(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2023-0054551
(43) 공개일자 2023년04월25일

(51) 국제특허분류(Int. Cl.)
G06F 9/30 (2018.01) *G06F 13/16* (2006.01)
G06F 16/2455 (2019.01) *G06F 9/38* (2006.01)
G06N 3/08 (2023.01) *G06Q 50/00* (2018.01)
G06Q 50/10 (2012.01)
(52) CPC특허분류
G06F 9/3004 (2013.01)
G06F 13/1663 (2013.01)
(21) 출원번호 10-2021-0137615
(22) 출원일자 2021년10월15일
심사청구일자 2021년10월15일

(71) 출원인
인하대학교 산학협력단
인천광역시 미추홀구 인하로 100(용현동, 인하대학교)
(72) 발명자
이채은
서울특별시 서초구 남부순환로 2183(방배동, 방배래미안타워) 102동 2102호)
조명근
경기도 수원시 권선구 금호로200번길10-7(탑동)202호
(74) 대리인
특허법인아이더스

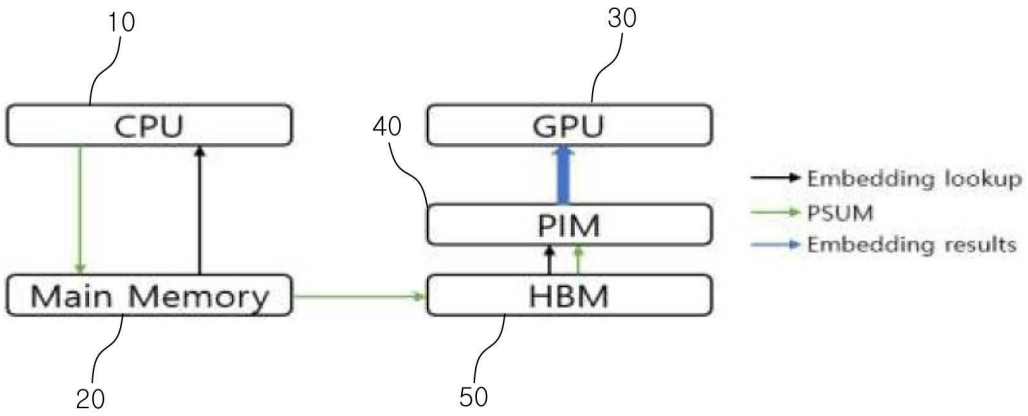
전체 청구항 수 : 총 6 항

(54) 발명의 명칭 **추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 구조 및 방법**

(57) 요 약

본 발명은 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 구조 및 방법에 관한 것으로, 추천 시스템의 임베딩시 기존 시스템의 한계를 극복할 수 있도록 니어-메모리 구조를 적용하여 메모리 용량과 대역폭의 부족을 개선시키고 데이터 이동량을 줄여 처리 속도를 향상시키는 하이브리드 니어-메모리 프로세싱 구조 및 방법을 제공한다.

대 표 도 - 도3



(52) CPC특허분류
G06F 16/24561 (2019.01)
G06F 9/30036 (2013.01)
G06F 9/3877 (2013.01)
G06N 3/08 (2023.01)
G06Q 50/01 (2013.01)
G06Q 50/10 (2015.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1415173072
과제번호	20010582
부처명	산업통상자원부
과제관리(전문)기관명	한국산업기술평가관리원
연구사업명	전자부품산업기술개발(R&D)
연구과제명	CMOS Image Sensor 영상 처리를 위한 저전력 deep learning HW IP 설계 기술 개발
기 여 율	1/1
과제수행기관명	인하대학교산학협력단
연구기간	2021.01.01 ~ 2021.12.31

명 세 서

청구범위

청구항 1

추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 구조에 있어서,

임베딩 벡터들(embedding vectors)의 임베딩 룩업(embedding lookup)이 일어나는 메인 메모리(Main Memory) 및 HBM와, 상기 룩업(lookup)된 임베딩 벡터들(embedding vectors)은 임베딩 축소(embedding reduction)가 이루어지는 CPU 및 PIM를 포함하며;

상기 PIM는 상기 PSUM(Partial SUM)에 따른 임베딩 결과(embedding result)를 만드는 것을 특징으로 하는 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 구조.

청구항 2

제 1항에 있어서,

상기 HBM에는 임베딩 룩업(embedding lookup) 횟수를 기준으로 접근이 많이 일어나는 임베딩 벡터들(embedding vectors)이 위치하고, 상기 CPU의 메인메모리에는 접근이 적게 일어나는 임베딩 벡터들(embedding vectors)이 위치하는 것을 특징으로 하는 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 구조.

청구항 3

제 1항에 있어서,

상기 임베딩 결과(embedding result)는 Top MLP 과정이 진행되도록 GPU로 전달되는 것을 특징으로 하는 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 구조.

청구항 4

추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 방법에 있어서,

메인 메모리(Main Memory)와 HBM에서 각각 임베딩 룩업(embedding lookup)이 일어나는 제1단계;

상기 룩업(lookup)된 임베딩 벡터들(embedding vectors)은 CPU와 PIM에서 임베딩 축소(embedding reduction)를 통해 PSUM(Partial SUM)이 발생하는 제2단계; 및

상기 PIM에서 상기 PSUM(Partial SUM)에 따른 임베딩 결과(embedding result)를 만드는 제3단계;를 포함하는 것을 특징으로 하는 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 방법.

청구항 5

제 4항에 있어서,

상기 제1단계 이전에 임베딩 룩업(embedding lookup) 횟수를 기준으로 접근이 많이 일어나는 임베딩 벡터들(embedding vectors)은 HBM에 위치하고, 나머지 임베딩 벡터들(embedding vectors)은 CPU의 메인메모리에 위치시키는 제4단계;를 더 포함하는 것을 특징으로 하는 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 방법.

청구항 6

제 4항에 있어서,

상기 제1단계 이후에 상기 임베딩 결과(embedding result)는 Top MLP 과정이 진행되도록 GPU로 전달되는 제5단계;를 더 포함하는 것을 특징으로 하는 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 구조 및 방법에 관한 것으로서, 더욱 상세하게는 추천 시스템의 임베딩시 기존 시스템의 한계를 극복하여 처리 속도를 향상시킬 수 있는 하이브리드 니어-메모리 프로세싱 구조 및 방법에 관한 것이다.

배 경 기 술

[0003] 최근 딥러닝 기술에 대한 관심이 높아지면서 다양한 분야에서 딥러닝 기술이 활용되고 있다. 딥러닝은 여러 층을 가진 인공신경망(ANN; Aatificial Neural Network)을 사용하여 기계학습(ML; Machine Learning)을 하는 것으로 심층학습이라고도 한다.

[0004] 특히, 딥러닝 기술을 적용시킨 추천 시스템(Recommendation System)은 개인화된 서비스를 선호하는 산업계의 동향과 맞물려 더욱 주목을 받고 있다.

[0005] 이러한 추천 시스템은 SNS(Social Network Service), 뉴스, 미디어 등 다양한 분야에서 활용되고 있으며, 딥러닝 기술을 이용한 대표적인 추천 시스템으로는 페이스북(Facebook)의 DLRM(Deep Learning Recommendation Models)(참고문헌 1 참고)이 있다.

[0006] 도 1은 페이스북(Facebook)의 DLRM의 구조를 도시한 도면이다. 이에 의하면 DLRM은 Top MLP(Multi-Layer Perceptron)(1), Bottom MLP(2), 임베딩 테이블(embedding table)(3), 및 이들의 특성 상호작용(feature interaction)(4)을 포함한다.

[0007] 이러한 DLRM 모델의 경우 밀집 특성(dense feature)과 희소 특성(sparse feature)을 이용하여 사용자에게 개인화된 콘텐츠를 제공한다.

[0008] 그리고, 상기 DLRM 모델은 다양한 특성(feature)들을 임베딩(embedding)을 통해 저장 및 처리되며, 모델의 정확도를 높이기 위해 수십 ~ 수백 기가바이트(GB; Giga Bytes)정도의 임베딩 테이블(embedding table)(3)을 포함한다.

[0009] 또한, 이 과정에서 임베딩 테이블 룩업(embedding table lookup) 작업은 불규칙한 메모리 접근(memory access)을 발생시킨다. 이러한 특성으로 인해 추천 시스템의 임베딩(embedding) 작업은 메모리 용량 부족과 메모리 병목 현상을 일으키며, 메모리 집약적인 작업이라고 할 수 있다.

선행기술문헌

비특허문헌

[0011] (비특허문헌 0001) 참고문헌 1 : M Naumov, et al. "Deep Learning Recommendation Model for Personalization and Recommendation Systems," In arxivorg, 2019

(비특허문헌 0002) 참고문헌 2 : https://www.kaggle.com/c/criteo-display-ad-challenge

발명의 내용

해결하려는 과제

[0012] 따라서, 본 발명은 이러한 문제점들을 해결하기 위한 것으로서 본 발명은 추천 시스템의 임베딩시 기존 시스템의 한계를 극복할 수 있는 하이브리드 니어-메모리 프로세싱 구조 및 방법을 제공하는데 그 목적이 있다.

[0013] 특히, 본 발명은 니어-메모리 구조를 적용하여 메모리 용량과 대역폭의 부족을 개선시키고 데이터 이동량을 줄여 처리 속도를 향상시키는 하이브리드 니어-메모리 프로세싱 구조 및 방법을 제공하는데 그 목적이 있다.

과제의 해결 수단

[0015] 이와 같은 기술적 과제를 해결하기 위해 본 발명은;

[0016] 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 구조에 있어서, 임베딩 벡터들(embedding vectors)의 임베딩 룩업(embedding lookup)이 일어나는 메인 메모리(Main Memory) 및 HBM과, 상기 룩업(lookup)된 임베딩 벡터들(embedding vectors)은 임베딩 축소(embedding reduction)가 이루어지는 CPU 및 PIM를 포함하며; 상기 PIM는 상기 PSUM(Partial SUM)에 따른 임베딩 결과(embedding result)를 만드는 것을 특징으로 하는 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 구조를 제공한다.

[0017] 이때, 상기 HBM에는 임베딩 룩업(embedding lookup) 횟수를 기준으로 접근이 많이 일어나는 임베딩 벡터들(embedding vectors)이 위치하고, 상기 CPU의 메인메모리에는 접근이 적게 일어나는 임베딩 벡터들(embedding vectors)이 위치하는 것을 특징으로 한다.

[0018] 그리고, 상기 임베딩 결과(embedding result)는 Top MLP 과정이 진행되도록 GPU로 전달되는 것을 특징으로 한다.

[0020] 또한, 본 발명은;

[0021] 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 방법에 있어서, 메인 메모리(Main Memory)와 HBM에서 각각 임베딩 룩업(embedding lookup)이 일어나는 제1단계; 상기 룩업(lookup)된 임베딩 벡터들(embedding vectors)은 CPU와 PIM에서 임베딩 축소(embedding reduction)를 통해 PSUM(Partial SUM)이 발생하는 제2단계; 및 상기 PIM에서 상기 PSUM(Partial SUM)에 따른 임베딩 결과(embedding result)를 만드는 제3단계;를 포함하는 것을 특징으로 하는 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 방법도 제공한다.

[0022] 이때, 상기 제1단계 이전에 임베딩 룩업(embedding lookup) 횟수를 기준으로 접근이 많이 일어나는 임베딩 벡터들(embedding vectors)은 HBM에 위치하고, 나머지 임베딩 벡터들(embedding vectors)은 CPU의 메인메모리에 위치시키는 제4단계;를 더 포함하는 것을 특징으로 한다.

[0023] 그리고, 상기 제1단계 이후에 상기 임베딩 결과(embedding result)는 Top MLP 과정이 진행되도록 GPU로 전달되는 제5단계;를 더 포함하는 것을 특징으로 한다.

발명의 효과

[0025] 본 발명에 따르면, 하이브리드 니어-메모리 프로세싱 구조를 적용하여 메모리 용량과 대역폭의 부족을 개선시키고 데이터 이동량을 줄여 처리 속도를 향상시키는 효과가 있다.

[0026] 특히, 본 발명에 따르면 메인 메모리와 HBM을 같이 사용함으로써 메모리의 용량 부족 문제를 해결할 수 있을 뿐만 아니라 CPU와 PIM에서 데이터를 임베딩 조정(embedding manipulate)을 함으로써 데이터 이동량을 줄일 수 있고, HBM의 넓은 대역폭을 통해 메모리 병목 현상도 극복할 수 있다.

도면의 간단한 설명

[0028] 도 1은 페이스북의 DLRM(Deep Learning Recommendation Models) 구조를 도시한 도면이다.

도 2는 Criteo Ad kaggle datasets을 이용하여 추론시 일부 임베딩 테이블(embedding table)에 대한 결과이다.

도 3은 본 발명에 따른 하이브리드 니어-메모리 프로세싱 구조를 도시한 도시한 도면이다.

도 4는 본 발명에 따른 하이브리드 니어-메모리 프로세싱 구조의 임베딩 과정시 데이터 이동시간을 비교한 표이다.

발명을 실시하기 위한 구체적인 내용

[0029] 이하, 본 발명에 따른 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 구조 및 방법을 첨부한 도면을 참고로 하여 상세히 기술되는 실시 예에 의하여 그 특징들을 이해할 수 있을 것이다.

[0030] 이에 앞서, 본 명세서 및 청구범위에 사용된 용어나 단어는 통상적이거나 사전적인 의미로 한정해서 해석되어서는 아니 되며, 발명자는 그 자신의 발명을 가장 최선의 방법으로 설명하기 위해 용어의 개념을 적절하게 정의할 수 있다는 원칙에 입각하여 본 발명의 기술적 사상에 부합하는 의미와 개념으로 해석되어야만 한다.

[0031] 따라서, 본 명세서에 기재된 실시 예와 도면에 도시된 구성은 본 발명의 가장 바람직한 일 실시 예에 불과할 뿐이고, 본 발명의 기술적 사상을 모두 대변하는 것은 아니므로, 본 출원시점에 있어서 이들은 대체할 수 있는 다양한 균등물과 변형 예들이 있을 수 있음을 이해하여야 한다.

[0033] 도 2는 참고문헌 2의 Criteo Ad kaggle datasets을 이용하여 추론시 일부 임베딩 테이블(embedding table)에 대한 결과를 도시한 그래프이다.

[0034] 이에 의하면 임베딩 룩업(embedding lookup) 횟수를 기준으로 임베딩 벡터들(embedding vectors)을 내림차순으로 정렬하였고, 전체 임베딩 룩업(embedding lookup) 횟수에 의해 정규화된 접근 횟수를 나타낸다. 이를 통해 임베딩 벡터들(embedding vectors)은 일부 상위 데이터들에 대해 접근이 많이 일어나는 것을 알 수 있다.

[0036] 이에 본 발명에서는 추천 시스템의 임베딩을 위해, 메모리 용량과 대역폭의 부족을 개선시키고 데이터 이동량을 줄여 처리 속도를 향상시키고자 하이브리드(Hybrid) 니어-메모리 프로세싱(Near-Memory Processing) 구조를 적용한다.

[0037] 도 3은 본 발명에 따른 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱(Hybrid Near-memory processing) 구조를 도시한 도시한 도면이다. 이를 참고하면 본 발명에 따른 하이브리드 니어-메모리 프로세싱 구조는 CPU(Central Processing Unit)(10)와, 메인메모리(Main Memory)(20), GPU(Graphics Processing Unit)(30), PIM(Processing-In-Memory)(40) 및 HBM(High Bandwidth Memory)(50)을 포함한다.

[0038] 상기 CPU(10)는 명령어의 해석과 자료의 연산, 비교 등의 처리를 제어하는 컴퓨터 시스템의 핵심적인 장치이고, 상기 GPU(30)는 그래픽 연산을 빠르게 처리하여 결과값을 모니터에 출력하는 연산 장치이다.

[0039] 이러한 CPU(10)와 GPU(30)는 모두 내부에 크게 연산을 담당하는 산출연산처리장치(ALU)와 명령어를 해석·실행하는 컨트롤유닛(CU), 각종 데이터를 담아두는 캐시를 포함한다.

[0040] 하지만, 상기 CPU(10)는 명령어가 입력된 순서대로 데이터를 처리하는 직렬(순차) 처리 방식에 특화된 구조를 가지고 있어 한 번에 한 가지의 명령어만 처리하며, 이에 연산을 담당하는 ALU(Arithmetic Logic Unit)의 개수가 많을 필요가 없으며, CPU(10) 내부의 절반 이상은 CPU(10)와 메인 메모리(20)인 램(RAM)과의 속도차이로 발행하는 병목현상을 막기 위한 캐시 메모리로 채워진다.

[0041] 이에 반해 GPU(30)는 여러 명령어를 동시에 처리하는 병렬 처리 방식을 가지고 있고, 캐시 메모리(Cache memory) 비중이 크지 않고 연산을 할 수 있는 ALU 개수가 많다.

[0042] 또한, 상기 PIM(40)은 AI(Artificial Intelligence)에 최적화된 반도체로서, 1개의 코어에는 수백, 수천개의 ALU가 장착된다. 이러한 PIM(40)은 메모리에서 CPU(10)까지 데이터를 가져오는 대신 메모리 안에서 데이터를 처리하기 때문에 CPU(10)와 메인 메모리(20) 사이의 대역폭 제한에 영향을 받지 않고 성능을 높일 수 있고, 메모리 안에서 계산을 수행할 경우 데이터를 CPU(10)까지 가져올 필요가 없어 데이터 이동에 소모되는 에너지 또한

크게 줄일 수 있다.

[0043] 그리고, 상기 HBM(50)은 고대역 메모리로서, 복수의 DRAM(Dynamic Random Access Memory)을 적층된 형태로 구현한 것으로, DRAM을 TVS(Through Slicon Via)를 통해 연결한 구조로 이루어진다.

[0045] 이러한 본 발명에 따른 Hybrid Near-memory 구조에서, 임베딩 룩업(embedding lookup) 횟수를 기준으로 접근이 많이 일어나는 임베딩 벡터들(embedding vectors)은 HBM(50)에 위치해 있고, 나머지 접근이 적게 일어나는 임베딩 벡터들(embedding vectors)은 CPU(10)의 메인메모리(20)에 있다.

[0046] 일단 임베딩(embedding)에서 메인 메모리(Main Memory)(20)와 HBM(50)에서 각각 임베딩 룩업(embedding lookup)이 일어난다.

[0047] 이와 같이 룩업(lookup)된 임베딩 벡터들(embedding vectors)은 CPU(10)와 PIM(40)에서 임베딩 축소(embedding reduction)가 이루어진다.

[0048] 그리고, 상기 CPU(10)와 PIM(40)에서 임베딩 축소(embedding reduction)를 통해 발생한 PSUM(Partial SUM)은 최종적으로 PIM(40)에서 임베딩 결과(embedding result)를 만들어 내고, 그 임베딩 결과(embedding result)는 GPU(3)로 전달되어 도 1의 Top MLP(Top Multi-Layer Perceptron)(1) 과정이 진행된다.

[0049] 이러한 본 발명의 Hybrid Near-memory 구조는 메인 메모리(Main Memory)(20)와 HBM(50)을 같이 사용함으로써 메모리의 용량 부족 문제를 해결할 수 있다.

[0050] 또한, CPU(10)와 PIM(40)에서 데이터를 임베딩 조정(embedding manipulate)을 함으로써 데이터 이동량을 줄일 수 있고, HBM(50)의 넓은 대역폭을 통해 메모리 병목 현상을 극복할 수 있다.

[0052] 이하, 도 4를 참고로 본 발명에 따른 Hybrid Near-memory 구조의 임베딩 실험 예를 설명한다.

[0053] 본 발명에서의 실험 환경은 4 channel DDR4-2666, HBM2E, PCIe 5.0을 사용하였다. 임베딩 룩업(embedding lookup) 시간과 PSUM 이동 시간은 "total size of embedding / Bandwidth"을 이용하여 구하였다.

[0054] 여기서, 도 4는 본 발명에 따른 Hybrid Near-memory 구조의 임베딩 과정시 데이터 이동시간을 비교한 표로서, 기존의 CPU 구조(CPU-only), GPU-HBM 구조(GPU-HBM), 그리고 본 발명의 Hybrid Near-memory 구조를 적용하여 한번 추론 시 임베딩(embedding) 과정에서 발생하는 데이터 이동 시간을 나타낸 것이다.

[0055] 이러한 도 4의 표는 본 발명의 Hybrid Near-memory 구조의 경우 메인 메모리(Main Memory)(20)와 HBM(50)에서 일어나는 임베딩 데이터 사이즈(embedding data size)의 크기가 같다고 가정하고, GPU-HBM 구조의 경우 HBM(50)의 용량이 무한이라고 가정하였을 때 결과이다.

[0056] 이러한 실험 결과에서와 같이 본 발명의 Hybrid Near-memory 구조는 CPU-only 구조 대비 임베딩 처리 시간(embedding processing time)이 약 46.5% 감소하였음을 확인하였다.

[0057] 또한, GPU-HBM 구조의 결과를 통해 본 발명의 Hybrid Near-memory 구조는 HBM()에 오프로딩(offloading)을 많이 할수록 약 0.2초(sec)에 가까워질 것을 예상할 수 있다.

[0059] 이러한 본 발명에서는 HBM(50)을 효과적으로 사용하기 위해 오프로딩(offloading) 규칙을 구체화함은 물론, 하이브리드(Hybrid) 구조에서 발생하는 오버헤드(overhead)를 분석하고 해결하여 추가적인 성능 개선 역시 필요하다.

[0061] 위에서 설명한 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 방법을 정리하면, 추천 시스템의 임베딩을 위한 하이브리드 니어-메모리 프로세싱 방법에 있어서, 메인 메모리(Main Memory)와 HBM에서 각각 임베딩 룩업(embedding lookup)이 일어나는 제1단계, 상기 룩업(lookup)된 임베딩 벡터들(embedding vectors)은 CPU와 PIM에서 임베딩 축소(embedding reduction)를 통해 PSUM(Partial SUM)이 발생하는 제2단계 및 상기 PIM에서 상기 PSUM(Partial SUM)에 따른 임베딩 결과(embedding result)를 만드는 제3단계를 포함하여 이루저 진다.

[0062] 이때, 상기 제1단계 이전에 임베딩 룩업(embedding lookup) 횟수를 기준으로 접근이 많이 일어나는 임베딩 벡터들(embedding vectors)은 HBM에 위치하고, 나머지 임베딩 벡터들(embedding vectors)은 CPU의 메인메모리에 위치시키는 제4단계를 더 포함한다.

[0063] 그리고, 상기 제1단계 이후에 상기 임베딩 결과(embedding result)는 Top MLP 과정이 진행되도록 GPU로 전달되는 제5단계를 더 포함한다.

[0065] 이상의 설명은 본 발명의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 다양한 수정 및 변형 가능한 것으로, 본 발명의 보호범위는 아래의 청구범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 발명의 권리범위에 포함되는 것으로 해석되어야 할 것이다.

부호의 설명

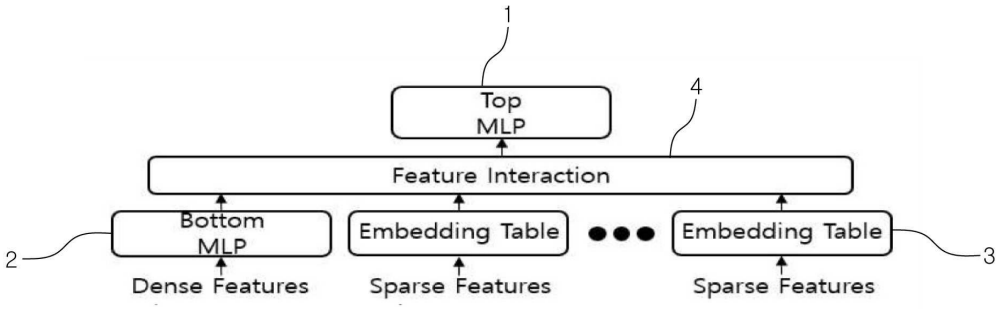
[0067] 10: CPU 20: 메인메모리

30: GPU 40: PIM

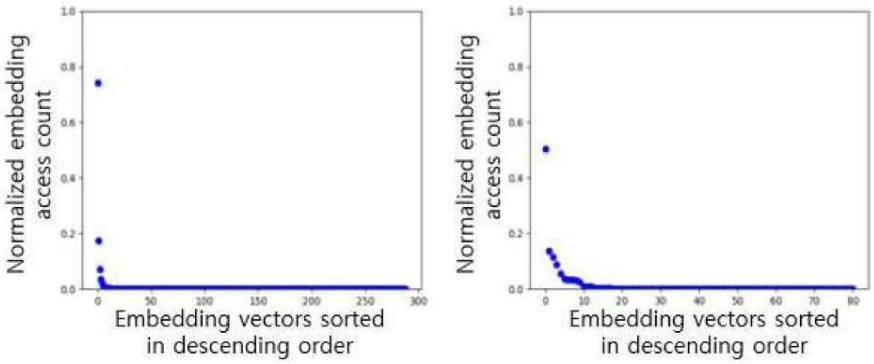
50: HBM

도면

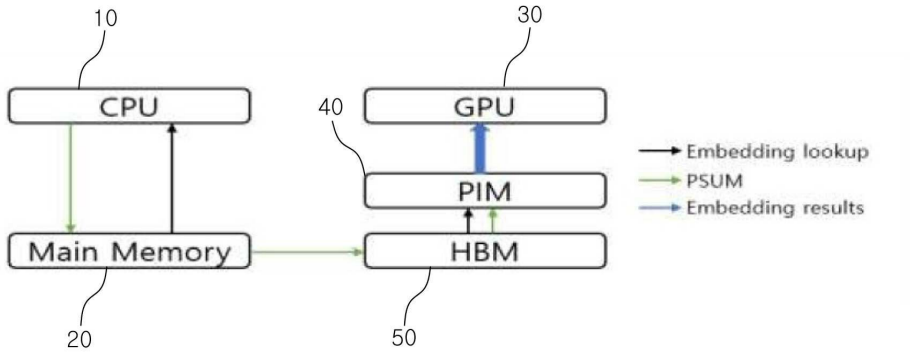
도면1



도면2



도면3



도면4

Architecture	Embedding processing time (ms)
CPU-only	4.3
GPU-HBM	0.2
Hybrid near memory	2.3