분산 학습 가속화를 위한 프로그래머블 스위치 기반의 인-네트워크 병합 연구 동향

김희원, 한수빈, 이호찬, 한솔, 백상헌
고려대학교

{harry0475, subin993, ghcks1000, hs1087, shpack}@korea.ac.kr

A Study on In-network Aggregation Using Programmable Switch for Accelerating Distributed Deep Learning

Heewon Kim, Subin Han, Hochan Lee, Sol Han, Sangheon Pack
Korea Univ.

요 약

일반적으로 많은 데이터를 병렬적으로 학습하기 위해 파라미터 서버 기반의 분산 심층 학습이 사용되고 있지만 트래픽이 파라미터 서버로 집중되면서 네트워크 혼잡이 발생하여 통신 비용이 크게 발생하는 문제점이 발생하고 있다. 최근 프로그래머블 스위치를 통해 패킷이 전송되는 중간에 데이터를 읽고 병합할 수 있는 인-네트워크 병합에 대한 연구가 활발히 진행되면서 이를 분산 심층 학습에 적용해 학습 중간에 발생하는 gradient 패킷을 병합함으로써 불필요한 트래픽을 줄이고 학습 처리량을 높이는 연구가 진행되고 있다. 본 논문은 최근까지 진행된 인-네트워크 병합을 통한 분산 심층 학습 가속화에 대한 연구들을 소개 및 분석하고 앞으로의 연구 방향성을 제시한다.

Ⅰ. 서 론

인공 신경망을 기반으로 한 심층 학습이 이미지 처리, 언어 식별 등 다양한 분야에 활용되면서 더 정교한 학습을 위해 심층 학습에 사용되는 학습 데이터가 방대해지고 있다. 이에 따라 데이터를 한 기기에서만 처리하는 것이 아니라 다중 기기에 데이터를 나누어 학습을 시키고 발생한 gradient 를 파라미터 서버를 통해 합쳐주는 파라미터 서버 기반의 데이터 병렬화 분산 심층 학습이 주를 이루고 있다. 하지만 다수의 기기에서 나온 gradient 를 소수의 파라미터 서버로 보내 합치는 방식은 트래픽을 파라미터 서버로 집중시켜 병목 현상이 일어나게 한다. 최근 GPU 의 처리 속도가 비약적으로 발전하면서 한 기기 내에서의 학습 속도 향상이 두드러지게 나타났지만 이에 비해 네트워크 장비들의 패킷 처리 속도 향상이 따라가지 못하면서 분산 심층 학습의 주된 비용이 계산 비용에서 통신 비용으로 옮겨가고 있다[1].

분산 학습에서의 통신 비용을 줄이고자 gradient 를 양자화 시키거나 계산 과정과 통신 과정을 중첩하여 scheduling 하는 기법들이 제안되었지만 모두 정확도와의 trade-off 가 발생하거나 전반적인 트래픽 양은 줄이지 못하는 등의 한계점이 존재했다. 최근 네트워크 스위치의 data plane 을 프로그래밍 할 수 있는 프로그래머블 스위치가 개발되면서 맵-리듀스 동작의 일부인 리듀스 동작을 스위치에서 수행해 종단 노드에서의 리듀스 시간과 트래픽 양을 크게 줄인 DAIET[2] 기법이 제안되었다. 이로 인해 인-네트워크 병합 기술이 분산 환경의 트래픽을 줄일 수 있는 방안으로 주목을 받게 되었고, 이를 분산 심층 학습에 적용하고자 하는 시도가 이어지게 되었다. 분산 심층 학습에 인-네트워크 병합을 적용하게 되면 각 기기에서 발생되는 gradient 패킷을 스위치 내부에서 병합하여 트래픽의 양을 크게 줄일 수가 있게 되고 결과적으로 학습 처리량을 증가시킬 수 있게 된다.

본 논문은 인-네트워크 병합을 이용한 분산 심층 학습을 구현해낸 SwitchML[3]과 단일 서비스를 기반으로 한 기존 연구를 확장해 인-네트워크 병합을 이용한 분산 심층 학습을 다중 랙으로 구성된 다중 서비스 환경에서 사용할 수 있도록 구현한 ATP[4]를 대표적으로 살펴보고 이를 통해 인-네트워크 병합을 통한 분산 심층 학습 가속화 연구의 방향성을 논의하고자 한다.

Ⅱ. 분산 학습 가속화를 위한 인-네트워크 병합

(1) 인-네트워크 병합

인-네트워크 병합은 프로그래머블 스위치를 이용해 패킷의 데이터를 읽어와 이전에 스위치 내부에 저장된 데이터와 합치는 것을 의미하며 스위치에서 패킷 데이터를 읽어 들여 여러 계산을 수행하는 인-네트워크 컴퓨팅 기술의 일종이다. 원래 네트워크 종단에서 일어나야 할 병합을 전송 과정에서 해주는 것이기 때문에 병합 과정이 pipelining 되어 병합에 걸리는 전체적인 시간이 감소하게 되며 또한 인-네트워크 병합 과정에서 패킷의 데이터를 읽고 필요 없는 대부분의 패킷은 버리기 때문에 전체 트래픽의 양이 크게 줄어들게 된다. 하지만 인-네트워크 병합을 분산 심층 학습에 바로 적용하기에는 한계가 존재하는데, 이를 정리하면 다음의 세가지와 같다. 첫 번째는 스위치 메모리 용량이 한정되어 있기 때문에 모든 gradient 를 한 번에 저장할 수 없다는 점이고, 두 번째는 스위치 내부에서 floating point 연산을 수행할 수 없기 때문에 floating point 으로 이루어진 gradient 를 별도로 가공하지 않으면 스위치 내부에서 연산을 할 수가 없다는 점이며, 세 번째로는 인-네트워크 병합 후 남은 패킷을 버리기 때문에 기존 네트워크의 종단 간 연결성에서 벗어나게 되고, 이로 인해 ACK 을 기반으로 동작하는 기존 TCP 알고리즘을 사용하지 못하기 때문에 새로운 네트워킹 스택이 필요하게 된다는 점이다.

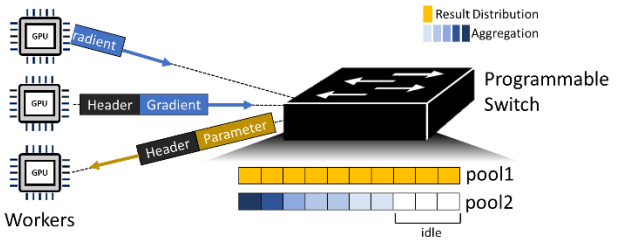

그림 1. Pool 기반의 SwitchML 병합 구조

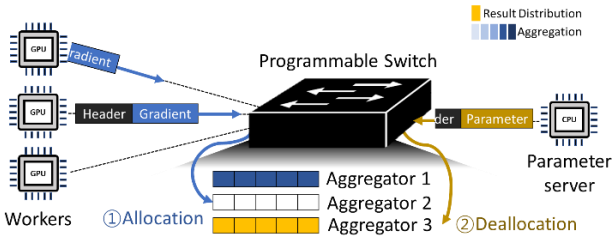

그림 2. ATP 의 aggregator 할당 및 해제 과정

(2) SwitchML

SwitchML 은 네트워크 스위치 내부에서의 gradient 병합을 구현해 스위치가 파라미터 서버의 역할을 대신할 수 있다는 것을 보여주었다. 이 기법은 스위치 메모리 용량의 한계로 모든 gradient 를 한 번에 저장할 수 없는 문제를 해결하기 위해 pool 기반의 스트리밍 방식의 병합을 사용하였다. 그림 1 은 2 개의 pool 을 통해 병합하는 과정을 나타낸 그림이다. Pool 은 여러 스위치 레지스터를 묶은 저장 공간인데, gradient 패킷이 들어오면 gradient segment 에 맞는 크기로 pool 의 일부를 할당한다. 이때 2 개의 Pool 은 서로 상호보완적으로 동작하게 되는데, 한쪽 pool 의 병합이 완료되면 이후의 병합은 다른 pool 에서 진행하고 자신은 병합된 결과인 parameter 패킷을 분배한 후에 메모리를 초기화하도록 동작한다. 다음으로 해당 기법은 스위치에서 연산이 가능하도록 gradient 에 일정 크기의 scaling factor 를 곱해주고 정수부분을 보내는 방법을 사용했다. 하지만 이런 방법은 변환 과정에서 loss 가 발생하여 정확도 성능이 떨어질 수 있는데, 이 기법은 scaling factor 의 크기를 변화시키면서 그때마다의 학습 정확도를 테스트해서 정확도에 거의 영향을 주지 않는 크기를 구해 놓았다. 마지막으로는 parameter 패킷을 ACK 으로 간주하여 일정 시간이 지나도록 보낸 gradient segment 에 해당하는 parameter 패킷이 도착하지 않으면 packet loss 가 발생한 것으로 간주하고 재전송을 해주는 timeout 기반의 reliability scheme 을 제안하였다. 하지만 실제 분산 심층 학습이 다중 스위치를 가진 복잡한 구조의 data center 내에서 다중 학습을 동시다발적으로 진행한다는 것을 고려할 때, SwitchML 은 단일 랙 스위치를 통한 단일 모델의 학습을 가정했다는 점에서 한계점을 가지고 있다고 말할 수 있다.

(3) ATP

ATP 는 인-네트워크 병합을 적용한 분산 심층 학습이 실제 data center 내에 적용되기 위해서는 다중 랙 스케일에서의 다중 서비스를 고려해야 함을 연구 동기로 삼았다. 먼저 SwitchML 과 달리 다중 스위치 환경에서 종단의 랙 스위치를 통해 병합이 일어나고 별도의 파라미터 서버에서 최종적인 병합이 일어나게 했다. 또한 다중 서비스를 고려하였을 때 많은 학습 모델이 각자 경쟁적으로 스위치 메모리를 선점하게 될 것이기 때문에 스위치 메모리를 동적으로 할당하여 더 효율적으로 메모리를 사용하도록 구현하였다. 그림 2 는 ATP 의 동적 메모리 할당 과정을 나타내고 있다. ATP 는 gradient segment 에 맞춰 메모리 공간의 단위를 재정의 하였는데, 이를 aggregator 라고 부른다. Gradient 패킷이 도착하게 되면 ATP 헤더에 존재하는 index 에 따라 몇 번째 aggregator 에 접근하게 될지가 결정되고 이때 만약 해당 aggregator 가 비어 있다면 할당된다 (①). 이후 파라미터 서버에서 최종적인 병합이 완료되면 이를 각 worker 로 분배해주게 되는데, parameter 패킷을 받은 스위치는 이를 통해 해당 aggregator 의 병합이 종료되었다는 것을 인지하고 할당을 해지하게 된다 (②). 마지막으로 ATP 는 새로운 인-네트워크 병합에 맞는 네트워킹 스택을 제안하였는데, timeout 기반이 아닌 gradient 순서와 다른 패킷 3 개가 연속해서 들어오면 이를 packet loss 로 간주하는 reliability scheme 을 사용하였으며 또한 ECN 마크와 parameter 패킷을 통한 slow start 기반의 congestion control 을 사용하여 효과적으로 congestion 을 제어하고 외부 트래픽과도 공평하게 네트워크 자원을 사용하는 것을 보여주었다.

Ⅲ. 결 론

분산 학습에서의 네트워크 병목 문제를 해결하기 위한 여러 시도의 한 방법으로 인-네트워크 병합이 떠오르면서 이를 분산 심층학습에 적용한 SwitchML 이 제안되었고, 이를 다중 랙에서의 다중 서비스 환경을 고려해 확장시킨 ATP 가 제안되었다. 현재 인-네트워크 병합을 이용한 분산 심층 학습 기법이 더 복잡한 상황을 가정한 실제 data center 에서도 적용될 수 있도록 하는 방향으로 연구가 진행되고 있다. 이를 구현하기 위해 한정된 스위치 자원을 우선순위에 따라 더 효율적으로 사용하기 위한 방법과 bursty 한 트래픽이 주기성을 가지고 발생하는 분산 심층 학습의 특성을 고려한 네트워킹 스택 알고리즘에 대한 심층적인 연구를 진행할 계획이다.

ACKNOWLEDGMENT

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터지원사업의 연구결과로 수행되었음 (IITP-2021-2017-0-01633)

참고 문헌

[1] H. Zhang, Z. Zheng, S. Xu, W. Dai, Q. Ho, X. Liang, Z. Hu, J. Wei, P. Xie, and P. Xing, “Poseidon: An Efficient Communication Architecture for Distributed Deep Learning on GPU Clusters,” in *Proc. 2017 USENIX Annual Technical Conference (NSDI 17),* USA, July 2017.

[2] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini and P. Kalnis, “In-Network Computation is a Dumb Idea Whose Time Has Come,” in *Proc. HotNets-XVI,* USA, November 2017.

[3] A. Sapio, M. Canini, C. Ho, J. Nelson, P. Kalnis, C. Kim, A. Krishnamurthy, M. Moshref, D. Ports, and P. Richtarik, “Scaling Distributed Machine Learning with In-Network Aggregation,” in *Proc. 18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21),* April 2021.

[4] C. Lao, Y. Le, K. Mahajan, Y. Chen, W. Wu, A. Akella, and M. Swift, “ATP: In-network Aggregation for Multi-tenant Learning,” in *Proc. 18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21),* April 2021.