(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년11월03일
(11) 등록번호 10-2463147
(24) 등록일자 2022년11월01일

(51) 국제특허분류(Int. Cl.)
G06N 3/08 (2006.01) *G06N 3/04* (2006.01)
(52) CPC특허분류
G06N 3/084 (2013.01)
G06N 3/0454 (2013.01)
(21) 출원번호 10-2021-0026460
(22) 출원일자 2021년02월26일
심사청구일자 2021년02월26일
(65) 공개번호 10-2022-0122175
(43) 공개일자 2022년09월02일
(56) 선행기술조사문헌
육동석 등., 심층 신경망 병렬 학습 방법 연구 동향, 한국음향학회지 제39권 제6호, 505-514pages (2020.)*
Shuxin Zheng et al., Asynchronous Stochastic Gradient Descent with Delay Compensation, arXiv:1609.08326v6, 1-20pages (2020. 2. 18.)*
Sixin Zhang et al., Deep learning with Elastic Averaging SGD, arXiv:1412.6651v8, 1-24pages (2015. 10. 25.)*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
육동석
경기도 남양주시 천마산로 65, 104동 2004호(호평동, 호평 파라곤)
유인철
서울특별시 성북구 인촌로7마길 7, 201호(안암동1가)
(뒷면에 계속)
(74) 대리인
송인호, 윤형근, 최관락

전체 청구항 수 : 총 5 항 심사관 : 양대경

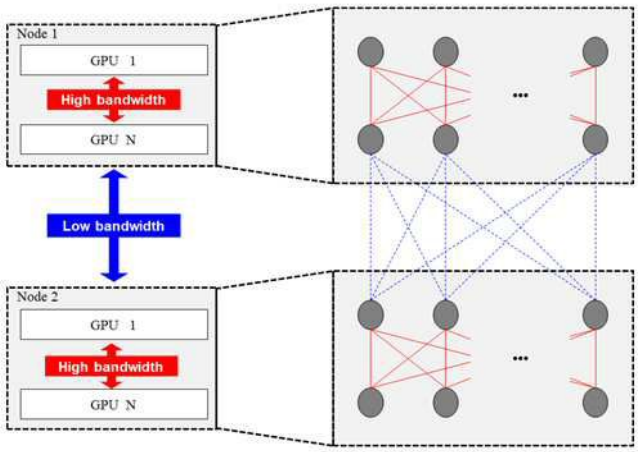
(54) 발명의 명칭 **초병렬 심층 학습 방법 및 장치**

(57) 요 약

본 발명은 범용 컴퓨터 클러스터에서의 초병렬 심층 학습 방법 및 장치를 개시한다. 복수의 컴퓨팅 노드 각각에 연산 수행을 위한 서로 다른 모델을 할당하는 단계; 상기 복수의 컴퓨팅 노드 각각이 순전파를 통해 입력 데이터, 파라미터와 상기 모델을 이용하여 계산한 출력값을 다음 컴퓨팅 노드로 전달하는 단계; 복수의 컴퓨팅

(뒷면에 계속)

대 표 도 - 도1



노드 각각이 상기 입력 데이터 및 상기 파라미터의 사본을 오류 없는 역전파 계산을 위해 유지하는 단계; 상기 복수의 컴퓨팅 노드 각각이 역전파를 통해 입력된 로스(loss), 상기 입력 데이터, 상기 파라미터를 이용하여 계산한 로스를 다음 컴퓨팅 노드로 전달하는 단계; 및 상기 복수의 컴퓨팅 노드 각각이 상기 입력된 로스를 이용하여 그라디언트를 계산하고 지연되지 않은 시간대의 그라디언트를 예측하거나, 미리 설정된 통신 주기에 따라 탄성력 기반으로 동기화하여 상기 파라미터를 비동기적으로 갱신하는 단계를 포함하는 초병렬 심층 학습 방법이 제공된다.

(52) CPC특허분류
G06N 3/049 (2013.01)

(72) 발명자
장봉원
서울특별시 서초구 명달로6길 31, 102동 405호(서초동, 서초1차이편한세상)

최석환
서울특별시 송파구 충민로6길 14, 607동 1407호(장지동, 송파파인타운6단지)

공지예외적용 : 있음

명 세 서

청구범위

청구항 1

범용 컴퓨터 클러스터에서의 초병렬 심층 학습 방법으로서,

복수의 컴퓨팅 노드 각각에 연산 수행을 위한 서로 다른 모델을 할당하는 단계;

상기 복수의 컴퓨팅 노드 각각이 순전파를 통해 입력 데이터, 파라미터와 상기 모델을 이용하여 계산한 출력값을 다음 컴퓨팅 노드로 전달하는 단계;

복수의 컴퓨팅 노드 각각이 상기 입력 데이터 및 상기 파라미터의 사본을 오류 없는 역전파 계산을 위해 유지하는 단계;

상기 복수의 컴퓨팅 노드 각각이 역전파를 통해 입력된 로스(loss), 상기 입력 데이터, 상기 파라미터를 이용하여 계산한 로스를 다음 컴퓨팅 노드로 전달하는 단계; 및

상기 복수의 컴퓨팅 노드 각각이 상기 입력된 로스를 이용하여 그라디언트를 계산하고 지연되지 않은 시간대의 그라디언트를 예측하거나, 미리 설정된 통신 주기에 따라 탄성력 기반으로 동기화하여 상기 파라미터를 비동기적으로 갱신하는 단계를 포함하되,

상기 복수의 컴퓨팅 노드 각각은 병렬화 유지를 위해 동일한 시점에 순전파와 역전파를 번갈아가면서 수행하며, 학습 초기에는 역전파를 위한 gradient가 존재하지 않으므로 순전파만 진행하는 초병렬 심층 학습 방법.

청구항 2

제1항에 있어서,

상기 초병렬 심층 학습은 Pipelined EASGD(Elastic Averaging Stochastic Gradient Descent) 또는 Pipelined DC-ASGD(Delay Compensation-Asynchronous SGD)를 기반으로 수행되는 초병렬 심층 학습 방법.

청구항 3

제2항에 있어서,

상기 Pipelined EASGD에 기반하여 상기 복수의 컴퓨팅 노드 각각은 마스터 파라미터와 복수의 로컬 파라미터 중 상기 복수의 로컬 파라미터만 사용해서 순전파 계산을 수행하고,

상기 통신 주기마다 탄성력을 이용해서 상기 마스터 파라미터와 상기 복수의 로컬 파라미터의 차이를 제한하고, 상기 입력된 로스를 이용하여 그라디언트를 계산하고, 상기 통신 주기를 확인하여 상기 복수의 로컬 파라미터를 갱신하는 초병렬 심층 학습 방법.

청구항 4

제2항에 있어서,

상기 Pipelined DC-ASGD에 기반하여 상기 복수의 컴퓨팅 노드 각각은 1F1B 스케줄링에 의해 갱신된 최신의 파라미터를 사용하여 순전파 계산을 수행하고, 상기 입력된 로스를 이용하여 그라디언트를 계산하고 taylor expansion을 통해 지연되지 않았을 때의 gradient를 예측하여 최신의 파라미터를 갱신하는 초병렬 심층 학습 방법.

청구항 5

삭제

청구항 6

초병렬 심층 학습 장치로서,

프로세서; 및

상기 프로세서에 연결되는 메모리를 포함하되,

상기 메모리는,

순전파를 통해 입력 데이터, 파라미터와 모델을 이용하여 계산한 출력값을 다음 컴퓨팅 노드로 전달하고,

상기 입력 데이터 및 상기 파라미터의 사본을 오류 없는 역전파 계산을 위해 유지하고,

역전파를 통해 입력된 로스(loss), 상기 입력 데이터, 상기 파라미터를 이용하여 계산한 로스를 다음 컴퓨팅 노드로 전달하고,

상기 입력된 로스를 이용하여 그라디언트를 계산하고 지연되지 않은 시간대의 그라디언트를 예측하거나, 미리 설정된 통신 주기에 따라 탄성력 기반으로 동기화하여 상기 파라미터를 비동기적으로 갱신하도록,

상기 프로세서에 의해 실행 가능한 프로그램 명령어들을 저장하되,

서로 다른 컴퓨팅 노드에 연산 수행을 위한 서로 다른 모델이 할당되며,

상기 서로 다른 컴퓨팅 노드 각각은 병렬화 유지를 위해 동일한 시점에 순전파와 역전파를 번갈아가면서 수행하며, 학습 초기에는 역전파를 위한 gradient가 존재하지 않으므로 순전파만 진행하는 초병렬 심층 학습 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 초병렬 심층 학습 방법 및 장치에 관한 것으로서, 보다 상세하게는, 범용 컴퓨터 클러스터 환경에서 대규모 심층 신경망을 구축하고 고속으로 학습시킬 수 있는 방법 및 장치에 관한 것이다.

배 경 기 술

[0002] 최근 심층 신경망이 영상이나 이미지 등 여러 분야에서 사람의 수준을 능가하는 성능을 보이며 실용성을 입증하기 시작하였다. 그런데, 심층 학습 분야에서 잘 알려진 이미지 분류 챌린지의 결과를 보면 우수한 성적을 낸 신경망의 크기가 매년 커지고 있는 경향을 보이고 있다.

[0003] 주목할 점은 이런 증가 추세에도 현재 심층 신경망의 복잡도는 아직 생물학적 신경망의 복잡도에 못 미치고 있다는 것이다. 예를 들어 이미지 인식이나 음성 인식 분야의 경우 현재 연구되고 있는 심층 신경망의 weight 수가 10^{7}-10^{8}정도이다.

[0004] 인간 두뇌의 synapse 수가 10^{14} 정도인 것을 고려할 때 현재 시도되고 있는 심층 신경망의 복잡도는 인간 수준의 지능을 가지기에는 아직도 턱없이 부족하다고 할 수 있다.

[0005] 즉, 신경망의 규모가 성능 향상과 연관이 있으며 아직 규모에 의한 성능 개선 여지가 남아있음을 알 수 있다. 제4차 산업 혁명의 핵심인 인공 지능이 진정한 의미의 지능을 발휘하고 강 인공지능(strong artificial intelligence)과 같은 진정한 인공 지능으로 도약하기 위해서는 이제까지 시도되지 않은 대규모 심층 신경망 구조 및 고속의 학습 방법에 대한 연구가 필요하다.

[0006] 심층 신경망은 뉴런을 층(layer) 단위로 쌓아 올려서 인간의 두뇌 구조를 모사한 계산 모델로서, 일반적으로 뉴런의 수가 많을수록 높은 성능을 보인다.

[0007] 뉴런의 수가 증가함에 따라 요구되는 연산량도 증가되므로, 단일 서버가 아닌 다수의 서버를 이용한 분산 처리 방식의 병렬 학습이 필요하다.

[0008] 그러나 심층 신경망은 뉴런 상호 간에 데이터 전송 횟수가 많아서 임의로 분할할 경우 데이터 전송 및 동기화에 많은 부하가 걸리게 되며, 이는 병렬화의 효율을 떨어뜨릴 수 있다.

[0009] 일반적으로 심층 신경망을 이용한 모델들의 경우 많은 연산을 필요로 한다. 특히, 빅데이터를 이용할 경우 매우 높은 계산 복잡도를 지니게 된다. 최근에 클러스터 병렬화 시도에서 GPU (graphics processing unit) 64개를 이용하는 연구가 보고되었다. 이러한 하드웨어의 연산 속도는 생물학적 뉴런의 속도보다 훨씬 빠르고 심층 신경망

규모 역시 소규모임에도 불구하고 그 연산 속도는 생물학적 신경망보다 뒤떨어지는 문제가 있다. 이러한 현상은 현재 컴퓨터의 병렬 연산이 생물학적 뉴런의 병렬 연산에 비하여 매우 비효율적이기 때문이다. 따라서, 심층 신경망의 효율적인 병렬화 연구가 필요하다.

[0010] 기존의 초대규모 심층 신경망 모델들은 주로 해당 모델 전용으로 설계된 하드웨어를 이용하였다. IBM에서는 슈퍼컴퓨터를 이용해 2천2백만 개의 뉴런과 110억 개의 시냅스에 대한 시뮬레이션을 수행했으며, 전용 칩을 이용한 구조를 통해 성능 향상을 시도하였다. 그러나 이런 전용 하드웨어를 이용한 기술은 모델을 구성하는데 소요되는 비용이 매우 크고 전용 하드웨어에 대한 기술이 추가로 필요하다.

[0011] 따라서 이런 문제를 회피하면서 대규모 신경망을 구축하기 위해서는 범용 하드웨어를 이용하여 확장성을 가지는 대규모 심층 신경망을 구축할 수 있는 방법에 관한 연구가 필요하다.

[0012] 그러나 일반적인 범용 컴퓨터 클러스터를 이용해 신경망의 학습 속도를 개선하는 데는 어려움이 많다. 이는 클러스터를 구성하는 각 컴퓨팅 노드 간의 통신 속도가 컴퓨팅 노드 내부의 데이터 교환 속도에 비해 매우 느리기 때문이다.

선행기술문헌

특허문헌

[0013] (특허문헌 0001) US 공개특허 2019-0114548

발명의 내용

해결하려는 과제

[0014] 본 발명에서는 확장성을 가지면서도 데이터 전송 및 동기화를 최소화할 수 있는 초병렬 심층 학습 방법 및 장치를 제안하고자 한다.

과제의 해결 수단

[0015] 상기한 바와 같은 목적을 달성하기 위하여, 본 발명의 일 실시예에 따르면, 범용 컴퓨터 클러스터에서의 초병렬 심층 학습 방법으로서, 복수의 컴퓨팅 노드 각각에 연산 수행을 위한 서로 다른 모델을 할당하는 단계; 상기 복수의 컴퓨팅 노드 각각이 순전파를 통해 입력 데이터, 파라미터와 상기 모델을 이용하여 계산한 출력값을 다음 컴퓨팅 노드로 전달하는 단계; 복수의 컴퓨팅 노드 각각이 상기 입력 데이터 및 상기 파라미터의 사본을 오류 없는 역전파 계산을 위해 유지하는 단계; 상기 복수의 컴퓨팅 노드 각각이 역전파를 통해 입력된 로스(loss), 상기 입력 데이터, 상기 파라미터를 이용하여 계산한 로스를 다음 컴퓨팅 노드로 전달하는 단계; 및 상기 복수의 컴퓨팅 노드 각각이 상기 입력된 로스를 이용하여 그라디언트를 계산하고 지연되지 않은 시간대의 그라디언트를 예측하거나, 미리 설정된 통신 주기에 따라 탄성력 기반으로 동기화하여 상기 파라미터를 비동기적으로 갱신하는 단계를 포함하는 초병렬 심층 학습 방법이 제공된다.

[0016] 상기 초병렬 심층 학습은 Pipelined EASGD(Elastic Averaging Stochastic Gradient Descent) 또는 Pipelined DC-ASGD(Delay Compensation-Asynchronous SGD)를 기반으로 수행될 수 있다.

[0017] 상기 Pipelined EASGD에 기반하여 상기 복수의 컴퓨팅 노드 각각은 마스터 파라미터와 복수의 로컬 파라미터 중 상기 복수의 로컬 파라미터만 사용해서 순전파 계산을 수행하고, 상기 통신 주기마다 탄성력을 이용해서 상기 마스터 파라미터와 상기 복수의 로컬 파라미터의 차이를 제한하고, 상기 입력된 로스를 이용하여 그라디언트를 계산하고, 상기 통신 주기를 확인하여 상기 복수의 로컬 파라미터를 갱신할 수 있다.

[0018] 상기 Pipelined DC-ASGD에 기반하여 상기 복수의 컴퓨팅 노드 각각은 1F1B 스케줄링에 의해 갱신된 최신의 파라미터를 사용하여 순전파 계산을 수행하고, 상기 입력된 로스를 이용하여 그라디언트를 계산하고 taylor expansion을 통해 지연되지 않았을 때의 gradient를 예측하여 최신의 파라미터를 갱신할 수 있다.

[0019] 상기 복수의 컴퓨팅 노드 각각은 병렬화 유지를 위해 동일한 시점에 순전파와 역전파를 번갈아가면서 수행하며, 학습 초기에는 역전파를 위한 gradient가 존재하지 않으므로 순전파만 진행할 수 있다.

[0020] 본 발명의 다른 측면에 따르면, 초병렬 심층 학습 장치로서, 프로세서; 및 상기 프로세서에 연결되는 메모리를

포함하되, 상기 메모리는, 순전파를 통해 입력 데이터, 파라미터와 모델을 이용하여 계산한 출력값을 다음 컴퓨팅 노드로 전달하고, 상기 입력 데이터 및 상기 파라미터의 사본을 오류 없는 역전파 계산을 위해 유지하고, 역전파를 통해 입력된 로스(loss), 상기 입력 데이터, 상기 파라미터를 이용하여 계산한 로스를 다음 컴퓨팅 노드로 전달하고, 상기 입력된 로스를 이용하여 그라디언트를 계산하고 지연되지 않은 시간대의 그라디언트를 예측하거나, 미리 설정된 통신 주기에 따라 탄성력 기반으로 동기화하여 상기 파라미터를 비동기적으로 갱신하도록, 상기 프로세서에 의해 실행 가능한 프로그램 명령어들을 저장하되, 서로 다른 컴퓨팅 노드에 연산 수행을 위한 서로 다른 모델이 할당되는 초병렬 심층 학습 장치가 제공된다.

발명의 효과

[0021] 본 발명에 따르면, 모델 및 데이터 병렬화를 수행하면서도 delayed gradient 문제를 해결할 수 있는 장점이 있다.

도면의 간단한 설명

[0022] 도 1은 본 발명의 바람직한 일 실시예에 따른 범용 컴퓨터 클러스터 기반의 심층 신경망 구조를 도시한 도면이다.

도 2는 동기적 방식으로 마스터 파라미터를 갱신하는 과정을 도시한 도면이다.

도 3은 비동기적 방식으로 마스터 파라미터를 갱신하는 과정을 도시한 도면이다.

도 4는 모델 병렬화를 설명하기 위한 도면이다.

도 5는 Pipelined 병렬화 구조를 도시한 도면이다.

도 6은 Pipelined 병렬화 알고리즘의 동작을 도시한 도면이다.

도 7은 본 실시예에 따른 Pipelined EASGD 구조를 도시한 도면이다.

도 8은 본 실시예에 따른 파라미터 갱신 알고리즘을 나타낸 도면이다.

발명을 실시하기 위한 구체적인 내용

[0023] 본 발명은 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세하게 설명하고자 한다.

[0024] 그러나, 이는 본 발명을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0026] 본 발명은 범용 컴퓨터 클러스터 환경에서 대규모 심층 신경망(Deep Neural Network, DNN)을 구축하고, 고속으로 학습시킬 수 있는 초병렬 심층 학습 알고리즘을 제안하고자 한다.

[0027] 도 1은 본 발명의 바람직한 일 실시예에 따른 범용 컴퓨터 클러스터 기반의 심층 신경망 구조를 도시한 도면이다.

[0028] 본 실시예에 따르면, GPU가 장착된 컴퓨터 수십 대를 기가비트 이더넷으로 연결하여 범용 컴퓨터 클러스터를 구축할 수 있다.

[0029] GPU는 일반적인 그래픽 처리에 사용되는 범용 부품이며, 기가비트 이더넷 역시 인터넷망을 구축하는데 가장 보편적으로 이용되는 연결 방식이다. 도 1에서 붉은색 실선으로 표시된 GPU 간의 연결은 높은 대역폭(high bandwidth)을 지니는 고속 연결이고, 푸른색 점선으로 표시된 노드 간의 연결은 상대적으로 낮은 대역폭(low bandwidth)을 지니는 연결이다.

[0030] 본 발명은 병목 현상이 발생하는 것을 방지하기 위해, 마스터 노드를 두지 않고, 인접한 컴퓨팅 노드들을 전용 통신선으로 연결한다.

[0031] 또한, 본 실시예에 따르면, 복수의 컴퓨팅 노드 각각에 연산 수행을 위한 서로 다른 모델을 할당하는 모델 병렬화와, 하나의 컴퓨팅 노드 내에 복수의 로컬 워커(Local Worker)를 배치하여 데이터 병렬화를 기반으로 심층 신경망 학습이 수행된다.

[0032] 각 컴퓨팅 노드에서는 순전파를 통해 입력 데이터, 파라미터와 상기 모델을 이용하여 계산한 출력값을 다음 컴

퓨팅 노드로 전달하고, 이때 상기 입력 데이터 및 상기 파라미터의 사본을 오류 없는 역전파 계산을 위해 유지한다.

[0033] 또한, 각 컴퓨팅 노드는 역전파를 통해 입력된 로스(loss), 상기 입력 데이터, 상기 파라미터를 이용하여 계산한 로스를 다음 컴퓨팅 노드로 전달한다.

[0034] 상기한 병렬화 기반의 순전파 및 역전파 수행에서 모델의 가중치와 같은 파라미터의 갱신이 필요하다.

[0035] 이를 위해, 본 실시예에서는 Pipelined EASGD(Elastic Averaging Stochastic Gradient Descent) 또는 Pipelined DC-ASGD(Delay Compensation-Asynchronous SGD)를 이용하여 순전파/역전파 및 파라미터 갱신을 수행한다.

[0036] 파라미터 갱신에 있어서, Pipelined EASGD 또는 Pipelined DC-ASGD를 이용하여 상기와 같이 입력된 로스로 그라디언트를 계산하고 지연되지 않은 시간대의 그라디언트를 예측하거나, 미리 설정된 통신 주기에 따라 탄성력 기반으로 동기화하여 파라미터를 갱신한다.

[0037] 본 실시예에 따르면, Pipeline 병렬화의 유지를 위해 각 컴퓨팅 노드는 동일한 시점에 순전파(forward)와 역전파(backward)를 번갈아가면서 수행한다. 학습 초기에는 역전파를 위한 gradient가 존재하지 않으므로 순전파만 진행한다.

[0038] 이는 1FIB 스케줄링으로 정의될 수 있다.

[0039] 순전파 수행에 있어, Pipelined EASGD는 마스터 파라미터와 여러 개의 로컬 파라미터 중 로컬 파라미터만 사용해서 순전파 계산을 수행한다.

[0040] 또한, Pipelined DC-ASGD는 1F1B 스케줄링에 의해 갱신된 최신의 파라미터를 사용하여 순전파 계산을 수행한다.

[0041] 본 실시예에 따른 각 컴퓨팅 노드는 프로세서 및 메모리를 포함하여 초병렬 심층 학습 과정을 수행할 수 있다.

[0042] 여기서, 프로세서는 컴퓨터 프로그램을 실행할 수 있는 CPU(central processing unit)나 그밖에 가상 머신 등을 포함할 수 있다.

[0043] 메모리는 고정식 하드 드라이브나 착탈식 저장 장치와 같은 불휘발성 저장 장치를 포함할 수 있다. 착탈식 저장 장치는 컴팩트 플래시 유닛, USB 메모리 스틱 등을 포함할 수 있다. 메모리는 각종 랜덤 액세스 메모리와 같은 휘발성 메모리도 포함할 수 있다.

[0044] 이와 같은 메모리에는 프로세서에 의해 실행 가능한 프로그램 명령어들이 저장된다.

[0046] 이하에서는 종래의 심층 신경망 학습 알고리즘을 대략적으로 살펴본 후, 본 실시예에 따른 초병렬 심층 학습 과정을 상세하게 설명한다.

[0047] 현재 가장 널리 사용되고 있는 심층 신경망 학습 방법은 contrastive divergence(CD)를 이용한 사전 학습(pre-training) 방법과 오류 역전파(error back propagation: EBP)를 이용한 학습 방법이다.

[0048] 이 두 가지 학습 방법은 로스 함수(loss function)만 다를 뿐, gradient descent 방식으로 모델 파라미터(DNN의 weight)를 최적화한다.

[0049] Batch gradient descent라고도 불리는 원래의 gradient descent 방식에서는 전체 학습 데이터를 이용하여 gradient를 계산한 후 모델 파라미터를 갱신하는 과정을 반복한다.

[0050] 그런데, Batch gradient descent는 데이터 병렬화 기법을 사용한 병렬 처리가 가능하지만, 많은 local minima를 가지는 DNN의 학습에는 적당하지 않다.

[0051] 반면에 stochastic gradient descent (SGD) 방식에서는 전체 학습 데이터를 이용하여 정확한 gradient를 구하는 대신에 매 학습 샘플마다 gradient를 구하고 모델 파라미터를 갱신하는 과정을 반복한다.

[0052] 이것은 일종의 random walk 현상을 제공하여 local minima를 탈출하는 효과가 있기 때문에 DNN 학습에 상대적으로 유리하다. 하지만 SGD는 순차적으로 동작하기 때문에 병렬화하기 어려워서 학습 시간이 오래 걸린다는 단점이 있다.

[0053] Batch gradient descent와 stochastic gradient descent를 절충한 mini-batch stochastic gradient descent 방식은 전체 학습 데이터를 mini-batch라고 불리는 여러 개의 작은 그룹으로 나누고, 하나의 mini-batch에 속한

모든 데이터를 이용하여 gradient를 구하고 모델 파라미터를 갱신하는 과정을 반복한다.

[0054] Mini-batch 간에는 순차적으로 수행되지만, mini-batch 내부에서는 GPU 등을 사용하여 batch gradient descent와 유사한 방법으로 데이터를 병렬화함으로써 수행 속도를 향상시킬 수 있기 때문에 현재 가장 널리 사용되고 있는 DNN 학습 방법이다.

[0055] 그러나 데이터 병렬화 효율을 높이기 위해서 mini-batch의 크기를 늘리면 random walk 효과가 줄어들어 학습이 어려워진다는 단점이 있다. 즉, 병렬화 효율이 제한적이다.

[0056] 수식 표현과 설명의 간략화를 위해서 알고리즘은 SGD를 기준으로 설명하지만, 이후에 나오는 모든 SGD는 mini-batch SGD를 의미한다.

[0057] SGD 알고리즘은 다음과 같이 손실 함수의 gradient 반대 방향으로 DNN의 가중치(weight)를 조금씩 수정하면서 학습을 진행한다.

수학식 1

[0058] $$w^{t+1} = w^{t} - \eta \nabla L(w^{t}; x^{t}, y^{t})$$

[0059] 여기서, ω는 weight, η는 learning rate, L은 손실 함수, x는 입력 데이터, y는 심층 신경망이 출력해야 하는 목표값(target value), 첨자 t는 반복 횟수(즉 mini-batch 인덱스)를 나타낸다.

[0060] 반복적 수행이 필요한 SGD 기반의 최적화 알고리즘은 순차적으로 진행되기 때문에 병렬화하기 어렵다. 따라서 SGD 기반의 병렬 학습 알고리즘은 다양한 근사(approximation) 방법을 사용한다. 이러한 근사 방법들은 SGD에서 필요한 gradient 값을 근사적인 방식으로 구하는데, 이하에서 설명할 delayed gradient가 성능 저하의 중요한 요인이 된다.

[0061] Pipelined SGD 병렬화 알고리즘은 각 컴퓨팅 노드들이 효율적으로 운영되며, DNN 모델 파라미터가 분산되어 메모리가 효율적으로 사용되기 때문에 본 실시예에서 목표로 하는 범용 하드웨어를 이용한 초대규모 DNN 학습에 적합하다. 하지만 다른 병렬 학습 알고리즘과 마찬가지로 delayed gradient 문제 때문에 학습 정확도가 저하되는 단점이 있으므로, 이를 해결하여야 대규모 DNN을 효과적으로 학습시킬 수 있다.

[0062] 본 실시예에서는, pipelined SGD(stochastic gradient descent) 병렬화 알고리즘의 장점들을 유지하면서 delayed gradient로 인한 학습 성능 저하 문제를 해결할 수 있는 방법을 제안한다.

[0063] 이를 위하여 범용 하드웨어로 연산 클러스터를 구축하고, pipelined SGD 알고리즘의 연산 효율성과 통신 비용을 분석하고 병렬 학습 알고리즘을 구현한다. 추가적으로, delayed gradient 문제를 완화할 수 있는 심층 신경망 학습 알고리즘을 제안한다.

[0064] 먼저 기존의 병렬화 방법과 그 문제점을 분석한다.

[0065] 일반적으로 병렬화 방법은 데이터 병렬화(data parallelism)를 활용한 방법과 모델 병렬화(model parallelism)를 활용한 방법으로 분류할 수 있다.

[0066] 데이터 병렬화는 데이터를 나누어 동시에 수행하는 것이다. 일반적으로 데이터 병렬화 기반 DNN 학습에는 마스터 파라미터를 관리 및 저장하는 파라미터 서버와 마스터 파라미터의 사본과 학습 데이터 일부를 전송받아서 gradient를 계산하는 컴퓨팅 노드들이 필요하다.

[0067] 여기서 컴퓨팅 노드란 하드웨어 구성에 따라서 GPU 또는 CPU (central processing unit) 코어(core) 등이 될 수 있다.

[0068] 전체 학습 데이터를 컴퓨팅 노드 개수로 나눈 일부 학습 데이터와 DNN 마스터 파라미터를 모든 컴퓨팅 노드에 전송하고, 각 컴퓨팅 노드는 서로 다른 학습 데이터를 사용하여 동시에 학습한다.

[0069] 각 컴퓨팅 노드의 DNN은 서로 다른 데이터를 사용하여 학습하기 때문에 결과적으로 서로 다른 로컬 파라미터 값을 갖게 되는데, 이러한 로컬 파라미터들, 즉 지역 DNN 모델의 weight들을 통합하여 하나의 마스터 모델을 만들어야 한다. 모든 컴퓨팅 노드의 지역 모델을 통합하여 마스터 모델을 만드는 방법에는 동기적(synchronous) 방

식과 비동기적(asynchronous) 방식이 있다.

[0070] 도 2는 동기적 방식으로 마스터 파라미터를 갱신하는 과정을 도시한 도면이다.

[0071] 도 2에 도시된 바와 같이, Synchronous SGD는 일정 시점에 모든 컴퓨팅 노드의 로컬 파라미터를 조합하여 마스터 파라미터를 만든다.

[0072] 보다 상세하게, (1) 마스터 모델에서 각 컴퓨팅 노드로 파라미터 사본을 전송한다; (2) 각 컴퓨팅 노드에서 gradient를 계산한다; (3) 계산된 gradient를 다시 파라미터 서버로 전송한다; (4) 마스터 모델의 파라미터를 갱신한다. (1)에서 (4) 단계가 계속하여 반복된다.

[0073] 가장 간단한 로컬 파라미터 조합 방법은 다음과 같이 모든 로컬 파라미터를 평균하여 마스터 파라미터를 생성하는 것이다.

수학식 2

[0074] $$\overline{w}^{t}=\frac{1}{K}\sum_{k=1}^{K}w_{k}^{t}$$

[0075] 여기서, $\overline{w}^{t}$는 마스터 파라미터이고, w_{k}^{t}는 k번째 컴퓨팅 노드의 로컬 파라미터이고, K는 컴퓨팅 노드의 개수이다. DNN weight 대신에 gradient를 전송해서 마스터 파라미터를 구할 수도 있다. 갱신된 마스터 모델이 다시 각 컴퓨팅 노드에 배포될 때 모든 지역 모델은 동일한 파라미터 값을 갖게 된다. 컴퓨팅 노드의 로컬 파라미터를 조합하는 방법과 빈도에 따라서 여러 가지 변형된 방법이 존재한다.

[0076] Mini-batch 크기를 고려해서 동기화 주기를 결정할 수도 있다. 즉, τ번째 mini-batch 학습 후 동기화가 수행되게 하고, 수렴 속도를 고려하여 τ를 최적화한다. 동기화에는 통신이 수반되기 때문에 동기화 주기 최적화 시 통신 비용도 함께 고려해야 한다. 이렇게 τ번째 mini-batch마다 동기화를 수행하는 것은 실질적인 mini-batch 크기를 증가하는 효과가 있는 것은 아니다.

[0077] 즉, hypothesis space에서 개별적으로 학습된 로컬 파라미터의 조합으로 마스터 파라미터를 주기적으로 갱신하는 것으로서, 로컬 파라미터들의 평균을 이용하는 방법 외에도 파라미터마다 가중치를 부여하는 등의 여러 가지 조합 방법이 가능하다.

[0078] Elastic averaging SGD (EASGD)에서는 각 로컬 파라미터가 개별적으로 유지되지만 마스터 파라미터로부터 멀어질 수 있는 정도가 제어된다. Block-wise model-update filtering (BMUF)에서는 이전 마스터 파라미터와 현재 로컬 파라미터 평균의 차이를 마스터 파라미터 변화량으로 간주하고 momentum을 추가하여 마스터 파라미터를 갱신한다.

[0079] 많은 컴퓨팅 노드를 사용하는 경우 일부 컴퓨팅 노드의 일시적인 통신 장애 발생과 같은 예기치 못한 상황으로 전체 동기화 속도가 느려질 수 있다. 이러한 문제를 해결하기 위해서 Sync-SGD에서는 K개 이상의 컴퓨팅 노드를 할당한 후 K개의 컴퓨팅 노드에서 gradient 계산이 끝나면 나머지 컴퓨팅 노드를 기다리지 않고 동기화를 진행한다. 동기화에서 배제된 컴퓨팅 노드에 할당된 학습 전체가 데이터가 학습에 사용되지 못하는 문제를 완화하기 위해서 각 컴퓨팅 노드는 매번 전체 학습 데이터에서 무작위로 mini-batch를 선택하여 컴퓨팅 노드 간에 사용하는 데이터에 중복이 발생할 수 있도록 한다.

[0080] 도 3은 비동기적 방식으로 마스터 파라미터를 갱신하는 과정을 도시한 도면이다.

[0081] 도 2를 참조하면, (1) 마스터 파라미터가 비동기적으로 각 컴퓨팅 노드에 전달된다; (2) 각 컴퓨팅 노드에서 gradient를 계산한다; (3) 계산된 gradient를 파라미터 서버로 비동기적으로 전송한다; (4) 마스터 파라미터를 비동기적으로 갱신한다. (1)에서 (4) 단계가 계속하여 반복된다.

[0082] 도 3에 도시된 바와 같이, asynchronous SGD는 synchronous SGD에 비해서 마스터 파라미터 갱신을 위한 동기화 비용이 적고, 상대적으로 느리거나 고장난 컴퓨팅 노드를 기다릴 필요가 없기 때문에 DNN 학습에 가장 널리 사용되는 병렬 학습 방법 중에 하나이다.

[0083] Asynchronous SGD에서 각 컴퓨팅 노드는 각자의 mini-batch 데이터를 이용해서 gradient를 계산한 후 로컬 파라미터를 갱신한다. 동시에 각 컴퓨팅 노드는 무작위로 선택된 인접 컴퓨팅 노드의 로컬 파라미터와 조합하여 파라미터를 동기화한다.

[0084] 이렇게 하면 다수의 컴퓨팅 노드가 동시에 파라미터 서버와 통신해야 하는 상황이 발생하지 않기 때문에 통신 병목 현상이 감소된다. 학습이 종료되면 모든 컴퓨팅 노드의 로컬 파라미터를 평균해서 마스터 파라미터를 구한다.

[0085] Asynchronous SGD에서는 컴퓨팅 노드들이 파라미터 서버에게 gradient를 비동기적으로 전송하기 때문에 먼저 도착한 gradient 순으로 마스터 파라미터에 적용된다. 늦게 도착한 gradient는 그 gradient를 계산하기 위해서 사용된 마스터 파라미터가 아니라 먼저 도착한 gradient에 의해서 이미 갱신된 마스터 파라미터에 적용되기 때문에 문제가 발생할 수 있다. 이러한 gradient를 delayed gradient라 한다.

[0086] 이하에서는 모델 병렬화를 설명한다.

[0087] 모델 병렬화는 모델을 나누어 동시에 수행하는 것이다.

[0088] 도 4는 모델 병렬화를 설명하기 위한 도면이다.

[0089] 도 4에 도시된 바와 같이, 모델 병렬화 기반 DNN 학습에서는 뉴런들을 여러 컴퓨팅 노드에 분산 배치함으로써, 나뉘어진 모델의 뉴런들이 여러 컴퓨팅 노드에서 동시에 계산에 참여할 수 있게 한다.

[0090] 예를 들어 convolutional neural network (CNN)의 경우 필터(filter)들이 여러 컴퓨팅 노드에 분산되어 동시에 convolution 연산을 수행할 수 있다. 각 뉴런 출력값과 출력층으로부터 뒤로 전파되는 오류값은 컴퓨팅 노드 간의 통신망을 통하여 전달된다. 생물학적 신경망과 같이 모든 뉴런들이 동시에 계산에 참여할 수 있기 때문에 이론적으로는 병렬성이 높지만, 컴퓨팅 노드 간의 통신 비용 때문에 범용 컴퓨팅 환경에서는 병렬화 효율에 한계가 있다.

[0091] 또한, DNN의 학습 과정이 근본적으로 feed-forward (forward pass)와 오류 역전파(backward pass)의 순차적 반복이기 때문에 임의로 뉴런들을 분산 배치할 경우 모든 뉴런이 동시에 작업할 수 없는 경우가 생겨서 연산 자원이 비효율적으로 사용될 가능성이 있다.

[0092] 도 5는 Pipelined 병렬화 구조를 도시한 도면이다.

[0093] 도 5에 도시된 바와 같이, Pipelined 병렬화 방식은 DNN의 층 단위로 모델을 분산하는 모델 병렬화의 일종이다. 즉, 같은 층에 속한 뉴런은 같은 컴퓨팅 노드에 할당한다.

[0094] 도 6은 Pipelined 병렬화 알고리즘의 동작을 도시한 도면이다.

[0095] 도 6에 도시된 바와 같이, 첫 번째 컴퓨팅 노드에 할당된 층에서 t번째 mini-batch의 forward pass를 마치면 다음 컴퓨팅 노드에 할당된 층에서 t번째 mini-batch의 forward pass를 진행한다. 이때, 첫 번째 컴퓨팅 노드는 t+1번째 mini-batch의 forward pass를 동시에 진행한다. 유사하게 backward pass도 각각 다른 mini-batch에 대하여 모든 컴퓨팅 노드가 동시에 진행한다.

[0096] 따라서 Pipelined 병렬화에서는 모든 컴퓨팅 노드가 효율적으로 운영되며, DNN 모델 파라미터가 분산되어 메모리가 효율적으로 사용되기 때문에 대규모 DNN 학습에 적합하다. 그러나 컴퓨팅 노드에 backward pass에 의해서 오류가 역전파되었을 때, 그 컴퓨팅 노드의 forward pass 값과 weight는 이미 다른 mini-batch에 의해서 변경되었기 때문에 잘못된 forward pass 값과 weight를 사용하는 문제가 발생한다. 즉, delayed gradient 문제가 발생하게 된다.

[0097] 본 발명은 연산 효율을 높이면서도 delayed gradient 문제 해결을 위해 Pipelined EASGD 또는 Pipelined DC-ASGD 알고리즘을 제안한다.

[0098] 전술한 바와 같이, EASGD(Elastic Averaging SGD)는 데이터 병렬화 알고리즘의 일종으로서, mini-batch에 상응하는 각 로컬 워커(Local Worker)들이 로컬 파라미터를 개별적으로 유지하고 학습하지만, 마스터 워커(Master Worker)의 마스터 파라미터에서 멀어지는 정도가 제한된다.

[0099] EASGD 기법에서 각 Local Worker들의 파라미터는 Master Worker의 파라미터와 Elastic Force(이하 탄성력)로 연결된 것처럼 유지된다.

[0100] 본 실시예에 따른 Pipelined EASGD에서는 데이터 병렬화 뿐만 아니라 모델 병렬화가 동시에 적용된다.

[0101] 즉, Pipelined EASGD는 전체 학습에서 모델 사본을 여러 개 두는 방식으로 데이터 병렬화를 적용하면서, 컴퓨팅 노드에 할당된 모델을 쪼개서 저장하기 때문에 모델 병렬화 기법도 적용되는 것이다.

[0102] 도 7은 본 실시예에 따른 Pipelined EASGD 구조를 도시한 도면이다.

[0103] 도 7에서는 각 컴퓨팅 노드에 4개의 로컬 워커가 제공되고, 신경망 모델이 AB 구조인 경우에 모델 A는 Node 0, 모델 B는 Node 1에 할당된 상태를 도시한 것이다.

[0104] Pipelined EASGD에서, 마스터 파라미터와 로컬 파라미터의 갱신은 비동기적으로 부분적으로 진행된다.

[0105] 도 8은 본 실시예에 따른 파라미터 갱신 알고리즘을 나타낸 도면이다.

[0106] 예를 들어, Node 1에서 파라미터 갱신이 수행된다면, 신경망 모델의 B에 해당하는 부분만 갱신된다. 하지만, delayed gradient의 시간 지연만큼 모델 사본을 만든다면, 로컬 워커들의 로컬 파라미터는 올바르게 갱신되며, 통신 주기 τ마다 수행하는 마스터 워커와 로컬 워커 간의 탄성력 기반 동기화도 스케줄링에 따라 올바르게 수행될 수 있다.

[0107] 다음 식처럼 k번째 연산 노드의 파라미터 w_k^{t+1} 는 loss 함수 L의 기울기와 반대 방향으로 갱신되면서, 마스터 모델의 파라미터 $\overline{w}^t$ 와의 차이가 적어지도록 유지된다.

수학식 3

[0108] $$w_k^{t+1}=w_k^t-\eta\nabla L(w_k^t;x^t,y^t)-\alpha(w_k^t-\overline{w})$$

[0109] 여기서 α는 탄성력의 강도를 조절하는 파라미터이다. 탄성력의 강도가 적을수록 자유로운 학습이 가능하다.

[0110] Nesterov's momentum을 적용하면 각 연산 노드의 개별 모델 파라미터 갱신 수식은 다음과 같이 표현할 수 있다.

수학식 4

[0111] $$v_k^{t+1}=\beta v_k^t-\eta\nabla L(w_k^t+\beta v_k^t;x^t,y^t)$$

수학식 5

[0112] $$w_k^{t+1}=w_k^t+v_k^{t+1}-\alpha(w_k^t-\overline{w}^t)$$

[0113] 여기서 v_k^t는 Nesterov's momentum이고, β는 Nesterov's momentum의 가중치이다.

[0114] 이러한 기법은 delayed gradient 문제를 완화하는데 이용될 수 있다. 즉, 각 층에서 주기적으로 모델 사본들을 통합할 때에, elastic averaging 방법을 적용하여 새로운 마스터 모델과 지역 모델에 해당하는 모델 사본들을 갱신할 수 있다.

[0115] Pipelined DC-ASGD는 Taylor series 기반 gradient를 도입하는 것이다. 즉, $t+\tau$번째 시간의 로스 함수에서 사용할 gradient를 t번째 시간의 gradient로 아래와 같이 추정할 수 있다. 즉, 지연되지 않은 시간대의 그라디언트를 예측하는 것이다.

수학식 6

[0116] $$\nabla L(w^{t+\tau};x^t,y^t)=\nabla L(w^t;x^t,y^t)+\nabla^2 L(w^t;x^t,y^t)(w^{t+\tau}-w^t)+\varepsilon$$

[0117] 여기에서 ∇^2는 로스 함수의 Hessian matrix에 해당한다. 심층 신경망 학습의 경우 파라미터의 수가 너무 많아서 실제 Hessian matrix를 구하기 어렵기 때문에 기존에는 아래와 같은 근사 수식을 제안한다.

수학식 7

[0118] $$\nabla L(w^{t+\tau};x^t,y^t)\approx\nabla L(w^t;x^t,y^t)+\gamma\nabla L(w^t;x^t,y^t)\odot\nabla L(w^t;x^t,y^t)\odot(w^{t+\tau}-w^t)$$

[0119] 여기에서 γ는 근사 수식의 안정성을 조정하는 hyperparameter이며, $\odot$는 element-wise product를 의미한다. 기존의 pipelined SGD 알고리즘에 위 수식을 적용하여 delayed gradient 문제를 완화할 수 있다. 즉, l번째 층에서 발생하는 delay가 τ인 경우 $t+\tau$번째 시간에서 l번째 층의 backward pass 계산시에 근사화된 $\nabla L(w^{t+\tau};x^t,y^t)$를 사용하여 $w^{t+\tau}$를 아래와 같이 갱신하게 된다.

수학식 8

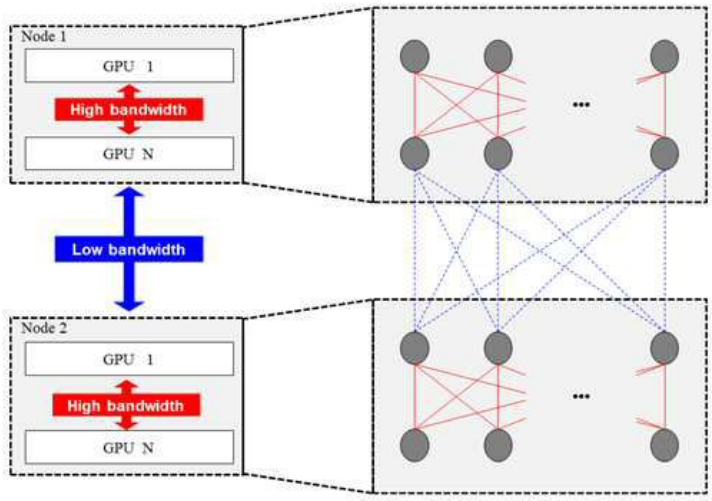
[0120] $$w_l^{t+\tau+1}=w_l^{t+\tau}-\eta\left(\nabla L(w_l^t;x^t,y^t)+\gamma\nabla L(w_l^t;x^t,y^t)\odot\nabla L(w_l^t;x^t,y^t)\odot(w_l^{t+\tau}-w_l^t)\right)$$

[0122] 상기한 본 발명의 실시예는 예시의 목적을 위해 개시된 것이고, 본 발명에 대한 통상의 지식을 가지는 당업자라면 본 발명의 사상과 범위 안에서 다양한 수정, 변경, 부가가 가능할 것이며, 이러한 수정, 변경 및 부가는 하기의 특허청구범위에 속하는 것으로 보아야 할 것이다.

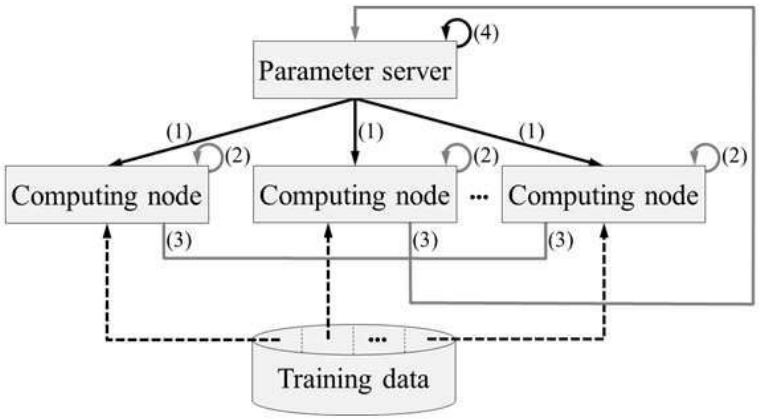
[0125]

도면

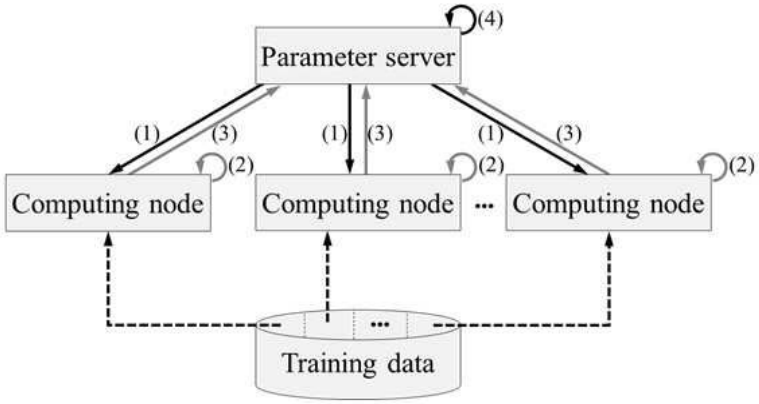
도면1



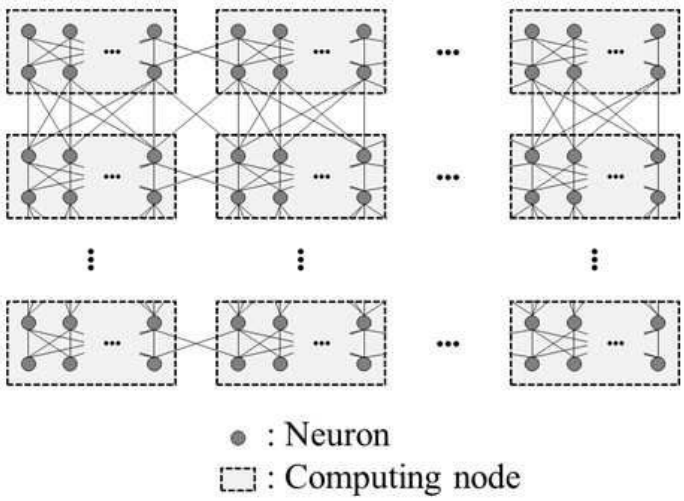
도면2


Parameter server
(4)
(1)
(1)
(1)
(2)
(2)
(2)
Computing node
Computing node
Computing node
(3)
(3)
(3)
Training data

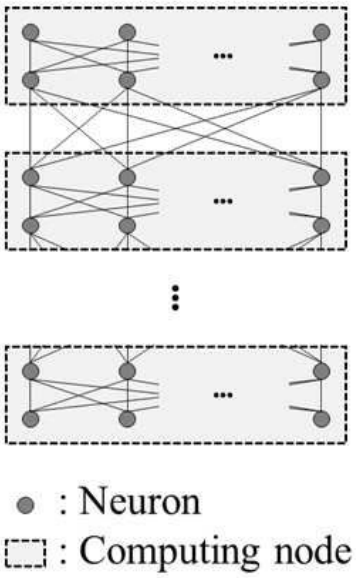
도면3


Parameter server
(4)
(1)
(3)
(1)
(3)
(1)
(3)
(2)
(2)
(2)
Computing node
Computing node
Computing node
Training data

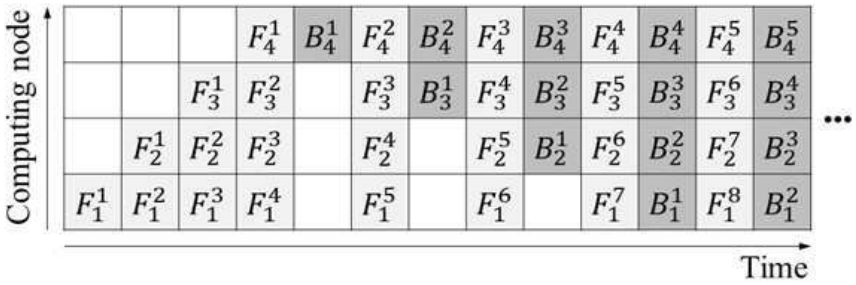
도면4


: Neuron
: Computing node

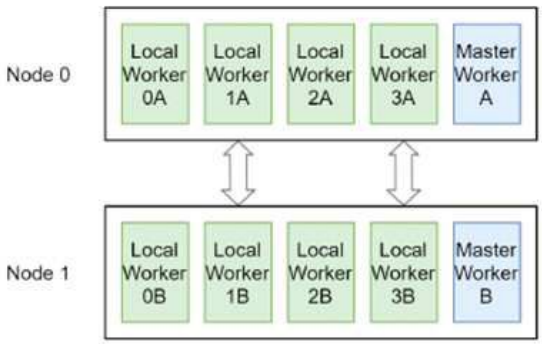
도면5


: Neuron
: Computing node

도면6


Computing node
Time

도면7


Node 0
Local Worker 0A
Local Worker 1A
Local Worker 2A
Local Worker 3A
Master Worker A
Node 1
Local Worker 0B
Local Worker 1B
Local Worker 2B
Local Worker 3B
Master Worker B

도면8

Algorithm : Pipelined EASGD
Processing by worker i and the master

parameter
i : Computing Node index
m : Local Worker index
* : Master Worker
t : mini-batch index

Repeat
$x^i \leftarrow x^i_m$
if (τ divides t^i_m) **then**
a) $x^i_m \leftarrow x^i_m - \alpha(x^i_m - x^*_m)$
b) $x^*_m \leftarrow x^*_m + \alpha(x^i_m - x^*_m)$
end

$x^i_m \leftarrow x^i_m - \eta \times g^i_{m,t\_i}(x)$
$t^i \leftarrow t^i + 1$

Until Forever