서버리스 컴퓨팅을 활용한 딥러닝 서빙 시스템

김현준, 이경용
국민대학교 컴퓨터공학부
4u_olion@naver.com, leeky@kookmin.ac.kr

Deep Learning Using Serverless Computing

Hyunjune Kim, Kyungyong Lee
Kookmin University

요 약

딥러닝은 대규모 표본 데이터를 사용하여 복잡한 문제를 분해와 분류를 통해 쉽게 해결할 수 있도록 도와주는 분석 작업이다. 최근 여러 기업이나 단체에서 축적된 데이터를 공유하면서 딥러닝 학습에 필요한 대규모 표본 데이터는 충분히 쉽게 얻을 수 있다. 그러나 대규모 표본 데이터를 학습하기 위해선 GPU와 같은 고가의 장비들이 필요하다. 이러한 비용적 부담으로 인해 누구나 쉽게 딥러닝을 활용하지 못 하는 단점이 있다. 따라서 우리는 비용을 최소화하기 위해 서버리스 컴퓨팅을 활용한 딥러닝 서빙 시스템을 구현했으며, 본 논문을 통해 제안된 구조가 많은 이들에게 비용 부담을 줄이고 적극적으로 활용될 수 있도록 공유한다.

1. 서 론

서버리스 컴퓨팅은 필요한 작업을 쉽게 확장할 수 있고, 할당한 컴퓨팅 자원과 사용된 시간만큼 비용을 지불한다. 이러한 이유로 최근 다양한 기업이나 연구소에서 딥러닝(deep learning)과 같이 대규모 데이터가 필요한 작업을 최소의 비용으로 효율적으로 처리하기 위해 서버리스 컴퓨팅을 활용한 시스템 구성에 많은 연구가 이루어지고 있다[1][2][3][4].

딥러닝은 이미지와 같은 학습 데이터들을 거듭 분해하고, 고유한 특징을 분류하는 작업이며, 특징을 정확하게 분류하기 위해 반복적으로 가중치를 조정하는 학습 작업을 수행한다. 일반적으로 딥러닝은 대규모 데이터를 활용하여 학습하며, 최근 Kaggle과 같은 온라인 단체에서 다양하고 방대한 데이터가 공유되면서, 충분할 만큼 대규모 학습 데이터를 수집할 수 있다.

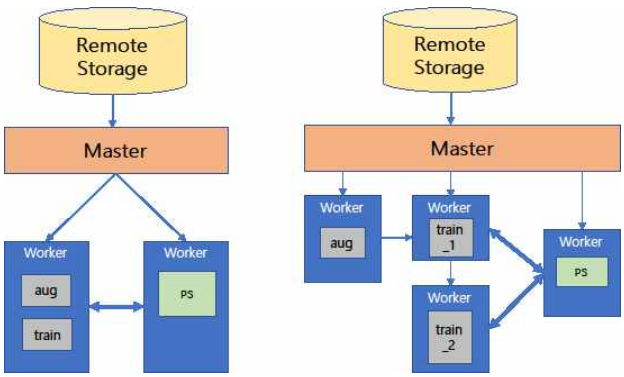

그림 1 대규모 표본 데이터 딥러닝 시스템 구조

그러나 대규모 데이터를 사용하는 딥러닝 작업은 GPU와 같은 고가의 장비가 필요하며, 이러한 원인은 많은 이들에게 딥러닝은 쉽게 활용하기 어려운 작업으로 받아들여진다. 게다가 학습을 마친 딥러닝 모델은 분류의 기능만 제공하면 되며, 추가 학습이 필요하더라도 소규모로 구성된 학습 데이터는 GPU와 같은 고가의 장비가 불필요하다. 먼저, 우리는 최근 딥러닝과 같은 시스템을 서버리스로 구현한 사례[5][6]를 확인했다. 그러나 이들 시스템에서 대규모 데이터를 사용하는 딥러닝 시스템은 여전히 GPU와 같은 고성능 컴퓨팅에 의지하며, 분류의 기능만 서버리스로 구현했다.

따라서 우리는 기존 사례가 서버리스 컴퓨팅을 효과적으로 구현하지 못한 것에 주목했다. 대규모 데이터에서 딥러닝 작업은 학습 데이터를 분해 및 분류하는 서빙 과정에서 가장 많은 컴퓨팅 자원이 필요하다. 따라서 본 논문은 서버리스 컴퓨팅을 활용한 병렬 작업 극대화로 딥러닝 작업을 필요로 하거나 시도하려는 이들에게 비용 부담을 덜어줄 수 있는 서빙 시스템을 제안 및 공유한다.

2. 관련 연구

2.1 딥러닝(Deep Learning)

딥러닝은 음성신호나 비디오, 이미지와 같은 고차원 데이터(원본)를 고유한 특징으로 표현할 수 있도록 저차원 데이터(특징)로 분해하는 작업이다. 분해 작업들은 다층 구조의 서로 다른 은닉층(hidden layer)으로 구성되며, 각 은닉층은 특수한 필터로 입력 데이터를 분해 및 분류하며, 컨볼루션(convolutional) 층이나 완전연결(fully connected) 층 등이 있다. 필터의 각 요소는 가중치라 불리는 값들로 구성되며, 반복 학습으로 가중치를 갱신한다. 딥러닝의 전체 작업은 데이터의 서빙과 가중치의 학습 단계로 이루어진 단순 반복 작업이다. 딥러닝은 수십 개 이상의 데이터를 병렬 처리할 수 있는 알고리즘으로 설계되어 있다. 대규모 데이터를 위한 딥러닝 프레임워크로는 Tensorflow나 MXNet이 있으며, 이들은 GPU와 같은 고비용 장비인 Worker를 사용하여 그림 1과 같이 데이터나 모델을 분산시킨다. 그러나 분산 환경을 위한 Worker의 비용은 부담이 크며, 다양한 산업에서 딥러닝을 시도하는 것에 큰 장벽이 되고 있다. 이러한 이유로 우리는 대규모 데이터에서 작업의 병렬화를 극대화할 수 있고 각 작업의 사용된 시간과 할당된 자원만큼 비용을 청구하는 서버리스 컴퓨팅을 활용한 딥러닝 서빙 시스템을 구축하고자 한다.

2.2 서버리스

서버리스 컴퓨팅은 Cloud의 FaaS 같은 서비스 어플리케이션을 활용한다. 대표적으로 AWS Lambda 및 Google Function이 있으며, Cloud에 사용자가 원하는 작업을 함수로 등록하고 작업에 필요한 컴퓨팅 자원을 할당하며, 할당된 크기와 서비스 실행 시간만큼 과금이 부과되는 구조다. 작업이 끝나면 곧바로

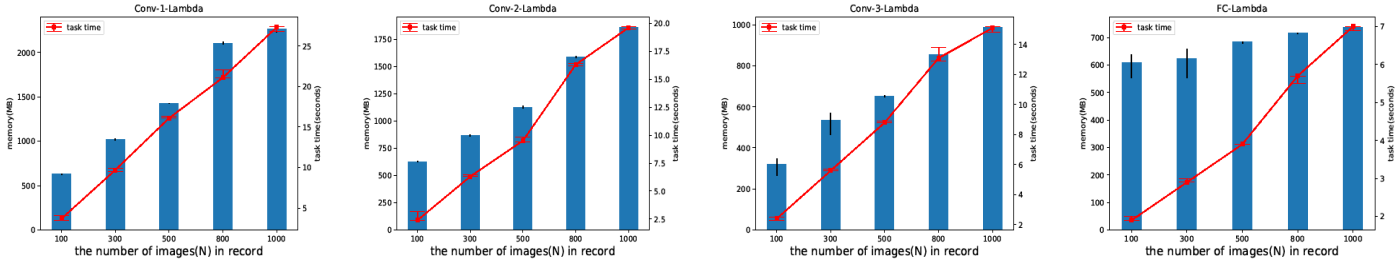

그림 3 AWS Lambda 별 레코드에 저장된 이미지 개수에 따른 메모리 사용량 및 작업 시간

과금이 중단되기 때문에 효율적으로 비용을 관리할 수 있다. 특히, 마이크로서비스 구조를 지향하기 때문에 작업의 병렬화를 극대화할 수 있으며, 필요한 작업을 빠르게 확장할 수 있다.

2.3 AWS Lambda

AWS를 활용한 서버리스 컴퓨팅의 핵심에는 FaaS 서비스 Lambda가 있으며, 할당한 메모리와 실행된 시간만큼 과금이 발생한다. AWS Lambda는 128-3008MB까지 메모리를 할당할 수 있으며, CPU는 최대 메모리와 할당한 메모리의 비율만큼 성능이 결정된다[7]. 각 AWS Lambda는 최대 512MB까지 파일을 저장할 수 있다.

2.4 프로토콜버퍼(Protocol buffer)를 사용한 레코드

서버리스 컴퓨팅을 활용한 딥러닝 시스템은 원격 저장소와 딥러닝 작업을 위한 컴퓨팅 시스템이 분리된 구조다. 이러한 구조에서 모든 학습용 데이터를 일일이 컴퓨팅 시스템으로 전달하는 것은 작업 시간과 비용 증가의 원인이 되며, 따라서 네트워크에서 발생하는 데이터 교환을 최소화하는 것은 매우 중요한 작업이다. 우리는 서버리스 컴퓨팅에서 데이터 교환을 최소화하기 위해, Google에서 개발한 프로토콜버퍼를 사용했다. 프로토콜버퍼는 데이터 정보를 바이트 값으로 대체하여 저장한다. 이와 같은 작업을 직렬화(serialization)라고 한다. 프로토콜버퍼는 다양한 언어(Python, Java, C++)를 지원하며, 사용법이 매우 단순하다. 우리는 프로토콜버퍼로 여러 이미지 데이터를 하나의 파일에 저장했다. 본 논문에서는 이를 레코드(Record)라 정의했으며, 네트워크에서 데이터 교환을 최소화할 수 있었다.

3. 제안되는 구조

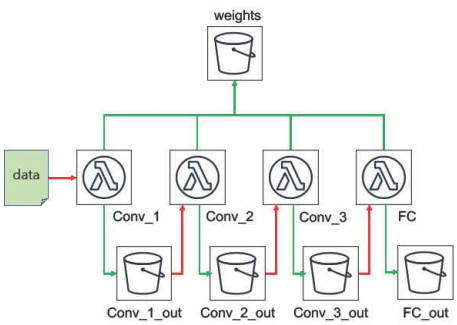

그림 2 서버리스 딥러닝 시스템 구조

우리가 제안한 시스템은 AWS Cloud의 Lambda와 S3를, 딥러닝 프레임워크는 Tensorflow를 사용하여 구현했다. 우리는 서버리스의 병렬화 비용과 GPU를 비교하기 위해, AWS Lambda에 3008MB 메모리를 할당하여 성능을 최대한 끌어올렸다. 서버리스 딥러닝에서 사용한 모델은 3개의 컨볼루션층(convolutional layer)과 4개의 완결연결층(fully connected)으로 구성된 Alexnet을 사용했다. 그리고 같은 지역 내 AWS Lambda들과 매우 빠른 속도로 데이터를 송수신할 수 있는 S3에 레코드를 저장했다. 따라서 제안된 시스템의 구조는 그림 2와 같이 컨볼루션층(Conv_1, Conv_2, Conv3)과 완전연결층(FC)이 서로 다른 AWS Lambda로 분산했으며, 가중치(Weight)를 포함한 중간 결과물(Conv_1_out, Conv_2_out, Conv_3_out, FC_out)은 AWS S3 버킷에 나누어 저장했다. 본 논문에서 제안된 시스템은 학습 시스템을 제외했으며, 오로지 서빙 시스템의 성능과 비용을 분석 및 비교한다.

4. 실험 및 평가

4.1 레코드 크기 별 최대 메모리 사용량

우리는 AWS의 Lambda를 활용한 분산 작업으로 기존 CPU 인스턴스들보다 대규모 데이터를 빠르게 분해하고 분류할 수 있는 비용 효율적인 딥러닝 서빙 시스템 구현이 목적이다. 그래서 AWS Lambda에 필요한 메모리 크기와 작업 시간만큼 발생하는 비용 구조를 이해할 필요가 있었다. 우리는 먼저 레코드 별로 저장된 이미지 개수에 따른 메모리 사용량과 작업 시간을 분석했다. 우리의 서버리스 컴퓨팅 시스템 환경은 AWS Lambda에 최대 메모리(3008MB)를 할당했으며, 런타임으로 Python을 사용한다. 학습 데이터는 ImageNet을 사용하고, 딥러닝 프레임워크로 Tensorflow을 사용했다.

우리가 준비한 실험에서 AWS Lambda가 최대 몇 개의 이미지가 저장된 레코드를 사용할 수 있는지 실험했다. 실험 방법은 AWS Lambda 별로 레코드에 저장된 이미지 개수를 늘리며 메모리 사용량과 각각의 분산된 모델을 통과하는 서빙 작업 시간을 측정하는 것에 초점을 두며 그림 3과 같은 결과를 얻었다. 그림 3의 4개의 그래프는 딥러닝 모델이 분산된 AWS Lambda 별 메모리 사용량과 작업 시간을 측정하여 나타낸 것이다. 각 그래프에서 x축은 레코드에 저장된 이미지 개수를 보여준다. 좌측 y축은 메모리의 최대 사용량이며, 우측 y축은 레코드의 작업 시간을 초 단위로 나타냈다. 메모리의 최대 사용량은 막대 그래프이며, 작업 시간은 꺾은선 그래프로 표현했다.

우리의 실험에서 하나의 레코드는 최대 1,000개의 이미지를 저장할 수 있었다. 그림 3에 4개의 그래프를 보면 Conv_1 Lambda가 메모리 사용량과 작업 시간이 가장 많았고, 그 중 FC Lambda가 가장 적은 것을 볼 수 있다. 이유는 딥러닝에서 분해 작업이 거듭되면 데이터를 구성하고 있는 특징이 감소됨에 따라 메모리 사용량이 감소하고 작업 시간이 줄어든다. 우리는 그림 3에서 그래프 별로 레코드마다 메모리 사용량과 작업 시간 사이의 간격을 비교했으며, 레코드에 저장된 이미지 수가 적어질수록 메모리의 사용량보다 작업 시간이 더 많이 감소하는 것을 확인했다. 특히, 이러한 간격은 Conv_2와 Conv_3에서 더 크게 나타났으며, FC Lambda에서 매우 극단적으로 벌어지

는 것을 목격했다. 이러한 간격은 딥러닝의 층이 깊어질수록 분해 작업에 의해 특징이 줄어듦에 따라 메모리 사용량이 급격히 줄어들며, 유휴자원이 증가하기 때문이다. 전체적인 유휴자원의 비율은 그림 3에서 레코드에 저장된 이미지가 500개보다 적을 때 급격히 많아진다.

4.2 서버리스 컴퓨팅 분산 작업 성능 및 비용

우리는 서버리스 컴퓨팅에서 분산 작업의 성능과 비용의 이득을 분석하기 전, 우리가 제안한 서빙 시스템이 활용 가능한지 확인해야 했으며, NVIDIA TITAN X Pascal GPU 1대와 비교했다. 그림 4는 레코드당 이미지 수를 늘렸을 때 작업 시간을 나타낸다. x축은 레코드당 증가된 이미지의 개수이며, y축은 하나의 레코드가 서빙에 소모된 컴퓨팅 시간을 나타냈다.

GPU에서 1,000개의 이미지가 저장된 레코드에서 서빙 작업은 약 176초가 소요됐다. 반면에, 우리가 제안한 시스템은 레코드 20개를 분산 및 병렬 처리하여 99초 시간이 소요됐다. 그리고 500개의 이미지가 저장된 레코드는 57초, 100개의 이미지가 저장된 레코드는 28초가 소요됐다. 우리의 시스템은 AWS Lambda의 제한된 자원을 적극적으로 활용하여 규모가 큰 딥러닝 모델을 은닉층별로 분산 및 병렬화하고, 최대한 많은 이미지가 AWS Lambda에 저장되도록 서빙 시스템을 구현했다. 결과적으로 고성능 컴퓨팅에 가까운 성능을 얻을 수 있다.

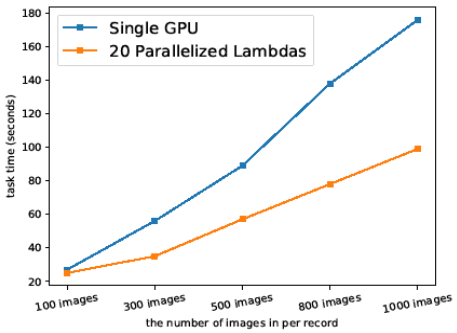

그림 4 서버리스 컴퓨팅과 GPU 성능 비교

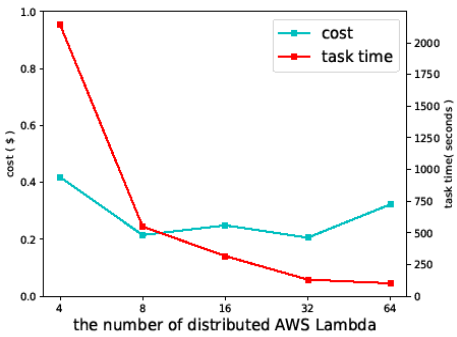

그림 5 서버리스 컴퓨팅
분산 작업 성능 및 비용

이전 실험을 바탕으로, 우리는 대규모 데이터 서빙에서 분산 작업 수에 따라 성능과 발생하는 비용이 달라질 것으로 생각했다. 따라서 100,000개 이미지 데이터를 준비했으며, 레코드별로 1,000개씩 저장했다. 그림 5는 AWS Lambda의 분산 작업 수 증가에 따른 성능과 비용의 변화를 보여준다. 그림 5에서 빨간색 그래프는 작업 시간을 나타내며, 청록색 그래프는 비용 그래프를 나타낸다. x축은 AWS Lambda의 분산 작업 수를 보여주며, 좌측 y축은 비용을, 우측 y축은 총 작업 시간을 보여준다. 그림 5의 결과에서 AWS Lambda의 분산 작업 수가 증가할수록 서빙 작업 시간이 줄어드는 것을 볼 수 있다. 우리의 실험에서 GPU는 약 902초의 컴퓨팅 시간이 필요했다. 그러나 제안된 시스템은 AWS Lambda의 분산 작업 수가 8개 이상일 때, 필요한 컴퓨팅 시간이 GPU보다 적었다. 또한, 그림 5는 AWS Lambda의 분산 작업 수가 32부터 64까지는 제안된 서빙 시스템의 성능에 변화가 거의 없으며, 비용이 증가하는 결과를 보여준다. 따라서 AWS Lambda의 분산 작업을 증가시킬 경우 비용만 증가할 수 있는 점에 주의해야 한다.

4.3 AWS Lambda 별 메모리 최소화 및 비용 비교

우리는 4.1에서 각각의 AWS Lambda 별로 메모리 사용량이 서로 다른 점과 이미지 개수에 따른 유휴자원 증가에 주목했으며, 이를 바탕으로 표 1과 같이 1,000개의 이미지가 저장된 레코드와 500개의 이미지가 저장된 레코드 2가지 실험을 준비했다. memMax는 AWS Lambda에서 설정 가능한 최대 메모리(3008MB)를 할당한 기존 설정이며, memMin은 AWS Lambda 하나에서 이미지 개수에 따른 최소로 필요한 메모리를 할당한 설정이다. 표 2는 작업 시간과 발생하는 비용을 나타냈다. 표 2의 결과에 따르면 AWS Lambda에 과하게 메모리를 할당하는 것은 불필요한 비용을 발생시킬 수 있다. 따라서 우리는 낭비되는 메모리를 제거함으로써 약 2배의 비용을 절약할 수 있었다.

표 1 AWS Lambda 별 메모리 설정

	AWS Lambda 메모리 크기(MB)			
레코드 당 표본 수	Conv-1	Conv-2	Conv-3	FC
memMax-1000	3008	3008	3008	3008
memMin-1000	2048	1280	1024	768
memMax-500	3008	3008	3008	3008
memMin-500	1280	1024	768	768

표 2 비용 최소화 실험 비교

레코드 당 표본 수	작업 시간(초)	작업 비용(USD)
memMax-1000	99	0.0970
memMin-1000	134	0.0582
memMax-500	57	0.0558
memMin-500	90	0.0281

5. 결 론

딥러닝 시스템은 데이터 서빙에서 가장 많은 컴퓨팅 자원을 필요로 하며, 이를 해결하기 위해 서버리스 컴퓨팅을 활용함으로써 비용 효율적인 접근 방법에 대한 가능성을 연구했다. 본 연구를 마치며, 우리는 서버리스 컴퓨팅을 활용한 딥러닝 서빙 시스템 결과가 학습 시스템과의 결합에서도 긍정적인 결과를 가져올 것으로 기대하며, 다양한 분야에서 딥러닝을 시도하려는 이들에게 비용적 부담을 최소화하고 즉흥적인 성과를 이뤄낼 수 있는 도움을 줄 수 있을 것으로 희망한다.

6. 사 사

본 연구는 과학기술정보통신부 정보통신기획평가원의 ICT 연구과제 (2017-0-00396), 한국연구재단 이공분야 기초연구 사업 (NRF-2016R1C1B2015135) 및 선도연구센터지원사업 (NRF-2015R1A5A7037615) 의 지원을 받아 수행됨.

6. 참조문헌

[1] Kim, Y., & Lin, J. (2018, July). Serverless data analytic
s with flint. In 2018 IEEE 11th International Conference on
Cloud Computing (CLOUD) (pp. 451-455). IEEE.
[2] Lynn, T., Rosati, P., Lejeune, A., & Emeakaroha, V. (2
017, December). A preliminary review of enterprise serverle
ss cloud computing (function-as-a-service) platforms. In 2
017 IEEE International Conference on Cloud Computing Te
chnology and Science (CloudCom) (pp. 162-169). IEEE.
[3] Werner, S., Kuhlenkamp, J., Klems, M., Muller, J., & T
ai, S. (2018, December). Serverless Big Data Processing u
sing Matrix Multiplication as Example. In 2018 IEEE Internat
ional Conference on Big Data (Big Data) (pp. 358-365). IE
EE.
[4] Sampe, J., Sanchez-Artigas, M., Garcia-Lopez, P., &
Paris, G. (2017, December). Data-driven serverless functions f
or object storage. In Proceedings of the 18th ACM/IFIP/USENI
X Middleware Conference (pp. 121-133). ACM.
[5] Ishakian, V., Muthusamy, V., & Slominski, A. (2018, Ap
ril). Serving deep learning models in a serverless platform.
In 2018 IEEE International Conference on Cloud Engineerin
g (IC2E) (pp. 257-262). IEEE.
[6] Carreira, J., Fonseca, P., Tumanov, A., Zhang, A., &
Katz, R. (2018). A Case for Serverless Machine Learning. I
n Workshop on Systems for ML and Open Source Software
at NeurIPS (Vol. 2018).
[7] Kim, J., Park, J., & Lee, K. (2019, June). Network Res
ource Isolation in Serverless Cloud Function Service. In 20
19 IEEE 4th International Workshops on Foundations and A
pplications of Self* Systems (FAS* W) (pp. 182-187). IEE
E.