**데이터와 모델의 특성에 따른 분산 머신 러닝 아키텍쳐 별 성능 분석**

**분산 TensorFlow와 Horovod의 트래픽 특성 분석**

**1. 과제의 필요성**

**1.1. Abstract**

1.1.1. Research Background

데이터가 크거나 모델이 큰 경우 머신 러닝 모델의 교육 효율성을 향상시키기 위해 일반적으로 다중 GPU 분산 교육이 채택된다. 병렬 모드에 따르면 분산 학습은 일반적으로 데이터 병렬 처리와 모델 병렬 처리로 나눈다.

모델 병렬 처리 : 분산 시스템의 다른 GPU는 네트워크 모델의 다른 부분을 담당한다.

데이터 병렬 처리 : 다른 GPU에는 동일한 모델의 여러 복사본이 있으며 각 GPU에는 다른 데이터가 할당 된 다음 모든 GPU의 결과를 어떤 특정 방식으로 결합한다.

  Horovod는 Uber 오픈 소스를 위한 또 다른 딥러닝 도구이다. Facebook의 "정확한 대형 미니 배치 SGD : 1 시간 만에 ImageNet 교육"[1]과 Baidu의 Ring Allreduce [2]의 이점을 활용하여 사용자가 분산 교육을 받을 수 있도록 도와준다. 이 기사에서는 이 프레임 워크의 기능을 간략하게 소개했었다..

 최근에 딥러닝은 이미지 처리, 음성 인식 및 예측에서 엄청난 발전을 가져 왔다. Uber는 자율 주행 검색 경로에서 방어 사기에 이르기까지 딥러닝을 비즈니스에 적용하고 또한 딥러닝을 통해 데이터 과학자와 엔지니어는 사용자에게 더 나은 경험을 제공 할 수 있다.

  TensorFlow는 Uber에서 선호하는 딥 러닝 라이브러리library가 되었다. 이 프레임 워크는 현재 가장 널리 사용되는 오픈 소스 딥 러닝 프레임 워크이므로 새로운 개발자에게 매우 친숙하다. 고성능 및 저수준 모델 세부 디버깅 기능을 결합했다. 예를 들어, 맞춤형 Nvidia CUDA 도구를 사용하면서 고급 API 인 Keras를 사용할 수 있다. 또한 TensorFlow는 실험적 탐색에서 프로덕션 레벨 모델의 배포, 클라우드 서버, 모바일 APP 및 자율 주행 차량에 이르기까지 다양한 딥 러닝 사용 사례에 대한 엔드 투 엔드 지원을 제공한다..

1.1.2. 연구 목표

 본 제안서는 TensorFlow와 Horovod의 성능을 테스트하고 두 방법의 성능 차이를 비교하고자 한다. 두 가지 방법을 분석 한 후 실제 테스트를 통해 두 가지 방법의 장단점을 비교하고 향후 적용 시나리오를 예측하고 평가하려고 한다.

**1.2. Introduction**

  회사와 연구소에서 TensorFlow에 대해 점점 더 많은 머신 러닝 모델을 교육함에 따라 프로젝트의 데이터 및 컴퓨팅 성능 요구 사항이 급격히 증가하고 있다. 대부분의 경우 모델은 단일 또는 다중 GPU 플랫폼 서버에서 실행될 수 있지만 데이터 세트가 커지고 트레이닝 시간이 증가함에 따라 트레이닝이 1 주일 이상이 소요될 수 있습니다. 따라서 개발자는 교육을 배포 할 수있는 방법을 찾아야 한다. 표준 분산 TensorFlow는 많은 새로운 개념을 도입한다.

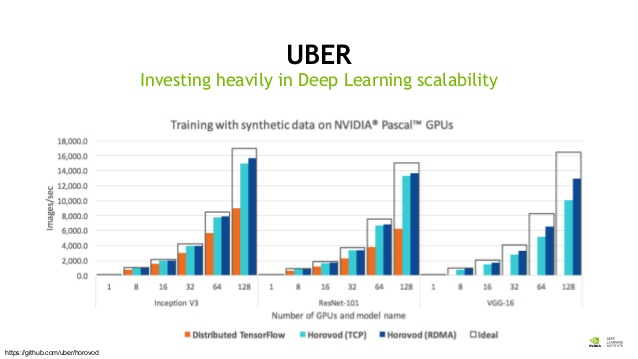
tf.Server(),

tf.ClusterSpec(),

tf.train.SyncReplicasOptimizer()

and tf.train.replicas\_device\_setter()

경우에 따라 최적화 할 수 있지만 트레이닝 속도를 늦추는 버그를 찾기가 어렵다. 또한 학습 비용 증가 외에도 표준 TensorFlow 벤치 마크 제품을 사용하여 128 개의 NVIDIA Pascal GPU를 테스트 할 때 Inception V3 및 ResNet-101가 GPU 전력을 거의 절반을 낭비한다.



<그래프 1>

1.2.1. TensorFlow

"데이터 병렬 처리"접근 방식에는 분산 된 교육에서 데이터의 병렬 분할 및 여러 노드에 대한 교육이 포함된다. 동기화의 경우, 서로 다른 데이터 배치의 Gradient는 서로 다른 노드에서 개별적으로 계산되지만 각 노드의 모델 사본에 일관된 업데이트를 적용하기 위해 노드 간 평균화된다. 특정 알고리즘은 다음 단계로 단순화 할 수 있다.

1. 각 트레이닝 스크립트의 여러 사본을 실행한다.

a) 데이터 블록을 읽는다. b) 모델에 입력한다. c) 모델 업데이트 계산 (gradation)

2.이 복사 gradation 의 평균을 계산한다.

3. 모델 업데이트

4. 1a 단계를 반복한다.

1.2.2. Hovorod

 ring-all reduce 접근 방식이 유용성과 성능을 향상 시킨다는 것을 인식하고 Uber 개발자는 UberTensorFlow의 요구를 계속 연구하도록 동기를 부여한다. 그들은 Bais TensorFlow ring-all reduce 알고리즘을 구현하고 이를 기반으로 구축했다. 과정은 다음과 같다.

1. 코드를 독립형 Python 패키지 Horovod로 변환한다. 이름은 전통적인 러시아 민속 무용에서 유래 한 것으로 댄서들이 춤을 추고 춤을 추며 분산 TensorFlow 프로세스는 Horovod를 사용하여 서로 통신한다. Uber 팀마다 다른 버전의 TensorFlow를 사용할 수 있다. 그들은 모든 팀이 최신 버전의 TensorFlow로 업데이트하지 않고도 ring-all reduce 알고리즘을 사용하고 패치를 사용하며 프레임 워크를 구축하기를 원한다. 별도의 Python 패키지가 있으면 하드웨어 조건에 따라 Horovod 설치 시간이 1 시간에서 몇 분으로 줄어 든다.

  2. Baidu의 ring-all reduce 구현을 NCCL로 교체한다. NCCL은 최적화 된 버전의 ring-all reduce를 제공하는 NVIDIA의 collection 통신 라이브러이다. NCCL 2를 사용하면 여러 시스템간에 링 올 리 duct를 실행할 수 있으므로 여러 가지 성능 향상을 활용할 수 있다.

  3. 지원 모델은 단일 서버와 여러 GPU에 적용되며 원본 버전은 단일 GPU 모델 만 지원한다.

4. 마지막으로 API는 많은 초기 사용자의 피드백을 기반으로 여러 가지 방식으로 개선되었다. 특히, 브로드 캐스트 작업은 모델이 모든 작업자 스레드에서 일관된 초기화를 수행 할 수 있도록 구현된다. 새로운 API를 사용하면 단일 GPU 프로젝트에서 사용자의 계산 노력을 4로 줄일 수 있다.

**2. 선행연구 및 기술현황**

2.1 연구와 기술현황

  현재Horovod의 응용이 uber이라면 Tensorflow을 기반으로 관련된 연구와 기술현황은 Nvidia의 vDNN[3]에서 사용한 메모리확장법을최근 가장 널리 사용되고 있는 Deep learning Framework인 Tensorflow를 통해 구현하여 그 성능을 확인하는 것이 있다. 그리고 결과 20% 성능이 향상되었다는 것이다.

  또한 TensorFlow딥 러닝 영상처리를 기반으로 지능형 실종자 색인 시스템 구현[4]하는데도 도움을 주고 있다. 이 연구는 영상처리를 위해TensorFlow deep running 도구에서 합성곱 신경망(Convolutional Neural Network, CNN) 알고리즘을 사용하였고 최소한의 전처리를 사용하며, 병렬처리를 이용하여 상대적으로 빠른 처리속도의 프로세싱 능력을 가진 다 계층의 알고리즘이라고 한다. 결과적으로 OpenCV와 TensorFlow를 이용하여 가공한 정형화된 데이터를 색인에 포함 함으로서 사용자에게 요구되는 시간비용을 줄일 수 있어 장기 실종사건으로 발전될 수 있는 사건을 미연에 방지할 수 있는 기대효과를 지니고 있다.

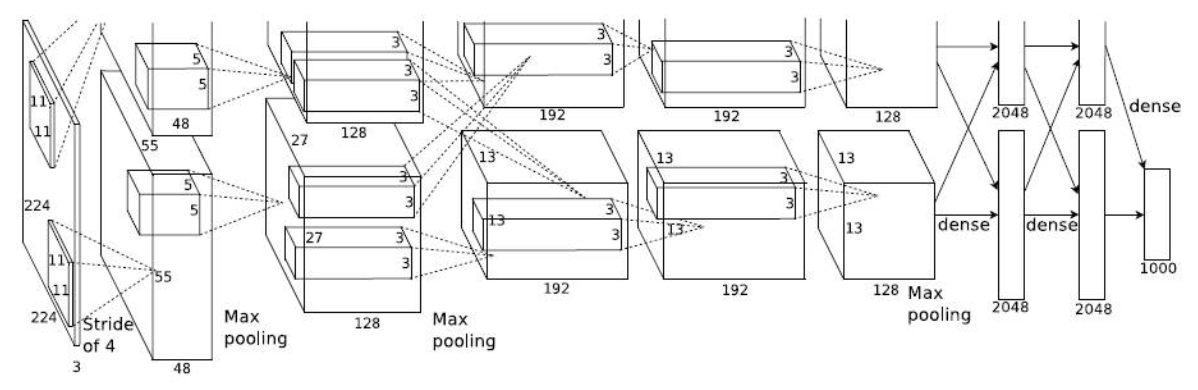
2.2 제안목표와 방향

TensorFlow와 Horovod의 트래픽이 다르다는 것은 이미 알려져 있는 것 이다. 하지만 얼마나 다르고 어떻게 다른지는 아직 답을 얻을 수 가 없다. 트래픽이 다르고 데이터 사이즈가 다르기 때문이다. 어떤 모델에는 TensorFlow의 성능이 뛰어나고 Horovod모델의 성능이 떨어질 수 있고 또 다른 모델에는 반대로 될 수가 있다. 그러므로 이것을 테스트하고 분석을 하는 것이 본 제안서의 목표와 방향이다.

**3. 작품/논문 전체 진행계획 및 구성**

3. 1 분석 및 실행

아래와 같이 모델에는 서로 다른 모델이 존재한다. 모델 구조는 아래 그림과 같다. 입력 레이어를 계산하지 않는 8 개의 레이어 만 보지 말고 총 매개 변수의 양은 60M 이상이다. 사실, 매개 변수의 수량은 이후 네트워크보다 크다.



<그래프 2><http://www.sohu.com/a/134347664_642762>

이 레이어에서 계산 된 기능 맵이 분리되어 있지만 이전 레이어에서 사용 된 데이터는 연결의 점선에 따라 다르다. 입력 레이어가 분리 된 후 첫 번째 레이어와 두 번째 레이어 사이의 점선 즉, 두 번째 레이어의 128map은 상위 48map에 의해 계산되며, 마찬가지로, 세 번째 레이어 앞의 점선은 완전히 교차된다. 즉, 각 192map은 이전 128 + 128 = 256map으로 계산된다는 것이다.

모델은 올바른 형식의 논리 흐름이며 데이터를 붙여 넣으면 계산 결과가 나타납니다. 처리되는 데이터에 따라 사용되는 모델이 달라질 수 있다. Keras에서 사용 가능한 두 가지 주요 유형의 모델 인 [Sequential model](https://keras.io/models/sequential) 과 [Functional API와 함께 사용되는 Model 클래스가 있다](https://keras.io/models/model) .이 모델에는 여러 가지 방법과 속성이 공통적으로 있는 것 을 확인할 수 있다.

model.layers 도 1은 모델을 포함하는 층들의 평탄화 된 리스트이다.

model.inputs 모델의 입력 Tensor 목록이다.

model.outputs 모델의 출력 Tensor 목록이다.

model.summary()모델의 요약 표현을 인쇄다.

model.get\_config()모델의 구성을 포함하는 사전을 반환한다. 다음을 통해 구성에서 모델을 다시 인스턴스화 할 수 있는 것 을 확인할 수 있다.

config = model.get\_config()

model = Model.from\_config(config)*# or, for Sequential:*

model = Sequential.from\_config(config)

model.get\_weights() Numpy 배열로 모델의 모든 가중치 Tensor 목록을 리턴한다.

model.set\_weights(weights)Numpy 배열 목록에서 모델 가중치의 값을 설정한다. 목록의 배열은에서 반환 된 것과 같은 모양이어야한다 get\_weights().

model.to\_json()모델의 표현을 JSON 문자열로 반환한다. 표현에는 가중치가 포함되지 않으며 아키텍처 만 포함된다. 다음을 통해 JSON 문자열에서 동일한 모델 (다시 초기화 된 가중치)을 다시 인스턴스화 할 수 있다.

하지만 성능 문제로 위에서 언급 된 모델 및 데이터 세트 중에서 더 작은 크기로 테스트을 할 것이다. 만약 시간 등 조건이 된다면 보다 더 큰 사이즈로 테스트를 해 볼 것이다.  테스트한 결과는 Wireshark으로 패킷을 캡쳐하고 트래픽을 분석할 것이다. Wireshark[5] 는 세계에서 가장 널리 사용되는 네트워크 프로토콜 분석기이다. 또한 미세한 수준에서 네트워크에서 발생하는 상황을 확인할 수 있으며 많은 상업 및 비영리 기업, 정부 기관 및 교육 기관에서 사실상의 표준이다. Wireshark 개발은 전 세계 네트워킹 전문가의 자원 봉사 덕분에 성공했으며 1998 년 Gerald Combs가 시작한 프로젝트의 연속이다.

 위에서 머신 러닝의 성능과 트래픽 패턴을 함께 비교 분석한 뒤 데이터셋과 모델의 특성에 따라 성능이 저하된 원인을 파악한다. 그러므로 특성에 맞게 필요한 모델을 사용해서 원하는 결과를 얻을 수 있다. 이런 결과를 얻기 위해서는 분산 TensorFlow와 Horovod의 트래픽 특성을 이해하는 것 이 제일 중요하다.

**4. 기대효과 및 개선방향**

 분산 교육의 매개 변수 서버는 매개 변수 서버와 작업자 스레드의 비율에 따라 구성 할 수 있으며 각 매개 변수 서버는 다른 구성 데이터를 갖는다.

   이 방법으로 성능을 향상시킬 수 있지만 여전히 두 가지 주요 과제에 직면한다.

• 작업자 대 매개 변수 서버의 올바른 비율 결정 : 매개 변수 서버를 사용하면 네트워크 또는 컴퓨팅 병목 현상이 발생할 수 있다. 여러 개의 매개 변수 서버가 사용되는 경우 통신 모드가 "All-to-All"로 변경되어 네트워크 상호 연결이 포화 될 수 있다.

• TensorFlow 프로그램의 복잡성 증가 : 테스트에서 분산 TensorFlow를 사용하는 각 사례마다 초기 작업자 스레드와 매개 변수 서버를 지정해야 하며 모든 작업자 스레드의 호스트 및 포트와 같은 서비스 검색 정보를 전달한다. 매개 변수 서버. 그리고 tf.Server ()는 트레이닝 프로그램을 조정하기 위해 적절한 tf.ClusterSpec ()으로 빌드된다. 또한 사용자는 모든 작업이 tf.train.device\_replica\_setter ()를 올바르게 사용하고 타워를 사용하여 코드가 서버의 다중 GPU 설정을 준수하도록해야 한다. 이것은 일반적으로 가파른 학습 곡선과 많은 코드 리팩토링을 초래하여 실제 모델링 시간을 압축한다.

  링-올 리 듀스 알고리즘에서, 각각의 N 노드는 다른 2 개의 노드와 2 \* (N-1) 번 통신한다. 이 통신 프로세스에서, 노드는 데이터 버퍼로부터 블록을 송수신한다. 첫 번째 N-1 반복에서는 수신 된 값이 노드 버퍼의 값에 추가된다. 두 번째 N-1 반복에서 수신 된 값은 노드 버퍼에 보유 된 값을 대체한다.또한 Baidu의 기사는 이 알고리즘이 대역폭에 최적화되어 있음을 증명한다. 즉, 버퍼가 충분히 크면 사용 가능한 네트워크를 최대한 활용할 수 있다.

 Horovod는 새로운 방식을 만든다. 노드 간 작업 타임 라인에 대한 높은 수준의 이해를 제공하기 위해 Horovod 타임 라인이 구축되었다. 사용자는 Horovod 타임 라인을 사용하여 교육 프로세스의 각 단계에서 각 노드의 상태를 명확하게 볼 수 있다. 이는 버그를 식별하고 성능 문제를 해결하는 데 도움이 된다. 사용자는 단일 환경 변수를 설정하여 타임 라인을 활성화하고 chrome : // tracing을 통해 브라우저에서 분석 결과를 볼 수 있다.

 그러므로 Horovod, Tensor Fusion 및 기타 방법을 사용하여 Michelangelo 플랫폼에 구축 된 기타 기능은 머신 러닝 시스템에서 모델의 효율성, 속도 및 사용 편의성을 향상시킬 수 있다.

**5. 기타**

5. 1 팀원간의 역할분담

강경룡 TensorFlow 트래픽 패턴 구현 및 테스트

이상우 Horovod 트래픽 패턴 구현 및 테스트

5. 2 비용 분석

실제환경 및 가상환경

Linux node 2대

GPU 2개

**6. 참고문헌**

**1. “**[**Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour**](https://link.zhihu.com/?target=https%3A//research.fb.com/wp-content/uploads/2017/06/imagenet1kin1h5.pdf)**”,Uber，2017**

**2.  “**[**Bringing HPC Techniques to Deep Learning**](https://link.zhihu.com/?target=http%3A//research.baidu.com/bringing-hpc-techniques-deep-learning/)**”,baidu，2017**

[**https://eng.uber.com/horovod/**](https://eng.uber.com/horovod/)

[**https://github.com/horovod/horovod**](https://github.com/horovod/horovod)

3. [**김영랑**](http://www.riss.kr.sa.skku.edu:8080/search/Search.do?detailSearch=true&searchGubun=true&queryText=znCreator,%EA%B9%80%EC%98%81%EB%9E%91%28Youngrang+Kim%29&colName=re_a_kor) 등, [**Tensorflow 기반 GPU 메모리 효율적인 활용 방안 연구,**](http://www.riss.kr.sa.skku.edu:8080/search/detail/DetailView.do?p_mat_type=1a0202e37d52c72d&control_no=3393ee3dd704d1f66aae8a972f9116fb) [**한국정보과학회 학술발표논문집**](http://www.riss.kr.sa.skku.edu:8080/search/detail/DetailView.do?p_mat_type=3a11008f85f7c51d&control_no=95949233a7b69e78)**,2018**

**4. 백영태 등,OpenCV 영상처리와 TensorFlow 딥 러닝 영상처리 기반의 지능형 실종자 색인 시스템 구현,**[**韓國컴퓨터情報學會論文誌**](http://www.riss.kr.sa.skku.edu:8080/search/detail/DetailView.do?p_mat_type=3a11008f85f7c51d&control_no=9084bd3d9d22959effe0bdc3ef48d419)**,2017**

**5.** <https://www.wireshark.org/>