ETRI

2016
Electronics and
Telecommunications
Trends

딥러닝 분산처리 기술동향

Trends on Distributed Frameworks for Deep Learning

안신영 (S.Y. Ahn) 고성능컴퓨팅시스템연구실 선임연구원
박유미 (Y.M. Park) 고성능컴퓨팅시스템연구실 책임연구원
임은지 (E.J. Lim) 고성능컴퓨팅시스템연구실 선임연구원
최완 (W. Choi) 고성능컴퓨팅시스템연구실 책임연구원

* 본 연구는 2016년도 미래창조과학부의 정보통신·방송 기술개발사업의 일환으로 수행되었음 (대규모 딥러닝 고속 처리를 위한 HPC 시스템 개발).

최근 알파고를 통해 인공지능 기술이 전 세계인의 이목을 집중시켰던 반면, 인공지능 연구자들은 인공지능 부활에 결정적 역할을 한 딥러닝 기술에 주목하고 있다. 딥러닝은 다계층 인공신경망 기반의 기계학습 기술로서 최근 컴퓨터 비전, 음성인식, 자연어 처리 분야에서 인식 성능을 높이는 데 중요한 역할을 하고 있다. 딥러닝 기술을 이용하여 기계가 수천만장의 이미지를 학습하여 객체를 인식하게 하고, 수천 시간의 음성 데이터를 학습하여 사람의 말을 알아듣게 처리하는 데에는 다수의 고성능 컴퓨터가 필요하다. 따라서 딥러닝에는 다수의 컴퓨터를 효율적으로 이용하기 위한 분산처리 기술이 필수적이며 관련 연구들이 활발히 진행되고 있다. 이에 본고는 다중 컴퓨터 노드들에서 딥러닝 모델을 분산처리할 수 있는 기존의 프레임워크들을 비교 분석하고 딥러닝 분산처리 기술에 대한 발전 방향을 전망한다.

Ⅰ. 머리말

최근 알파고로 인해 인공지능 기술에 전 세계의 이목이 집중되었다. 알파고는 Convolutional Neural Net (CNN)이라는 딥러닝 기법과 몬테카를로 탐색 기법을 이용한 바둑게임 프로그램으로 프로기사 이세돌 9단과의 대국 시 1,920개의 CPU와 280개의 General-Purpose computing on Graphics Processing Units (GPGPU)를 이용한 분산 시스템에서 실행되었다[1]. 알파고의 승리는 뛰어난 인공지능 알고리즘뿐만 아니라 빠른 시간 내에 계산을 마쳐 정해진 시간 내에 응수할 수 있게 했던 천 대 가까운 컴퓨터 덕분이기도 하다. 그러나 알파고를 개발한 구글 딥마인드의 데미스 하사비스는 컴퓨터 HW의 용량을 개선하면 오히려 알파고의 성능이 떨어져 HW 용량을 무조건 늘리지는 않았다고 한다[2]. 이는 컴퓨팅 자원이 무조건 많다고 일을 빠르게 잘하는 것이 아니라 응용의 요구에 맞추어 얼마나 효율적으로 자원을 이용하느냐에 달렸다는 것을 의미한다.

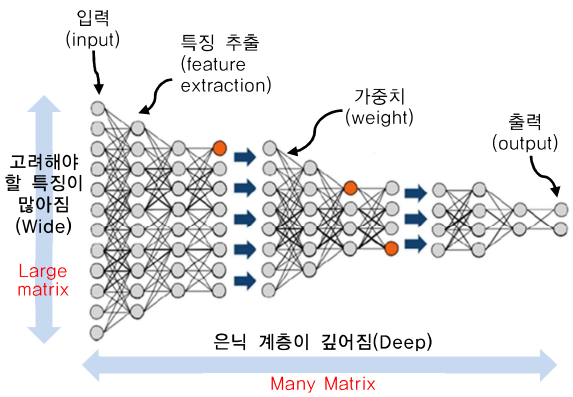
수많은 컴퓨터를 이용해야 하는 딥러닝 분산처리의 요구사항을 도출하기 위해 딥러닝 기술을 분석하면 크게 트레이닝(training)과 인퍼런스(inference)로 나뉜다. 트레이닝은 입력 데이타를 통해 모델을 학습하는 과정이고 인퍼런스는 학습된 모델로 인식 등의 서비스를 수행하는 과정이다. 이 중 딥러닝 트레이닝은 반복이 많은 계산 집중형 프로세스로서 처리 시간이 오래 걸리는 문제로 인해 분산 병렬처리 기술의 연구가 활발히 진행되고 있다. 이에 본고에서는 딥러닝 트레이닝의 관점에서 분산처리 기술동향을 고찰한다. Ⅱ장에서는 딥러닝 분산 처리 기술이 왜 필요한지, 그리고 어떤 기술적 이슈들이 있는지 설명하고, Ⅲ장에서 Ⅴ장까지는 사례들을 소개하고, Ⅵ장에서 이를 비교 분석한다. 마지막으로 Ⅶ장에서 결론을 맺는다.

Ⅱ. 딥러닝 분산처리 기술 개요

1. 필요성

딥러닝이란 사람의 신경세포(Biological Neuron)를 모사하여 기계가 학습하도록 하는 인공신경망(Artificial Neural Network) 기반의 기계 학습법으로서 최근 이미지 인식, 음성인식, 자연어 처리의 발전에 기여하며 주목받고 있다. 최근 딥러닝 모델들은 응용의 인식 성능을 높이기 위해 (그림 1)과 같이 모델의 계층이 깊어지고(Deep), 특징(Feature)이 많아지는(Wide) 대규모 모델로 진화하고 있다.

이에 인공지능 선두 기업들을 중심으로 대규모 딥러닝 모델을 개발할 수 있는 고성능 컴퓨팅 인프라를 앞다투어 구축하고 있지만 여전이 딥러닝 처리, 특히 트레이닝에 많은 시간이 걸리고 있다. Microsoft에서는 2012년 음성인식용 모델(Context-Dependent DNN-HMM) 트레이닝 시 Fisher 데이터셋으로 트레이닝하는데 4 GPGPU를 장착한 1대의 컴퓨터로 17.8일이 소요되었고[3], 같은 해 Google은 1천만건의 동영상을 CNN으로 모델링하여 트레이닝 하는데 GPGPU없이 16,000 CPU 코어를 장착한 1,000대의 컴퓨터로 일주일이 걸렸다[4]. GPGPU의 이용이 점차 확산되며 2013년 스탠포드 대학교에서는 이미지 1천만건을 CNN 모델로 트레이닝하는 데에 16대의 컴퓨터에서 98,304 GPGPU 코어로



(그림 1) 대규모 딥러닝 모델의 특징

3일이 소요되었다[5]. 바이두에서는 2015년 이미지 인식용 슈퍼컴 Minwa를 개발하여 32대 컴퓨터에 92,160 GPGPU 코어로 512×512 픽셀 이미지를 CNN 모델로 트레이닝하며 정확성 80%에 도달하는 데 8.6시간이 소요되었다[6].

이처럼 딥러닝 응용, 즉 이미지 인식 또는 음성인식의 정확성을 높이기 위해 딥러닝 모델의 규모가 커지고 입력 데이터의 양이 많아질수록 수많은 컴퓨터가 필요하고 그에 따른 효율적인 분산처리가 필수적이다.

현재까지 공표된 딥러닝 분산처리 지원 기술들은 크게 세 부류로 구분된다. 첫째, 빅데이터 분산처리 프레임워크를 딥러닝 분산처리에 확장하는 기술, 둘째, 기존의 기계학습 프레임워크를 확장하는 기술, 마지막으로 딥러닝용 분산처리 프레임워크를 새롭게 개발하는 기술들이다. III장부터 차례로 세 부류에 해당하는 분산처리 기술들을 소개한다.

2. 기술적 이슈

딥러닝 트레이닝은 기본적으로 피드 포워드(feed forward) 과정과 백 프로파게이션(back propagation) 과정의 반복이다. 즉, 딥러닝 트레이닝은 입력 계층부터 여러 은닉 계층을 거쳐 출력 계층까지 특징값과 목적함수를 계산해 나가는 피드 포워드 과정과 오류(피드 포워드의 결과와 정답 간의 차이)를 반영하여 출력 계층으로부터 은닉 계층을 거쳐 입력 계층까지 가중치를 수정하는 백 프로파게이션 과정의 반복이다. 트레이닝 과정에서 수정되는 가중치는 오류가 최소화될 때까지 반복적으로 갱신되는데, 분산처리 시에는 모든 컴퓨터가 가중치와 특징값(딥러닝 파라미터라고 통칭함)들을 공유해야 한다. 여기에서 분산된 데이터와 모델의 병렬처리 방법, 파라미터 공유 기법과 통신 프로토콜, 동기화 문제들이 파생되며 본 절에서는 이와 관련된 기술적 이슈를 간략히 설명한다.

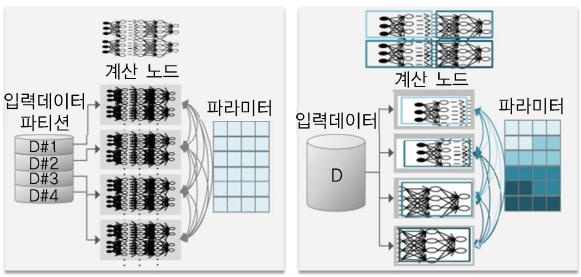
가. 딥러닝 트레이닝 분산 병렬처리

딥러닝 분산 트레이닝 시 분산 병렬처리를 통하여 트레이닝 가속을 시도하는데 이의 방법으로 데이터 병렬처리(Data Parallelism)와 모델 병렬처리(Model Parallelism) 방법이 있다. 데이터 병렬처리란 학습해야 하는 입력 데이터 셋을 다수의 컴퓨터가 나누어 트레이닝하는 방법이고, 모델 병렬처리란 딥러닝 모델을 나누어 다수의 컴퓨터들이 트레이닝하는 방법이다.

(그림 2a)는 데이터 병렬처리 방법으로서 분산 컴퓨터마다 모델 전체가 로드되고 입력 데이터를 부분으로 나누어 트레이닝하며, 과정이 반복될 때마다 각 컴퓨터에서 수정된 로컬 가중치를 다른 분산된 컴퓨터와 교환하게 된다.

한편 모델이 한 컴퓨터에 로드되어 처리될 수 없을 정도로 방대한 경우 (그림 2b)와 같이 다수의 컴퓨터에서 모델을 나누어서 처리해야 하는데, 이때 각 컴퓨터에서는 모든 입력데이터에 대해 트레이닝을 수행하며 부분적으로 계산된 로컬 가중치를 다른 분산된 컴퓨터들과 서로 교환하게 된다.

대규모 딥러닝 모델을 분산처리하는 대부분의 기술에서 데이터 병렬처리와 모델 병렬처리 방법을 널리 이용하고 있으며, 경우에 따라 함께 적용하기도 한다. 이의 사례는 III~VI장에서 설명한다.

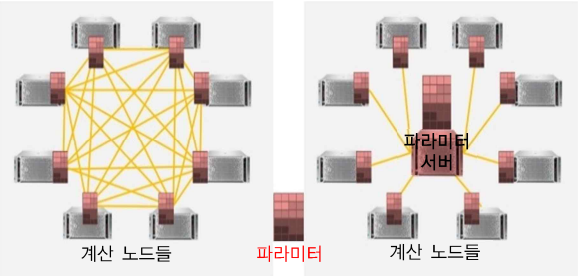


(a) 데이터 병렬처리 (b) 모델 병렬처리

(그림 2) 데이터 병렬처리와 모델 병렬처리 비교

나. 파라미터 공유 방법

가 항에서 기술한 바와 같이, 딥러닝 트레이닝 분산



(a) 풀 메시 토폴로지 기반 공유

(b) 스타 토폴로지 기반 공유

(그림 3) 파라미터 공유 기법

병렬처리 시에는 로컬 가중치와 특징값 등의 파라미터들을 모든 컴퓨터가 공유해야 한다. 파라미터를 공유하는 방법에는 각 컴퓨터가 다른 모든 컴퓨터에 직접 파라미터를 전달하는 풀 메시(full mesh) 토폴로지 기반 공유 방법(그림 3a)와 모든 분산 컴퓨터들이 공유 장소를 이용하여 가중치를 읽고 쓰는 스타(star) 토폴로지 기반의 공유 방법이다[(그림 3b) 참조]. 후자의 경우 파라미터 공유 저장소에 대한 동시성 제어, 동기화가 필요하나 전자의 경우는 공유 저장소 관리가 필요 없는 장점이 있다. 그러나, 전자는 분산 컴퓨터의 수가 늘어날수록 통신 횟수가 N^2으로 증가하여 확장성이 떨어지는 반면 후자는 컴퓨터 수가 늘어나도 통신 횟수가 크게 증가하지 않으므로 확장성에서 유리하다.

다. 동기식과 비동기식 처리

나 항에서 기술한 스타 토폴로지 기반 파라미터 공유 방법에서는 분산된 컴퓨터들이 각각 파라미터를 중앙 집중형으로 업데이트하므로 때문에 오류에 따라 가중치를 갱신해야 하는 트레이닝 반복(일반적으로 1 epoch이라 부름)마다 컴퓨터 간 파라미터 동기화가 필요하다. 그러나 동기식 업데이트의 경우, 딥러닝을 분산처리하는 컴퓨터들의 HW 상태, 현재 워크로드에 따라 서로 실행시간이 다르므로 가장 늦게 파라미터를 서버로 전송하는 컴퓨터의 성능에 전체 트레이닝 성능이 맞춰지게 된다. 이러한 문제를 해결하기 위해 비동기식 업데이트 방식으로 처리하기도 한다. 즉, 파라미터 서버가 컴퓨터들로부터 늦거나 빨리 도착하는 파라미터들의 동기를 맞추지 않고 트레이닝을 진행하는 방법이다. 동기식에 비해 정확성을 크게 희생시키지 않으면서 빠르게 트레이닝할 수 있는 장점이 있다. 대부분의 분산 프레임워크들에서는 동기식 방법을 기본으로 하고, 일부 프레임워크들에서 트레이닝 속도개선을 위해 비동기식 방법을 추가로 제공하고 있다.

Ⅲ. 빅데이터 처리 계열의 기술 동향

1. Apache SPARK

현재 빅데이터 처리를 위해 가장 많이 이용되는 플랫폼인 Hadoop MapReduce는 반복이 많은 분석 작업을 할 때 디스크를 통한 데이터 공유로부터 발생하는 성능 저하 단점이 있었으며, 이를 극복하기 위해 메모리를 읽기 전용 방식으로 이용하는 인메모리 기반의 빅데이터 처리 플랫폼으로 Spark이 출현하게 되었다.

Spark은 여러 분산 컴퓨터의 메모리에 저장되는 변경이 불가능한 데이터 집합인 Resilient Distributed Datasets(RDD)와 이 RDD를 이용하기 위한 인터페이스(RDD Operations)를 제공한다. RDD는 파일을 읽어서 생성되거나 기존 RDD를 RDD Operation으로 변환할 때 생성된다. 장애 발생 시 RDD가 어떻게 생성되는지 그 순서만 알고 있으면 해당 데이터의 복구가 가능하므로 높은 내고장성을 제공한다. Spark은 Hadoop, Mesos, EC2 등 클라우드 플랫폼에서 실행가능하며, HDFS, Cassandra, HBase, S3 등 분산 클라우드 파일 시스템들과 연동하여 데이터 입출력을 지원한다.

Spark은 플랫폼안에 스트리밍, Structured Query Language(SQL), 머신러닝 등 다양한 형태로 데이타를 처리할 수 있는 장점이 있다. 딥러닝 플랫폼 분야에 있어 Spark은 기 구축된 클라우드를 이용할 수 있다는 점과 Scale-up 방식인 GPGPU를 이용한 방법의 한계인

GPGPU 메모리의 한계 때문에 많은 관심을 받고 있으나, 지원하는 딥러닝 라이브러리가 미흡하고(Spark MLLib은 Regression, Classification, Clustering, L-BFGS 등 기계학습 라이브러리 위주 지원), 배치작업에 최적화된 분산처리 플랫폼으로서 복잡한 알고리즘을 수행하는 분산된 프로세스 간에 상태정보를 서로 교환하고 데이터를 빈번하게 공유해야 하는 병렬처리에서는 한계가 있다.

2. H2O 및 Sparkling Water

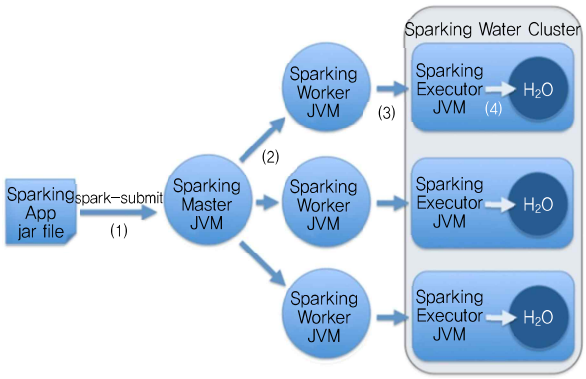
가. H2O

H2O는 H2O.ai라는 벤처기업에서 시작된 Java 기반 오픈 소스 인메모리 기계학습 플랫폼으로 Spark의 MLLib에서 제공하지 못하는 다양한 기계학습/딥러닝 알고리즘을 지원할 뿐 아니라, 웹 인터페이스(REST API)를 제공하며, R, Java, Python, Scala와 같은 하이레벨 언어들과 쉽게 결합된다.

H2O 클라우드는 하나 이상의 노드(자바 가상 머신, JVM)들로 구성되며, 각 노드는 언어 레이어, 알고리즘 레이어, 코어 레이어로 구성된다. 언어 레이어는 R, Scala 등 언어를 이용한 H2O 모델 개발을 지원하며, 알고리즘 레이어는 데이터 파싱과 다양한 기계학습 알고리즘 및 예측 엔진을 제공한다. H2O의 코어 계층에서는 인메모리 처리의 실시간 응답성과 작은 단위의 데이터 분산을 지원하는 분산 키/밸류 저장소 및 컬럼-압축 저장을 제공하는 Fluid vector frame을 메모리 관리 기능으로 제공하고, 작업관리, MapReduce Task 관리, Fork/Join 관리를 CPU 관리 기능으로 제공한다[8].

나. Sparkling Water

Spark의 MLlib과 H2O의 ML 라이브러리를 모두 이용하기 위해 H2O와 Spark을 통합하여 기계학습 및 딥러닝을 지원하는 플랫폼이 Sparkling Water이다.



(그림 4) Sparkling Water 응용 라이프사이클[9]

(그림 4)를 보면 H2O와 Spark 모두 Java로 구현되어 있으므로 ① jar 파일 형태의 Sparkling 응용이 Spark 마스터 노드에 제출되면 ② 이 응용은 각 Spark 워커 노드로 분배되고, ③ 각 워커는 Executor JVM을 실행하여 ④ H2O 응용을 실행한다.

Sparkling Water에서 Spark은 HDFS와 같은 데이터 저장소로부터 데이터를 읽어 RDD를 구성하고, 이 RDD는 H2O 인스턴스 내에서 H2ORDD로 변환되고, H2O가 H2ORDD로 딥러닝 트레이닝을 수행한 후 다시 Spark RDD로 변환되어 다음 단계의 입력 데이터로 사용된다.

3. DeepSPARK과 SparkNet

서울대에서는 Spark와 멀티 GPGPU를 지원하는 단일 컴퓨터 딥러닝 트레이닝 엔진(Caffe)의 연동을 통한 분산 딥러닝 통합 프레임워크인 DeepSpark를 개발 중이다. DeepSpark은 많은 저변에도 불구하고, 단일 컴퓨터용이라는 Caffe의 단점인 대규모 딥러닝 모델 트레이닝에 대한 확장성 부족 문제에 대한 대안으로 볼 수 있다.

DeepSpark은 기존 Caffe 프레임워크를 이용할 수 있도록 새로운 Caffe Wrapper를 개발하였으며, 데이터 병렬성을 제공하기 위해 비동기 업데이트를 수행하는 파라미터 서버 개념의 Parameter Exchanger를 Spark 마

스터 노드에 개발하였다. DeepSpark은 Spark을 이용하여 Caffe가 실행되는 노드로 워크로드와 파라미터를 자동 분배하고, 비동기 업데이트 기법을 개발하여 Spark의 동기 프로세싱 능력을 보완한다[10].

UC Berkeley의 SparkNet 역시 Spark와 Caffe를 연동하여 분산 딥러닝 트레이닝이 가능하도록 하는 통합 프레임워크로서 Spark RDD로부터 데이터를 읽어오는 인터페이스, Caffe로의 Scala 인터페이스, Caffe로부터 메모리 복사 없이 데이터와 모델 계산을 용이하게 해주는 라이브러리를 포함하고 있다. 딥러닝 모델 분산 트레이닝은 하나의 Spark 마스터노드에서 다수의 워커노드로 모델 파라미터를 전송하고, 각 워커노드는 일정 횟수의 트레이닝 반복(또는 일정 시간) 후에 학습된 모델 파라미터를 다시 마스터 노드로 전송하면 마스터 노드에서 평균값으로 새로운 모델 파라미터를 생성하는 방식(동기식)으로 운영된다[11].

DeepSpark과 SparkNet 모두 빅데이터 수집, 가공 처리에 우수한 Spark와 Caffe의 많은 딥러닝 라이브러리를 이용할 수 있다는 장점이 있으나, JNA(Java Native Access)의 성능 제약으로 인해 Spark와 Caffe의 연동 시 전용 딥러닝 프레임워크 대비 성능 최적화에 제약이 있다[12].

4. REEF

Retainable Evaluator Execution Framework(REEF)는 Microsoft에 의해 개발되어 Apache 오픈소스로 공개된 빅데이터 분석 플랫폼이며, 2016년 3월 현재 0.13.0 버전이 공개되어 있다[13][14].

REEF는 SQL, 그래프 처리, 기계학습 등의 다양한 데이터 분석 애플리케이션을 효과적으로 개발, 수행할 수 있는 통합 환경을 제공하는 응용 프레임워크이다. REEF는 자원관리자가 제공하지 않는 런타임 관리 기능들, 즉, 작업 모니터링과 재시작, 데이터 이동 및 통신, 그리고 분산 상태 관리와 같은 기능을 지원하고, 특정한 프로그래밍 모델을 제공하는 것이 아니라 새로운 응용을 빠르게 개발하고 실행할 수 있는 환경을 제공한다[15].

내고장성 획득을 위해 태스크의 상태를 디스크에 체크포인팅 할 수 있고, 태스크를 다른 머신으로 이동하는 기능을 제공하여 동적 부하 분산을 가능하게 한다. 이러한 기능을 통해 장기 실행 작업이나 큰 단위 작업에 유리한 실행 환경을 제공하여 고성능 그래프 처리나 기계학습 알고리즘을 효과적으로 지원한다.

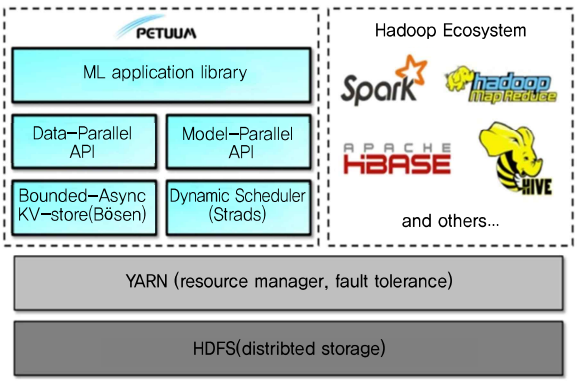
자원 관리자로는 Apache YARN 또는 Apache Mesos, Google Omega, Facebook Corona를 사용 할 수 있지만 현재 0.13.0 버전에서는 YARN만을 지원하고 있다.

Ⅳ. 기계학습 분산처리 기술동향

1. Petuum

Petuum은 카네기멜론 인텔랩에서 개발한 오픈 소스 분산 기계학습 프레임워크이다. (그림 5)와 같이 Petuum은 크게 Bösen이라 불리는 비동기 분산 키-밸류 저장소, Strads라 명명된 모델 병렬화 스케줄러와 다양한 기계학습 라이브러리들로 구성된다[16][17].

Bösen은 데이터 병렬화를 위한 파라미터 서버 목적으로 개발된 분산 키-밸류 저장소로 Microsoft Distributed Machine Learning Toolkit(DMTK)에서도 사용되고 있



(그림 5) Petuum 아키텍처와 에코시스템[17]

다. Bösen은 제한적인 동기화 방법인 Stale Synchronous Parallel(SSP) 모델을 이용하여 데이터 병렬화 기법 중에서 비동기 방식에 근접한 성능을 보여준다. Strads는 성능에 영향을 주는 안전하지 않은 병렬 오퍼레이션을 피하면서 계산 작업의 우선순위를 매기고, 작은 단위로 파라미터 업데이트 동작의 스케줄링을 통해 모델 병렬화를 지원하는 동적 기계학습 업데이트 스케줄러이다.

Petuum은 DNN, CNN과 같은 딥러닝 라이브러리 외에 Latent Dirichlet Allocation(LDA), MedLDA, sparse coding, k-means 등과 같이 10개 이상의 기계학습 라이브러리를 지원한다. 또한, Petuum은 개별 클러스터 및 아마존 EC2 또는 구글 GCE와 같은 클라우드에서 분산 실행을 지원하며, 빅데이터 및 대규모 모델을 지원하는 기계학습을 실행하기 위한 분산 프로그래밍 도구를 제공한다[16].

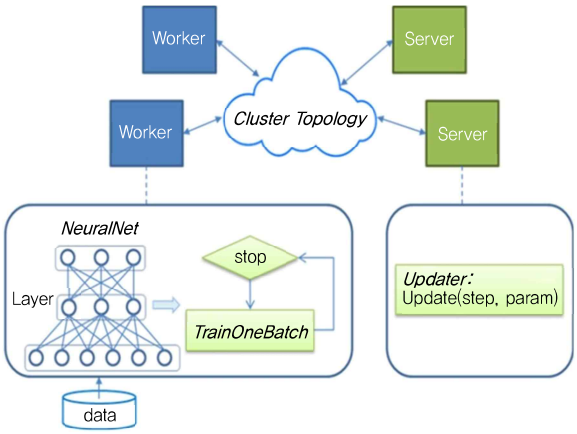
Petuum의 Bösen을 Caffe 프레임워크와 통합한 분산 GPGPU 딥러닝 프레임워크로 Poseidon이 있는데, Poseidon은 Caffe에 분산된 머신 간 통신을 위한 시스템 아키텍처를 제공한다[18].

V. 딥러닝 전용 분산처리 기술동향

1. SINGA

싱가폴 대학이 개발하고 Apache 오픈소스로 공개된 SINGA는 분산 딥러닝 플랫폼의 이용성(Usability)과 확장성(Scalability) 향상을 목표로 개발된 분산 딥러닝 플랫폼이다. 이용성은 딥러닝 모델의 구조 및 훈련 알고리즘의 추상화를 통해서 달성하고, 확장성은 워커와 워커 그룹을 이용한 하이브리드 아키텍처 디자인을 통해 달성한다[(그림 6) 참조]. 워커 그룹들 간에는 비동기적으로 동작하고, 워커 그룹 내의 워커들은 동기적으로 실행된다. 또한, 서버를 두어 워커 간의 파라미터 공유가 가능하도록 하였다.

딥러닝 모델(NeuralNet)을 분할하는 방법으로 레이어



(그림 6) SINGA 딥러닝 플랫폼 아키텍처[19]

간 파티션닝, 싱글 레이어 파티셔닝(batch dimension, feature dimension), 하이브리드 파티셔닝 방법을 지원하며, 설정된 클러스터 정보에 따라 워커들과 서버들을 분산된 클러스터에 배치하고, 분할된 학습 데이터와 모델을 분산배치하여 이 워커들을 실행한다.

SINGA는 장애복구를 위해서 체크포인트 및 복구 기능을 지원한다. 아파치 주키퍼를 이용하여 클러스터 설정을 간소화했으며, 자원 관리를 위해 Mesos에 통합될 수 있고 Docker Image 생성을 지원한다. SINGA는 Multi-Layer Perceptron(MLP), CNN, RBM, Recurrent Neural Networks(RNN), AlexNet, cuDNN과 같은 모델을 지원한다.

2. 삼성 Veles

Veles는 삼성 러시아 R&D 센터에서 개발되어 github에 공개된 딥러닝 응용 개발을 위한 분산 플랫폼으로서 모델 개발, 훈련, 서비스(응용)까지 통합 개발할 수 있는 플랫폼이다. znicz 라는 인공신경망 엔진을 이용하여 DNN, CNN, Autoencoder, RBM, RNN, Long Short Term Memory(LSTM) 등 인공신경망 학습을 지원하고 그 외 유전 알고리즘(Genetic Algorithm)을 지원한다. Veles에서는 딥러닝 모델 개발을 위한 단위 기능을 유닛(예, gradient descent)으로 제공하고 이들을 조합하여

워크플로우 형태로 모델을 개발할 수 있는 특징이 있다. 기본적으로 유닛 간의 제어 플로우 링크를 통해 데이터 처리를 실행하며 이러한 구조하에서 데이터와 모델 병렬처리를 지원한다.

분산처리를 지원하기 위해서는 마스터(Master) 프로세스와 슬레이브(Slave) 프로세스가 ZeroMQ라는 메시지큐 기반의 분산 오퍼레이션을 수행하며 최대 100대의 컴퓨터까지 지원한다. 분산처리 시 슬레이브가 죽어도 마스터에 영향을 끼치지 않고 동적으로 임의의 백엔드 서버를 추가할 수 있고 장애복구 후 다른 백엔드 서버로 다시 트레이닝을 시작할 수 있게 하는 고장 감내형 특성을 지닌다.

또한, 딥러닝 계산 가속장치를 사용하기 위해 OpenCL, CUDA, NumPy도 지원하는 등 다양한 오픈 소스들을 활용하여 개발된 플랫폼으로서 python, C/C++, Java-script, JAVA 등 다양한 언어의 소스로 구성되어 있다.

3. Microsoft CNTK

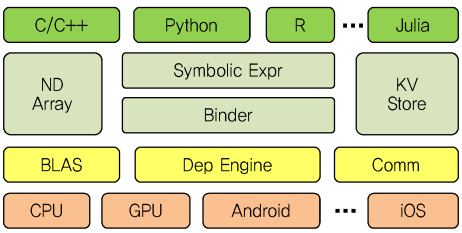
Computational Network Toolkit(CNTK)는 Microsoft Research에서 개발하여 2015년 4월에 처음 공개한 분산 딥러닝 플랫폼이다 Microsoft는 CNTK에 포함된 Deep Residual Network(ResNet)으로 컴퓨터 비전의 가장 큰 대회인 ImageNet Large Scale Visual Recognition Competition(ILSVRC)와 마이크로소프트 Common Objects in Context(COCO) 챌린지에서 2015년에 1위를 휩쓸었다. 번역 기술, 음성인식, 이미지 인식 등과 관련한 트레이닝을 할 때 이용되고, 실제로 Microsoft에서 코타나(Cortana) 혹은 Skype 번역 응용에 사용하고 있다. CNTK는 GPGPU 클러스터를 효과적으로 지원하고 Windows와 Linux 운영체제를 지원한다. 모듈화 기능을 제공하고 있어서 계산 네트워크, 실행 엔진, 학습 알고리즘, 모델 디스크립션, 데이터 입력기를 모두 분리할 수 있다. 또한, C++, Network Definition Language (NDL), Model Editing Language(MEL), Brain Script 등의 다양한 수단으로 모델을 쉽게 기술하고 수정할 수 있다. CNTK에서 제공하는 1-bit Stochastic Gradient Descent(SGD) 기술은 그래디언트 양자화 방식을 이용하여 통신 데이터의 양을 감소시키는 기술인데, 이를 통해 병렬 분산 트레이닝에서 통신 비용을 감소시켜서 여러 머신에 걸친 많은 GPGPU를 이용하는 고확장 병렬 트레이닝의 성능을 향상시킬 수 있다[21][22].

4. MXNet

MXNet은 CXXNet, Minerva, Purine2 개발자들이 연합하여, 과거의 경험을 반영하여 효율성과 유연성을 목표로 하여 설계한 딥러닝 프레임워크이며 2015년 github에 공개되었다. MXNet은 딥러닝을 포함한 기계학습 알고리즘을 쉽게 개발할 수 있도록 하는 다중 언어 기계학습 라이브러리로서 R, Python, Julia, C++를 지원하며, 모바일 장치로부터 GPGPU 클러스터까지 다양한 이종 시스템상에서 실행 가능하다[(그림 7) 참조].

MXNet은 프로그래밍 방식에서 선언적(Declarative) 프로그래밍과 지시적(Imperative) 프로그래밍 두 가지를 혼합하는 형태를 택하고 있다. 선언적 프로그래밍은 '무엇'을 명시하는 것이고, 지시적 프로그래밍은 '어떻게'를 명시하는 방식인데, 딥러닝에 있어서 선언적 프로그래밍은 계산 구조를 명시하는 데 유용하고, 지시적 프로그래밍은 파라미터 업데이트와 인터렉티브 디버깅에 유용하다. 이러한 점은 Theano와 Tensorflow도 유사하다.

프로그래밍 인터페이스로는 계산 그래프를 선언하기



(그림 7) MXNet 아키텍처[23]

위한 Symbol과 지시적 연산을 나타내는 NDArray, 그리고 여러 장치 사이에서 데이터를 동기화 하기 위한 파라미터 서버 기반의 분산 키-밸류 저장소인 KVStore를 지원한다. 그리고, Dependency engine은 계산 그래프와 지시적 연산에서 데이터 의존성을 추적하고 리소스의 효율과 병렬성을 높이도록 연산을 스케쥴링한다.

MXNet은 최근의 벤치마크에서 TensorFlow, Torch, Caffe와 같은 프레임워크와 성능이 유사하거나 더 뛰어난 것으로 나타났다[23].

Ⅵ. 분산처리 프레임워크 비교 분석

〈표 1〉은 상기 설명한 분산처리 프레임워크들을 다양한 관점에서 비교 분석한다.

개발언어 관점에서 보면 대체로 빅데이터 계열은 JAVA로 개발되었고, 기계학습/딥러닝 전용 프레임워크들은 C++로 개발하였다.

GPGPU 이용 관점에서 보면 SINGA, CNTK, MXNet은 프레임워크 자체적으로 GPGPU를 지원하며, 빅데이터 처리 계열에서는 DeepSpark과 SparkNet이 Caffe를 연동하여 하나의 분산 프레임워크로 확장하려고 시도하고 있고, 기계학습 플랫폼인 Petuum역시 Caffe와 연동하려고 노력하고 있다.

배치 형태의 작업처리 특성을 가진 빅데이터 처리 계열 프레임워크들은 데이터 병렬처리만을 지원하지만, 기계학습/딥러닝 전용 프레임워크들은 대부분 데이터 병렬처리와 모델 병렬처리를 모두 지원한다.

파라미터 업데이트에서 빅데이터 처리 계열 프레임워크들은 대체로 동기화 방식을, 기계학습/딥러닝 전용 프레임워크들은 대체로 동기/비동기 방식을 모두 지원하나, H2O와 DeepSpark의 경우 비동기 방식을 도입하여 속도를 개선하려고 노력하고 있다.

파라미터 서버의 경우 H2O, DeepSpark과 같은 빅데이터 처리 계열 프레임워크과 Petuum, CNTK, MXNet과 같은 기계학습/딥러닝 전용 프레임워크들은 키-밸류 저장소 형태의 파라미터 서버를 지원하고 있다. 본고에서 분석한 분산처리 프레임워크는 모두 오픈 소스이며 대부분 Apache 2.0 라이선스 정책을 가지는데 CNTK의 경우 MIT 2.0 라이선스, Petuum은 BSD 3-clause license 정책을 가지고 있다.

〈표 1〉 딥러닝 분산처리 프레임워크 비교 분석

분산처리 프레임워크	개발사 (개발자)	개발언어	지원 OS	GPGPU 지원	병렬처리 (모델/데이터)	파라미터 서버 지원	동기화 방식	특징
SPARK	Apache	JAVA	Linux, Mac OS, Windows	X	데이터	X	동기	빅데이터 분산처리용
H2O	H2O.ai	JAVA	Linux, Mac OS, Windows	X	데이터	O	동기, 비동기	Spark+H20
DeepSPARK	서울대	JAVA	Linux, Mac OS, Windows	O (Caffe)	데이터	O	비동기	Spark+Caffe
SparkNet	UC Berkeley	JAVA	Linux, Mac OS, Windows	O (Caffe)	데이터	X	동기	Spark+Caffe
REEF	Microsoft	JAVA	Linux	X	데이터	X	동기	동적 부하 분산
Petuum	CMU	C++	Linux	X	데이터/모델	O	동기, 비동기	SSP 모델
SINGA	Apache (싱가폴대)	C++	Linux	O	데이터/모델	X	동기, 비동기	하이브리드
Veles	삼성 러시아연구소	Python, C++	Linux	O	데이터/모델	X		유닛을 조합하여 워크플로우 생성
CNTK	Microsoft	C++	Windows, Linux	O	데이터	O	동기, 비동기	그래프 모델 이용
MXNet	DMCL team	C++	Linux,Mac OS, Windows Android, iOS	O	데이터/모델	O	동기	선언적/지시적 프로그래밍 혼용

Ⅶ. 맺음말

본고에서는 딥러닝 분산처리 프레임워크들을 세 부류로 구분하고, 각 부류에 해당하는 기술들을 소개하고 비교 분석하였다.

SPARK을 중심으로 하는 빅데이터 처리 계열 딥러닝 분산처리 기술들에서는 계산 집중형 딥러닝 분산처리를 위해 GPGPU를 이용하는 별도의 딥러닝 프레임워크와 결합해야 하는 등 최고의 성능을 보장하기 어렵다. 기계학습 분산처리 기술과 딥러닝 전용 분산처리 기술들은 딥러닝 모델이 분산되었을 경우 파라미터 공유를 위한 통신 오버헤드를 안고 있으며 이를 SW 기법으로 해결하려는 상황이다.

결과적으로 효율적 딥러닝 분산처리의 핵심은 분산된 컴퓨터 간의 파라미터 공유를 위한 통신 오버헤드를 어떻게 줄이느냐에 있다. 따라서 확장성을 고려한 분산 토폴로지와 통신 오버헤드를 줄이기 위한 시스템 HW, SW 간의 협업 연구가 필요할 것으로 판단된다.

용어해설

GPGPU(General-Purpose computing on Graphics Processing Units) 일반적으로 컴퓨터 그래픽스를 위한 계산만 맡았던 그래픽 처리 장치(GPU)를, 전통적으로 중앙 처리 장치(CPU)가 맡았던 응용 프로그램들의 계산에 사용하는 기술

기계학습(Machine Learning) 인공지능의 한 분야로 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야

딥러닝(Deep Learning) 사람의 신경세포(Biological Neuron)를 모사하여 기계가 학습하도록 하는 인공신경망(Artificial Neural Network) 기반의 기계 학습법

약어 정리

BPTT	Back Propagation Through Time
CNN	Convolutional Neural Network
CNTK	Computational Network Toolkit
COCO	Common Objects in Context
DMTK	Distributed Machine Learning Toolkit
DNN	Deep Neural Network
GPGPU	General-Purpose computing on Graphics Processing Units
HDFS	HaDoop File System
HMM	Hidden Markov Model
ILSVRC	ImageNet Large Scale Visual Recognition Competition
JVM	Java Virtual Machine
LDA	Latent Dirichlet Allocation
LSTM	Long Short Term Memory
MEL	Model Editing Language
MLP	Multi-Layer Perceptron
NDL	Network Definition Language
RBM	Restricted Boltzmann Machines
RDD	Resilient Distributed Datasets
REEF	Retainable Evaluator Execution Framework
ResNet	Deep Residual Network
REST	Representational State Transfer
RNN	Recurrent Neural Networks
SGD	Stochastic Gradient Descent
SQL	Structured Query Language
SSP	Stale Synchronous Parallelism
TCP	Transmission Control Protocol
YARN	Yet Another Resource Negotiator

참고문헌

[1] 헤럴드경제, "[이세돌 vs 알파고 3국]구글 딥마인드, '불공정 게임 말도 안된다'," 2016. 3. 12.

[2] 조선비즈, "[이세돌 vs 알파고] 이지수 슈퍼컴 박사 '알파고 시스템 100억원대 슈퍼컴퓨터...알고리즘으로 승부'," 2016. 3. 10.

[3] X. Chen et al., "Pipelined Back-Propagation for Context-Dependent Deep Neural Networks," Proc. InterSpeech, Sept. 2012.

[4] Q. Le et al., "A. Building High-Level Features Using Large Scale Unsupervised Learning," *International Conference on Machine Learning*, 2012.

[5] A. Coates et al., "Deep Learning with COTS HPC Systems," *Proc. 30th International Conference on Machine Learning*, 2013, pp. 1337-1345.

[6] R. Wu et al., "Deep Image: Scaling up Image Recognition," 2015.

[7] Spark Lightning-Fast Cluster Computing, http://spark.apache.org/

[8] H2O, http://www.h2o.ai/

[9] H20 World Training, "Sparkling Water," 2014, https://h2o.gitbooks.io/h2o-training-day/content/hands-on_training/sparkling_water.html

[10] H.J. Kim et al., "DeepSpark: Spark-Based Deep Learning Supporting Asynchronous Updates and Caffe Compatibility," *ACM KDD*, 2016, http://arxiv.org/abs/1602.08191

[11] P. Moritz et al., "SparkNet: Training Deep Networks in Spark," *ICLR*, 2016.

[12] N. Irizarry Jr, "Mixing C and Java™ for High Performance Computing," MITRE Technical Report, Sept. 2013.

[13] B.-G. Chun, T. Condie, and C. Curino, "Reef: Retainable Evaluator Execution Framework," Proceedings of the VLDB Endowment, 2013, pp. 1370-1373.

[14] M. Weimer et al., "Reef: Retainable Evaluator Execution Framework," *Proc. ACM SIGMOD International Conference on Management of Data*, 2015, pp. 1343-1355.

[15] Apache REEF, https://reef.apache.org/

[16] Petuum, http://petuum.github.io/

[17] E.P. Xing and Q. Ho, "A New Look at the System, Algorithm and Theory Foundations of Large-Scale Distributed Machine Learning," Tutorials at KDD, 2015.

[18] H. Zhang et al., "Poseidon: A System Architecture for Efficient GPU-based Deep Learning on Multiple Machines," Dec. 2015, http://arxiv.org/abs/1512.06216

[19] W. Wang et al., "SINGA: Putting Deep Learning in the Hands of Multimedia Users," *Proc. 23rd ACM International Conference on Multimedia*, 2015, pp. 25-34.

[20] Veles, https://velesnet.ml/jenkins/job/VELES_Python_Veles_Tests/Veles_Machine_Learning_Platform_Documentation/

[21] Computational Network Toolkit(CNTK), https://cntk.codeplex .com/

[22] D. Yu et al., "An Introduction to Computational Networks and the Computational Network Toolkit," Tech. Rep. MSR, Microsoft Research, 2014, http://codebox/cntk

[23] T. Chen et al., "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems," arXiv Preprint arXiv:1512.01274, Dec. 2015.