컨테이너 환경에서 추론 작업 동시 실행 시 GPU 메모리 부족으로 인한 작업 실패 문제 분석†

김형준[1], 강지훈[2,‡]
[1]고려대학교 컴퓨터학과
[2]고려대학교 4단계 BK21 컴퓨터학교육연구단
ledzep0830@korea.ac.kr, k2j23h@korea.ac.kr

Analyzing problem of job failures due to low GPU memory when concurrent running inference jobs in a container environment

HyungJun Kim[1], Jihun Kang[2,‡]
[1]Department of Computer Science and Engineering, Korea University
[2]BK21 Four R&E Center for Computer Science and Engineering, Korea University

요 약

인공지능의 추론 작업은 대규모 연산 자원을 필요로 하는 학습 작업과는 다르게 단일 서버에서 다수의 작업을 동시 실행하는 것이 가능하며, 실행 시간이 상대적으로 빠르다는 특성으로 인해 작업 실행을 위해 컴퓨팅 자원을 점유하고 빠르게 작업을 완료한 후 자원을 반환하기 때문에 다수의 추론 작업을 동시에 운용하는데 용이하다. 하지만, 단일 서버의 컴퓨팅 자원은 제한적이다. 이로 인해 컴퓨팅 자원의 허용 범위 내에서 작업을 운용해야 하며, 허용 범위를 초과하는 규모의 추론 작업이 동시에 실행되면 자원 부족으로 인한 경쟁이 발생한다. 본 논문에서는 컨테이너 환경에서 다수의 추론 작업이 동시에 실행될 때 GPU 메모리 부족으로 인한 작업 실패 문제를 실험을 통해 확인한다. 또한, 다수의 추론 작업 사이에서 발생하는 GPU 자원 경쟁과 실행을 실패하는 추론 작업의 GPU 메모리 낭비로 인한 자원 활용률 저하 문제를 분석한다.

1. 서론

인공지능 기반 서비스는 대량의 데이터를 통한 학습을 통해 AI 모델을 생성하고 생성된 모델을 활용한 추론 작업을 통해 사용자에게 유의미한 결과를 반환한다. AI 학습 작업은 대량의 데이터를 활용하기 때문에 대규모 연산 자원과 긴 연산 시간을 필요로 한다. 하지만, 학습이 완료된 후에는 추론 작업을 기반으로 서비스가 운용된다. 추론 작업은 대규모 연산 자원이 필요한 학습 작업과는 다르게 상대적으로 컴퓨팅 자원을 적게 사용하며, 실행 시간도 빠르다. 이러한 특성은 단일 서버에서 다수의 추론 작업을 동시에 운용하는데 큰 이점을 갖는다.

추론 작업은 앞서 설명한 것과 같이 학습 작업과 비교하여 상대적으로 훨씬 작은 규모의 연산 자원만을 필요로 하며, 일반적으로 실행 시간이 짧기 때문에 연산 자원을 장기간 점유하지 않는다. 이로 인해 서버에서 단일 추론 작업만 운용하게 되면, 연산 자원 전체를 사용하는 대량의 추론 작업이 끊임없이 요청되지 않는 이상 연산 자원의 유휴 상태가 필연적으로 발생한다. 이는 자원 활용률을 저하시키고 불필요한 자원 낭비로 이어진다. 하지만, 단일 서버에서 다수의 추론 작업을 동시에 운용하게 되면, 자원 활용률을 증가시킬 수 있기 때문에 불필요한 연산 자원 낭비를 방지할 수 있다.

앞서 설명한 것과 같이 추론 작업과 같이 상대적으로 연산 자원을 적게 사용하고 실행 시간이 빠르기 때문에 단일 서버에서 동시에 운용하는 것이 자원 활용률 측면에서 효율적이다. 하지만 단일 서버의 연산 자원은 제한적이기 때문에 한 번에 운용할 추론 작업의 개수를 엄격하게 관리해야한다. 특히, AI 작업을 위해 많이 사용되는 GPU 장치의 경우 GPU 메모리의 초과 사용을 허용하지 않는다[1]. 또한, GPU 코어를 활용한 연산을 수행하기 위해서는 GPU 메모리에 데이터가 필수적으로 입력되어있어

† 이 논문은 2023년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2022R1I1A1A01063551)
‡ 교신저자

야 한다[1]. 이로 인해 GPU 기반 연산을 실행할 때 GPU 메모리의 부족은 GPU 기반 작업이 연산을 위한 데이터 입력을 불가능하게 하며, 이는 작업의 실행 실패로 이어진다.

본 논문에서는 컨테이너 환경에서 실험을 통해 동시에 실행되는 추론 작업의 규모가 GPU 자원의 허용 범위를 초과할 때 발생하는 GPU 메모리 부족으로 인한 추론 작업의 실패 문제를 확인하고 이로 인해 발생하는 GPU 자원의 낭비와 추론 작업의 성능에 대해 분석한다. 실험에서는 모든 컨테이너가 동시에 실행하는 추론 작업이 정상적으로 실행을 완료한 경우와 한 개 이상의 컨테이너가 작업 실행을 실패한 경우에서의 추론 작업 성능을 확인하며, GPU 메모리 부족으로 인한 추론 작업의 실행 실패와 그에 따른 GPU 메모리 낭비에 대해 분석한다.

2. 단일 GPU에서 추론 작업의 동시 실행 성능

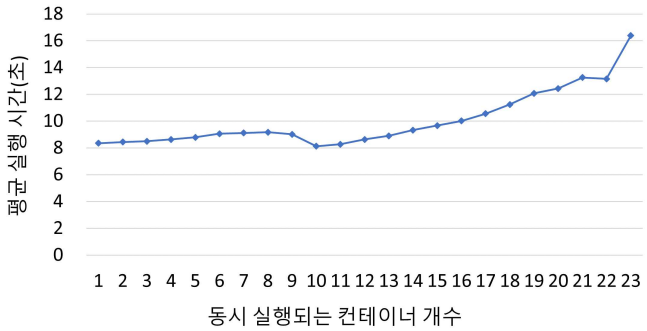
본 논문에서는 Docker[2] 기반 컨테이너 환경을 사용하며, 각 컨테이너는 Tensorflow[3]로 구현된 추론 작업을 실행한다. 추론 작업에 사용할 데이터 셋은 MNIST[4]를 사용한다. 추론 작업에 사용할 데이터 셋은 컨테이너 사이에서 공유되지 않으며, 각 컨테이너의 로컬 저장소에 개별적으로 저장된다. 컨테이너에서 실행될 추론 작업은 MNIST 데이터 셋에서 60,000개의 데이터에 대한 추론 작업을 수행하며, 배치 사이즈는 32로 설정하여 수행한다. 실험 환경은 <표 1>과 같다.

<표 1> 실험 환경

분류	성능
CPU	i9-10920X(3.5GHz, 12 Cores, 24 Threads)
Memory	128GB
GPU	RTX 3090(24,265MiB Memory)
OS	Ubuntu 18.04
Docker	nvidia Docker2[4]

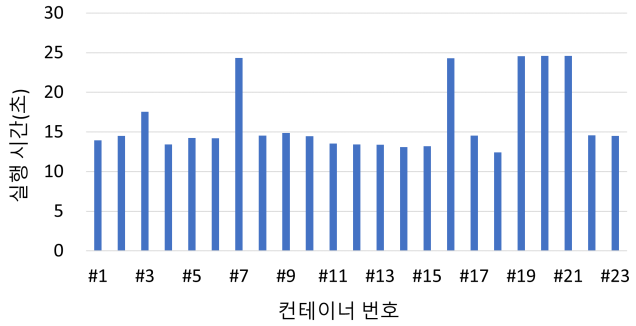
Tensorflow의 경우 기본적으로 GPU 메모리의 단편화를 방지하기 위해 단일 AI 작업이 GPU 메모리 전체를 점유하도록 작동한다[5]. 이로 인해 AI 작업 하나가 실행되면 해당 작업이 GPU 메모리를 전부 점유하게 되어 다른 AI 작업은 실행되지 못한다. 따라서 본 논문의 실험에서 각 추론 작업의 GPU 메모리 사용량을 제한하기 위해 실제 사용량만큼 GPU 메모리를 할당하도록 *tf.config.experimental.set_memory_growth* 메서드[6]를 사용하여 추론 작업을 구현한다. 이를 통해 단일 AI 작업이 GPU 메모리 전체를 점유하지 못하도록 방지하여 다수의 추론 작업이 동시에 실행될 수 있도록 구성하며, 각 추론 작업은 1,205MiB의 GPU 메모리를 사용한다.

이번 장에서 수행할 첫 번째 실험은 추론 작업의 실행 실패가 발생하지 않는 범위에서 추론 작업의 동시 실행 성능을 측정한다. (그림 1)에서는 추론 작업의 동시 실행에 대한 기본 성능을 보여준다.



(그림 1) 추론 작업의 동시 실행 시 평균 실행 시간.

(그림 1)에서 보여주는 것과 같이 추론 작업을 동시에 실행하는 컨테이너의 개수가 2개~23개일 때는 모든 추론 작업이 정상적으로 완료된다. 하지만 동시에 실행되는 추론 작업의 개수가 증가할수록 실행 시간이 증가하는 것을 확인할 수 있다. 특히, 추론 작업을 실행하는 컨테이너가 23개일 때는 22개 이하일 때와 비교해서 실행 시간이 크게 증가한다. (그림 2)는 (그림 1)의 실험 결과에서 동시에 실행되는 컨테이너가 23개일 때 각 컨테이너의 추론 작업 실행 시간을 보여준다.



(그림 2) GPU 메모리 경쟁으로 인한 성능 저하

(그림 2)에서 보여주는 것과 같이 23개의 컨테이너가 추론 작업을 동시에 실행하면, 일부 컨테이너의 추론 작업 실행 시간이 크게 증가한 것을 확인할 수 있다. 23개의 컨테이너가 추론 작업을 동시에 실

행했을 때 실행 시간의 최솟값은 12.418초이며, 최댓값은 24.608초이다. 23개의 컨테이너가 추론 작업을 동시에 실행하게 되면, 총 27,715MiB의 GPU 메모리가 존재해야 GPU 메모리 경쟁 없이 모든 컨테이너가 추론 작업을 완료할 수 있다. 하지만, 실험에서 사용한 GPU는 24,265MiB의 가용 용량을 갖고 있기 때문에 OOM(Out of Memory) 문제가 발생한다. 이로 인해 성능이 저하된 컨테이너는 각 배치사이즈만큼 추론 작업을 실행하다가 GPU 메모리가 부족해지면 다음 배치를 처리하지 못하고 GPU 메모리가 확보될 때 까지 작업을 중지한다. 또한, 추론 작업을 실행하는 모든 컨테이너가 GPU 메모리로 데이터를 입력하는 데이터 입력 작업이 동시에 실행하면서 발생하는 경쟁도 성능 저하의 원인 중 하나이다. 이로 인해 전체 실행 시간이 증가하고 성능 저하가 발생하는 컨테이너는 추론 작업 실행을 위한 메서드의 실행 시점, GPU 메모리의 선점 상태에 따라 무작위로 발생한다.

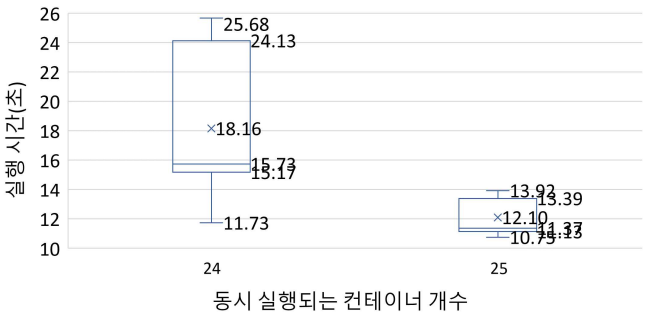
<표 2> 동시 실행되는 추론 작업의 실행 시간 편차

동시 실행되는 컨테이너의 개수	최솟값	최댓값	OOM 발생 컨테이너 개수
2	8.383	8.502	0
3	8.473	8.535	0
4	8.485	8.727	0
5	8.701	8.932	0
6	8.97	9.135	0
7	9.005	9.21	0
8	8.935	9.313	0
9	8.816	9.182	0
10	7.909	8.272	0
11	8.016	8.454	0
12	8.44	8.84	0
13	8.661	9.131	0
14	8.919	9.651	0
15	9.283	10.922	0
16	9.568	10.401	0
17	10.075	10.906	0
18	10.483	15.053	0
19	11.094	15.987	0
20	11.508	16.632	0
21	11.781	15.952	5
22	11.554	13.858	7
23	12.418	24.608	8

<표 2>는 추론 작업을 동시에 실행하는 컨테이너의 개수에 따른 추론 작업의 실행 시간에 대한 최대, 최솟값과 각 배치 크기만큼 추론 작업을 수행하다 한번이상 OOM 에러가 발생한 컨테이너의 개수를 보여준다. <표 2>에서 보여주는 것과 같이 각 컨테이너의 추론 작업 실행 시간에 대한 편차는 추론 작업을 실행하는 컨테이너의 개수가 18개 이상일 때 크게 발생하였으며, 특히, 21개 이상의 컨테이너가 추론 작업을 동시에 실행했을 때는 OOM 문제가 발생한 컨테이너의 개수가 증가했다.

3. GPU 메모리 부족으로 인한 추론 작업의 실행 실패와 GPU 메모리 낭비 문제

다음 실험은 이전 장에서 수행한 실험과 동일한 환경에서 더 많은 개수의 컨테이너를 사용해 추론 작업을 동시에 실행 한다. 본 논문의 실험에서 24개 이상의 컨테이너가 추론 작업을 동시에 실행할 때 실행 실패 문제가 발생했다. 실험에서는 GPU 메모리 부족으로 인한 추론 작업의 실행 실패를 확인하기 위해 24와 25개의 컨테이너를 사용해 추론 작업을 동시에 실행하고 성능과 작업 실패 빈도를 측정한다. 실험에서는 다른 자원 경쟁 요소가 증가되는 것을 방지하고 GPU 메모리에 초점을 두기 위해 24개와 25개의 컨테이너를 활용하여 실험을 수행한다. 추론 작업의 평균 실행 시간, 중앙값, 최대, 최소 편차는 (그림 3)에서 보여준다.



(그림 3) 추론 작업 실패가 발생했을 때의 실행 시간

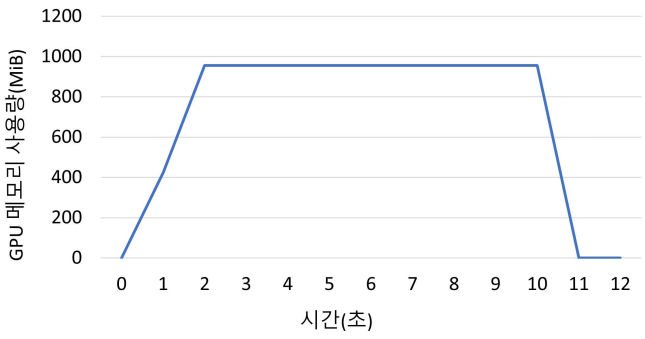
(그림 3)의 그래프에서 실행 시간의 1분위와 3분위는 상자의 위, 아래 경계선이며, 상자 내부의 선은 중앙값을 x 표시는 평균을 나타낸다. 그리고 상자 외부의 위, 아래 선은 각각 최대, 최솟값을 나타낸다. 실험 결과에서 보여주는 것과 같이 추론 작업을 실행하는 컨테이너의 개수가 24개일 때 실행 시간의 편차가 크게 증가한다. 그리고 25개의 컨테이너가 추론 작업을 동시에 실행할 때는 실행 시간에 대한 편차가 상대적으로 매우 적게 발생하며, 전체 실행 시간도 감소한 것을 확인할 수 있다.

하지만 <표 3>에서 보여주는 것과 같이 24개의 컨테이너가 동시에 실행될 때에는 6개의 컨테이너가 작업 실행을 실패했으며, 25개의 컨테이너가 동시 실행되었을 때에는 15개의 컨테이너가 추론 작업의 실행을 실패하고 추론 작업이 강제 종료된다.

<표 3> 추론 작업의 실행 시간 및 실행 실패

번호	동시 실행 컨테이너 개수			
	24개		25개	
	실행 시간	실행 완료 여부	실행 시간	실행 완료 여부
1	11.728	x	13.800	o
2	15.662	o	13.788	o
3	24.017	o	11.044	x
4	15.650	o	13.413	o
5	15.126	o	11.322	x
6	18.208	o	11.187	x
7	24.174	o	13.022	o
8	13.838	o	11.118	x
9	14.322	o	11.209	x
10	15.724	o	11.149	x
11	13.803	o	12.993	o
12	15.354	o	11.169	x
13	15.305	o	11.507	x
14	15.756	o	11.366	x
15	25.498	x	13.452	o
16	18.147	o	11.089	x
17	25.678	x	11.287	x
18	13.992	o	11.068	x
19	25.569	x	13.365	o
20	15.810	o	13.182	o
21	25.592	x	11.027	x
22	15.740	o	10.750	x
23	15.447	o	13.924	o
24	25.606	x	13.847	o
25	-	-	11.385	x

추론 작업을 실행할 때 Tensorflow 프레임워크는 추론 작업을 위한 CUDA 라이브러리를 한다. GPU 메모리 부족으로 인해 라이브러리를 호출하고 활용하는데 필요한 가용 GPU 메모리가 존재하지 않아 라이브러리 초기화를 실패하고 프로세스가 종료된다. 특히 25개의 컨테이너가 추론 작업을 동시에 실행했을 때 60%의 컨테이너가 작업 실행을 실패하고 프로세스가 종료된다. 이로 인해 GPU 자원 경쟁이 줄어들고 실행 시간이 상대적으로 감소한다.



(그림 4) 실패한 추론 작업의 GPU 메모리 사용 패턴

또한, (그림 4)에서 보여주는 것과 같이 작업을 실패한 추론 작업은 초기 GPU 메모리에 데이터 입력 작업을 실행하는 도중에 필요로 하는 GPU 메모리 용량인 1,205MiB 만큼 점유하지 못하고 작업 실행을 실패하며, 프로세스가 완전히 종료될 때 까지 점유한 상태로 유지된다. 작업 실행을 실패한 추론 작업은 자신이 필요로 하는 GPU 메모리 용량 중 일부만 할당된 상태에서 프로세스가 종료되고 GPU 메모리를 반환하기 전까지 해당 GPU 메모리 영역을 활용하지 못하는 문제가 발생한다. 이로 인해 컨테이너가 증가할수록 제한된 GPU 메모리를 여러 컨테이너가 나누어 사용해야하기 때문에 온전하게 GPU 메모리를 점유하고 작업을 성공하는 컨테이너의 개수가 줄어드는 문제가 발생한다. 또한, 본 논문에서는 여러 번의 실험을 수행한 결과 작업 실행을 실패하는 컨테이너가 7개~16개까지 다양하게 발생하는 것을 확인했다.

4. 결론 및 향후 연구

본 논문에서는 실험을 통해 컨테이너 기반 클라우드 환경에서 다수의 추론 작업을 동시에 실행했을 때 발생하는 성능 저하와 추론 작업의 실패 문제에 대해 분석했다. 이전 장의 실험에서 설명한 것과 같이 동시 실행되는 컨테이너의 개수가 증가할수록 성능이 저하되고 일정 수준을 초과하면 컨테이너 사이에서 추론 작업의 실행 시간에 대한 편차의 증가를 확인했다. 또한, 동시 실행되는 컨테이너가 24개 이상일 때에는 추론 작업을 위한 라이브러리 초기화 실패로 인해 추론 작업의 실행을 실패하는 문제와 함께 활용되지 못하는 GPU 메모리 영역이 존재하는 것도 확인했다. 본 논문의 추후 연구로는 앞서 설명한 실험 결과를 기반으로 다중 추론 작업 혹은 전이 학습 운용 환경에서 GPU 메모리 부족으로 인한 작업 실패 문제를 방지하기 위한 컨테이너 배치 및 작업 순서 스케줄링 기법을 연구할 계획이다.

참고문헌

[1] CUDA, https://docs.nvidia.com/cuda
[2] Docker, https://www.docker.com/
[3] TensorFlow, https://www.tensorflow.org/?hl=ko
[4] MNIST, http://yann.lecun.com/exdb/ mnist/
[5] TensorFlow GPU, https://www.tensorflow.org/guide/gpu
[6] TensorFlow API, https://www.tensorflow.org/api_docs/python/tf/config/experimental/set_memory_growth