KIPS Trans. Comp. and Comm. Sys.
Vol.10, No.7 pp.191~198
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2021.10.7.191

Hybrid All-Reduce Strategy with Layer Overlapping for Reducing Communication Overhead in Distributed Deep Learning

Daehyun Kim† · Sangho Yeo†† · Sangyoon Oh†††

ABSTRACT

Since the size of training dataset become large and the model is getting deeper to achieve high accuracy in deep learning, the deep neural network training requires a lot of computation and it takes too much time with a single node. Therefore, distributed deep learning is proposed to reduce the training time by distributing computation across multiple nodes. In this study, we propose hybrid allreduce strategy that considers the characteristics of each layer and communication and computational overlapping technique for synchronization of distributed deep learning. Since the convolution layer has fewer parameters than the fully-connected layer as well as it is located at the upper, only short overlapping time is allowed. Thus, butterfly allreduce is used to synchronize the convolution layer. On the other hand, fully-connecter layer is synchronized using ring all-reduce. The empirical experiment results on PyTorch with our proposed scheme shows that the proposed method reduced the training time by up to 33% compared to the baseline PyTorch.

Keywords : Distributed Deep Learning, Synchronization, Layer Overlapping, Allreduce

분산 딥러닝에서 통신 오버헤드를 줄이기 위해 레이어를 오버래핑하는 하이브리드 올-리듀스 기법

김 대 현† · 여 상 호†† · 오 상 윤†††

요 약

분산 딥러닝은 각 노드에서 지역적으로 업데이트한 지역 파라미터를 동기화는 과정이 요구된다. 본 연구에서는 분산 딥러닝의 효과적인 파라미터 동기화 과정을 위해, 레이어 별 특성을 고려한 allreduce 통신과 연산 오버래핑(overlapping) 기법을 제안한다. 상위 레이어의 파라미터 동기화는 하위 레이어의 다음 전파과정 이전까지 통신/계산(학습) 시간을 오버랩하여 진행할 수 있다. 또한 이미지 분류를 위한 일반적인 딥러닝 모델의 상위 레이어는 convolution 레이어, 하위 레이어는 fully-connected 레이어로 구성되어 있다. Convolution 레이어는 fully-connected 레이어 대비 적은 수의 파라미터를 가지고 있고 상위에 레이어가 위치하므로 네트워크 오버랩 허용시간이 짧고, 이를 고려하여 네트워크 지연시간을 단축할 수 있는 butterfly all-reduce를 사용하는 것이 효과적이다. 반면 오버랩 허용시간이 보다 긴 경우, 네트워크 대역폭을 고려한 ring all-reduce를 사용한다. 본 논문의 제안 방법의 효과를 검증하기 위해 제안 방법을 PyTorch 플랫폼에 적용하여 이를 기반으로 실험 환경을 구성하여 배치크기에 대한 성능 평가를 진행하였다. 실험을 통해 제안 기법의 학습시간은 기존 PyTorch 방식 대비 최고 33% 단축된 모습을 확인하였다.

키워드 : 분산딥러닝, 동기화, 레이어 오버래핑, 올리듀스

1. 서 론

최근 IoT 기술의 상용화로 생성된 빅데이터와 딥러닝 알고리즘 발전으로 인해, 딥러닝 학습의 효율과 정확성이 높아지고 있다. 이에 따라 많은 실제 응용 분야에서 딥러닝이 사용되고 있으며, 대표적으로는 음성 인식, 시각적 객체 인식 그리고 텍스트 처리 분야가 있다. 또한, 머신러닝 알고리즘의 적용이 가능한 머신러닝 플랫폼들 역시 혁신적인 발전이 이루어지고 있다.

최근 딥러닝의 학습 데이터 크기는 수 테라바이트(Terabyte) 급 이상으로 증가하고 있으며, 더 높은 정확도를 달성하기 위하여 더 복잡한(깊은) 딥러닝 모델을 사용하고 있다[1]. 이로 인하여 딥러닝은 대규모 연산이 요구되며[2,3], 단일 노드를

※ 이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단 기초연구사업 (2018R1D1A1B07043858)과 과학기술정보통신부/정보통신기획평가원의 대학ICT연구센터지원사업(IITP-2021-2018-0-01431)의 지원을 받아 연구되었음.
† 비 회 원 : LG CNS 스마트F&C사업부 사원
†† 비 회 원 : 아주대학교 인공지능학과 박사과정
††† 종신회원 : 아주대학교 소프트웨어학과 교수
Manuscript Received : December 28, 2020
Accepted : March 2, 2021
* Corresponding Author : Sangyoon Oh(syoh@ajou.ac.kr)

통해 딥러닝 알고리즘을 수행하는 경우에는 딥러닝 학습에 며칠 혹은 수주까지 걸리는 문제점이 발생한다[4].

이를 해결하는 방안으로 딥러닝 학습의 대규모 연산부하를 다수의 노드로 분산하여 학습하는 분산 딥러닝 개념이 주목을 받고 있고, 다양한 머신러닝 프레임워크들[5-8]에서 이러한 분산 딥러닝 기능을 지원하고 있다. 분산 딥러닝은 각 노드에서 파라미터를 지역적으로 업데이트하며 학습을 진행한다. 이로 인하여 각 노드의 지역 파라미터는 서로 다른 값을 가지게 되므로 이를 동기화하여 전역 파라미터로 업데이트하는 과정이 필수로 요구된다.

분산 딥러닝은 작은 미니배치(mini-batch)로 학습을 진행하여, 특정 데이터에 한하여 높은 오버피팅(overfitting) 되어 정확도가 낮아지는 문제가 발생한다. 단일 노드에서 학습하는 딥러닝에선 배치 정규화를 통해 이를 해결하지만, 분산 딥러닝에선 각 노드의 미니배치가 매우 작으므로 배치 정규화를 진행하여도 이를 해결할 수 없다.

딥러닝 알고리즘을 분산/병렬화 하는 방법은 크게 모델 병렬화(Model Parallelism), 데이터 병렬화(Data Parallelism) 기법으로 나눌 수 있다. Model Parallelism은 단일 노드에서 메모리 대역폭 혹은 연산능력의 한계로 인해 처리될 수 없는 모델을 분산된 노드에 나누어서 처리할 수 있다는 장점이 있지만, backward propagation(역전파)뿐만 아니라 forward propagation(순전파)에서도 추가적인 동기화를 위한 통신이 요구된다.

Data Parallelism 기법은 학습 데이터를 각 노드들이 나누어 학습을 진행하는 기법이다. Data Parallelism 기법에서 파라미터를 동기화하는 방법은 크게 파라미터 서버(Parameter Server)와 집합통신(Collective communication)으로 나눌 수 있다. 파라미터 서버는 직관적이며 구현이 단순하다는 장점이 있지만, 파라미터 서버로 통신 부하가 집중되어 네트워크 병목이 발생할 수 있다. 집합통신을 사용한 동기화 기법은 NCCL(NVIDIA Collective Communications Library)[9]과 MPI(Message Passing Interface)[10]와 같은 라이브러리에서 제공해준다. 기존의 대부분의 머신러닝 프레임워크들[5-8]에서 이러한 분산 딥러닝 동기화를 위한 집합통신을 제공하고 있다.

그러나, 머신러닝 프레임워크들에서 제공하는 집합통신은 딥러닝 모델의 전체 파라미터를 동기화하기 때문에 네트워크 대역폭이 전체 파라미터 수의 크기보다 작은 경우 네트워크 병목현상이 발생하여 파라미터 동기화 시간이 증가할 수 있다. 이러한 파라미터 동기화 시간을 단축하기 위해, 통신 부하를 고려하여 all-reduce 알고리즘 변경[11,12]하거나 계층적인 구조[13,14]로 통신을 진행하는 방법, 특정 파라미터만 통신하는 방법[15-17]들이 연구되었다.

Ring all-reduce 방식[11]은 파라미터를 분산 딥러닝에 참여하는 노드의 개수로 분할하여 분할된 개수만큼 reduce 연산을 수행한다. 따라서 이 방식은 분할된 파라미터를 통신함으로 네트워크 대역폭이 충분하지 못한 환경에서 효과적으로 동기화를 진행할 수 있고, 단계별 reduce 연산은 모든 노드에서 이루어지므로 기존 특정 노드(master 노드)로 하나에서 reduce 연산이 이루어져 발생하는 네트워크 병목을 완화할 수 있다. 하지만 네트워크 대역폭이 파라미터 수보다 큰 환경에서는 네트워크 병목이 일어나지 않으므로 이 기법에서의 성능향상은 일어나지 않는다. 또한, 이 기법은 노드의 개수만큼 선형적으로 통신 횟수가 증가하여 발생하는 네트워크 지연시간 증가에 대한 해결이 필요하다.

분산 딥러닝 동기화 향상을 위한 연구들[11-17]은 네트워크 대역폭 또는 통신 횟수의 개선을 중심으로 연구하였다. 하지만 레이어 별 파라미터 수와 통신 / 계산 오버랩 허용시간을 고려하지는 않는다. AlexNet 모델은 가장 적은 파라미터를 가진 레이어의 파라미터 수는 가장 많은 파라미터를 가진 레이어의 파라미터 수의 0.09% 밖에 되지가 않으며, 딥러닝의 역전파 과정은 chain rule(연쇄 법칙)로 계산이 되어 상위 레이어의 파라미터 동기화는 하위 레이어의 다음 전파과정 시간까지 통신/계산 오버랩 허용시간이 주어진다. 이에 따라 하위 레이어는 오버랩 허용시간이 적어 오버랩을 충분히 할 수 없지만 상위 레이어는 오버랩을 허용시간 내에 충분히 할 수 있다.

본 논문에서는 분산 딥러닝 동기화 성능 향상을 위해 딥러닝 모델의 레이어별 파라미터 수와 오버랩 허용시간에 따라 동기화 기법을 다르게 적용하여 학습을 진행하는 기법을 제안한다. 제안 기법에서는 레이어별 계산 과정과 통신 과정을 오버랩하여 분산 딥러닝의 네트워크 오버헤드를 완화하는 방법을 사용하였으며, 동시에 레이어의 파라미터 수와 오버랩 허용시간을 고려한 all-reduce 방식을 사용하여 분산 딥러닝의 동기화를 개선하는 방법을 사용하였다. 본 제안 기법의 성능평가를 위해 기존 분산 딥러닝의 기능을 제공하는 플랫폼에 적용하여 성능평가를 진행하였다.

본 논문은 구성은 다음과 같다. 2장에서는 분산 딥러닝 파라미터 동기화 과정의 성능을 위한 기존 연구 기법들을 소개한다. 3장에서는 분산 딥러닝에서 통신 오버헤드를 줄이기 위해 레이어를 오버래핑하는 하이브리드 올-리듀스 기법에 대해 설명하며, 제안한 기법이 갖는 이점 및 기존 방법과의 차이를 설명한다. 4장 실험에서는 제안 방법과 기존 방법의 성능 차이 비교를 통해 제안 방법의 효과를 검증한다. 본 연구 논문은 5장 결론을 통해 연구내용을 정리한다.

2. 관련 연구

최근 많은 딥러닝 응용에서 학습 속도를 향상시키기 위해 많은 노드를 사용하여 분산으로 학습을 진행하고 있다. 하지만, 분산 딥러닝에 참여하는 노드의 수가 증가할수록 파라미터를 동기화하는 시간이 증가하게 되는 문제가 발생한다. 최근 많은 연구자들이 딥러닝 모델의 학습 시간을 단축하기 위

한 분산 딥러닝 동기화 기법들을 제안하였다. 본 장은 분산 딥러닝의 성능을 높이기 위해 파라미터 동기화과정을 최적화하는 다양한 연구를 요약한다.

2.1 분산딥러닝을 위한 Allreduce 알고리즘 개선 연구

기존 MPI 기반의 all-reduce는 분산 딥러닝 파라미터 동기화 연산을 진행할 때 각 worker가 파라미터 전체를 마스터 노드에게 전송하기 때문에 마스터 노드의 통신부하가 노드의 개수에 비례하여 증가하게 되는 문제가 있다. 이를 해결하기 위해, 최근 많은 연구들은 효율적인 대규모 클러스터 환경의 활용을 위하여 all-reduce의 통신부하가 마스터 노드로 집중되는 문제를 해결하거나 딥러닝 모델의 많은 파라미터를 고려하여 네트워크 대역폭을 최적화하는 목표로 한다.

baidu사의 연구[11]는 딥러닝 모델의 많은 수의 파라미터를 고려하여 네트워크 대역폭에 최적화된 ring all-reduce 방식을 제안하였다. 이 방식은 통신할 메시지를 통신에 참여하는 노드의 개수로 나누어 단계별로 집합 통신을 수행하는 통신 방식이다. 이 방식의 reduce 연산은 매 단계에서 모든 노드에서 이루어지므로 기존 특정 노드(master 노드)로 하나에서 reduce 연산이 이루어지기 때문에 발생하는 네트워크 병목을 완화할 수 있다. 또한 이 방식은 메시지를 노드의 수로 나누어서 진행하므로 all-reduce에 쓰이는 메시지의 수가 네트워크 대역폭보다 커 네트워크 대역폭이 부족한 환경일 때 효과적이다. 하지만, 메시지의 수가 네트워크 대역폭보다 작은 경우에 대해서는 네트워크 대역폭을 효율적으로 사용하지 못한다. 또한, 노드의 개수가 늘어날수록, 통신 횟수가 노드 수인 N 에 관하여 (1)과 같이 선형 증가하여 통신 시간이 증가하게 되므로 이에 대한 해결이 필요하다.

Thakur et al.[12]의 연구는 네트워크 지연시간을 고려한 butterfly 형식의 all-reduce 방식을 제안하였다. 이 방식의 통신 과정은 각 단계마다 reduce-scatter 연산을 수행한 후, all-gather 연산을 수행한다. 각 단계인 i, 인덱스 x를 가진 노드 $Worker\,(x)$와 전체 노드 수인 N에 관하여 단계별 reduce 연산은 $Worker\,(x)$와 $Worker\,(x+N/2^i)$의 노드에서 이루어지고, $Worker\,(x)$ 노드가 바로 옆 노드 $Worker\,(x+1)$와 통신할 때까지 단계를 진행한다. 따라서 이 방식의 통신 횟수는 노드 수인 N에 관하여 (2)과 같이 나온다. 이 방식은 위에서 언급한 ring all-reduce[11]의 통신 횟수보다 더 적은 수를 통신 횟수를 가진다는 이점이 있다. 하지만 이 방식의 모든 통신 단계에선 전체 메시지를 전달하여 reduce 연산을 수행하기 때문에, 네트워크 대역폭이 충분하지 않은 상황에선 네트워크 오버헤드가 증가한다.

Xianyan et al.[13]의 연구는 Multi-GPU technology (다중 그래픽 처리 기술) 환경에서 all-reduce 연산을 최적화하기 위해 계층적으로 통신하는 hierarchical all-reduce를 제안하였다. 이 방식은 먼저 동일한 그룹 내의 PCIe[18] 또는 NVLink[19]로 연결된 GPU 들이 reduce 연산을 수행한다. 그 후, reduce의 연산 결과를 가진 각 그룹의 마스터 GPU들은 ring all-reduce를 수행한다. 마지막으로 각 그룹의 마스터 GPU는 전체 노드들에 대해 all-reduce된 파라미터 값을 가지므로 파라미터 값을 그룹 내 존재하는 다수의 GPU로 전파한다. 이러한 통신 방식은 노드 N개와 그룹 P개일 때 통신 횟수가 (3)로 앞서 살펴본 ring all-reduce[11]의 횟수보다 더 적어진다. 하지만 계층 간 통신에 사용되는 마스터 GPU의 네트워크 병목현상이 발생하는 문제가 있다.

$$RingCount_{comm} = 2(N-1) \tag{1}$$

$$ButterflyCount_{comm} = 2\log(N) \tag{2}$$

$$HierarchicalCount_{comm} = 4(N-1)+2(P/N-1) \tag{3}$$

2.2 분산딥러닝 동기화 최적화 연구

Tensor fusion 전략[15]은 동기화 과정에서 작은 크기의 파라미터를 한 버퍼 공간에 복사한 후, 사용자가 설정한 threshold 만큼의 버퍼가 채워져야만 동기화를 수행한다. 이로 인하여 tensor fusion은 네트워크 대역폭을 효율적으로 활용할 수 있다. 이 전략은 추가적으로 버퍼에 메모리를 복사하는 작업을 수행해야 하므로 메모리 소모가 높다.

Lin et al.[16]의 연구는 네트워크 트래픽을 줄이기 위해 fine-grained sparse communication(미시적 희소통신)를 사용하여 threshold 보다 큰 파라미터만 선택하여 통신하는 심층 기울기 압축(deep gradient compression)기법을 제안하였다. 이 기법은 정확성이 저하되지 않고 최대 99.9% 네트워크 트래픽을 감소시켰다. 하지만 이 기법은 sparse 데이터의 연산으로 인하여 계산 비용이 GPU 환경에 적합하지 않다.

Sun et al.[17]의 연구는 tensor fusion 전략[15]의 높은 메모리 소모 문제를 해결하기 위해 lazy all-reduce 전략을 제안했다. lazy all-reduce는 업데이트된 파라미터에 대해 즉시 all-reduce를 수행하는 대신 메모리 풀에 배치한다. 메모리 풀에 축척 된 파라미터 크기가 설정한 threshold 보다 클 때까지 다음 레이어의 파라미터의 축척을 대기한다. 이후 threshold를 초과한 경우 메모리 풀에 대기 중인 모든 파라미터에 대해 단일 all-reduce 작업을 수행한다. 이로 인하여 추가 메모리 복사 없이 메모리 풀에서 all-reduce를 직접 수행하여 tensor fusion의 높은 메모리 소모 문제를 완화하였다. 하지만 [15,17]의 연구 모두에서 파라미터 축적과정은 초반부 레이어의 동기화를 지연시키므로 다음 순전파 과정 시작이 지연될 수 있다. 또한, Sun et al.[17]의 연구는 L1 정규화(normalization)를 통해 중요한 파라미터를 선택하여 all-reduce 연산을 진행하는 coarse-grained sparse communication (거시적 희소 통신)을 제안하였다. 이 기법은 [16] 기법에서 계산

비용이 높다는 문제를 L1 정규화를 적용함으로써 해결하였다. 하지만 단순히 L1 정규화 값이 큰 파라미터만 통신하는 것은 정확도가 저하될 수 있고, 통신할 파라미터의 비율을 Heuristic 알고리즘에 비율이 정해지므로, 비율의 일관성을 보장하지 않는다.

3. 레이어 오버래핑 기반 하이브리드 올리듀스 기법

본 연구에서는 기존의 분산 딥러닝의 효과적인 파라미터 동기화 과정을 위한 레이어 별 특성을 고려한 all-reduce 및 통신과 연산 오버래핑(overlapping) 기법을 제안한다. 본 제안의 레이어별 동기화 기법은 정확도 성능에 영향을 미치지 않고 결과를 낼 수 있도록 역전파 과정의 순서에 따라 레이어별 동기화를 진행한다.

본 장에서는 제안하는 딥러닝의 효과적인 파라미터 동기화 방법에 대해 자세히 설명한다. 먼저 딥러닝 모델의 레이어 간의 순전파 과정과 역전파 과정에서의 영향을 분석하고, 이를 기반으로 레이어별 동기화 기법과 통신과 연산 오버래핑 기법을 설명한다. 또한 딥러닝 모델의 레이어별 파라미터 수와 오버랩 허용시간을 고려한 Hybrid Communication 기법을 제안한다. 또한, 본 제안이 딥러닝 모델의 특징에 따라 발생 가능한 본 제안에서의 향후 개선점에 대해 후술한다.

3.1 레이어별 계산/통신 오버랩 동기화 기법

본 제안은 레이어의 파라미터를 메모리에 복사하여 통신을 진행하지 않으므로 기존 방식 대비 메모리 소모가 낮다. 본 절에서는 동기화 시간을 단축하기 위한 레이어별 동기화 기법과 계산/통신을 오버랩하는 기법을 설명하며, 동시에 이로 인해 발생 가능한 오버헤드를 설명한다.

1) 레이어별 동기화 기법

Fig. 1은 기존 머신러닝 플랫폼의 분산 딥러닝 동기화 과정을 보여준다. 기존 머신러닝 플랫폼에서 딥러닝의 역전파 과정은 효율적으로 연쇄 법칙(chain rule)을 계산하기 위해 전체 레이어의 파라미터 값을 하나의 메모리 공간에서 계산한다. 이로 인하여 분산 딥러닝 파리미터 동기화 과정은 같은 메모리 공간에 있는 딥러닝 모델의 전체 파라미터를 일시에 동기화 하여 네트워크 병목이 발생할 수 있다.

본 레이어별 동기화 기법은 동기화 시간을 단축시키기 위하여 레이어별 동기화를 진행한다. 역전파 과정은 하위 레이어부터 순서를 나타내며 역전파 과정이므로 레이어의 역순으로 진행된다. 역전파 과정은 chain rule(연쇄 법칙)로 계산이 되어 특정 레이어의 파라미터는 상위 레이어의 파라미터의 영향을 받지만, 하위 레이어의 파라미터엔 영향을 받지 않는다. 제안하는 기법에서의 딥러닝 모델 학습에선 상위 레이어의 역전파 과정을 완료하면 해당 레이어의 파라미터가 즉시 업데이트 된다. 이로 인하여 전체 학습 파라미터를 동기화하여 생기는 네트워크 지연시간에 따른 오버헤드를 완화할 수 있다. 하지만 ResNet50[18]와 같은 적은 수의 파라미터를 가진 레이어(예: convolution 레이어)가 많은 경우 네트워크 대역폭을 충분히 활용하지 못하고, 잦은 네트워크 커넥션으로 인해 동기화 시간이 늘어난다.

2) 계산/통신 오버래핑 기법

본 제안 기법은 위에서 제안한 레이어별 동기화 과정을 전파, 역전파 과정과 오버래핑하여 진행하여 동기화 시간을 단축한다. Fig. 1의 역전파 과정의 $Layer_n$의 역전파 과정이 완료된 시간 B_n에서 $Layer_n$의 파라미터는 다음 전파 과정의 $Layer_n$의 시작 시간 F_n까지 딥러닝 학습 계산에 이용되지 않는다. 따라서 $Layer_n$의 파라미터의 오버랩 허용시간은 다음 단계에서 $Layer_n$의 전파과정인 시작 시간에서 역전파 과정이 완료된 시간을 뺀 $F_n - B_n$이다. 본 제안은 $Layer_n$의 파라미터 동기화를 F_n 시간까지 하위 레이어의 역전파 과정과 상위 레이어의 전파 과정과 오버랩하여 진행한다. 그러나 상위에 위

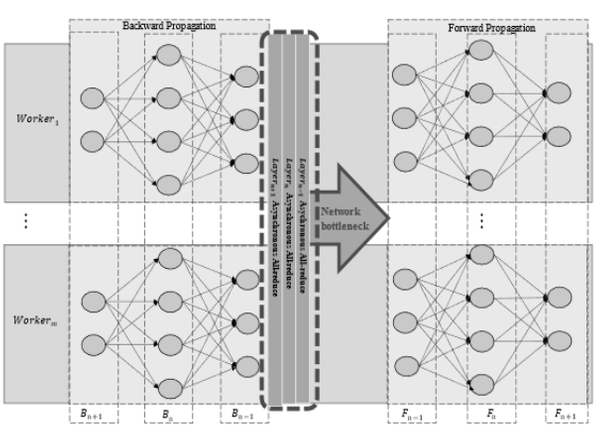

Fig. 1. Distributed Deep Learning Synchronization Process of Existing Machine Learning Platform

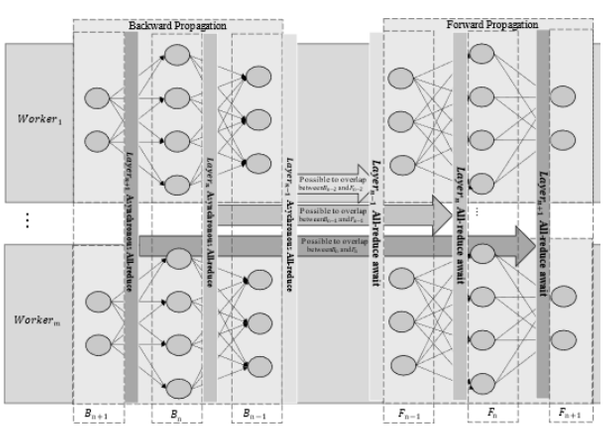

Fig. 2. Per Layer Synchronization with Overlapping Calculation/Communication Processes

Table 1. Number of Parameters in AlextNet[19] Model

	Number of Parameters	Payload
Conv_1	34,944	0.06%
Conv_2	614,656	0.99%
Conv_3	885,120	1.42%
Conv_4	1,327,488	2.13%
Conv_5	884,992	1.42%
F.C_1	37,752,832	60.52%
F.C_2	16,781,312	26.90%
F.C_3	4,097,000	6.57%

치하는 레이어들은 오버랩 허용시간이 적고, 최상위 레이어는 오버랩 허용시간이 거의 없다. 오버랩 허용시간이 적은 상위에 위치한 레이어들이 파라미터의 수가 많다면 오버랩이 되지 않아 학습 시간이 단축되지 않는다. Fig. 2는 레이어별 동기화 기법과 계산/통신 기법을 적용한 학습 과정을 보여준다.

3.2 레이어의 특성에 따른 Hybrid Communication 기법

본 제안에서는 레이어별 계산/통신 오버랩 허용시간과 파라미터의 크기를 고려하여 서로 다른 All-reduce 알고리즘을 적용하게 된다. 일반적으로 이미지 분류를 위한 딥러닝 모델[19-21]은 특징을 추출하는 convolution 레이어는 상위 레이어로, fully-connected 레이어는 하위 레이어로 구성된다. 딥러닝 모델의 convolution 레이어는 상위에 위치하므로 동기화 통신 과정과 계산 과정을 오버래핑할 수 있는 허용시간이 부족하다. 또한 convolution 레이어는 필터를 사용하여 특징을 추출하므로 fully-connected 레이어에 비해 적은 수의 파라미터를 갖고 있다. 이에 따라 기존 연구에서 제안한 네트워크 대역폭을 고려하여 동기화에 사용하는 통신을 여러 단계로 나누어 진행하는 기법[11]을 convolution 레이어에 적용하게 되면 동기화 시간이 증가한다.

Table 1은 AlextNet 모델의 레이어별 파라미터 수와 payload를 나타낸다. 따라서 Hybrid Communication 기법은 초반부 convolution 레이어는 네트워크 지연시간에 최적화된 butterfly all-reduce[12] 알고리즘을 적용하여 동기화하고, 후반부 Fully-connected 레이어는 네트워크 대역폭에 최적화된 ring all-reduce[11] 알고리즘을 적용하여 동기화 한다.

본 제안에서 사용한 butterfly all-reduce[12] 알고리즘의 통신단계 수인 $2\log(N)$은 ring all-reduce의 통신인 $2(N-1)$보다 적으므로, butterfly all-reduce 알고리즘은 네트워크 대역폭을 초과하지 않는 레이어라면 ring all-reduce의 비해 통신단계 수 경감에 따른 동기화 시간을 단축을 기대할 수 있다. 하지만 fully-connected 레이어와 같은 파라미터 수가 많은 레이어에 butterfly all-reduce 알고리즘을 사용하여 동기화를 진행하게 되면 네트워크 대역폭을 초과하여 네트워크 병목현상이 발생할 수 있다. 이는 모든 레이어에 동기화 과정을 ring all-reduce 알고리즘을 적용한 기존 기법보다 동기화 시간이 증가하게 된다. 따라서 위 제안은 딥러닝 모델에 상위 레이어가 convolution 레이어인 경우를 위해 제안되어 상위 레이어가 convolution 레이어가 아닌 경우 네트워크 대역폭을 초과하여 발생하는 네트워크 병목현상으로 인해 늘어난 동기화 시간을 보여 비효율적이다. Fig. 3은 레이어별 동기화 기법을 적용하고, Hybrid Communication 기법 적용하여 동기화 하는 과정을 보여준다.

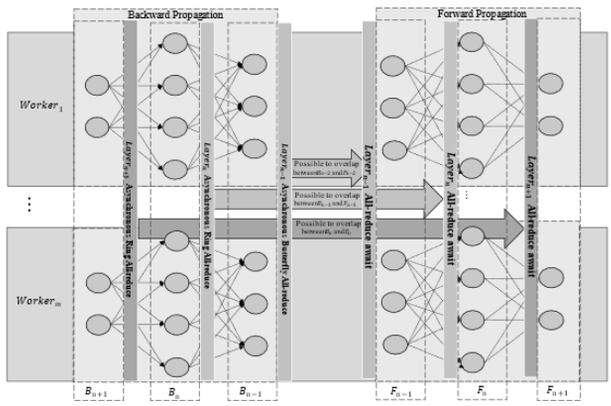

Fig. 3. Hybrid Communication Technique

4. 성능 평가

본 장에서는 제안한 레이어 오버래핑 기반 하이브리드 올-리듀스 기법을 PyTorch[7] 플랫폼에 적용하여 AlextNet 모델을 학습하는데 소요되는 지연시간(즉, 100 epoch까지 학습하는데 소요되는 시간)을 측정하고, 이를 기존 PyTorch[7]의 분산 딥러닝과 비교하여 성능 평가를 진행한다. 이를 통해 제안기법의 파라미터를 통신하여 발생할 수 있는 네트워크 오버헤드를 완화하는 효과를 확인할 수 있다.

4.1 실험 환경 및 조건

1) 실험 환경

본 실험의 진행을 위해 4개의 동일 컴퓨팅 성능을 가진 노드로 소규모 클러스터를 구성하였다. 클러스터 구성을 위해 사용된 각 노드의 컴퓨팅 자원의 상세한 사양은 Table 2와 같다.

2) 분산 딥러닝 환경

딥러닝 모델의 주요 파라미터 설정 값은 Table 3과 같다. 일반적인 이미지 분류를 위한 딥러닝 모델들에서 제안 기법의 성능을 향상하기 위해, 상위 레이어는 특징을 추출하는 convolution 레이어로 하위 레이어는 fully-connected 레이

Table 2. Experiment Environment

The Number of Nodes	4
CPU	Intel i7-8700, 3.20 GHz
Memory(GB)	32G
Storage(TB)	1TB (HDD)
Network Bandwidth(Gbps)	1Gbps
Operating System	Ubuntu 18.04

Table 3. Deep Learning Environment

The Number of Epochs	100
Model	AlexNet[19]
DataSet	CIFAR10
Batch Size	128, 192, 256, 512
Learning Rate	0.1

Table 4. Communication Method for Each Layer of AlexNet

	PyTorch[7]	Hybrid Communication
Conv_1	Ring all-reduce[11]	Ring all-reduce[11]
Conv_2		
Conv_3		
Conv_4		
Conv_5		
F.C_1		Butterfly all-reduce[12]
F.C_2		
F.C_3		

어로 구성되어있는 AlexNet[19] 모델을 사용하였다. AlexNet은 5개의 convoultion 레이어와 3개의 fully- connected 레이어로 구성되어 있다. 또한, 배치 크기에 따른 성능을 비교하기 위해, 배치 크기를 다르게(예: 128, 192, 256, 512) 설정하여 실험을 진행하였다.

Table 4는 실험에서 사용한 AlexNet의 각 레이어별 파라미터 수와 기존 기법과 제안 기법의 동기화를 위한 집합 통신 기법을 나타낸다. Conv_1 – Conv_5 레이어의 파라미터 수가 F.C_1 – F.C_3 레이어의 파라미터 수에 비해 매우 적고 오버랩 허용시간 부족하기 때문에 제안 기법에서는 ring all-reduce[11]로 사용하여 동기화를 진행하였다.

4.2 실험 결과

본 실험 결과에선 배치 크기에 따른 기존 기법인 PyTorch[7]와 제안 기법의 총 학습 시간과을 비교하고 이를 분석한다.

1) 배치 크기에 따른 총 학습 시간

Fig. 4는 PyTorch와 제안 기법의 배치 크기가 128, 192, 256, 512일 때 총 학습 시간을 그래프로 나타낸다. 배치 크기가 128일 때 학습시간은 제안 기법이 기존 기법보다 약 7.6% 시간을 단축시켰고, Batch 크기가 512 일 때 제안 기법이 기존 기법 보다 약 33% 시간을 단축시켰다. 배치 크기가 증가할수록 제안 기법이 기존 기법에 비해 더 적은 시간으로 학습을 할 수 있는 이유로는 크게 다음과 같다. 배치 크기가 증가함에 따라 계산량이 증가하여 계산 / 통신 오버랩 허용시간이 늘어나고, 상위 레이어의 동기화 시간이 단축되어 다음 순전파가 더 이르게 시작하기 때문이다.

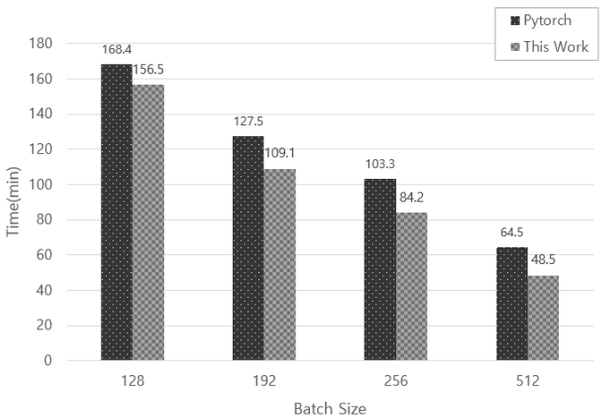

Fig. 4. Comparison of Execution Time

5. 결론 및 향후 연구

본 논문에서는 분산 딥러닝의 동기화 과정에서 발생할 수 있는 네트워크 오버헤드를 완화하기 위하여 레이어 오버래핑 기반 하이브리드 올-리듀스 기법을 제안하였다. 본 논문의 제안 기법은 딥러닝 모델의 레이어별 파라미터 수와 오버랩 허용시간을 고려하여 ring all-reduce, butterfly all-reduce로 동기화하고, 역전파 과정이 끝난 레이어를 즉시 오버랩하여 동기화한다. 실험 결과에서 제안 기법을 동기화에 적용한 AlexNet 모델의 분산 딥러닝 학습 시간은 PyTorch 대비 평균 약 20% 증가하는 모습을 확인하였다. 제안 기법은 배치 크기가 증가할수록 동기화에서 네트워크 오버헤드가 발생하여 PyTorch보다 향상되는 정도가 함께 커졌다.

하지만 제안 기법은 AlexNet[19]과 같은 경량화된 모델에서는 효과를 기대할 수 있지만 ResNet[18]과 같은 작은 convolution 레이어가 많이 구성된 모델 또는 자연어처리를 위한 BERT[22]나 XLNet[23] 모델에서는 효과를 기대하기 어렵다. 따라서 향후에는 이러한 딥러닝 모델에도 특징에 맞는 동기화를 진행하여 네트워크 오버헤드를 완화하는 연구를 진행하고자 한다. 또한, 본 논문에서는 4개의 노드로 구성된 클러스터를 통해 실험을 진행하였으며, 노드 수가 증가함에 따른 추가 실험을 통해 확장성을 보여주는 연구를 진행하고자 한다.

References

[1] S. Teerapittayanon, B. McDanel and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *IEEE International Conference on Distributed Computing Systems, Atlanta*, pp.328-339, 2017.

[2] X. W. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, Vol. 2, pp.514-524, 2014.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, Vol.521, pp.436-444, 2015.

[4] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, Vol.2, No.1, 2015.

[5] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, and M. Kudlur, "Tensorflow: A system for large-scale machine learning," in *USENIX Symposium on Operating Systems Design and Implementation*, Savannah, pp.265-283, 2016.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*, pp.675-678, 2014.

[7] R. Collobert, K. Kavukcuoglu and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, No. CONF, 2011.

[8] F. Seide and A. Agarwal, "CNTK: Microsoft's open-source deep-learning toolkit," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.2135-2135, 2016.

[9] NVIDIA Developer, NCCL, [Internet] https://developer.nvidia.com/nccl.

[10] The Software in the Public Interest non-profit organization, Open MPI, [Internet] https://www.open-mpi.org/.

[11] Baidu Research, Ring all-reduce, [Internet] https://github.com/baidu-research/baidu-allreduce.

[12] R. Thakur, R. Rabenseifner, and, W. Gropp, "Optimization of collective communication operations in MPICH," *The International Journal of High Performance Computing Applications*, Vol.19, No.1, pp.49-66, 2005.

[13] X. Jia, S. Song, W. He, Y. Wang, H. Rong, F. Zhou, L. Xie, Z. Guo, Y. Yang, L. Yu, G. Hu, S. Shi, X. Chu, and T. Chen, "Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes," arXiv preprint arXiv:1807.11205, 2018.

[14] H. Mikami, H. Suganuma, Y. Tanaka, and Y. Kageyama, "ImageNet/ResNet-50 Training in 224 Seconds," arXiv preprint arXiv:1811.05233, 2018.

[15] A. Sergeev and M. Del Balso, "Horovod: Fast and easy distributed deep learning in TensorFlow," arXiv preprint arXiv:1802.05799, 2018.

[16] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," arXiv preprint arXiv:1712.01887, 2017.

[17] P. Sun, Y. Wen, R. Han, W. Feng, and S. Yan, "GradientFlow: Optimizing Network Performance for Large-Scale Distributed DNN Training," *IEEE Transactions on Big Data*, 2019.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, pp.770-778, 2016.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, Vol.60, No.6, pp.84-90, 2017.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, pp1-9, 2015.

[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, pp.4700-4708, 2017.

[22] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[23] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing Systems*, pp.5753-5763, 2019.

김 대 현

https://orcid.org/0000-0003-4271-7395
e-mail : eogussla12@gmail.com
2013년 ~ 2019년 아주대학교 소프트웨어학과(학사)
2020년 ~ 2021년 아주대학교 인공지능학과(석사)
2021년 ~ 현 재 LG CNS 스마트F&C사업부 사원
관심분야 : 분산딥러닝

여 상 호

https://orcid.org/0000-0002-9194-7552
e-mail : soboru963@ajou.ac.kr
2017년 아주대학교 소프트웨어학과(학사)
2017년 ~ 현 재 아주대학교 인공지능학과 박사과정
관심분야 : 분산딥러닝, 강화학습

오 상 윤

https://orcid.org/0000-0001-5854-149X
e-mail : syoh@ajou.ac.kr
2006년 미국 인디애나대학교 컴퓨터공학과(박사)
2006년 ~ 2007년 SK텔레콤 전략기술부문
2007년 ~ 현 재 아주대학교 소프트웨어학과 교수
관심분야 : 분산딥러닝, 고성능컴퓨팅, 빅데이터 처리