



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2021-0154080
(43) 공개일자 2021년12월20일

- | | |
|---|--|
| <p>(51) 국제특허분류(Int. Cl.)
G06N 3/08 (2006.01) G06N 3/04 (2006.01)
G06T 1/20 (2018.01)</p> <p>(52) CPC특허분류
G06N 3/08 (2013.01)
G06N 3/04 (2013.01)</p> <p>(21) 출원번호 10-2021-0060088</p> <p>(22) 출원일자 2021년05월10일
심사청구일자 없음</p> <p>(30) 우선권주장
1020200071021 2020년06월11일 대한민국(KR)</p> | <p>(71) 출원인
울산과학기술원
울산광역시 울주군 언양읍 유니스트길 50</p> <p>(72) 발명자
최영리
울산광역시 울주군 언양읍 유니스트길 50</p> <p>정연혁
울산광역시 울주군 언양읍 유니스트길 50
(뒷면에 계속)</p> <p>(74) 대리인
특허법인 고려</p> |
|---|--|

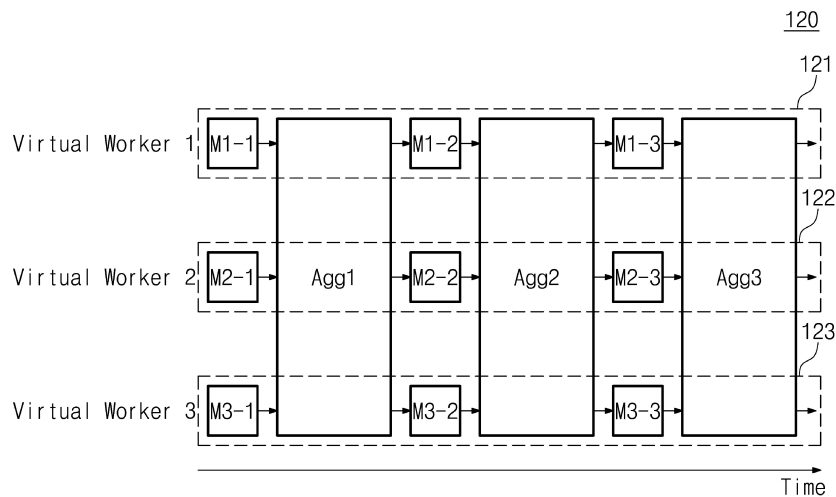
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 이기종 그래픽 프로세서 유닛 클러스터를 이용한 마이크로서비스 기반의 트레이닝 시스템 및 그 동작 방법

(57) 요약

본 개시의 일 실시예에 따른 복수의 신경망 모델들에 대한 트레이닝을 병렬적으로 수행하는 트레이닝 시스템은 제1 신경망 모델에 대한 트레이닝 요청을 수신하고, 제1 신경망 모델에 대응하는 제1 트레이닝 잡을 제1 마이크로서비스들로 분할하는 제1 잡 프록시, 제2 신경망 모델에 대한 트레이닝 요청을 수신하고, 제2 신경망 모델에 대응하는 제2 트레이닝 잡을 제2 마이크로서비스들로 분할하는 제2 잡 프록시, 제1 마이크로서비스들 및 제2 마이크로서비스들을 이종의 프로세싱 유닛들에게 동적으로 스케줄링하는 스케줄러, 이종의 프로세싱 유닛들을 포함하고 제1 마이크로서비스들 및 제2 마이크로서비스들 각각을 순차적으로 실행하는 클러스터, 및 제1 마이크로서비스들 및 제2 마이크로서비스들의 실행에 응답하여 발생된 매개 변수들을 저장하는 인-메모리 분산 데이터베이스를 포함한다.

대표도 - 도1b



(52) CPC특허분류

G06T 1/20 (2013.01)

(72) 발명자

이승민

울산광역시 울주군 언양읍 유니스트길 50

주성현

울산광역시 울주군 언양읍 유니스트길 50

명세서

청구범위

청구항 1

복수의 신경망 모델들에 대한 트레이닝을 병렬적으로 수행하는 트레이닝 시스템에 있어서,

제1 신경망 모델에 대한 트레이닝 요청을 수신하고, 상기 제1 신경망 모델에 대응하는 제1 트레이닝 잡을 제1 마이크로서비스들로 분할하는 제1 잡 프록시;

제2 신경망 모델에 대한 트레이닝 요청을 수신하고, 상기 제2 신경망 모델에 대응하는 제2 트레이닝 잡을 제2 마이크로서비스들로 분할하는 제2 잡 프록시;

상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들을 이종의 프로세싱 유닛들에게 동적으로 스케줄링하는 스케줄러;

상기 이종의 프로세싱 유닛들을 포함하고, 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들 각각을 순차적으로 실행하는 클러스터; 및

상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들의 실행에 응답하여 발생된 매개 변수들을 저장하는 인-메모리 분산 데이터베이스를 포함하는 트레이닝 시스템.

청구항 2

제1 항에 있어서,

상기 제1 잡 프록시는 상기 제1 트레이닝 잡에 대한 제1 트레이닝 데이터를 제1 미니 배치들로 분할하고, 상기 제1 미니 배치들 각각에 대한 처리들을 상기 제1 마이크로서비스들에 대응시키고,

상기 제2 잡 프록시는 상기 제2 트레이닝 잡에 대한 제2 트레이닝 데이터를 제2 미니 배치들로 분할하고, 상기 제2 미니 배치들 각각에 대한 처리들을 상기 제2 마이크로서비스들에 대응시키는 트레이닝 시스템.

청구항 3

제2 항에 있어서,

상기 제1 잡 프록시는 상기 제1 마이크로서비스들을 상기 제1 미니 배치들 각각에 대한 포워드 패스 또는 백워드 패스를 통해 상기 제1 미니 배치들에 대한 제1 매개 변수들을 계산하는 제1 유형의 마이크로서비스들, 및 상기 제1 매개 변수들을 총합하는 제2 유형의 마이크로서비스들로 분류하고,

상기 제2 잡 프록시는 상기 제2 마이크로서비스들을 상기 제2 미니 배치들 각각에 대한 포워드 패스 또는 백워드 패스를 통해 상기 제2 미니 배치들에 대한 제2 매개 변수들을 계산하는 제3 유형의 마이크로서비스들, 및 상기 제2 매개 변수들을 총합하는 제4 유형의 마이크로서비스들로 분류하는 트레이닝 시스템.

청구항 4

제3 항에 있어서,

상기 제1 잡 프록시 및 상기 제2 잡 프록시는 상기 제1 유형의 마이크로서비스들, 상기 제2 유형의 마이크로서비스들, 상기 제3 유형의 마이크로서비스들, 및 상기 제4 유형의 마이크로서비스들을 유형 정보와 함께 상기 스케줄러에 제출하는 트레이닝 시스템.

청구항 5

제3 항에 있어서,

상기 스케줄러는 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들 각각의 로컬리티 및 서비스 품질 중 적어도 하나를 기초로, 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들을 상기 이종의 프로세싱 유닛들에게 스케줄링하는 트레이닝 시스템.

청구항 6

제5 항에 있어서,

상기 스케줄러는 상기 로컬리티를 기초로 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들 중 병렬적으로 실행되는 제1 부분 마이크로서비스들을 상기 이종의 프로세싱 유닛들 중 동일한 노드에 배치되는 제1 프로세싱 유닛들에 스케줄링하는 트레이닝 시스템.

청구항 7

제6 항에 있어서,

상기 스케줄러는 상기 서비스 품질을 기초로 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들 중 병렬적으로 실행되는 제2 부분 마이크로서비스들을 상기 이종의 프로세싱 유닛들 중 상기 제1 프로세싱 유닛들의 처리 속도 보다 빠른 제2 프로세싱 유닛들에 스케줄링하고,

상기 제2 부분 마이크로서비스들은 상기 제1 부분 마이크로서비스들과 서로 다른 시기에 병렬적으로 실행되는 트레이닝 시스템.

청구항 8

제7 항에 있어서,

상기 제1 부분 마이크로서비스들 및 상기 제2 부분 마이크로서비스들은 동일한 트레이닝 잡을 실행하기 위한 마이크로서비스들인 트레이닝 시스템.

청구항 9

제1 항에 있어서,

상기 클러스터는 상기 제1 트레이닝 잡을 상기 제1 마이크로서비스들의 시퀀스로 실행하는 하나 이상의 제1 가상 작업자들 및 상기 제2 트레이닝 잡을 상기 제2 마이크로서비스들의 시퀀스로 실행하는 하나 이상의 제2 가상 작업자들을 포함하는 트레이닝 시스템.

청구항 10

제9 항에 있어서,

상기 제1 가상 작업자들 및 상기 제2 가상 작업자들은 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들을 기반으로 상기 제1 트레이닝 잡 및 상기 제2 트레이닝 잡을 병렬적으로 실행하는 트레이닝 시스템.

청구항 11

제10 항에 있어서,

상기 제1 가상 작업자들 및 상기 제2 가상 작업자들 각각은 파이프라인(pipeline) 방식으로 복수의 미니 배치들을 병렬적으로 처리하는 트레이닝 시스템.

청구항 12

제9 항에 있어서,

상기 트레이닝 시스템은 상기 클러스터들 각각의 자원 사용 상태를 기반으로, 상기 제1 트레이닝 잡에 대응하는 상기 제1 가상 작업자들의 수 및 상기 제2 트레이닝 잡에 대응하는 상기 제2 가상 작업자들의 수를 동적으로 조절하는 자원 관리자는 더 포함하는 트레이닝 시스템.

청구항 13

제1 항에 있어서,

상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들 중 오류(fault)가 발생한 마이크로서비스를 검출하고, 상기 검출된 마이크로서비스의 동작을 재수행하는 폴트 핸들러를 더 포함하는 트레이닝 시스템.

청구항 14

복수의 신경망 모델들에 대한 트레이닝을 병렬적으로 수행하는 트레이닝 시스템의 동작 방법에 있어서,
제1 신경망 모델 및 제2 신경망 모델을 포함하는 복수의 신경망 모델들에 대한 트레이닝 요청을 수신하는 단계;
상기 제1 신경망 모델에 대응하는 제1 트레이닝 잡을 제1 마이크로서비스들로 분할하는 단계;
상기 제2 신경망 모델에 대응하는 제2 트레이닝 잡을 제2 마이크로서비스들로 분할하는 단계;
상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들을 이종의 프로세싱 유닛들에게 동적으로 스케줄링하는 단계; 및
상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들 각각을 순차적으로 실행하는 단계를 포함하는 트레이닝 시스템의 동작 방법.

청구항 15

제14 항에 있어서,
상기 제1 마이크로서비스들로 분할하는 단계는:
상기 제1 트레이닝 잡에 대한 제1 트레이닝 데이터를 제1 미니 배치들로 분할하는 단계; 및
상기 제1 미니 배치들 각각에 대한 처리들을 상기 제1 마이크로서비스들에 대응시키는 단계를 포함하고,
상기 제2 마이크로서비스들로 분할하는 단계는:
상기 제2 트레이닝 잡에 대한 제2 트레이닝 데이터를 제2 미니 배치들로 분할하는 단계; 및
상기 제2 미니 배치들 각각에 대한 처리들을 상기 제2 마이크로서비스들에 대응시키는 단계를 포함하는 트레이닝 시스템의 동작 방법.

청구항 16

제15 항에 있어서,
상기 제1 마이크로서비스들로 분할하는 단계는 상기 제1 마이크로서비스들을 상기 제1 미니 배치들 각각에 대한 포워드 패스 또는 백워드 패스를 통해 상기 제1 미니 배치들에 대한 제1 매개 변수들을 계산하는 제1 유형의 마이크로서비스들, 및 상기 제1 매개 변수들을 총합하는 제2 유형의 마이크로서비스들로 분류하는 단계를 더 포함하고,
상기 제2 마이크로서비스들로 분할하는 단계는 상기 제2 마이크로서비스들을 상기 제2 미니 배치들 각각에 대한 포워드 패스 또는 백워드 패스를 통해 상기 제2 미니 배치들에 대한 제2 매개 변수들을 계산하는 제3 유형의 마이크로서비스들, 및 상기 제2 매개 변수들을 총합하는 제4 유형의 마이크로서비스들로 분류하는 단계를 더 포함하는 트레이닝 시스템의 동작 방법.

청구항 17

제16 항에 있어서,
상기 스케줄링하는 단계는 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들 각각의 로컬리티 및 서비스 품질 중 적어도 하나를 기초로, 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들을 상기 이종의 프로세싱 유닛들에게 스케줄링하는 단계를 포함하는 트레이닝 시스템의 동작 방법.

청구항 18

제17 항에 있어서,
상기 로컬리티를 기초로 스케줄링하는 단계는 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들 중 병렬적으로 실행되는 제1 부분 마이크로서비스들을 상기 이종의 프로세싱 유닛들 중 동일한 노드에 배치되는 제1 프로세싱 유닛들에 스케줄링하는 동작 방법.

청구항 19

제18 항에 있어서,

상기 서비스 품질을 기초로 스케줄링하는 단계는 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들 중 병렬적으로 실행되는 제2 부분 마이크로서비스들을 상기 이중의 프로세싱 유닛들 중 상기 제1 프로세싱 유닛들의 처리 속도 보다 빠른 제2 프로세싱 유닛들에 스케줄링하고,

상기 제2 부분 마이크로서비스들은 상기 제1 부분 마이크로서비스들과 서로 다른 시기에 병렬적으로 실행되는 트레이닝 시스템의 동작 방법.

청구항 20

제14 항에 있어서,

상기 실행하는 단계는:

상기 제1 트레이닝 잡을 상기 제1 마이크로서비스들의 시퀀스로 실행하는 단계; 및

상기 제2 트레이닝 잡을 상기 제2 마이크로서비스들의 시퀀스로 실행하는 단계를 더 포함하고,

상기 제1 트레이닝 잡 및 상기 제2 트레이닝 잡은 상기 제1 마이크로서비스들 및 상기 제2 마이크로서비스들을 기반으로 병렬적으로 실행되는 트레이닝 시스템.

발명의 설명

기술 분야

[0001] 실시예들은 이기종 그래픽 프로세서 유닛 클러스터를 이용한 마이크로서비스 기반의 트레이닝 시스템 및 그 동작 방법에 관한 것이다.

배경 기술

[0002] 하나의 심층 신경망(Deep Neural Network; DNN)을 트레이닝시키는 작업은 여러 그래픽 프로세싱 유닛들(Graphic Processing Units; 이하, 'GPU's')을 독점적으로 오랫동안 사용할 수 있다. 예를 들어, 멀티-테넌트(multi-tenant)와 같이 많은 사람들이 같은 기능을 사용하거나, 또는 하나의 소프트웨어 인스턴스가 한 대의 서버 위에서 동작하면서 접근 권한을 공유하는 사용자들을 서비스하는 경우, 해당 작업에 한번 할당된 GPU는 계속적으로 사용되므로 할당된 GPU들 간의 로컬리티(locality)가 중요하다. 하지만, GPU들 간의 로컬리티를 위해 스케줄링이 계속 지연될 수 있다. 여기서, 로컬리티는 예를 들어, 같은 트레이닝 잡(training job)을 실행하는 GPU들이 같은 노드 또는 랙 등에 가까이 배치되어 있는 것과 같은 공간적 근접도로 이해될 수 있다. GPU들이 가까이 배치되는 경우, 통신 오버헤드가 줄어들게 된다. 특히, 서로 다른 기종의 GPU들로 구성된 이기종 클러스터(cluster)에서는 여러 트레이닝 잡을 동시에 실행해야 하는데, 이 경우에 이기종 GPU를 어떤 잡(job)에 할당해야 시스템 전체의 성능 및 각 잡의 성능이 향상되는지를 결정하는 것이 더 복잡해질 수 있다. 또한, 이기종 클러스터에서 데이터 병렬화를 기반으로 트레이닝 하는 경우 스트레글러(straggler) 이슈에 의해 성능이 저하될 수도 있다.

[0003] 예를 들어, 최근 들어 많이 활용되고 있는 아마존 람다(Amazon Lambda), 및 구글 클라우드 함수(Google Cloud function) 등과 같은 마이크로서비스 시스템에서 계산이 함수 단위인 마이크로서비스(microservice)로 실행됨으로써 확장성, 자원을 사용하는데 있어서의 탄력성, 및 낮은 부하 오버헤드(low load overhead) 등과 같은 장점을 가질 수 있다. 다만, 현재의 마이크로서비스 시스템은 이와 같은 장점이 있으나, 독점적(monolithic)으로 실행되는 심층 신경망의 트레이닝에 활용되기는 어렵다.

[0004] 진술한 배경기술은 발명자가 본원의 개시 내용을 도출하는 과정에서 보유하거나 습득한 것으로서, 반드시 본 출원 전에 일반 공중에 공개된 공지기술이라고 할 수는 없다.

발명의 내용

해결하려는 과제

- [0005] 일 실시예에 따르면, 하나의 이기종 GPU 클러스터를 사용하여 다중 트레이닝 잡(training job)을 더 효율적으로 실행할 수 있다.
- [0006] 일 실시예에 따르면, 마이크로서비스(microservice) 기반의 기계 학습을 통해 이기종 그래픽 프로세싱 유닛들을 포함하는 클러스터로 구성된 심층 신경망을 트레이닝할 수 있다.
- [0007] 일 실시예에 따르면, 마이크로서비스 개념의 함수 단위로 딥 러닝 모델을 분할하고, 이기종 프로세서에서 딥 러닝 모델을 트레이닝 함으로써 복잡한 딥 러닝 모델을 보다 효율적으로 트레이닝할 수 있다.
- [0008] 일 실시예에 따르면, 이기종 프로세서들의 성능에 따라 마이크로서비스를 스케줄링 함으로써 보다 효율적으로 머신 러닝을 수행할 수 있다.
- [0009] 일 실시예에 따르면, 독점적인(monolithic) 심층 신경망 모델의 트레이닝을 마이크로서비스들에 기반하여 모듈화 함으로써 이기종 그래픽 프로세싱 유닛들을 포함하는 클러스터로 분산 처리가 가능하도록 실행할 수 있다.
- [0010] 일 실시예에 따르면, 복수의 트레이닝 잡을 동시에 지원하는 멀티-테넌트 그래픽 프로세싱 유닛 클러스터에서 이중 자원 관리의 복잡성을 줄여 공평성, 확장성, 효율성, 및 탄력성이 높은 트레이닝 시스템을 제공할 수 있다.

과제의 해결 수단

- [0011] 본 개시의 일 실시예에 따른 복수의 신경망 모델들에 대한 트레이닝을 병렬적으로 수행하는 트레이닝 시스템은, 제1 신경망 모델에 대한 트레이닝 요청을 수신하고, 제1 신경망 모델에 대응하는 제1 트레이닝 잡을 제1 마이크로서비스들로 분할하는 제1 잡 프록시, 제2 신경망 모델에 대한 트레이닝 요청을 수신하고, 제2 신경망 모델에 대응하는 제2 트레이닝 잡을 제2 마이크로서비스들로 분할하는 제2 잡 프록시, 제1 마이크로서비스들 및 제2 마이크로서비스들을 이중의 프로세싱 유닛들에게 동적으로 스케줄링하는 스케줄러, 이중의 프로세싱 유닛들을 포함하고 제1 마이크로서비스들 및 제2 마이크로서비스들 각각을 순차적으로 실행하는 클러스터, 및 제1 마이크로서비스들 및 제2 마이크로서비스들의 실행에 응답하여 발생된 매개 변수들을 저장하는 인-메모리 분산 데이터베이스를 포함한다.
- [0012] 본 개시의 일 실시예에 따른 복수의 신경망 모델들에 대한 트레이닝을 병렬적으로 수행하는 트레이닝 시스템의 동작 방법은, 제1 신경망 모델 및 제2 신경망 모델을 포함하는 복수의 신경망 모델들에 대한 트레이닝 요청을 수신하는 단계, 제1 신경망 모델에 대응하는 제1 트레이닝 잡을 제1 마이크로서비스들로 분할하는 단계, 제2 신경망 모델에 대응하는 제2 트레이닝 잡을 제2 마이크로서비스들로 분할하는 단계, 제1 마이크로서비스들 및 제2 마이크로서비스들을 이중의 프로세싱 유닛들에게 동적으로 스케줄링하는 단계, 및 제1 마이크로서비스들 및 제2 마이크로서비스들 각각을 순차적으로 실행하는 단계를 포함한다.

발명의 효과

- [0013] 일 측에 따르면, 하나의 이기종 GPU 클러스터를 사용하여 다중 트레이닝 잡(training job)을 더 효율적으로 실행할 수 있다.
- [0014] 일 측에 따르면, 마이크로서비스(microservice) 기반의 기계 학습을 통해 이기종 그래픽 프로세싱 유닛들을 포함하는 클러스터로 구성된 심층 신경망을 트레이닝할 수 있다.
- [0015] 일 측에 따르면, 마이크로서비스 개념의 함수 단위로 딥 러닝 모델을 분할하고, 이기종 프로세서에서 딥 러닝 모델을 트레이닝 함으로써 복잡한 딥 러닝 모델을 보다 효율적으로 트레이닝할 수 있다.
- [0016] 일 측에 따르면, 이기종 프로세서들의 성능에 따라 마이크로서비스를 스케줄링 함으로써 보다 효율적으로 머신 러닝을 수행할 수 있다.
- [0017] 일 측에 따르면, 독점적인(monolithic) 심층 신경망 모델의 트레이닝을 마이크로서비스들에 기반하여 모듈화 함으로써 이기종 그래픽 프로세싱 유닛들을 포함하는 클러스터로 분산 처리가 가능하도록 실행할 수 있다.
- [0018] 일 측에 따르면, 복수의 트레이닝 잡을 동시에 지원하는 멀티-테넌트 그래픽 프로세싱 유닛 클러스터에서 이중 자원 관리의 복잡성을 줄여 공평성, 확장성, 효율성, 및 탄력성이 높은 트레이닝 시스템을 제공할 수 있다.

도면의 간단한 설명

- [0019] 도 1a 내지 도 1d는 일 실시예에 따른 트레이닝 시스템의 동작 개념을 설명하기 위한 도면이다.
- 도 2는 일 실시예에 따른 트레이닝 시스템의 구성을 도시한 도면이다.
- 도 3은 일 실시예에 따른 트레이닝 시스템의 동작 방법을 나타낸 흐름도이다.
- 도 4a 내지 도 4c는 일 실시예에 따른 트레이닝 시스템이 복수의 트레이닝 잡을 수행하는 동작을 설명하기 위한 도면이다.
- 도 5는 일 실시예에 따른 트레이닝 시스템의 블록도이다.

발명을 실시하기 위한 구체적인 내용

- [0020] 이하에서, 첨부된 도면을 참조하여 실시예들을 상세하게 설명한다. 그러나, 실시예들에는 다양한 변경이 가해질 수 있어서 특허출원의 권리 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 실시예들에 대한 모든 변경, 균등물 내지 대체물이 권리 범위에 포함되는 것으로 이해되어야 한다.
- [0021] 도 1a 내지 도 1d는 일 실시예에 따른 트레이닝 시스템의 동작 개념을 설명하기 위한 도면이다. 도 1a는 트레이닝 시스템(100)의 트레이닝 과정을 개략적으로 나타낸 도면이고, 도 1b 내지 도 1c는 트레이닝 잡(120)에 대한 구체적인 예시이다.
- [0022] 도 1a를 참조하면, 트레이닝 시스템(100)은 신경망 모델에 대한 트레이닝을 수행할 수 있다. 실시 예에 따라, 신경망 모델은 거대 심층 신경망(Deep Neural Network, DNN) 모델(110)일 수 있다. 거대 DNN 모델(110)은 트레이닝 시스템(100)에 의해 트레이닝될 수 있고, 이러한 일련의 과정은 트레이닝 잡(120)으로 명명될 수 있다.
- [0023] 머신 러닝의 트레이닝 잡은 하나의 작업자(worker) 또는 성능 향상을 위해 여러 작업자들에 의해 실행될 수 있다. 여기서, 작업자는 예를 들어, GPU(Graphic Processing Unit)일 수 있다. 그러나 본 개시에서는 트레이닝 잡을 여러 마이크로서비스들의 형태로 수행하는데 있어 가상 작업자(virtual worker)의 개념을 사용할 수 있다. 가상 작업자에 관한 상세한 설명은 도 1b 내지 도 1d에서 후술된다.
- [0024] 트레이닝 잡(120)은 복수의 마이크로서비스들(M1, M2, M3, M4)로 구성될 수 있다. 마이크로서비스는 마이크로서비스 시스템에서의 계산 단위, 즉 하나의 함수(function)로서, 하나의 응용(application)은 마이크로서비스들의 시퀀스(sequence of microservices)로 실행될 수 있다. 실시 예에 따라, 트레이닝 데이터가 복수의 미니 배치(minibatch)들을 포함하는 트레이닝 시스템(100)에서, 마이크로서비스는 하나의 미니 배치의 처리, 예를 들어, 하나의 미니 배치에 대한 포워드 패스(forward pass) 또는 백워드 패스(backward pass)일 수 있다. 또는, 마이크로서비스는 복수의 미니 배치들의 처리 시퀀스일 수 있다.
- [0025] 즉, 마이크로서비스는 트레이닝 시스템(100)에서 하나의 처리 단위를 의미할 뿐, 그 처리의 대상은 실시 예에 따라 달라질 수 있다. 트레이닝 시스템(100)에서, 거대 DNN 모델(110)의 트레이닝을 위한 트레이닝 잡(120)은 함수 단위의 작업체인 마이크로서비스들(M1, M2, M3, M4)의 시퀀스로 실행될 수 있다. 트레이닝 시스템(100)은 마이크로서비스들(M1, M2, M3, M4)을 스케줄러(130)로 제출(submit)할 수 있다.
- [0026] 스케줄러(130)는 마이크로서비스들(M1, M2, M3, M4)을 스케줄러(130)의 큐(미도시)에 추가할 수 있다. 스케줄러(130)는 마이크로서비스들(M1, M2, M3, M4)을 클러스터(140)에 동적으로 스케줄링할 수 있다. 스케줄러(130)는 마이크로서비스들(M1, M2, M3, M4) 각각의 로컬리티(locality), 신경망 모델의 쓰루풋(throughput), 서비스 품질(QoS) 등을 고려하여 스케줄링할 수 있다.
- [0027] 클러스터(140)는 트레이닝 잡(120)을 실행하여 거대 신경망 모델(110)을 트레이닝할 수 있다. 클러스터(140)는 복수의 프로세싱 유닛들(PU1, PU2)을 포함할 수 있다. 프로세싱 유닛들(PU1, PU2)은 예를 들어, 그래픽 프로세싱 유닛들(GPUs) 또는 센트럴 프로세싱 유닛들(CPUs)을 포함할 수 있다. 또한, 프로세싱 유닛들(PU1, PU2)은 예를 들어, 동종의 프로세싱 유닛들일 수도 있고, 또는 이종의 프로세싱 유닛들일 수도 있다.
- [0028] 실시 예에 따라, 트레이닝 시스템(100)은 클러스터(140)에서 복수의 트레이닝 잡들을 실행할 수 있다. 복수의 트레이닝 잡들 각각은 복수의 마이크로서비스들로 모듈화될 수 있다. 이를 위해 각각의 마이크로서비스는 클러스터(140)의 프로세싱 유닛들(PU1, PU2) 중 이용가능한 프로세싱 유닛에 동적으로 스케줄링될 수 있다.
- [0029] 도 1a 및 도 1b를 참조하면, 트레이닝 잡(120)은 제1 태스크(121), 제2 태스크(122), 및 제3 태스크(123)을 포

함할 수 있다. 실시 예에 따라, 제1 태스크(121), 제2 태스크(122), 및 제3 태스크(123)는 미니 배치별로 분할될 수 있다. 제1 태스크(121), 제2 태스크(122), 및 제3 태스크(123)는 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)에 의해 병렬적으로 실행될 수 있다.

[0030] 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)이 제1 태스크(121), 제2 태스크(122), 및 제3 태스크(123)를 실행하는 것은 트레이닝 잡(120)을 실행시켜 거대 DNN 모델(110)에 대한 트레이닝을 수행하는 것을 의미할 수 있다. 설명의 편의를 위해 가상 작업자들의 수가 3인 것으로 예시하였지만, 본 개시는 이에 한정되지 않는다. 예를 들어, 트레이닝 시스템(100)은 하나의 가상 작업자를 이용하여, 트레이닝 잡(120)을 실행할 수 있다.

[0031] 가상 작업자는 실제로 존재하는 것이 아니라, 마이크로서비스들의 시퀀스(sequence of microservices)로 실행되는 개념일 수 있다. 가상 작업자는 마이크로서비스들을 순차적으로 실행시켜 GPU에 의한 통상의 작업자를 대체할 수 있다. 그러나, 가상 작업자는 통상의 작업자와는 다르게, 물리적 GPU와 디커플링(decoupling)되어 실행될 수 있다.

[0032] 실시 예에 따라, 트레이닝 시스템(100)은 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)을 통해 전용된(dedicated) GPU의 할당 없이, 모듈화된 마이크로서비스들의 형태로 트레이닝 잡을 실행할 수 있다. 예를 들어, 제1 내지 제3 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3) 각각은 제1 내지 제3 태스크들(121, 122, 123) 각각을 실행할 수 있다. 제1 내지 제3 태스크들(121, 122, 123) 각각은 복수의 마이크로서비스들을 포함할 수 있다.

[0033] 제1 태스크(121)는 제1 가상 작업자(Virtual Worker 1)에 의해 실행될 수 있다. 제1 태스크(121)는 마이크로서비스들(M1-1, Agg1, M1-2, Agg2, M1-2, Agg3)을 포함할 수 있다. 제1 가상 작업자(Virtual Worker 1)는 마이크로서비스들(M1-1, Agg1, M1-2, Agg2, M1-2, Agg3)을 순차적으로 실행할 수 있다.

[0034] 제2 태스크(122)는 제2 가상 작업자(Virtual Worker 2)에 의해 실행될 수 있다. 제2 태스크(122)는 마이크로서비스들(M2-1, Agg1, M2-2, Agg2, M2-2, Agg3)을 포함할 수 있다. 제2 가상 작업자(Virtual Worker 2)는 마이크로서비스들(M2-1, Agg1, M2-2, Agg2, M2-2, Agg3)을 순차적으로 실행할 수 있다.

[0035] 제3 태스크(123)는 제3 가상 작업자(Virtual Worker 3)에 의해 실행될 수 있다. 제3 태스크(123)는 마이크로서비스들(M3-1, Agg1, M3-2, Agg2, M3-2, Agg3)을 포함할 수 있다. 제3 가상 작업자(Virtual Worker 3)는 마이크로서비스들(M3-1, Agg1, M3-2, Agg2, M3-2, Agg3)을 순차적으로 실행할 수 있다.

[0036] 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)은 마이크로서비스들을 기반으로 트레이닝 잡(120)을 제1 태스크(121), 제2 태스크(122), 및 제3 태스크(123)로 분할하여 병렬적으로 실행할 수 있다. 이로써, 3개의 GPU를 통해 트레이닝 잡(120)을 실행하는 것에 상응하는 효율성을 나타낼 수 있다. 그러나, 트레이닝 모델의 크기가 일정 크기 이상인 경우, 트레이닝 모델 자체를 서브 모델들로 분할하여 가상 작업자의 작업 효율성을 높일 필요가 있다. 이에 관한 상세한 설명은 도 1c에서 후술된다.

[0037] 실시 예에 따라, 마이크로서비스들은 제1 유형 마이크로서비스들(M1-1, M1-2, M1-3, M2-1, M2-2, M2-3, M3-1, M3-2, M3-3) 및 제2 유형 마이크로서비스들(Agg1, Agg2, Agg3)로 분할될 수 있다. 제1 유형 마이크로서비스들(M1-1, M1-2, M1-3, M2-1, M2-2, M2-3, M3-1, M3-2, M3-3)은 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)이 가중치들(weights)을 산출하는 작업일 수 있다.

[0038] 제1 유형 마이크로서비스들은 제1 부분 마이크로서비스들(M1-1, M2-1, M3-1), 제2 부분 마이크로서비스들(M1-2, M2-2, M3-2), 및 제3 부분 마이크로서비스들(M1-3, M2-3, M3-3)을 포함할 수 있다. 제1 부분 마이크로서비스들(M1-1, M2-1, M3-1), 제2 부분 마이크로서비스들(M1-2, M2-2, M3-2), 및 제3 부분 마이크로서비스들(M1-3, M2-3, M3-3)은 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)에 의해 병렬적으로 실행될 수 있다.

[0039] 제2 유형 마이크로서비스들(Agg1, Agg2, Agg3)은 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3) 각각이 산출한 가중치들을 총합(aggregate)하는 작업일 수 있다. 제2 유형 마이크로서비스들(Agg1, Agg2, Agg3) 각각은 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)에 의해 함께 수행될 수 있다.

[0040] 도 1b를 참조하면, 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)은 제1 부분 마이크로서비스들(M1-1, M2-1, M3-1)을 병렬적으로 실행하여 제1 가중치들을 산출하고, 제2 유형 마이크로서비

스(Agg1)를 실행하여 제1 가중치들을 총합할 수 있다. 그 후, 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)은 제2 부분 마이크로서비스들(M1-2, M2-2, M3-2)을 병렬적으로 실행하여 제2 가중치들을 산출하고, 제2 유형 마이크로서비스(Agg2)를 실행하여 제2 가중치들을 총합할 수 있다. 마지막으로, 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)은 제3 부분 마이크로서비스들(M1-3, M2-3, M3-3)을 병렬적으로 실행하여 제3 가중치들을 산출하고, 제2 유형 마이크로서비스(Agg3)를 실행하여 제3 가중치들을 총합할 수 있다.

[0041] 다시 말해, 제1 부분 마이크로서비스들(M1-1, M2-1, M3-1), 제2 부분 마이크로서비스들(M1-2, M2-2, M3-2), 및 제3 부분 마이크로서비스들(M1-3, M2-3, M3-3) 각각은 서로 다른 시기에 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3)에 의해 병렬적으로 실행될 수 있다.

[0042] 도 1a 내지 도 1c를 참조하면, 거대 DNN 모델(110)은 제1 내지 제3 서브 모델들(111, 113, 115)로 분할될 수 있다. 설명의 편의를 위해, 서브 모델들의 수를 3개로 예시하였지만, 서브 모델들의 수는 이에 한정되지 않는다. 제1 내지 제3 서브 모델들(111, 113, 115)은 클러스터(140)에 의해 트레이닝될 수 있다.

[0043] 트레이닝 잡(120)은 제1 서브 모델(111), 제2 서브 모델(113), 및 제3 서브 모델(115) 각각에서의 포워드 패스(forward pass) 및 백워드 패스(backward pass) 간을 오가는 반복 동작들에 의해 매개 변수들을 산출하는 과정일 수 있다. 매개 변수는 트레이닝 시에 신경망의 각 레이어들에 적용되는 가중치(weight)에 해당할 수 있다. 이하, '매개 변수'는 '가중치'와 동일한 의미로 이해될 수 있다.

[0044] 실시 예에 따라, 트레이닝 잡(120)은 하나 또는 다수 개의 미니 배치들(mini-batches)을 반복적으로 처리하여 매개 변수들을 생성하는 과정일 수 있다. 이러한 매개 변수들의 크기는 거대 DNN 모델(110)에서 일정 크기 이상이기 때문에 하나의 프로세싱 유닛에 의해 처리되기 어려울 수 있고, 다수 개의 프로세싱 유닛들로 분산되어 처리될 수 있다. 이때, 미니 배치의 처리는 포워드 패스 및 백워드 패스로 구성될 수 있다.

[0045] 미니 배치를 이용하는 이유는 다음과 같다. 예를 들어, 트레이닝 데이터가 200 개 있다고 하자. 트레이닝 시스템은 200 개의 트레이닝 데이터와 신경망 모델이 예측한 값 간의 차이로 인한 손실(loss)을 계산한 다음 손실이 최소화되도록 신경망 모델의 가중치를 업데이트 함으로써 신경망 모델을 트레이닝할 수 있다. 이때, 200개의 트레이닝 데이터는 예를 들어, 32개의 단위로 분할될 수 있으며, 분할된 각 단위는 미니 배치에 해당할 수 있다. 즉, 트레이닝 시스템은 미니 배치를 이용함으로써 계산량을 줄여 가중치를 빠른 시간 내에 업데이트할 수 있다. 미니 배치를 이용하는 경우, 각 클러스터 또는 프로세싱 유닛은 200개가 아닌 32개의 트레이닝 데이터에 대한 신경망의 손실을 구한 뒤 가중치를 업데이트하므로 빠른 시간에 트레이닝을 완료할 수 있다.

[0046] 실시 예에 따라, 마이크로서비스(M1-1)는 하나의 미니 배치에 대해 제1 내지 제3 서브 모델들(111, 113, 115)에 대응하는 마이크로서비스들(1F0, 1F1, 1F2, 1B2, 1B1, 1B0)로 세분화될 수 있다. 각 마이크로서비스에 기재된 'F'는 해당 마이크로서비스가 포워드 패스(forward pass)에 대응한 작업임을 나타내고, 'B'는 해당 마이크로서비스가 백워드 패스(backward pass)에 대응한 작업임을 나타낼 수 있다. 예를 들어, 마이크로서비스(1F0)는 제1 미니 배치에 대해 제1 서브 모델(111)에서의 포워드 패스에 대응하는 작업을 의미할 수 있다. 예를 들어, 마이크로서비스(1B2)는 제1 미니 배치에 대해 제3 서브 모델(115)에서의 백워드 패스에 대응하는 작업을 의미할 수 있다.

[0047] 실시 예에 따라, 복수의 가상 작업자들(Virtual Worker 1, Virtual Worker 2, Virtual Worker 3) 각각은 하나의 미니 배치에 대한 처리뿐만 아니라, 동시에 여러 미니 배치들을 처리하여 작업 효율성을 향상시킬 수 있다.

[0048] 도 1a 내지 도 1d를 참조하면, 제1 가상 작업자(Virtual Worker 1)는 여러 마이크로서비스들을 파이프라인(pipeline)으로 실행하여 여러 미니 배치들을 동시에 처리할 수 있다.

[0049] 제1 태스크(121)는 제1 서브 태스크(121-1), 제2 서브 태스크(121-2), 제3 서브 태스크(121-3)를 포함할 수 있다. 제1 가상 작업자(Virtual Worker 1)는 제1 서브 태스크(121-1), 제2 서브 태스크(121-2), 제3 서브 태스크(121-3)를 동시에 및/또는 병렬적으로 처리할 수 있다. 제1 가상 작업자(Virtual Worker 1)에 의해 처리된 제1 내지 제3 서브 태스크들(121-1, 121-2, 121-3)은 총합 되어 트레이닝 결과로서 출력될 수 있다.

[0050] 제1 내지 제3 서브 태스크들(121-1, 121-2, 121-3) 각각은 트레이닝을 위한 작업들을 수행하는 마이크로서비스들을 포함할 수 있다. 예를 들어, 제1 서브 태스크(121-1)는 제1 미니 배치를 처리하는 마이크로서비스들(1F0, 1F1, 1F2, 1B2, 1B1, 1B0) 및 갱신을 수행하는 마이크로서비스(Update)를 포함할 수 있다. 제1 미니 배치를 처리하는 마이크로서비스들(1F0, 1F1, 1F2, 1B2, 1B1, 1B0)에 대한 상세한 설명은 도 1c의 마이크로서비스들(1F0, 1F1, 1F2, 1B2, 1B1, 1B0)과 유사하므로 생략된다. 갱신을 수행하는 마이크로서비스(Update)는 제1 가상 작업자

(Virtual Worker 1) 내에서 산출된 매개 변수들을 다른 가상 작업자에 의해 사용될 수 있도록 업데이트하는 과정에 해당할 수 있다. 실시 예에 따라, 갱신을 수행하는 마이크로서비스(Update)는 도 1b의 제2 유형 마이크로 서비스들(Agg1, Agg2, Agg3) 중 하나에 대응될 수 있다.

[0051] 제1 서버 태스크(121-1)에 대한 설명은 제2 서버 태스크(121-2) 및 제3 서버 태스크(121-3)에 대한 설명에 적용될 수 있다. 예를 들어, 제2 서버 태스크(121-2)는 제2 미니 배치를 처리하는 마이크로서비스들(2F0, 2F1, 2F2, 2B2, 2B1, 2B0) 및 갱신을 수행하는 마이크로서비스(Update)를 포함할 수 있다. 예를 들어, 제3 서버 태스크(121-3)는 제3 미니 배치를 처리하는 마이크로서비스들(3F0, 3F1, 3F2, 3B2, 3B1, 3B0) 및 갱신을 수행하는 마이크로서비스(Update)를 포함할 수 있다. 제1 가상 작업자(Virtual Worker 1)는 제1 내지 제3 서버 태스크들(121-1, 121-2, 121-3)을 파이프라인으로 실행하여 제1 내지 제3 미니 배치들에 대한 처리를 동시에 및/또는 병렬적으로 수행할 수 있다. 즉, 제1 가상 작업자(Virtual Worker 1)는 마이크로서비스 기반의 태스크 병렬화를 통해 작업 효율성을 높일 수 있다.

[0053] 도 2는 일 실시예에 따른 트레이닝 시스템의 구성을 도시한 도면이다. 도 2를 참조하면, 일 실시예에 따른 트레이닝 시스템(200)은 예를 들어, 잡 프록시(Job proxy)(210), 컨트롤러(Controller)(220), 폴트 핸들러(Fault handler)(230), 및 액션 데이터베이스(Action Database)(240)를 포함할 수 있다. 또한, 트레이닝 시스템(200)은 분산 파일 시스템(Distributed File System)(250) 및 분산된 인-메모리 데이터베이스(Distributed In-memory Database)(260)를 더 포함할 수 있다. 도 2에 도시된 '고래 아이콘은 예를 들어, docker와 같은 컨테이너(container)를 의미할 수 있다.

[0054] 트레이닝 시스템(200)은 트레이닝 잡을, 신경망 모델의 가상 작업자들에 대응하는 태스크들로 분할한다. 이때, 태스크들은 마이크로서비스들에 의해 수행된다. 트레이닝 시스템(200)은 마이크로서비스들을 신경망 모델을 위한 클러스터들 각각에 포함된 이종의 그래픽 프로세싱 유닛들에게 스케줄링 함으로써 신경망 모델을 트레이닝한다. 이를 위한 트레이닝 시스템(200)의 각 구성 요소들의 동작은 다음과 같다.

[0055] 잡 프록시(210)는 신경망의 트레이닝을 위한 트레이닝 데이터에 대해 전처리(pre-processing)를 수행할 수 있다. 여기서, 전처리는 예를 들어, 트레이닝 데이터를 마이크로서비스 단위로 나누는 것일 수 있다. 또한, 하나의 입력 데이터는 하나의 신경망 모델 또는 여러 신경망 모델을 트레이닝하는 데에 사용될 수 있다. 잡이 복수 개 있는 경우, 트레이닝 시스템(200)은 잡 프록시(210)를 잡 별로 생성할 수 있다. 잡 프록시(210)는 전처리된 트레이닝 데이터에 대한 계산 유형에 따라 태스크들을 마이크로서비스들로 분할할 수 있다.

[0056] 잡 프록시(210)는 입력 관리자(input manager)(213) 및 잡 관리자(job manager)(216)를 포함할 수 있다.

[0057] 입력 관리자(213)는 사용자(205)로부터 트레이닝 요청과 함께 트레이닝 데이터가 입력되면, 입력된 트레이닝 데이터를 분산 파일 시스템(250)에 저장하고, 전처리를 통해 트레이닝 데이터를 각 마이크로서비스들이 처리할 수 있도록 만들어 줄 수 있다.

[0058] 입력 관리자(213)는 트레이닝 데이터를 마이크로서비스들에 대응하는 형태로 가공할 수 있다. 입력 관리자(213)는 예를 들어, 트레이닝 데이터를 태스크들 별로 분할하거나, 또는 트레이닝 데이터를 마이크로서비스 단위로 분할할 수 있다.

[0059] 잡 관리자(216)는 분할된 마이크로서비스들을 예를 들어, 매개 변수를 산출하는 마이크로서비스들과 매개 변수를 총합하는 마이크로서비스들로 분류할 수 있다. 또한, 잡 관리자(216)는 분류된 마이크로서비스들에 대한 정보를 스케줄러(226)에게 전달할 수 있다. 예를 들어, 잡 관리자(216)는 분류된 마이크로서비스들의 계산 유형(예를 들어, 매개 변수 산출 또는 매개 변수 총합 등), 수행해야 할 미니 배치의 정보, 및 이터레이션 수 등을 포함한 스케줄링 시 필요한 정보를 스케줄러(226)에게 전달할 수 있다.

[0060] 전술한 바와 같이, 가상 작업자들은 자신에게 할당된 태스크들에 의해 복제된 서버 모델들을 트레이닝할 수 있다. 이때, 트레이닝 과정이 완료된 최종 매개 변수들은 예를 들어, 분산 파일 시스템(250)에 저장될 수 있다. 또한, 트레이닝 과정에서 지역적으로 계산된 상기 신경망 모델의 매개 변수, 그래디언트(gradient), 및 활성 출력 등과 같은 중간 데이터(intermediate data), 또는 트레이닝 과정에서 전역적으로 공유되는 매개 변수는 예를 들어, 분산된 인-메모리 데이터베이스(260)에 저장될 수 있다. 매개 변수들은 트레이닝 과정의 반복 처리 시에 가상 작업자들 간에 공유될 수 있으며, 매개 변수(가중치)를 동기화하는 다양한 정책들이 존재할 수 있다.

[0061] 잡 관리자(216)는 매개 변수에 대한 동기화 정책을 기초로, 마이크로서비스들을 계산 유형에 따라 분류할 수 있

다. 이때, 매개 변수를 동기화하는 정책들로는 BSP(Bulk Synchronous Parallel) 모델, ASP(Asynchronous Parallel) 모델, 및 SSP(Stale Synchronous Parallel) 모델 등을 이용하는 동기화 정책들을 예로 들 수 있으며, 반드시 이에 한정되지는 않는다. BSP 모델은 장벽 동기화를 이용하며, 장벽에 도달하는 가상 작업자는 다른 모든 가상 작업자가 도달할 때까지 기다림으로써 모든 가상 작업자들에서 동기화가 함께 진행되도록 한다. ASP 모델은 다른 가상 작업자의 작업 속도와는 무관하게 가상 작업자 개별로 가중치를 동기화할 수 있다. SSP 모델은 동기화를 수행하지만 BSP 모델의 엄격한 동기화 요구 사항을 완화한 것으로서, ASP 모델과 BSP 모델의 중간 접근 방식을 제공한다. SSP 모델은 원칙적으로, 가장 빠른 가상 작업자와 가장 느린 가상 작업자 간의 반복 차이를 모니터링하고, 가장 빠른 가상 작업자가 동기화가 가능해질 때까지 기다림으로써 임계 값 내에 있도록 제한할 수 있다. 일 실시예에서는 자원의 상황이나 신경망 모델의 특성에 따라 전술한 BSP, ASP, 또는 SSP 모델 등의 동기화 정책을 동적으로 변경하여 신경망의 트레이닝을 수행할 수 있다.

[0062] 잡 관리자(216)는 트레이닝에 의해 발생하는 매개 변수에 대한 동기화 정책을 기초로, 분류된 마이크로서비스들에 대한 정보를 스케줄러(223)에게 전달할 수 있다. 예를 들어, 각 미니 배치를 처리할 때마다 신경망 모델의 매개 변수에 대한 값이 산출될 수 있다. 이러한 미니 배치들이 여러 개의 가상 작업자들에 의해 실행되는 경우, 매개 변수의 값들은 동기화 정책에 따라 총합되어야 한다. 잡 관리자(216)는 이러한 총합 작업을 수행하는 마이크로서비스를 전술한 동기화 정책에 따라 생성하여 스케줄러(223)에게 전달할 수 있다.

[0063] 컨트롤러(220)는 잡 프록시(210)에 의해 분할된 마이크로서비스들을 이종의 그래픽 프로세싱 유닛들에게 스케줄링하고, 스케줄링에 따라 자원(resource)을 할당하는 등의 자원 관리를 수행할 수 있다.

[0064] 컨트롤러(220)는 예를 들어, 스케줄러(scheduler)(223) 및 자원 관리자(resource manager)(226)를 포함할 수 있다.

[0065] 스케줄러(223)는 복수의 마이크로서비스들을 클러스터에 포함된 이종 그래픽 프로세싱 유닛(GPU)들에게 스케줄링할 수 있다. 스케줄러(223)는 예를 들어, 마이크로서비스들 각각의 로컬리티(locality), 동일한 클러스터 내에서 실행되는 다른 종류의 태스크의 마이크로서비스들 간의 간섭, 및 신경망 모델의 정확도, 신경망 모델의 쓰루풋(throughput), 마이크로서비스들의 소요 시간을 포함하는 서비스 품질(QoS), 중 적어도 하나를 기초로, 마이크로서비스들을 이종의 그래픽 프로세싱 유닛들에게 스케줄링할 수 있다. 여기서, 마이크로서비스들의 스케줄링 시에 '마이크로서비스들 각각의 로컬리티'에 기초한다는 것은 하나의 잡(job)으로 실행되는 여러 개의 마이크로서비스들이 서로 가까운 위치 또는 거리에 스케줄링(배치)되게 한다는 의미로 이해될 수 있다.

[0066] 마이크로서비스 각각의 로컬리티는 예를 들어, 마이크로서비스들 각각이 사용해야하는 중간 데이터와의 거리에 해당할 수 있다. 분산 인-메모리 데이터베이스에 저장된 마이크로서비스가 사용해야할 중간 데이터의 위치와 마이크로서비스가 실행될 GPU가 있는 머신의 클라우드 상에서의 위치는 같은 머신일 수도, 같은 랙 내 다른 머신일 수도, 또는 다른 랙 내의 머신일 수도 있다. 이를 고려해 스케줄링 함으로써 데이터의 이동에 소요되는 시간을 줄이고 트레이닝 효율을 극대화할 수 있다.

[0067] 또한, 마이크로서비스들의 스케줄링 시에 '마이크로서비스들 간의 간섭'에 기초한다는 것의 의미는 다음과 같다. 예를 들어, 한 노드가 멀티 코어 및 멀티 GPU로 설정되어 있는 경우, 한 노드에서 여러 마이크로서비스들이 동시에 실행될 수 있다. 이와 같이 한 노드에서 동시에 여러 마이크로서비스들이 실행되면, 같은 노드의 물리적 자원(예: host memory, network, PCI-e 등)에 대한 여러 마이크로서비스들의 경쟁적 사용으로 인한 간섭으로 성능이 느려질 수 있다. 이러한 경우, 다른 종류의 태스크의 마이크로서비스들 간의 간섭이 최소화되도록 여러 개의 마이크로서비스들이 스케줄링될 수 있다.

[0068] 이 밖에도, 스케줄러(223)는 사용자가 명시한 비용, 성능 등에 대한 서비스 품질에 따라 이기종 GPU 자원을 활용할 수 있다. 예를 들어, 해당 잡 또는 마이크로서비스들이 고성능을 요구하는 경우, 스케줄러(223)는 GPU 클러스터에서 성능이 좋은 GPU를 해당 잡 또는 마이크로서비스들에 배치할 수 있다.

[0069] 또한, 스케줄러(223)는 자원 상황을 고려하여 동일한 클러스터에 여러 잡을 스케줄링할 수 있다. 이 경우 클러스터의 개별 잡 수행 능력은 감소할 수 있지만 클러스터의 활용도를 극대화시킬 수 있다.

[0070] 예를 들어, 신경망 모델의 쓰루풋을 일정 수준 이상 확보해야 하는 경우 등과 같이 특정 QoS를 만족해야 하는 경우, 스케줄러(223)는 잡을 수행 중인 클러스터의 상태를 모니터링하여 기준에 미치지 못하는 잡을 다른 자원에 스케줄링하는 등의 방식으로 작동할 수 있다.

[0071] 실시 예에 따라, 스케줄러(223)는 가상 작업자들 각각이 마이크로서비스들에 의해 처리하는 미니-배치(mini-batch)의 포워드 패스(forward pass) 및 백워드 패스(backward pass) 중 어느 하나를 이종의 프로세싱 유닛들

에게 동적으로 스케줄링할 수 있다. 또는, 스케줄러(223)는 가상 작업자들 각각이 마이크로서비스들에 의해 처리하는 미니-배치(mini-batch)에 대한 트레이닝에 의해 발생하는 매개 변수에 대한 동기화 정책을 기초로, 가상 작업자들 각각에서 처리되는 미니 배치에 대응하는 매개 변수들을 총합 하는 마이크로서비스를 이종의 프로세싱 유닛들에게 동적으로 스케줄링할 수 있다.

[0072] 일반적으로 다중 트레이닝 잡을 처리하는 경우, 실행 중인 잡이 오랜 시간 실행되면, 새롭게 제공된 잡은 이전 잡이 끝날 때까지 기다리거나 선취 방식(preemption) 등의 복잡한 처리를 통해 수행될 수 있다. 이와 달리, 일 실시예에 따른 마이크로서비스를 이용하여 다중 트레이닝 잡을 처리하는 경우, 잡을 모듈화하여 실행하므로 새로운 잡의 마이크로서비스가 제공되었을 때에 실행 중인 잡에 대한 선취 방식의 처리가 없이도 바로 스케줄링이 가능해진다. 일 실시예에서는 이러한 방식으로 자원 사용에 대한 공평성을 향상시킬 수 있다.

[0073] 예를 들어, 해당 잡의 공평성에 따라 사용하는 자원 양을 줄여야 하는 경우, 잡 관리자(216)는 클러스터의 상태를 파악하여 가상 작업자의 수를 줄여서 마이크로서비스들을 계속적으로 제공할 수 있다. 또는 잡 관리자(216)는 이와 반대로 클러스터에 자원이 남은 경우, 가상 작업자의 수를 증가시켜 자원 효율성을 높이는 한편, 자원 활용에 있어서의 탄력성 (elasticity)를 높일 수 있다. 일 실시예에서는 마이크로서비스를 기반으로 동적으로 잡을 처리함으로써 잡들 간의 공평성(fairness)을 고려한 스케줄링을 가능케 한다. 또한, 일 실시예에 따르면, 전체적으로 자원 관리 및 스케줄링에 있어서 각 마이크로서비스를 스케줄링 함으로 자원 관리의 확장성 (scalability)을 높일 수 있다.

[0074] 자원 관리자(226)는 스케줄러(223)의 스케줄링에 따라 사용되는 가상 작업자들의 개수를 동적으로 조절할 수 있다. 자원 관리자(226)는 예를 들어, 클러스터들 각각의 자원 사용 상태에 기초하여 잡에 대응하는 가상 작업자들의 수를 동적으로 조절할 수 있다. 예를 들어, 잡이 복수 개인 경우, 자원 관리자(226)는 클라우드 상에 남은 자원이 많을 때, 이를 효율적으로 사용하기 위해 특정 잡에 대한 가상 작업자를 추가하여 트레이닝 시스템의 쓰루풋(throughput)을 향상시킬 수 있다.

[0075] 폴트 핸들러(Fault Handler)(230)는 마이크로서비스들 중 오류(fault)가 발생한 마이크로서비스를 검출하고, 상기 검출된 마이크로서비스의 동작을 재수행할 수 있다.

[0076] 액션 데이터베이스(240)는 기존의 모노리식(monolithic)한 트레이닝 잡을 마이크로서비스의 함수 단위로 작게 쪼개 실행 단위인 액션(action)을 저장할 수 있다. 이 액션은 포워드 패스 또는 백워드 패스 과정의 일부 단계이거나 매개 변수를 처리하는 과정일 수 있다.

[0077] 분산 파일 시스템(Distributed File System)(250)은 예를 들어, 트레이닝을 위한 트레이닝 데이터, 및 트레이닝이 종료된 후의 최종 매개 변수 등과 같이 영구적인 데이터(permanent data)를 저장할 수 있다. 트레이닝 데이터는 예를 들어, 입력 관리자(213)가 사용자로부터 수신한 트레이닝 데이터일 수 있다. 분산 파일 시스템 (250)은 예를 들어, 하둡 분산 파일 시스템(Hadoop Distributed File System; HDFS)일 수 있다.

[0078] 분산된 인-메모리 데이터베이스(Distributed In - memory Database)(260)는 예를 들어, 트레이닝 과정에서 지역적으로 계산된 그래디언트와 같은 중간 데이터를 저장할 수 있다. 분산된 인-메모리 데이터베이스(260)는 예를 들어, 키(Key)와 값(value)이 매핑된 단순한 맵 데이터 저장소인 레디스 클러스터(redis cluster)에 의해 구현될 수 있다. 레디스 클러스터는 고성능 키 값(key-value) 저장소로서, 예를 들어, 리스트, 해시, 정렬된 셋 등 여러 형식의 자료 구조를 지원하는 NoSQL이다. 레디스 클러스터는 서로 다른 클러스터들 혹은 서로 다른 클러스터들의 프로세싱 유닛들의 메모리에 상주하면서 관계형 데이터베이스 관리 시스템(Relational Database Management System; RDBMS)의 캐시 솔루션으로서 주로 사용될 수 있다.

[0080] 도 3은 일 실시예에 따른 트레이닝 시스템의 동작 방법을 나타낸 흐름도이다. 도 3을 참조하면, 일 실시예에 따른 트레이닝 시스템은 트레이닝을 위한 잡을, 신경망 모델의 가상 작업자들에 대응하는 태스크들로 분할한다(310). 태스크들은 마이크로서비스들에 의해 수행된다. 잡은 예를 들어, 포워드 패스(forward pass) 또는 백워드 패스(backward pass) 중 적어도 하나의 패스로 구성되는 하나 또는 복수의 미니 배치들로 분할될 수 있다.

[0081] 트레이닝 시스템은 마이크로서비스들을 신경망 모델을 위한 클러스터들 각각에 포함된 이종의 그래픽 프로세싱 유닛들에게 스케줄링한다(320). 트레이닝 시스템은 예를 들어, 트레이닝을 위한 트레이닝 데이터에 대해 전처리를 수행하고, 전처리된 트레이닝 데이터에 대한 계산 유형에 따라 태스크들을 마이크로서비스들로 분할할 수 있다. 트레이닝 시스템은 마이크로서비스들을 이종의 그래픽 프로세싱 유닛들에게 스케줄링할 수 있다. 트레이닝 시스템은 예를 들어, 태스크들을, 트레이닝을 수행하는 제1 계산 유형의 마이크로서비스와 트레이닝에 따른 매

개 변수를 산출하는 제2 계산 유형의 마이크로서비스로 분류할 수 있다. 트레이닝 시스템은 예를 들어, 트레이닝에 의해 발생하는 매개 변수에 대한 동기화 정책을 기초로, 분류된 마이크로서비스들에 대한 정보를 스케줄러에게 전달할 수 있다.

- [0082] 또는 트레이닝 시스템은 예를 들어, 가상 작업자들 각각이 마이크로서비스들에 의해 처리하는 미니-배치에 대한 트레이닝에 의해 발생하는 매개 변수에 대한 동기화 정책을 기초로, 복수의 마이크로서비스들을 이종의 프로세싱 유닛들에게 동적으로 스케줄링할 수 있다.
- [0083] 트레이닝 시스템은 스케줄링에 따라 자원을 할당할 수 있다.
- [0084] 트레이닝 시스템은 단계(320)의 스케줄링을 기초로, 가상 작업자들에 의해 신경망 모델을 트레이닝 한다(330).
- [0086] 도 4a 내지 도 4c는 일 실시예에 따른 트레이닝 시스템이 복수의 트레이닝 잡을 수행하는 동작을 설명하기 위한 도면이다. 도 4a 내지 도 4c를 참조하면, 복수의 트레이닝 잡들은 제1 내지 제3 트레이닝 잡들(410, 420, 430)을 포함할 수 있다. 제1 내지 제3 트레이닝 잡들(410, 420, 430) 각각은 서로 다른 트레이닝 모델에 대한 트레이닝을 수행할 수 있다. 제1 내지 제3 트레이닝 잡들(410, 420, 430) 각각은 하나 이상의 가상 작업자들에 의해 실행될 수 있다.
- [0087] 제1 트레이닝 잡(410)은 제1 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3, A_Agg 1, A_Com 2-1, A_Com 2-2, A_Com 2-3, A_Agg 2)의 시퀀스로 실행될 수 있다. 제1 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3, A_Agg 1, A_Com 2-1, A_Com 2-2, A_Com 2-3, A_Agg 2)은 제1 유형의 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3, A_Com 2-1, A_Com 2-2, A_Com 2-3) 및 제2 유형의 마이크로서비스들(A_Agg 1, A_Agg 2)로 분류될 수 있다.
- [0088] 제1 유형의 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3, A_Com 2-1, A_Com 2-2, A_Com 2-3) 각각은 하나 이상의 미니 배치들의 처리, 예를 들어, 포워드 패스 또는 백워드 패스를 의미할 수 있다. 제1 유형의 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3, A_Com 2-1, A_Com 2-2, A_Com 2-3)은 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3) 및 제2 부분 마이크로서비스들(A_Com 2-1, A_Com 2-2, A_Com 2-3)을 포함할 수 있다.
- [0089] 제2 유형의 마이크로서비스들(A_Agg 1, A_Agg 2)은 제1 총합 마이크로서비스(A_Agg 1) 및 제2 총합 마이크로서비스(A_Agg 2)를 포함할 수 있다. 제1 총합 마이크로서비스(A_Agg 1)는 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3)에 의해 생성된 매개 변수들을 총합하고, 제2 총합 마이크로서비스(A_Agg 2)는 제2 부분 마이크로서비스들(A_Com 2-1, A_Com 2-2, A_Com 2-3)에 의해 생성된 매개 변수들을 총합할 수 있다.
- [0090] 즉, 제1 트레이닝 잡(410)은 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3), 제1 총합 마이크로서비스(A_Agg 1), 제2 부분 마이크로서비스들(A_Com 2-1, A_Com 2-2, A_Com 2-3), 및 제2 총합 마이크로서비스(A_Agg 2)의 시퀀스로 실행될 수 있다.
- [0091] 실시 예에 따라, 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3)은 하나 이상의 가상 작업자들에 의해 병렬적으로 실행되고, 제1 총합 마이크로서비스(A_Agg 1)는 하나 이상의 가상 작업자들에 의해 함께 실행될 수 있다. 예를 들어, 제1 가상 작업자는 제1-1 마이크로서비스(A_Com 1-1)를 실행할 수 있고, 제2 가상 작업자는 제1-2 마이크로서비스(A_Com 1-2)를 실행할 수 있고, 제3 가상 작업자는 제1-3 마이크로서비스(A_Com 1-3)를 실행할 수 있다. 이 경우, 제1 내지 제3 가상 작업자들은 제1 총합 마이크로서비스(A_Agg 1)를 함께 실행할 수 있다.
- [0092] 실시 예에 따라, 제2 부분 마이크로서비스들(A_Com 2-1, A_Com 2-2, A_Com 2-3)은 하나 이상의 가상 작업자들에 의해 병렬적으로 실행되고, 제2 총합 마이크로서비스(A_Agg 2)는 하나 이상의 가상 작업자들에 의해 함께 실행될 수 있다. 예를 들어, 제1 가상 작업자는 제2-1 마이크로서비스(A_Com 2-1)를 실행할 수 있고, 제2 가상 작업자는 제2-2 마이크로서비스(A_Com 2-2)를 실행할 수 있고, 제3 가상 작업자는 제2-3 마이크로서비스(A_Com 2-3)를 실행할 수 있다. 이 경우, 제1 내지 제3 가상 작업자들은 제2 총합 마이크로서비스(A_Agg 2)를 함께 실행할 수 있다.
- [0093] 제1 트레이닝 잡(410)에 대한 제1 잡 프로키는 제1 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3, A_Agg 1, A_Com 2-1, A_Com 2-2, A_Com 2-3, A_Agg 2)을 스케줄러(440)에 제출할 수 있다. 이 때, 제1 잡 프로키는 제1 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3, A_Agg 1, A_Com 2-1, A_Com 2-2, A_Com 2-3, A_Agg

2)의 정보를 함께 전달할 수 있다. 예를 들어 제1 잡 프록시는 스케줄러(440)에 제1 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3, A_Agg 1, A_Com 2-1, A_Com 2-2, A_Com 2-3, A_Agg 2)의 유형 정보 및 시퀀스 정보를 전달할 수 있다.

[0094] 제2 트레이닝 잡(420)은 제2 마이크로서비스들(B_Com 1-1, B_Com 1-2, B_Agg 1, B_Com 2-1, B_Com 2-2, B_Agg 2)의 시퀀스로 실행될 수 있다. 제2 트레이닝 잡(420)에 대한 제2 잡 프록시는 제2 마이크로서비스들(B_Com 1-1, B_Com 1-2, B_Agg 1, B_Com 2-1, B_Com 2-2, B_Agg 2)을 스케줄러(440)에 제출할 수 있다. 제3 트레이닝 잡(430)은 제3 마이크로서비스들(C_Com 1-1, C_Com 1-2, C_Agg 1, C_Com 2-1, C_Com 2-2, C_Agg 2)의 시퀀스로 실행될 수 있다. 제3 트레이닝 잡(430)에 대한 제3 잡 프록시는 제3 마이크로서비스들(C_Com 1-1, C_Com 1-2, C_Agg 1, C_Com 2-1, C_Com 2-2, C_Agg 2)을 스케줄러(440)에 제출할 수 있다. 이하 제2 트레이닝 잡(420) 및 제3 트레이닝 잡(430)에 대한 설명은 제1 트레이닝 잡(410)에 대한 설명과 유사하므로, 상세한 설명은 생략된다.

[0095] 스케줄러(440)는 제1 내지 제3 마이크로서비스들을 클러스터에 포함된 이중 그래픽 프로세싱 유닛(GPU)들에게 스케줄링할 수 있다. 실시 예에 따라, 스케줄러(440)는 마이크로서비스들 각각의 로컬리티 또는 서비스 품질(QoS) 중 적어도 하나를 기초로, 마이크로서비스들을 이중의 그래픽 프로세싱 유닛들에게 스케줄링할 수 있다.

[0096] 도 4a 및 도 4b를 참조하면, 스케줄러(440)에는 제1 내지 제3 마이크로서비스들 중 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3)이 제출되어 있을 수 있다. 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3)은 클러스터(450)에 스케줄링될 수 있다.

[0097] 클러스터(450)는 제1 노드(Node 1), 제2 노드(Node 2), 및 제3 노드(Node 3)를 포함할 수 있다. 제1 노드(Node 1)에는 제1 GPU들(GPU_X1, GPU_X2)이 배치될 수 있다. 제2 노드(Node 2)에는 제2 GPU들(GPU_Y1, GPU_Y2, GPU_Y3)이 배치될 수 있다. 제3 노드(Node 3)에는 제3 GPU들(GPU_Z1, GPU_Z2, GPU_Z3)이 배치될 수 있다. 제1 GPU들(GPU_X1, GPU_X2), 제2 GPU들(GPU_Y1, GPU_Y2, GPU_Y3), 및 제3 GPU들(GPU_Z1, GPU_Z2, GPU_Z3)은 동종 또는 이종일 수 있다.

[0098] 스케줄러(440)는 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3)의 로컬리티를 고려하여 클러스터(450)에 스케줄링할 수 있다. 예를 들어, 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3) 각각은 제2 GPU들(GPU_Y1, GPU_Y2, GPU_Y3)에 스케줄링될 수 있다. 이 경우, 제2 GPU들(GPU_Y1, GPU_Y2, GPU_Y3)은 제2 노드(Node 2)에 배치되어 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3)은 같은 노드에서 실행될 수 있다. 마이크로서비스들이 같은 노드에서 실행되는 경우, 클러스터(450)는 계산된 가중치를 같은 노드에 저장할 수 있다. 이로써, 제1 총합 마이크로서비스(A_Agg 1)는 가중치 총합에 있어서 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3)의 가중치를 빠르게 읽어 올 수 있다.

[0099] 도 4a 내지 도 4c를 참조하면, 스케줄러(440)에는 제1 내지 제3 마이크로서비스들 중 제2 부분 마이크로서비스들(A_Com 2-1, A_Com 2-2, A_Com 2-3)이 제출되어 있을 수 있다. 제2 부분 마이크로서비스들(A_Com 2-1, A_Com 2-2, A_Com 2-3)은 클러스터(450)에 스케줄링될 수 있다.

[0100] 실시 예에 따라, 제1 GPU들(GPU_X1, GPU_X2), 제2 GPU들(GPU_Y1, GPU_Y2, GPU_Y3), 및 제3 GPU들(GPU_Z1, GPU_Z2, GPU_Z3)은 이종이고, 제3 GPU들(GPU_Z1, GPU_Z2, GPU_Z3)의 성능이 제1 GPU들(GPU_X1, GPU_X2) 및 제2 GPU들(GPU_Y1, GPU_Y2, GPU_Y3) 보다 좋을 수 있다. 스케줄러(440)는 제2 부분 마이크로서비스들(A_Com 2-1, A_Com 2-2, A_Com 2-3)를 서비스 품질을 고려하여 클러스터(450)에 스케줄링할 수 있다.

[0101] 스케줄러(440)는 제1 부분 마이크로서비스들(A_Com 1-1, A_Com 1-2, A_Com 1-3)이 제2 GPU들(GPU_Y1, GPU_Y2, GPU_Y3)에서 실행되었을 때 제1 트레이닝 잡(410)의 쓰루풋을 모니터링할 수 있다. 제1 트레이닝 잡(410)의 쓰루풋이 설정된 쓰루풋 보다 낮을 때, 제2 부분 마이크로서비스들(A_Com 2-1, A_Com 2-2, A_Com 2-3)을 제3 GPU들(GPU_Z1, GPU_Z2, GPU_Z3)에 스케줄링할 수 있다.

[0102] 이 경우, 스케줄러(440)는 제3 GPU들(GPU_Z1, GPU_Z2, GPU_Z3)에서 다른 마이크로서비스들이 실행되고 있다면, 다른 마이크로서비스들의 실행이 끝난 후 바로 제2 부분 마이크로서비스들(A_Com 2-1, A_Com 2-2, A_Com 2-3)이 실행되도록 스케줄링할 수 있다. 즉, 스케줄러(440)는 제3 GPU들(GPU_Z1, GPU_Z2, GPU_Z3)에서 실행되고 있는 트레이닝 잡이 끝날 때까지 기다리거나 그 트레이닝 잡에 대한 선택 처리 없이, 마이크로서비스 기반으로 효율적인 스케줄링을 수행할 수 있다.

[0103] 통상의 작업자(worker)는 하나의 GPU를 독점적(monolithic)으로 사용하여 하나의 트레이닝 잡을 수행하였지만, 개시하는 시스템에서는 마이크로서비스 실행 기법을 기반으로 이중의 GPU들을 독점하지 않고 유연하게 사용하여

복수의 트레이닝 잡들을 수행할 수 있다. 즉, 마이크로서비스 기반의 트레이닝 시스템은 하나의 이기종 GPU 클러스터를 사용하여 복수의 트레이닝 잡들을 병렬적으로 실행함으로써 복수의 트레이닝 모델들에 대한 트레이닝을 수행할 수 있다.

- [0105] 도 5는 일 실시예에 따른 트레이닝 시스템의 블록도이다. 도 5를 참조하면, 일 실시예에 따른 트레이닝 시스템(500)은 통신 인터페이스(510), 프로세서(530) 및 메모리(550)를 포함할 수 있다. 통신 인터페이스(510), 프로세서(530) 및 메모리(550)는 통신 버스(505)를 통해 서로 통신할 수 있다.
- [0106] 통신 인터페이스(510)는 사용자로부터 트레이닝 데이터를 포함하는 트레이닝 요청을 수신할 수 있다.
- [0107] 프로세서(530)는 트레이닝 요청에 응답하여 트레이닝을 위한 잡을, 신경망 모델의 가상 작업자들에 대응하는 태스크들로 분할할 수 있다. 이때, 태스크들은 마이크로서비스들의 형태로 수행될 수 있다. 프로세서(530)는 마이크로서비스들을 신경망 모델을 위한 클러스터들 각각에 포함된 이종의 그래픽 프로세싱 유닛들에게 스케줄링할 수 있다. 프로세서(530)는 스케줄링을 기초로, 가상 작업자들에 의해 신경망 모델을 트레이닝할 수 있다.
- [0108] 메모리(550)는 상술한 프로세서(530)의 처리 과정에서 생성되는 다양한 정보들을 저장할 수 있다. 이 밖에도, 메모리(550)는 각종 데이터와 프로그램 등을 저장할 수 있다. 메모리(550)는 휘발성 메모리 또는 비휘발성 메모리를 포함할 수 있다. 메모리(550)는 하드 디스크 등과 같은 대용량 저장 매체를 구비하여 각종 데이터를 저장할 수 있다.
- [0109] 이 밖에도, 프로세서(530)는 도 1 내지 도 4를 통해 전술한 적어도 하나의 방법 또는 적어도 하나의 방법에 대응되는 알고리즘을 수행할 수 있다. 프로세서(530)는 목적하는 동작들(desired operations)을 실행시키기 위한 물리적인 구조를 갖는 회로를 가지는 하드웨어로 구현된 데이터 처리 장치일 수 있다. 예를 들어, 목적하는 동작들은 프로그램에 포함된 코드(code) 또는 인스트럭션들(instructions)을 포함할 수 있다. 예를 들어, 하드웨어로 구현된 예측 장치는 마이크로프로세서(microprocessor), 중앙 처리 장치(central processing unit), 프로세서 코어(processor core), 멀티-코어 프로세서(multi-core processor), 멀티프로세서(multiprocessor), ASIC(Application-Specific Integrated Circuit), FPGA(Field Programmable Gate Array)를 포함할 수 있다.
- [0110] 프로세서(530)는 프로그램을 실행하고, 트레이닝 시스템(500)을 제어할 수 있다. 프로세서(530)에 의하여 실행되는 프로그램 코드는 메모리(550)에 저장될 수 있다. 프로세서(530)는 예를 들어, CPU(Central Processing Unit), GPU(Graphics Processing Unit), 또는 NPU(Neural network Processing Unit)으로 구성될 수 있다.
- [0112] 실시 예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.
- [0113] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embodiment)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0114] 이상과 같이 실시예들이 비록 한정된 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기를 기초로 다양한 기술적 수정 및 변형을 적용할 수 있다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

[0115] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 청구범위의 범위에 속한다.

부호의 설명

[0116] 500: 트레이닝 시스템

505: 통신 버스

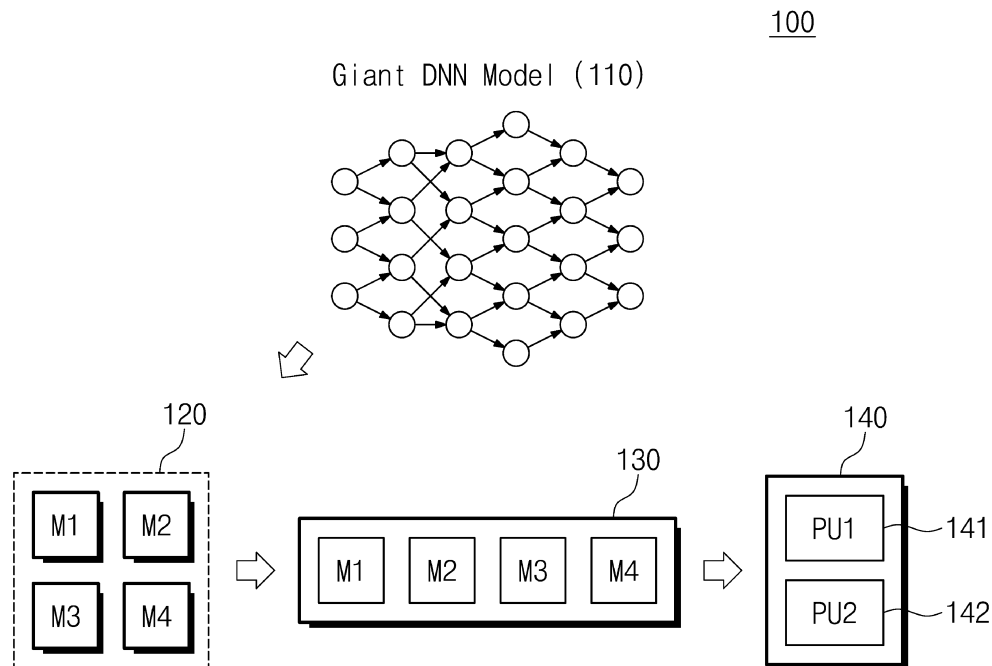
510: 통신 인터페이스

530: 프로세서

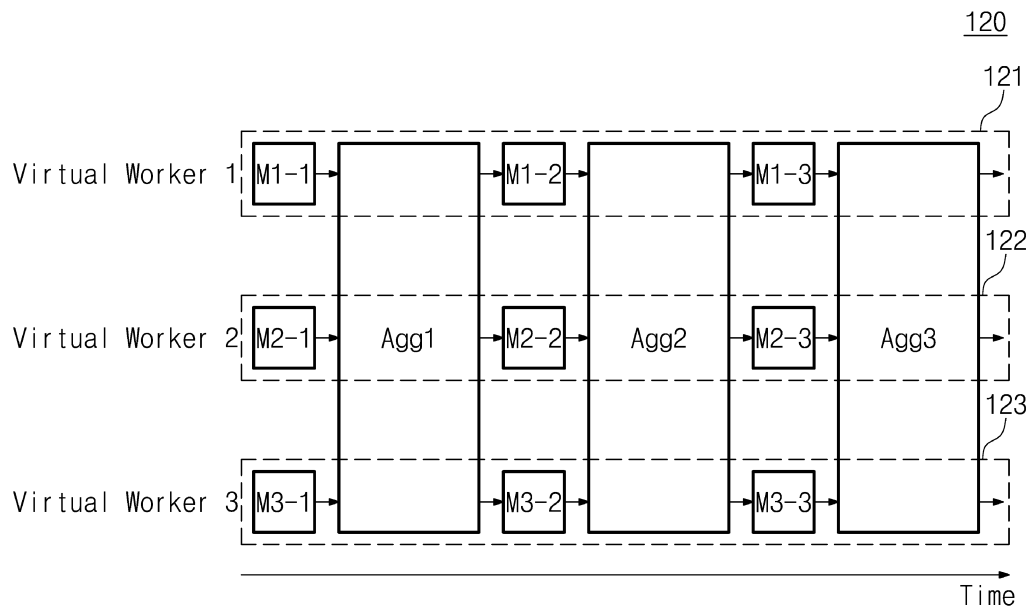
550: 메모리

도면

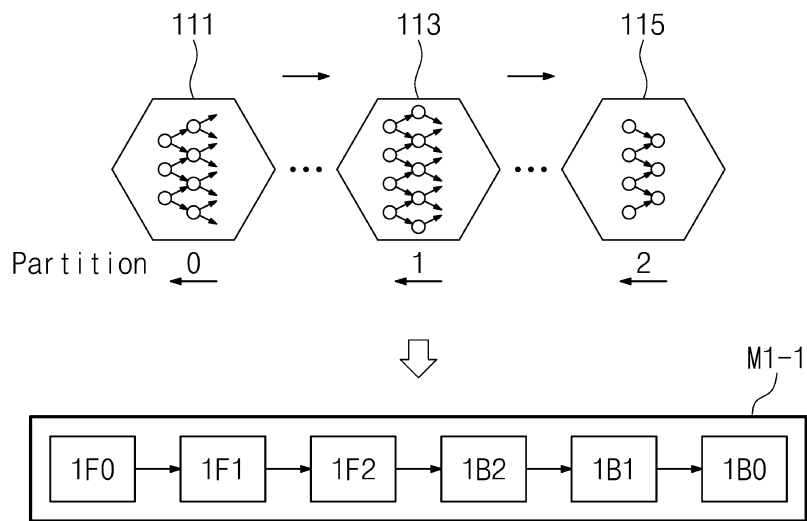
도면1a



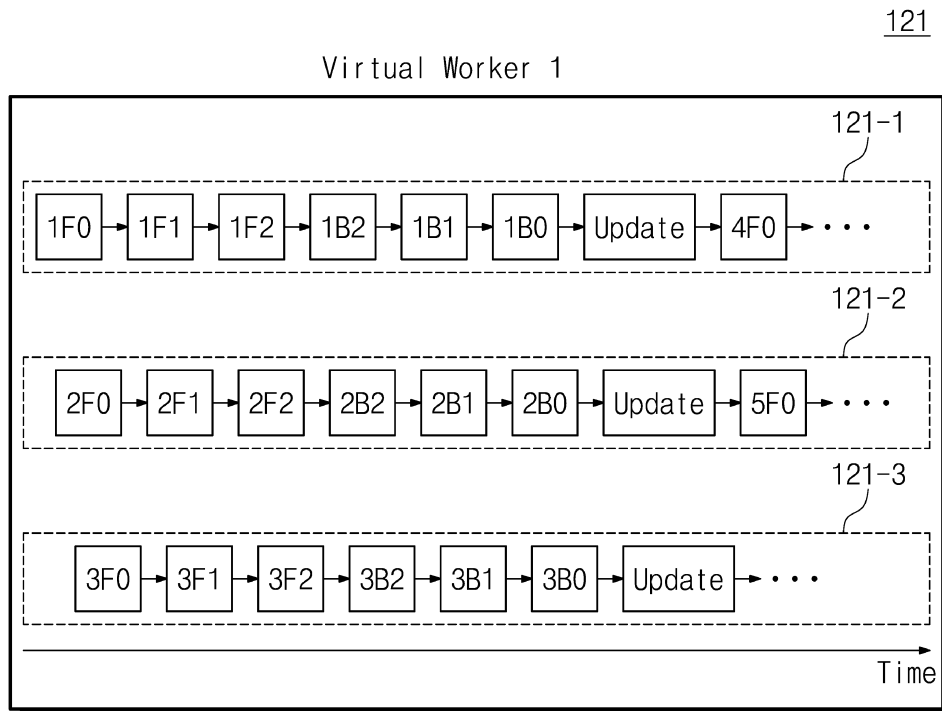
도면1b



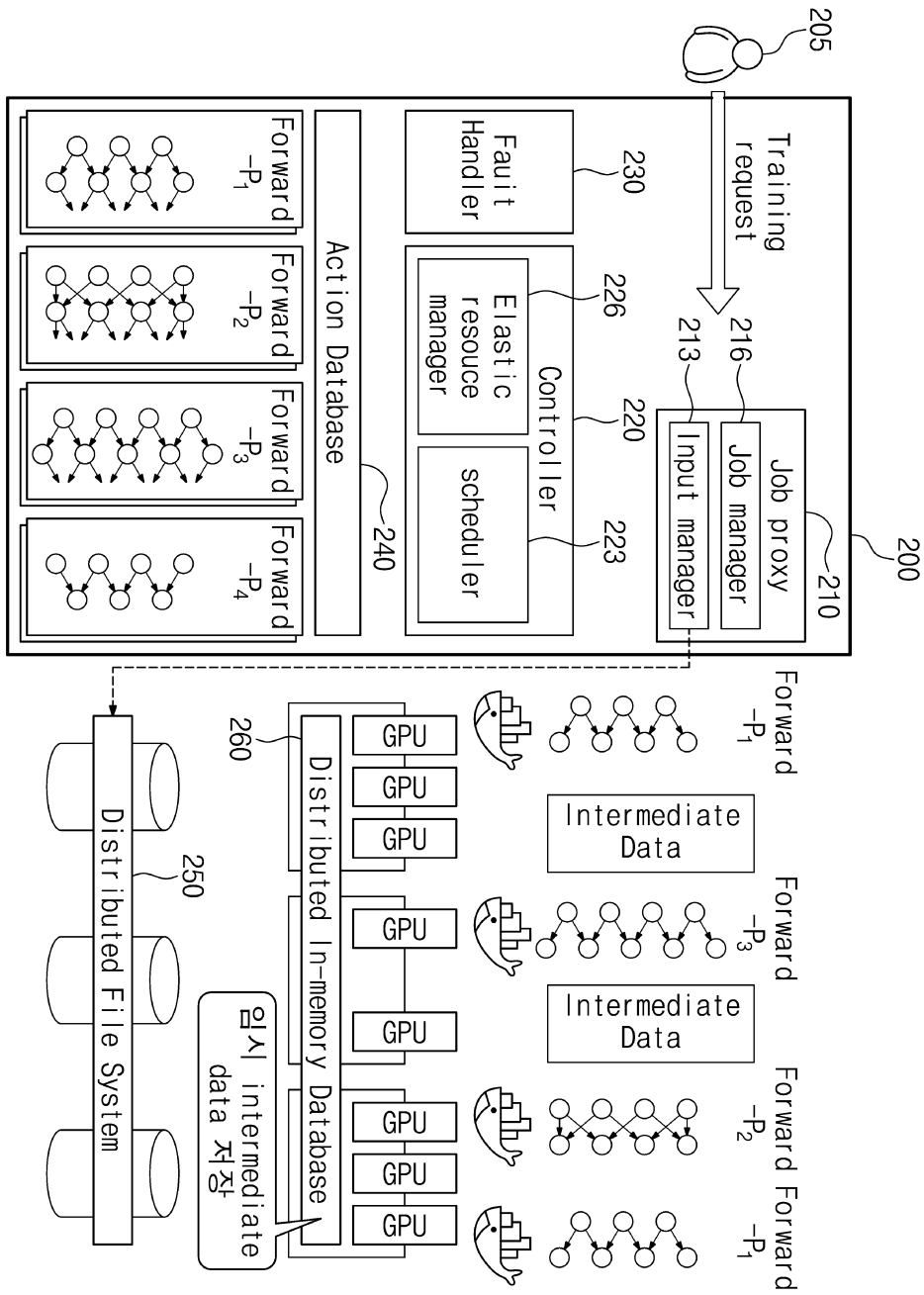
도면1c



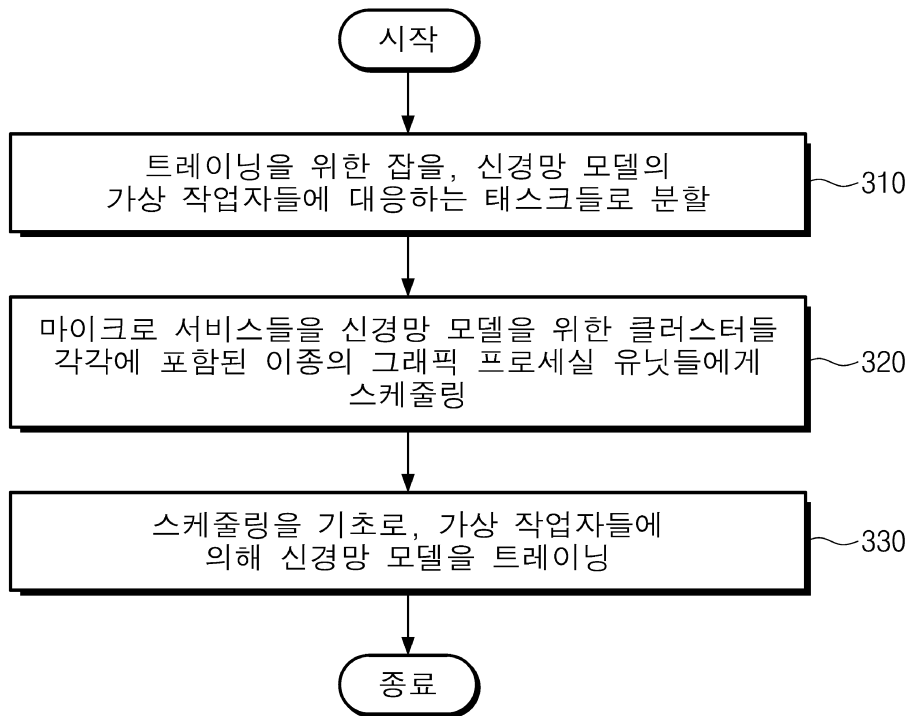
도면1d



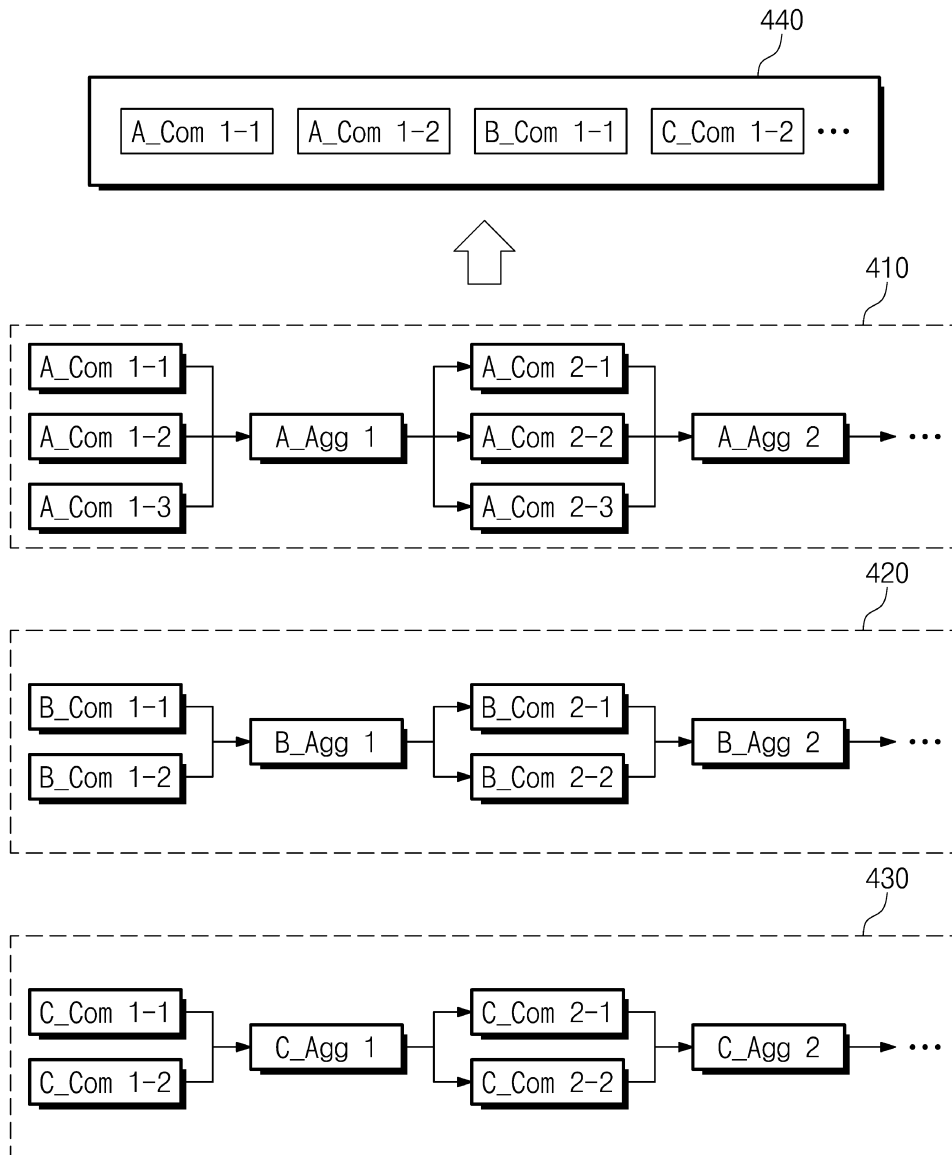
도면2



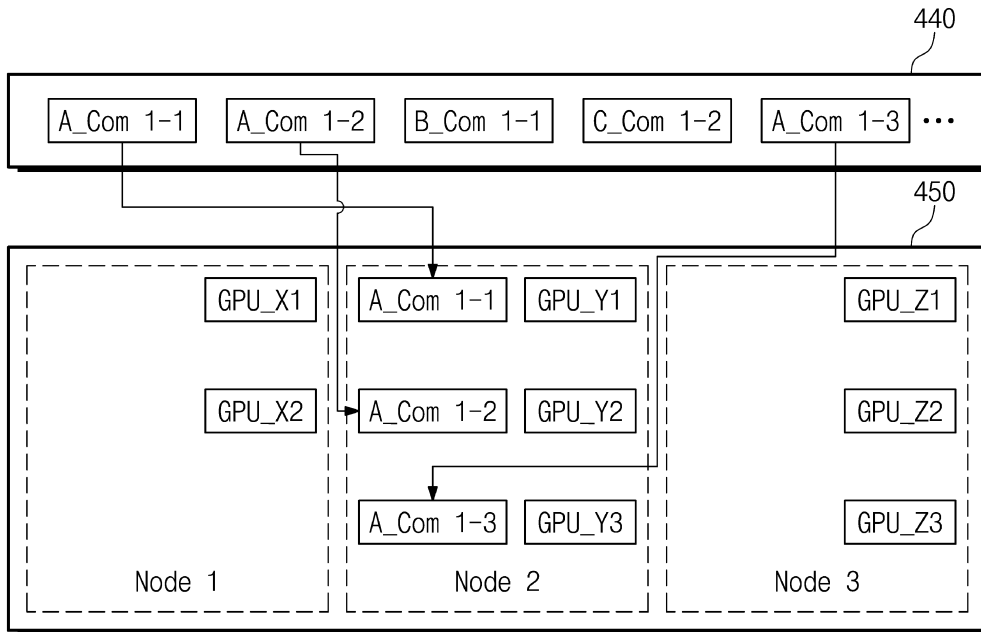
도면3



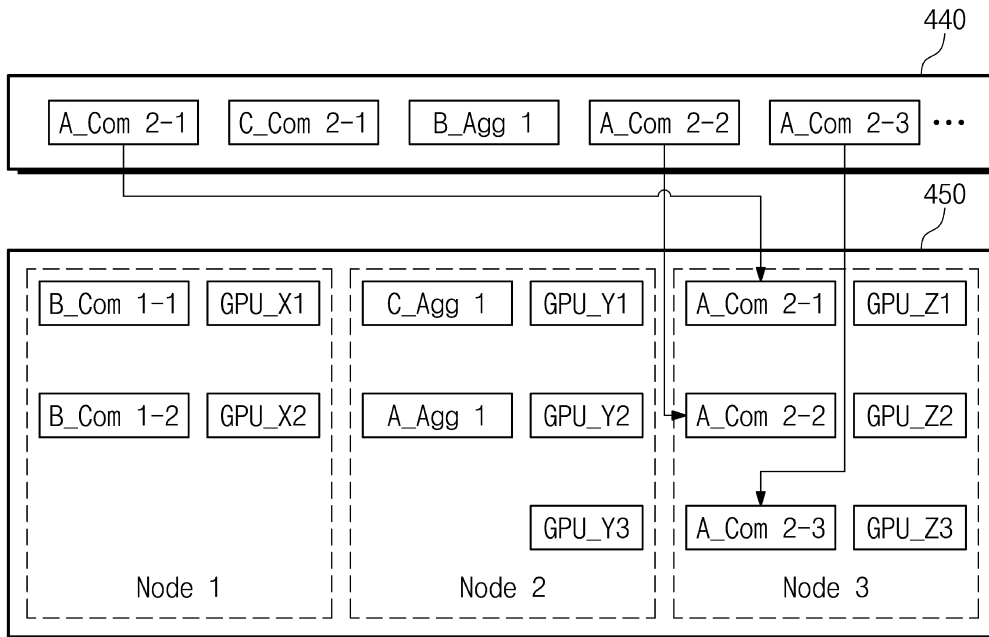
도면4a



도면4b



도면4c



도면5

