

**Parallel Computing
Assignment Presentation**

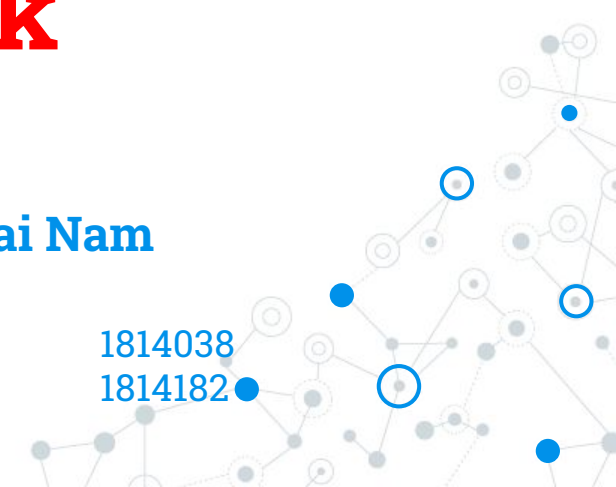
Distributed Deep Learning on Apache Spark

Lecturer: Mr. Thoai Nam

Student:

Vo Cong Thanh
Phan Khanh Thinh

1814038
1814182



Outline

1. Introduction
 - 1.1 Apache Spark
 - 1.2 BigDL
2. Implementation
 - 2.1 Problem Introduction
 - 2.2 BigDL Execution
 - 2.3 System Configuration
3. BigDL performance evaluation of system
 - 3.1 Execution running time
 - 3.2 Computation evaluation (SPEED UP)
4. Conclusions and Future work



A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, suggesting different levels of connectivity or importance. The lines are thin and gray, creating a mesh-like structure.

1. **Introduce**

A decorative network diagram in the bottom-right corner, similar to the one in the top-left, but smaller and less dense. It also features interconnected nodes and lines, with some nodes having concentric circles.

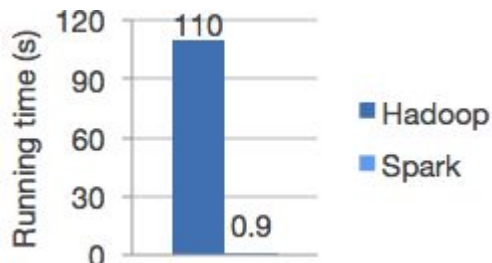
Introduce Apache Spark

- **Apache Spark is an open-source, distributed processing system used for big data workloads**
- **It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size.**

Features of Apache Spark

Speed: Run workloads 1000x faster than Hadoop

Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.



Ease of Use: Write applications quickly in Java, Scala, Python, R, and SQL.

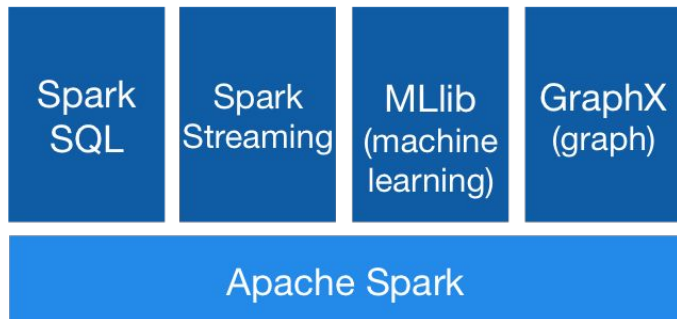
Spark offers over 80 high-level operators that make it easy to build parallel apps.

Can use it interactively from the Scala, Python, R, and SQL shells

Features of Apache Spark

Generality: Combine SQL, streaming, and complex analytics.

Can combine these libraries seamlessly in the same application.



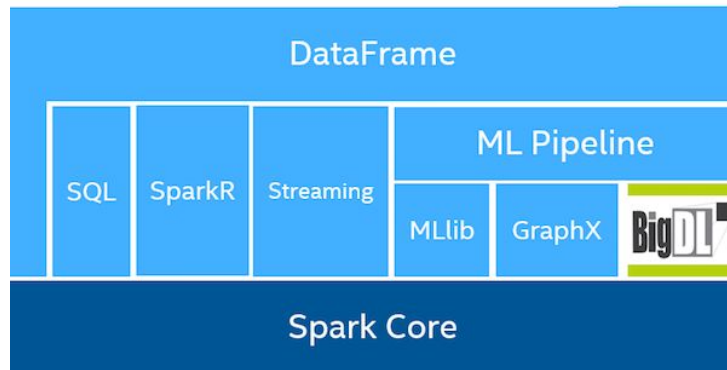
Runs everywhere: Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud.

It can access diverse data sources.



Introduce BigDL

- **BigDL is a distributed deep learning library for Apache Spark**
- **Can write deep learning applications as standard Spark programs, which can directly run on top of existing Spark or Hadoop clusters**



Features of BigDL

Rich deep learning support.

Modeled after Torch, BigDL provides comprehensive support for deep learning, including numeric computing (via Tensor) and high level neural networks

Users can load pre-trained Caffe or Torch models into Spark programs using BigDL

Extremely high performance.

To achieve high performance, BigDL uses Intel oneMKL, oneDNN and multi-threaded programming in each Spark task.

Consequently, it is orders of magnitude faster than out-of-box open source Caffe or Torch on a single-node Xeon (i.e., comparable with mainstream GPU).

Features of BigDL

Efficiently scale-out.

BigDL can efficiently scale out to perform data analytics at "Big Data scale", by leveraging Apache Spark (a lightning fast distributed data processing framework), as well as efficient implementations of synchronous SGD and all-reduce communications on Spark

Reasons choose BigDL

- **You want to analyze a large amount of data on the same Big Data (Hadoop/Spark) cluster where the data are stored (in, say, HDFS, HBase, Hive, Parquet, etc) .**
- **You want to add deep learning functionalities (either training or prediction) to your Big Data (Spark) programs and/or workflow.**
- **You want to leverage existing Hadoop/Spark clusters to run your deep learning applications, which can be then dynamically shared with other workloads .**

Utility of BigDL

- **BigDL helps us in balancing our needs:**
- **Big Compute: Fast Linear Algebra, Intel MKL library**
- **Big Data: I/O parallelized to run on many CPUs**
- **BigDL allows Massive Scalability:**
- **Runs on Spark**
- **Works with Hadoop eco system (via Spark)**
- **Hadoop is The Big Data platform for on-premise deployments**
- **Plays nicely with other BigDL frameworks:**
- **Use existing Tensorflow or Caffe at scale in BigDL**
- **Train new models based on existing TF/Caffe models**

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, suggesting different levels of connectivity or importance. The lines are thin and gray, creating a mesh-like structure.

2.

Implementation

A decorative network diagram in the bottom-right corner, similar to the one in the top-left, showing a cluster of nodes connected by lines. The nodes vary in size and some have concentric circles, and the lines are thin and gray.

2. Implementation

2.1 Problem Introduction

- **The MNIST handwritten digit classification problem is a standard dataset used in computer vision and deep learning**
- **This dataset was created by mixing different sets inside the original National Institute of Standards and Technology (NIST) sets**

2.1 Problem Introduction

● Dataset Description

- It is a dataset of 60,000 rows, square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.
- MNIST data includes:
 - Training data: 60,000 rows
 - Test data: 10,000 rows

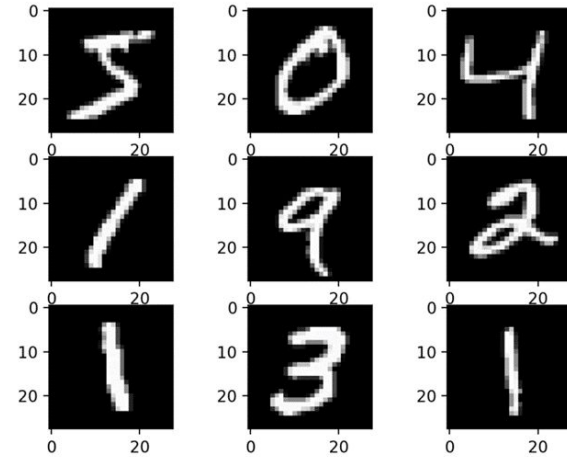


Figure 1: Illustration of the shape of numbers

2.1 Problem Introduction

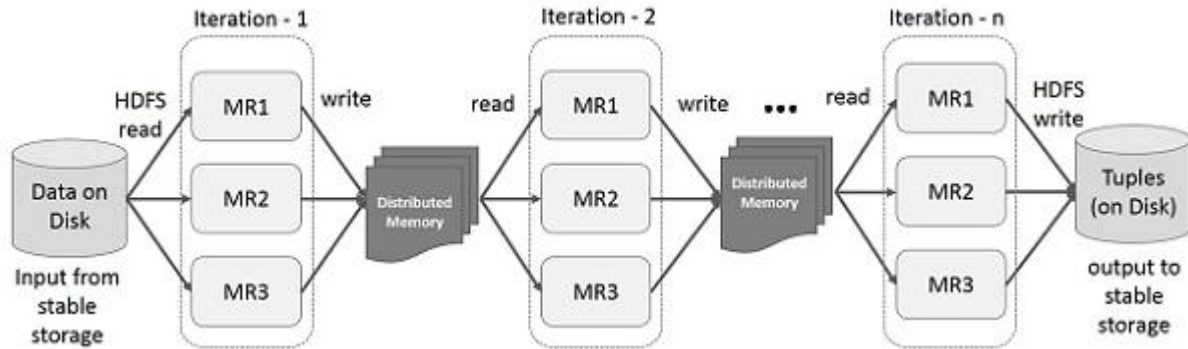
- **Problem Goal**

- The task is to classify a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9, inclusively.
- The MNIST dataset has been the target of so many research done in recognizing handwritten digits.

2.1 Problem Introduction

- **Data as a RDD**

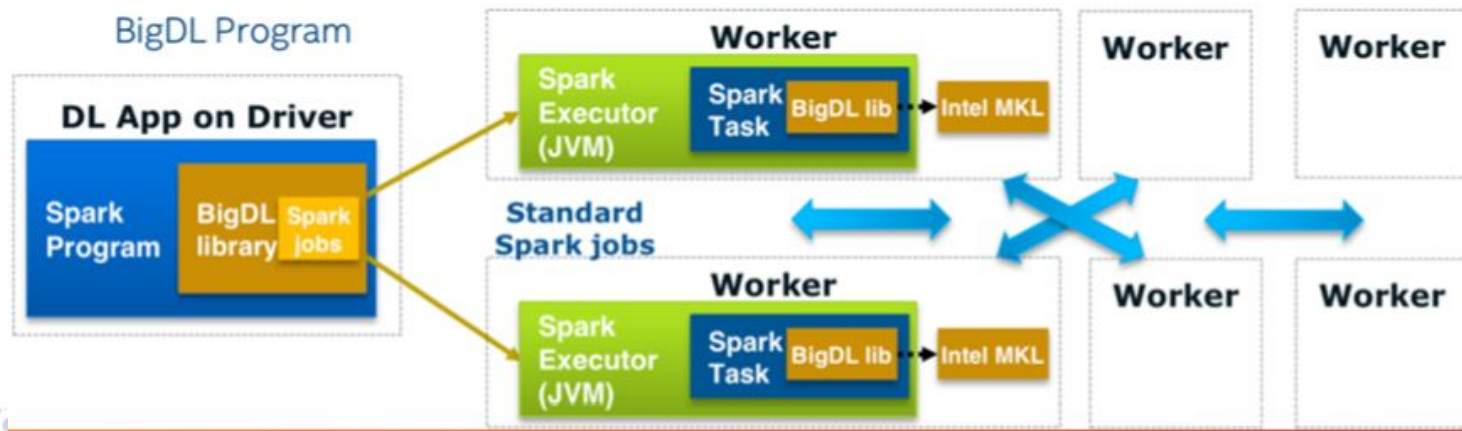
- Resilient Distributed Datasets (RDD) is an underlying data structure of Spark.
- RDD represents a fixed set of records that have been partitioned for parallel processing.



2.2 BigDL Execution

- **Spark execution model**

- A Spark cluster consists of a single driver node and multiple worker nodes
- The driver is responsible for coordinating tasks in a Spark job



2.2 BigDL Execution

- **BigDL training model**
 - We use LSTM model for MNIST digit classification problem:

```
1 def build_model(input_size, hidden_size, output_size):  
2     model = Sequential()  
3     recurrent = Recurrent()  
4     recurrent.add(LSTM(input_size, hidden_size))  
5     model.add(InferReshape([-1, input_size], True))  
6     model.add(recurrent)  
7     model.add(Select(2, -1))  
8     model.add(Linear(hidden_size, output_size))  
9     return model  
10 rnn_model = build_model(n_input, n_hidden, n_classes)
```

2.2 BigDL Execution

- **BigDL training model**
 - **We use LSTM model for MNIST digit classification problem:**
 - **Model Parameter Setup**
 - `batch_size = 64`
 - `n_input = 28` # *MNIST data input (img shape: 28*28)*
 - `n_hidden = 128` # *hidden layer num of features*
 - `n_classes = 10` # *MNIST total classes (0-9 digits)*

2.2 BigDL Execution

- **System Configuration**

- **Program run on BKHCM's HPC system includes:**
- **Options**
 - **System/Host Processor: Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz**
 - **CPU(s): 48**
 - **Core(s) per socket:12**
 - **Socket(s): 2**
 - **Memory: ~183 G (free)**

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric rings, suggesting a hierarchical or multi-layered structure. The lines are thin and gray, connecting the nodes in a non-linear fashion.

3.

BigDL performance evaluation

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It shows a cluster of nodes connected by lines, with some nodes being larger and having concentric rings, indicating a similar hierarchical or multi-layered structure. The lines are thin and gray.

3. BigDL performance evaluation

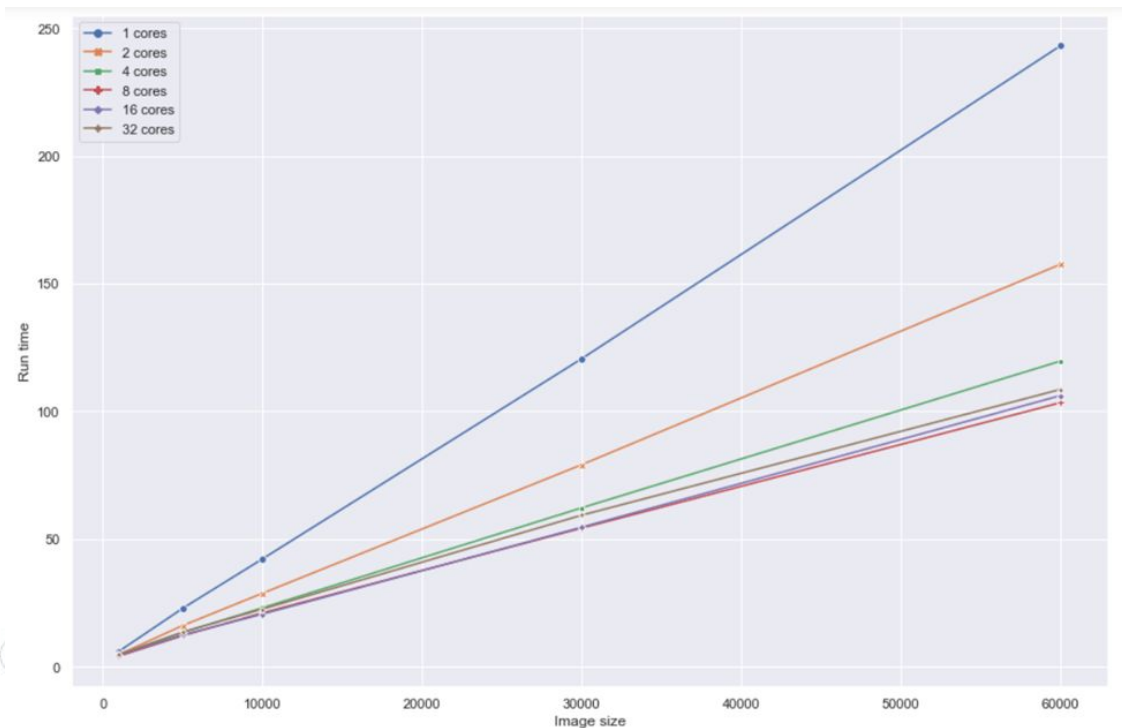
3.1 Execution running time

Image Size	1 cores	2 cores	4 cores	8 cores	16 cores	32 cores
1000	6.11	4.87	4.37	4.28	4.54	4.94
5000	22.97	16.22	13.50	12.27	12.55	13.69
10000	42.30	28.86	23.18	21.19	20.73	22.68
30000	120.62	79.16	62.29	54.37	54.71	59.41
60000	243.05	157.56	119.71	103.45	106.21	108.63

Table 1: Table show Runtime of cores on each relative image size. Bold's the best.

3. BigDL performance evaluation

3.1 Execution running time



Line graph show **RUNTIME** of cores on each relative image size

3. BigDL performance evaluation

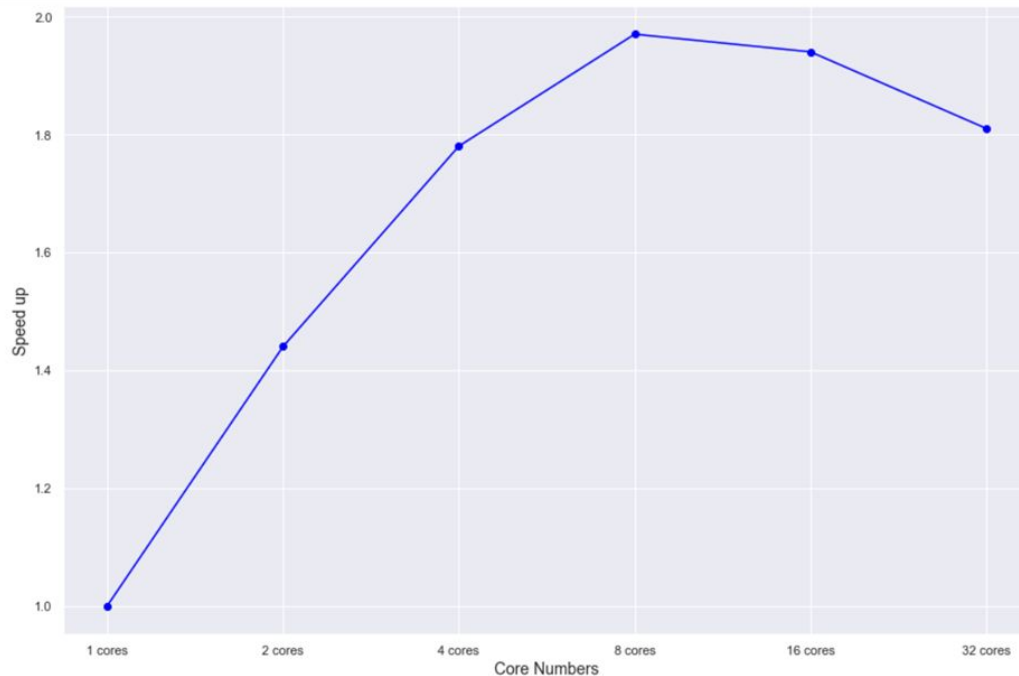
3.1 Computation evaluation (SPEED UP)

Image Size	1 cores	2 cores	4 cores	8 cores	16 cores	32 cores
1000	1	1.25	1.40	1.43	1.35	1.24
5000	1	1.42	1.70	1.87	1.83	1.68
10000	1	1.47	1.82	2.00	2.04	1.87
30000	1	1.52	1.94	2.22	2.20	2.03
60000	1	1.54	2.03	2.35	2.29	2.24
Average SPEED UP	1	1.44	1.78	1.97	1.94	1.81


Table 2: Table show SPEED UP of Image Training of cores

3. BigDL performance evaluation


3.1 Execution running time



Line Graph show SPEED UP of Image Training of cores

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, suggesting a hierarchical or multi-layered structure. The lines are thin and gray, connecting the nodes in a non-linear fashion.

4. Demo

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It shows a cluster of nodes connected by lines, with some nodes being larger and having concentric circles, indicating a similar hierarchical or multi-layered structure. The lines are thin and gray.

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, while others are smaller and solid. The lines are thin and gray, connecting the nodes in a non-linear fashion. The overall structure is dense and organic, resembling a molecular or biological network.

5.

Conclusions and future work

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It features a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, while others are smaller and solid. The lines are thin and gray, connecting the nodes in a non-linear fashion. The overall structure is dense and organic, resembling a molecular or biological network.

4. Conclusions and future work

- We have described BigDL, including its distributed execution model, data integration, computation performance.
- It provides efficient and scalable distributed training directly on top of the functional compute model of Spark.
- Continue to improve the BigDL model as well as dig deeper into running on spark for better performance.



Thanks for listening!