



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0108239
(43) 공개일자 2024년07월09일

(51) 국제특허분류(Int. Cl.)
G06N 3/098 (2023.01) G06F 9/50 (2018.01)
G06N 3/063 (2023.01)
(52) CPC특허분류
G06N 3/098 (2023.01)
G06F 9/5027 (2013.01)
(21) 출원번호 10-2023-0153939
(22) 출원일자 2023년11월08일
심사청구일자 없음
(30) 우선권주장
1020220191027 2022년12월30일 대한민국(KR)

(71) 출원인
울산과학기술원
울산광역시 울주군 언양읍 유니스트길 50
(72) 발명자
최영리
울산광역시 울주군 언양읍 유니스트길 50
정연혁
울산광역시 울주군 언양읍 유니스트길 50
(뒷면에 계속)
(74) 대리인
리앤목특허법인

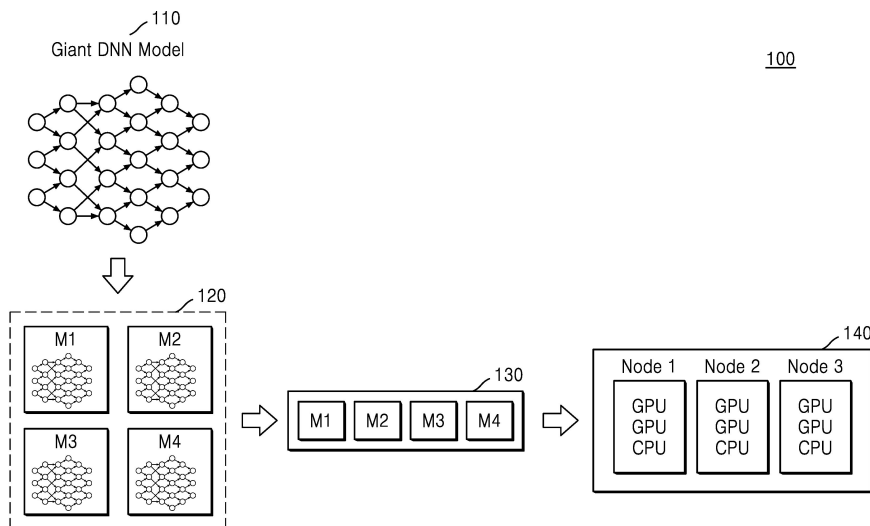
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 트레이닝 시스템 및 그 동작 방법

(57) 요약

트레이닝 시스템 및 그 동작 방법이 제공된다. 트레이닝 시스템은 신경망 모델에 대응하는 트레이닝 잡(training job)을 복수의 논리 작업자에 의해 실행되는 복수의 마이크로서비스(a plurality of microservices)로 분할하는 잡 프록시(job proxy), 및 상기 복수의 마이크로서비스를 복수의 프로세싱 유닛(a plurality of processing units)에 스케줄링하는 스케줄러(scheduler)를 포함하고, 상기 복수의 마이크로서비스는 상기 복수의 논리 작업자 중 제1 논리 작업자에 의해 실행되는 복수의 제1 마이크로서비스 및 상기 복수의 논리 작업자 중 제2 논리 작업자에 의해 실행되는 복수의 제2 마이크로서비스를 포함하고, 상기 스케줄러는, 상기 복수의 프로세싱 유닛의 가용 상태에 기초하여, 상기 복수의 제1 마이크로서비스 및 상기 복수의 제2 마이크로서비스를 상기 복수의 프로세싱 유닛 중 어느 하나의 프로세싱 유닛에 스케줄링한다.

대표도



(52) CPC특허분류

G06N 3/063 (2013.01)

(72) 발명자

이승민

울산광역시 울주군 언양읍 유니스트길 50

주성현

울산광역시 울주군 언양읍 유니스트길 50

명세서

청구범위

청구항 1

신경망 모델에 대응하는 트레이닝 잡(training job)을 복수의 논리 작업자에 의해 실행되는 복수의 마이크로서비스(a plurality of microservices)로 분할하는 잡 프록시(job proxy); 및

상기 복수의 마이크로서비스를 복수의 프로세싱 유닛(a plurality of processing units)에 스케줄링하는 스케줄러(scheduler)를 포함하고,

상기 복수의 마이크로서비스는 상기 복수의 논리 작업자 중 제1 논리 작업자에 의해 실행되는 복수의 제1 마이크로서비스 및 상기 복수의 논리 작업자 중 제2 논리 작업자에 의해 실행되는 복수의 제2 마이크로서비스를 포함하고,

상기 스케줄러는, 상기 복수의 프로세싱 유닛의 가용 상태에 기초하여, 상기 복수의 제1 마이크로서비스 및 상기 복수의 제2 마이크로서비스를 상기 복수의 프로세싱 유닛 중 어느 하나의 프로세싱 유닛에 스케줄링하는, 트레이닝 시스템.

청구항 2

제1 항에 있어서,

상기 스케줄러는, 가용 프로세싱 유닛(available processing unit)의 개수가 상기 복수의 논리 작업자의 개수보다 적음에 따라, 상기 복수의 제1 마이크로서비스 및 상기 복수의 제2 마이크로서비스를 상기 어느 하나의 프로세싱 유닛에 순차적으로 스케줄링하는, 트레이닝 시스템.

청구항 3

제1 항에 있어서,

상기 스케줄러는, 상기 복수의 제1 마이크로서비스와 상기 복수의 제2 마이크로서비스를 동일한 컨테이너(container)에 스케줄링하는, 트레이닝 시스템.

청구항 4

제1 항에 있어서,

상기 복수의 마이크로서비스는, 상기 트레이닝 잡에 대한 트레이닝 데이터가 분할된 복수의 미니 배치(minibatch)를 처리하는 함수를 포함하고,

상기 스케줄러는 상기 어느 하나의 프로세싱 유닛에서 상기 복수의 제1 마이크로서비스 중 어느 하나의 미니 배치 처리와 상기 복수의 제2 마이크로서비스 중 어느 하나의 미니 배치 처리가 복수의 단계(multiple phases)로 실행되도록 스케줄링하는, 트레이닝 시스템.

청구항 5

제1 항에 있어서,

자원 관리자를 더 포함하고,

상기 자원 관리자는, 상기 신경망 모델에 대응하는 상기 트레이닝 잡에 상기 복수의 프로세싱 유닛을 2^n 단위로 할당하는, 트레이닝 시스템(n 은 0 또는 자연수).

청구항 6

제1 항에 있어서,

자원 관리자를 더 포함하고,

상기 자원 관리자는, 상기 신경망 모델에 대응하는 상기 트레이닝 잡이 2k개의 논리 작업자를 포함하는 경우, 상기 복수의 프로세싱 유닛을 2k의 약수 중 어느 하나의 단위로 할당하고,

상기 트레이닝 잡이 2k-1개의 논리 작업자를 포함하는 경우, 상기 복수의 프로세싱 유닛을 2k-1의 약수 중 어느 하나 또는 2k의 약수 중 1과 2k를 제외한 어느 하나의 단위로 할당하는, 트레이닝 시스템(k는 자연수).

청구항 7

제1 항에 있어서,

복수의 신경망 모델에 대응하는 복수의 트레이닝 잡 각각에 상기 복수의 프로세싱 유닛을 할당하는 자원 관리자를 더 포함하고,

상기 자원 관리자는,

최단 잔여 서비스 시간(shortest remaining service time)을 갖는 트레이닝 잡부터 순서대로 클러스터에 존재하는 프로세싱 유닛을 할당하고,

상기 클러스터에 잔여 프로세싱 유닛(remaining processing unit)이 존재함에 따라, 필요한 프로세싱 유닛의 수보다 적은 수의 프로세싱 유닛이 할당된 트레이닝 잡에 상기 잔여 프로세싱 유닛을 할당하는, 트레이닝 시스템.

청구항 8

제1 항에 있어서,

복수의 신경망 모델에 대응하는 복수의 트레이닝 잡 각각에 상기 복수의 프로세싱 유닛을 할당하는 자원 관리자를 더 포함하고,

상기 자원 관리자는,

상기 복수의 트레이닝 잡 각각에 프로세싱 유닛을 할당하고,

큐(queue)에 저장된 대기 트레이닝 잡(queueing training job)이 존재함에 따라, 대기 트레이닝 잡의 대기 시간(queueing time)이 하나 이상의 트레이닝 잡이 복수의 단계(multiple phases)로 실행됨에 따른 예상 증가 시간(expected increase time)보다 긴 경우, 상기 대기 트레이닝 잡에 상기 프로세싱 유닛을 재할당하는, 트레이닝 시스템.

청구항 9

제1 항에 있어서,

상기 복수의 마이크로서비스는,

복수의 미니 배치 각각에 대한 가중치를 계산하는 계산 함수; 및

상기 복수의 미니 배치 각각에 대한 가중치를 총합한 글로벌 파라미터를 계산하는 집계 함수를 포함하는, 트레이닝 시스템.

청구항 10

제9 항에 있어서,

상기 복수의 제1 마이크로서비스의 제1 계산 함수와 상기 복수의 제2 마이크로서비스의 제2 계산 함수가 동일한 인터레이션에서 순차적으로 실행되는, 트레이닝 시스템.

청구항 11

제9 항에 있어서,

상기 복수의 제1 마이크로서비스의 제1 계산 함수는, 상기 글로벌 파라미터(global parameter)를 읽고, 상기 글로벌 파라미터를 상기 복수의 제2 마이크로서비스의 제2 계산 함수에 전달하는, 트레이닝 시스템.

청구항 12

제1 항에 있어서,

상기 복수의 마이크로서비스는, 상기 복수의 논리 작업자 중 제3 논리 작업자에 의해 실행되는 복수의 제3 마이크로서비스 및 상기 복수의 논리 작업자 중 제4 논리 작업자에 의해 실행되는 복수의 제4 마이크로서비스를 포함하고,

상기 스케줄러는, 상기 복수의 제3 마이크로서비스 및 상기 복수의 제4 마이크로서비스를 상기 복수의 프로세싱 유닛 중 다른 하나의 프로세싱 유닛에 스케줄링하고,

상기 복수의 제3 마이크로서비스는 상기 복수의 제1 마이크로서비스와 병렬적으로 실행되는, 트레이닝 시스템.

청구항 13

신경망 모델에 대응하는 트레이닝 잡을 복수의 논리 작업자 각각에 의해 실행되는 복수의 마이크로서비스로 분할하는 단계; 및

상기 복수의 마이크로서비스를 복수의 프로세싱 유닛에 스케줄링하는 단계를 포함하고,

상기 스케줄링하는 단계는, 상기 복수의 프로세싱 유닛의 가용 상태에 기초하여, 상기 복수의 논리 작업자 중 제1 논리 작업자에 의해 실행되는 복수의 제1 마이크로서비스 및 상기 복수의 논리 작업자 중 제2 논리 작업자에 의해 실행되는 복수의 제2 마이크로서비스를 상기 복수의 프로세싱 유닛 중 어느 하나의 프로세싱 유닛에 스케줄링하는 단계를 포함하는, 트레이닝 시스템의 동작 방법.

청구항 14

제13 항에 있어서,

상기 스케줄링하는 단계는, 가용 프로세싱 유닛의 개수가 상기 복수의 논리 작업자의 개수보다 적다고 판단된 것에 기초하여, 상기 복수의 제1 마이크로서비스 및 상기 복수의 제2 마이크로서비스를 상기 어느 하나의 프로세싱 유닛에 순차적으로 스케줄링하는 단계를 포함하는, 트레이닝 시스템의 동작 방법.

청구항 15

제13 항에 있어서,

상기 스케줄링하는 단계는, 상기 복수의 제1 마이크로서비스와 상기 복수의 제2 마이크로서비스를 동일한 컨테이너에 스케줄링하는 단계를 포함하는, 트레이닝 시스템의 동작 방법.

청구항 16

제13 항에 있어서,

상기 어느 하나의 프로세싱 유닛에서 상기 복수의 제1 마이크로서비스 중 어느 하나의 미니 배치 처리와 상기 복수의 제2 마이크로서비스 중 어느 하나의 미니 배치 처리가 복수의 단계(multiple phases)로 실행되도록 스케줄링하는 단계를 더 포함하는, 트레이닝 시스템의 동작 방법.

청구항 17

제13 항에 있어서,

상기 신경망 모델에 대응하는 상기 트레이닝 잡에 상기 복수의 프로세싱 유닛을 2^n 단위로 할당하는 단계를 더 포함하는, 트레이닝 시스템의 동작 방법(n 은 0 또는 자연수).

청구항 18

제13 항에 있어서,

복수의 신경망 모델에 대응하는 복수의 트레이닝 잡 각각에 상기 복수의 프로세싱 유닛을 할당하는 단계를 더 포함하고,

상기 복수의 트레이닝 잡 각각에 상기 복수의 프로세싱 유닛을 할당하는 단계는,

최단 잔여 서비스 시간(shortest remaining service time)을 갖는 트레이닝 잡부터 순서대로 클러스터에 존재하는 프로세싱 유닛을 할당하는 단계; 및

상기 클러스터에 잔여 프로세싱 유닛(remaining processing unit)이 존재함에 따라, 필요한 프로세싱 유닛의 수보다 적은 수의 프로세싱 유닛이 할당된 트레이닝 잡에 상기 잔여 프로세싱 유닛을 할당하는 단계를 포함하는, 트레이닝 시스템의 동작 방법.

청구항 19

제13 항에 있어서,

복수의 신경망 모델에 대응하는 복수의 트레이닝 잡 각각에 상기 복수의 프로세싱 유닛을 할당하는 단계; 및

큐(queue)에 저장된 대기 트레이닝 잡(queuing training job)이 존재함에 따라, 대기 트레이닝 잡의 대기 시간(queuing time)이 하나 이상의 트레이닝 잡이 복수의 단계(multiple phases)로 실행됨에 따른 예상 증가 시간(expected increase time)보다 긴 경우, 상기 대기 트레이닝 잡에 상기 프로세싱 유닛을 재할당하는 단계를 더 포함하는, 트레이닝 시스템의 동작 방법.

청구항 20

제13 항에 있어서,

상기 복수의 마이크로서비스는, 상기 복수의 논리 작업자 중 제3 논리 작업자에 의해 실행되는 복수의 제3 마이크로서비스 및 상기 복수의 논리 작업자 중 제4 논리 작업자에 의해 실행되는 복수의 제4 마이크로서비스를 포함하되,

상기 스케줄링하는 단계는, 상기 복수의 제3 마이크로서비스 및 상기 복수의 제4 마이크로서비스를 상기 복수의 프로세싱 유닛 중 다른 하나의 프로세싱 유닛에 스케줄링하는 단계; 및

상기 복수의 제3 마이크로서비스는 상기 복수의 제1 마이크로서비스와 병렬적으로 실행되는 단계를 포함하는, 트레이닝 시스템의 동작 방법.

발명의 설명

기술 분야

[0001] 본 개시의 실시예들은 트레이닝 시스템 및 그 동작 방법에 관한 것으로, 보다 구체적으로는 마이크로서비스 기반의 트레이닝 시스템 및 그 동작 방법에 관한 것이다.

배경 기술

[0002] 최근 대용량 계산(HTC, High Throughput Computing)이 요구되는 애플리케이션 등이 다양한 분야로 확대되고 있고 그 규모가 커지고 있다.

[0003] 이에, 대용량 계산이 요구되는 애플리케이션을 효율적으로 실행하기 위해서 여러 사용자에게 의해 자원을 공유하는 공유 클러스터 환경이 대두된다. 특히, 하나의 심층 신경망(Deep Neural Network; DNN)을 트레이닝시키는 작업은 여러 GPU를 전용하여 사용하는 장기 실행 작업이 대부분이다. 그러나, 공유 클러스터 환경에서 사용자들은 한정된 자원을 나누어 각자의 작업을 실행하게 되며, 각자의 작업을 실행하기 위해 필요한 자원의 양이 제한되는 경우가 발생할 수 있다. 이에 따라, 여러 사용자에게 자원을 최적으로 할당하고, 스케줄링하는 자원 관리 정책에 대한 요구가 증가하고 있다.

[0004] 한편, 최근 들어 많이 활용되고 있는 아마존 람다(Amazon Lambda), 및 구글 클라우드 함수(Google Cloud function) 등과 같은 서버리스(serverless) 시스템에서, 대용량 계산이 함수 단위인 마이크로서비스(microservice)로 실행될 수 있다. 마이크로서비스 실행 방식은 하나의 애플리케이션(application)을 여러 개의 작은 애플리케이션으로 쪼개어 각각의 애플리케이션들을 작고 독립적인 서비스로 구성하는 것을 의미한다. 마이크로서비스 실행 방식은 독립적인 마이크로서비스로 실행되므로, 자원을 사용하는 데에 있어서 탄력성 등의 장점을 가질 수 있으나, 현재의 마이크로서비스 실행 방식은 일반적으로 모놀리식(monolithic)으로 실행되는 심

층 신경망의 트레이닝에 활용되기는 어렵다.

[0005] 이에 마이크로서비스 실행 방식을 사용하는 서버리스 시스템을 심층 신경망의 트레이닝에 활용한 효율적인 자원 관리 시스템에 대한 연구가 필요한 실정이다.

발명의 내용

해결하려는 과제

과제의 해결 수단

- [0006] 본 개시의 일 실시예에 따른 트레이닝 시스템은 신경망 모델에 대응하는 트레이닝 잡(training job)을 복수의 논리 작업자에 의해 실행되는 복수의 마이크로서비스(a plurality of microservices)로 분할하는 잡 프록시(job proxy), 및 상기 복수의 마이크로서비스를 복수의 프로세싱 유닛(a plurality of processing units)에 스케줄링하는 스케줄러(scheduler)를 포함하고, 상기 복수의 마이크로서비스는 상기 복수의 논리 작업자 중 제1 논리 작업자에 의해 실행되는 복수의 제1 마이크로서비스 및 상기 복수의 논리 작업자 중 제2 논리 작업자에 의해 실행되는 복수의 제2 마이크로서비스를 포함하고, 상기 스케줄러는, 상기 복수의 프로세싱 유닛의 가용 상태에 기초하여, 상기 복수의 제1 마이크로서비스 및 상기 복수의 제2 마이크로서비스를 상기 복수의 프로세싱 유닛 중 어느 하나의 프로세싱 유닛에 스케줄링한다.
- [0007] 본 개시의 일 실시예에 따른 스케줄러는, 가용 프로세싱 유닛(available processing unit)의 개수가 상기 복수의 논리 작업자의 개수보다 적음에 따라, 상기 복수의 제1 마이크로서비스 및 상기 복수의 제2 마이크로서비스를 상기 어느 하나의 프로세싱 유닛에 순차적으로 스케줄링할 수 있다.
- [0008] 본 개시의 일 실시예에 따른 스케줄러는, 상기 복수의 제1 마이크로서비스와 상기 복수의 제2 마이크로서비스를 동일한 컨테이너(container)에 스케줄링할 수 있다.
- [0009] 본 개시의 일 실시예에 따른 복수의 마이크로서비스는, 상기 트레이닝 잡에 대한 트레이닝 데이터가 분할된 복수의 미니 배치(minibatch)를 처리하는 함수를 포함하고, 상기 스케줄러는 상기 어느 하나의 프로세싱 유닛에서 상기 복수의 제1 마이크로서비스 중 어느 하나의 미니 배치 처리와 상기 복수의 제2 마이크로서비스 중 어느 하나의 미니 배치 처리가 복수의 단계(multiple phases)로 실행되도록 스케줄링할 수 있다.
- [0010] 본 개시의 일 실시예에 따른 트레이닝 시스템은 자원 관리자를 더 포함할 수 있다.
- [0011] 본 개시의 일 실시예에 따른 자원 관리자는, 상기 신경망 모델에 대응하는 상기 트레이닝 잡에 상기 복수의 프로세싱 유닛을 2^n 단위로 할당할 수 있다.
- [0012] 본 개시의 일 실시예에 따른 자원 관리자는, 상기 신경망 모델에 대응하는 상기 트레이닝 잡이 2k개의 논리 작업자를 포함하는 경우, 상기 복수의 프로세싱 유닛을 2k의 약수 중 어느 하나의 단위로 할당하고, 상기 트레이닝 잡이 2k-1개의 논리 작업자를 포함하는 경우, 상기 복수의 프로세싱 유닛을 2k-1의 약수 중 어느 하나 또는 2k의 약수 중 1과 2k를 제외한 어느 하나의 단위로 할당할 수 있다.
- [0013] 본 개시의 일 실시예에 따른 자원 관리자는, 복수의 신경망 모델에 대응하는 복수의 트레이닝 잡 각각에 상기 복수의 프로세싱 유닛을 할당할 수 있다.
- [0014] 본 개시의 일 실시예에 따른 자원 관리자는, 최단 잔여 서비스 시간(shortest remaining service time)을 갖는 트레이닝 잡부터 순서대로 클러스터에 존재하는 프로세싱 유닛을 할당하고, 상기 클러스터에 잔여 프로세싱 유닛(remaining processing unit)이 존재함에 따라, 필요한 프로세싱 유닛의 수보다 적은 수의 프로세싱 유닛이 할당된 트레이닝 잡에 상기 잔여 프로세싱 유닛을 할당할 수 있다.
- [0015] 본 개시의 일 실시예에 따른 자원 관리자는, 상기 복수의 트레이닝 잡 각각에 프로세싱 유닛을 할당하고, 큐(queue)에 저장된 대기 트레이닝 잡(queuing training job)이 존재함에 따라, 대기 트레이닝 잡의 대기 시간(queuing time)이 하나 이상의 트레이닝 잡이 복수의 단계(multiple phases)로 실행됨에 따른 예상 증가 시간(expected increase time)보다 긴 경우, 상기 대기 트레이닝 잡에 상기 프로세싱 유닛을 재할당할 수 있다.
- [0016] 본 개시의 일 실시예에 따른 복수의 마이크로서비스는, 복수의 미니 배치 각각에 대한 가중치를 계산하는 계산 함수, 및 상기 복수의 미니 배치 각각에 대한 가중치를 총합한 글로벌 파라미터를 계산하는 집계 함수를 포함할

수 있다.

- [0017] 본 개시의 일 실시예에 따른 복수의 제1 마이크로서비스의 제1 계산 함수와 상기 복수의 제2 마이크로서비스의 제2 계산 함수가 동일한 이터레이션에서 순차적으로 실행될 수 있다.
- [0018] 본 개시의 일 실시예에 따른 복수의 제1 마이크로서비스의 제1 계산 함수는, 상기 글로벌 파라미터(global parameter)를 읽고, 상기 글로벌 파라미터를 상기 복수의 제2 마이크로서비스의 제2 계산 함수에 전달할 수 있다.
- [0019] 본 개시의 일 실시예에 따른 복수의 마이크로서비스는, 상기 복수의 논리 작업자 중 제3 논리 작업자에 의해 실행되는 복수의 제3 마이크로서비스 및 상기 복수의 논리 작업자 중 제4 논리 작업자에 의해 실행되는 복수의 제4 마이크로서비스를 포함하고, 상기 스케줄러는, 상기 복수의 제3 마이크로서비스 및 상기 복수의 제4 마이크로서비스를 상기 복수의 프로세싱 유닛 중 다른 하나의 프로세싱 유닛에 스케줄링하고, 상기 복수의 제3 마이크로서비스는 상기 복수의 제1 마이크로서비스와 병렬적으로 실행될 수 있다.
- [0020] 본 개시의 일 실시예에 따른 트레이닝 시스템의 동작 방법은 신경망 모델에 대응하는 트레이닝 잡을 복수의 논리 작업자 각각에 의해 실행되는 복수의 마이크로서비스로 분할하는 단계, 및 상기 복수의 마이크로서비스를 복수의 프로세싱 유닛에 스케줄링하는 단계를 포함한다. 상기 스케줄링하는 단계는, 상기 복수의 프로세싱 유닛의 가용 상태에 기초하여, 상기 복수의 논리 작업자 중 제1 논리 작업자에 의해 실행되는 복수의 제1 마이크로서비스 및 상기 복수의 논리 작업자 중 제2 논리 작업자에 의해 실행되는 복수의 제2 마이크로서비스를 상기 복수의 프로세싱 유닛 중 어느 하나의 프로세싱 유닛에 스케줄링하는 단계를 포함한다.
- [0021] 본 개시의 일 실시예에 따른 스케줄링하는 단계는, 가용 프로세싱 유닛의 개수가 상기 복수의 논리 작업자의 개수보다 적다고 판단된 것에 기초하여, 상기 복수의 제1 마이크로서비스 및 상기 복수의 제2 마이크로서비스를 상기 어느 하나의 프로세싱 유닛에 순차적으로 스케줄링하는 단계를 포함할 수 있다.
- [0022] 본 개시의 일 실시예에 따른 스케줄링하는 단계는, 상기 복수의 제1 마이크로서비스와 상기 복수의 제2 마이크로서비스를 동일한 컨테이너에 스케줄링하는 단계를 포함할 수 있다.
- [0023] 본 개시의 일 실시예에 따른 트레이닝 시스템의 동작 방법은 상기 어느 하나의 프로세싱 유닛에서 상기 복수의 제1 마이크로서비스 중 어느 하나의 미니 배치 처리와 상기 복수의 제2 마이크로서비스 중 어느 하나의 미니 배치 처리가 복수의 단계(multiple phases)로 실행되도록 스케줄링하는 단계를 더 포함할 수 있다.
- [0024] 본 개시의 일 실시예에 따른 트레이닝 시스템의 동작 방법은 상기 신경망 모델에 대응하는 상기 트레이닝 잡에 상기 복수의 프로세싱 유닛을 2^n 단위로 할당하는 단계를 더 포함할 수 있다.
- [0025] 본 개시의 일 실시예에 따른 트레이닝 시스템의 동작 방법은 복수의 신경망 모델에 대응하는 복수의 트레이닝 잡 각각에 상기 복수의 프로세싱 유닛을 할당하는 단계를 더 포함할 수 있다.
- [0026] 본 개시의 일 실시예에 따른 복수의 트레이닝 잡 각각에 상기 복수의 프로세싱 유닛을 할당하는 단계는, 최단 잔여 서비스 시간(shortest remaining service time)을 갖는 트레이닝 잡부터 순서대로 클러스터에 존재하는 프로세싱 유닛을 할당하는 단계, 및 상기 클러스터에 잔여 프로세싱 유닛(remaining processing unit)이 존재함에 따라, 필요한 프로세싱 유닛의 수보다 적은 수의 프로세싱 유닛이 할당된 트레이닝 잡에 상기 잔여 프로세싱 유닛을 할당하는 단계를 포함할 수 있다.
- [0027] 본 개시의 일 실시예에 따른 트레이닝 시스템의 동작 방법은 복수의 신경망 모델에 대응하는 복수의 트레이닝 잡 각각에 상기 복수의 프로세싱 유닛을 할당하는 단계, 및 큐(queue)에 저장된 대기 트레이닝 잡(queueing training job)이 존재함에 따라, 대기 트레이닝 잡의 대기 시간(queueing time)이 하나 이상의 트레이닝 잡이 복수의 단계(multiple phases)로 실행됨에 따른 예상 증가 시간(expected increase time)보다 긴 경우, 상기 대기 트레이닝 잡에 상기 프로세싱 유닛을 재할당하는 단계를 더 포함할 수 있다.
- [0028] 본 개시의 일 실시예에 따른 복수의 마이크로서비스는, 상기 복수의 논리 작업자 중 제3 논리 작업자에 의해 실행되는 복수의 제3 마이크로서비스 및 상기 복수의 논리 작업자 중 제4 논리 작업자에 의해 실행되는 복수의 제4 마이크로서비스를 포함하되, 상기 스케줄링하는 단계는, 상기 복수의 제3 마이크로서비스 및 상기 복수의 제4 마이크로서비스를 상기 복수의 프로세싱 유닛 중 다른 하나의 프로세싱 유닛에 스케줄링하는 단계, 및 상기 복수의 제3 마이크로서비스는 상기 복수의 제1 마이크로서비스와 병렬적으로 실행되는 단계를 포함할 수 있다.

도면의 간단한 설명

- [0029] 도 1은 일 실시예에 따른 트레이닝 시스템의 동작 개념을 설명하기 위한 도면이다.
- 도 2는 일 실시예에 따른 트레이닝 시스템의 동작 방법을 나타낸 흐름도이다.
- 도 3은 일 실시예에 따른 트레이닝 시스템에서, 복수의 마이크로서비스와 복수의 논리 작업자를 설명하기 위한 도면이다.
- 도 4는 일 실시예에 따른 트레이닝 시스템에서, 트레이닝 잡에 프로세싱 유닛을 스케줄링하는 과정을 나타낸 예시 도면이다.
- 도 5는 일 실시예에 따른 트레이닝 시스템에서, 복수의 마이크로서비스를 실행하는 과정을 나타낸 도면이다.
- 도 6은 일 실시예에 따른 트레이닝 시스템에서, 프로세싱 유닛을 통해 복수의 마이크로서비스를 실행하는 과정을 나타낸 도면이다.
- 도 7 및 도 8은 일 실시예에 따른 트레이닝 시스템에서, 프로세싱 유닛에 스케줄링된 트레이닝 잡을 예시한 도면이다.
- 도 9는 일 실시예에 따른 트레이닝 시스템의 동작 방법을 나타낸 흐름도이다.
- 도 10은 일 실시예에 따른 트레이닝 시스템의 구성을 도시한 도면이다.
- 도 11은 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡에 프로세싱 유닛을 할당하는 과정을 나타낸 일 예이다.
- 도 12는 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡에 프로세싱 유닛을 할당하는 과정을 나타낸 일 예이다.
- 도 13은 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡에 프로세싱 유닛을 할당하는 동작 방법을 나타낸 흐름도이다.
- 도 14는 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡에 프로세싱 유닛을 할당하는 과정을 나타낸 일 예이다.
- 도 15는 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡에 프로세싱 유닛을 할당하는 과정을 나타낸 일 예이다.
- 도 16은 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡에 프로세싱 유닛을 할당하는 동작 방법을 나타낸 흐름도이다.
- 도 17은 일 실시예에 따른 트레이닝 시스템의 블록도이다.

발명을 실시하기 위한 구체적인 내용

- [0030] 본 명세서는 청구항의 권리범위를 명확히 하고, 본 개시의 실시예들이 속하는 기술분야에서 통상의 지식을 가진 자가 청구항에 기재된 실시예를 실시할 수 있도록, 실시예들의 원리를 설명하고 개시한다. 개시된 실시예들은 다양한 형태로 구현될 수 있다.
- [0031] 명세서 전체에 걸쳐 동일 참조 부호는 동일 구성요소를 지칭한다. 본 명세서가 실시예들의 모든 요소들을 설명하는 것은 아니며, 본 개시의 실시예들이 속하는 기술분야에서 일반적인 내용 또는 실시예들 간에 중복되는 내용은 생략한다. 명세서에서 사용되는 “모듈” 또는 “부”(unit)라는 용어는 소프트웨어, 하드웨어 또는 펌웨어 중 하나 또는 둘 이상의 조합으로 구현될 수 있으며, 실시예들에 따라 복수의 “모듈” 또는 “부”가 하나의 요소(element)로 구현되거나, 하나의 “모듈” 또는 “부”가 복수의 요소들을 포함하는 것도 가능하다.
- [0032] 실시예를 설명함에 있어서, 관련된 공지 기술에 대한 구체적인 설명이 본 개시의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다. 또한, 명세서의 설명 과정에서 이용되는 숫자(예를 들어, 제 1, 제2 등)는 하나의 구성요소를 다른 구성요소와 구분하기 위한 식별기호에 불과하다.
- [0033] 이하 첨부된 도면들을 참고하여 본 개시의 실시예들의 작용 원리 및 다양한 실시예들에 대해 설명한다.

- [0034] 도 1은 일 실시예에 따른 트레이닝 시스템의 동작 개념을 설명하기 위한 도면이다.
- [0035] 도 1을 참조하면, 트레이닝 시스템(100)은 신경망 모델에 대한 트레이닝을 수행할 수 있다. 일 실시예에 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(training job)(120)으로 명명될 수 있다.
- [0036] 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)를 포함할 수 있다. 그러나 본 개시의 일 실시예에서, 트레이닝 잡을 여러 마이크로서비스들의 형태로 실행하는데 있어 논리 작업자(Logical Worker)의 개념을 사용할 수 있다. 논리 작업자에 관한 상세한 설명은 도 2 및 도 3에서 후술한다.
- [0037] 본 개시의 일 실시예에서, 트레이닝 잡(120)은 복수의 마이크로서비스(a plurality of microservices)(M1, M2, M3, M4)로 구성될 수 있다. 본 개시에서, 마이크로서비스는 서버리스 시스템에서의 계산 단위인 하나의 함수(function)일 수 있고, 하나의 어플리케이션(application)은 마이크로서비스들의 사슬(chain of microservices)(또는, 마이크로서비스들의 시퀀스(sequence of microservices))로 실행될 수 있다. 마이크로서비스는 신경망 모델(110)의 트레이닝을 위한 트레이닝 데이터를 마이크로서비스 단위로 나눈 개념일 수 있다. 본 개시의 일 실시예에 따라, 트레이닝 데이터가 복수의 미니 배치(minibatch)를 포함하는 트레이닝 시스템(100)에서, 마이크로서비스는 미니 배치 처리 함수에 대응될 수 있다. 예를 들어, 마이크로서비스는 하나의 미니 배치에 대한 포워드 패스(forward pass) 및 백워드 패스(backward pass)를 처리할 수 있다. 다른 예를 들어, 마이크로서비스는 복수의 미니 배치를 순차적으로 처리하는, 복수의 미니 배치 처리의 사슬에 대응될 수 있다.
- [0038] 본 개시의 일 실시예에서, 마이크로서비스는 트레이닝 시스템(100)에서 하나의 처리 단위를 의미할 뿐, 그 처리의 대상은 실시예에 따라 달라질 수 있다.
- [0039] 본 개시의 일 실시예에 따른 트레이닝 시스템(100)에서, 거대 DNN 모델(110)의 트레이닝을 위한 트레이닝 잡(120)은 함수 단위의 작업인 복수의 마이크로서비스(M1, M2, M3, M4)의 사슬로 실행될 수 있다. 트레이닝 시스템(100)은 복수의 마이크로서비스(M1, M2, M3, M4)를 스케줄러(130)로 제출(submit)할 수 있다.
- [0040] 본 개시의 일 실시예에서, 스케줄러(130)는 복수의 마이크로서비스(M1, M2, M3, M4)를 스케줄러(130)의 큐(queue)(미도시)에 추가할 수 있다. 스케줄러(130)는 복수의 마이크로서비스(M1, M2, M3, M4)를 클러스터(140)에 스케줄링할 수 있다. 일 실시예에서, 스케줄러(130)는 복수의 마이크로서비스(M1, M2, M3, M4) 중 적어도 일부를 어느 하나의 프로세싱 유닛에 순차적으로 스케줄링할 수 있다. 스케줄러(130)는 복수의 마이크로서비스(M1, M2, M3, M4) 각각의 로컬리티(locality), 오버헤드(overhead), 간섭(interference) 등을 고려하여 복수의 프로세싱 유닛에 스케줄링할 수 있다.
- [0041] 본 개시의 일 실시예에서, 클러스터(140)는 트레이닝 잡(120)을 실행하여 거대 DNN 모델(110)을 트레이닝할 수 있다. 클러스터(140)는 복수의 노드를 고속 네트워크로 엮은 분산 처리 시스템으로서, 계산 능력과 메모리 용량이 다른 다양한 유형의 GPU들 또는 CPU로 구성된 이기종 자원을 포함하며, 병렬 컴퓨팅을 수행할 수 있다. 일 실시예에서, 클러스터(140)의 복수의 노드(예를 들어, 도 1의 Node 1, Node 2, Node 3) 각각은 GPU 또는 CPU에 대응되는 복수의 프로세싱 유닛을 포함하며, 복수의 노드 각각에 포함된 복수의 프로세싱 유닛은 트레이닝 잡(120)에 할당될 수 있다.
- [0042] 본 개시의 일 실시예에서, 트레이닝 시스템(100)은 서버리스(serverless) 시스템을 통해 복수의 마이크로서비스를 실행할 수 있다. 서버리스(serverless) 시스템은 개발자가 서버를 관리할 필요 없이 애플리케이션을 빌드(build)하고 실행할 수 있도록 하는 클라우드 개발 모델을 의미하며, 개발자는 배포를 위해 코드를 컨테이너(container)에 패키징할 수 있다. 본 개시의 일 실시예에서, 서버리스 시스템 기반의 마이크로서비스는, 독립적으로 실행가능하고, 스토리지를 통해 통신할 수 있다.
- [0043] 기존의 전용된(dedicated)된 GPU 사용 방식 및 all-or-nothing 방식(예컨대, 가용 자원이 필요한 자원보다 적은 경우 자원을 할당하지 않는 방식)의 스케줄링과 달리, 본 개시의 일 실시예에 따른 트레이닝 시스템(100)은 독립적으로 실행가능한 복수의 마이크로서비스(M1, M2, M3, M4)에 비전용(non-dedicated) GPU를 동적으로 스케줄링할 수 있다. 이에 따라, 트레이닝 시스템(100)은 트레이닝 잡(120)을 트레이닝하기 위해 자원을 탄력적으로 스케줄링할 수 있다. 이에 따라, 공유 클러스터 환경에서 한정된 자원을 동적으로 할당하고, 스케줄링함으로써 트레이닝 시스템(100)의 효율성을 향상시킬 수 있다.

- [0044] 일 실시예에서, 트레이닝 시스템(100)은 트레이닝 잡(120)이 실행되기 위해 필요한 자원보다 적은 수의 자원을 할당하고, 스케줄링할 수 있다. 트레이닝 시스템(100)은 필요한 자원보다 적은 수의 자원으로 트레이닝 잡(120)을 실행할 수 있다. 예를 들어, 트레이닝 잡(120)에 포함된 복수의 마이크로서비스(M1, M2, M3, M4)가 실행되기 위해서는 4개의 프로세싱 유닛이 필요할 수 있다. 예를 들어, 복수의 마이크로서비스(M1, M2, M3, M4)는 클러스터(140)에 속한 하나의 프로세싱 유닛에 스케줄링될 수 있다. 또는, 예를 들어, 복수의 마이크로서비스(M1, M2, M3, M4) 중 적어도 일부(예컨대, M1 및 M2)는 복수의 프로세싱 유닛 중 어느 하나의 프로세싱 유닛에 스케줄링되고, 복수의 마이크로서비스(M1, M2, M3, M4) 중 적어도 다른 일부(예컨대, M3 및 M4)는 복수의 프로세싱 유닛 중 다른 하나의 프로세싱 유닛에 스케줄링될 수 있다.
- [0045] 본 개시의 일 실시예에서, 트레이닝 잡(120)에 포함된 복수의 마이크로서비스(M1, M2, M3, M4)는 클러스터(140)에 의해 병렬적으로 트레이닝될 수 있다. 일 실시예에서, 트레이닝 시스템(100)은 데이터 병렬 처리를 통해 거대 DNN 모델(110)을 트레이닝함으로써, 클러스터의 이기종 GPU 자원의 활용도 뿐만 아니라 클러스터의 성능 또한 향상시킬 수 있다.
- [0046] 또한, 본 개시의 일 실시예에서, 트레이닝 잡(120)에 포함된 복수의 마이크로서비스(M1, M2, M3, M4) 중 적어도 일부는 클러스터(140)에 속한 하나의 프로세싱 유닛에 의해 순차적으로 실행될 수 있다. 이 경우, 복수의 마이크로서비스(M1, M2, M3, M4) 중 적어도 일부는, 하나의 프로세싱 유닛에서 복수의 단계(multi phases)로 실행될 수 있다. 예컨대, M1 및 M2는 어느 하나의 프로세싱 유닛에 의해 순차적으로 실행되고, M3 및 M4는 다른 하나의 프로세싱 유닛에 의해 순차적으로 실행될 수 있다.
- [0048] 도 2는 일 실시예에 따른 트레이닝 시스템의 동작 방법을 나타낸 흐름도이다.
- [0049] 도 2를 참조하면, 동작 S210에서, 일 실시예에 따른 트레이닝 시스템(100)은 신경망 모델에 대응하는 트레이닝 잡(120)을 복수의 논리 작업자(Logical Worker) 각각에 의해 실행되는 복수의 마이크로서비스로 분할한다. 예를 들어, 트레이닝 잡(120)은 트레이닝 시스템(100)을 통해 복수의 마이크로서비스(M1, M2, M3, M4)로 분할될 수 있다. 복수의 마이크로서비스(M1, M2, M3, M4) 각각은 복수의 논리 작업자 각각에 의해 실행될 수 있다.
- [0050] 본 개시에서, 트레이닝 잡은 복수의 마이크로서비스(M1, M2, M3, M4)로 분할되며, 복수의 마이크로서비스((M1, M2, M3, M4) 각각은 논리 작업자에 의해 마이크로서비스들의 사슬(chain of microservices)로 실행될 수 있다. 논리 작업자는, 실제로 존재하는 것이 아니라, 복수의 마이크로서비스((M1, M2, M3, M4) 각각을 마이크로서비스들의 사슬(chain of microservices)로 실행시키는 작업자일 수 있다. 논리 작업자는, 마이크로서비스들을 사슬 형태로 순차적으로 실행시키므로, GPU에 의한 통상의 작업자를 대체할 수 있다. 그러나, 논리 작업자는 통상의 작업자와는 다르게, 물리적으로 GPU와 디커플링(decoupling)되어 실행될 수 있다. 본 개시에서, 논리 작업자에 의해 마이크로서비스가 실행된다는 것은, 마이크로서비스들의 사슬을 처리하여 가중치(weight)를 산출함으로써, 거대 DNN 모델(110)에 대한 트레이닝을 수행하는 것을 의미할 수 있다.
- [0051] 본 개시에서 하나의 논리 작업자는 하나의 마이크로서비스들의 사슬(a chain of microservices)을 실행하므로, 논리 작업자의 수는 마이크로서비스의 사슬을 실행하여 트레이닝을 수행하기 위해 필요한 GPU 자원의 수에 대응될 수 있다. 즉, 논리 작업자의 수는 트레이닝 잡에 속한 마이크로서비스의 사슬들 각각을 병렬적으로 실행하기 위한 GPU 자원의 수에 대응될 수 있다.
- [0052] 동작 S220에서, 일 실시예에 따른 트레이닝 시스템(100)은 클러스터(140)에 포함된 복수의 프로세싱 유닛의 가용 상태에 기초하여, 복수의 논리 작업자 중 제1 논리 작업자에 의해 실행되는 복수의 제1 마이크로서비스 및 복수의 논리 작업자 중 제2 논리 작업자에 의해 실행되는 복수의 제2 마이크로서비스를 복수의 프로세싱 유닛 중 어느 하나의 프로세싱 유닛에 스케줄링한다. 이에 따라, 제1 마이크로서비스와 제2 마이크로서비스는 동일한 프로세싱 유닛을 통해 동일한 이터레이션(iteration)에서 순차적으로 실행될 수 있다. 즉, 하나의 프로세싱 유닛은 두개 이상의 마이크로서비스를 순차적으로 실행될 수 있고, 1회의 이터레이션(iteration)은 두개 이상의 마이크로서비스가 하나의 프로세싱 유닛에 의해 순차적으로 실행되는 복수의 단계(multiple phases)로 수행될 수 있다. 본 개시에서, 프로세싱 유닛의 가용 상태는, 트레이닝 잡(120)에 할당된(allocated) 프로세싱 유닛의 존재 여부 또는 개수를 의미할 수 있다.
- [0053] 예를 들어, 트레이닝 시스템(100)은 가용 프로세싱 유닛(available processing unit)의 개수가 논리 작업자의 개수보다 적다고 판단된 것에 기초하여, 복수의 제1 마이크로서비스 및 복수의 제2 마이크로서비스를 어느 하나의 프로세싱 유닛에 순차적으로 스케줄링할 수 있다. 본 개시에서, 가용 프로세싱 유닛은 가용 상태에 해당하는

프로세싱 유닛을 의미할 수 있다.

- [0054] 복수의 마이크로서비스는, 트레이닝 데이터를 분할한 미니 배치에 대응될 수 있다. 이에 따라, 복수의 제1 마이크로서비스 중 어느 하나의 미니 배치 처리와 복수의 제2 마이크로서비스 중 어느 하나의 미니 배치 처리는 어느 하나의 프로세싱 유닛에서 순차적으로 수행될 수 있다. 즉, 복수의 제1 마이크로서비스 중 어느 하나의 미니 배치 처리와 제2 마이크로서비스 중 어느 하나의 미니 배치 처리는 동일한 이터레이션에 속할 수 있다(도 5 참조).
- [0055] 본 개시의 일 실시예에 따른 트레이닝 시스템(100)은 여러 개의 논리 작업자를 동일한 프로세싱 유닛에 스케줄링할 수 있다. 이 경우, 복수의 마이크로서비스(M1, M2, M3, M4)는 동일한 프로세싱 유닛에서 순차적으로 실행될 수 있다.
- [0056] 본 개시의 일 실시예에 따른 트레이닝 시스템(100)은 일반적인 트레이닝 시스템과 마찬가지로 여러 개의 논리 작업자를 병렬로 스케줄링할 수도 있다. 이 경우, 복수의 마이크로서비스(M1, M2, M3, M4)는 복수의 논리 작업자에 의해 병렬적으로 실행될 수 있다.
- [0058] 도 3은 일 실시예에 따른 트레이닝 시스템에서, 복수의 마이크로서비스와 복수의 논리 작업자를 설명하기 위한 도면이다.
- [0059] 도 3을 참조하면, 트레이닝 잡(120)은 복수의 제1 마이크로서비스(M1), 복수의 제2 마이크로서비스(M2), 복수의 제3 마이크로서비스(M3), 및 복수의 제4 마이크로서비스(M4)를 포함할 수 있다. 일 실시예에 따라, 복수의 제1 마이크로서비스(M1), 복수의 제2 마이크로서비스(M2), 복수의 제3 마이크로서비스(M3), 및 복수의 제4 마이크로서비스(M4) 각각은 미니 배치별로 분할될 수 있다. 즉, 복수의 제1 마이크로서비스(M1), 복수의 제2 마이크로서비스(M2), 복수의 제3 마이크로서비스(M3), 및 복수의 제4 마이크로서비스(M4) 각각은 미니 배치별로 분할된 마이크로서비스들의 사슬(chain of microservices) 형태로 존재할 수 있다.
- [0060] 복수의 제1 마이크로서비스(M1), 복수의 제2 마이크로서비스(M2), 복수의 제3 마이크로서비스(M3), 및 복수의 제4 마이크로서비스(M4)는 복수의 논리 작업자(Logical Worker 1, Logical Worker 2, Logical Worker 3, Logical Worker 4)(이하, LW1, LW2, LW3, LW4 라고 함.)에 의해 병렬적으로 실행될 수 있다. 복수의 논리 작업자(LW1, LW2, LW3, LW4)가 복수의 마이크로서비스(M1, M2, M3, M4)를 실행하는 것은, 트레이닝 잡(120)을 실행시켜 거대 DNN 모델(110)에 대한 트레이닝을 수행하는 것을 의미할 수 있다. 이하 설명의 편의를 위해 마이크로서비스들의 사슬의 수가 4개이고, 논리 작업자의 수가 4개인 것으로 예시하였지만, 본 개시의 실시예는 이에 한정되지 않는다.
- [0061] 일 실시예에 따라, 트레이닝 시스템(100)은 복수의 논리 작업자(LW1, LW2, LW3, LW4)를 통해 전용된(dedicated) GPU의 할당 없이, 마이크로서비스의 형태로 트레이닝 잡(120)을 실행할 수 있다. 예를 들어, 제1 내지 제4 논리 작업자(LW1, LW2, LW3, LW4) 각각은 복수의 제1 내지 제4 마이크로서비스(M1, M2, M3, M4) 각각을 실행할 수 있다.
- [0062] 예를 들어, 복수의 제1 마이크로서비스(M1)는 제1 논리 작업자(LW1)에 의해 복수의 제1 마이크로서비스(M1)의 사슬로 실행될 수 있다. 복수의 제1 마이크로서비스(M1)는 제1 계산 함수(C: C1-1, C1-2, C1-3) 및 제1 집계 함수(Agg: Agg1-1, Agg1-2, Agg1-3)를 포함할 수 있다. 제1 논리 작업자(LW1)는 제1 계산 함수(C1-1, C1-2, C1-3)와 제1 집계 함수(Agg1-1, Agg1-2, Agg1-3)의 사슬로 이루어진 복수의 제1 마이크로서비스(M1)를 순차적으로 실행할 수 있다.
- [0063] 예를 들어, 복수의 제2 마이크로서비스(M2)는 제2 논리 작업자(LW2)에 의해 복수의 제2 마이크로서비스(M2)의 사슬로 실행될 수 있다. 복수의 제2 마이크로서비스(M2)는 제2 계산 함수(C: C2-1, C2-2, C2-3) 및 제2 집계 함수(Agg: Agg2-1, Agg2-2, Agg2-3)를 포함할 수 있다. 제2 논리 작업자(LW2)는 제2 계산 함수(C2-1, C2-2, C2-3)와 제2 집계 함수(Agg2-1, Agg2-2, Agg2-3)의 사슬로 이루어진 복수의 제2 마이크로서비스(M2)를 순차적으로 실행할 수 있다.
- [0064] 예를 들어, 복수의 제3 마이크로서비스(M3)는 제3 논리 작업자(LW3)에 의해 복수의 제3 마이크로서비스(M3)의 사슬로 실행될 수 있다. 복수의 제3 마이크로서비스(M3)는 제3 계산 함수(C: C3-1, C3-2, C3-3) 및 제3 집계 함수(Agg: Agg3-1, Agg3-2, Agg3-3)를 포함할 수 있다. 제3 논리 작업자(LW3)는 제3 계산 함수(C3-1, C3-2, C3-3)와 제3 집계 함수(Agg3-1, Agg3-2, Agg3-3)의 사슬로 이루어진 복수의 제3 마이크로서비스(M3)를 순차적으로

실행할 수 있다.

- [0065] 예를 들어, 복수의 제4 마이크로서비스(M4)는 제4 논리 작업자(LW4)에 의해 복수의 제4 마이크로서비스(M4)의 사슬로 실행될 수 있다. 복수의 제4 마이크로서비스(M4)는 제4 계산 함수(C: C4-1, C4-2, C4-3) 및 제4 집계 함수(Agg: Agg4-1, Agg4-2, Agg4-3)를 포함할 수 있다. 제4 논리 작업자(LW4)는 제4 계산 함수(C4-1, C4-2, C4-3)와 제4 집계 함수(Agg4-1, Agg4-2, Agg4-3)의 사슬로 이루어진 복수의 제4 마이크로서비스(M4)를 순차적으로 실행할 수 있다.
- [0066] 복수의 논리 작업자(LW1, LW2, LW3, LW4)는 마이크로서비스들을 기반으로 트레이닝 잡(120)을 분할하여 병렬적으로 실행할 수 있다. 이로써, 4개의 GPU를 통해 트레이닝 잡(120)을 실행하는 것에 상응하는 효율성을 나타낼 수 있다. 본 개시의 실시예에 따라, 마이크로서비스는 계산 함수(Computation function)(C)와 집계 함수(Aggregation function)(Agg)를 포함할 수 있다. 본 개시에서, 계산 함수(C)는 제1 유형의 마이크로서비스로 지칭되고, 집계 함수(Agg)는 제2 유형의 마이크로서비스로 지칭될 수 있다. 한편, 도면에서 집계 함수와 계산 함수 사이에 도시된 화살표는 각 함수들 간의 데이터 통신(또는, 데이터 디펜던시)을 나타낸다. 본 개시에서, 각 함수들은 직접 통신이 아닌, 스토리지(예컨대, 분산된 인-메모리 데이터베이스(도 10의 260))를 통한 간접 통신을 수행할 수 있다.
- [0067] 계산 함수(C)는 복수의 논리 작업자(LW1, LW2, LW3, LW4)가 모델 파라미터, 다시 말해, 가중치(weight)를 산출하는 작업일 수 있다. 여기서, '모델 파라미터(model parameter)'는 예를 들어, 신경망의 모델 파라미터, 또는 그라디언트(gradient) 등과 같이 심층 신경망의 각 계층들에서 사용되는 값에 해당할 수 있다. 계산 함수(C)는 포워드 패스(forward pass) 및 백워드 패스(backward pass)를 오가는 반복 동작들을 통해 수행될 수 있으며, 이러한 반복 동작들에 의해 많은 파라미터들이 생성될 수 있다. 여기서, 반복 동작은 '이터레이션(iteration)'으로 지칭되며, 최적화된 트레이닝을 위해 많은 양의 이터레이션이 수행될 수 있다. 계산 함수(C)는 이터레이션마다 미니 배치를 처리하여 가중치를 산출할 수 있다. 산출된 가중치는 동기화를 위해 인-메모리 데이터베이스(도 10의 260)에 업데이트(update)될 수 있다. 이하, '매개 변수'는 '가중치'와 동일한 의미로 이해될 수 있다.
- [0068] 예를 들어, 복수의 제1 마이크로서비스(M1)의 제1 계산 함수(C1-1, C1-2, C1-3)는 제1 이터레이션(iteration 1)을 수행하는 제1 계산 함수(C1-1), 제2 이터레이션(iteration 2)을 수행하는 제1 계산 함수(C1-2), 및 제3 이터레이션(iteration 3)을 수행하는 제1 계산 함수(C1-3)를 포함할 수 있다.
- [0069] 또한, 예를 들어, 복수의 제2 마이크로서비스(M2)의 제2 계산 함수(C2-1, C2-2, C2-3)는 제1 이터레이션(iteration 1)을 수행하는 제2 계산 함수(C2-1), 제2 이터레이션(iteration 2)을 수행하는 제2 계산 함수(C2-2), 제3 이터레이션(iteration 3)을 수행하는 제2 계산 함수(C2-3)를 포함할 수 있다. 복수의 제3 마이크로서비스(M3)의 제3 계산 함수(C3-1, C3-2, C3-3)는 제1 이터레이션(iteration 1)을 수행하는 제3 계산 함수(C3-1), 제2 이터레이션(iteration 2)을 수행하는 제3 계산 함수(C3-2), 제3 이터레이션(iteration 3)을 수행하는 제3 계산 함수(C3-3)를 포함할 수 있다. 복수의 제4 마이크로서비스(M4)의 제4 계산 함수(C4-1, C4-2, C4-3)는 제1 이터레이션(iteration 1)을 수행하는 제4 계산 함수(C4-1), 제2 이터레이션(iteration 2)을 수행하는 제4 계산 함수(C4-2), 제3 이터레이션(iteration 3)을 수행하는 제4 계산 함수(C4-3)를 포함할 수 있다.
- [0070] 집계 함수(Agg)는 복수의 논리 작업자(LW1, LW2, LW3, LW4) 각각이 산출한 가중치를 총합(aggregate)하는 작업일 수 있다. 집계 함수(Agg)는 CPU 자원만을 사용하여 수행될 수도 있다. 집계 함수(Agg)는 CPU 자원을 사용함에 따라, 복수의 집계 함수(예컨대, Agg1-1, Agg2-1, Agg3-1, Agg4-1)가 동시에 실행될 수 있다. 집계 함수(Agg)는 계산 함수(C)의 처리에 의해 산출된 가중치를 총합하여, 가중치를 동기화할 수 있다. 동기화된 가중치는, 글로벌 파라미터(global parameter)일 수 있다. 예를 들어, 집계 함수(Agg)는 인-메모리 데이터베이스(260)에 글로벌 파라미터를 업데이트할 수 있다. 글로벌 파라미터는 집계 함수(Agg)에 의해 그 다음 이터레이션을 수행하는 계산 함수(C)에 업데이트될 수 있다.
- [0071] 그 다음 이터레이션을 수행하는 계산 함수(C)는 집계 함수(Agg)에 의해 업데이트된 글로벌 파라미터를 읽고(read), 이터레이션마다 미니 배치를 처리하여 가중치를 산출할 수 있다.
- [0072] 예를 들어, 복수의 제1 마이크로서비스(M1)는 제1 집계 함수(Agg1-1, Agg1-2, Agg1-3)를 포함하고, 복수의 제2 마이크로서비스(M2)는 제2 집계 함수(Agg2-1, Agg2-2, Agg2-3)를 포함하고, 복수의 제3 마이크로서비스(M3)는 제3 집계 함수(Agg3-1, Agg3-2, Agg3-3)를 포함하고, 복수의 제4 마이크로서비스(M4)는 제4 집계 함수(Agg4-1, Agg4-2, Agg4-3)를 포함할 수 있다.
- [0073] 예를 들어, 제1 이터레이션(iteration 1)에서 복수의 제1 마이크로서비스(M1) 중 제1 집계 함수(Agg1-1), 복수

의 제2 마이크로서비스(M2) 중 제2 집계 함수(Agg2-1), 복수의 제3 마이크로서비스(M3) 중 제3 집계 함수(Agg3-1), 및 복수의 제4 마이크로서비스(M4) 중 제4 집계 함수(Agg4-1)는 제1 계산 함수(C1-1)로부터 산출된 가중치, 제2 계산 함수(C2-1)로부터 산출된 가중치, 제3 계산 함수(C3-1)로부터 산출된 가중치, 및 제4 계산 함수(C4-1)로부터 산출된 가중치를 총합하여, 글로벌 파라미터를 업데이트할 수 있다. 업데이트된 글로벌 파라미터는 제1 집계 함수(Agg1-1), 제2 집계 함수(Agg2-1), 제3 집계 함수(Agg3-1), 및 제4 집계 함수(Agg4-1)에 의해 그 다음 이터레이션(예를 들어, iteration 2)을 수행하는 제1 계산 함수(C1-2), 제2 계산 함수(C2-2), 제3 계산 함수(C3-2), 및 제4 계산 함수(C4-2)에 업데이트될 수 있다. 그 다음 이터레이션(iteration 2)을 수행하는 제1 계산 함수(C1-2), 제2 계산 함수(C2-2), 제3 계산 함수(C3-2), 및 제4 계산 함수(C4-2) 각각은 업데이트된 글로벌 파라미터를 읽고, 새로운 가중치를 산출할 수 있다.

[0074] 상술한 바와 마찬가지로, 집계 함수(Agg1-1, Agg2-1, Agg3-1, Agg4-1)의 업데이트를 기초로, 그 다음 이터레이션(iteration 2)에서 제1 계산 함수(C1-2), 제2 계산 함수(C2-2), 제3 계산 함수(C3-2), 및 제4 계산 함수(C4-2) 각각은 가중치를 산출할 수 있다. 산출된 가중치는 집계 함수(Agg1-2, Agg2-2, Agg3-2, Agg4-2)에 의해 그 다음 이터레이션(예를 들어, iteration 3)의 계산 함수(C1-3, C2-3, C3-3, C4-3)에 업데이트될 수 있다.

[0075] 본 개시의 일 실시예에서, 네 개의 논리 작업자는 데이터 병렬성을 이용하여 트레이닝 잡을 훈련하고, BSP를 통해 가중치를 동기화할 수 있다. 이에 관한 설명은 도 10에서 후술한다.

[0077] 도 4는 일 실시예에 따른 트레이닝 시스템에서, 트레이닝 잡에 프로세싱 유닛을 스케줄링하는 과정을 나타낸 예시 도면이다.

[0078] 도 4를 참조하면, 일 실시예에 따른 트레이닝 시스템(100)은 클러스터(140)에 포함된 복수의 프로세싱 유닛의 가용 상태에 기초하여, 복수의 논리 작업자를 복수의 프로세싱 유닛에 스케줄링할 수 있다.

[0079] 여러 사용자에게 의해 자원을 공유하는 공유 클러스터 환경에서, 사용자들은 한정된 자원을 나누어 각자의 작업을 실행하게 된다. 이에 따라, 트레이닝 잡을 실행하기 위해 필요한 논리 작업자의 수보다 가용 상태의 프로세싱 유닛의 수가 적을 수 있다.

[0080] 본 개시의 일 실시예에서, 트레이닝 잡은 서버리스 시스템에서 복수의 마이크로서비스로 분할되며, 복수의 마이크로서비스는 서로 독립적으로 실행될 수 있다. 복수의 마이크로서비스는 서로 독립적으로 실행될 수 있으므로, 복수의 마이크로서비스를 실행하기 위해 필요한 논리 작업자의 수보다 가용 상태의 프로세싱 유닛의 수가 적더라도 동적인 자원 스케줄링을 통해 트레이닝을 수행할 수 있다.

[0081] 예를 들어, 클러스터(140)는 복수의 프로세싱 유닛으로 제1 프로세싱 유닛(141) 및 제2 프로세싱 유닛(142)을 포함할 수 있다. 예를 들어, 제1 프로세싱 유닛(141) 및 제2 프로세싱 유닛(142)은 자원 관리자(도 7의 150)에 의해 트레이닝 잡(120)에 할당될 수 있다. 일 실시예에 따른 자원 관리자(150)는 클러스터(140)에 제출된 트레이닝 잡에 하나 이상의 프로세싱 유닛을 할당할 수 있다. 일 실시예에 따른 자원 관리자(150)는 트레이닝 잡(120)을 실행하기 위해 필요한 논리 작업자의 수보다 적은 수의 프로세싱 유닛을 할당할 수 있다. . 예를 들어, 자원 관리자(150)는 복수의 마이크로서비스(M1, M2, M3, M4)를 포함하는 트레이닝 잡(120)에 제1 프로세싱 유닛(141) 및 제2 프로세싱 유닛(142)을 할당할 수 있다. 제1 프로세싱 유닛(141) 및 제2 프로세싱 유닛(142)은 트레이닝 잡(120)에 할당된 자원이며, 트레이닝 잡(120)에 대한 가용 프로세싱 유닛일 수 있다.

[0082] 본 개시의 일 실시예에서, 스케줄러(130)는 복수의 마이크로서비스(M1, M2, M3, M4)를 두개의 프로세싱 유닛에 동적으로 스케줄링할 수 있다. 스케줄러(130)는 트레이닝 잡(120)에 할당된 프로세싱 유닛의 개수가 필요한 자원의 개수보다 적더라도, 프로세싱 유닛을 트레이닝 잡(120)에 스케줄링할 수 있다. 예를 들어, 복수의 마이크로서비스(M1, M2, M3, M4) 중 복수의 제1 마이크로서비스(M1) 및 복수의 제2 마이크로서비스(M2)에 제1 프로세싱 유닛(141)을 스케줄링하고, 복수의 마이크로서비스(M1, M2, M3, M4) 중 복수의 제3 마이크로서비스(M3) 및 복수의 제4 마이크로서비스(M4)에 제2 프로세싱 유닛(142)을 스케줄링할 수 있다.

[0083] 예를 들어, 복수의 제1 마이크로서비스(M1)와 복수의 제2 마이크로서비스(M2)는 동일한 제1 프로세싱 유닛(141)을 통해 동일한 이터레이션(iteration) 동안 순차적(sequentially)으로 실행될 수 있다. 즉, 제1 프로세싱 유닛(141)은 두 개의 마이크로서비스를 순차적으로 실행하며, 1회의 이터레이션(iteration)은 복수의 단계(multiple phases)로 수행될 수 있다. 예를 들어, 복수의 제3 마이크로서비스(M3)와 복수의 제4 마이크로서비스(M4)는 동일한 제2 프로세싱 유닛(142)을 통해 동일한 이터레이션(iteration) 동안 순차적(sequentially)으로 실행될 수 있다. 이 경우, 제1 프로세싱 유닛(141)에 의해 실행되는 복수의 제1 마이크로서비스(M1)와 제2 프로

세싱 유닛(142)에 의해 실행되는 복수의 제3 마이크로서비스(M3)는 병렬적으로 실행될 수 있다.

[0084] 스케줄러(130)는 클러스터(140)의 노드에 마이크로서비스를 실행할 컨테이너(container)를 스케줄링할 수 있다. 스케줄러(130)는 컨테이너의 콜드 스타트(cold-start)에 의한 오버헤드(overhead)를 줄이기 위해, 동일한 컨테이너(container)를 사용하여 트레이닝을 수행하도록 마이크로서비스를 스케줄링할 수 있다. 컨테이너는 동일한 트레이닝 잡에 대해 계속해서 재사용될 수 있으므로, 워م 컨테이너(warm-container)로 사용될 수 있다. 예를 들어, 제1 마이크로서비스(M1)와 제2 마이크로서비스(M2)는 동일한 컨테이너에서 순차적으로 실행되고, 제3 마이크로서비스(M3)와 제4 마이크로서비스(M4)는 동일한 컨테이너에서 순차적으로 실행될 수 있다.

[0086] 도 5는 일 실시예에 따른 트레이닝 시스템에서, 복수의 마이크로서비스를 실행하는 과정을 나타낸 도면이다.

[0087] 도 5를 참조하여, 4개의 작업자가 필요한 트레이닝 잡에 2개의 프로세싱 유닛이 할당된 경우, 트레이닝 잡을 실행하는 과정을 설명한다.

[0088] 본 개시의 일 실시예에서, 4개의 논리 작업자를 사용하는 트레이닝 잡은 제1 프로세싱 유닛(141)과 제2 프로세싱 유닛(142)에 의해 실행될 수 있다. 하나의 프로세싱 유닛은 2개의 논리 작업자에 의해 실행되는 복수의 마이크로서비스를 순차적으로 실행할 수 있다. 이에 따라, 동일한 프로세싱 유닛에서 복수의 마이크로서비스는 1회의 이터레이션 동안 2단계(2 phase)로 실행될 수 있다. 예를 들어, 제1 프로세싱 유닛(141)은 제1 논리 작업자(LW1)에 대응하는 제1 마이크로서비스(M1)와 제2 논리 작업자(LW2)에 대응하는 제2 마이크로서비스(M2)를 순차적으로 실행하여, 제1 마이크로서비스(M1)와 제2 마이크로서비스(M2)는 1회의 이터레이션 동안 2단계(2 phase)로 수행될 수 있다. 예를 들어, 제2 프로세싱 유닛(142)은 제3 논리 작업자(LW3)에 대응하는 제3 마이크로서비스(M3)와 제4 논리 작업자(LW4)에 대응하는 제4 마이크로서비스(M4)를 순차적으로 실행하여, 제3 마이크로서비스(M3)와 제4 마이크로서비스(M4)는 1회의 이터레이션 동안 2단계(2 phase)로 수행될 수 있다.

[0089] 구체적으로, 복수의 프로세싱 유닛(141, 142) 각각은 마이크로서비스 중 미니 배치를 처리하는 2개의 계산 함수(C)를 1회의 이터레이션 동안 순차적으로 실행할 수 있다. 예를 들어, 제1 논리 작업자(LW1)에 연관된 제1 계산 함수(C1-1)와 제2 논리 작업자(LW2)에 연관된 제2 계산 함수(C2-1)는 제1 이터레이션(iteration 1) 동안 제1 프로세싱 유닛(141)을 통해 순차적으로 실행될 수 있다. 예를 들어, 제3 논리 작업자(LW3)에 연관된 제3 계산 함수(C3-1)와 제4 논리 작업자(LW4)에 연관된 제4 계산 함수(C4-1)는 제1 이터레이션(iteration 1) 동안 제2 프로세싱 유닛(142)을 통해 순차적으로 실행될 수 있다. 본 개시의 일 실시예에서, 제1 계산 함수(C1-1)와 제2 계산 함수(C2-1)는 제1 프로세싱 유닛(141)의 동일한 컨테이너에서 실행되고, 제3 계산 함수(C3-1)와 제4 계산 함수(C4-1)는 제2 프로세싱 유닛(142)의 동일한 컨테이너에서 실행될 수 있으나, 이에 제한되지 않는다.

[0090] 마찬가지로, 제1 계산 함수(C1-2)와 제2 계산 함수(C2-2)는 제2 이터레이션(iteration 2) 동안 제1 프로세싱 유닛(141)을 통해 순차적으로 실행되고, 제3 계산 함수(C3-2)와 제4 계산 함수(C4-2)는 제2 프로세싱 유닛(142)을 통해 순차적으로 실행될 수 있다. 제1 계산 함수(C1-3)와 제2 계산 함수(C2-3)는 제3 이터레이션(iteration 3) 동안 제1 프로세싱 유닛(141)을 통해 순차적으로 실행되고, 제3 계산 함수(C3-3)와 제4 계산 함수(C4-3)는 제2 프로세싱 유닛(142)을 통해 순차적으로 실행될 수 있다.

[0091] 또한, 일 실시예에서 제1 프로세싱 유닛(141)에 의해 실행된 제1 논리 작업자(LW1)의 제1 계산 함수(C1-1)와 제2 프로세싱 유닛(142)에 의해 실행된 제3 논리 작업자(LW3)의 제3 계산 함수(C3-1)는 서로 병렬적으로 수행될 수 있다.

[0092] 본 개시의 일 실시예에서, 계산 함수(C)는 이터레이션마다 가중치를 산출하고, 산출된 가중치는 간접 통신을 통해 분산된 인-메모리 데이터베이스(도 10의 260)에 업데이트될 수 있다. 도 3에서 설명한 바와 같이, 하나의 이터레이션(예를 들어, iteration 1)에서, 제1 집계 함수(Agg1-1), 제2 집계 함수(Agg2-1), 제3 집계 함수(Agg3-1), 및 제4 집계 함수(Agg4-1) 각각은 제1 계산 함수(C1-1)로부터 산출된 가중치, 제2 계산 함수(C2-1)로부터 산출된 가중치, 제3 계산 함수(C3-1)로부터 산출된 가중치, 및 제4 계산 함수(C4-1)로부터 산출된 가중치를 총합하여 가중치를 동기화할 수 있다. 예를 들어, 제1 집계 함수(Agg1-1)는 제1 계산 함수(C1-1)로부터 산출된 가중치, 제2 계산 함수(C2-1)로부터 산출된 가중치, 제3 계산 함수(C3-1)로부터 산출된 가중치, 및 제4 계산 함수(C4-1)로부터 산출된 가중치를 총합하여, 글로벌 파라미터를 업데이트할 수 있다. 그 다음 이터레이션(예를 들어, iteration 2)을 수행하는 제1 계산 함수(C1-2), 제2 계산 함수(C2-2), 제3 계산 함수(C3-2), 및 제4 계산 함수(C4-2)는 집계 함수(Agg1-1, Agg2-1, Agg3-1, Agg4-1)에 의해 업데이트된 글로벌 파라미터를 읽고, 새로운 가중치를 산출할 수 있다.

- [0093] 서버리스 환경에서, 계산 함수(C)와 집계 함수(Agg)는 스토리지(예를 들어, 도 10의 분산된 인-메모리 데이터베이스(260))를 통해 간접적으로 통신하고, 스토리지는 간접 통신을 통해 가중치를 동기화할 수 있다. 집계 함수(Agg)는 CPU 자원만을 사용하여 실행될 수 있다. 이에 제한되는 것은 아니나, 계산 함수(C)를 위한 컨테이너와 짝을 이루는 집계 함수(Agg)는 계산 함수(C)와 동일한 노드에서 스케줄링될 수 있다. 본 개시의 일 실시예에 따른 트레이닝 시스템은 모델 동기화를 수행하는 AllReduce 통신과 유사할 수 있으나, 이에 제한되지 않는다.
- [0095] 도 6은 일 실시예에 따른 트레이닝 시스템에서, 프로세싱 유닛을 통해 복수의 마이크로서비스를 실행하는 과정을 나타낸 도면이다.
- [0096] 도 6에는 4개의 작업자가 필요한 트레이닝 잡에 2개의 프로세싱 유닛이 할당되고, 동일한 프로세싱 유닛에서 복수의 마이크로서비스가 각 이터레이션 동안 2단계(2 phase)로 실행되는 경우를 예시하고 있다.
- [0097] 도 6을 참조하면, 본 개시의 일 실시예에 따른 트레이닝 시스템(100)에서, 가중치를 계산하는 계산 함수(C)와 계산된 가중치를 총합하여 글로벌 파라미터를 동기화하는 집계 함수(Agg)가 동시에 실행될 수 있다. 또한, 본 개시의 일 실시예에 따른 트레이닝 시스템(100)에서, 하나의 프로세싱 유닛은 각 이터레이션 마다 복수의 단계(multiple phases)를 실행할 수 있는데, 복수의 단계 중 첫번째 단계의 계산 함수만 글로벌 파라미터를 읽고, 나머지 단계의 계산 함수들은 첫번째 단계의 계산 함수가 읽은 글로벌 파라미터를 사용할 수 있다. 또한, 본 개시의 일 실시예에 따른 트레이닝 시스템에서, 각 이터레이션마다 마지막 단계의 계산 함수만 이전 단계의 계산 함수들에 의해 계산된 가중치를 총합하여, 분산된 인-메모리 데이터베이스에 업데이트할 수 있다. 집계 함수는 분산된 인-메모리 데이터베이스에 업데이트된 가중치의 사본(copy)을 읽을 수 있다. 본 개시의 일 실시예에 따른 트레이닝 시스템(100)은, 계산 함수(C)와 집계 함수(Agg) 간의 통신 오버헤드를 최소화할 수 있다.
- [0098] 이하, 도 5의 일 실시예에 따른 트레이닝 시스템과의 차이점을 중심으로 설명하고, 중복된 내용은 생략한다.
- [0099] 본 개시의 일 실시예에서, 계산 함수(C)와 집계 함수(Agg)는 동시에 실행될 수 있다. 도 5에서, 계산 함수(C)가 하나의 이터레이션 동안 실행된 후에 집계 함수(Agg)가 실행된 것과 달리, 계산 함수(C)의 실행과 집계 함수(Agg)의 실행이 오버랩(overlap)될 수 있다.
- [0100] 예를 들어, 제1 계산 함수(C1-1)와 제2 계산 함수(C2-1)가 제1 프로세싱 유닛(141)을 통해 가중치를 계산하는 동안, 제1 집계 함수(Agg1-1), 제2 집계 함수(Agg2-1), 제3 집계 함수(Agg3-1), 및 제4 집계 함수(Agg4-1)는 미리 실행되어, 각 계산 함수(C)의 가중치 계산이 완료되는 즉시, 가중치를 총합할 수 있다. 또한, 예를 들어, 제3 계산 함수(C3-1)와 제4 계산 함수(C4-1)가 제2 프로세싱 유닛(142)을 통해 가중치를 계산하는 동안, 제1 집계 함수(Agg1-1), 제2 집계 함수(Agg2-1), 제3 집계 함수(Agg3-1), 및 제4 집계 함수(Agg4-1)는 미리 실행되어, 각 계산 함수(C)의 가중치 계산이 완료되는 즉시, 가중치를 총합할 수 있다.
- [0101] 본 개시의 일 실시예에서, 복수의 단계 중 첫번째 단계의 계산 함수(C)만 글로벌 파라미터를 읽고, 나머지 단계의 계산 함수(C)들은 첫번째 단계의 계산 함수(C)가 읽은 글로벌 파라미터를 사용할 수 있다. 복수의 단계 중 첫번째 단계의 계산 함수(C)는 나머지 단계의 계산 함수(C)들에 글로벌 파라미터를 전달할 수 있다. 복수의 단계에서 실행되는 계산 함수(C)들은 동일한 이터레이션에 대해 동일한 컨테이너를 사용하여 실행되므로, 첫번째 단계의 계산 함수(C)만 글로벌 파라미터를 읽더라도 나머지 단계의 계산 함수(C)들도 동기화된 가중치를 획득할 수 있다.
- [0102] 예를 들어, 제2 이터레이션(iteration 2)에서 복수의 제1 마이크로서비스(M1)의 제1 계산 함수(C1-2)와 복수의 제2 마이크로서비스(M2)의 제2 계산 함수(C2-2)는 복수의 단계인 2단계로 실행될 수 있다. 제1 계산 함수(C1-2)만 제1 집계 함수(Agg1-1), 제2 집계 함수(Agg2-1), 제3 집계 함수(Agg3-1), 및 제4 집계 함수(Agg4-1)로부터 글로벌 파라미터를 읽을 수 있다. 제1 계산 함수(C1-2)는 글로벌 파라미터를 제2 계산 함수(C2-2)에 전달할 수 있다. 제2 계산 함수(C2-2)는 제1 계산 함수(C1-2)와 동일한 컨테이너에서 실행되므로, 제1 계산 함수(C1-2)가 읽은 글로벌 파라미터를 사용하며, 가중치를 계산할 수 있다.
- [0103] 일 실시예에서, 계산 함수(C)는 간접 통신을 통해 분산된 인-메모리 데이터베이스(260)에 저장된 글로벌 파라미터를 읽는데, 복수의 단계 중 첫번째 단계만 간접 통신을 수행하여 글로벌 파라미터를 읽을 수 있으므로, 통신 오버헤드가 감소할 수 있다. 예컨대, q개의 논리 작업자가 존재하는 트레이닝 잡이 p 단계로 실행되는 경우, 분산된 인-메모리 데이터베이스(260)를 통해 글로벌 파라미터를 q/p번을 읽을 수 있다(p, q는 자연수). 도 5에서 복수의 단계에서 계산 함수들이 글로벌 파라미터를 q번 읽는 것에 대비하여 통신 오버헤드가 감소할 수 있다.

- [0104] 본 개시의 일 실시예에서, 각 이터레이션마다 복수의 단계들의 계산 함수(C)들은 계산된 가중치를 각각 그 다음 단계의 계산 함수(C)에 전달할 수 있다. 마지막 단계의 계산 함수(C)는 이전 단계들의 계산 함수(C)들 각각에서 계산된 가중치를 총합하여, 분산된 인-메모리 데이터베이스에 업데이트할 수 있다. 집계 함수(Agg)는 분산된 인-메모리 데이터베이스에 업데이트된 가중치의 사본(copy)을 읽을 수 있다.
- [0105] 예를 들어, 제2 이터레이션(iteration 2)에서 복수의 제1 마이크로서비스(M1)의 제1 계산 함수(C1-2)는 계산된 가중치를 복수의 제2 마이크로서비스(M2)의 제2 계산 함수(C2-2)에 전달할 수 있다. 하나의 이터레이션에서 제1 계산 함수(C1-2)와 제2 계산 함수(C2-2)가 동일한 컨테이너를 사용하여 실행되기 때문이다. 제2 계산 함수(C2-2)는 제1 계산 함수(C1-2)로부터 수신한 가중치와 제2 계산 함수(C2-2)에서 계산된 가중치를 총합할 수 있다. 제2 계산 함수(C2-2)는 총합한 가중치를 분산된 인-메모리 데이터베이스에 업데이트할 수 있다. 제1 집계 함수(Agg1-1), 제2 집계 함수(Agg2-1), 제3 집계 함수(Agg3-1), 및 제4 집계 함수(Agg4-1) 각각은 업데이트된 가중치를 읽을 수 있다.
- [0106] 일 실시예에서, 계산 함수(C)는 간접 통신을 통해 분산된 인-메모리 데이터베이스(260)에 가중치를 업데이트할 수 있는데, 복수의 단계 중 마지막 단계만 간접 통신을 수행하여 가중치를 업데이트할 수 있으므로, 통신 오버헤드가 감소할 수 있다. 예컨대, q개의 논리 작업자가 존재하는 트레이닝 잡이 p 단계를 통해 실행되는 경우, 분산된 인-메모리 데이터베이스(260)에 가중치를 q/p번 업데이트할 수 있다. 도 5에서 복수의 단계의 계산 함수가 가중치를 q번 업데이트하는 것에 대비하여 통신 오버헤드가 감소할 수 있다. 한편, 상술한 논리 작업자와 가용 프로세싱 유닛의 개수는 상술한 예시에 제한되지 않는다.
- [0107] 이하, 도 7 및 도 8을 참조하여, 트레이닝 잡에 할당되는 프로세싱 유닛의 개수를 결정하는 과정에 대해 설명한다.
- [0108] 도 7 및 도 8은 일 실시예에 따른 트레이닝 시스템에서, 복수의 프로세싱 유닛에 스케줄링된 트레이닝 잡을 예시한 도면이다.
- [0109] 본 개시의 실시예에서, 트레이닝 시스템(100)은 트레이닝 잡을 스케줄러(130)에 제출할 수 있고, 스케줄러(130)는 제출된 트레이닝 잡을 큐(queue)에 추가할 수 있다. 자원 관리자(150)는 큐에 대기 중인 트레이닝 잡에 할당할 자원의 개수, 예컨대 프로세싱 유닛의 개수를 결정할 수 있다. 자원 관리자(150)는 트레이닝 잡에 소정의 개수의 프로세싱 유닛을 할당할 수 있다.
- [0110] 예를 들어, 자원 관리자(150)는 공평성을 고려하여 최대-최소 공평성(Max-min fairness) 방식으로 자원을 할당할 수 있다. 또는, 예를 들어, 자원 관리자(150)는 효율성을 고려하여 SRSF(shortest-remaining-service-first) 방식으로 자원을 할당할 수도 있다. 스케줄러(130)는 프로세싱 유닛의 가용 상태, 즉 할당된 프로세싱 유닛의 개수에 기초하여, 트레이닝 잡에 속한 복수의 마이크로서비스에, 할당된 프로세싱 유닛을 스케줄링할 수 있다.
- [0111] 본 개시의 실시예에서, 자원 관리자(150)는 트레이닝 잡을 실행하기 위해 필요한 작업자의 개수와 같거나 그보다 적은 수의 프로세싱 유닛을 할당할 수 있다. 스케줄러(130)는 할당된 프로세싱 유닛의 개수가 필요한 작업자의 개수와 같은 경우, 복수의 마이크로서비스가 단일 단계(single phase)로 실행되도록 프로세싱 유닛을 스케줄링할 수 있다. 또한, 스케줄러(130)는 할당된 프로세싱 유닛의 개수가 필요한 작업자의 개수보다 적은 경우, 복수의 마이크로서비스가 복수의 단계(multiple phases)로 순차적으로 실행되도록 프로세싱 유닛을 스케줄링할 수 있다. 본 개시의 실시예에서, 트레이닝 잡에 속한 복수의 마이크로서비스 중 일부는 어느 하나의 프로세싱 유닛에서 p단계로 실행될 수 있다. 여기서, p단계는 단일 단계(single phase) 또는 복수의 단계(multiple phases)일 수 있다(p는 자연수).
- [0112] 본 개시의 실시예에서, 트레이닝 시스템(100)은 트레이닝 잡에 프로세싱 유닛을 2^n 단위로 할당할 수 있다(n은 0 또는 자연수). 예를 들어, 어느 하나의 트레이닝 잡이 트레이닝을 수행하기 위해 2^n 개의 작업자가 필요한 경우, 자원 관리자(150)는 2^n 단위로 프로세싱 유닛을 할당할 수 있다. 예를 들어, 트레이닝 잡이 2^n 개의 논리 작업자에 의해 개별적으로 실행되는 2^n 개의 마이크로서비스를 갖는 경우, 자원 관리자(150)는 해당 트레이닝 잡에 프로세싱 유닛을 2^n 의 약수 개만큼 할당할 수 있다. 스케줄러(130)는 해당 트레이닝 잡에 할당된 2^n 의 약수 개의 프로세싱 유닛을 복수의 마이크로서비스에 스케줄링할 수 있다.
- [0113] 도 7을 참조하여 트레이닝 잡(721)이 트레이닝을 수행하기 위해 8개의 작업자가 필요한 경우를

예시한다(예컨대, $n=3$). 이 경우, 어느 하나의 트레이닝 잡(721)은 8개의 논리 작업자를 가지며, 작업 관리자(150)는 8개의 논리 작업자에 8의 약수 개(즉, 1, 2, 4, 또는 8개)의 프로세싱 유닛을 할당할 수 있다. 작업 관리자(150)는 8개의 논리 작업자와 동일한 수 또는 그보다 적은 수의 프로세싱 유닛을 할당할 수 있다. 예를 들어, 복수의 논리 작업자의 개수와 할당된 프로세싱 유닛의 개수가 같은 경우, 스케줄러(130)는 각각의 논리 작업자에 대응하는 복수의 마이크로서비스 각각에 일대일로 프로세싱 유닛을 스케줄링할 수 있다. 이 경우, 복수의 마이크로서비스 각각은 단일 단계(예컨대, $p=1$)로 실행될 수 있다. 또는, 예를 들어, 복수의 논리 작업자의 개수보다 할당된 프로세싱 유닛의 개수가 적은 경우, 스케줄러(130)는 복수의 논리 작업자에 대응하는 복수의 마이크로서비스를 적어도 어느 하나의 프로세싱 유닛에 스케줄링하며, 복수의 마이크로서비스는 적어도 하나의 프로세싱 유닛에서 복수의 단계(예컨대, $p>1$)로 실행될 수 있다.

[0114] 도 7에는, 8개의 작업자가 필요한 트레이닝 잡(721)에 4개의 프로세싱 유닛(예컨대, GPU1, GPU2, GPU3, GPU4)이 할당되는 경우가 예시되어 있다. 작업 관리자(150)는 트레이닝 잡(721)에 4개의 프로세싱 유닛(GPU1, GPU2, GPU3, GPU4)을 할당할 수 있다. 이 경우, 스케줄러(130)는 2개의 논리 작업자를 하나의 프로세싱 유닛에 스케줄링할 수 있으므로, 복수의 논리 작업자에 대응되는 복수의 마이크로서비스는 2단계(예컨대, $p=2$)로 순차 실행될 수 있다. 트레이닝 잡(721)에 대응하는 8개의 마이크로서비스의 사슬은 4개의 프로세싱 유닛을 이용하여 2단계로 실행될 수 있다.

[0115] 보다 구체적으로, 트레이닝 잡(721)은 제1 논리 작업자(LW1), 제2 논리 작업자(LW2), 제3 논리 작업자(LW3), 제4 논리 작업자(LW4), 제5 논리 작업자(LW5), 제6 논리 작업자(LW6), 제7 논리 작업자(LW7), 및 제8 논리 작업자(LW8)를 포함할 수 있다. 즉, 트레이닝 잡은 8개의 논리 작업자를 포함하며, 트레이닝 잡을 실행하기 위해 8개의 작업자가 필요할 수 있다. 제1 논리 작업자(LW1), 제2 논리 작업자(LW2), 제3 논리 작업자(LW3), 제4 논리 작업자(LW4), 제5 논리 작업자(LW5), 제6 논리 작업자(LW6), 제7 논리 작업자(LW7), 및 제8 논리 작업자(LW8) 각각은 복수의 제1 마이크로서비스(M1), 복수의 제2 마이크로서비스(M2), 복수의 제3 마이크로서비스(M3), 복수의 제4 마이크로서비스(M4), 복수의 제5 마이크로서비스(M5), 복수의 제6 마이크로서비스(M6), 복수의 제7 마이크로서비스(M7), 및 복수의 제8 마이크로서비스(M8)에 대응할 수 있다.

[0116] 복수의 논리 작업자(LW1, LW2, LW3, LW4, LW5, LW6, LW7, LW8)에는 복수의 프로세싱 유닛(GPU1, GPU2, GPU3, GPU4)이 스케줄링될 수 있다. 예를 들어, 복수의 논리 작업자(LW1, LW2, LW3, LW4, LW5, LW6, LW7, LW8)에는 8의 약수 개 중 하나인, 4개의 프로세싱 유닛이 스케줄링될 수 있다. 예를 들어, 제1 논리 작업자(LW1) 및 제2 논리 작업자(LW2)에 제1 프로세싱 유닛(GPU1)이 스케줄링되고, 제3 논리 작업자(LW3) 및 제4 논리 작업자(LW4)에 제2 프로세싱 유닛(GPU2)이 스케줄링되고, 제5 논리 작업자(LW5) 및 제6 논리 작업자(LW6)에 제3 프로세싱 유닛(GPU3)이 스케줄링되고, 제7 논리 작업자(LW7) 및 제8 논리 작업자(LW8)에 제4 프로세싱 유닛(GPU4)이 스케줄링될 수 있다. 이 경우, 제1 논리 작업자(LW1)에 대응하는 복수의 제1 마이크로서비스(M1) 및 제2 논리 작업자(LW2)에 대응하는 복수의 제2 마이크로서비스(M2)는 제1 프로세싱 유닛(GPU1)을 통해 2단계로 순차 실행될 수 있다. 제3 논리 작업자(LW3)에 대응하는 복수의 제3 마이크로서비스(M3) 및 제4 논리 작업자(LW4)에 대응하는 복수의 제4 마이크로서비스(M4)는 제2 프로세싱 유닛(GPU2)을 통해 2단계로 순차 실행될 수 있다. 복수의 제5 마이크로서비스(M5), 복수의 제6 마이크로서비스(M6), 복수의 제7 마이크로서비스(M7), 및 복수의 제8 마이크로서비스(M8)에 대한 설명은 상술한 복수의 제1 마이크로서비스(M1) 및 복수의 제2 마이크로서비스(M2)와 동일하므로 생략한다.

[0117] 도 8을 참조하면, 2^n 단위의 작업자가 필요한 트레이닝 잡(821)에 2^n 단위가 아닌 프로세싱 유닛이 할당된 경우를 예시하고 있다. 즉, 트레이닝 잡(821)에 2^n 의 약수가 아닌 개수의 프로세싱 유닛이 할당된 경우를 예시하고 있다. 이 경우, 적어도 하나의 프로세싱 유닛이 처리해야하는 마이크로서비스의 수는, 다른 프로세싱 유닛들 각각이 처리해야하는 마이크로서비스의 수와 다를 수 있다. 예를 들어, 마이크로서비스를 단일 단계로 실행하는 프로세싱 유닛의 실행 시간(execution time)은 마이크로서비스를 복수의 단계로 실행하는 프로세싱 유닛의 실행 시간보다 짧을 수 있다. 이 경우, 적어도 하나의 프로세싱 유닛은, 다른 프로세싱 유닛들이 이터레이션을 마무리할 때까지 유휴(idle) 상태에 처할 수 있다. 유휴(idle) 상태는, 프로세싱 유닛이 어떠한 프로그램에 의해서도 사용되지 않는 상태를 의미한다. 유휴 상태의 프로세싱 유닛이 존재하는 경우, 자원 할당의 효율성이 감소할 수 있다.

[0118] 보다 구체적으로, 8개의 작업자가 필요한 트레이닝 잡(821)에 5개의 프로세싱 유닛(예컨대, GPU1, GPU2, GPU3, GPU4, GPU5)이 할당되는 경우가 예시되어 있다. 작업 관리자(150)는 트레이닝 잡(821)에 5개의 프로세싱 유닛(GPU1, GPU2, GPU3, GPU4, GPU5)을 할당할 수 있다. 이 경우, 제7 논리 작업자(LW7)에 제4 프로세싱 유닛

(GPU4)이 할당되고, 제8 논리 작업자(LW8)에 제5 프로세싱 유닛(GPU5)이 할당될 수 있다. 제4 프로세싱 유닛(GPU4)과 제5 프로세싱 유닛(GPU5)은 각각 마이크로서비스를 단일 단계(single phase)로 실행할 수 있다. 제4 프로세싱 유닛(GPU4)과 제5 프로세싱 유닛(GPU5)의 실행 시간은, 다른 프로세싱 유닛들(예컨대, GPU1, GPU2, GPU3)이 복수의 단계(multiple phases)에 따른 이터레이션을 마무리할 때까지 유휴 상태에 처할 수 있다.

[0119] 따라서, 본 개시의 일 실시예에서 트레이닝 시스템(100)은 2^n 개의 논리 작업자를 포함하는 트레이닝 잡에 2^n 단위의 프로세싱 유닛을 할당하고, 스케줄링할 수 있다. 트레이닝 시스템(100)은 프로세싱 유닛을 도 7과 같이 2^n 의 약수 개로 할당하는 경우, 성능을 최적화하면서 자원 할당을 최소화할 수 있다. 한편, 본 개시의 일 실시예에서, 신경망 모델에 대응하는 트레이닝 잡이 2^n 단위의 논리 작업자를 포함하지 않는 경우, 트레이닝 시스템(100)은 해당 트레이닝 잡에 할당하는 프로세싱 유닛의 개수를 다르게 결정할 수도 있다.

[0120] 예를 들어, 트레이닝 잡은 $2k$ (k 는 자연수)개의 논리 작업자를 포함할 수 있다. 트레이닝 시스템(100)의 스케줄러(130)는, 트레이닝 잡이 $2k$ 개의 논리 작업자를 포함하는 경우, 복수의 프로세싱 유닛을 $2k$ 의 약수 중 어느 하나의 단위로 할당할 수 있다. 예를 들어, 트레이닝 잡이 6개의 논리 작업자를 포함하는 경우, 스케줄러(130)는 복수의 프로세싱 유닛을 1, 2, 3, 6개 중 어느 하나의 단위로 할당할 수 있다. 이에 따라, 하나의 프로세싱 유닛에서 논리 작업자는 p 단계로 실행될 수 있다(p 는 자연수).

[0121] 예를 들어, 트레이닝 잡은 $2k-1$ 개의 논리 작업자를 포함할 수 있다. 트레이닝 시스템(100)의 스케줄러(130)는, 트레이닝 잡이 $2k-1$ 개의 논리 작업자를 포함하는 경우, 복수의 프로세싱 유닛을 $2k-1$ 의 약수 중 어느 하나 또는 $2k$ 의 약수 중 1과 $2k$ 을 제외한 어느 하나의 단위로 할당할 수 있다. 예를 들어, 트레이닝 잡이 9개의 논리 작업자를 포함하는 경우, 스케줄러(130)는 복수의 프로세싱 유닛을 9의 약수인 1, 3, 9 중 어느 하나의 단위로 할당하거나, 10의 약수 중 1과 10을 제외한 2, 5 중 어느 하나의 단위로 할당할 수 있다. 이에 따라, 하나의 프로세싱 유닛에서 논리 작업자는 p 단계로 실행될 수 있다. 한편, 스케줄러(130)가 논리 작업자에 복수의 프로세싱 유닛을 $2k$ 의 약수 중 어느 하나인 2 또는 5의 단위로 할당하는 경우, 도 8과 같이 유휴 상태의 프로세싱 유닛이 발생할 수는 있다.

[0123] 도 9는 일 실시예에 따른 트레이닝 시스템의 동작 방법을 나타낸 흐름도이다.

[0124] 도 9를 참조하면, 동작 S910에서, 트레이닝 시스템(100)은 트레이닝 잡(120)에 대한 트레이닝 데이터를 복수의 미니 배치로 분할할 수 있다. 복수의 마이크로서비스는 복수의 미니 배치를 처리하는 함수를 포함할 수 있다. 예를 들어, 200개의 트레이닝 데이터는 32개의 단위로 분할될 수 있으며, 분할된 각 단위는 미니 배치에 해당할 수 있다.

[0125] 예를 들어, 복수의 마이크로서비스는 복수의 미니 배치에 대한 포워드 패스(forward pass) 및 백워드 패스(backward pass)를 통해 복수의 미니 배치 각각을 처리하여, 복수의 미니 배치 각각에 대한 가중치를 계산하는 계산 함수(C)의 사슬을 포함할 수 있다. 복수의 미니 배치 각각에 대한 가중치는 복수의 마이크로서비스의 집계 함수(Agg)에 의해 합쳐져, 동기화될 수 있다.

[0126] 동작 S920에서, 트레이닝 시스템(100)은 신경망 모델에 대응하는 트레이닝 잡에 복수의 프로세싱 유닛을 2^n 단위로 할당할 수 있다. 예를 들어, 트레이닝 잡을 수행하기 위해 2^n 단위의 논리 작업자가 필요한 경우, 자원 관리자(150)는 2^n 의 약수 개의 프로세싱 유닛을 할당할 수 있다. 동작 S920은, 도 10에서 후술할 자원 관리자(226)에 의해 수행될 수 있다.

[0127] 동작 S930에서, 트레이닝 시스템(100)은 가용 프로세싱 유닛의 개수가 복수의 논리 작업자의 개수보다 적다고 판단된 것에 기초하여, 복수의 제1 마이크로서비스 및 복수의 제2 마이크로서비스를 어느 하나의 프로세싱 유닛에 순차적으로 스케줄링할 수 있다. 스케줄러(130)는 복수의 제1 마이크로서비스 중 제1 유형의 마이크로서비스와 복수의 제2 마이크로서비스 중 제1 유형의 마이크로서비스가 동일한 이터레이션에서 순차적으로 실행되도록 스케줄링할 수 있다. 또한, 복수의 논리 작업자 중 제3 논리 작업자에 의해 실행되는 복수의 제3 마이크로서비스 및 복수의 논리 작업자 중 제4 논리 작업자에 의해 실행되는 복수의 제4 마이크로서비스를 포함하되, 스케줄러(130)는 복수의 제3 마이크로서비스 및 복수의 제4 마이크로서비스를 복수의 프로세싱 유닛 중 다른 하나의 프로세싱 유닛에 순차적으로 실행되도록 스케줄링할 수 있다.

- [0128] 본 개시의 일 실시예에서, 스케줄러(130)는 복수의 제1 마이크로서비스와 복수의 제2 마이크로서비스를 동일한 컨테이너에 스케줄링할 수 있다.
- [0129] 본 개시의 일 실시예에서, 스케줄러(130)는 컨테이너의 콜드 스타트에 의한 오버헤드, 복수의 마이크로서비스 간의 로컬리티, 또는 복수의 마이크로서비스 간의 성능 간섭에 기초하여 복수의 마이크로서비스를 복수의 프로세싱 유닛에 재스케줄링할 수 있다.
- [0130] 본 개시의 일 실시예에서, 스케줄러(130)는 어느 하나의 프로세싱 유닛에서 복수의 제1 마이크로서비스 중 어느 하나의 미니 배치 처리와 복수의 제2 마이크로서비스 중 어느 하나의 미니 배치 처리를 복수의 단계(multiple phases)로 수행하도록 스케줄링할 수 있다. 예를 들어, 스케줄러(130)는 복수의 제1 마이크로서비스(M1) 중 어느 하나의 미니 배치를 처리하는 제1 계산 함수(C1-1)와 복수의 제2 마이크로서비스(M2) 중 어느 하나의 미니 배치를 처리하는 제2 계산 함수(C2-1)는 복수의 단계로 실행되도록 스케줄링할 수 있다(도 5 참조).
- [0131] 동작 S940에서, 트레이닝 시스템(100)은 스케줄링에 기초하여, 어느 하나의 프로세싱 유닛에서 복수의 제1 마이크로서비스 중 어느 하나의 미니 배치 처리와 복수의 제2 마이크로서비스 중 어느 하나의 미니 배치 처리를 복수의 단계(multiple phases)로 수행할 수 있다. 예를 들어, 도 5를 참조하면, 제1 이터레이션(iteration 1)에서 복수의 제1 마이크로서비스(M1) 중 어느 하나의 미니 배치를 처리하는 제1 계산 함수(C1-1)와 복수의 제2 마이크로서비스(M2) 중 어느 하나의 미니 배치를 처리하는 제2 계산 함수(C2-1)는 복수의 단계로 실행될 수 있다.
- [0132] 한편, 복수의 제3 마이크로서비스(M3)의 미니 배치 처리와 복수의 제1 마이크로서비스(M1)의 미니 배치 처리는 병렬적으로 실행될 수 있다.
- [0134] 도 10은 일 실시예에 따른 트레이닝 시스템의 구성을 도시한 도면이다.
- [0135] 도 10을 참조하면, 일 실시예에 따른 트레이닝 시스템(200)은 예를 들어, 잡 프록시(Job proxy)(210), 컨트롤러(Controllor)(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 10에 도시된 '고래 아이콘'은 예를 들어, docker와 같은 컨테이너(container)를 의미할 수 있다.
- [0136] 트레이닝 시스템(200)은 트레이닝을 위한 잡을 신경망 모델의 논리 작업자에 의해 수행되는 복수의 마이크로서비스로 분할한다. 트레이닝 시스템(200)은 복수의 마이크로서비스의 신경망 모델을 복수의 클러스터 각각에 포함된 이종의 그래픽 프로세싱 유닛에게 스케줄링함으로써 신경망 모델을 트레이닝한다. 이를 위한 트레이닝 시스템(200)의 각 구성 요소들의 동작은 다음과 같다.
- [0137] 잡 프록시(210)는 신경망 모델의 트레이닝을 위한 트레이닝 데이터에 대해 전처리(pre-processing)를 수행할 수 있다. 여기서, 전처리는 예를 들어, 트레이닝 데이터를 마이크로서비스 함수 단위로 나누는 것일 수 있다. 잡 프록시(210)는 전처리된 트레이닝 데이터에 대해 복수의 논리 작업자가 사용할 초기 가중치를 생성하고, 초기 가중치를 분산 파일 시스템(250)에 저장할 수 있다. 잡 프록시(210)는 전처리된 트레이닝 데이터에 대한 계산 유형(예를 들어, 계산 함수 또는 집계 함수)에 따라 트레이닝 잡을 복수의 마이크로서비스로 분할할 수 있다. 잡 프록시(210)는 트레이닝 잡을 마이크로서비스들의 사슬로 변환할 수 있다.
- [0138] 잡 프록시(210)는 입력 관리자(input manager)(213) 및 잡 관리자(job manager)(216)를 포함할 수 있다.
- [0139] 입력 관리자(213)는 트레이닝 데이터를 복수의 마이크로서비스에 대응하는 형태로 가공할 수 있다. 입력 관리자(213)는 예를 들어, 트레이닝 데이터를 미니 배치별로 분할하거나, 또는 마이크로서비스 단위로 분할할 수 있다.
- [0140] 잡 관리자(216)는 분할된 복수의 마이크로서비스를, 예를 들어 매개 변수(또는 가중치)를 산출하는 제1 유형의 마이크로서비스와 매개 변수를 총합하는 제2 유형의 마이크로서비스로 분류할 수 있다. 또한, 잡 관리자(216)는 분류된 복수의 마이크로서비스에 대한 정보를 스케줄러(223)에 전달할 수 있다. 예를 들어, 잡 관리자(216)는 분류된 복수의 마이크로서비스의 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(223)에 전달할 수 있다.
- [0141] 전술한 바와 같이, 복수의 논리 작업자는 자신에게 할당된 트레이닝 데이터를 트레이닝할 수 있다. 이때, 트레이닝 과정이 완료된 최종 매개 변수는 예를 들어, 분산 파일 시스템(250)에 저장될 수 있다. 또한, 트레이닝 과

정에서 지역적으로 계산된 신경망 모델의 매개 변수, 그래디언트(gradient), 및 활성 출력과 같은 중간 데이터(intermediate data), 또는 트레이닝 과정에서 전역적으로 공유되는 매개 변수는 분산된 인-메모리 데이터베이스(260)에 저장될 수 있다. 매개 변수들은 트레이닝 과정의 계속되는 이터레이션을 통해 논리 작업자들 간에 공유될 수 있으며, 매개 변수를 동기화하는 다양한 정책들이 존재할 수 있다.

[0142] 잡 관리자(216)는 매개 변수에 대한 동기화 정책을 기초로, 복수의 마이크로서비스를 계산 유형에 따라 분류할 수 있다. 이때, 매개 변수를 동기화하는 정책들로는 도 2에서 예시한 BSP(Bulk Synchronous Parallel) 모델 외에도, ASP(Asynchronous Parallel) 모델, 및 SSP(Stale Synchronous Parallel) 모델 등이 예시될 수 있다. BSP 모델은 장벽 동기화를 이용하며, 장벽에 도달하는 논리 작업자는 다른 모든 논리 작업자가 도달할 때까지 기다림으로써 모든 논리 작업자들에서 동기화가 함께 진행되도록 한다. 일 실시예에서는 자원의 상황이나 신경망 모델의 특성에 따라 전술한 BSP, ASP, 또는 SSP 모델 등의 동기화 정책을 동적으로 변경하여 신경망의 트레이닝을 수행할 수 있다.

[0143] 잡 관리자(216)는 트레이닝에 의해 발생하는 매개 변수에 대한 동기화 정책을 기초로, 분류된 복수의 마이크로서비스에 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.

[0144] 컨트롤러(220)는 잡 프로시(210)에 의해 분할된 복수의 마이크로서비스를 이종의 그래픽 프로세싱 유닛에게 스케줄링하고, 스케줄링에 따라 자원(resource)을 할당하는 등의 자원 관리를 수행할 수 있다.

[0145] 컨트롤러(220)는 예를 들어, 스케줄러(scheduler)(223) 및 자원 관리자(resource manager)(226)를 포함할 수 있다. 스케줄러(223)는 도 1의 스케줄러(130)에 대응하고, 자원 관리자(226)는 도 7의 자원 관리자(150)에 대응할 수 있다.

[0146] 스케줄러(223)는 복수의 마이크로서비스를 클러스터에 포함된 복수의 그래픽 프로세싱 유닛에 스케줄링할 수 있다. 스케줄러(223)는 그래픽 프로세싱 유닛의 가용 상태에 기초하여, 복수의 마이크로서비스를 복수의 그래픽 프로세싱 유닛에 동적으로 할당할 수 있다. 예를 들어, 스케줄러(223)는 클러스터의 자원 상황을 고려하여, 트레이닝 잡을 실행하기 위해 필요한 논리 작업자의 수보다 가용 상태의 프로세싱 유닛의 수가 적은 경우, 동일한 프로세싱 유닛에 복수의 마이크로서비스를 스케줄링할 수 있다. 다시 말해, 스케줄러(223)는 신경망 모델의 트레이닝을 위해 필요한 논리 작업자의 수보다 적은 수의 자원을 할당할 수 있으므로, 자원 활용의 탄력성(elasticity)을 높일 수 있다. 또한, 스케줄러(223)를 통해 자원을 할당하는 경우, 작업자의 수 또는 로컬 배치(local batch)의 사이즈와 같은 하이퍼파라미터(hyper-parameter)가 변경되지 않으므로, 컨버전스 특성(convergence behavior)을 유지할 수 있다.

[0147] 스케줄러(223)는 컨테이너의 콜드 스타트에 의한 오버헤드, 로컬리티, 성능간섭 등에 기초하여 복수의 마이크로서비스에 프로세싱 유닛을 스케줄링할 수 있다.

[0148] 예를 들어, 스케줄러(223)는 동일한 컨테이너(container)에 복수의 마이크로서비스가 실행될 수 있도록, 복수의 마이크로서비스를 스케줄링할 수 있다. 스케줄러(223)는 동일한 컨테이너(container)에 복수의 마이크로서비스를 스케줄링함으로써, 컨테이너의 콜드 스타트(cold-start)에 의한 오버헤드(overhead)를 최소화할 수 있다. 마이크로서비스가 호출되면 마이크로서비스를 실행하기 전에 컨테이너를 초기화하는 동작인 "콜드 스타트"를 위한 시간이 소비되며, 콜드 스타트에 의해 통신 오버헤드가 발생할 수 있다. 본 개시의 일 실시예에서, 스케줄러(223)는 적어도 두개의 마이크로서비스(예컨대, 도 5의 제1 계산 함수(C1-1)와 제2 계산 함수(C2-1))를 실행하기 위해 동일한 컨테이너를 재사용하므로, 웜 컨테이너(warm container) 상태로 사용할 수 있다. 따라서, 마이크로서비스를 실행하기 전에 컨테이너를 초기화하는 동작이 생략되므로, 콜드 스타트에 의한 오버헤드를 최소화할 수 있다.

[0149] 예를 들어, 스케줄러(223)는 트레이닝 잡의 마이크로서비스를 실행하는 데 사용되는 컨테이너의 로컬리티(locality)를 고려할 수 있다. 마이크로서비스들 간 로컬리티(locality)는, 예를 들어 마이크로서비스들 각각이 사용해야 하는 중간 데이터와의 거리에 해당할 수 있다. 스케줄러(223)는 분산된 인-메모리 데이터베이스(260)에 저장된 마이크로서비스가 사용해야 할 중간 데이터의 위치와 마이크로서비스가 실행될 GPU의 위치를 고려하여 마이크로서비스들을 스케줄링할 수 있다. 스케줄러(223)는 로컬리티에 민감한 트레이닝 잡이 최소한의 노드를 사용하여 실행될 수 있도록 스케줄링할 수 있다. 예를 들어, 최초에 스케줄링된 컨테이너들 간의 거리가 멀어 로

컬리타가 좋지 않은 경우, 스케줄러(223)는 더 나은 로컬리타를 갖는 노드가 가용 상태로 변하면 마이크로서비스에 대한 컨테이너를 재스케줄링(rescheduling)할 수 있다. 따라서, 스케줄러(223)은 복수의 마이크로서비스 간 로컬리타를 향상시킬 수 있다.

[0150] 예를 들어, 스케줄러(223)는 마이크로서비스들 간의 간섭(interference)을 최소화할 수 있다. 예를 들어, 한 노드는 멀티 코어 및 멀티 GPU로 설정된 경우, 한 노드에서 동시에 여러 마이크로서비스들이 실행될 수 있다. 마이크로서비스들 간의 간섭은 공유 클러스터 환경에서 한 노드에서 실행되는 다른 사용자의 서버리스 함수와의 경쟁으로 인해 노드의 물리적 자원에 대한 한계로 성능이 저하되는 것을 의미한다. 스케줄러(223)는 최초로 스케줄링된 노드의 성능이 저하된다면, 가용 상태로 변한 다른 노드에 마이크로서비스를 재스케줄링(rescheduling)할 수 있다. 따라서, 스케줄러(223)는 마이크로서비스들 간의 간섭을 최소화하고, 성능을 향상시킬 수 있다.

[0151] 자원 관리자(226)는 스케줄러(223)의 스케줄링에 따라 할당되는 복수의 프로세싱 유닛의 개수를 조절할 수 있다. 예를 들어, 자원 관리자(226)는 트레이닝 잡에 어떤 클러스터를 할당하는 지, 몇 개의 프로세싱 유닛을 할당하는 지를 결정할 수 있다. 예를 들어, 자원 관리자(226)는 논리 작업자의 개수가 2^n 단위인 경우, 2^n 의 약수 단위의 프로세싱 유닛을 할당하도록 결정할 수 있다.

[0152] 본 개시의 일 실시예에서, 자원 관리자(226)는 최대-최소 공평성(Max-min fairness) 방식으로 자원을 할당할 수 있다. 예를 들어, 최대-최소 공평성 방식을 통해 $1/m$ 의 자원을 보장함으로써 자원 할당을 공평하게 수행할 수 있다. 여기서, m 은 클러스터에 제출된 트레이닝 잡의 개수일 수 있다. 따라서, 자원 관리자(226)는 m 개의 트레이닝 잡들 각각에 자원을 2^n 단위로 할당하는 경우, 동일한 수의 프로세싱 유닛을 필요로 하는 트레이닝 잡들 중 먼저 제출된 트레이닝 잡에 더 많은 수의 프로세싱 유닛을 할당할 수 있다.

[0153] 본 개시의 일 실시예에 따른 자원 관리자(226)는 SRSF(shortest-remaining-service-first) 방식으로 자원을 할당할 수 있다. SRSF 방식은 최단 잔여 서비스 우선 스케줄링 방식으로 지칭될 수 있다. SRSF 방식은 효율성을 고려한 GPU 스케줄링 방식으로서, 잔여 서비스 시간(remaining service time)이 가장 짧은 트레이닝 잡을 우선 실행시키는 스케줄링 방식을 의미한다. 트레이닝 잡의 잔여 시간은, 트레이닝 잡을 실행 완료하기까지 남아있는 시간, 즉 남은 실행 시간을 의미할 수 있다. 트레이닝 잡의 잔여 서비스 시간은 해당 트레이닝 잡의 남은 실행 시간(execution time)에, 해당 트레이닝 잡을 실행하기 위해 필요한 GPU의 개수를 곱하여 산출될 수 있다. 예를 들어, 자원 관리자(226)는 m 개의 트레이닝 잡을 잔여 서비스 시간이 짧은 순으로 정렬하고, 잔여 서비스 시간이 가장 짧은 트레이닝 잡부터 순서대로 프로세싱 유닛을 할당할 수 있다. 여기서, m 은 클러스터에 제출된 트레이닝 잡의 개수일 수 있다.

[0154] 한편, SRSF 방식은 일반적인 CPU 스케줄링 방식 중 하나인 SRT 스케줄링(shortest-remaining-time-scheduling, 최단 잔여 시간 우선 스케줄링) 방식이나 SJF 스케줄링(shortest-job-first-scheduling, 최단 작업 우선 스케줄링) 방식 등과 유사할 수 있다. 본 개시의 일 실시예에 따른 자원 관리자(226)는 최대-최소 공평성 방식과 SRSF 방식을 모두 고려하여 공평성 및 효율성을 동시에 고려한 자원 할당 방식을 사용할 수도 있다. 이에 관해서는 도 11 이하에서 상세히 설명하도록 한다.

[0155] 본 개시의 일 실시예에 따른 자원 관리자(226)는 큐(queue)에 저장된 대기 트레이닝 잡(queuing training job)이 존재하는 경우, 자원을 재할당할 수 있다. 예를 들어, 자원 관리자(226)는 대기 트레이닝 잡의 대기 시간(queuing time)이 하나 이상의 트레이닝 잡이 복수의 단계(multiple phases)로 실행됨에 따른 예상 증가 시간보다 긴 경우, 대기 트레이닝 잡에 프로세싱 유닛을 재할당할 수 있다. 이에 관해서는 도 14 내지 도 16에서 설명한다.

[0156] 또한, 본 개시의 일 실시예에 따른 자원 관리자(226)는 우선순위 기반 정책(Priority based policy), 데드라인 기반 정책(deadline-aware policy), 및 이기종 기반 정책(heterogeneity-aware policy) 등을 사용할 수 있고, DRF(Dominant resource fairness), two-level hierarchy Hadoop Fair Scheduler와 같은 일반적인 자원 공유 클러스터 관리에 사용하는 여러 기술을 적용할 수 있다.

[0157] 폴트 핸들러(Fault Handler)(230)는 복수의 마이크로서비스 중 오류(fault)가 발생한 마이크로서비스를 검출하고, 검출된 마이크로서비스의 동작을 재수행할 수 있다.

[0158] 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 및 백워드 패스 과정의 일부 단계이

거나 매개 변수를 처리하는 과정일 수 있다.

- [0159] 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.
- [0160] 분산된 인-메모리 데이터베이스(Distributed In - memory Database)(260)는 예를 들어, 트레이닝 과정에서 지역적으로 계산된 그래디언트와 같은 중간 데이터(intermediate data)를 저장할 수 있다. 예를 들어, 분산된 인-메모리 데이터베이스(260)는 복수의 마이크로서비스 간의 간접 통신(indirect communication)을 위해 복수의 마이크로서비스의 실행에 의해 발생한 가중치를 저장할 수 있다. 예를 들어, 분산된 인-메모리 데이터베이스(260)는, 계산 함수의 실행에 따라 발생한 가중치를 저장할 수 있다. 예를 들어, 분산된 인-메모리 데이터베이스(260)는, 집계 함수의 실행에 따라 총합한 가중치인, 글로벌 파라미터를 저장할 수 있다. 예를 들어, 그 다음 이터레이션에서 실행되는 계산 함수는 분산된 인-메모리 데이터베이스(260)에 저장된 글로벌 파라미터를 읽을 수 있다. 예를 들어, 계산 함수는 상기 집계 함수와 상기 분산된 인-메모리 데이터베이스(260)를 통해 가중치를 전달할 수 있다.
- [0161] 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.
- [0163] 이하, 도 11 내지 도 13을 참조하여 일 실시예에 따른 트레이닝 시스템(1100)에 복수의 트레이닝 잡이 제출된 경우에 대해 설명한다.
- [0164] 도 11은 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡들에 프로세싱 유닛을 할당하는 과정을 나타낸 일 예이다.
- [0165] 도 11을 참조하여 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)에 7개의 트레이닝 잡이 제출되며, 클러스터(1140)에 6개의 GPU 자원이 존재하는 경우를 예시하여 설명한다.
- [0166] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)에는 복수의 트레이닝 잡이 제출될 수 있다. 복수의 트레이닝 잡은 제1 트레이닝 잡 내지 제7 트레이닝 잡(1121, 1122, 1123, 1124, 1125, 1126, 1127)을 포함할 수 있다. 복수의 트레이닝 잡 각각은 서로 다른 신경망 모델에 대응할 수 있다. 복수의 트레이닝 잡 각각은 도 1의 트레이닝 잡(120)과 같이, 복수의 마이크로서비스로 분할될 수 있고, 하나 이상의 논리 작업자에 의해 실행될 수 있다.
- [0167] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)은 제1 트레이닝 잡 내지 제7 트레이닝 잡(1121, 1122, 1123, 1124, 1125, 1126, 1127)을 컨트롤러(controller)로 제출(submit)할 수 있다. 컨트롤러는 스케줄러(1130) 및 자원 관리자(1150)를 포함할 수 있다. 스케줄러(1130)는 복수의 트레이닝 잡을 스케줄러(1130)의 큐(queue)에 추가할 수 있다. 복수의 트레이닝 잡은 큐에 대기 중일 수 있다.
- [0168] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)은 SRSF 방식에 따라 복수의 트레이닝 잡을 잔여 서비스 시간(remaining service time) 순으로 정렬할 수 있다. 트레이닝 잡의 잔여 서비스 시간은 해당 트레이닝 잡의 잔여 실행 시간(execution time)에 해당 트레이닝 잡을 실행하기 위해 필요한 프로세싱 유닛의 개수를 곱하여 산출될 수 있다. 예를 들어, 트레이닝 시스템(1100)은 복수의 트레이닝 잡 각각의 잔여 서비스 시간을 산출하고, 산출된 잔여 서비스 시간에 기초하여 복수의 트레이닝 잡을 잔여 서비스 시간이 짧은 순으로 정렬할 수 있다.
- [0169] 예를 들어, 제1 트레이닝 잡(1121)에서 제7 트레이닝 잡(1127)으로 갈수록 잔여 서비스 시간이 증가할 수 있다. 제1 트레이닝 잡(1121)이 최단 잔여 서비스 시간을 가지므로, 제1 트레이닝 잡(1121)에 우선적으로 프로세싱 유닛이 할당될 경우, 최단 시간 내에 작업이 완료될 수 있다. 예를 들어, 제7 트레이닝 잡(1127)이 최장 잔여 서비스 시간을 가질 수 있다. 트레이닝 시스템(1100)은 제1 트레이닝 잡(1121)부터 제7 트레이닝 잡(1127)의 순서대로 프로세싱 유닛을 할당하기 위한 우선 순위를 부여할 수 있다. 큐에는 제1 트레이닝 잡(1121)부터 제7 트레이닝 잡(1127) 순으로 정렬될 수 있다.

- [0170] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)은 최대-최소 공평성 방식과 SRSF 방식을 동시에 고려한 자원 할당 방식을 사용할 수 있다. 트레이닝 시스템(1100)은 최대-최소 공평성 방식에 따라, 복수의 트레이닝 잡 각각에 $1/m$ 의 자원을 최대한 보장하기 위해 최소 개수의 프로세싱 유닛을 할당할 수 있다. 트레이닝 시스템(1100)은 최대-최소 공평성 방식과 SRSF 방식에 따라, 최단 잔여 서비스 시간을 갖는 트레이닝 잡부터 순서대로 최소 개수의 프로세싱 유닛을 할당할 수 있다. 이에 따라, 트레이닝 시스템(1100)은 최대한 많은 수의 트레이닝 잡에 프로세싱 유닛을 할당할 수 있다. 여기서, m 은 트레이닝 시스템(1100)에 제출된 트레이닝 잡의 개수일 수 있다.
- [0171] 예를 들어, 트레이닝 시스템(1100)은 복수의 트레이닝 잡에 $1/7$ 의 자원을 최대한 보장하기 위해 복수의 트레이닝 잡 각각에 1개의 프로세싱 유닛을 할당할 수 있다. 예컨대, 클러스터(1140)에 6개의 프로세싱 유닛이 존재하는 경우, 최소 개수의 프로세싱 유닛은 1개일 수 있다. 트레이닝 시스템(1100)은 잔여 서비스 시간이 가장 짧은 제1 트레이닝 잡(1121)에 먼저 하나의 프로세싱 유닛(예컨대, GPU1)을 할당할 수 있다. 트레이닝 시스템(1100)은 제2 트레이닝 잡(1122), 제3 트레이닝 잡(1123), 제4 트레이닝 잡(1124), 제5 트레이닝 잡(1125), 및 제6 트레이닝 잡(1126) 순서대로 프로세싱 유닛을 1개씩 할당할 수 있다. 트레이닝 시스템(1100)은 클러스터(1140)에 잔여 프로세싱 유닛(remaining processing unit)이 없는 경우, 잔여 서비스 시간이 가장 긴 제7 트레이닝 잡(1127)에는 프로세싱 유닛을 할당하지 않을 수 있다. 예를 들어, 클러스터(1140)에 존재하는 6개의 프로세싱 유닛(GPU1, GPU2, GPU3, GPU4, GPU5, GPU6) 각각은 6개의 트레이닝 잡에 동일한 개수로 공평하게 할당될 수 있다. 클러스터(1140)에는 남아있는 프로세싱 유닛, 즉 잔여 프로세싱 유닛이 존재하지 않을 수 있다.
- [0172] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)은 최대-최소 공평성 방식을 사용함으로써, 최대한 많은 양의 트레이닝 잡에 공평하게 자원을 할당할 수 있다. 또한, 트레이닝 시스템(1100)은 SRSF 방식을 사용함으로써, 처리 시간이 짧은 트레이닝 잡들이 빠르게 트레이닝되도록, 효율적으로 자원을 할당할 수 있다. 본 개시의 일 실시예에 따르면, 트레이닝 시스템(1100)에 제출된 복수의 트레이닝 잡의 평균 작업 완료 시간(Average Job Completion Time, Average JCT)이 최소화될 수 있다. 평균 작업 완료 시간은 복수의 트레이닝 잡 각각의 트레이닝이 완료되는 시간을 평균하여 산출된다.
- [0173] 도 12는 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡들에 프로세싱 유닛을 할당하는 과정을 나타낸 일 예이다.
- [0174] 도 12를 참조하여 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)에 4개의 트레이닝 잡이 제출되며, 클러스터(1140)에 6개의 GPU 자원이 존재하는 경우를 예시하여 설명한다.
- [0175] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)에는 복수의 트레이닝 잡이 제출될 수 있다. 복수의 트레이닝 잡은 제1 트레이닝 잡 내지 제4 트레이닝 잡(1221, 1222, 1223, 1224)을 포함할 수 있다. 복수의 트레이닝 잡 각각은 서로 다른 신경망 모델에 대응할 수 있다. 복수의 트레이닝 잡 각각은 도 1의 트레이닝 잡(120)과 같이, 복수의 마이크로서비스로 분할될 수 있고, 하나 이상의 논리 작업자들에 의해 실행될 수 있다.
- [0176] 예를 들어, 제1 트레이닝 잡(1221)은 2개의 논리 작업자에 의해 실행될 수 있다. 제1 트레이닝 잡(1221)이 실행되기 위해서는 2개의 프로세싱 유닛이 필요할 수 있다. 제1 트레이닝 잡(1221)은 2개의 논리 작업자에 의해 실행되는 2개의 마이크로서비스(M11, M12)를 포함할 수 있다. 또한, 예를 들어, 제2 트레이닝 잡(1222)은 2개의 논리 작업자에 의해 실행되는 2개의 마이크로서비스(M21, M22)를 포함할 수 있다. 또한, 예를 들어, 제3 트레이닝 잡(1223)은 2개의 논리 작업자에 의해 실행되는 2개의 마이크로서비스(M31, M32)를 포함할 수 있다. 제4 트레이닝 잡(1224)은 1개 또는 그 이상의 논리 작업자에 의해 실행되는 것으로 예시한다.
- [0177] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)은 도 11과 같이 제1 트레이닝 잡 내지 제4 트레이닝 잡(1221, 1222, 1223, 1224)을 컨트롤러(controller)로 제출(submit)할 수 있다. 스케줄러(1130)는 복수의 트레이닝 잡을 스케줄러(1130)의 큐(queue)에 추가할 수 있다. 복수의 트레이닝 잡은 큐에 대기 중일 수 있다.
- [0178] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)은 SRSF 방식에 따라 복수의 트레이닝 잡을 잔여 서비스 시간 순으로 정렬할 수 있다. 예를 들어, 제1 트레이닝 잡(1221)에서 제4 트레이닝 잡(1224)으로 갈수록 잔여 서비스 시간이 증가할 수 있다. 트레이닝 시스템(1100)은 제1 트레이닝 잡(1221)에서 제4 트레이닝 잡(1224)의 순서대로 프로세싱 유닛을 할당하기 위한 우선 순위를 부여할 수 있다. 스케줄러(1130)의 큐에는 제1 트레이닝 잡(1221)에서 제4 트레이닝 잡(1224) 순으로 정렬될 수 있다.
- [0179] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)은 도 11과 같이 최대-최소 공평성 방식과 SRSF 방식을 동시에 고려한 자원 할당 방식을 사용할 수 있다. 예를 들어, 트레이닝 시스템(1100)은 복수의 트레이닝 잡에 $1/4$ 의

자원을 최대한 보장하기 위해 복수의 트레이닝 잡 각각에 1개의 프로세싱 유닛을 할당할 수 있다. 예컨대, 클러스터(1140)에 6개의 프로세싱 유닛이 존재하는 경우, 최소 개수의 프로세싱 유닛은 1개일 수 있다. 트레이닝 시스템(1100)은 잔여 서비스 시간이 가장 짧은 제1 트레이닝 잡(1221)에 먼저 하나의 프로세싱 유닛(예컨대, GPU1)을 할당할 수 있다. 트레이닝 시스템(1100)은 제2 트레이닝 잡(1222), 제3 트레이닝 잡(1223), 및 제4 트레이닝 잡(1224) 순서대로 프로세싱 유닛을 1개씩 할당할 수 있다.

[0180] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)은 클러스터(1140)에 잔여 프로세싱 유닛이 존재하는 경우, SRSF 방식에 따라 잔여 프로세싱 유닛을 할당할 수 있다. 트레이닝 시스템(1100)이 최대한 많은 수의 트레이닝 잡에 최소 개수의 프로세싱 유닛을 할당함에 따라, 클러스터(1140)에는 잔여 프로세싱 유닛이 존재할 수 있다. 잔여 프로세싱 유닛은, 복수의 트레이닝 잡 각각에 공평하게 프로세싱 유닛이 할당되고 난 후, 클러스터(1140)에 남은 프로세싱 유닛을 의미할 수 있다. 트레이닝 시스템(1100)은 잔여 서비스 시간이 가장 짧은 트레이닝 잡부터 순서대로 잔여 프로세싱 유닛을 더 할당할 수 있다. 예를 들어, 클러스터(1140)에는 2개의 프로세싱 유닛(예컨대, GPU5 및 GPU6)이 남아있을 수 있다. 트레이닝 시스템(1100)은 제1 트레이닝 잡(1221)에서 제4 트레이닝 잡(1224) 순서대로 잔여 프로세싱 유닛을 할당할 수 있다.

[0181] 본 개시의 일 실시예에 따른 트레이닝 잡(1100)은 잔여 서비스 시간이 가장 짧은 트레이닝 잡부터 순서대로, 필요한 자원의 수보다 적은 수의 자원이 할당된 트레이닝 잡에 잔여 프로세싱 유닛을 할당할 수 있다. 각 트레이닝 잡에 필요한 자원의 수는, 각 트레이닝 잡에 속한 마이크로서비스를 실행하기 위한 논리 작업자의 수에 대응할 수 있다. 예를 들어, 제1 트레이닝 잡(1221)에 1개의 프로세싱 유닛(예컨대, GPU1)이 할당되는 경우나, 제2 트레이닝 잡(1222)에 1개의 프로세싱 유닛(예컨대, GPU2)이 할당되는 경우, 필요한 자원의 수보다 할당된 프로세싱 유닛의 수가 적을 수 있다. 트레이닝 시스템(1100)은 제1 트레이닝 잡(1221)에 잔여 프로세싱 유닛(예컨대, GPU5)을 더 할당하고, 제2 트레이닝 잡(1222)에 잔여 프로세싱 유닛(예컨대, GPU6)을 더 할당할 수 있다. 이에 따라, 제1 트레이닝 잡(1221) 및 제2 트레이닝 잡(1222)에는 각각 2개의 프로세싱 유닛이 할당되고, 제3 트레이닝 잡(1223) 및 제4 트레이닝 잡(1224)에는 각각 1개의 프로세싱 유닛이 할당될 수 있다.

[0182] 본 개시의 일 실시예에 따른 트레이닝 시스템(1100)에서, 잔여 프로세싱 유닛이 더 할당된 트레이닝 잡은, 동일한 프로세싱 유닛에서 실행되는 단계(phase)가 감소할 수 있다. 예를 들어, 제1 트레이닝 잡(1221)에 1개의 프로세싱 유닛(예컨대, GPU1)이 할당되는 경우, 제1 트레이닝 잡(1221)은 GPU1에서 2단계(2 phase)로 실행될 수 있다. 예를 들어, 제1 트레이닝 잡(1221)에 2개의 프로세싱 유닛(예컨대, GPU1 및 GPU5)이 할당되는 경우, 제1 트레이닝 잡(1221)은 GPU1에서 1단계로 실행되며 GPU5에서 1단계로 실행될 수 있다.

[0183] 본 개시의 일 실시예에 따른 트레이닝 잡(1100)은 각 트레이닝 잡에 필요한 자원의 수와 동일한 수의 프로세싱 유닛을 모두 할당할 때까지 잔여 프로세싱 유닛을 할당하는 과정을 반복 수행할 수 있다. 트레이닝 시스템(1100)은 복수의 단계(multiple phases)로 실행되는 트레이닝 잡이 최대한 빠르게 완료되도록 하기 위해, 복수의 단계로 실행되는 트레이닝 잡에 잔여 프로세싱 유닛을 더 할당할 수 있다. 트레이닝 시스템(1100)은 복수의 단계로 실행되는 트레이닝 잡에 잔여 프로세싱 유닛을 더 할당함으로써, 트레이닝 잡이 더 적은 단계 또는 단일의 단계(single phase)로 실행되도록 할 수 있다.

[0184] 본 개시의 일 실시예에 따른 트레이닝 잡(1100)은 클러스터(1140)에 잔여 프로세싱 유닛이 존재하지 않을 때까지 각 트레이닝 잡에 잔여 프로세싱 유닛을 할당하는 과정을 반복 수행할 수 있다. 예를 들어, 제1 트레이닝 잡(1221) 및 제2 트레이닝 잡(1222)에 잔여 프로세싱 유닛이 할당되고 난 후 클러스터(1140)에 잔여 프로세싱 유닛이 존재하지 않으면, 제3 트레이닝 잡(1223) 및 제4 트레이닝 잡(1224)에는 잔여 프로세싱 유닛이 할당되지 않을 수 있다. 이 경우, 제3 트레이닝 잡(1223)에는 1개의 프로세싱 유닛(예컨대, GPU3)이 할당되어, 제3 트레이닝 잡(1223)은 GPU3에서 2단계로 실행될 수 있다.

[0185] 본 개시의 일 실시예에 따른 트레이닝 잡(1100)은 잔여 서비스 시간이 짧고 복수의 단계로 실행되는 트레이닝 잡에 대한 자원 할당량을 늘림으로써, 최대한 많은 수의 트레이닝 잡이 빠르게 트레이닝되도록 할 수 있다.

[0187] 도 13은 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡들에 프로세싱 유닛을 할당하는 동작 방법을 나타낸 흐름도이다.

[0188] 도 13을 참조하면, 동작 S1310에서, 트레이닝 시스템(1100)은 최단 잔여 서비스 시간을 갖는 트레이닝 잡부터 순서대로 클러스터(1140)에 존재하는 프로세싱 유닛을 할당할 수 있다. 트레이닝 시스템(1100)은 최대한 많은 수의 트레이닝 잡에 최소 개수의 프로세싱 유닛을 할당할 수 있다. 트레이닝 시스템(1100)은 최대-최소 공평성

방식과 SRSF 방식을 동시에 고려한 자원 할당 방식을 사용할 수 있다.

- [0189] 트레이닝 시스템(1100)은 SRSF 방식에 따라 복수의 트레이닝 잡을 잔여 서비스 시간 순으로 정렬할 수 있다. 예를 들어, 트레이닝 시스템(1100)은 복수의 트레이닝 잡 각각의 잔여 서비스 시간을 산출하고, 산출된 잔여 서비스 시간에 기초하여 복수의 트레이닝 잡을 잔여 서비스 시간이 적은 순으로 정렬할 수 있다. 트레이닝 시스템(1100)은 잔여 서비스 시간이 적은 트레이닝 잡에 우선적으로 프로세싱 유닛을 할당하기 위한 우선 순위를 부여할 수 있다.
- [0190] 트레이닝 시스템(1100)은 최대-최소 공평성 방식에 따라, 복수의 트레이닝 잡 각각에 $1/m$ 의 자원을 최대한 보장하기 위해 최소 개수의 프로세싱 유닛을 할당할 수 있다. 트레이닝 시스템(1100)은 최대-최소 공평성 방식과 SRSF 방식에 따라, 최단 잔여 서비스 시간을 갖는 트레이닝 잡부터 순서대로 최소 개수의 프로세싱 유닛을 할당할 수 있다.
- [0191] 동작 S1320에서, 트레이닝 시스템(1100)은 클러스터(1140)에 잔여 프로세싱 유닛이 존재하는 지 여부를 식별할 수 있다. 잔여 프로세싱 유닛은, 복수의 트레이닝 잡 각각에 공평하게 프로세싱 유닛이 할당되고 난 후, 클러스터(1140)에 남은 프로세싱 유닛을 의미할 수 있다.
- [0192] 트레이닝 시스템(1100)이 최대-최소 방식에 따라 복수의 트레이닝 잡에 프로세싱 유닛을 할당함에 따라, 클러스터(1140)에는 잔여 프로세싱 유닛이 존재하거나, 존재하지 않을 수 있다. 예를 들어, 클러스터(1140)에는 도 11과 같이 잔여 프로세싱 유닛이 존재하지 않을 수 있다. 예를 들어, 클러스터(1140)에는 도 12와 같이 잔여 프로세싱 유닛이 존재할 수 있다.
- [0193] 클러스터(1140)에 잔여 프로세싱 유닛이 존재하지 않는 경우, 트레이닝 시스템(1100)은 프로세싱 유닛을 할당하는 동작을 완료할 수 있다. 예를 들어, 트레이닝 시스템(1100)은 스케줄러(1130)를 통해 각 트레이닝 잡에 할당된 프로세싱 유닛을 스케줄링할 수 있다.
- [0194] 동작 S1330에서, 트레이닝 시스템(1100)은 클러스터(1140)에 잔여 프로세싱 유닛이 존재함에 따라, 필요한 프로세싱 유닛의 수보다 적은 수의 프로세싱 유닛이 할당된 트레이닝 잡에 잔여 프로세싱 유닛을 할당할 수 있다. 트레이닝 시스템(1100)은 잔여 서비스 시간이 가장 짧은 트레이닝 잡부터 순서대로, 필요한 프로세싱 유닛의 수보다 적은 수의 프로세싱 유닛이 할당된 트레이닝 잡에 잔여 프로세싱 유닛을 할당할 수 있다.
- [0195] 본 개시의 일 실시예에서, 필요한 프로세싱 유닛의 수보다 적은 수의 프로세싱 유닛이 할당된 트레이닝 잡에는 해당 트레이닝 잡에 필요한 프로세싱 유닛의 개수만큼 잔여 프로세싱 유닛이 더 할당될 수 있다. 잔여 프로세싱 유닛이 더 할당된 트레이닝 잡은, 동일한 프로세싱 유닛에서 실행되는 단계(phase)가 감소할 수 있다. 예를 들어, 트레이닝 잡에 필요한 프로세싱 유닛의 수보다 적은 수의 프로세싱 유닛이 할당된 경우, 트레이닝 잡은 복수의 단계(multiple phases)로 실행될 수 있다. 트레이닝 잡에 잔여 프로세싱 유닛이 더 할당되는 경우, 동일한 프로세싱 유닛은 복수의 단계보다 더 적은 수의 단계 또는 단일의 단계(single phase)를 실행할 수 있다.
- [0196] 본 개시의 일 실시예에 따른 트레이닝 잡(1100)은 각 트레이닝 잡에 필요한 자원의 수와 동일한 수의 프로세싱 유닛을 모두 할당할 때까지 잔여 프로세싱 유닛을 할당하는 과정을 반복 수행할 수 있다. 트레이닝 시스템(1100)은 복수의 단계(multiple phases)로 실행되는 트레이닝 잡에 잔여 프로세싱 유닛을 더 할당함으로써, 트레이닝 잡이 더 적은 단계 또는 단일의 단계(single phase)로 실행되도록 할 수 있다.
- [0197] 본 개시의 일 실시예에 따른 트레이닝 잡(1100)은 클러스터(1140)에 잔여 프로세싱 유닛이 존재하지 않을 때까지 트레이닝 잡에 잔여 프로세싱 유닛을 할당하는 과정을 반복 수행할 수 있다.
- [0198] 본 개시의 일 실시예에 따른 트레이닝 잡(1100)은 잔여 서비스 시간이 짧고 복수의 단계로 실행되는 트레이닝 잡에 대한 자원 할당량을 늘림으로써, 최대한 많은 수의 트레이닝 잡이 빠르게 트레이닝되도록 할 수 있다.
- [0200] 이하, 도 14 내지 도 16을 참조하여 일 실시예에 따른 트레이닝 시스템(1400)에 복수의 트레이닝 잡이 제출된 경우에 대해 설명한다.
- [0201] 도 14는 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡에 프로세싱 유닛을 할당하는 과정을 나타낸 일 예이다.
- [0202] 도 14를 참조하여 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)에 2개의 트레이닝 잡(1410, 1420)이 제출되고, 클러스터에 2개의 GPU 자원(GPU1, GPU2)이 존재하는 경우를 예시하여 설명한다.

- [0203] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 SRSF 방식에 따라 자원을 할당할 수 있다. 예를 들어, 제1 트레이닝 잡(1410)의 잔여 서비스 시간이 제2 트레이닝 잡(1420)의 잔여 서비스 시간보다 짧을 수 있다.
- [0204] 1400A를 참조하면, 트레이닝 시스템(1400)은 제1 트레이닝 잡(1410)에 우선적으로 필요한 수만큼의 자원(예컨대, 1개의 GPU)을 할당할 수 있다. 제2 트레이닝 잡(1420)은 스케줄러의 큐에서 대기 중일 수 있다. 본 개시에서, 큐에 대기 중인 제2 트레이닝 잡(1420)은, 대기 트레이닝 잡(queueing training job)으로 지칭될 수 있다.
- [0205] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 평균 작업 완료 시간(Average JCT)이 최소화되는 방향으로 자원을 할당할 수 있다. 예를 들어, 복수의 트레이닝 잡 각각에 필요한 개수보다 적은 수의 프로세싱 유닛이 할당되고, 복수의 트레이닝 잡 각각이 복수의 단계로 실행되는 제1 방식이 평균 작업 완료 시간을 최소화할 수 있다. 또는, 예를 들어, 복수의 트레이닝 잡 각각에 필요한 개수의 프로세싱 유닛이 할당되고, 복수의 트레이닝 잡 각각이 단일의 단계로 실행되는 제2 방식이 평균 작업 완료 시간을 최소화할 수도 있다. 트레이닝 시스템(1400)은 제1 방식에 따라 평균 작업 완료 시간을 최소화할 수도 있고, 제2 방식에 따라 평균 작업 완료 시간을 최소화할 수도 있다.
- [0206] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 제1 방식에 따라 하나 이상의 트레이닝 잡이 복수의 단계로 실행되는 경우 예측되는 평균 작업 완료 시간과 제2 방식에 따라 하나 이상의 트레이닝 잡이 단일의 단계로 실행되는 경우 예측되는 평균 작업 완료 시간을 비교할 수 있다. 트레이닝 시스템(1400)은 트레이닝 잡이 복수의 단계로 실행되는 경우 평균 작업 완료 시간이 더 짧다면, 제1 방식에 따라 프로세싱 유닛을 할당할 수 있다. 트레이닝 시스템(1400)은 트레이닝 잡이 단일의 단계로 실행되는 경우 평균 작업 완료 시간이 더 짧다면, 제2 방식에 따라 프로세싱 유닛을 할당할 수 있다.
- [0207] 이하, 본 개시에서는 트레이닝 잡이 복수의 단계로 실행됨에 따라 평균 작업 완료 시간이 짧아지는 제1 방식에 대해 구체적으로 설명한다.
- [0208] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 대기 트레이닝 잡이 존재하는 경우, 대기 트레이닝 잡의 대기 시간(queueing time)과 하나 이상의 트레이닝 잡이 단일의 단계에서 복수의 단계로 실행됨에 따른 예상 증가 시간을 비교할 수 있다. 트레이닝 시스템(1400)은 대기 트레이닝 잡의 대기 시간(queueing time)이 하나 이상의 트레이닝 잡이 복수의 단계로 실행됨에 따른 예상 증가 시간보다 긴 경우, 대기 트레이닝 잡에 상기 프로세싱 유닛을 재할당할 수 있다.
- [0209] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 단일의 단계에서 복수의 단계로 실행되는 하나 이상의 트레이닝 잡으로 대기 트레이닝 잡만을 고려할 수도 있고(도 14), 대기 트레이닝 잡과 자원이 기 할당된 트레이닝 잡까지 고려할 수 있다(도 15).
- [0210] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 큐에 저장된 대기 트레이닝 잡(1420)의 대기 시간(queueing time)(qt)과 대기 트레이닝 잡(1420)이 복수의 단계(multiple phases)로 실행됨에 따른 예상 증가 시간(expected increase time)(t2-t1)을 비교할 수 있다. 대기 트레이닝 잡(1420)이 복수의 단계로 실행됨에 따른 예상 증가 시간(t2-t1)은, 대기 트레이닝 잡(1420)이 복수의 단계로 실행되는 실행 시간(execution time)(t2)과 대기 트레이닝 잡(1420)이 단일의 단계로 실행되는 실행 시간(t1)의 차에 해당할 수 있다.
- [0211] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 대기 트레이닝 잡의 대기 시간(qt)이 대기 트레이닝 잡이 복수의 단계로 실행됨에 따른 예상 증가 시간(t2-t1)보다 긴 경우, 대기 트레이닝 잡(1420)을 복수의 단계로 실행시킬 수 있다. 예를 들어, 트레이닝 시스템(1400)은 유휴 프로세싱 유닛을 대기 트레이닝 잡(1420)에 할당할 수 있다.
- [0212] 유휴 프로세싱 유닛(idle processing unit)은 다른 프로세싱 유닛들이 이터레이션을 마무리할 때까지 어떠한 프로그램에 의해서도 사용되지 않는 상태에 처한 프로세싱 유닛을 의미할 수 있다. 유휴 프로세싱 유닛은 클러스터의 단편화(fragmentation)에 대응할 수 있다. 예를 들어, 제1 프로세싱 유닛(GPU1)은 유휴 프로세싱 유닛에 해당할 수 있다.
- [0213] 예를 들어, 1400A 및 1400B를 참조하면, 제2 트레이닝 잡(1420)의 대기 시간(qt)은 100초이고, 제2 트레이닝 잡(1420)이 2개의 GPU(GPU1, GPU2)에 의해 1단계로 실행되는 실행 시간(t1)은 60초이고, 제2 트레이닝 잡(1425)이 1개의 GPU(GPU1)에 의해 2단계로 실행되는 실행 시간(t2)은 110초일 수 있다. 예상 증가 시간(t2-t1)은 50초일 수 있다. 이와 같이, 트레이닝 시스템(1400)의 통신 오버헤드가 감소하므로, 트레이닝 잡이 복수의 단계, 예

컨대 2단계로 실행되는 시간(ex. 110초)은 단일의 단계로 실행되는 시간(ex. 60초)의 2배 이거나, 2배보다 짧을 수 있다.

[0214] 예를 들어, 대기 시간(q_t)(ex. 100초)이 예상 증가 시간(t_2-t_1)(ex. 50초)보다 긴 경우, 트레이닝 시스템(1400)은 평균 작업 완료 시간을 줄이기 위해, 큐에 대기 중인 제2 트레이닝 잡(1420)에 제1 프로세싱 유닛(GPU1)을 할당할 수 있다. 제1 프로세싱 유닛(GPU1)이 할당된 제2 트레이닝 잡(1425)은 제1 프로세싱 유닛(GPU1)에 의해 2단계로 실행될 수 있다.

[0215] 트레이닝 시스템(1400)은 제1 트레이닝 잡(1410)과 제2 트레이닝 잡(1420)을 병렬적으로 실행시킬 수 있다. 제2 트레이닝 잡(1420)이 2단계로 실행되는 경우, 1단계로 실행되는 경우보다 작업 시간이 증가하지만, 제2 트레이닝 잡(1420)의 대기 시간(q_t)이 감소하는 정도가 더 크므로, 트레이닝 시스템(1400)의 평균 작업 완료 시간은 감소할 수 있다. 한편, 대기 시간(q_t)이 예상 증가 시간(t_2-t_1)보다 짧은 경우, 유휴 프로세싱 유닛을 대기 트레이닝 잡(1420)에 할당하면 평균 작업 완료 시간이 증가하므로, 트레이닝 시스템(1400)은 유휴 프로세싱 유닛 외에 다른 프로세싱 유닛을 확보(reserve)할 수 있다. 트레이닝 시스템(1400)은 확보된 다른 프로세싱 유닛을 대기 트레이닝 잡(1420)에 할당할 수 있다. 이에 관한 내용은 도 15에서 설명한다.

[0216] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 큐에 대기 트레이닝 잡이 존재하는 경우, 유휴 프로세싱 유닛을 이용하여 큐에 대기 중인 대기 트레이닝 잡에 필요한 자원의 개수보다 적은 자원을 할당할 수 있다. 트레이닝 시스템(1400)은 큐에 대기 중인 대기 트레이닝 잡의 대기 시간을 줄임으로써, 복수의 트레이닝 잡의 평균 작업 완료 시간을 최소화할 수 있다.

[0218] 도 15는 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡에 프로세싱 유닛을 할당하는 과정을 나타낸 일 예이다.

[0219] 도 15를 참조하여 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)에 2개의 트레이닝 잡(1510, 1520)이 제출되고, 클러스터에 4개의 GPU 자원(GPU1, GPU2, GPU3, GPU4)이 존재하는 경우를 예시하여 설명한다.

[0220] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 SRSF 방식에 따라 자원을 할당할 수 있다. 예를 들어, 제1 트레이닝 잡(1510)의 잔여 서비스 시간이 제2 트레이닝 잡(1520)의 잔여 서비스 시간보다 짧을 수 있다.

[0221] 1500A를 참조하면, 트레이닝 시스템(1400)은 제1 트레이닝 잡(1510)에 우선적으로 필요한 수만큼의 자원(예컨대, 4개의 GPU)을 할당할 수 있다. 제2 트레이닝 잡(1520)은 스케줄러의 큐에서 대기 중일 수 있다. 제2 트레이닝 잡(1520)은 대기 트레이닝 잡(queueing training job)으로 지칭될 수 있다.

[0222] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 대기 트레이닝 잡이 존재하고, 대기 트레이닝 잡의 대기 시간(q_t)이 대기 트레이닝 잡이 복수의 단계로 실행됨에 따른 예상 증가 시간(t_2-t_1)보다 짧은 경우, 자원이 기 할당된 트레이닝 잡을 제1 방식에 따라 복수의 단계로 실행할 지 여부를 추가로 고려할 수 있다. 예를 들어, 트레이닝 시스템(1400)은 자원이 기 할당된 트레이닝 잡이 복수의 단계로 실행되는 제1 방식의 평균 작업 완료 시간이, 자원이 기 할당된 트레이닝 잡이 단일의 단계로 실행되는 제2 방식의 평균 작업 완료 시간보다 짧은 경우, 기 할당된 트레이닝 잡을 복수의 단계로 실행시키도록 자원을 재할당할 수 있다. 자원이 기 할당된 트레이닝 잡은 제1 트레이닝 잡(1510)에 대응할 수 있다.

[0223] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 대기 트레이닝 잡(1520)의 대기 시간(q_t)과 제1 트레이닝 잡(1510)이 복수의 단계로 실행됨에 따른 예상 증가 시간($t_2'-t_1'$)을 비교할 수 있다. 제1 트레이닝 잡(1510)이 복수의 단계로 실행됨에 따른 예상 증가 시간($t_2'-t_1'$)은, 제1 트레이닝 잡(1510)이 복수의 단계로 실행되는 실행 시간(t_2')과 제1 트레이닝 잡(1510)이 단일의 단계로 실행되는 실행 시간(t_1')의 차에 해당할 수 있다.

[0224] 트레이닝 시스템(1400)은 대기 트레이닝 잡(1520)의 대기 시간(q_t)과 제1 트레이닝 잡(1510)이 복수의 단계로 실행됨에 따른 예상 증가 시간($t_2'-t_1'$)보다 긴 경우, 제1 트레이닝 잡(1510)을 복수의 단계로 실행시키고, 제1 트레이닝 잡(1510)에 기 할당된 프로세싱 유닛을 대기 트레이닝 잡(1520)에 재할당할 수 있다. 트레이닝 시스템(1400)은 유휴 프로세싱 유닛 외에도 기 할당되었던 프로세싱 유닛을 확보하여, 대기 트레이닝 잡(1520)에 재할당할 수 있다.

[0225] 예를 들어, 1500A를 참조하면, 대기 트레이닝 잡(1520)의 대기 시간(q_t)은 120초이고, 제1 트레이닝 잡(1510)이 4개의 GPU(GPU1, GPU2, GPU3, GPU4)에 의해 1단계로 실행되는 실행 시간(t_1')은 120초이고, 제1 트레이닝 잡(1515)이 2개의 GPU(GPU1, GPU2)에 의해 2단계로 실행되는 실행 시간(t_2')은 200초일 수 있다. 예상 증가 시간

($t_2' - t_1'$)은 80초일 수 있다.

- [0226] 예를 들어, 대기 시간(q_t)(ex. 120초)이 예상 증가 시간($t_2' - t_1'$)(ex. 80초)보다 긴 경우, 트레이닝 시스템(1400)은 평균 작업 완료 시간을 줄이기 위해, 제1 트레이닝 잡(1510)에 기 할당된 4개의 GPU 중 2개의 GPU(예컨대, GPU1, GPU2)를 가용 프로세싱 유닛으로 확보할 수 있다. 제1 트레이닝 잡(1515)은 기 할당된 4개의 GPU 중 나머지 2개의 GPU(예컨대, GPU3, GPU4)에 의해 2단계로 실행될 수 있다.
- [0227] 1500B를 참조하면, 트레이닝 시스템(1400)은 큐에 대기 중인 대기 트레이닝 잡(1520)에 확보된 2개의 GPU(예컨대, GPU1, GPU2)를 할당할 수 있다. 대기 트레이닝 잡(1525)에는 2개의 GPU(예컨대 GPU1, GPU2)가 할당될 수 있다.
- [0228] 트레이닝 시스템(1400)은 자원이 기 할당된 트레이닝 잡의 실행이 완료되고 나서 대기 트레이닝 잡을 실행하는 대신, 자원이 기 할당된 트레이닝 잡으로부터 확보한 자원을 이용하여 자원이 기 할당된 트레이닝 잡과 대기 트레이닝 잡을 병렬적으로 실행할 수 있다. 제1 트레이닝 잡(1510)이 2단계로 실행되는 경우, 1단계로 실행되는 경우보다 작업 시간이 증가하지만, 대기 트레이닝 잡(1520)의 대기 시간이 감소하는 정도가 더 크므로, 트레이닝 시스템(1400)의 평균 작업 완료 시간은 감소할 수 있다.
- [0229] 본 개시의 일 실시예에 따른 트레이닝 시스템(1400)은 제1 트레이닝 잡에 기 할당된 프로세싱 유닛을 대기 트레이닝 잡에 재할당함으로써, 큐에 대기 중인 대기 트레이닝 잡의 대기 시간을 줄일 수 있다. 이에 따라, 복수의 트레이닝 잡의 평균 작업 완료 시간을 최소화할 수 있다.
- [0231] 도 16은 일 실시예에 따른 트레이닝 시스템에서, 복수의 트레이닝 잡에 프로세싱 유닛을 할당하는 동작 방법을 나타낸 흐름도이다.
- [0232] 도 16을 참조하면, 동작 S1610에서, 트레이닝 시스템(1400)은 자원 관리자를 통해 복수의 트레이닝 잡 각각에 프로세싱 유닛을 할당할 수 있다. 트레이닝 시스템(1400)은 SRSF 방식에 따라 자원을 할당할 수 있다. 트레이닝 시스템(1400)은 최대-최소 공평성 방식에 따라 자원을 할당할 수 있다. 트레이닝 시스템(1400)은 도 13의 동작 S1310와 같이 최대-최소 공평성 방식과 SRSF 방식을 동시에 고려하여 자원을 할당할 수 있다.
- [0233] 동작 S1620에서, 트레이닝 시스템(1400)은 대기 트레이닝 잡이 존재하는 지 여부를 식별할 수 있다. 대기 트레이닝 잡은 자원이 할당되지 않고 큐에 저장되어 대기 중인 트레이닝 잡을 나타낼 수 있다. 대기 트레이닝 잡은, 자원이 스케줄링되고 처리될 때까지 큐에서 대기할 수 있다.
- [0234] 트레이닝 시스템(1400)은 대기 트레이닝 잡이 존재하는 경우, 자원 할당 동작, 예컨대 동작 S1630 내지 S1660을 수행할 수 있다. 트레이닝 시스템(1400)은 대기 트레이닝 잡이 존재하지 않는 경우, 자원 할당 동작을 종료할 수 있다.
- [0235] 일 실시예에서, 트레이닝 시스템(1400)은 트레이닝 잡이 복수의 단계로 실행되는 제1 방식과 트레이닝 잡이 단일의 단계로 실행되는 제2 방식을 비교할 때, 대기 트레이닝 잡만을 고려할 수 있다(동작 S1630 및 동작 S1640). 관련된 예시는 도 14에서 설명하였다.
- [0236] 일 실시예에서, 트레이닝 시스템(1400)은 트레이닝 잡이 복수의 단계로 실행되는 제1 방식과 트레이닝 잡이 단일의 단계로 실행되는 제2 방식을 비교할 때, 대기 트레이닝 잡과 자원이 기 할당된 트레이닝 잡을 고려할 수 있다(동작 S1650 및 동작 S1660). 관련된 예시는 도 15에서 설명하였다.
- [0237] 동작 S1630에서, 트레이닝 시스템(1400)은 대기 트레이닝 잡이 존재함에 따라, 대기 트레이닝 잡의 대기 시간(q_t)과 대기 트레이닝 잡이 복수의 단계로 실행됨에 따른 예상 증가 시간($t_2 - t_1$)을 비교함으로써, 대기 트레이닝 잡을 복수의 단계로 실행시킬 지 여부를 결정할 수 있다.
- [0238] 동작 S1640에서, 트레이닝 시스템(1400)은 대기 트레이닝 잡의 대기 시간(q_t)이 대기 트레이닝 잡이 복수의 단계로 실행됨에 따른 예상 증가 시간($t_2 - t_1$)보다 긴 경우, 대기 트레이닝 잡을 복수의 단계로 실행시킬 수 있다. 예를 들어, 트레이닝 시스템(1400)은 유휴 프로세싱 유닛을 대기 트레이닝 잡에 할당할 수 있다.
- [0239] 대기 트레이닝 잡이 복수의 단계로 실행되는 경우, 복수의 트레이닝 잡의 평균 작업 완료 시간은 감소할 수 있다.
- [0240] 한편, 트레이닝 시스템(1400)은 대기 시간(q_t)이 예상 증가 시간($t_2 - t_1$)보다 짧은 경우, 트레이닝 시스템(1400)

0)은 유희 프로세싱 유닛 외에 다른 프로세싱 유닛을 확보할 수 있다. 예를 들어, 트레이닝 시스템(1400)은 어느 하나의 트레이닝 잡에 기 할당된 프로세싱 유닛을 이용하여 대기 트레이닝 잡에 재할당할 수도 있다. 트레이닝 시스템(1400)은 대기 트레이닝 잡 뿐 아니라 자원이 기 할당된 트레이닝 잡까지 고려하여 해당 트레이닝 잡을 복수의 단계로 실행시킬 지 여부를 결정할 수 있다. 이에 관해서는 동작 S1650에서 설명한다.

[0241] 동작 S1650에서, 트레이닝 시스템(1400)은 대기 트레이닝 잡의 대기 시간(q_t)이 대기 트레이닝 잡이 복수의 단계로 실행됨에 따른 예상 증가 시간(t_2-t_1)보다 짧은 경우, 대기 트레이닝 잡의 대기 시간(q_t)과 프로세싱 유닛이 기 할당된 트레이닝 잡이 복수의 단계로 실행됨에 따른 예상 증가 시간($t_2'-t_1'$)을 비교할 수 있다. 도 16에서, 자원이 기 할당된 트레이닝 잡을 제1 트레이닝 잡으로 지칭한다.

[0242] 트레이닝 시스템(1400)은 큐에 저장된 트레이닝 시스템(1400)은 대기 시간(q_t)과 예상 증가 시간($t_2'-t_1'$)을 비교함으로써, 제1 트레이닝 잡을 복수의 단계로 실행시킬 지 여부를 결정할 수 있다.

[0243] 동작 S1660에서, 트레이닝 시스템(1400)은 대기 트레이닝 잡의 대기 시간(q_t)이 예상 증가 시간($t_2'-t_1'$)보다 긴 경우, 제1 트레이닝 잡을 복수의 단계로 실행시킬 수 있다.

[0244] 예를 들어, 트레이닝 시스템(1400)은 제1 트레이닝 잡에 기 할당된 프로세싱 유닛 중 적어도 하나를 대기 트레이닝 잡에 재할당할 수 있다. 트레이닝 시스템(1400)은 제1 트레이닝 잡에 기 할당된 프로세싱 유닛을 확보하여, 대기 트레이닝 잡에 재할당할 수 있다. 이에 따라, 제1 트레이닝 잡에는 필요한 자원보다 적은 개수의 자원이 할당되며, 제1 트레이닝 잡은 복수의 단계로 실행될 수 있다. 또는 트레이닝 시스템(1400)은 유희 프로세싱 유닛을 대기 트레이닝 잡에 할당할 수도 있다.

[0245] 한편, 대기 시간(q_t)이 예상 증가 시간($t_2'-t_1'$)보다 짧은 경우, 트레이닝 시스템(1400)은 제1 트레이닝 잡에 기 할당된 프로세싱 유닛을 대기 트레이닝 잡에 재할당하지 않을 수 있다.

[0247] 도 17은 일 실시예에 따른 트레이닝 시스템의 블록도이다.

[0248] 도 17을 참조하면, 일 실시예에 따른 트레이닝 시스템(1700)은 통신 인터페이스(1710), 프로세서(1730) 및 메모리(1750)를 포함할 수 있다. 통신 인터페이스(1710), 프로세서(1730) 및 메모리(1750)는 통신 버스(1705)를 통해 서로 통신할 수 있다. 일 실시예에 따른 트레이닝 시스템(1700)은 트레이닝 시스템(100), 트레이닝 시스템(1100), 및 트레이닝 시스템(1400) 각각에 대응할 수 있다.

[0249] 통신 인터페이스(1710)는 사용자로부터 트레이닝 데이터를 포함하는 트레이닝 요청을 수신할 수 있다.

[0250] 프로세서(1730)는 트레이닝 요청에 응답하여 트레이닝을 위한 잡을, 신경망 모델의 복수의 논리 작업자에 의해 수행되는 복수의 마이크로서비스로 분할할 수 있다. 프로세서(1730)는 그래픽 프로세싱 유닛의 가용 상태에 기초하여, 복수의 마이크로서비스를 복수의 그래픽 프로세싱 유닛에 동적으로 할당할 수 있다. 프로세서(1730)는 클러스터의 자원 상황을 고려하여, 트레이닝 잡을 실행하기 위해 필요한 논리 작업자의 수보다 가용 상태의 프로세싱 유닛의 수가 적은 경우, 동일한 프로세싱 유닛에 복수의 마이크로서비스를 스케줄링할 수 있다. 프로세서(1730)는 스케줄링을 기초로, 복수의 논리 작업자에 의해 신경망 모델을 트레이닝할 수 있다.

[0251] 프로세서(1730)는 신경망 모델에 대응하는 트레이닝 잡(training job)을 복수의 논리 작업자에 의해 실행되는 복수의 마이크로서비스(a plurality of microservices)로 분할한다. 프로세서(1730)는 복수의 마이크로서비스를 복수의 프로세싱 유닛(a plurality of processing units)에 스케줄링한다. 프로세서(1730)는 상기 복수의 프로세싱 유닛의 가용 상태에 기초하여, 상기 복수의 제1 마이크로서비스 및 상기 복수의 제2 마이크로서비스를 상기 복수의 프로세싱 유닛 중 어느 하나의 프로세싱 유닛에 스케줄링한다.

[0252] 프로세서(1730)는 가용 프로세싱 유닛(available processing unit)의 개수가 상기 복수의 논리 작업자의 개수보다 적음에 따라, 상기 복수의 제1 마이크로서비스 및 상기 복수의 제2 마이크로서비스를 상기 어느 하나의 프로세싱 유닛에 순차적으로 스케줄링할 수 있다.

[0253] 프로세서(1730)는 상기 복수의 제1 마이크로서비스와 상기 복수의 제2 마이크로서비스를 동일한 컨테이너(container)에 스케줄링할 수 있다.

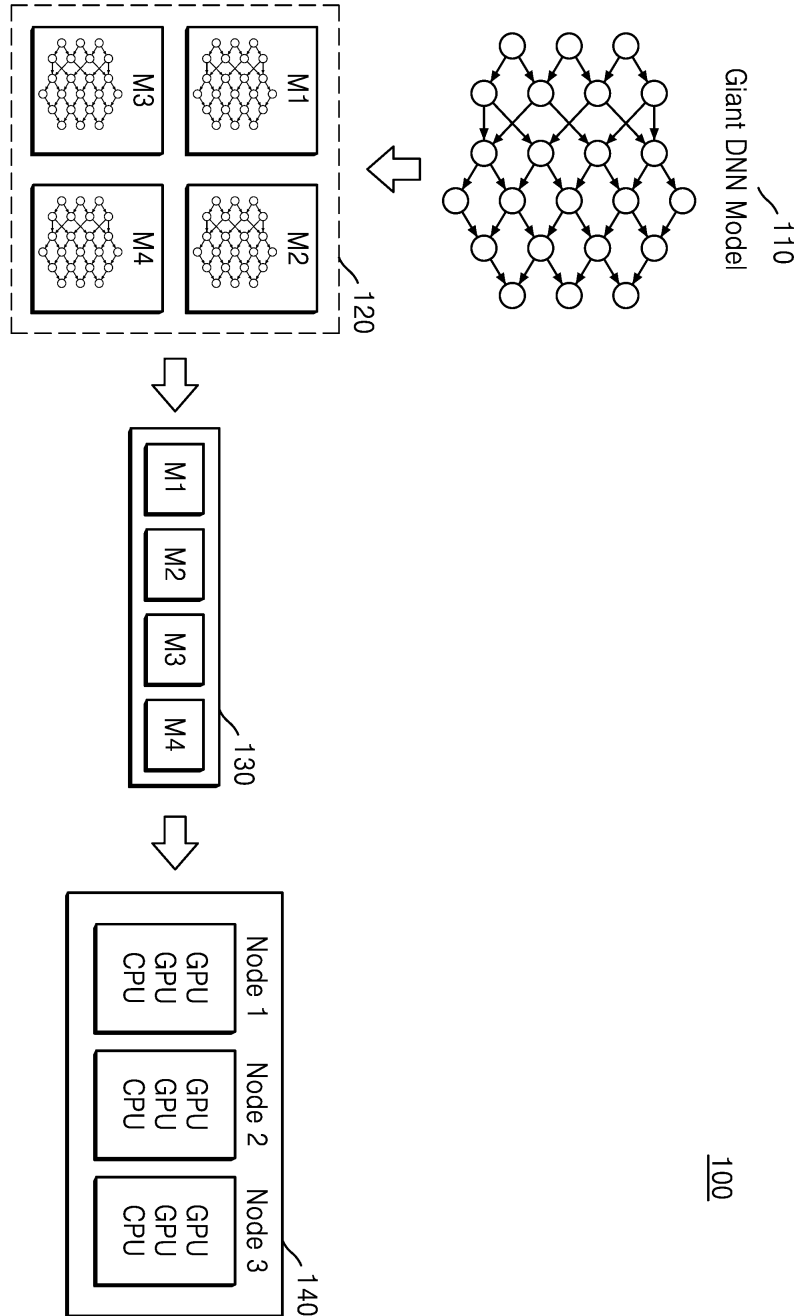
[0254] 프로세서(1730)는 상기 어느 하나의 프로세싱 유닛에서 상기 복수의 제1 마이크로서비스 중 어느 하나의 미니 배치 처리와 상기 복수의 제2 마이크로서비스 중 어느 하나의 미니 배치 처리가 복수의 단계(multiple phases)로 실행되도록 스케줄링할 수 있다.

- [0255] 프로세서(1730)는 상기 신경망 모델에 대응하는 상기 트레이닝 잡에 상기 복수의 프로세싱 유닛을 2^n 단위로 할당할 수 있다.
- [0256] 프로세서(1730)는 상기 신경망 모델에 대응하는 상기 트레이닝 잡이 2k개의 논리 작업자를 포함하는 경우, 상기 복수의 프로세싱 유닛을 2k의 약수 중 어느 하나의 단위로 할당할 수 있다. 프로세서(1730)는 상기 트레이닝 잡이 2k-1개의 논리 작업자를 포함하는 경우, 상기 복수의 프로세싱 유닛을 2k-1의 약수 중 어느 하나 또는 2k의 약수 중 1과 2k를 제외한 어느 하나의 단위로 할당할 수 있다.
- [0257] 프로세서(1730)는 최단 잔여 서비스 시간을 갖는 트레이닝 잡부터 순서대로 클러스터에 존재하는 프로세싱 유닛을 할당할 수 있다. 프로세서(1730)는 상기 클러스터에 잔여 프로세싱 유닛(remaining processing unit)이 존재함에 따라, 필요한 프로세싱 유닛의 수보다 적은 수의 프로세싱 유닛이 할당된 트레이닝 잡에 상기 잔여 프로세싱 유닛을 할당할 수 있다.
- [0258] 프로세서(1730)는 복수의 트레이닝 잡 각각에 프로세싱 유닛을 할당할 수 있다. 프로세서(1730)는 큐(queue)에 저장된 대기 트레이닝 잡(queueing training job)이 존재하는 지를 식별할 수 있다. 프로세서(1730)는 대기 트레이닝 잡이 존재함에 따라, 대기 트레이닝 잡의 대기 시간(queueing time)이 하나 이상의 트레이닝 잡이 복수의 단계(multiple phases)로 실행됨에 따른 예상 증가 시간(expected increase time)보다 긴 경우, 대기 트레이닝 잡에 상기 프로세싱 유닛을 재할당할 수 있다. 하나 이상의 트레이닝 잡은 대기 트레이닝 잡 및 대기 트레이닝 잡과 프로세싱 유닛이 기 할당된 제1 트레이닝 잡 중 적어도 하나를 포함할 수 있다.
- [0259] 예를 들어, 프로세서(1730)는 대기 트레이닝 잡의 대기 시간과 상기 대기 트레이닝 잡이 복수의 단계로 실행됨에 따른 예상 증가 시간을 비교함으로써, 유휴 프로세싱 유닛을 상기 대기 트레이닝 잡에 할당할 지 여부를 결정할 수 있다.
- [0260] 예를 들어, 프로세서(1730)는 대기 트레이닝 잡의 대기 시간과 프로세싱 유닛이 기 할당된 제1 트레이닝 잡이 복수의 단계로 실행됨에 따른 예상 증가 시간과 비교함으로써, 제1 트레이닝 잡에 기 할당된 프로세싱 유닛을 상기 대기 트레이닝 잡에 재할당할 지 여부를 결정할 수 있다.
- [0261] 메모리(1750)는 상술한 프로세서(1730)의 처리 과정에서 생성되는 다양한 정보들을 저장할 수 있다. 이 밖에도, 메모리(1750)는 각종 데이터와 프로그램 등을 저장할 수 있다. 메모리(1750)는 휘발성 메모리 또는 비휘발성 메모리를 포함할 수 있다. 메모리(1750)는 하드 디스크 등과 같은 대용량 저장 매체를 구비하여 각종 데이터를 저장할 수 있다.
- [0262] 이 밖에도, 프로세서(1730)는 도 1 내지 도 16을 통해 전술한 적어도 하나의 방법 또는 적어도 하나의 방법에 대응되는 알고리즘을 수행할 수 있다. 프로세서(1730)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.
- [0263] 프로세서(1730)는 프로그램을 실행하고, 트레이닝 시스템(1700)을 제어할 수 있다. 프로세서(1730)에 의하여 실행되는 프로그램 코드는 메모리(1750)에 저장될 수 있다. 프로세서(1730)는 예를 들어, CPU(Central Processing Unit), GPU(Graphics Processing Unit), 또는 NPU(Neural network Processing Unit)으로 구성될 수 있다.
- [0264] 기기로 읽을 수 있는 저장매체는, 비일시적(non-transitory) 저장매체의 형태로 제공될 수 있다. 여기서, '비일시적 저장매체'는 실재(tangible)하는 장치이고, 신호(signal)(예: 전자기파)를 포함하지 않는다는 것을 의미할 뿐이며, 이 용어는 데이터가 저장매체에 반영구적으로 저장되는 경우와 임시적으로 저장되는 경우를 구분하지 않는다. 예로, '비일시적 저장매체'는 데이터가 임시적으로 저장되는 버퍼를 포함할 수 있다.
- [0265] 일 실시예에 따르면, 본 문서에 개시된 다양한 실시예들에 따른 방법은 컴퓨터 프로그램 제품(computer program product)에 포함되어 제공될 수 있다. 컴퓨터 프로그램 제품은 상품으로서 판매자 및 구매자 간에 거래될 수 있다. 컴퓨터 프로그램 제품은 기기로 읽을 수 있는 저장 매체(예: compact disc read only memory (CD-ROM))의 형태로 배포되거나, 또는 어플리케이션 스토어를 통해 또는 두개의 사용자 장치들(예: 스마트폰들) 간에 직접, 온라인으로 배포(예: 다운로드 또는 업로드)될 수 있다. 온라인 배포의 경우에, 컴퓨터 프로그램 제품(예: 다운

로더블 앱(downloadable app))의 적어도 일부는 제조사의 서버, 어플리케이션 스토어의 서버, 또는 중계 서버의 메모리와 같은 기기로 읽을 수 있는 저장 매체에 적어도 일시 저장되거나, 임시적으로 생성될 수 있다.

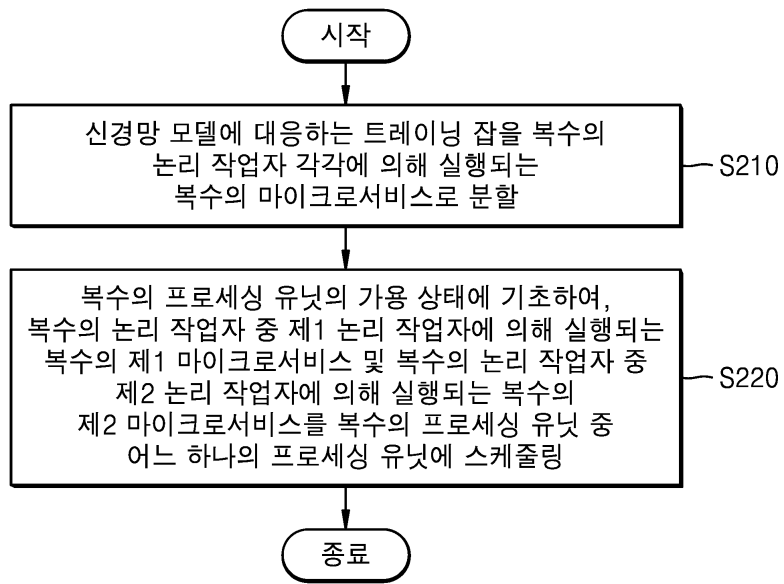
도면

도면1

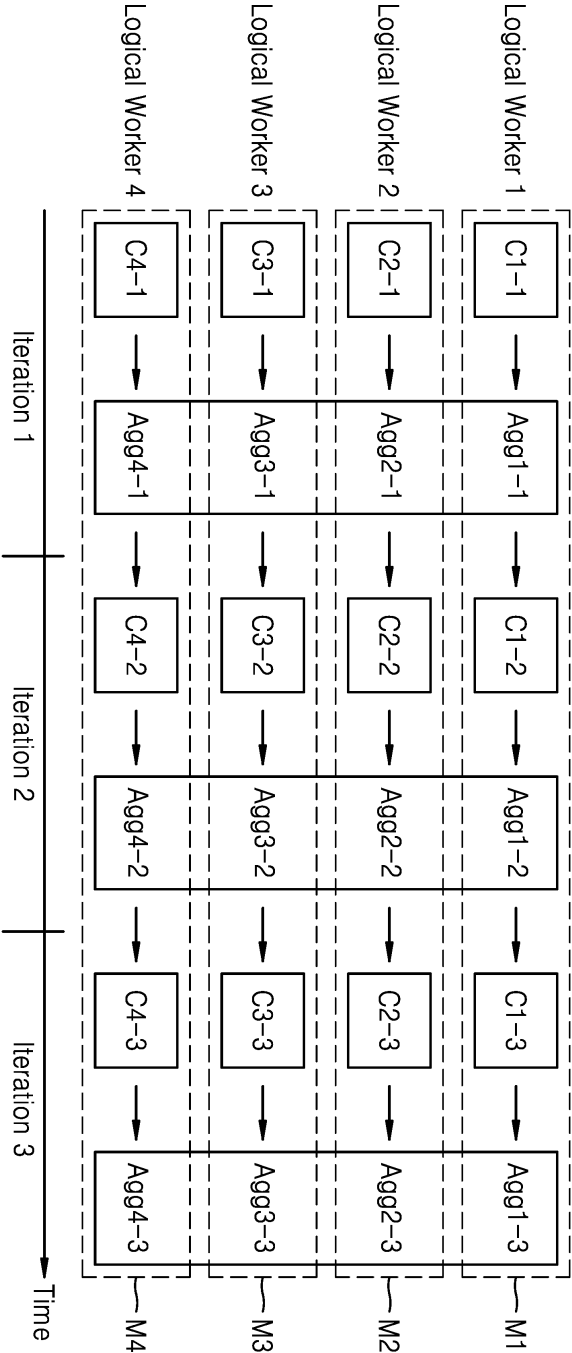


100

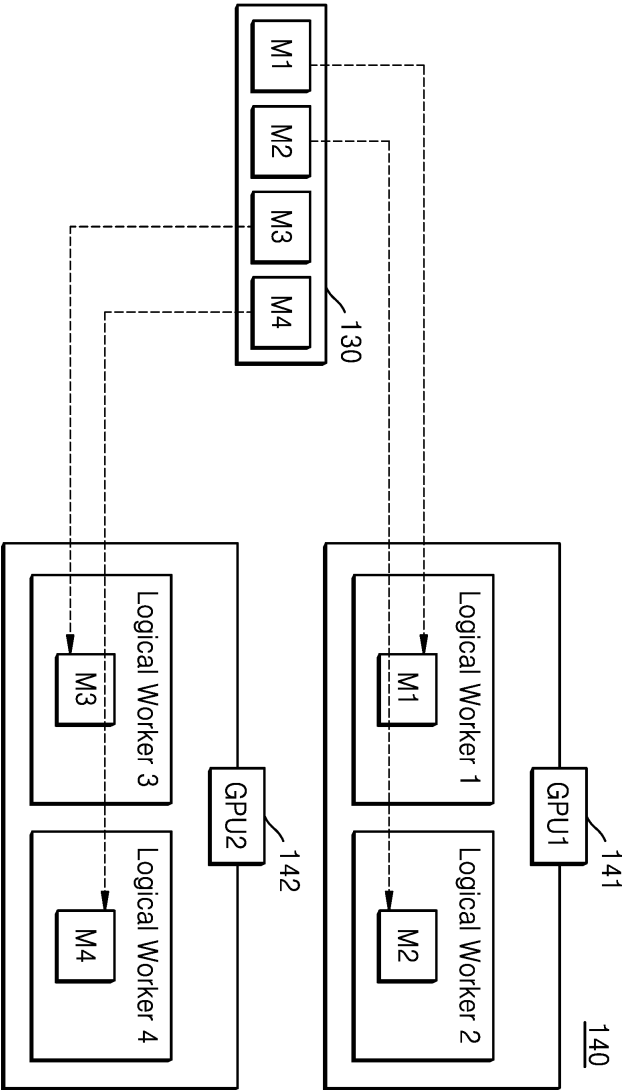
도면2



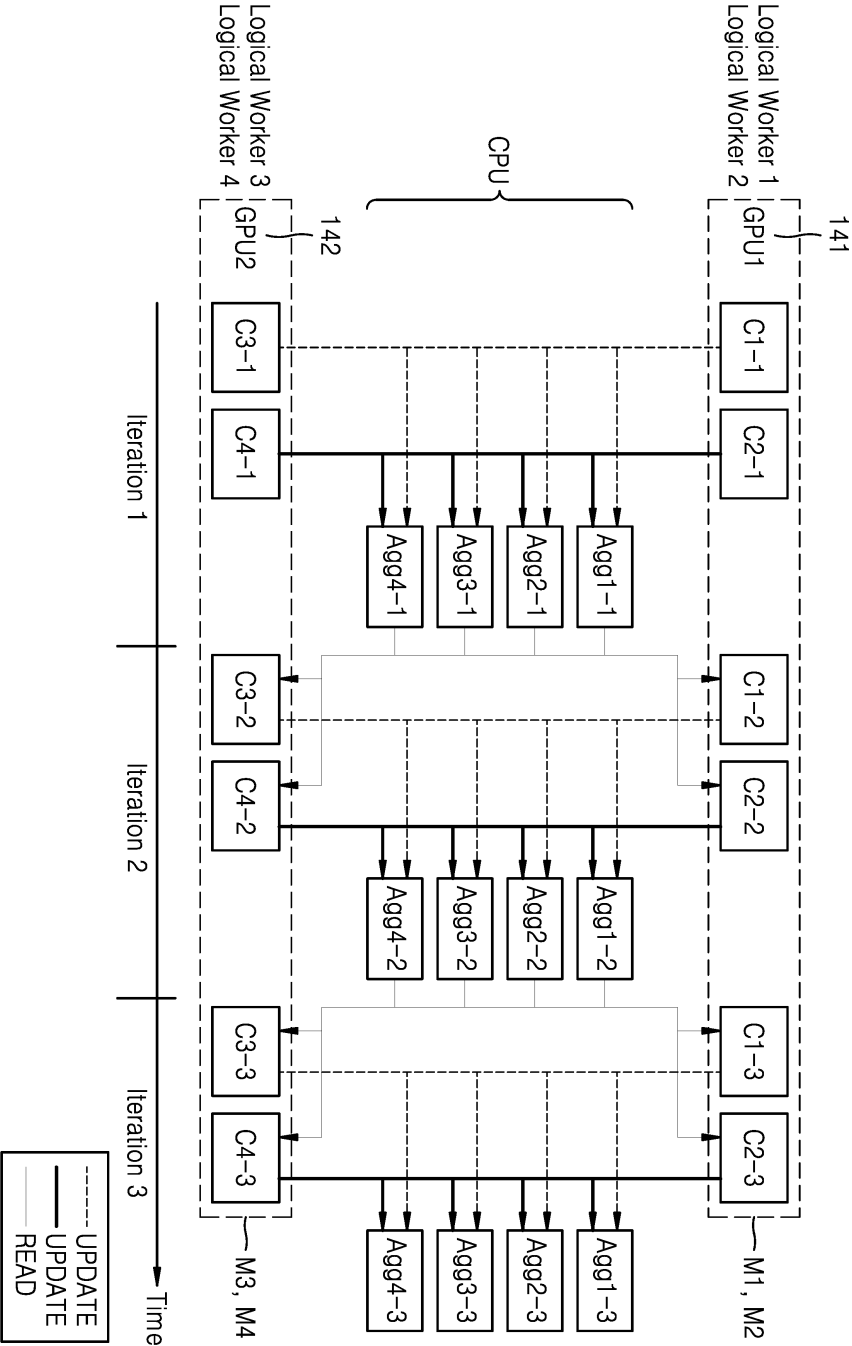
도면3



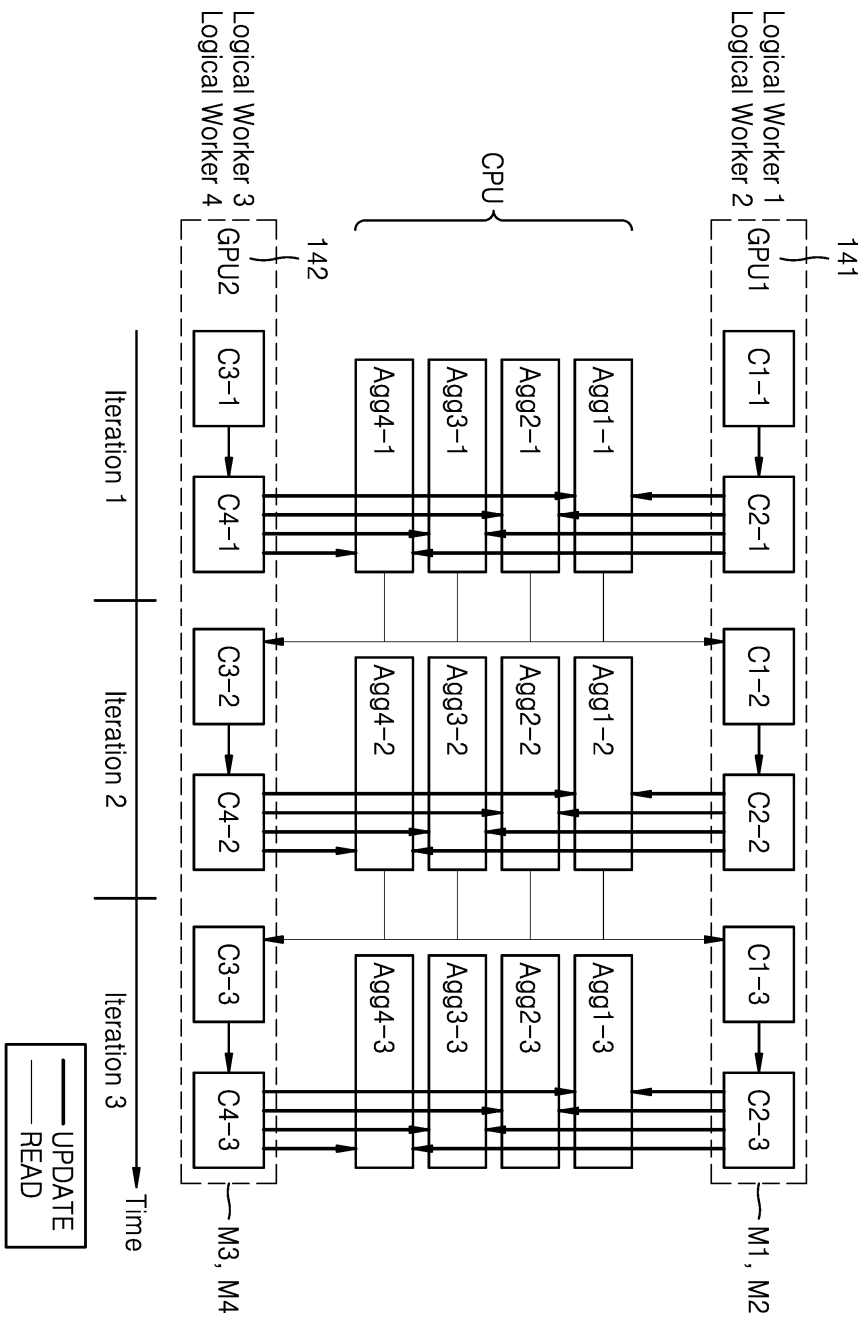
도면4



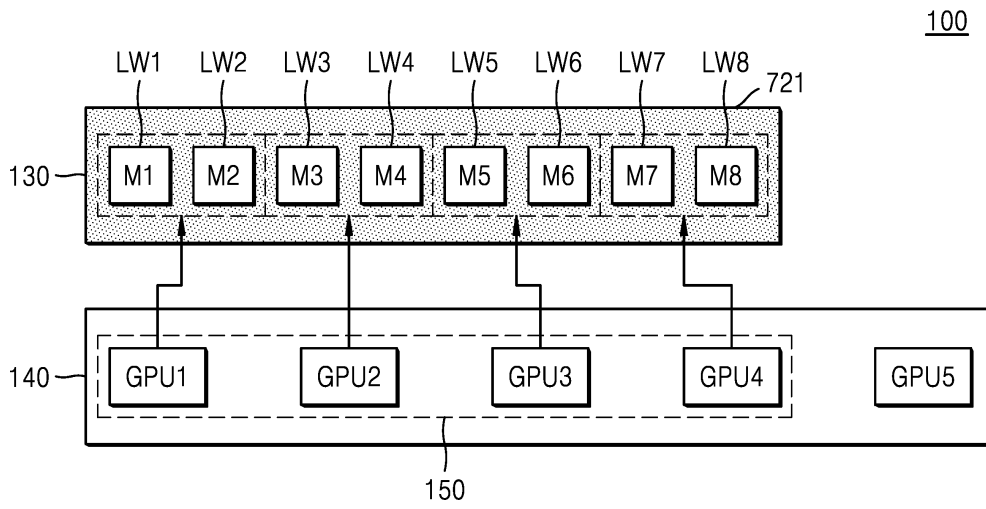
도면5



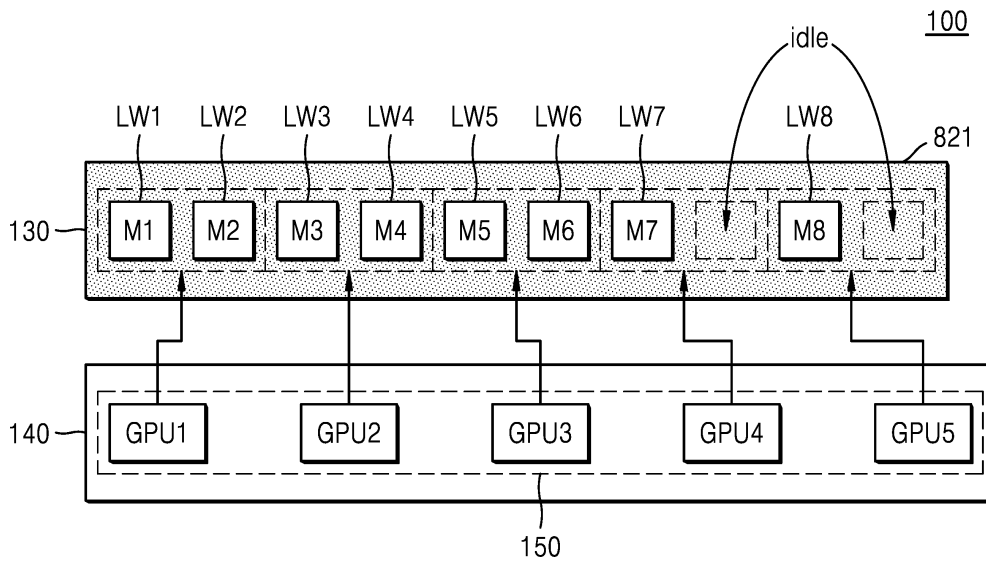
도면6



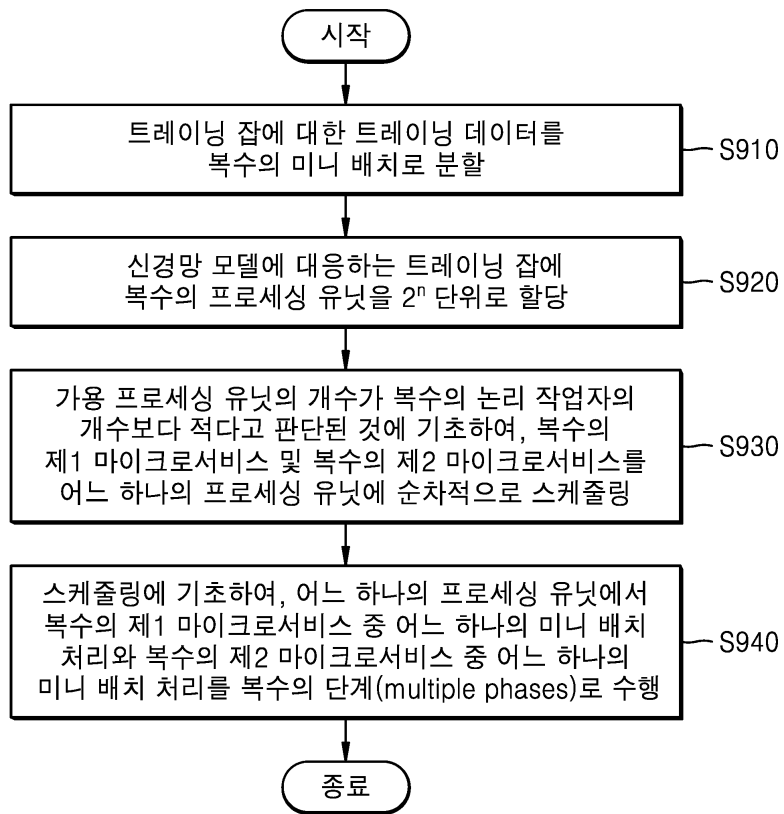
도면7



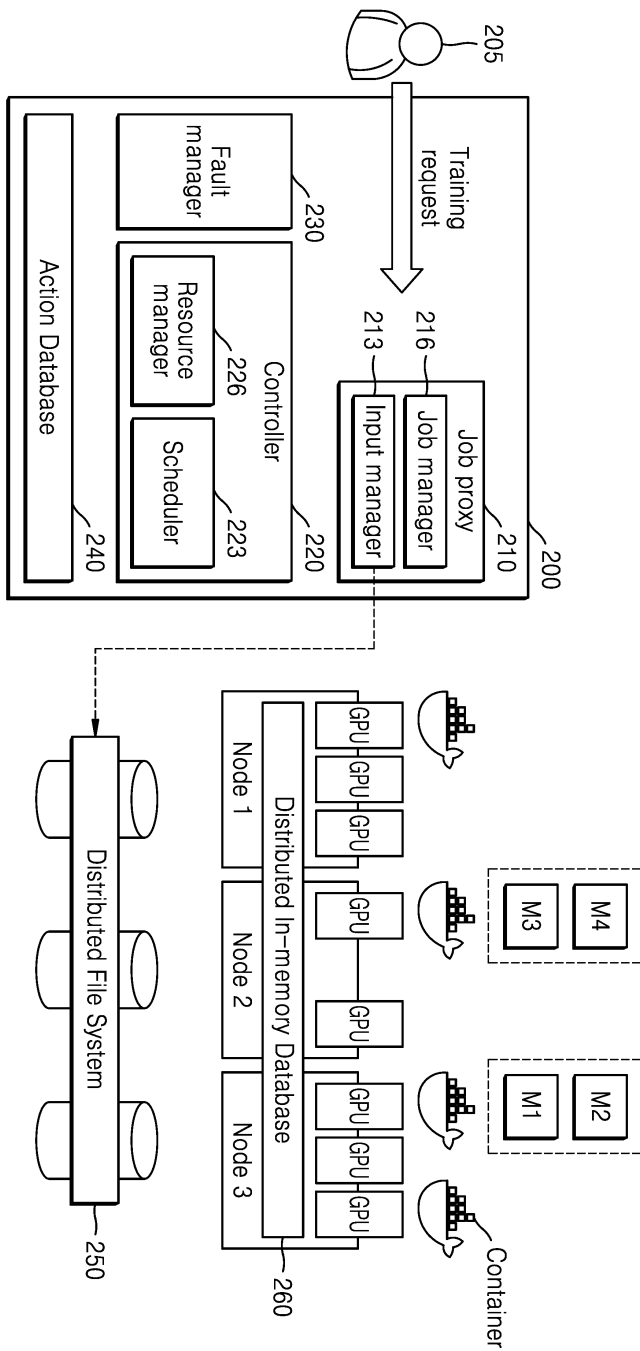
도면8



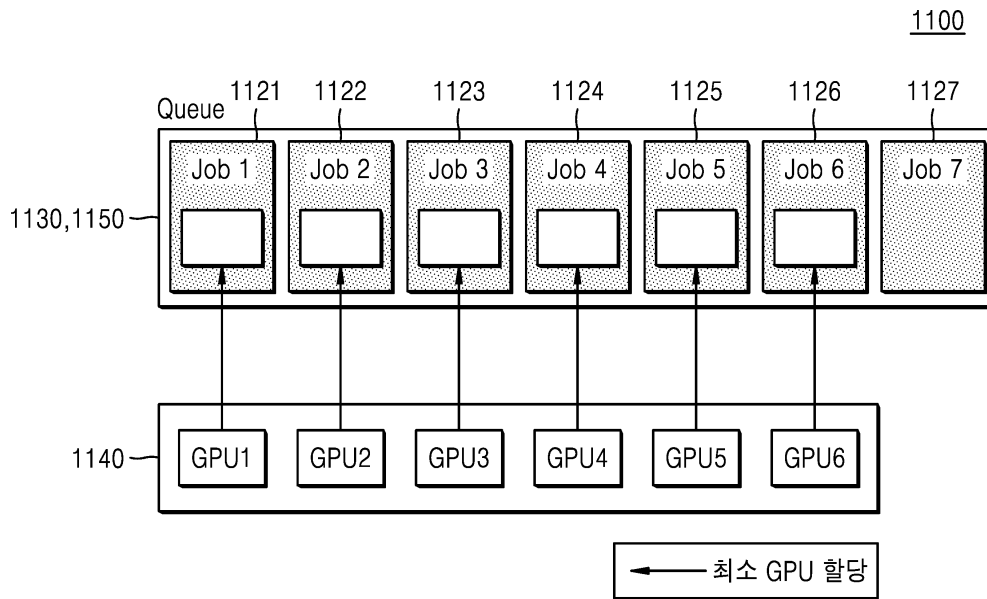
도면9



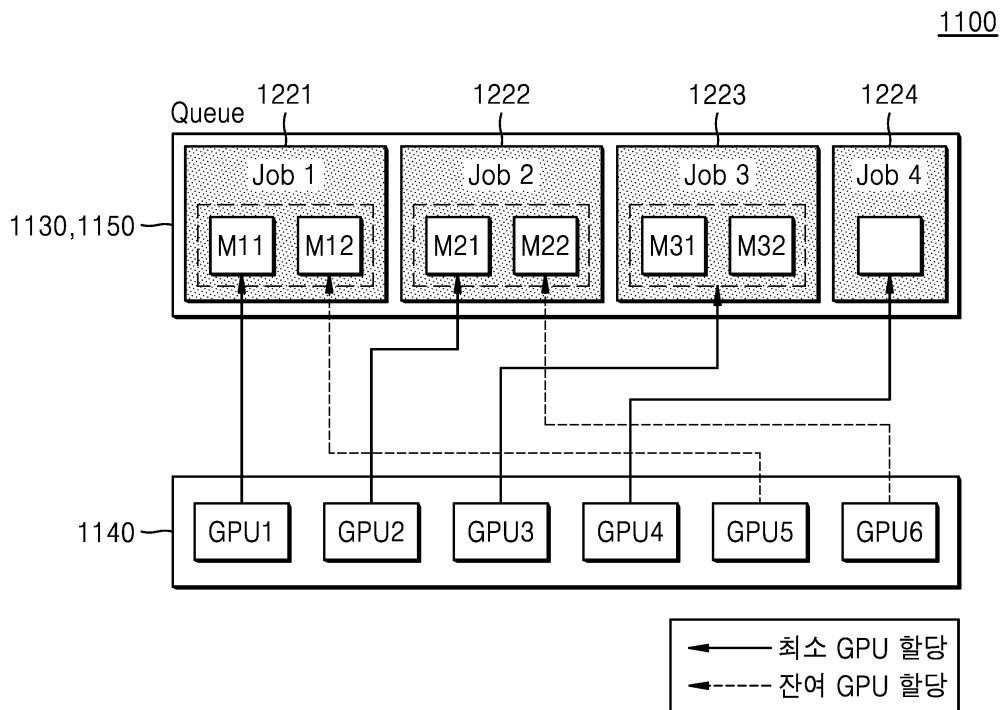
도면 10



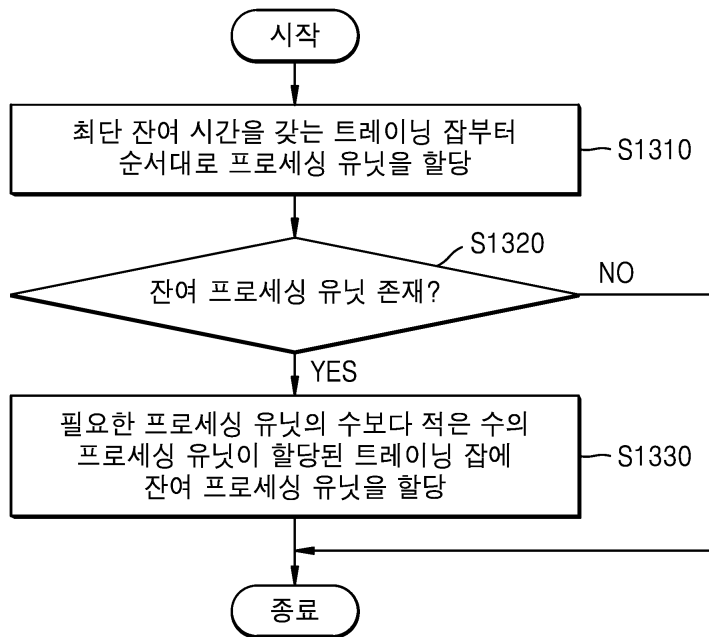
도면11



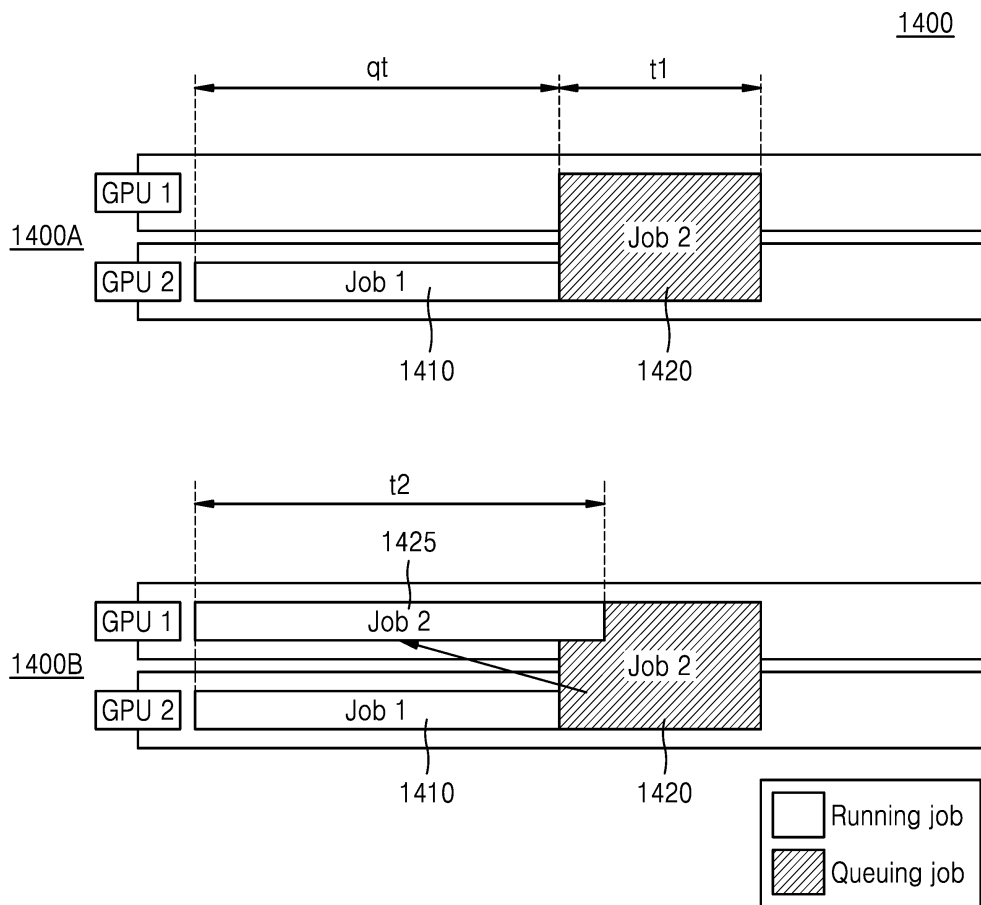
도면12



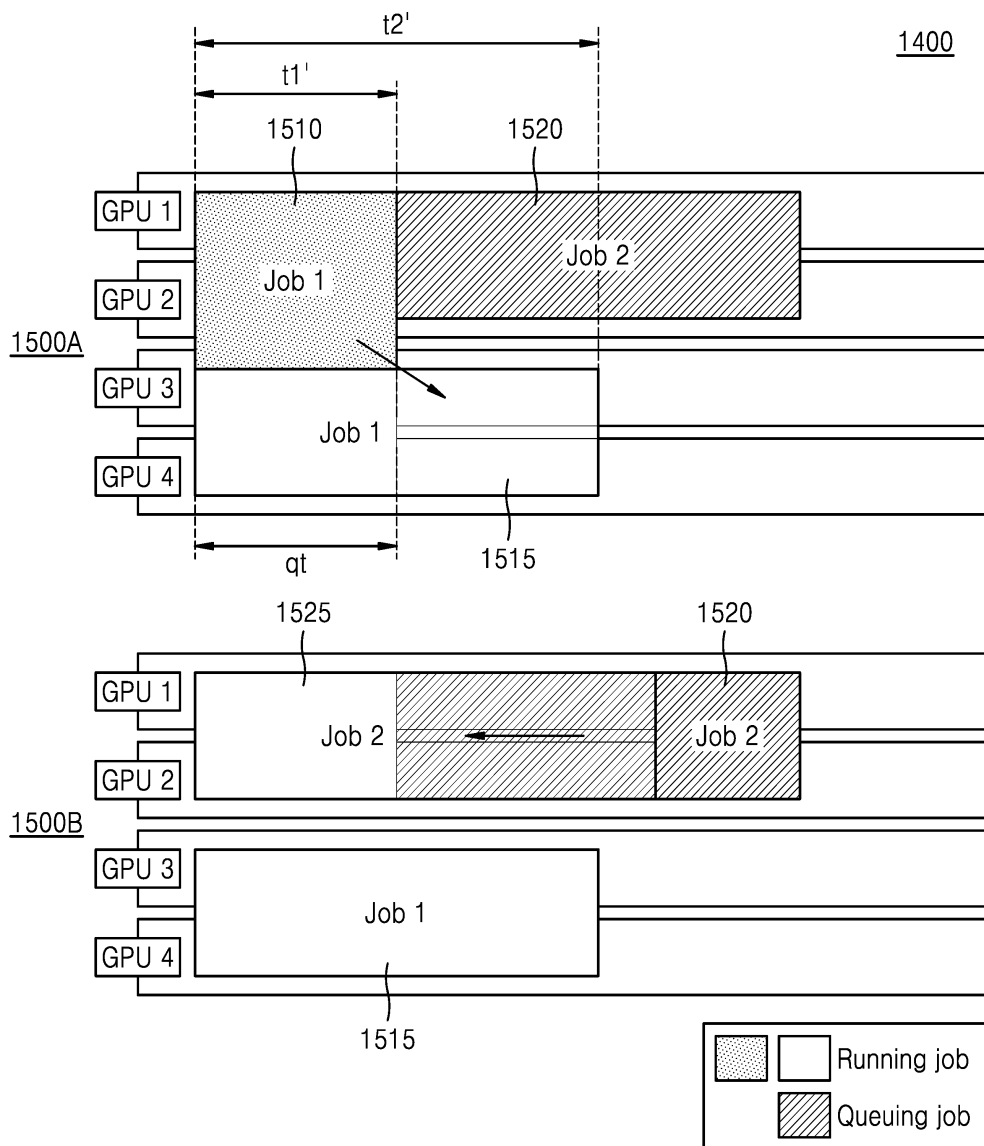
도면13



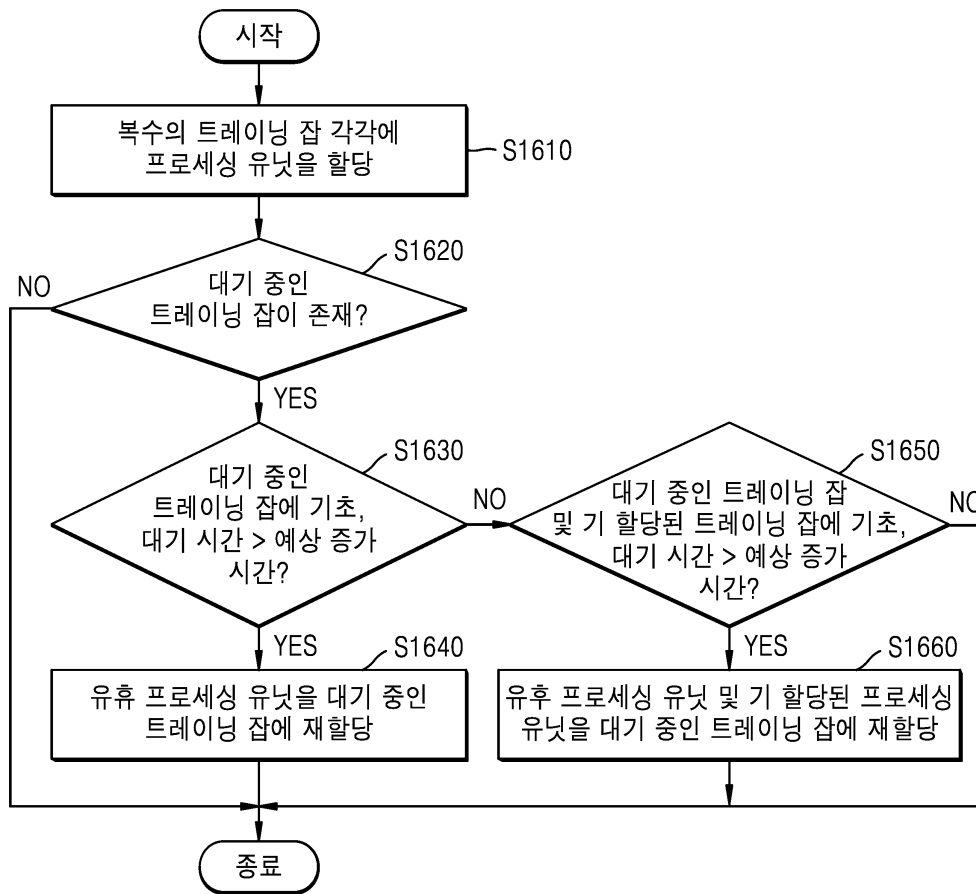
도면14



도면 15



도면16



도면17

