PUNICA: 멀티 테넌트 LORA 서비스

레쿤 첸 [*1] 지하오 예 [*1] 용지 우 [2] 단양 주오 [2] 루이스 세제 [1] 아르빈드 크리슈나무르티 [1]

초록

낮은 순위 적응(LoRA)은 사전 학습된 모델을 특정 도메인에 적용하는 데 중요하고 널리 사용되는 방법이 되었습니다. 유니티는 공유 GPU 클러스터에서 여러 LoRA 모델을 서비스하는 시스템인 Punica를 소개합니다. 퓨니카에는 새로운 CUDA 커널 설계가 적용되어 다양한 LoRA 모델에 대한 GPU 작업을 일괄 처리할 수 있습니다. 이를 통해 여러 개의 서로 다른 LoRA 모델을 서비스할 때 GPU는 사전 학습된 기본 모델의 사본을 하나만 보유할 수 있으므로 메모리와 연산 측면에서 GPU 효율성이 크게 향상됩니다. 유니티의 스케줄러는 멀티테넌트 LoRA 서비스 워크로드를 공유 GPU 클러스터에 통합합니다. 퓨니카의 평가에 따르면, 고정된 크기의 GPU 클러스터를 통해 여러 LoRA 모델을 서비스할 때 최신 LLM 서비스 시스템에 비해 12배 높은 처리량을 달성하는 동시에 토큰당 지연 시간은 2ms밖에 추가되지 않습니다. Punica는 https://github.com/punica-ai/punica 에서 오픈 소스로 제공됩니다.

1 소개

최소한의 학습 데이터로 사전 학습된 대규모 언어 모델(LLM)을 도메인별 작업에 특화하기 위해 저순위 적응(LoRA)(Hu et al., 2022)이 점점 더 인기를 얻고 있습니다. LoRA는 사전 학습된 모델의 가중치를 유지하고 학습 가능한 순위 분해 행렬을 Transformer 아키텍처의 각 계층에 도입하여 학습 가능한 매개변수의 수를 크게 줄이고 테넌트가 저렴한 비용으로 다양한 LoRA 모델을 학습할 수 있도록 합니다. LoRA는 널리 사용되는 많은 미세 조정 프레임워크에 통합되었습니다(Mangrulkar 외., 2022). 따라서 ML 제공업체는 테넌트의 필요에 따라 다수의 전문화된 LoRA 모델을 동시에 제공해야 합니다.

단순히 처음부터 독립적으로 학습된 것처럼 LoRA 모델을 서비스하는 것은 GPU 리소스를 낭비합니다. 각 LoRA 모델을 서비스하는 데 *k개의* GPU가 필요하다고 가정할 때, 서로 다른 *n개의* LoRA 모델을 서비스하려면 겉보기에는 *k개의* GPU가 필요할 것입니다. 이러한 단순한 접근 방식은 동일한 사전 학습된 모델에서 비롯된 LoRA 모델 간의 잠재적인 가중치 상관관계를 간과합니다.

여러 개의 서로 다른 LoRA 모델을 지원하는 효율적인 시스템은 세 가지 설계 지침을 따라야 한다고 생각합니다. (G1) GPU는 비싸고 희소한 자원이므로 멀티테넌트 LoRA 서비스 워크로드를 소수의 GPU로 통합하여 전체 GPU 활용도를 높여야 합니다. (G2) 선행 연구에서도 이미 확인된 바와 같이(Yu et al., 2022), 일괄 처리는 성능과 GPU 활용도를 개선하기 위해 ML 워크로드를 통합하는 가장 효과적인 접근 방식 중 하나이기는 하지만 가장 효과적인 접근 방식은 아닙니다. 그러나 일괄 처리는

*Equal 기여 [1]대학 의 워싱턴 [2]듀크 대학교. 대응 담 당 자 : Lequn Chen <lqchen@cs.washington.edu>.

퀘스트는 정확히 동일한 모델에 대한 것입니다. 따라서 다양한 LoRA 모델에 대해 일괄 처리를 활성화해야 합니다. (G3) 디코드 단계는 모델 서비스 비용의 주요 요인입니다. 따라서 디코드 단계의 성능에만 집중하면 됩니다. 모델 제공의 다른 측면은 덜 중요하며, LoRA 모델 가중치의 온디맨드 로딩과 같은 간단한 기술을 적용할 수 있습니다.

이 세 가지 가이드라인을 기반으로 유니티는 공유 GPU 클러스터에서 LoRA 모델을 위한 멀티테넌트 서비스 프레임워크인 Punica를 설계하고 구현했습니다. 새로운 CUDA 커널인 **세그먼트 개더 매트릭스-벡터 곱셈** (SGMV)을 설계한 것이 한 가지 주요 특징입니다. SGMV를 사용하면 여러 개의 서로 다른 LoRA 모델을 동시에 실행하기 위해 GPU 연산을 일괄 처리할 수 있습니다. SGMV를 사용하면 GPU는 사전 훈련된 모델의 사본을 메모리에 하나만 저장하면 되므로 메모리와 연산 측면에서 GPU 효율성이 크게 향상됩니다. 유니티는 이 새로운 CUDA 커널과 일련의 최첨단 시스템 최적화 기술을 결합했습니다.

SGMV를 사용하면 서로 다른 LoRA 모델의 요청을 일괄 처리할 수 있으며, *놀랍게도* 동일한 LoRA 모델을 일괄 처리할 때와 다른 LoRA 모델을 일괄 처리할 때 성능 차이가 미미하게 관찰되었습니다. 동시에 LoRA 모델의 온디맨드 로딩은 지연 시간이 밀리초 수준에 불과합니다. 따라서 퓨니카는 GPU에서 이미 실행 중인 LoRA 모델에 제약을 받지 않고 사용자 요청을 소수의 GPU로 통합할 수 있는 유연성을 제공합니다.

따라서 Punica는 다음 두 가지 방법으로 멀티테넌트 워크로드를 스케줄링합니다. 새로운 요청의 경우, Punica는 요청을 소수의 활성 GPU로 라우팅하여 해당 GPU가 최대 용량에 도달할 수 있도록 합니다. 기존 GPU가 완전히 활용되는 경우에만 Punica는 추가 GPU 리소스를 할당합니다. 기존 요청의 경우, Punica는 다음을 위해 주기적으로 요청을 마이그레이션합니다.

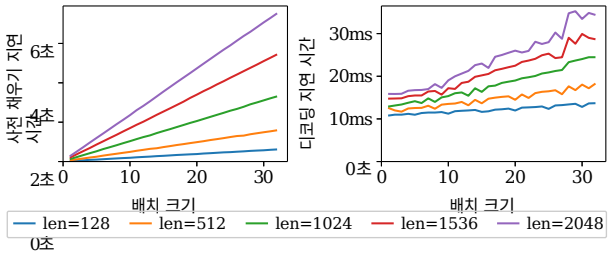

그림 1. 프리필 단계 및 디코드 단계에서의 배치 효과

통합. 이를 통해 Punica에 할당된 GPU 리소스를 확보할 수 있습니다.

저희는 NVIDIA A100 GPU 클러스터에서 Llama2 7B, 13B, 70B 모델(투브론 외, 2023)을 적용한 LoRA 모델을 평가합니다. 동일한 양의 GPU 리소스가 주어졌을 때, Punica는 토큰당 지연 시간이 2밀리초밖에 추가되지 않으면서도 최첨단 LLM 서비스 시스템에 비해 12배 높은 처리량을 달성합니다.

이 백서는 다음과 같은 기여를 합니다:

- 여러 개의 서로 다른 LoRA 모델에 대한 리퀘스트를 일괄 처리할 수 있는 기회를 확인했습니다.
- 유니티는 여러 LoRA 모델을 동시에 실행할 수 있는 효율적인 CUDA 커널을 설계하고 구현합니다.
- 멀티테넌트 LoRA 워크로드를 통합하기 위한 새로운 스케줄링 메커니즘을 개발합니다.

2 배경

먼저 트랜스포머 모델의 텍스트 생성 프로세스를 소개합니다. 그런 다음 트랜스포머 모델의 로우랭크 적응(LoRA)에 대해 설명합니다.

2.1 트랜스포머 및 텍스트 생성

트랜스포머 기반 LLM은 일련의 토큰으로 작동합니다. 토

(*h*, *h*). 디코딩 단계의 경우, *s가* 과거 시퀀스 길이를 나타낸다고 가정할 때, 자기 주의 계층의 계산은 (1, *d*) (*d*, *s* + 1) (*s* + 1, *d*)이고 MLP 계산은 (1, *h*) (*h*, *h*) *입니다*. 디코드 단계는 입력이 단일 벡터이기 때문에 GPU 사용률이 낮습니다.

그림 1은 다양한 배치 크기에 대한 프리필 단계와 디코드 단계의 지연 시간을 보여줍니다. GPU의 계산 능력( )은 프리필 단계에서 완전히 활용됩니다.

큰은 영어 단어의 약 ¾ 크기입니다. LLM의 작동은 두 단계로 구성됩니다. *미리 채우기* 단계에서는 사용자 프롬프트를 수락하고 후속 토큰과 키-값 캐시(KvCache)를 생성합니다. *디코딩* 단계에서는 토큰과 KvCache를 수락한 다음 토큰을 하나 더 생성하고 KvCache에 열을 추가합니다. 디코딩 단계는 반복적인 프로세스입니다. 생성된 토큰은 다음 단계의 입력이 됩니다. 이 프로세스는 시퀀스 종료 토큰이 생성되면 종료됩니다.

트랜스포머 블록에는 자기 주의 레이어와 다층 퍼셉트론(MLP)이 포함되어 있습니다. 프롬프트의 길이가 s이고 주의 헤드 차원이 *d라고* 가정하면, 프리필 단계에서 자기 주의 레이어의 계산은 (*s*, *d*) ×(*d*, *s*) ×(*s*, *d*)이고 MLP 계산은 (*s*, *h*) ×입니다.

사전 채우기 지연 시간은 배치 크기에 비례합니다. 그러나 디코드 단계에서는 그렇지 않습니다. 배치 크기를 1에서 32로 늘리면 디코딩 단계 지연 시간은 짧은 시퀀스의 경우 11ms에서 13ms로, 긴 시퀀스의 경우 17ms에서 34ms로 증가합니다. 즉, 일괄 처리를 통해 디코딩 단계에서 GPU 활용도를 크게 향상시킬 수 있습니다. Orca(Yu et al., 2022)는 이 기회를 활용하여 효율적인 LLM 서빙 시스템을 구축했습니다. 이러한 유형의 배칭은 디코드 단계에서 주로 긴 출력 길이 응답에 대한 서빙 지연 시간을 결정하기 때문에 특히 중요합니다.

2.2 로우랭크 적응(LoRA)

미세 조정을 통해 사전 학습된 모델을 새로운 도메인이나 새로운 작업에 맞게 조정하거나 새로운 학습 데이터로 개선할 수 있습니다. 하지만 LLM은 규모가 크기 때문에 모든 모델 파라미터를 미세 조정하는 데 리소스를 많이 사용합니다.

저순위 적응(LoRA)(Hu et al., 2022)은 미세 조정 중에 학습해야 하는 파라미터의 수를 크게 줄여줍니다. 주요 관찰 사항은 사전 학습된 모델과 미세 조정 후 모델 간의 가중치 차이가 낮은 순위를 갖는다는 것입니다. 따라서 이 가중치 차이는 작고 조밀한 두 행렬의 곱으로 나타낼 수 있습니다. LoRA 미세 조정은 작고 조밀한 신경망을 훈련하는 것과 유사합니다. 공식적으로 사전 훈련된 모델의 가중치를 $W \in \mathbb{R}^{h_1 \times h_2}$로 가정해 보겠습니다. LoRA 미세 조정은 두 개의 행렬 $A \in \mathbb{R}^{h_1 \times r}$과 $B \in \mathbb{R}^{r \times h_2}$를 훈련합니다. 여기서 r은 LoRA 랭크입니다. $W + AB$는 미세 조정된 모델의 새로운 가중치입니다. LoRA 순위는 일반적으로 원래 차원보다 훨씬 작습니다(예: 4096이 아닌 16). LoRA는 빠른 미세 조정 외에도 스토리지 및 메모리 오버헤드가 매우 낮습니다. 미세 조정된 각 모델은 모델 무게의 0.1%~1%만 추가됩니다. LoRA는 일반적으로 주의 메커니즘의 쿼리-키-값-출력 프로젝션과 MLP를 포함하여 트랜스포머 계층의 모든 고밀도 프로젝션에 적용됩니다(Dettmers et al., 2023). 자기 주의 연산 자체에는 가중치가 포함되지 않습니다.

공유 GPU 클러스터에서 멀티테넌트 LoRA 모델을 효율적으로 서비스하는 방법은 무엇인가요? LoRA는 LLM을 미세 조정할 수 있는 효율적인 알고리즘을 제공합니다. 이제 문제는 이러한 LoRA 모델을 어떻게 효율적으로 제공할 수 있을까요? 한 가지 접근 방식은 각 LoRA 모델을 독립적인 모델로 간주하고 전통적인 LLM 서비스 시스템(예: vLLM)을 사용하는 것입니다. 그러나 이는 서로 다른 LoRA 모델 간의 가중치 공유를 무시합니다.

∈ ∈ ∈

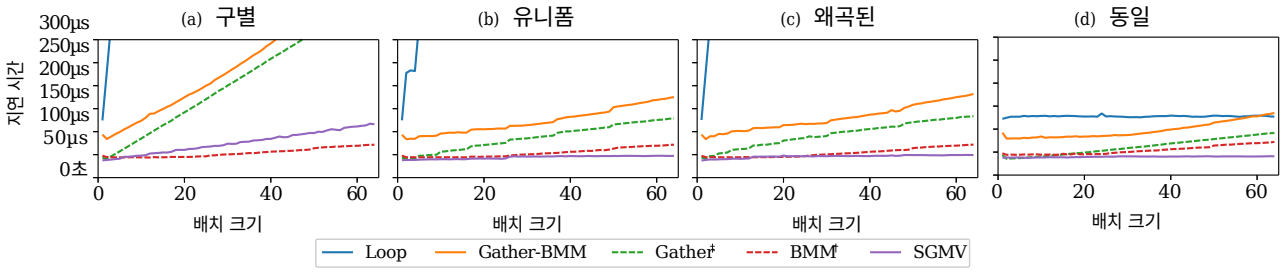

그림 8. LoRA 사업자 구현을 위한 마이크로벤치마크. †참조를 위해 Gather와 BMM을 별도로 측정했습니다.

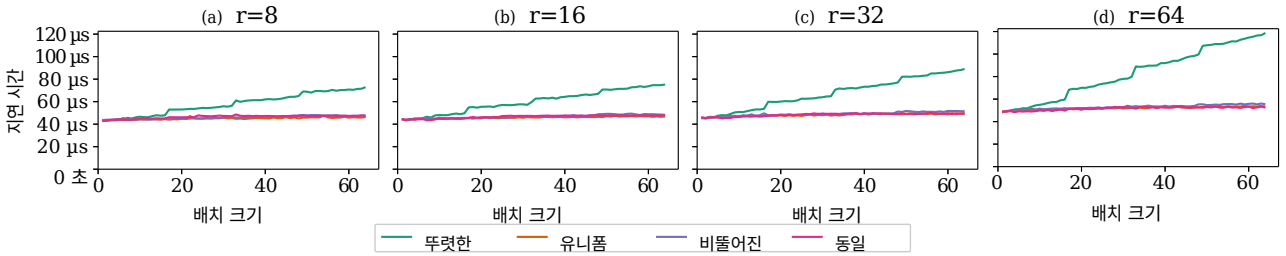

그림 9. 다양한 LoRA 등급에 따른 LoRA 운영자의 마이크로 벤치마크.

전반적으로 SGMV는 워크로드에 관계없이 기준 구현을 훨씬 능가하는 성능을 발휘합니다.

또한 테스트베드 #1에서 다양한 LoRA 랭크의 마이크로벤치마크를 실행했습니다. 그림 9는 LoRA 랭크 8, 16, 32, 64의 레이턴시를 보여줍니다. **Distinct의** 경우 지연 시간이 점차 증가합니다. 단일 요청 배치의 지연 시간은 네 가지 등급 모두에서 약 42µs인 반면, 배치 크기 64는 각각 72µs, 75µs, 89µs, 118µs까지 올라갑니다. 워크로드에 가중치 공유(**균등**, **왜곡**, **동일**)**가** 있는 경우 지연 시간은 배치 크기 1에서 64까지 약 42µs에서 45µs로 거의 동일하게 유지됩니다.

트랜스포머 레이어 벤치마크 다음으로 LoRA 연산자를 통합한 후 트랜스포머 레이어 성능을 평가합니다. LLM은 대략적으로 트랜스포머 레이어의 스택이므로 레이어 성능에 따라 전체 모델 성능이 결정됩니다. 테스트베드 #1에서 7B 및 13B 모델 구성과 512 및 2048의 시퀀스 길이를 기준으로 레이어 벤치마크를 실행합니다. 그림 10은 레이어 레이턴시를 보여줍니다. 시퀀스 길이가 짧을수록 배치 효과가 더 강해집니다. 시퀀스 길이가 512일 때 배치 크기가 1에서 32로 증가할 때 레이턴시는 72%만 증가합니다. 시퀀스 길이가 길면 셀프 어텐션에 더 오랜 시간이 걸리므로 레이어별 일괄 처리 효과가 감소합니다.

커널 마이크로벤치마크와 달리 레이어 레이턴시는 여러 워크로드에서 거의 동일하다는 것을 알 수 있습니다. 이는 백본 밀도 투영과 셀프 어텐션에 비해 LoRA 애드온의 계산 시간이 짧기 때문입니다. *이 LoRA 모델에 구애받지 않는 성능 속성을 통해 다양한 LoRA 모델을 다음과 같이 예약할 수 있습니다.*

로 설정할 수 있습니다. 그러면 스케줄링 알고리즘이 개별 LoRA 모델 배치 대신 전체 처리량에 집중할 수 있으며, 이것이 바로 우리가 Punica를 설계하는 방식입니다.

7.2 텍스트 생성

다음으로 Punica와 기준 시스템의 텍스트 생성 성능을 연구합니다.

단일 GPU에서 7B 및 13B 모델 서비스 제공 테스트베드 #1에서 단일 GPU의 Punica 및 기준 시스템을 사용하여 텍스트 생성을 평가했습니다. 단일 GPU 성능은 클러스터 전체 배포를 위한 기본 사례로 사용됩니다. 1,000개의 요청(약 101,000개의 토큰 생성)을 생성하고 각 시스템이 선착순으로 배치하도록 제한했습니다. 최대 배치 크기는 모든 시스템에 대해 32로 설정되어 있습니다. Punica는 서로 다른 LoRA 모델에 걸쳐 일괄 처리할 수 있으며, 기간계 시스템은 동일한 LoRA 모델에 대한 요청만 일괄 처리할 수 있습니다.

그림 11 (a)와 (b)는 각각 7B 모델과 13B 모델에서의 결과를 보여줍니다. Punica는 워크로드에 관계없이 일관되게 높은 처리량을 제공합니다. Punica는 7B 모델과 13B 모델에서 각각 1044톡/s와 693톡/s를 달성했습니다. 대부분의 기준선은 **동일한** 경우 상대적으로 높은 처리량을 달성할 수 있지만, LoRA 모델이 여러 개일 경우 성능이 저하됩니다.

Distinct 사례에서는 모든 기준 시스템이 배치 크기 1로 실행되므로 처리량이 낮습니다. **균일** 및 **왜곡된** 경우, 기준 시스템의 대부분의 배치 크기는 매우 작습니다(1-3).