BigDL

A Distributed Deep Learning Framework for Big Data
by Dai et al.

Nitin Nikamanth Appiah Balaji
What Is Big Data and BigDL?

1. **Big data** is a field that develops ways to
   - analyze
   - systematically extract information from, and
   - otherwise deal with data sets that are too large or complex

1. Apache Spark is one of the popular Big Data analytics tool.

1. BigDL is a DL framework implemented using Spark primitives.
Why do we need a new DL framework?

- We already have many **frameworks** -
  - Caffe, Torch, TensorFlow, MXNet, Chainer, PyTorch
- We also have **distributed learning frameworks** -
  - Horovod, PyTorch distributed, TF Distributed
- We can train any amounts of data using pre existing distributed DL frameworks.
- Then, why do we need a **new framework BigDL** for Big Data?
40 ZETTABYTES
of data will be created by 2020, an increase of 100 times from 2005

6 BILLION PEOPLE
have cell phones

30 BILLION PIECES OF CONTENT
are shared on Facebook every month

Most companies in the U.S. have at least
100 TERABYTES
of data stored

The New York Stock Exchange captures
1 TB of trade information
during each trading session

27% of respondents
in one survey were unsure of
how much of their data was inaccurate

4 BILLION+ HOURS OF VIDEO
are watched on YouTube each month

1 IN 3 BUSINESS
don’t trust the information
they use to make decisions

2.5 QUINTILLION BYTES
of data is created each day

4 BILLION TWEETS
are sent per day by
about 200 million
monthly active users

402 MILLION
of data in healthcare was
estimated to be
150 EXABYTES

As of 2011, the global size

Why do we need a new DL framework?

- **Characteristics of Big Data:**
  - Greater variety
  - Arriving in increasing volumes
  - More velocity

- **Development Cycle**
  - Preprocess data
  - Train model
Main Idea

- To **unify** DL training with *Big Data* analytics tools.
- Implement DL **without** hardware **modification** or software modification.
  - Use the same Spark cluster CPUs
  - Use Spark functions like `shuffle` and `broadcast` to implement distributed training
- Use Spark’s Driver-Worker task distribution engine
- Make it easy for software developers to concentrate on one framework
- To reduce latency of transferring huge datasets from Spark to separate DL frameworks
Motivation

- DL Community - focused on Accuracy on ImageNet or SQuAD - curated and explicitly labelled datasets.
- Big Data - Dynamic and messy
- Live data can be implicitly labelled; require complex processing
  - explicit feedback in recommendation engines
- Real-world data analytics pipeline is an iterative and recurrent process
- Processing data on a Spark cluster, and then export the processed data to a separate TensorFlow cluster for training/inference
Three ways to implement workflow pipeline

1. Separate Big Data Analysis tool and DL framework
2. Connector Approach
   - TFX, CaffeOnSpark, TensorFlowOnSpark, SageMaker
   - Time wasted in:
     - Inter-process Communication
     - Data serialization
     - Persistency
     - Impedance mismatches
3. DL framework built inside Big Data Analysis toolkit
   - BigDL for Apache Spark
BigDL Execution Model

- Key novelty of BigDL is how to efficiently implement these functionalities on a functional, coarse-grained compute model of Spark.
  - Coarse-grained: map, reduce, filter.
- Contrary to the conventional wisdom of the machine learning community (that fine-grained data access and in-place updates are critical for efficient distributed training).
- Copy-on-write: Dataset are immutable and can only be transformed into new dataset without side effects.
BigDL Execution Model

ex: \( rdd2 = rdd.map(lambda x: x ** 2) \)

- Resilient Distributed Dataset (RDD) - Data representation in Spark
- RDDs are immutable collection of records partitioned across a cluster, and can only be transformed to derive new RDDs (i.e., copy-on-write)
- Two different types of operations:
  - Transformation - map, filter, join, union.
  - Actions - reduce, first, count.
- Collects the transformations and performs the operations only when action operations are called.
Spark Driver Worker Structure

Figure 2: A Spark job consists of many Spark tasks; the driver node is responsible for scheduling and dispatching the tasks to worker nodes, which runs the actual Spark tasks.
Data Parallel Training Algorithm

Algorithm 1 Data-parallel training in BigDL

1: for \( i = 1 \) to \( M \) do
2:    //"model forward-backward" job
3:    for each task in the Spark job do
4:      read the latest \texttt{weights};
5:    get a random \texttt{batch} of data from local \textit{Sample} partition;
6:    compute local \texttt{gradients} (forward-backward on local \textit{model} replica);
7:    end for
8:    //"parameter synchronization" job
9:    aggregate (sum) all the \texttt{gradients};
10: update the \texttt{weights} per specified optimization method;
11: end for
Distributed training

- BigDL provides **synchronous data-parallel training**.
- Large number of clusters and local training takes few seconds on each node.
- So they say that synchronous training is much efficient than asynchronous training.
- BigDL runs on **Intel Xeon CPU servers**, which usually have large (100s of GB) memory size and can easily hold very large models.
- Hence, **model parallelism is not required**.
Local training as tasks

Figure 3: The "model forward-backward" spark job, which computes the local gradients for each model replica in parallel.

Each worker has a replica of the model parameters as an RDD. It also has the RDD data partition on each node. Trained as local tasks on each worker.
Parameter Synchronization

Algorithm 2 "Parameter synchronization" job

1: for each task \( n \) in the "parameter synchronization" job do
2: \hspace{1em} shuffle the \( n^{th} \) partition of all gradients to this task;
3: \hspace{1em} aggregate (sum) these gradients;
4: \hspace{1em} updates the \( n^{th} \) partition of the weights;
5: \hspace{1em} broadcast the \( n^{th} \) partition of the updated weights;
6: end for
"Parameter synchronization" job
Synchronization Efficiency

The implementation of AllReduce in BigDL has similar performance characteristics compared to Ring AllReduce from Baidu Research.

Total number of data transfers:

- Ring AllReduce: $2K(N-1)/N$
- BigDL: $2K$

$N$ is the number of nodes and $K$ is the total size of the parameters.
Evaluation on Single Node

MLPerf as the benchmarking tool

Neural Collaborative Filtering:

1. movie recommender using the MovieLens 20 Million dataset (ml-20m)
2. 20 million ratings and 465,000 tags applied to 27,000 movies by 138,000 users
3. a single Nvidia P100 GPU (PyTorch) vs (BigDL) Intel Skylake 8180 2.5GHz server (with 56 cores in total and 384 GB memory)
4. BigDL is 1.6X faster than PyTorch
Scalability of distributed training

- Showed scalability using Inception v1 model trained on ImageNet dataset
- Xeon servers (dual-socket Intel Broadwell 2.20GHz, 256GB RAM and 10GbE network)
- The scalability of distributed training in BigDL is determined by the efficiency (or overheads) of its parameter synchronizations.
- Lower overhead for sync -> better performance
Scalability of distributed training

- Reports by Cray
- Training throughput scales almost linearly up to 96 nodes.
- Continues to Scale upto 256 nodes.
Efficient Task Scheduling

- Each task runs for just a couple of seconds.
- Have to schedule large number of clusters in short period of time.
- **Drizzle**, a low latency execution engine for Spark, can help schedule multiple iterations (or a group) of computations at once, so as to greatly reduce scheduling overheads.

![Graph showing Overheads of task scheduling and dispatch](image-url)
Applications - Object Detection

- Show the comparison of a real-world object detection inference pipeline running on **BigDL vs. Caffe**.
- Reported by **JD.com**
- BigDL running on 24 Intel Xeon servers vs Caffe running on 5 servers and 20 GPU cards.
- BigDL is 3.83x faster than Caffe
Applications - Object detection
Image Throughput Comparison during Inferencing (Caffe vs BigDL)

- **Connector Approach:**
  GPU cluster consists of 5 node, 20 NVIDIA Tesla K40 cards.

- **HBase + Caffe - Impedance Mismatch**

- **BigDL:** Xeon cluster consists of 1200 logical cores (with each Intel Xeon E5-2650 v4 2.2GHz server running 50 logical cores).
Seq2Seq model built by Cray (convolutional LSTM network)

- Combination of TF and Spark -
  - Data movement overhead.
  - Affects development cycle.

Figure 11: End-to-end precipitation nowcasting workflow (using sequence-to-sequence models) on Spark and BigDL [45].

Figure 12: Predicting precipitation patterns for the next hour (i.e., a sequence of images for the future time steps of the next hour) on Spark and BigDL [45].
Call classification in Call centers by GigaSpaces

Process speech in realtime, classify using BigDL model and reroute to respective department.

Figure 13: The end-to-end workflow of real-time streaming speech classification on Kafka, Spark Streaming and BigDL [53].
Conclusion

- Integrating DL frameworks with Big Data.
- Ease of development and Deployment.
- Efficient use of hardware clusters and reliability.

Analysis

- A new DL framework means it needs good community to develop model architectures in that framework for software developers.
- There is an option to import pretrained TF, Torch or Caffe models, but training a model in BigDL is our goal.
- No scaling for asych and no GPU support.