Deep Learning Implementation - Lab 9 : Using vllm (paged attention)

20230683 박한비, 20230308 최승희

본 과제에서는 paged attention 을 기반으로 하는 vllm 에 대하여 프레임워크를 이해하고 실제로 구동해보는 것을 목적으로 한다.

Step 1) HuggingFace Transformers vs. vllm

(why should we use vllm?)Complete the code

본 단계에서는 기본적인 HuggingFace Transformers 생성 파이프라인과 vLLM 의 최적화된 엔진을 비교한다. comparison 을 진행하기 위해 기존에 주어진 vllm_vs_transformers.py 를 활용하였다.

vllm_vs_transformers.py 를 성공적으로 실행하기 위해 import vllm 코드를 추가하여 전체 코드를 완성하였다. vllm_vs_transformers.py 를 실행한 결과 아래와 같은 결과가 출력되는 것을 보아 실행이 성공적으로 이루어졌다는 것을 알 수 있었다.

Type	Batch Size	Latency (s)	Throughput (tok/s)
Transformers	16	9.75	420.0
vllm	16	4.46	874.0

batch 크기를 16 으로 동일하게 맞춘 상태에서 Transformers 와 vllm 의 latency, throughput 은 표와 같고, 결과적으로 vLLM 이 훨씬 더 빠르고 더 많은 토큰을 초당 처리할 수 있음을 확인했다.

이는 두 프레임워크의 서빙 **구**조와 메모리 관리 방식에서 비롯된다. vLLM 은 KV 캐시를 고정 크기 블록(block)으로 나누어 관리하는 Paged Attention 을 사용하며 이는 시퀀스 길이가 서로 다른 여러 요청을 동시에 처리할 때 발생하는 메모리 단편화를 줄이고 사용하지 않는 KV 공간을 최소화해서 더 많은 요청을 GPU 메모리에 같이 올릴 수 있다. 그 결과, vLLM 은 같은 배치 크기에서도 실질적으로 더 많은 토큰을 병렬로 처리할 수 있어 throughput 이 증가하고, 디코딩 단계에서 불필요한 메모리 복사·재할당이 줄어 latency 가 감소한다.

또한 vLLM 은 요청을 큐에 넣어 continuous batching 을 수행하며, GPU 가 놀지 않도록 요청들을 효율적으로 묶어서 실행한다. 이 때문에 GPU 연산 자원이 더 빽빽하게 채워지고, **연산 단위당 오버헤드가 줄어** throughput 이 증가한다.

Step 2) Checking the necessity of paged attention within vLLM

2-1) Test with default setting

본 과제에서는 vLLM 내부에서 paged attention 이 실제 성능에 어떤 영향을 미치는지 확인하고자 default setting 에서 PagedAttn ON, OFF 모드를 비교했다. 기존에 주어진 paged_attention_comparison.py 를 활용해 비교를 진행하였다.

vLLM 에는 기본적으로 paged attention 을 disable 하는 옵션이 없기 때문에, block size 를 2048 로 매우 크게 설정하여 이를 simulate 한다.

Model Type	Paged Attention	Batch Size	Latency (s)	Throughput (tok/s)
Llama-2-7b	ON	8	4.21	415.4
Llama-2-7b	OFF	8	8.18	218.5

Paged Attention 을 disable 시킨 경우, latency 가 두 배 가까이 증가하였으며, throughput 역시 절반 수준으로 감소한 것을 확인할 수 있었다.

PagedAttention 은 메모리를 작은 블록 단위로 나눠서 관리함으로써 내부 fragmention 과 불필요한 reservation 을 방지하고, 캐시 효율을 높여 더 좋은 성능을 제공한다.

2-2) Test with smaller & larger batch sizes

본 과제에서는 Batch Size 가 달라질 때 vLLM 의 성능 변화 및 paged attention 여부에 따른 성능 차이의 변화를 확인하고자 한다. 이를 위해 기존에 주어진 paged_attention_comparison.py 의 batch_size 값을 1, 4, 32, 64 로 바꾸며 실행하였다.

Model Type	Paged Attention	Batch Size	Latency (s)	Throughput (tok/s)
Llama-2-7b	ON	1	0.67	64.2
Llama-2-7b	OFF	1	0.66	64.9

Batch Size 를 1 로 설정했을 경우의 latency, throughput 은 위 표와 같고, Paged Attention 여부에 따라 성능이 크게 달라지지 않는 것을 확인할 수 있었다.

Model Type	Paged Attention	Batch Size	Latency (s)	Throughput (tok/s)
Llama-2-7b	ON	4	4.10	186.2
Llama-2-7b	OFF	4	4.05	188.7

Batch Size 를 4 로 설정했을 경우의 latency, throughput 은 위 표와 같고, Paged Attention 여부에 따라 성능이 크게 달라지지 않는 것을 확인할 수 있었다.

Model Type	Paged Attention	Batch Size	Latency (s)	Throughput (tok/s)
Llama-2-7b	ON	32	5.09	1550.7
Llama-2-7b	OFF	32	32.65	240.2

Batch Size 를 32 로 설정했을 경우의 latency, throughput 은 위 표와 같고, Paged Attention 여부에 따라 성능이 크게 달라지는 것을 확인할 수 있었다.

Model Type	Paged Attention	Batch Size	Latency (s)	Throughput (tok/s)
Llama-2-7b	ON	64	7.44	2075.8
Llama-2-7b	OFF	64	65.43	243.2

Batch Size 를 64 로 설정했을 경우의 latency, throughput 은 위 표와 같고, Paged Attention 여부에 따라 성능이 크게 달라지는 것을 확인할 수 있었다.

Batch Size 가 작을 경우 동시에 처리하는 시퀀스 수가 적기 때문에 KV 캐시를 어떻게 쪼개서 배치하느냐(=block 크기)가 성능에 큰 영향을 주지 않아 Paged Attention 의 효과가 미미하다. GPU 가 충분히 여유롭기 때문에 Paged Attention 의 세밀한 메모리 관리가 매우 큰 block size(2048)로 인해 없어지고 단편화 손해가 생긴 것이 크게 체감되지 않는다. 따라서 Paged Attention 의 ON/OFF 여부가 변하여도 latency 와 throughput 의 차이가 크게 발생하지 않는다.

그러나 Batch Size 가 큰 경우에는 동시에 처리하는 시퀀스 수가 많아지기 때문에 Paged Attention OFF 에서는 각 시퀀스가 길이에 상관없이 2048 사이즈의 큰 block 을 통째로 차지해 버리므로, 실제로 쓰지 않는 KV 공간이 많이 생긴다. 이러한 낭비 공간 발생으로 인해 GPU 메모리에 동시에 올릴 수 있는 유효한 시퀀스 수가 줄어들고, 내부적으로 block 재사용/재배치가 더 자주 일어나면서 오버헤드가 커진다. 이로 인해 batch size 를 32, 64 로 늘려도 실질적인 병렬 처리량이 따라 올라가지 못하고 throughput 이 Batch Size 증가에 비례하는 만큼 증가하지 못한다. 또한 메모리 낭비와 재배치/할당 오버헤드 때문에 한 배치를 끝내는 데 시간이 많이 걸리기 때문에 latency 는 Batch Size 증가에 비례하여 크게 늘어난다. 따라서 Paged Attention 의 ON/OFF 여부에 따라 latency 와 throughput 의 차이가 극심하다.

결과적으로 Paged Attention 의 ON 상황에서는 KV 캐시를 block 단위로 잘게 쪼개서 여러 시퀀스를 효율적으로 packing 하므로, batch size 가 증가하면 한 번에 처리하는 병렬 처리량이 크게 늘고, 그에 비해 추가 오버헤드는 작아서 latency 는 완만하게

증가하고 throughput 은 크게 증가한다. Paged Attention 의 OFF 상황에서는 각 시퀀스가 불필요하게 큰 block 을 통째로 차지해서 KV 메모리 낭비가 심하고 많은 시퀀스를 동시에 올릴 때 block 재할당/복사 오버헤드가 커져서 batch size 가 증가하면 어느 순간부터 latency 만 크게 늘고 throughput 은 크게 늘지 않는다.

2-3) Test with another model

기존에 주어진 paged_attention_comparison.py 의 MODEL_NAME = "NousResearch/Llama-2-7b-hf" 를 주석 처리하고 MODEL_NAME = "NousResearch/Meta-Llama-3-8B"의 주석을 풀어 실행하였다.

Model Type	Paged Attention	Batch Size	Latency (s)	Throughput (tok/s)
Meta-Llama-3-8B	ON	8	4.55	402.5
Meta-Llama-3-8B	OFF	8	4.68	391.6

Llama-2-7B 대신 Llama-3-8B 모델을 사용해 동일한 실험을 수행한 결과, PagedAttention 사용 여부에 따른 성능 차이는 매우 제한적이었다. 위의 결과와 같이 latency 와 throughput 모두 큰 차이가 없는 것을 확인할 수 있었다.

이는 모델의 구조적 차이에서 비롯된 것으로 해석된다. Llama-2-7B 는 기존의 Multi-Head Attention(MHA) 구조를 사용하는 반면, Llama-3-8B 는 Grouped Query Attention(GQA)를 적용한다.

GQA 는 query head 를 묶어서 처리하기 때문에 MHA 에 비해 K/V 캐시의 메모리 사용량이 줄어든다. 따라서 block size 를 2048 로 설정했을 때도 PagedAttention 을 사용하지 않아도 GPU 메모리 여유가 충분히 확보되어, 성능 차이가 뚜렷하게 나타나지 않은 것으로 보인다.

이러한 모델 간의 차이는 batch size 가 작을 때에 한정될 것으로 예상된다. 이전 실험에서 확인했던 것처럼, batch size 가 커지면 blocksize=2048 서는 KV 캐시가 모델 구조와 상관없이 연속된 대규모 메모리를 필요로 할 것이기 때문이다.