지리 분산 학습 환경에서 GPU 성능을 고려한 배치 크기 및 로컬 스텝 수 조정 기법

신정윤, 신창용, 유혁, 양경식
고려대학교 정보대학 컴퓨터학과

sjysjy2002@korea.ac.kr, {cyshin,chuckyoo}@os.korea.ac.kr, g_yang@korea.ac.kr

GPU Performance-Aware Batch Size and Local Step Adjustment for Efficient Geo-Distributed Training

Jungyoon Shin, Changyong Shin, Chuck Yoo, Gyeongsik Yang
Department of Computer Science and Engineering, Korea University

요 약

본 논문은 지리적으로 분산된 환경에서 거대 언어 모델 학습 시 GPU 의 이기종성으로 인해 발생하는 GPU 유휴 문제를 완화하기 위한 GPU 성능 기반의 배치 크기 및 로컬 스텝 수 조정 기법을 제안한다. 지리 분산 학습 환경에서는 GPU 성능의 차이로 인해 동기화 시점에 빠른 노드가 느린 노드를 기다리면서 GPU 유휴가 발생하여 자원 활용률이 저하된다. 이를 해결하기 위해 본 연구에서는 GPU 연산 성능과 메모리 크기를 기준으로 배치 크기 및 로컬 스텝 수를 조정하는 기법을 제시하고, 이를 실험적으로 평가하였다. 실험 결과, 배치 크기 조정은 GPU 유휴 시간을 오히려 증가시켰으나, GPU 성능 기반의 로컬 스텝 수 조정은 GPU 유휴 시간을 약 39% 감소시켜, GPU 자원의 효율적인 활용과 동기화 병목 완화에 효과적임을 확인하였다.

Ⅰ. 서 론

최근 자연어 처리 분야에서는 거대 언어 모델(Large Language Model, LLM)이 다양한 과제를 효과적으로 해결하며 높은 성능을 보이고 있다. 이에 따라 기업뿐 아니라 개인 연구자들도 자신만의 LLM 을 구축하려는 수요가 증가하고 있다. 그러나 수십억 개 이상의 파라미터를 가진 LLM 을 학습하기 위해서는 대규모 GPU 클러스터가 필요하며, 단일 지역 내에서 많은 양의 GPU 자원을 확보하는 것은 어려움이 있다[1].

이러한 한계를 극복하기 위해 지리적으로 분산된 GPU 자원을 통합해 학습을 수행하는 지리적 분산 학습(Geo-Distributed Training, GDT)이 주목받고 있다. GDT 환경에서는 광역 통신망(wide-area network, WAN)의 낮은 대역폭과 높은 지연으로 인해, 각 노드가 학습한 그래디언트(gradient)의 동기화 시 병목이 발생한다. 이를 완화하기 위해 Local SGD[2]나 DiLoCo[3]와 같은 계층적 최적화기법이 제안되었다.

하지만 기존 연구는 GPU 의 이기종성(heterogeneity), 즉 GPU 의 연산 성능 및 메모리 크기 차이를 충분히 고려하지 않았다. 이로 인해 성능이 뛰어난 GPU 를 갖춘 노드는 상대적으로 성능이 낮은 GPU 를 갖춘 노드가 로컬 학습을 완료할 때까지 대기하게 되며, 이는 동기화 시점에서 GPU 자원의 비효율적인 유휴 시간을 초래한다. 이에 본 연구는 GPU 연산 성능 및 메모리 크기에 따라 배치 크기와 로컬 스텝 수를 조정하는 기법을 제안하며, GDT 환경에서 GPU 유휴를 줄이고 LLM 학습 효율을 개선하는 것을 목표로 한다.

Ⅱ. 배경지식 및 관련 연구

지리 분산 학습. 지리적으로 분산된 환경에서의 학습은 모델의 일관성과 안정적인 수렴을 위해 개별 노드가 학습한 그래디언트에 대한 동기화를 필수적으로 요구한다. 이때 동기화 과정에서 WAN 네트워크의 고지연, 저대역폭 특성으로 인해 병목이 발생한다.

DiLoCo. DiLoCo[3]는 지리적으로 분산된 환경에서 거대 언어 모델의 효율적인 학습을 위해 계층적 최적화 기법을 제안한다. 구체적으로, 각 지리 분산 노드는 사전에 정의된 여러 번의 로컬 학습(스텝) 및 최적화를 수행하며, 이때는 노드 간 동기화를 실시하지 않는다. 각 노드가 모두 로컬 스텝 수만큼 학습한 이후, 노드들은 생성한 그래디언트를 동기화 하기 위해 전역 최적화를 수행한다. 즉 이러한 계층적 접근법을 통해 동기화 횟수를 감소시켜 지리 분산 환경에서도 통신 비용을 최소화하면서 모델의 수렴을 보장하고 학습 효율을 극대화할 수 있다.

Ⅲ. 배치 크기 및 로컬 스텝 수 조정 기법

본 연구는 지리 분산 학습에서 각 노드의 GPU 유휴 시간을 개선하기 위해 아래 세가지 기법을 통해 배치 크기 및 로컬 스텝 수를 조정한다.

A. GPU 연산 성능에 비례하여 배치 크기 조정
B. GPU 메모리 크기를 고려하여 배치 크기 조정
C. GPU 연산 성능에 비례하여 로컬 스텝 수 조정

이를 위해 본 논문에서 사용한 GPU 들의 부동소수점 연산 성능과 GPU 메모리 크기를 NVIDIA 에서 제공하는 GPU 성능 지표를 참고하여 정리했다(표 1). 실험을 위해 총 4 대의 이기종 GPU 를 보유한 노드를 사용하였으며, 각 노드는 각각 4 개의 GPU 를 장착하고 있다. 부동소수점 연산 성능의 경우 노드 2 는 BF16 연산 성능을 기준으로 삼았으며, 나머지 노드는 FP16 연산 성능 지표를 사용했다. 또한 각 기법의 성능을 평가하기 위한 기준점(baseline)을 설정하고자 기존 연구를 참고하여 배치 크기는 4, 로컬 스텝 수는 100 으로 설정하였다.

표1. GPU 성능 지표

노드 번호	GPU 정보		부동소수점 연산 성능		GPU 메모리 크기(GB)
	기종	개수	FP16 (TFLOPS)	BF16 (TFLOPS)	
1	2080 Ti	4	**113.8**	-	11
2	2080 Ti	3	**113.8**	-	11
	Titan RTX	1	**130**	-	24
3	A30	4	165	**165**	24
4	RTX 3090	2	**142**	71	24
	V100	2	**125**	-	32

배치 크기 조정 기법. 노드별 GPU 이기종성을 고려하여 배치 크기를 조정하기 위한 기법으로 다음 두 가지 기법 A, B 를 제안한다. 기법 A 는 GPU 부동소수점 연산 성능에 비례하여 배치 크기를 조정하는 기법이다. 표 2 는 각 노드별 평균 부동소수점 연산 성능 및 정규화 성능 지표이다. 노드 1, 노드 3 은 각각 FP16, BF16 자료형의 부동소수점 연산 성능을 사용했다. 노드 2, 노드 4 의 경우 다른 GPU 기종이 장착되어 있기 때문에, 각 GPU 의 부동소수점 연산 성능의 평균을 사용했다. 이후 가장 작은 부동소수점 연산 성능 값으로 정규화 한 뒤, 정규화 성능에 비례하게 배치 크기를 조정하였다.

기법 B 는 각 노드에서 배치 크기를 4 에서 두배씩 키워가며 out-of-memory 오류가 발생하지 않는 최대 배치 크기를 탐색했다. 이를 통해 GPU 메모리 자원을 최대한 활용할 수 있으며, 탐색 및 배치 크기 조정 결과는 표 2 와 같다.

로컬 스텝 수 조정 기법. 로컬 스텝 수 조정 기법(이하 기법 C)은 기법 A 의 방법론을 토대로 정규화 성능에 비례하게 로컬 스텝 수를 조정하였고, 조정 결과는 표 2 와 같다.

표2. 노드별 연산 성능 지표 및 기법별 조정 결과

노드 번호	성능 지표		기법 A	기법 B	기법 C
	평균 연산 성능	정규화 성능	**배치 크기 조정**	**배치 크기 조정**	**로컬 스텝 수 조정**
1	113.8	1.000	4 → **4**	4 → **8**	100 → **100**
2	117.9	1.036	4 → **4**	4 → **8**	100 → **104**
3	165	1.450	4 → **6**	4 → **32**	100 → **145**
4	133.5	1.173	4 → **5**	4 → **16**	100 → **117**

IV. 실험 환경 및 실험 결과

실험 환경. 본 연구에서는 표 1 에 명시한 4 개의 이기종 GPU 노드로 지리 분산 학습 환경을 구성하여 실험을 진행했다. 각 노드는 1G 이더넷을 통해 연결되어 있으며, 학습에는 LLaMA 기반의 150M 개의 학습 파라미터 갖는 언어 모델을 사용했다. 전체 학습 스텝 수는 총 10 회의 동기화 및 전역 최적화를 실시하도록 설정했다. 지리 분산 학습 실험에는 DiLoCo 연구를 오픈소스로 구현한 OpenDiLoCo[4]를 각 노드가 서로 다른 배치 크기 및 로컬 스텝 수를 사용하여 학습할 수 있도록 수정하여 사용했다. CUDA 는 12.4, GPU 드라이버는 550.67, Python 은 3.11, Torch 는 2.3.1 버전을 사용했다.

실험 결과. 표 3 은 baseline 과 각 제안 기법의 GPU 유휴 시간을 측정하고 비교한 결과를 나타낸다. Baseline 의 경우, 가장 높은 부동소수점 연산 성능을 갖춘 노드 3(표 1 참조)의 GPU 유휴 시간이 가장 긴 반면, 상대적으로 성능이 낮은 노드 1 과 2 는 유휴 시간이 거의 0 에 가깝다는 것을 확인할 수 있다. 배치 크기를 조정한 기법 A 와 B 의 총 GPU 유휴 시간은

표3. 기법 별 GPU 유휴 시간 측정 결과

기법	GPU 유휴 시간 (초)				총 GPU 유휴 시간 (초)
	노드 1	노드 2	노드 3	노드 4	
Baseline	1.1	0.0	14.6	4.9	20.6
A	11.9	0.0	14.7	13.8	40.4
B	6.8	7.2	0.0	14.4	28.4
C	2.1	0.0	8.6	1.9	**12.6**

각각 40.4 초와 28.4 초로, baseline 대비 각각 약 96%, 38% 증가하였다. 이는 배치 크기만을 조정하는 방식이 이기종 GPU 의 연산 성능 차이를 충분히 반영하지 못함을 시사한다.

반면 로컬 스텝 수를 조정한 기법 C 의 경우 총 GPU 유휴 시간이 12.6 초로, baseline 대비 약 39% 감소하여 GPU 자원의 효율성을 향상시켰다. 이는 성능이 뛰어난 GPU 가 더 많은 로컬 스텝을 수행하도록 하여, 동기화 지점까지 각 노드에서 소요되는 로컬 학습 시간을 효과적으로 균형(balance) 있게 조정한 결과라고 해석할 수 있다.

V. 결론 및 향후 연구

본 연구는 지리적으로 분산된 환경에서 거대 언어 모델의 학습 시 GPU 의 이기종성으로 인해 발생하는 GPU 유휴 문제를 해결하고자 GPU 연산 성능 및 메모리 크기를 고려하여 배치 크기와 로컬 스텝 수를 적응적으로 조정하는 기법을 제안하였다. 실험 결과, GPU 성능에 비례하여 로컬 스텝 수를 조정한 기법 C 의 경우, baseline 대비 GPU 유휴 시간이 약 39% 감소하여 가장 뛰어난 효율성을 보였다. 반면, 배치 크기만을 조정한 기법 A 와 B 는 오히려 GPU 유휴 시간이 증가하여, GPU 이기종성을 해소하는 데 한계를 보였다.

향후 연구로는 본 논문에서 수행한 로컬 스텝 수 조정과 배치 크기 조정 기법을 결합한 하이브리드 접근법을 탐색하고자 한다. 또한 노드 간 네트워크 특성을 반영한 보다 정교한 최적화 알고리즘을 설계하여 통신 병목까지 종합적으로 완화할 수 있는 연구를 진행할 계획이다.

ACKNOWLEDGMENT

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(RS-2023-NR077249, RS-2024-00336564), 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(RS-2021-NR060143), 또한 본 연구는 Google Cloud Research Credits program 의 지원을 받아 수행한 연구임.

참 고 문 헌

[1] Gandhi, Rohan, et al. "Improving training time and GPU utilization in geo-distributed language model training." arXiv preprint arXiv:2411.14458 (2024).

[2] McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." Artificial intelligence and statistics. PMLR, 2017.

[3] Douillard, Arthur, et al. "Diloco: Distributed low-communication training of language models." arXiv preprint arXiv:2311.08105 (2023).

[4] Jaghouar, Sami, Jack Min Ong, and Johannes Hagemann. "Opendiloco: An open-source framework for globally distributed low-communication training." arXiv preprint arXiv:2407.07852 (2024).