SPFHP 알고리즘을 기반으로 한 GPU 자원 활용 최적화 방안 연구

정승범, 윤수연*
국민대학교, *국민대학교

jp1925@kookmin.ac.kr, *1104py@kookmin.ac.kr

Research on how to optimize GPU resource utilization based on SPFHP algorithm

Seung-Bum Jeong, Soo-Yeon Yoon*
Kookmin Univ., *Kookmin Univ.

요 약

본 연구는 GPU 자원의 활용도를 극대화하고 워크로드 간의 공정한 자원 분배를 실현하기 위해, SPFHP(Shortest Pack First with High Priority) 알고리즘 기반의 동적 배치 기법을 제안한다. 기존의 배치 기법들은 데이터 단순 할당에 초점을 맞췄기에, 특정 GPU 에 편향된 자원을 분배하는 문제를 야기해왔다. SPFHP 는 길이 분포에 기반한 히스토그램 정렬과 우선순위 기반 조합 전략을 통해, 모델에 입력되는 데이터를 길이를 균일하게 만들었다. 그리고 패드에 의한 자원 낭비를 최소화하기 위해 행 방향 배치를 활용해 패드 없는 연산을 가능하게 만들었다. 실험 결과, SPFHP 는 단순 행 방향 배치 방식에 비해 약 1.6 배의 학습 속도 향상을 보였으며, GPU 메모리 사용 효율 또한 유의미하게 개선함을 확인하였다.

1. 서 론

최근 초거대 언어 모델의 성능 향상에 따른 파라미터 수 증가로, 학습자원 최적화의 필요성이 더욱 강조되고 있다. 그러나 성능 향상을 위해 배치를 높이는 과정에서, 길이를 맞추기 위한 패드가 과도하게 삽입되어 전체 연산량 증가를 초래하게 되었다.[1] 또한 여러 대의 GPU 로 분산학습을 수행하게 되면서 불균등한 자원 분배로 인해 일부 장비에서 유휴자원이 발생해 자원을 낭비하는 문제가 발생했다.

이에 본 연구는 SPFHP 알고리즘과 행 방향 배치를 활용해, 적용하기 쉬우면서도 초거대 언어 모델의 분산 학습 환경에서 발생하는 자원낭비 문제를 최소화하는 방식을 제안하고자 한다. 이를 RTX 3090 24GB GPU 4 장과 DeepSpeed ZeRO3 를 활용한 분산 학습 환경에서 Gemma2-2B 모델로 유효성을 검증하고자 한다. 결과 학습속도가 1.6 배 정도 속도가 개선되었다.

2. 관련 연구

2.1 SPFHP 알고리즘 개요

시퀀스 패킹은 유사한 길이의 데이터로 배치를 구성해 삽입되는 패드의 양을 줄이는 최적화 기법이다. 패킹의 성능은 압축률로 측정되며, 이는 패킹 적용 전후의 시퀀스 수 비율로 정의된다. 특히 SPFHP 알고리즘은 전통적인 LPT(Longest Processing Time)[4] 기반 알고리즘과 비교해 경쟁력 있는 성능을 보이며, SPFHP 는 99.323%의 압축률을 기록한 반면 LPT 는 99.321%를 달성해 미세하지만 우수한 결과를 보였다. 이러한 높은 압축률과 더불어 빠른 처리 속도를 갖춘 SPFHP 알고리즘은 본 연구의 시퀀스 패킹 기법으로 채택했다.[2]

2.2 동적 배치를 활용한 패드 최소화

동적 배치는 데이터의 특성과 자원 상황에 따라 배치 크기를 유연하게 조절하는 기법이다. 기존에는 모델의 성능에 맞춰 배치 크기를 능동적으로 조정했으나, 모델 파라미터의 급격한 증가로 인해 이러한 접근은 점차 어려워지고 있다. 대안으로 유사한 길이의 데이터를 묶어 여러 번의 학습을 수행하는 커리큘럼 방법이 제안되었으나, 번거로운 과정에 구현이 복잡하다는 한계를 지닌다.[3] 따라서 간편하게 적용 가능하면서도 자원 낭비를 최소화할 수 있는 방식이 요구된다.

최근에는 인과적 언어 모델을 대상으로, 배치 내 데이터를 행 방향으로 연결하여 패드 없이 학습을 수행하는 기법이 제안되었다[1]. 기존의 동적 배치 방식은 패드를 줄이는 데 그쳤던 반면, 이 방식은 패드를 제거해 연산 낭비를 해소할 수 있게 되었다.

3. SPFHP 기반 동적 배치 설계

본 연구에서는 GPU 자원 활용 효율을 극대화하기 위해 Pytorch 의 DataLoader 내의 Sampler 와 Collator 를 수정하는 방식을 채택한다. Sampler 는 학습에 사용할 데이터를 선택하는 역할을 수행하며, Collator 는 선택된 데이터를 모델 입력 형식에 맞게 정렬하고 병합하는 기능을 담당한다. 본 설계에서는 Sampler 에 SPFHP 알고리즘을 적용하여 시퀀스 길이에 따른 불균형한 자원 분배를 최소화하고, Collator 에는 flash-attention 에 적합한 행 방향 배치 방식을 도입해 패드 없이 학습이 가능하도록 했다.

3.1 동적 배치 구조

본 기법은 시퀀스의 길이 분포를 히스토그램 형태로 분석하고, 이를 최적화 문제로 환원하여 고르게 패킹하는 방식으로 메모리 효율성과 학습 처리량을 동시에 향

상시킨다. 우선, 각 시퀀스 길이 l에 대한 빈도수 H_i를 집계하여 시퀀스 길이 분포 히스토그램 H를 구성하고, 이를 역순으로 정렬한 H_{rev}를 통해 긴 시퀀스를 우선 고려하는 방식으로 패킹 전략을 수립한다. 이후 다음의 조건을 만족하는 집합 P를 반복적으로 탐색한다.

$$P = \{(p_1, p_2, \dots, p_k) \mid \sum_{j=1}^{k} p_j \le L_{max} \wedge k \le K\}$$

여기서 L_{max} 는 하나의 배치에 입력 가능한 최대 시퀀스 길이이며, K는 하나의 팩에 포함될 수 있는 최대 샘플 수를 의미한다. SPFHP 는 가능한 모든 조합 중에서 시퀀스 길이 합이 L_{max}에 근접하면서도 총합이 가장 짧은 조합을 우선적으로 선택하여 패킹 효율을 극대화한다. 이와 같은 방식은 시퀀스 간 길이 차이로 인해 발생하는 패딩 낭비와 연산 자원 불균형 문제를 효과적으로 감소시킨다. 패킹이 완료된 시퀀스들은 행 방향으로 연결되어 하나의 긴 시퀀스처럼 입력되며, 이는 패딩 없이 연속적인 학습이 가능하도록 만든다.

그러나 이러한 연속 패킹 시 직접적인 self-attention 연산을 적용할 경우, 각 시퀀스 간 정보가 혼합되는 교차 오염문제가 발생할 수 있다. 이를 방지하기 위해 본 논문에서는 그림 1 과 같이 각 시퀀스별로 독립적인 attention mask 와 position ids 를 구성한다. attention mask 는 연산 범위를 해당 시퀀스 내부로만 제한하고, position ids 는 각 시퀀스의 상대적 위치 정보를 독립적으로 유지하여, 모델이 이들을 하나의 긴 시퀀스가 아닌 개별 시퀀스로 인식하도록 유도한다.

이러한 처리 과정을 통해 SPFHP 는 시퀀스 패킹의 효율성과 학습 정확도를 동시에 만족시키는 실용적인 기법으로 기능하며, 특히 자연어 처리와 같이 시퀀스 기반의 입력 데이터가 주를 이루는 작업에서 탁월한 성능을 보인다.

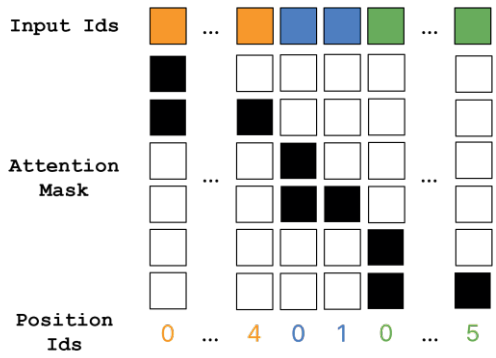

그림 1. 교차 오염 방지를 위한 attention mask 와 position ids

4. 실험 결과 및 성능 평가

본 연구의 실험은 RTX 3090 24GB GPU 4 장과 DeepSpeed ZeRO3 를 활용한 분산 학습 환경에서 진행되었으며, 실험 대상 모델은 Gemma2-2B 이다. 학습 데이터는 OpenOrca 데이터셋에서 총 4,177,316 건을 사용하였으며, 전체 데이터를 1 epoch 동안 학습시키는 데 소요되는 시간을 기준으로 알고리즘의 효율성을 비교하였다.

배치 크기는 OOM 이 발생하지 않는 최대값으로 설정해 GPU 메모리 자원을 완전히 활용하는 상황을 구성했으며, 모든 실험은 bf16 환경에서 flash-attention 을 적용해 진행했다. 비교는 두 가지 조건에 대해 이루어졌다. 첫번째는 단순히 행 방향 배치만 적용한 경우, 두번째는 행 방향 배치에 더해 SPFHP 알고리즘을 sampler 에 적용한 경우다. SPFHP 알고리즘에서의 최대 시퀀스 길이 L_{max} 는 2024, 하나의 패킹에 포함될 수 있는 최대 시퀀스 수 K는 20 으로 설정했다.

표 1. 알고리즘 적용 조건에 따른 학습 시간 비교

조건	학습 시간	배치 크기
행 방향 배치만 적용	170:22:06	6
SPFHP + 행 방향 배치 적용	**106:13:39**	2

5. 결론 및 시사점

실험 결과, 행 방향 배치만 적용한 경우 전체 학습에는 약 170 시간 22 분이 소요되었으며, local batch size 는 6 으로 설정되었다. 반면, SPFHP 알고리즘을 함께 적용한 경우에는 학습 시간이 106 시간 13 분으로 단축되었고, 이때 local batch size 는 2 로 설정되었다. 두 실험 조건 간 학습 시간 비교 결과, SPFHP 알고리즘 적용 시 약 1.6 배의 학습 속도 향상을 확인했다.

이번 연구를 통해 분산학습 환경에서 발생하는 자원 낭비 문제를 효과적으로 해소하는 방안을 제시했다. 특히 시퀀스 길이를 히스토그램 최적화 문제로 환원해 불균형 문제를 최소화하는 SPFHP 알고리즘과 행 방향 배치 방식을 결합해, 기존 방식대비 처리 속도를 1.6 배 개선할 수 있었다. 이러한 성능 향상을 간단한 알고리즘 변경만으로 효율을 크게 향상시킬 수 있음을 의미하며, 복잡한 코딩 없이도 실무에 쉽게 적용할 수 있음을 나타낸다.

참 고 문 헌

[1] Kundu, A., Lee, R. D., Wynter, L., Ganti, R. K., & Mishra, M. (2024). Enhancing training efficiency using packing with flash attention. arXiv preprint arXiv:2407.09105.

[2] Krell, M. M., Kosec, M., Perez, S. P., & Fitzgibbon, A. (2021). Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. arXiv preprint arXiv:2107.02027.

[3] Pouransari, H., Li, C. L., Chang, J. H., Anasosalu Vasu, P. K., Koc, C., Shankar, V., & Tuzel, O. (2024). Dataset decomposition: Faster llm training with variable sequence length curriculum. Advances in Neural Information Processing Systems, 37, 36121-36147.

[4] Imoneoi. (2023). multipack_sampler [Computer software]. GitHub. https://github.com/imoneoi/multipack_sampler