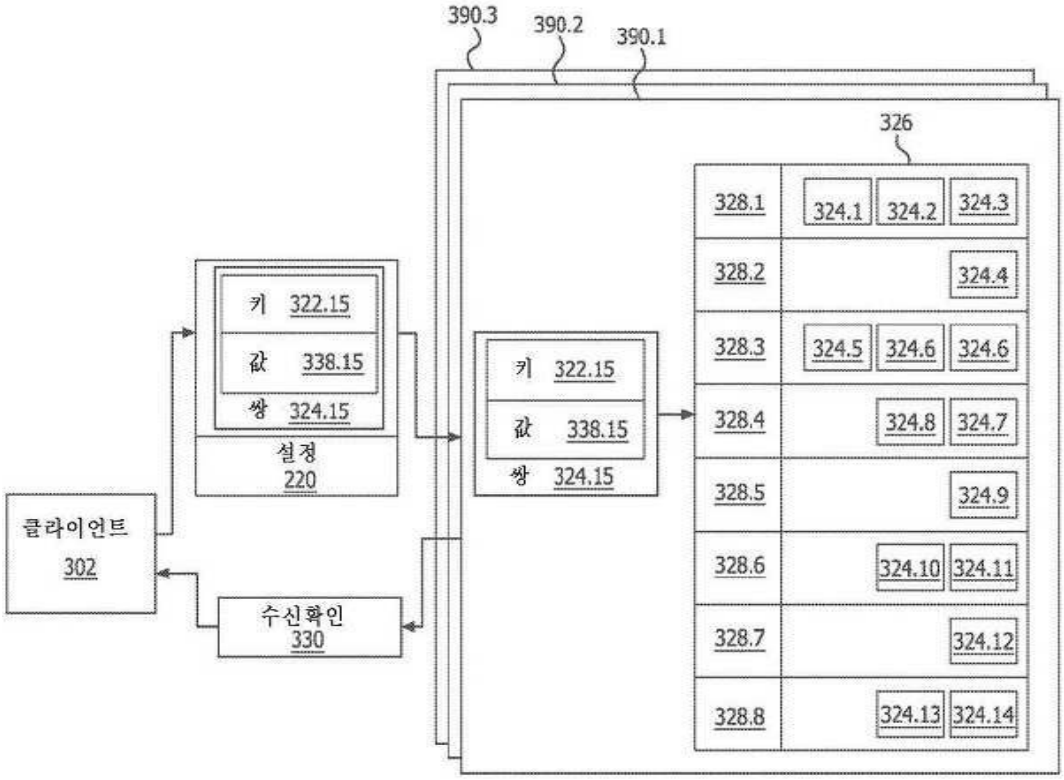
도면1


100
저장매체
106
102
프로세서
104
112
CPU 128
코어
132
GPU 130
CU
134
입력 드라이버
메모리
136
137
메모리
138
메모리
입력 디바이스
원격
저장매체
124
108
출력 드라이버
114
출력 디바이스
110
원격 저장매체
124

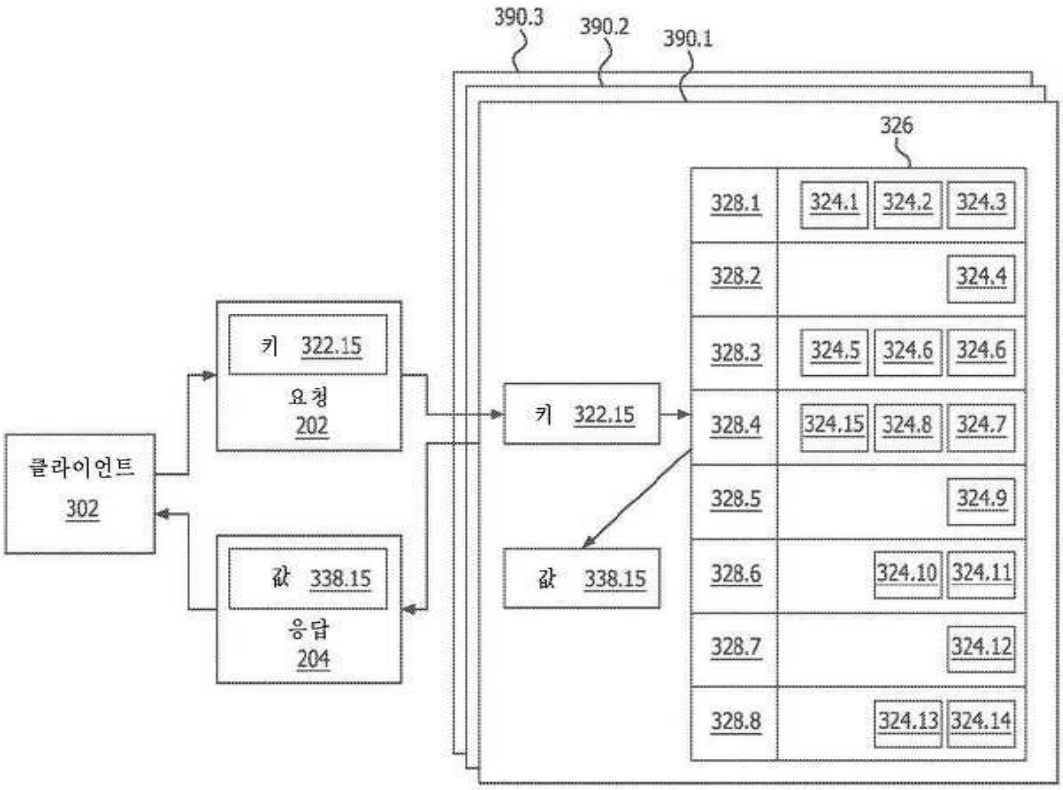
도면2


CPU 128
코어 132
요청
202
응답
204
설정
220
네트워크
스레드
206
호스트 스레드
208
200
아웃바운드 큐
210.1
CPU 데이터
구조 216
데이터
222
메모리 136
GPU 130
CU 134
GPU 스레드
212.n
CU 134
GPU 스레드
212.n*m
CU 134
GPU 스레드
212.1
CU 134
GPU 스레드
212.n*(m - 1) +1
인바운드 큐
214.1
GPU 데이터
구조
218
포인터
224
메모리 138

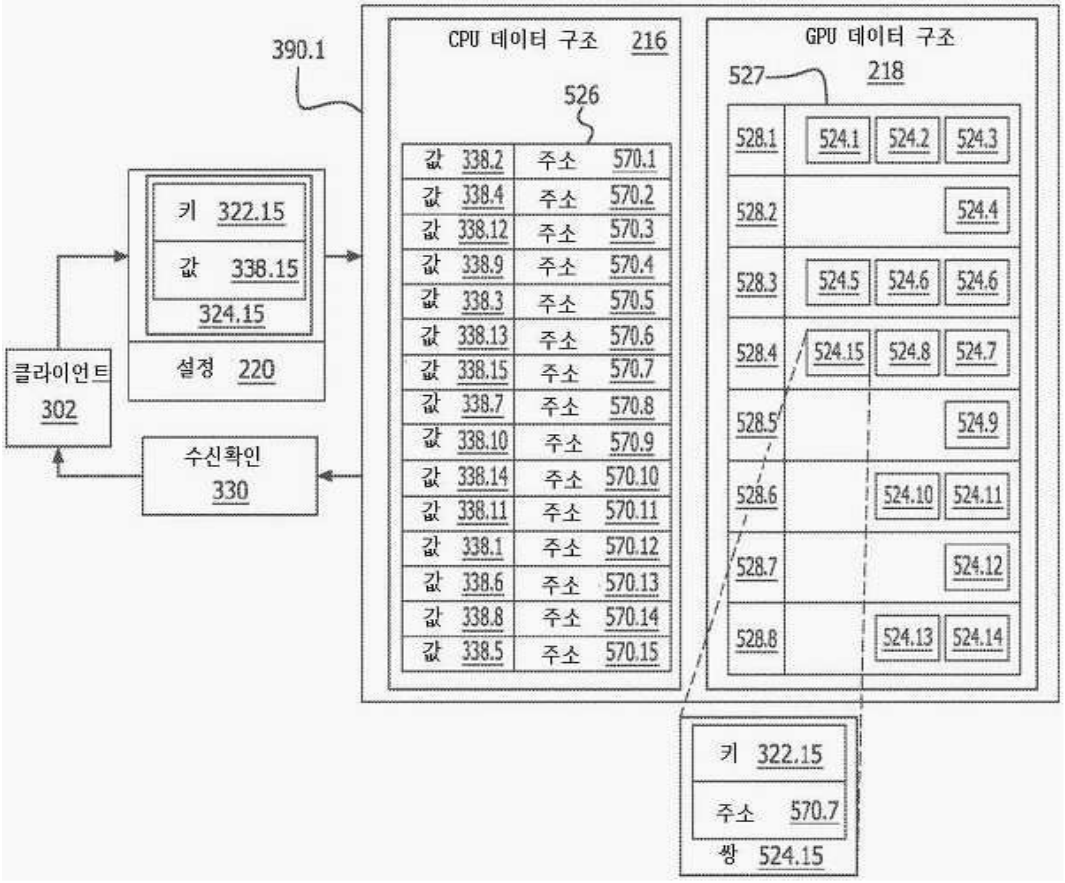
도면3


390.3
390.2
390.1
326
328.1
324.1
324.2
324.3
328.2
324.4
328.3
324.5
324.6
324.6
328.4
324.8
324.7
328.5
324.9
328.6
324.10
324.11
328.7
324.12
328.8
324.13
324.14
키 322.15
값 338.15
쌍 324.15
설정 220
클라이언트 302
수신확인 330

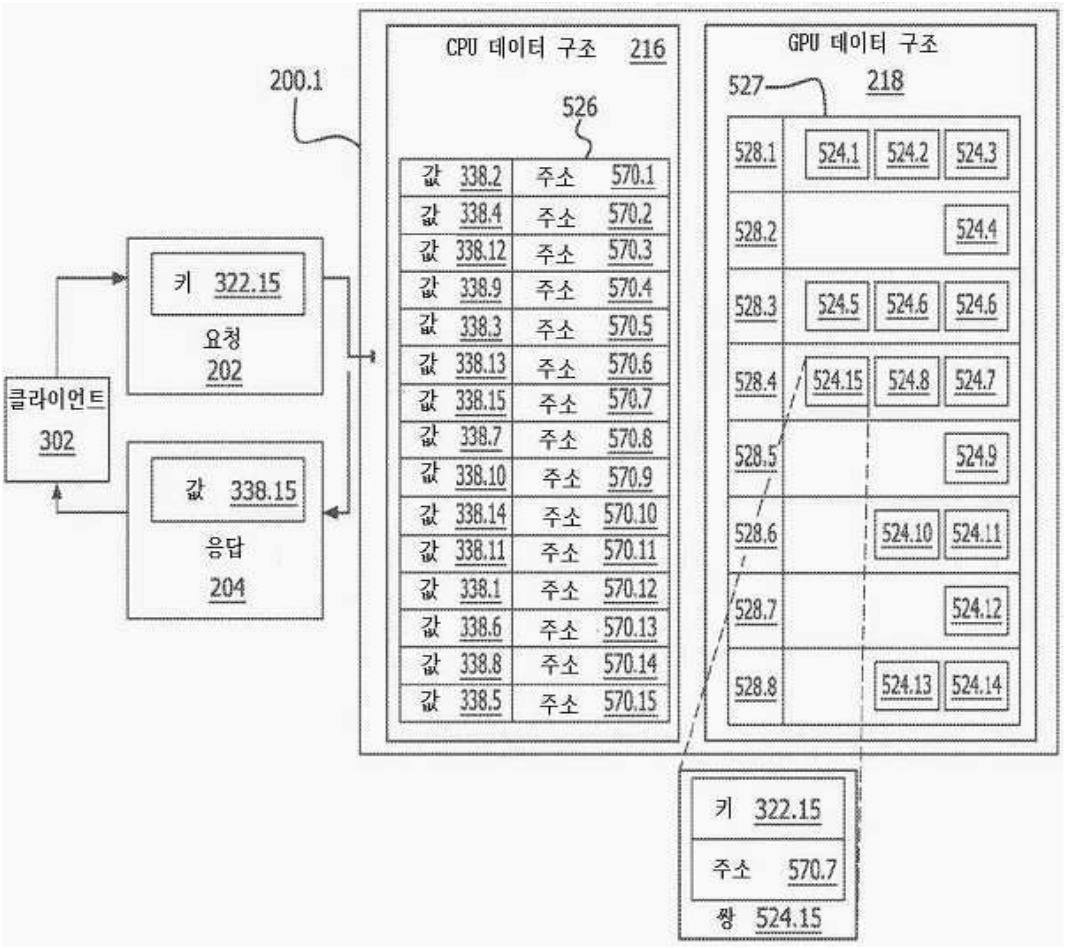
도면4


390.3
390.2
390.1
326
328.1
324.1
324.2
324.3
328.2
324.4
328.3
324.5
324.6
324.6
328.4
324.15
324.8
324.7
328.5
324.9
328.6
324.10
324.11
328.7
324.12
328.8
324.13
324.14
키 322.15
요청 202
키 322.15
값 338.15
응답 204
클라이언트 302

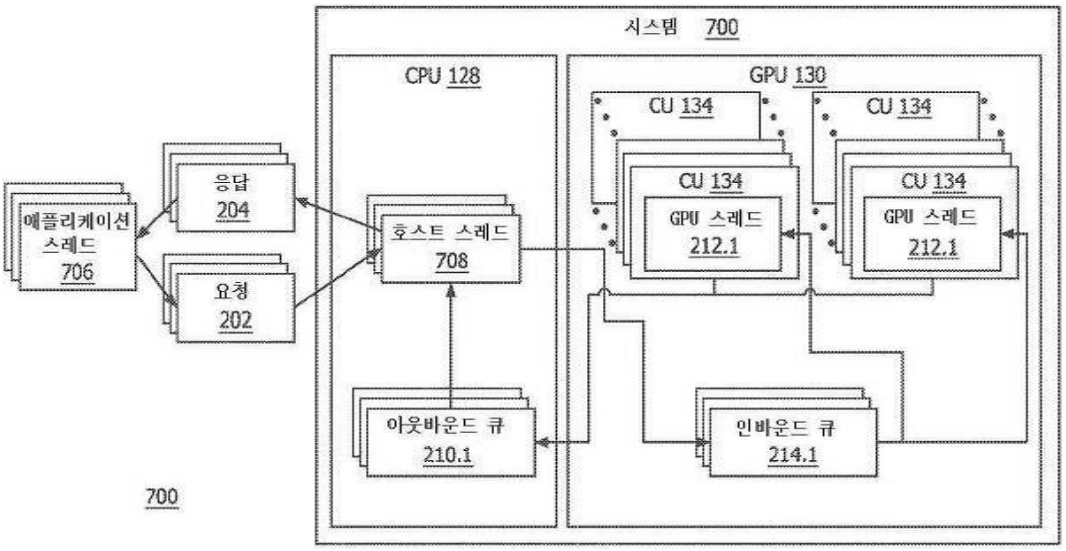
도면5


390.1
CPU 데이터 구조 216
526
값 338.2 주소 570.1
값 338.4 주소 570.2
값 338.12 주소 570.3
값 338.9 주소 570.4
값 338.3 주소 570.5
값 338.13 주소 570.6
값 338.15 주소 570.7
값 338.7 주소 570.8
값 338.10 주소 570.9
값 338.14 주소 570.10
값 338.11 주소 570.11
값 338.1 주소 570.12
값 338.6 주소 570.13
값 338.8 주소 570.14
값 338.5 주소 570.15
GPU 데이터 구조
527
218
528.1 524.1 524.2 524.3
528.2 524.4
528.3 524.5 524.6 524.6
528.4 524.15 524.8 524.7
528.5 524.9
528.6 524.10 524.11
528.7 524.12
528.8 524.13 524.14
키 322.15
값 338.15
324.15
설정 220
클라이언트 302
수신확인 330
키 322.15
주소 570.7
쌍 524.15

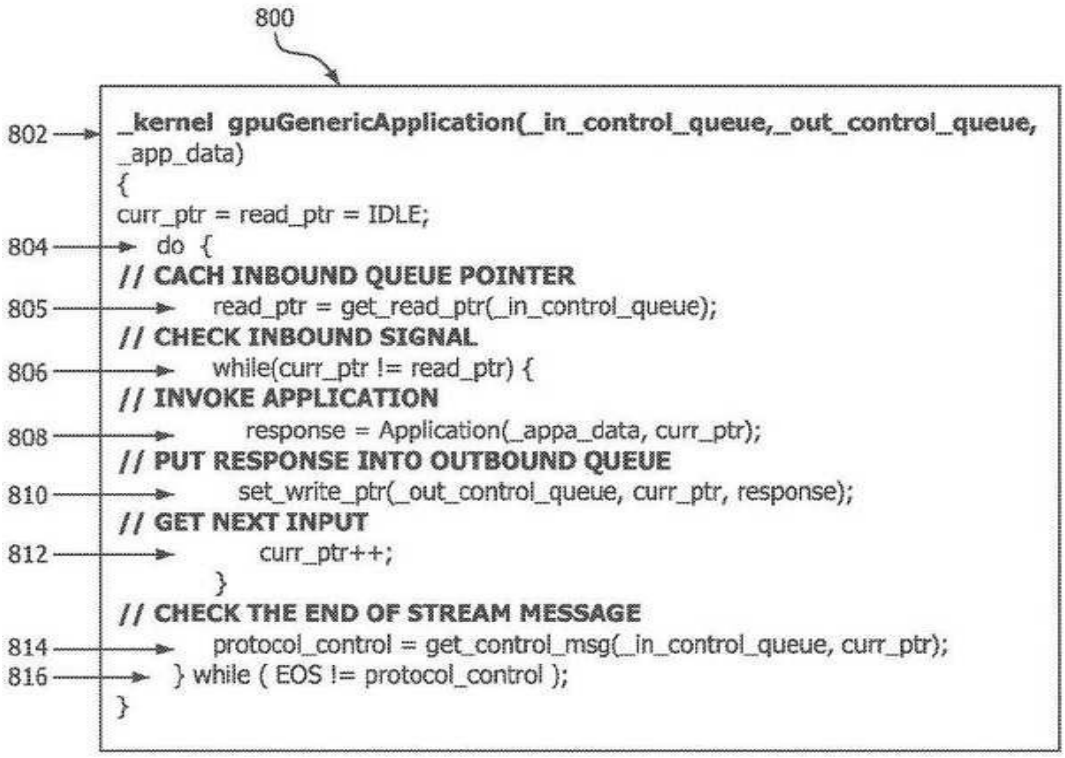
도면6


CPU 데이터 구조 216
526
200.1
값 338.2 주소 570.1
값 338.4 주소 570.2
값 338.12 주소 570.3
값 338.9 주소 570.4
값 338.3 주소 570.5
값 338.13 주소 570.6
값 338.15 주소 570.7
값 338.7 주소 570.8
값 338.10 주소 570.9
값 338.14 주소 570.10
값 338.11 주소 570.11
값 338.1 주소 570.12
값 338.6 주소 570.13
값 338.8 주소 570.14
값 338.5 주소 570.15
GPU 데이터 구조
527
218
528.1 524.1 524.2 524.3
528.2 524.4
528.3 524.5 524.6 524.6
528.4 524.15 524.8 524.7
528.5 524.9
528.6 524.10 524.11
528.7 524.12
528.8 524.13 524.14
키 322.15
요청
202
클라이언트
302
값 338.15
응답
204
키 322.15
주소 570.7
쌍 524.15

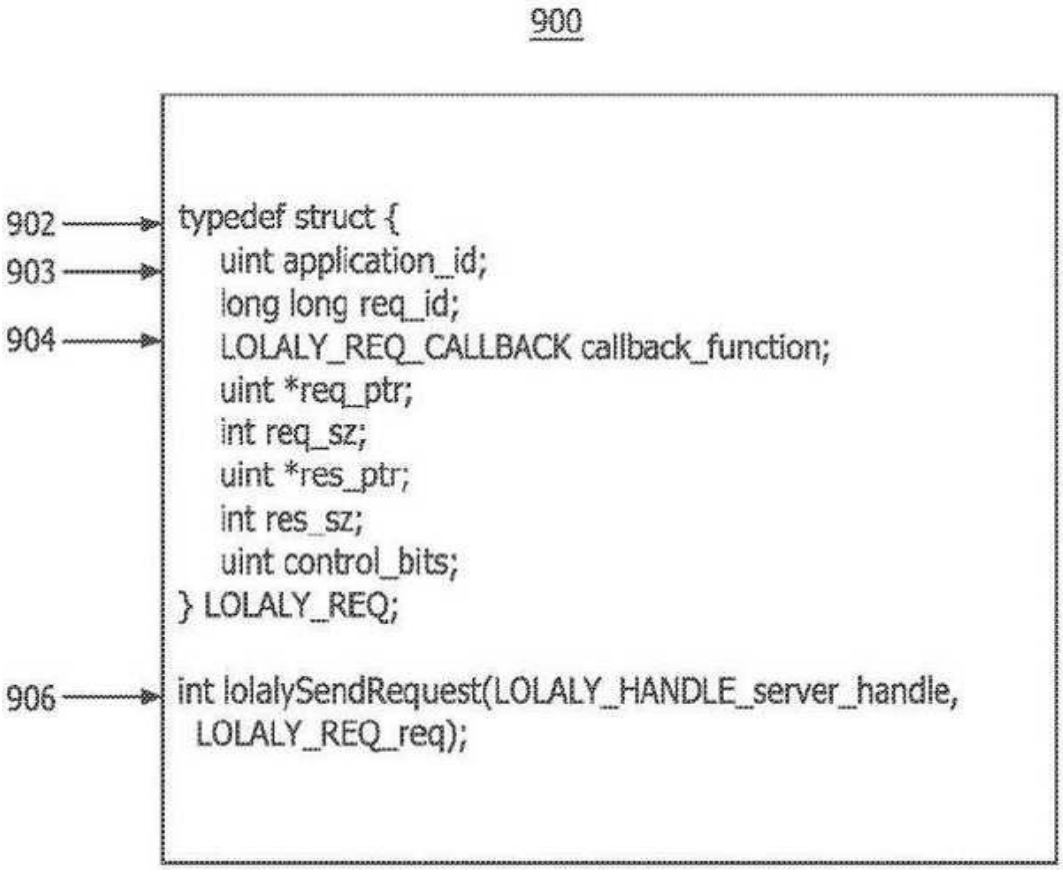
도면7


시스템 700
CPU 128
GPU 130
CU 134
CU 134
CU 134
CU 134
GPU 스레드
212.1
GPU 스레드
212.1
응답
204
요청
202
애플리케이션
스레드
706
호스트 스레드
708
아웃바운드 큐
210.1
인바운드 큐
214.1
700

도면8


800
802
804
805
806
808
810
812
814
816
_kernel gpuGenericApplication(_in_control_queue,_out_control_queue,
_app_data)
{
curr_ptr = read_ptr = IDLE;
do {
// CACH INBOUND QUEUE POINTER
read_ptr = get_read_ptr(_in_control_queue);
// CHECK INBOUND SIGNAL
while(curr_ptr != read_ptr) {
// INVOKE APPLICATION
response = Application(_appa_data, curr_ptr);
// PUT RESPONSE INTO OUTBOUND QUEUE
set_write_ptr(_out_control_queue, curr_ptr, response);
// GET NEXT INPUT
curr_ptr++;
}
// CHECK THE END OF STREAM MESSAGE
protocol_control = get_control_msg(_in_control_queue, curr_ptr);
} while (EOS != protocol_control);
}

도면9


900
902
903
904
906
typedef struct {
uint application_id;
long long req_id;
LOLALY_REQ_CALLBACK callback_function;
uint *req_ptr;
int req_sz;
uint *res_ptr;
int res_sz;
uint control_bits;
} LOLALY_REQ;
int lolalySendRequest(LOLALY_HANDLE_server_handle,
LOLALY_REQ_req);

도면10

1000

1002 요청 사이즈	1004 GPU/주파수	1006 처리량 MRs	1008 대역폭 GBs	1010 평균 레이턴시 US	1012 GPU/주파수	1014 처리량 MRs	1016 대역폭 GBs	1018 평균 레이턴시 US
	APU 브라조스, HD6310/430				APU 트리니티 HD7660/600			
128		1.02	1.56	85.4		2.43	3.74	42
256		0.76	1.96	97.3		1.87	4.78	71.5
512		0.41	1.9	107.6		1.17	5.4	88.8
1024		0.25	2.18	150.2		0.52	4.5	110
1020 → 2048		0.19	3.23	197		0.31	5.26	140
4096		0.06	2.15	492		0.16	5.52	165.4

【심사관 직권보정사항】

【직권보정 1】

【보정항목】 청구범위

【보정세부항목】 청구항 11

【변경전】

상기 키 값 및 상기 대응 위치를

【변경후】

상기 키 값 및 상기 대응하는 위치를