딥러닝 모델의 분산 학습 통신 병목 분석

구인회, 양경식, 유혁
고려대학교 정보대학 컴퓨터학과

noblekoo@korea.ac.kr, ksyang@os.korea.ac.kr, chuckyoo@os.korea.ac.kr

Communication Bottleneck Analysis on Distributed Training of Deep Learning Models

Inhoe Koo, Gyeongsik Yang, Chuck Yoo
Department of Computer Science and Engineering, Korea University

요 약

분산 딥러닝은 딥러닝 학습을 진행함에 있어 여러 서버에서 병렬적으로 딥러닝 학습을 진행하는 방법으로 최근 방대한 입력 데이터와 복잡한 딥러닝 모델을 학습하는 과정에 있어 필수적인 기술이다. 본 논문은 분산 딥러닝 과정의 병목 분석을 위해 가장 널리 사용되는 텐서플로우를 기준으로 파라미터 서버 기반의 분산 딥러닝 학습을 수행할 때, 라이브러리 내부의 콜체인을 상세하게 분석한다. 이를 기반으로, 1 회의 iteration 동안 콜체인의 함수레벨 프로파일링을 수행한다. 실험 결과를 바탕으로, 본 논문은 최종적으로 분산 딥러닝 학습에서 가장 많은 시간을 소요하는 요소는 모델 업데이트를 위한 통신을 수행하고 worker 의 학습을 진행하는 과정임을 도출하며, 이를 기반으로 향후 연구 방향을 고찰한다.

1. 서 론

정교한 딥러닝 학습을 위해 더 많은 입력 데이터와 더 복잡한 딥러닝 모델이 필요해지면서, 기존의 방식대로 한 서버에서 딥러닝 학습을 진행하는 것은 너무 많은 시간이 걸리는 문제가 존재한다. 이에 다수의 서버로 병렬적으로 딥러닝 학습을 진행하는 분산 딥러닝이 주목받고 있다. 분산 딥러닝은 학습결과의 취합 및 파라미터 수정 등의 추가적인 연산이 존재하며 이 과정에서의 지연 또한 발생한다. 기존 연구인 Optimus[1], Gandiva[2], Tiresias[3], TensorExpress[4], PCN[5] 등은 분산 딥러닝 자체의 성능을 개선하기 위한 스케줄링 기법 개선, 병렬 통신 방법 등을 제안하고 있으나, 프레임워크 자체에서 각 구간별로 얼마나 병목이 발생하는 지에 대한 병목의 분석이 정량적으로 수행된 바가 없다.

이에 본 논문은 대중적으로 사용하고 있는 오픈소스 머신러닝 플랫폼인 TensorFlow[6]를 기반으로 1) 학습을 진행할 때의 콜체인을 분석하고, 2) 실제 분산 딥러닝 환경에서 콜체인 내의 각 함수가 얼마만큼의 시간을 소요하는지 측정한다. 이 과정에서 콜체인 함수별 소요시간을 기준으로 성능 개선의 가능성을 탐색하고, 향후 연구의 방향 및 목표를 고찰한다.

2. 배경지식

분산 딥러닝 학습 방식 중 하나인 “파라미터 서버” 방식은, 분산 딥러닝 학습을 진행할 때 GPU 등의 가속기가 탑재된 각 서버가 학습을 수행한 연산 결과(그라디언트)를 수집하고, 모델의 파라미터(parameter)를 독립적인 서버 (파라미터 서버, parameter server)를 통해 갱신하는 방식이다. 특히, 텐서플로우를 비롯한 많은 딥러닝 프레임워크에서 파라미터 서버 기반의 분산 딥러닝을 지원한다. 파라미터 서버 기반의 분산 딥러닝 학습은 1) 전달받은 parameter 로 학습을 수행하고 그 결과인 gradient 를 파라미터 서버에게 넘겨주는 역할을 하는 worker 와 2) worker 가 학습한 gradient 를 취합하고 parameter 를 업데이트 하여 worker 에 제공하는 역할을 하는 파라미터 서버로 구성된다. 텐서플로우에서는 파라미터 서버와 worker 간 통신을 위해 gRPC 를 사용한다.

3. 텐서플로우 분산 딥러닝 그래프 학습 콜체인

분산 딥러닝 환경을 구성하기 위해 유저 코드에서 클러스터와 서버를 생성하고 Run 함수를 호출하면 그림 1 과 같은 콜체인을 따르면서 한 iteration 을 진행한다. RunPartitionsHelper 함수는 worker 에 학습할 parameter 를 전달하기에 앞서 worker 와 파라미터 서버 사이의 partition 의 개수나 timeline, cost 와 같은 정보와 request 와 response 명령을 처리하기 위한 함수를 설정하고 전체 그래프를 정해진 파티션의 개수만큼 나눈 후, RunGraphAsync 함수를 호출해 worker 가 그래프 연산을 진행할 수 있도록 한다.

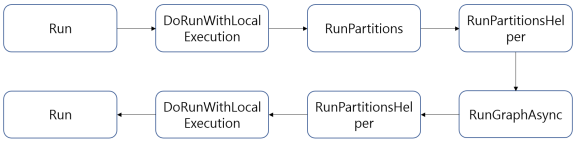


그림 1. 텐서플로우 분산 환경 학습 주요 함수

먼저, Run 함수는 유저 코드에서 파라미터 서버에 호출하는 함수로, 기존에 어떤 iteration 을 실행 중이던 기록이 없으면 DoRunWithLocalExecution 함수를 호출한다. DoRunWithLocalExecution 함수는 그래프 저장을 하고 세션을 생성하고 몇 번째 iteration 인지 기록하고 RunPartitions 함수를 호출한다. RunPartitions 함수는 그래프 학습에서 사용할 tensor 의 이름과 index 를

연결하고 RunPartitionsHelper 함수를 호출한다. 그 후, RunPartitionsHelper 함수에서 파라미터 서버는 worker 가 학습을 다 끝내고 calls.Wait()을 통해 gradient 를 반환하기를 기다리며, 반환을 하면 fetch 와 메타 데이터를 수집해 DoRunWithLocalExecution 함수로 반환한다. 해당 함수는 세션 진행 시간과 같은 세션 통계를 내고 상태를 Run 함수로 반환하면 Run 함수도 받은 상태를 반환하면서 1 회의 iteration 이 끝난다.

4. 텐서플로우 분산 딥러닝 환경 오버헤드 측정

4.1. 측정 방법

실험의 수행을 위해 Intel i7-4770 CPU (3.4GHz) CPU 및 10GB 메모리의 Ubuntu 18.04 머신에서 MNIST 데이터셋을 학습하는 CNN 모델을 생성한다. 각 함수의 콜체인별 오버헤드(시간) 측정을 위해, TensorFlow 1.15.0 버전의 코드에 구간별 프로파일링 코드를 추가 구현한다. 본 실험에서는 한 개의 파라미터 서버와 한 개의 worker 를 같은 서버에서 apache2 서버를 통해 통신하는 방식을 선택한다. 그래프 학습 과정에서의 소요 시간 분석은 10 회의 학습 중 평균값 및 가장 특이값이 적고 대표성을 지니는 학습 회차를 기준으로 한다.

4.2. 그래프 학습 소요 시간 측정 결과

함수별 학습 소요시간. 각 함수가 iteration 마다 소요하는 시간을 그래프로 만들어 나타낸 결과는 그림 2 와 같다.

Run 함수(그림 2a)의 경우 규칙성 없이 대부분 100~600μs 사이의 고른 시간을 소요한다. DoRunWithLocalExecution 함수(그림 2b)의 경우 그래프 연산을 시작할 때와 끝낼 때 약 3 초 가까이를 소요하는 것을 확인할 수 있는데, 이는 그래프 빌드와 세션 통계 관리 때문임을 알 수 있다. RunPartitions 함수(그림 2c)의 경우 10~60μs 또는 90~140μs 사이의 두 그룹의 소요 시간을 가진다.

RunPartitionsHelper 함수 (그림 2d)는 첫 번째와 마지막 iteration 을 제외하면 1) 10000μs 이하 또는 2) 110000~150000μs 사이의 소요시간이 교차하여 나타나며 다른 함수들에 비해 긴 시간을 소모한다. 이 함수 내부에서 가장 많은 시간을 소모하는 부분을 코드 상에서 확인해본 결과, 이는 worker 에서 통신을 기다리는 부분이며, 실험 결과 상 전체 RunPartitionsHelper 실행 시간의 86%를 차지한다. 이 부분은 파라미터 서버가 worker 에 gradient 를 전달하는 통신 시간과 worker 가 학습을 진행하는 시간, 그리고 worker 가 파라미터 서버로 parameter 를 전달하는 시간을 모두 합한 시간으로서, 전체 학습 시간 중 가장 큰 병목에 해당한다.

각 함수별 실행을 기반으로, 1 회의 iteration 단위로 학습 시간을 분석한다. 1 회의 iteration 은 평균 85647μs 가 소모되며, 세부적으로 iteration 에서 각 함수가 차지하는 비중은 Run 함수는 평균적으로 약 0.3% (약 253μs), DoRunWithLocalExecution 함수는 약 13% (약 11200μs) RunPartitions 함수는 약 0.005% (약 40μs), 마지막으로 RunPartitionsHelper 함수는 약 87% (약 74492μs)를 소요한다. 즉, 전체 시간 대비 RunPartitionsHelper 함수가 87%로 가장 많은 시간을 차지하는 것을 확인할 수 있다.

5. 결론 및 향후연구

본 논문은 오픈소스 머신러닝 플랫폼 텐서플로우에서 분산 딥러닝 학습을 진행할 때 성능 병목을 분석한다. 분석을 위해 1) 내부 분산 학습의 콜체인을 상세분석하고, 2) 함수레벨 상세 프로파일링을 수행하며, 3) 실험 결과를 분석하여 RunPartitionsHelper 함수가 대부분의 시간을 소요한다는 것을 확인하였다.

이러한 결과는 텐서플로우의 분산 딥러닝 환경에서 파라미터 서버와 worker 사이의 통신이 병목이 되어, 자원 사용 등 큰 비효율이 발생하고 있을 수도 있음을 시사한다. 이러한 문제점이 존재하는지 확인하고 해결하기 위해, 1) worker 의 학습 시간을 측정해 통신 병목이 존재하는지 확인하고, 2) 다른 학습모델에 대해서 어느 정도의 통신 병목 시간이 나타나는 지를 상세하게 비교 분석하고, 3) 이를 기반으로 네트워크 병목 개선을 기반으로 한 학습 시간 개선 연구를 수행할 예정이다.

ACKNOWLEDGMENT

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No. 2015-0-00280, (SW 스타랩) 성능 및 보안 SLA 보장이 가능한 차세대 클라우드 인프라 SW 개발)과 한국연구재단-차세대 공학연구자 육성사업의 지원을 받아 수행된 연구임(No. NRF-2019H1D8A2105513).

참 고 문 헌

[1] Peng, Yanghua, et al. "Optimus: an efficient dynamic resource scheduler for deep learning clusters." Proceedings of the Thirteenth EuroSys Conference. 2018.

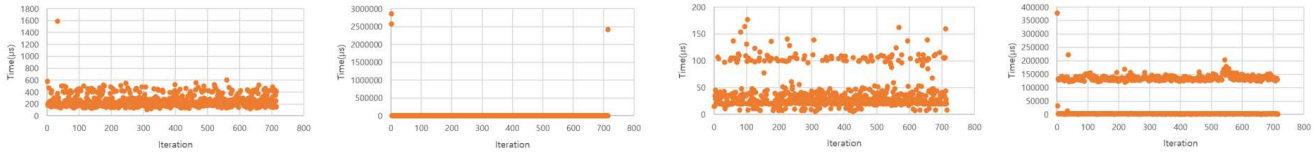
[2] Xiao, Wencong, et al. "Gandiva: Introspective cluster scheduling for deep learning." 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). 2018.

[3] Gu, Juncheng, et al. "Tiresias: A GPU cluster manager for distributed deep learning." 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19). 2019.

[4] Kang, Minkoo, et al. "TensorExpress: In-Network Communication Scheduling for Distributed Deep Learning." 2020 IEEE 13th International Conference on Cloud Computing (CLOUD). IEEE, 2020.

[5] Kang, Minkoo, et al. "Proactive Congestion Avoidance for Distributed Deep Learning." Sensors 21.1 (2021): 174.

[6] Abadi, Martín, et al. "Tensorflow: A system for large-scale machine learning." 12th USENIX symposium on operating systems design and implementation (OSDI 16). 2016.



(a) Run (b) DoRunWithLocalExecution (c) RunPartitions (d) RunPartitionsHelper

그림 2. 그래프 연산 콜체인의 매 iteration 마다의 소요 시간 그래프