푸니카(PUNICA): 다중 테넌트 LORA 서비스

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

추상적인

LoRA(Low-Rank Adaptation)는 사전 훈련된 모델을 특정 모델에 적용하는 중요하고 널리 사용되는 방법이 되었습니다.
도메인. 공유 GPU 클러스터에서 여러 LoRA 모델을 제공하는 시스템인 Punica를 소개합니다. 푸니카에는 다음이 포함되어 있습니다.
다양한 LoRA 모델에 대한 GPU 작업 일괄 처리를 허용하는 새로운 CUDA 커널 디자인. 이를 통해
여러 개의 서로 다른 LoRA를 제공할 때 기본 사전 훈련된 모델의 단일 복사본만 보유하는 GPU
모델을 사용하여 메모리와 계산 측면에서 GPU 효율성을 크게 향상시킵니다. 우리의 스케줄러
공유 GPU 클러스터에 멀티 테넌트 LoRA 제공 워크로드를 통합합니다. 고정된 크기의 GPU 클러스터를 사용하면
우리의 평가에 따르면 Punica는 여러 LoRA 모델을 제공하는 데 있어 기존에 비해 12배 더 높은 처리량을 달성한 것으로 나타났습니다.
토큰당 2ms의 대기 시간만 추가하면서 최첨단 LLM 서비스 시스템에 적용할 수 있습니다. Punica는 오픈소스입니다.
https://github.com/punica-ai/punica.

1. 소개

LoRA(Low-Rank Adaptation) (Hu et al., 2022) 는 사전 훈련된 대
규모 전문화 분야에서 점점 인기를 얻고 있습니다.
최소한의 훈련 데이터를 사용하여 언어 모델(LLM)을 도메인별 작업에 적용
합니다 . LoRA는 사전 훈련된 모델의 가중치를 유지 하고 훈련 가능한 순
위 분해를 도입합니다.
Transformer 아키텍처의 각 계층에 행렬을 추가하여 훈련 가능한 매개
변수의 수를 크게 줄이고
임차인이 낮은 비용으로 다양한 LoRA 모델을 훈련할 수 있도록 허용
비용. LoRA는 많은 인기 있는 미세 조정 프레임워크 에 통합되었습니다
(Mangrulkar et al., 2022). 따라서,
ML 제공업체는 수많은 전문 서비스를 제공해야 합니다.
LoRA는 임차인의 요구에 맞춰 동시에 모델을 제공합니다.

단순히 LoRA 모델을 독립적인 것처럼 제공하기만 하면 됩니다.
처음부터 훈련하면 GPU 리소스가 낭비됩니다. 우리가 가정
n 개의 서로 다른 LoRA 모델을 제공하려면 k개의 GPU가 필요합니다.
LoRA 모델에는 k × n GPU가 필요한 것으로 보입니다 . 이것
간단한 접근 방식은 잠재적인 무게를 간과합니다.
LoRA 모델 간의 상관관계
동일한 사전 훈련된 모델에서.

우리는 다양한 서비스를 제공하는 효율적인 시스템을 믿습니다.
LoRA 모델은 세 가지 설계 지침을 따라야 합니다. (G1)
GPU는 비싸고 부족한 리소스이므로
멀티 테넌트 LoRA 서비스 워크로드를 소규모로 통합
GPU 수를 늘려 전반적인 GPU 활용도를 높입니다. (G2)
이전 연구에서 이미 알아차렸듯이 (Yu et al., 2022), 일괄 처리 는 가장 효
과적인 접근 방식 중 하나입니다.
ML 워크로드를 통합하여 성능을 향상하고
GPU 활용도. 그러나 일괄 처리는 다시 수행할 때만 작동합니다.
쿼스트는 정확히 동일한 모델에 대한 것입니다. 따라서 우리는 다음을 활성화해야 합니다.
다양한 LoRA 모델에 대한 일괄 처리. (G3) 디코드 단계
모델 제공 비용의 주요 요소입니다. 우리
따라서 디코드 단계 성능에만 집중하면 됩니다.
모델 제공의 다른 측면은 덜 중요하며
우리는 주문형과 같은 간단한 기술을 적용할 수 있습니다.
LoRA 모델 가중치 로딩.

이 세 가지 지침을 바탕으로 우리는 디자인하고 구현합니다.
LoRA 모델을 위한 다중 테넌트 제공 프레임워크인 Punica
공유 GPU 클러스터에서. 핵심 참신함 중 하나는 디자인입니다.
새로운 CUDA 커널, Segmented Gather Matrix-Vector
곱셈 (SGMV). SGMV는 여러 개의 서로 다른 작업을 동시에 실행하기
위해 일괄 처리 GPU 작업을 허용합니다.
LoRA 모델. SGMV를 사용하면 GPU는 저장만 하면 됩니다.
사전 훈련된 모델의 단일 복사본을 메모리에 저장하여 두 메모리 측면에
서 GPU 효율성을 크게 향상시킵니다.
그리고 계산. 우리는 이 새로운 CUDA 커널을
일련의 최첨단 시스템 최적화 기술.

SGMV는 다양한 LoRA 모델의 일괄 요청을 허용 하지만 놀랍게도 성능은
미미합니다.
동일한 LoRA 모델을 일괄 처리하는 것과
다양한 LoRA 모델을 일괄 처리합니다. 동시에 LoRA 모델의 온디맨드 로딩
은 밀리초 수준에 불과합니다.
지연 시간. 이를 통해 Punica는 사용자를 통합할 수 있는 유연성을 제공합니다.
제한 없이 소규모 GPU 세트에 대한 요청
LoRA 모델이 이미 GPU에서 실행되고 있는지 확인하세요.

따라서 Punica는 다음 두 가지 방법으로 다중 테넌트 작업 부하를 예약
합니다. 새로운 요청에 대해 Punica는
소규모 활성 GPU 세트에 요청하여 다음을 보장합니다.
최대 용량에 도달합니다. 기존 GPU가 있는 경우에만
완전히 활용되면 Punica는 추가 GPU 리소스를 할당합니다.
기존 요청의 경우 Punica는 주기적으로 요청을 마이그레이션합니다.

*동등기여 1워싱턴대학교
2듀크대학교. 대응: Lequn Chen
<lqchen@cs.washington.edu>.

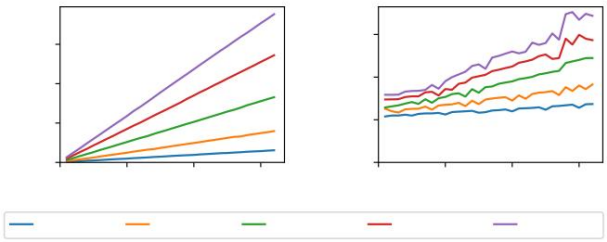

그림 1. 프리필(Prefill) 단계와 디코드(Decode) 단계의 일괄 처리 효과

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

우리는 NVIDIA A100 GPU 클러스터 에서 Llama2 7B, 13B 및 70B 모델 (Touvron et al., 2023) 을 적용한 LoRA 모델을 평가합니다 . 동일한 양의 GPU 리소스가 주어지면 Punica 는 최첨단 LLM 서비스 시스템에 비해 12배 더 높은 처리량을 달성하면서 토큰당 대기 시간은 2ms만 추가합니다.

이 문서는 다음과 같은 기여를 합니다.

- 우리는 다양한 LoRA 모델의 요청을 일괄 처리할 수 있는 기회를 식별합니다 .
- 여러 LoRA 모델을 동시에 실행하기 위한 효율적인 CUDA 커널을 설계하고 구현합니다 .
- 통합을 위한 새로운 일정 메커니즘을 개발합니다. 멀티 테넌트 LoRA 워크로드.

2 배경

먼저 변환기 모델의 텍스트 생성 프로세스를 제시합니다 . 그런 다음 변압기 모델의 LoRA(Low-Rank Adaptation)에 대해 설명합니다 .

2.1 변환기와 텍스트 생성

Transformer 기반 LLM은 일련의 토큰에서 작동합니다. 토큰은 대략 영어 단어의 3/4에 해당합니다. LLM의 운영 은 두 단계로 구성됩니다. 사전 채우기 단계에서는 사용자 프롬프트를 수락하고 후속 토큰과 키-값 캐시(KvCache)를 생성합니다. 디코드 단계에서는 토큰과 KvCache를 수락한 다음 토큰을 하나 더 생성하고 KvCache에 열을 추가합니다. 디코드 단계는 반복적인 프로세스입니다. 생성된 토큰은 다음 단계 의 입력이 됩니다 . 이 프로세스는 시퀀스 끝 토큰이 생성되면 종료됩니다.

변환기 블록에는 Self-Attention 레이어와 MLP(다층 퍼셉트론)가 포함되어 있습니다. 프롬프트의 길이 가 s 이고 주의 헤드 차원이 d라고 가정해 보겠습니다. Prefill 단계의 경우 self-attention 레이어의 계산은 (s, d)×(d, s)×(s, d)이고 MLP 계산은 (s, h)× 입니다. (h, h). 디코딩 단계에서 s가 과거 시퀀스 길이를 나타낸다고 가정하면 self-attention 계층의 계산 은 (1, d) × (d, s + 1) × (s + 1, d) 이고 MLP 계산은 (1) 입니다. ,h)×(h,h). 디코드 단계에서는 입력이 단일 벡터이기 때문에 GPU 활용도가 낮습니다.

그림 1은 다양한 배치 크기에 대한 사전 채우기 단계와 디코드 단계의 대기 시간을 보여줍니다 . GPU의 계산 기능은 사전 채우기 단계에서 완전히 활용됩니다.
사전 채우기 대기 시간은 배치 크기에 비례합니다. 그러나 디코드 단계에서는 그렇지 않습니다. 배치 크기를 1에서 32로 늘리면 디코드 단계 대기 시간이 짧은 시퀀스의 경우 11ms에서 13ms로, 긴 시퀀스의 경우 17ms에서 34ms로 늘어납니다. 이는 일괄 처리가 디코드 단계에서 GPU 활용도를 크게 향상시킬 수 있음을 의미합니다. Orca (Yu et al., 2022) 는 이 기회를 활용하여 효율적인 LLM 서비스 시스템을 구축했습니다 . 이러한 유형의 일괄 처리는 디코드 단계가 긴 출력 길이 응답에 대한 서비스 대기 시간을 주로 결정하기 때문에 특히 중요합니다 .

2.2 낮은 순위 적응(LoRA)

미세 조정을 통해 사전 훈련된 모델이 새로운 영역이나 새로운 작업 에 적응 하거나 새로운 훈련 데이터로 개선될 수 있습니다.
그러나 LLM은 크기 때문에 모든 모델 매개변수를 미세 조정하는 데 리소스가 많이 소요됩니다.

LoRA(Low-Rank Adaptation) (Hu et al., 2022) 는 미세 조정 중에 훈련해야 하는 매개변수의 수를 크게 줄입니다 . 중요한 관찰은 사전 훈련된 모델과 미세 조정 후 모델 간의 가중치 차이가 낮은 순위를 갖는다는 것입니다. 따라서 이 무게 차이는 두 개의 작고 조밀한 행렬의 곱으로 표현될 수 있습니다 . 그러면 LoRA 미세 조정은 작고 조밀한 신경망을 훈련하는 것과 유사해집니다. 공식적으로 사전 훈련된 모델의 가중치를 W ∈ R h1×h2 라고 가정해 보겠습니다. LoRA 미세 조정은 두 행렬 A ∈ R h1×r 및 B ∈ R r×h2 를 훈련합니다. 여기서 r 은 LoRA 순위입니다. W + AB 는 미세 조정 모델 의 새로운 가중치입니다 . LoRA 순위는 일반적으로 원래 차원보다 훨씬 작습니다(예: 4096 대신 16). 빠른 미세 조정 외에도 LoRA는 저장 및 메모리 오버헤드가 매우 낮습니다. 각각의 , 미세 조정된 모델은 모델 중량의 0.1%~1%만 추가합니다. LoRA는 일반적으로 어텐션 메커니즘과 MLP의 쿼리-키-값-출력 투영을 포함하여 변환기 계층 (Dettmers et al., 2023) 의 모든 조밀한 투영에 적용됩니다 . self-attention 작업 자체에는 가중치가 포함되어 있지 않습니다.

공유 GPU 클러스터 에서 다중 테넌트 LoRA 모델을 효율적으로 제공하는 방법은 무엇입니까 ? LoRA는 LLM을 미세 조정하는 효율적인 알고리즘을 제공합니다 . 이제 문제는 LoRA 모델을 효율적으로 제공하는 방법입니다 . 한 가지 접근 방식은 각 LoRA 모델을 독립적인 모델로 간주 하고 기존 LLM 서비스 시스템(예: vLLM)을 사용하는 것입니다. 그러나 이는 서로 다른 LoRA 모델 간의 가중치 공유를 무시합니다.

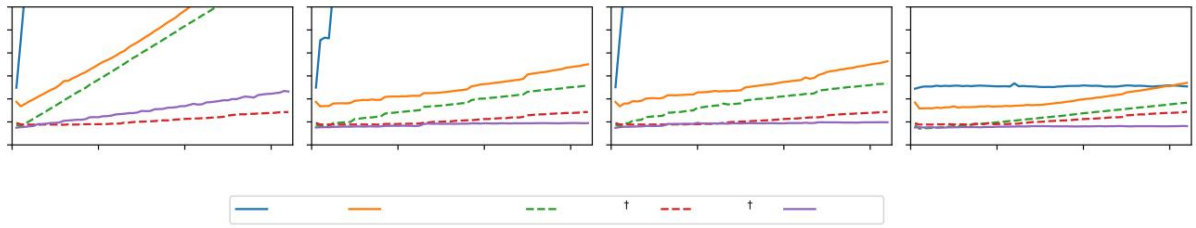

그림 8. LoRA 운영자 구현을 위한 마이크로벤치마크. † Gather와 BMM은 참고용으로 별도로 측정됩니다.

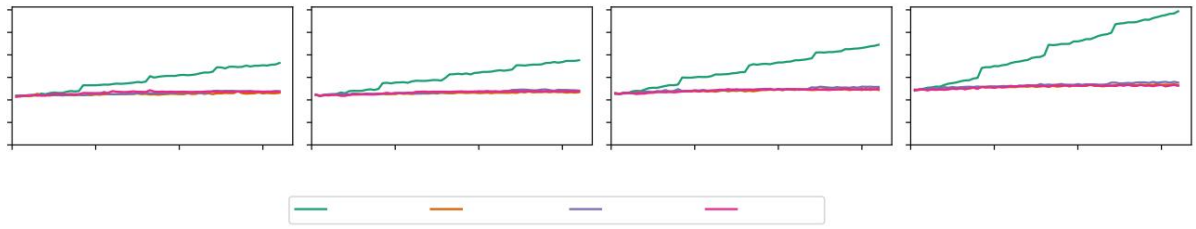

그림 9. 다양한 LoRA 등급의 LoRA 운영자에 대한 마이크로벤치마크.

전반적으로 SGMV는 워크로드에 관계없이 기본 구현보다 훨씬 뛰어난 성능을 발휘합니다.

또한 테스트베드 #1에서 다양한 LoRA 순위의 마이크로벤치마크를 실행합니다 . 그림 9는 LoRA 랭크 8, 16, 32, 64 에 대한 지연 시간을 보여줍니다 . Distinct 의 경우 지연 시간이 점차 증가합니다. 단일 요청 배치의 지연 시간은 4개 랭크 모두에서 약 42µs인 반면, 배치 크기 64는 각각 최대 72µs, 75µs, 89µs 및 118µs입니다. 워크로드에 가중치 공유 (균일, 편향 및 동일) 가 존재하는 경우 대기 시간은 배치 크기 1~ 64 전체에서 약 42~45µs로 거의 동일하게 유지됩니다 .

트랜스포머 레이어 벤치마크 다음으로 LoRA 연산자를 통합한 후 트랜스포머 레이어 성능을 평가합니다 . LLM은 대략적으로 변환기 레이어 스택이므로 레이어 성능이 전체 모델 성능을 결정합니다. 우리는 7B 및 13B 모델 구성과 512 및 2048의 시퀀스 길이를 기반으로 테스트베드 #1에서 레이어 벤치마크를 실행했습니다. 그림 10은 레이어 대기 시간을 나타냅니다.

시퀀스 길이가 짧을수록 일괄 처리 효과가 더 강해집니다. 시퀀스 길이가 512일 때 배치 크기가 1에서 32로 증가하면 대기 시간은 72%만 증가합니다 .
시퀀스가 길수록 self-attention 시간이 길어져 레이어별 일괄 처리 효과가 감소합니다.

커널 마이크로벤치마크와 달리 계층 대기 시간은 다양한 워크로드에서 거의 동일합니다.
이는 LoRA 애드온의 계산 시간이 백본 밀집 투영 및 셀프 어텐션 에 비해 작기 때문입니다. LoRA 모델에 구애받지 않는 이 성능 속성을 통해 다양한 LoRA 모델을 다음과 같이 예약할 수 있습니다.

한 모델이라면. 그러면 우리의 스케줄링 알고리즘은 개별 LoRA 모델 배치 대신 전체 처리량에 초점을 맞출 수 있습니다 . 이것이 바로 Punica를 설계하는 방식입니다.

7.2 텍스트 생성

다음으로 Punica 및 기본 시스템의 텍스트 생성 성능을 연구합니다 .

단일 GPU에서 7B 및 13B 모델 제공 테스트베드 #1의 단일 GPU 에서 Punica 및 기준 시스템을 사용하여 텍스트 생성을 평가합니다 . 단일 GPU 성능은 클러스터 전체 배포의 기본 사례로 사용됩니다. 우리는 1000개의 요청(약 101,000개 토큰 생성)을 생성 하고 선착순 방식으로 각 시스템을 일괄 처리하도록 제한합니다 . 모든 시스템에 대해 최대 배치 크기는 32로 설정됩니다.

Punica는 다양한 LoRA 모델에 걸쳐 일괄 처리할 수 있으며 기본 시스템은 동일한 LoRA 모델에 대해서만 일괄 요청을 수행할 수 있습니다.

그림 11 (a)와 (b)는 각각 7B 모델과 13B 모델 에 대한 결과를 보여줍니다 . Punica는 워크로드에 관계없이 지속적으로 높은 처리량을 제공합니다. Punica는 7B 및 13B 모델에서 각각 1044tok/s 및 693tok/s를 달성합니다. 대부분의 기준선은 동일한 경우 상대적으로 높은 처리량을 달성할 수 있지만 LoRA 모델이 여러 개인 경우 성능이 저하됩니다.

Distinct 의 경우 모든 기본 시스템은 배치 크기 1로 실행되므로 처리량이 낮습니다. 균일(Uniform) 및 왜곡(Skewed) 사례 에서 기본 시스템의 배치 대부분은 배치 크기(1-3)가 매우 작습니다.