BigDL: A Distributed Deep Learning Framework for Big Data

Jason (Jinquan) Dai[1], Yiheng Wang[2 ‡], Xin Qiu[1], Ding Ding[1], Yao Zhang[3 ‡], Yanzhang Wang[1], Xianyan Jia[4 ‡], Cherry (Li) Zhang[1], Yan Wan[4 ‡], Zhichao Li[1], Jiao Wang[1], Shengsheng Huang[1], Zhongyuan Wu[1], Yang Wang[1], Yuhao Yang[1], Bowen She[1], Dongjie Shi[1], Qi Lu[1], Kai Huang[1], Guoqiong Song[1]

[1]Intel, [2] Tencent, [3] Sequoia Capital, [4]Alibaba, ‡ *Work was done when the author worked at Intel*

Agenda

- Motivation
- BigDL Execution Model
- Experimental Evaluation
- Real-World Applications
- Future Work

Real-World ML/DL Systems Are Complex Big Data Analytics Pipelines

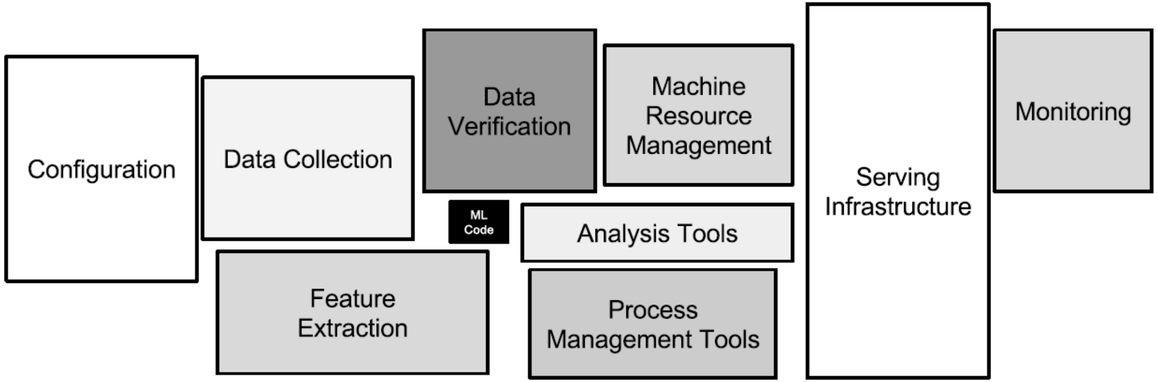


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

"Hidden Technical Debt in Machine Learning Systems", Sculley et al., Google, NIPS 2015 Paper

Big Data Analysis Challenges

Real-World data analytics and deep learning pipelines are challenging

- Deep learning benchmarks (ImageNet, SQuAD , etc.)
  - Curated and explicitly labelled Dataset
  - Suitable for dedicated DL systems

- Real-world production data pipeline
  - Dynamic, messy (and possibly implicitly labeled) dataset
  - Suitable for integrated data analytics and DL pipelines using BigDL

- Problems with “connector approaches”
  - TFX, TensorFlowOnSpark, Project Hydrogen, etc.
  - Adaptation overheads, impedance mismatch

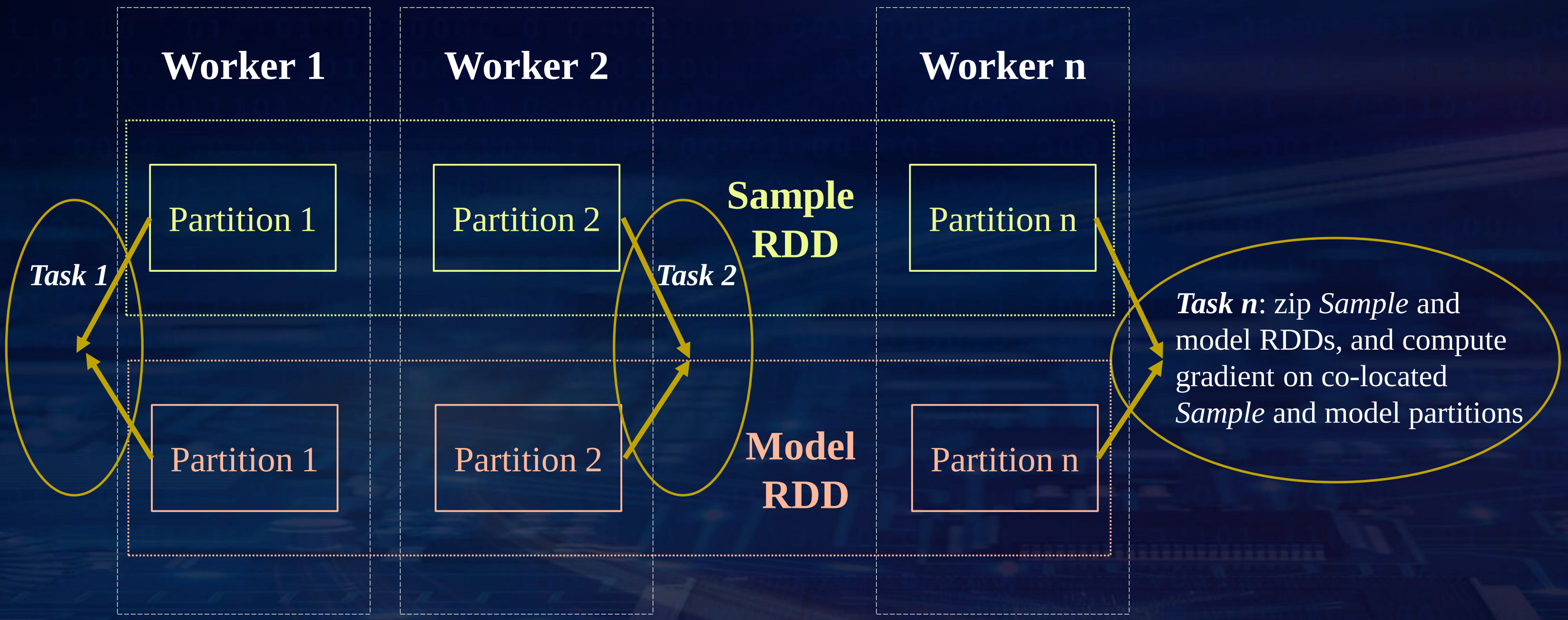
BigDL Execution Model

Distributed Training in BigDL

Data Parallel, Synchronous Mini-Batch SGD

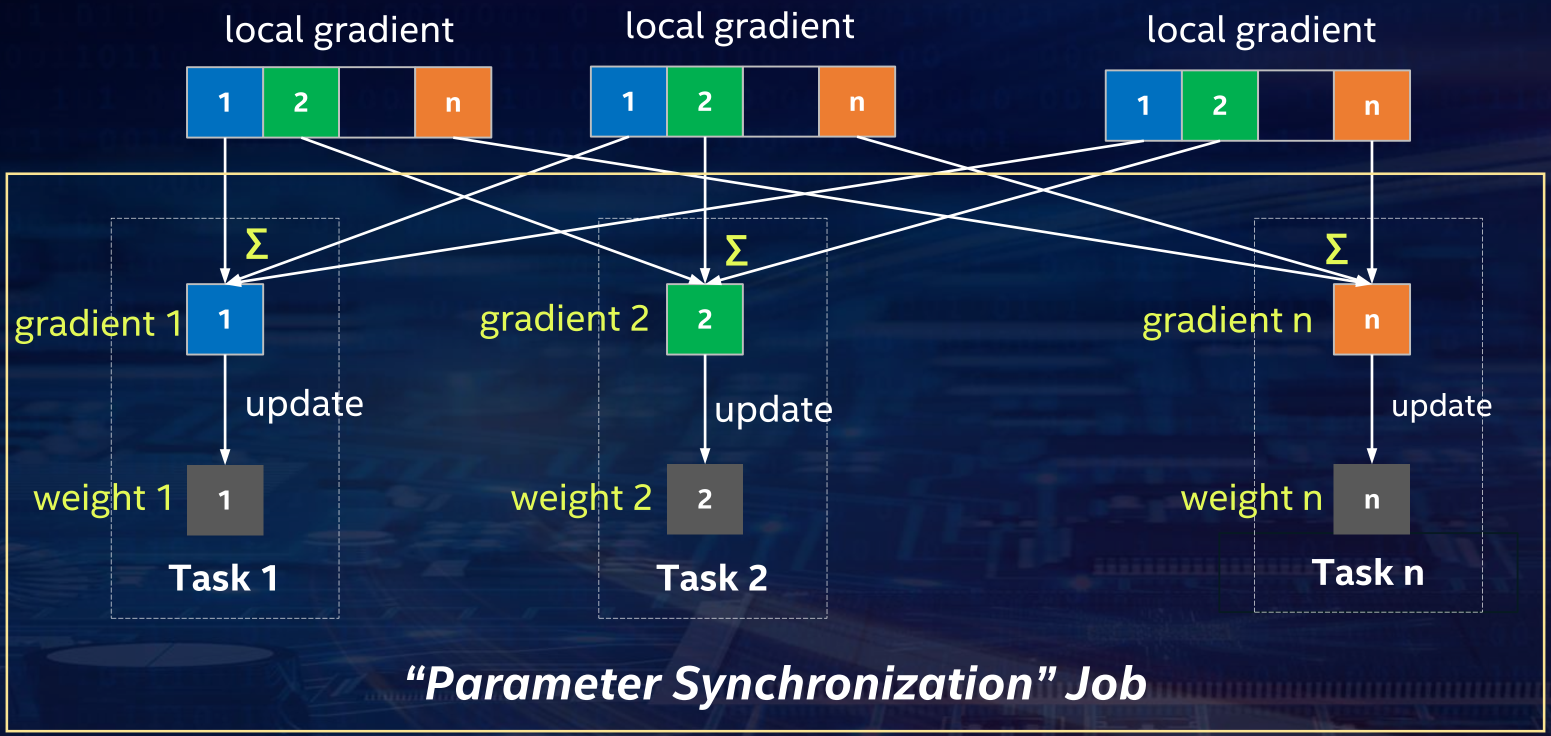
```
Prepare training data as an RDD of Samples
Construct an RDD of models (each being a replica of the original model)

for (i <- 1 to N) {
  //"model forward-backward" job
  for each task in the Spark job:
     read the latest weights
     get a random batch of data from local Sample partition
     compute errors (forward on local model replica)
     compute gradients (backward on local model replica)

  //"parameter synchronization" job
  aggregate (sum) all the gradients
  update the weights per specified optimization method
}
```

Data Parallel Training



"Model Forward-Backward" Job

Parameter Synchronization



Parameter Synchronization

```
For each task n in the "parameter synchronization" job {
    shuffle the nth partition of all gradients to this task
    aggregate (sum) the gradients
    updates the nth partition of the weights
    broadcast the nth partition of the updated weights
}
```

"Parameter Synchronization" Job
(managing n^{th} partition of the parameters - similar to a parameter server)

AllReduce Operation (directly on top of primitives in Spark)

- Gradient aggregation: *shuffle*
- Weight sync: *task-side broadcast*
- *In-memory* persistence

Difference vs. Classical AllReduce

Classical AllReduce architecture

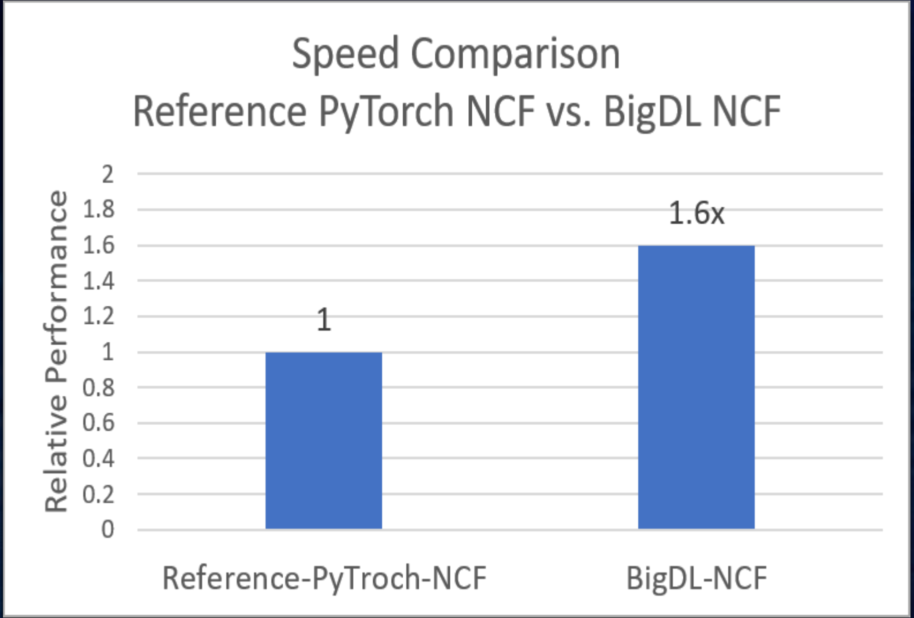
- Multiple long-running, potentially stateful tasks
- Interact with each other (in a blocking fashion for synchronization)
- Require fine-grained data access and in-place data mutation
- Not directly supported by existing big data systems

BigDL implementation

- Run a series of short-lived Spark jobs (e.g., two jobs per mini-batch)
- Each task in the job is stateless and non-blocking
- Automatically adapt to the dynamic resource changes (e.g., *preemption*, *failures*, *resource sharing*, etc.)
- Built on top of existing primitives in Spark (e.g., *shuffle*, *broadcast*, and *in-memory data persistence*)

Experimental Evaluation

Computing Performance



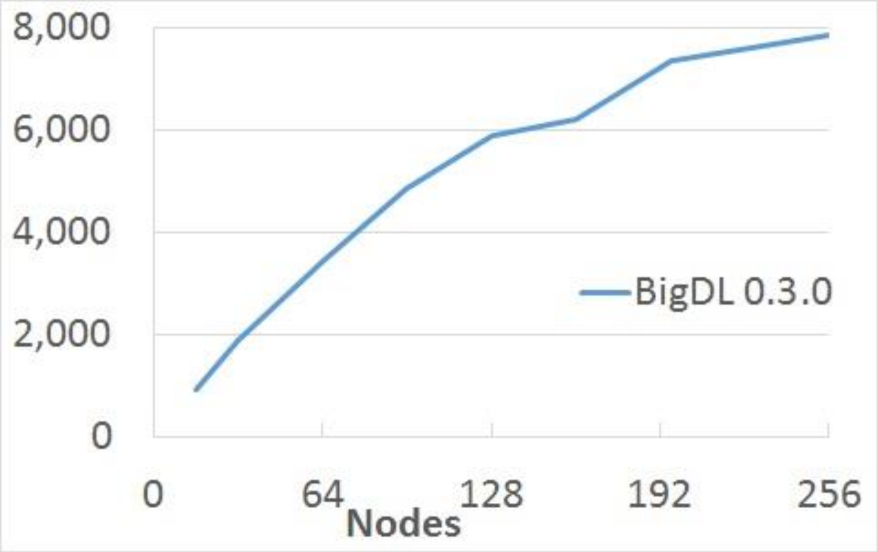
NCF training on single node:

- **PyTorch 0.4 on Nvidia P100 GPU**
- **BigDL 0.7.0 and Spark 2.1.0 on a dual-socket Intel Skylake 8180 server (56 cores and 384GB)**

The training performance of NCF using the BigDL implementation is 1.6x faster than the reference PyTorch implementation, as reported by MLPerf

MLPerf 0.5 training results URL: https://mlperf.org/training-results-0-5

Training Scalability

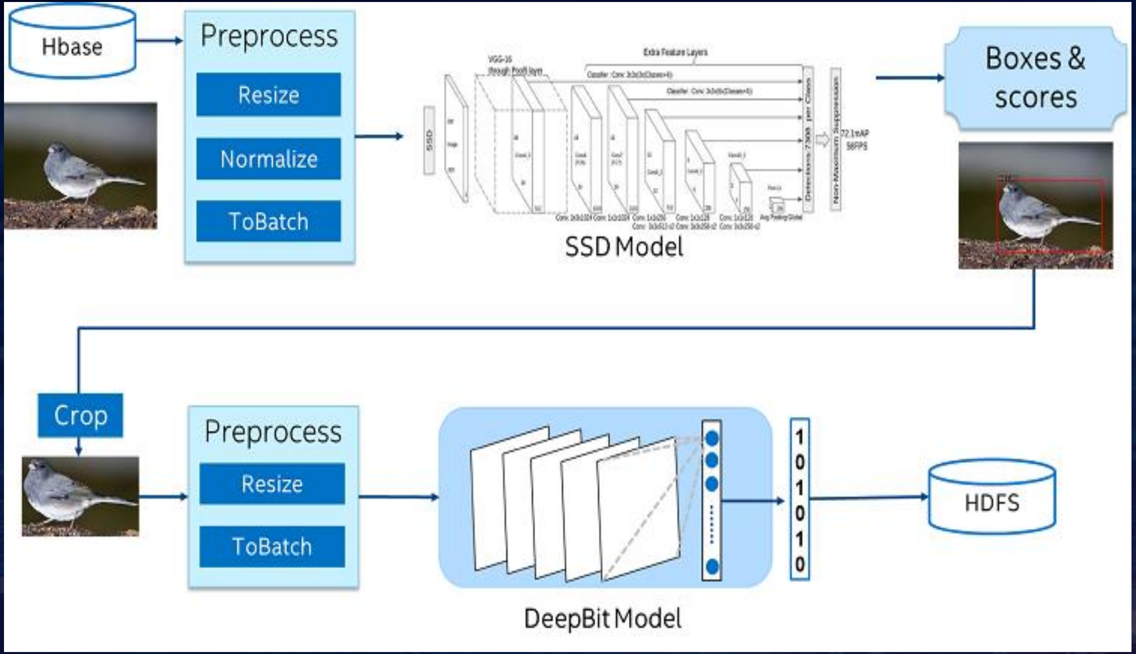


Throughput of ImageNet Inception v1 training (w/ BigDL 0.3.0 and dual-socket Intel Broadwell 2.1 GHz); the throughput scales almost linear up to 128 nodes (and continue to scale reasonably up to 256 nodes).

Source: Scalable Deep Learning with BigDL on the Urika-XC Software Suite
(https://www.cray.com/blog/scalable-deep-learning-bigdl-urika-xc-software-suite/)

Real-World Applications

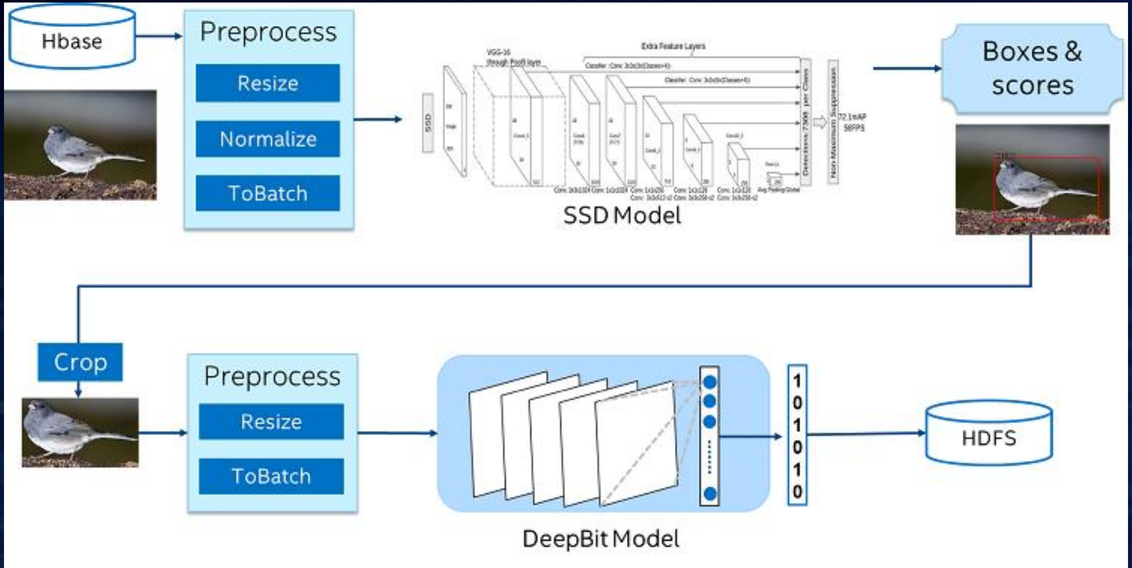
Object Detection and Image Feature Extraction at JD.com

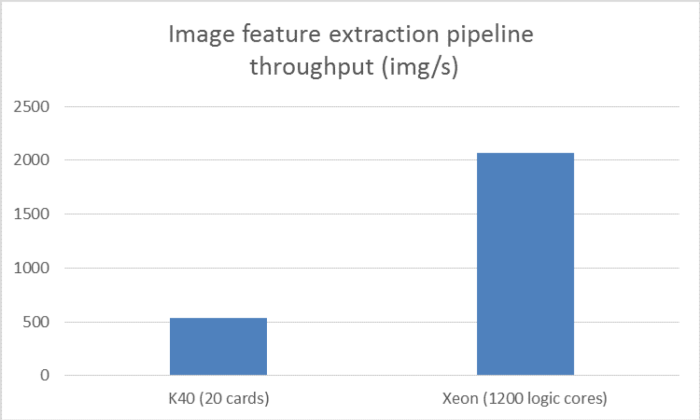


Problem with previous "connector approach" (similar to CaffeOnSpark)

- Very complex and error-prone in managing large-scale distributed systems
- Impedance mismatch
  - *Mismatch in the parallelism for data processing and for model compute*

Object Detection and Image Feature Extraction at JD.com





- Implement the entire data analysis and deep learning pipeline under a unified programming paradigm on Spark
- Greatly improves the efficiency of development and deployment
- Efficiently scale out on Spark with superior performance (***3.83x*** speed-up vs. GPU severs) as benchmarked by JD

https://software.intel.com/en-us/articles/building-large-scale-image-feature-extraction-with-bigdl-at-jdcom

And Many More

TECHNOLOGY

bluedata
cloudera
CRAY THE SUPERCOMPUTER COMPANY
databricks
DELL EMC
GIGASPACES innovate with confidence
Lightbend
Qubole

CLOUD SERVICE PROVIDERS

Alibaba Cloud aliyun.com
aws
Cloud Dataproc
Azure
IBM Cloud
Telefónica
KINGSOFT

END USERS

cdhi
中国电信 CHINA TELECOM
JD.COM 京东
Midea
UnionPay 银联

http://software.intel.com/bigdl/build

*Other names and brands may be claimed as the property of others.

Not a full list

Future Work

Analytics Zoo: Unified Data Analytics + AI Platform

Distributed TensorFlow, Keras, PyTorch and BigDL on Apache Spark

Use case	Recommendation; Anomaly Detection; Text Classification; Text Matching
Model	Image Classification; Object Detection; Seq2Seq; Transformer; BERT
Feature Engineering	image; 3D image; text; Time series
High Level Pipelines	tfpark: Distributed TF on Spark; Distributed Keras w/ autograd on Spark; nnframes: Spark Dataframes & ML Pipelines for Deep Learning; Distributed Model Serving (batch, streaming & online)
Backend/ Library	TensorFlow; Keras; PyTorch; BigDL; NLP Architect; Apache Spark; Apache Flink; Ray; MKLDNN; OpenVINO; Intel® Optane™ DCPMM; DL Boost (VNNI)

Analytics Zoo tutorial: https://github.com/jason-dai/aaai2019

https://github.com/intel-analytics/analytics-zoo

AI on Apache Spark™

BigDL

Distributed, High-Performance
Deep Learning Framework
for Apache Spark*

https://github.com/intel-analytics/bigdl

Analytics Zoo

Analytics + AI Platform

Distributed TensorFlow*, Keras*, PyTorch* and BigDL on Apache Spark*

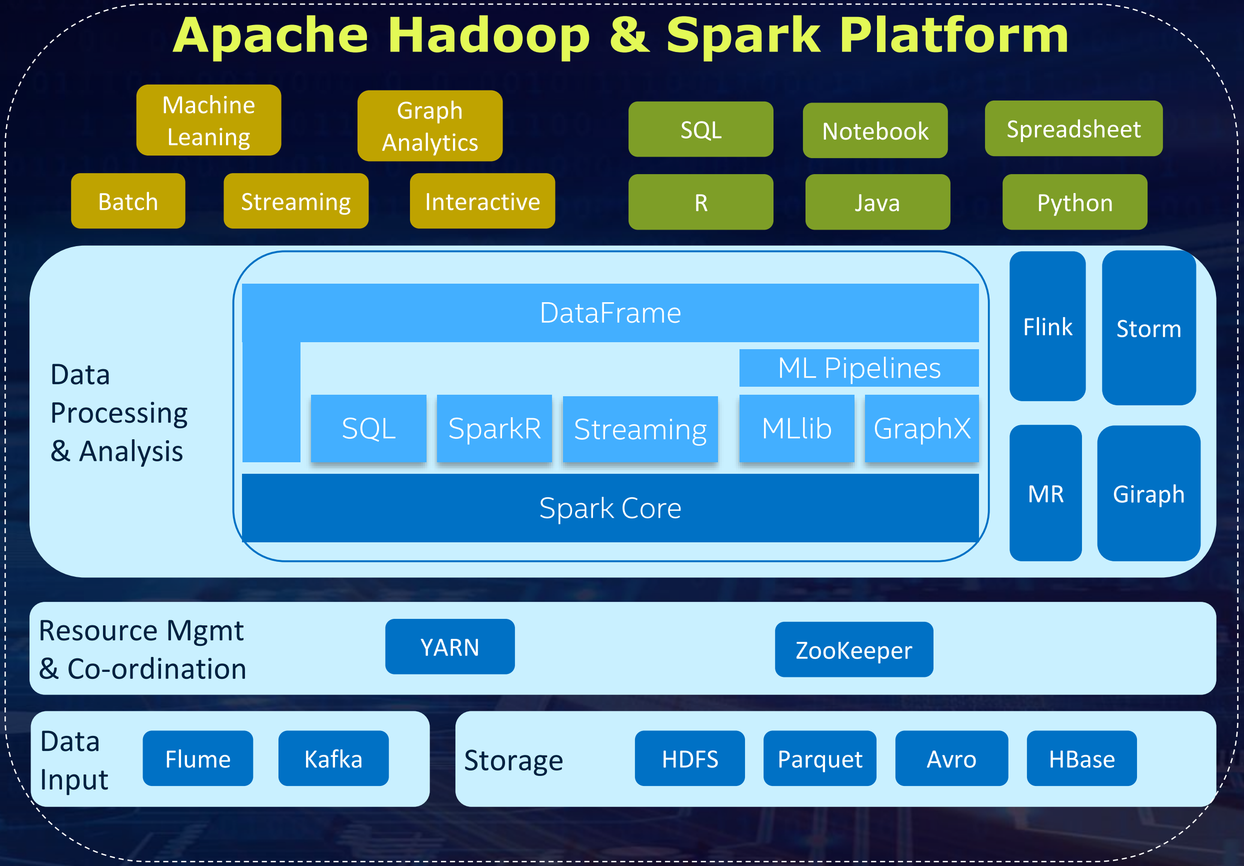
https://github.com/intel-analytics/analytics-zoo

Accelerating Data Analytics + AI Solutions At Scale

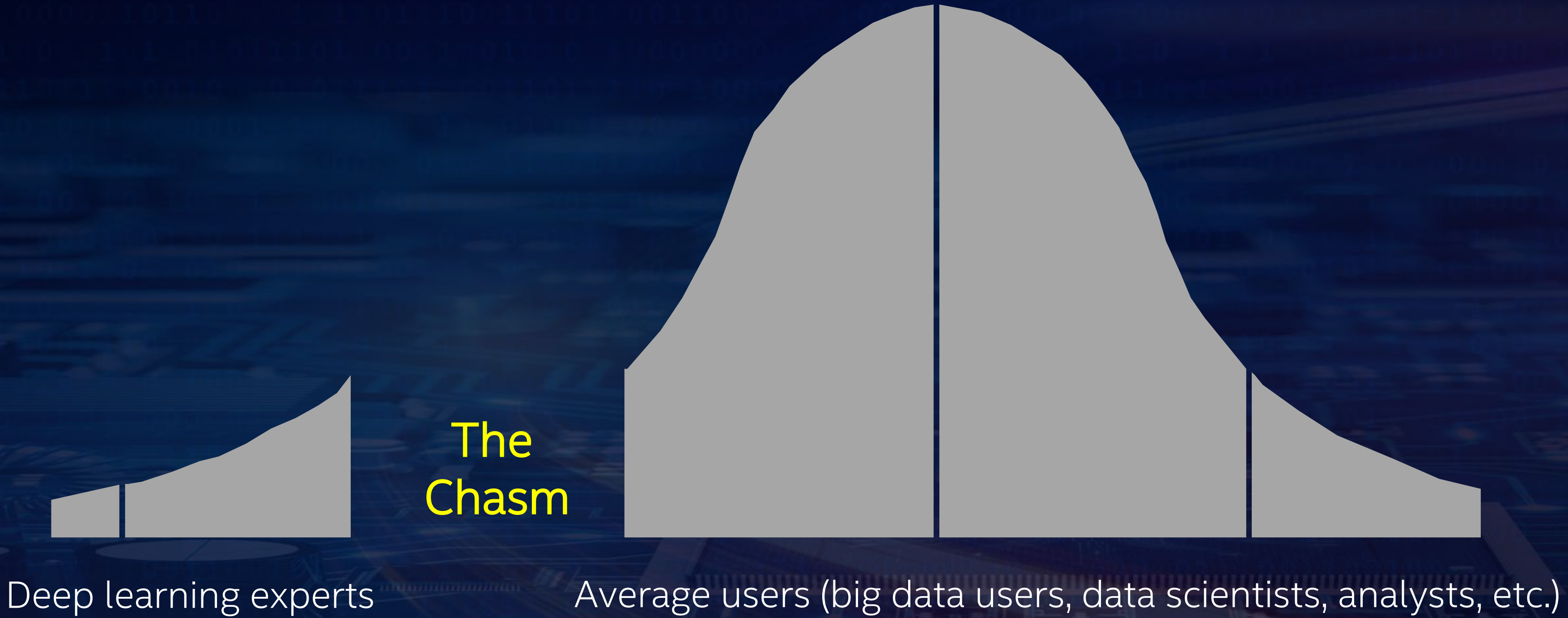
Q & A

Appendix

Unified Big Data Analytics Platform



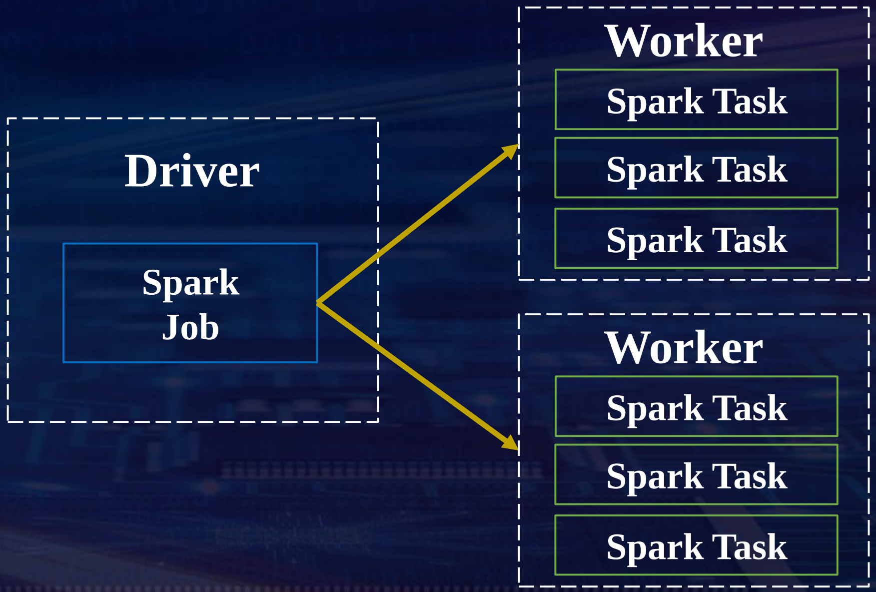
Chasm b/w Deep Learning and Big Data Communities



Apache Spark

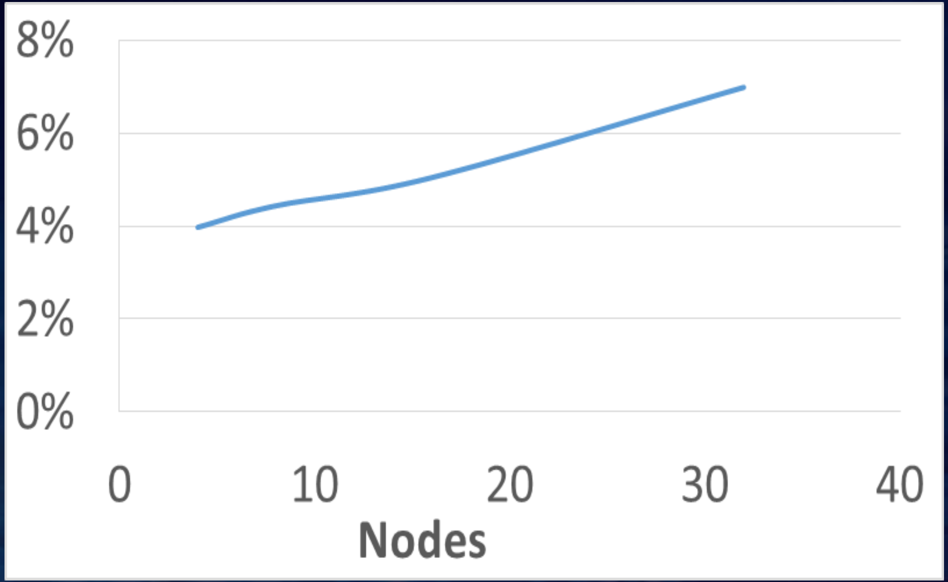
Low Latency, Distributed Data Processing Framework

- A Spark cluster consists of a single *driver* node and multiple *worker* nodes
- A Spark *job* contains many Spark *tasks*, each working on a data *partition*
- Driver is responsible for scheduling and dispatching the tasks to workers, which runs the actual Spark tasks



https://spark.apache.org

Training Scalability



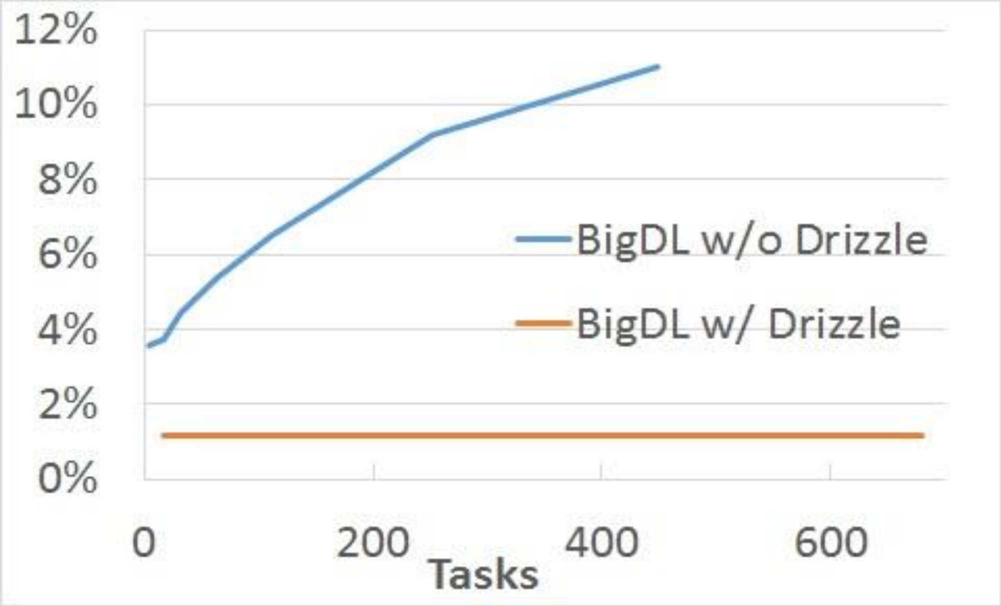
Overheads of parameter synchronization (as a fraction of average model computation time) of ImageNet Inception-v1 training in BigDL

Source: Scalable Deep Learning with BigDL on the Urika-XC Software Suite (https://www.cray.com/blog/scalable-deep-learning-bigdl-urika-xc-software-suite/)

Reducing Scheduling Overheads Using Drizzle

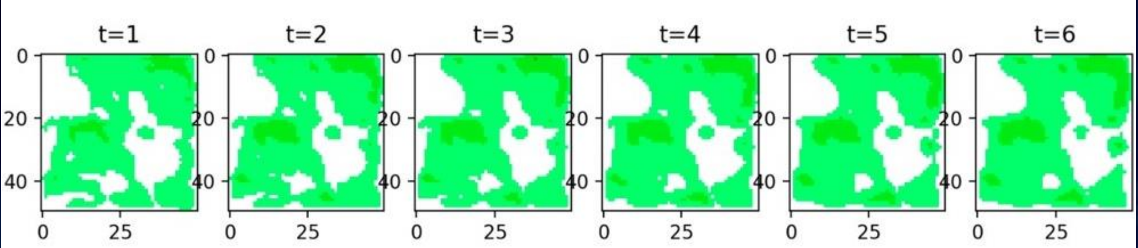
Scaling to even larger (>500) workers

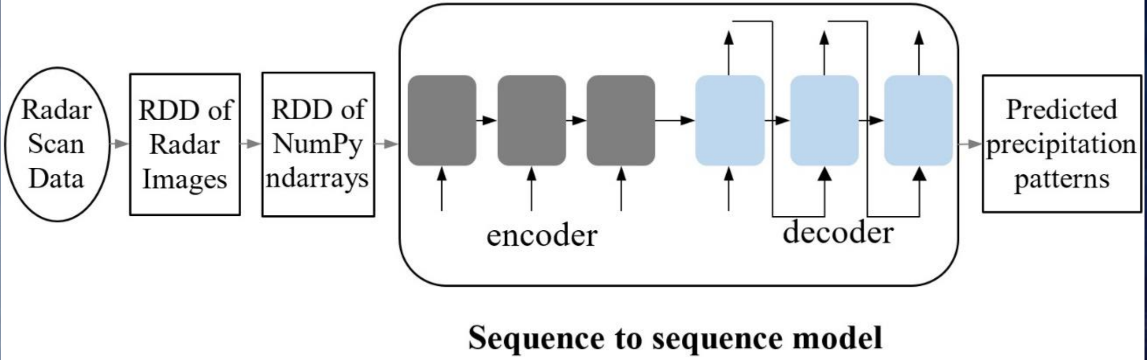
- Iterative model training
  - Same operations run repeatedly
- Drizzle
  - A low latency execution engine for Spark
  - *Group scheduling* for multiple iterations of computations at once



Source: Accelerating Deep Learning Training with BigDL and Drizzle on Apache Spark, Shivaram Venkataraman, Ding Ding, and Sergey Ermolin. (https://rise.cs.berkeley.edu/blog/accelerating-deep-learning-training-with-bigdl-and-drizzle-on-apache-spark/)

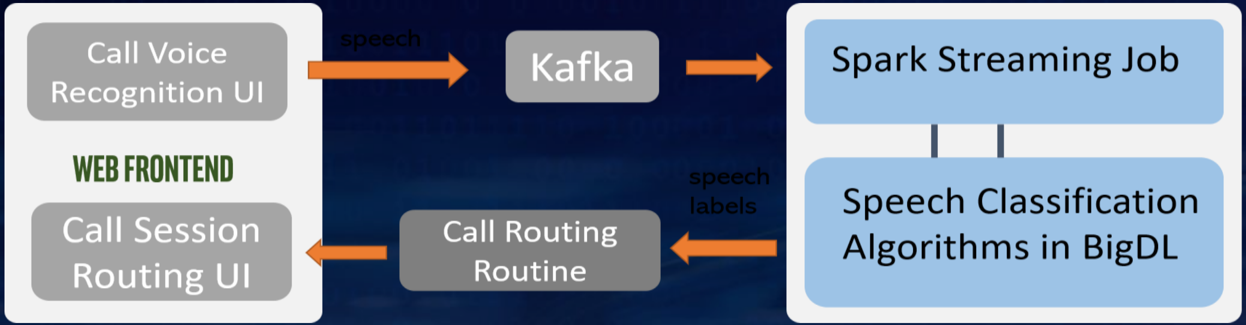
Precipitation nowcasting using sequence-to-sequence models in Cray





- Running data processing on a Spark cluster, and deep learning training on GPU cluster not only brings high data movement overheads, but hurts the development productivity due to the fragmented workflow
- Using a single unified data analysis and deep learning pipeline on Spark and BigDL improves the efficiency of development and deployment

Real-time streaming speech classification in GigaSpaces



The end-to-end workflow of real-time streaming speech classification on Kafka, Spark Streaming and BigDL in GigaSpaces.

- BigDL allows neural network models to be directly applied in standard distributed streaming architecture for Big Data (using Apache Kafka and Spark Streaming), and efficiently scales out to a large number of nodes in a transparent fashion.

https://www.gigaspaces.com/blog/gigaspaces-to-demo-with-intel-at-strata-data-conference-and-microsoft-ignite/

LEGAL DISCLAIMERS

- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.
- No computer system can be absolutely secure.
- Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.