

VERY LARGE-SCALE DISTRIBUTED DEEP LEARNING ON BIGDL

JASON DAI, DING DING

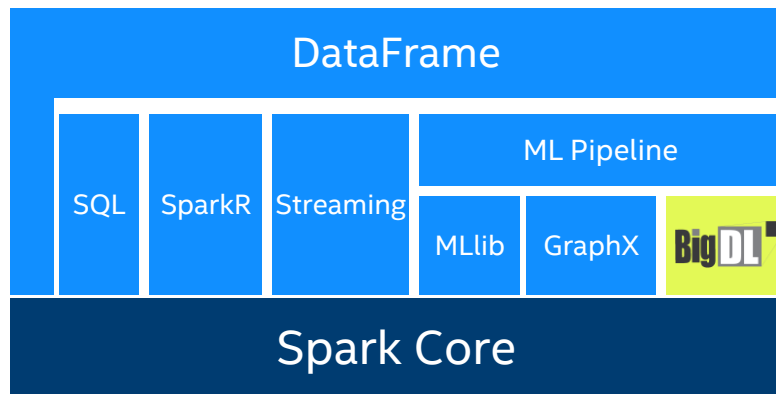


OVERVIEW OF BIGDL

BIGDL

BRINGING DEEP LEARNING TO BIG DATA PLATFORM

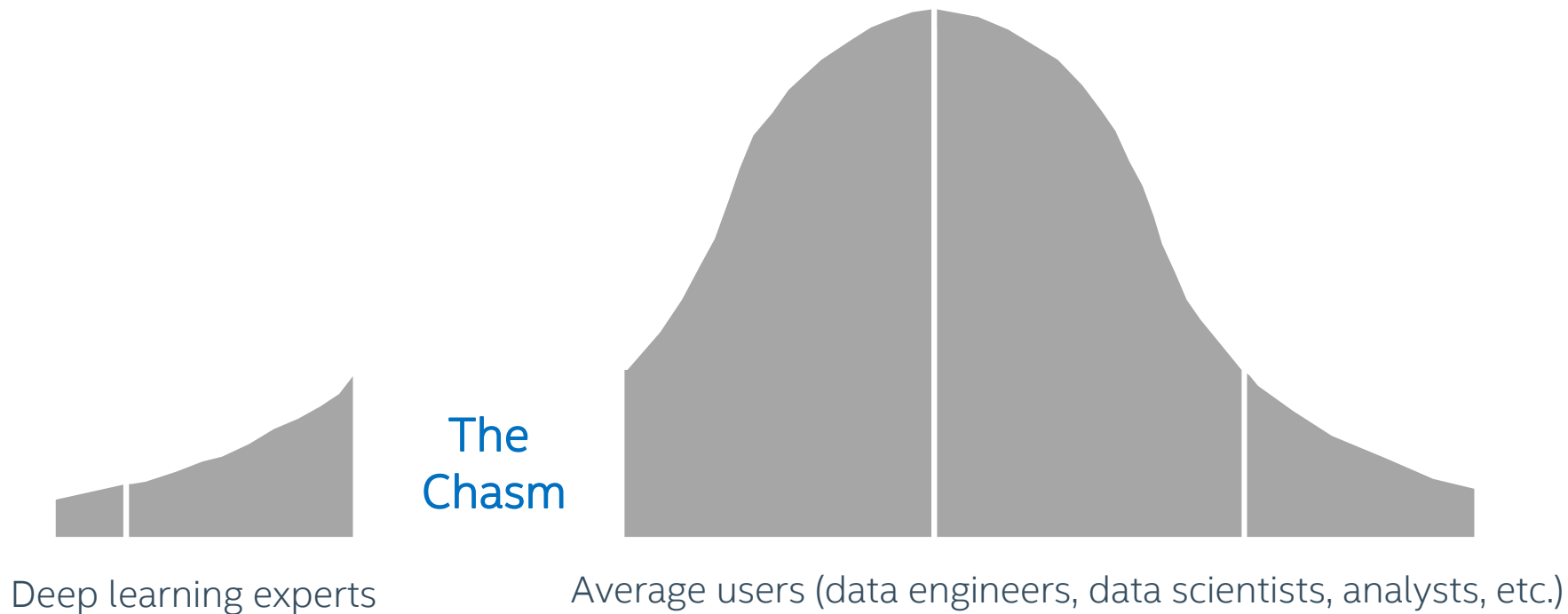
- **Distributed** deep learning framework for Apache Spark*
- Make deep learning more accessible to **big data users** and **data scientists**
 - Write deep learning applications as **standard Spark programs**
 - Run on existing Spark/Hadoop clusters (**no changes needed**)
- Feature parity with popular deep learning frameworks
 - E.g., Caffe, Torch, Tensorflow, etc.
- High performance
 - Powered by Intel MKL and multi-threaded programming
- Efficient scale-out
 - Leveraging Spark for distributed training & inference



<https://github.com/intel-analytics/BigDL>

<https://bigdl-project.github.io/>

CHASM B/W DEEP LEARNING AND BIG DATA COMMUNITIES

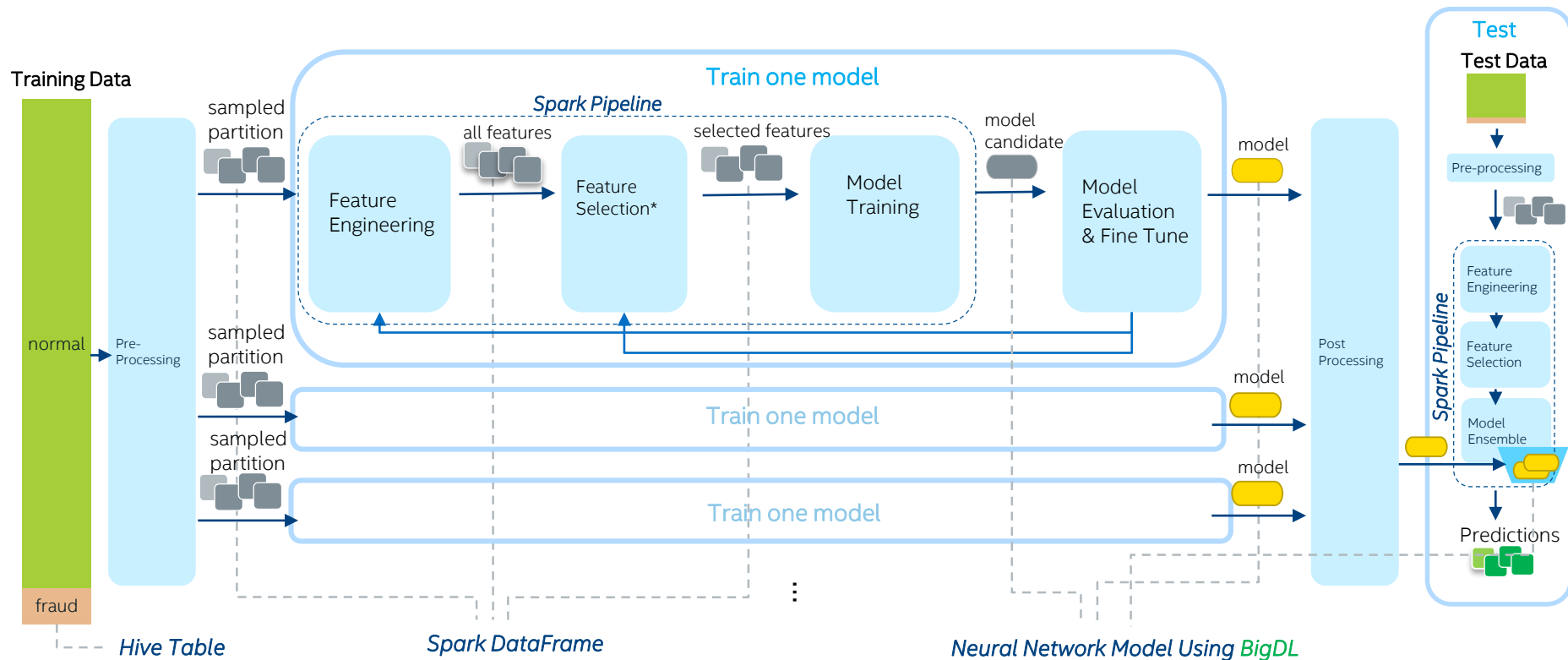


BIGDL ANSWERING THE NEEDS

Make deep learning more accessible to big data and data science communities

- Continue the use of familiar SW tools and HW infrastructure to build deep learning applications
- Analyze “big data” using deep learning on the same Hadoop/Spark cluster where the data are stored
- Add deep learning functionalities to the Big Data (Spark) programs and/or workflow
- Leverage existing Hadoop/Spark clusters to run deep learning applications
 - Shared with other workloads (e.g., *ETL, data warehouse, feature engineering, statistic machine learning, graph analytics, etc.*) in a dynamic and elastic fashion

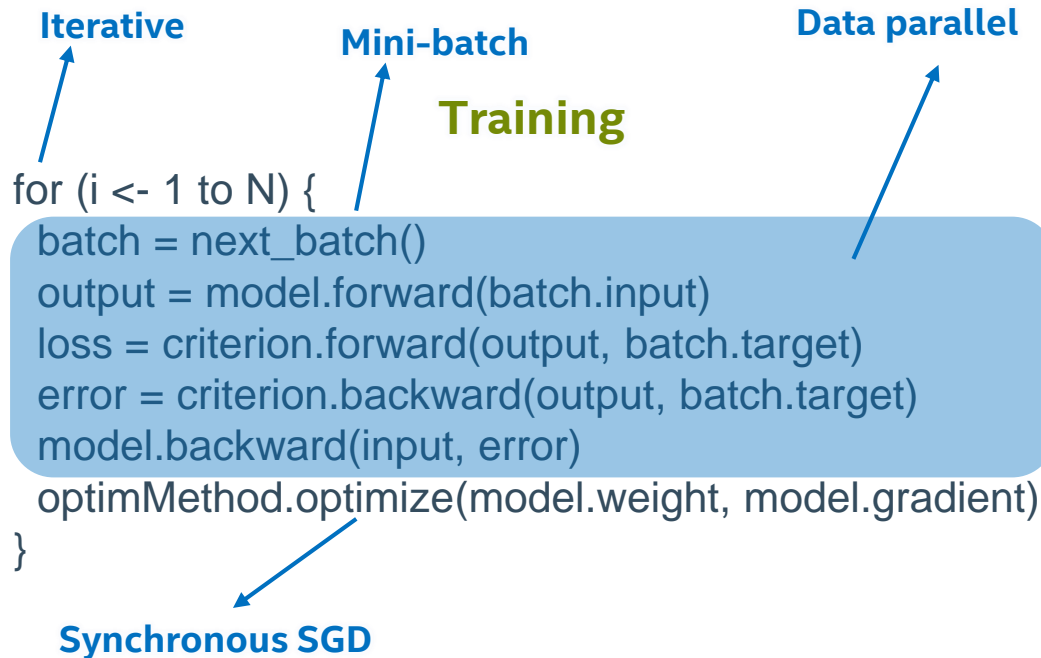
FRAUD DETECTION FOR UNIONPAY



https://mp.weixin.qq.com/s?__biz=MzI3NDAwNDUwNg==&mid=2648307335&idx=1&sn=8eb9f63eaf2e40e24a90601b9cc03d1f

DISTRIBUTED TRAINING ON BIGDL

DISTRIBUTED TRAINING ON BIGDL



RUN AS STANDARD SPARK PROGRAM

Standard Spark jobs

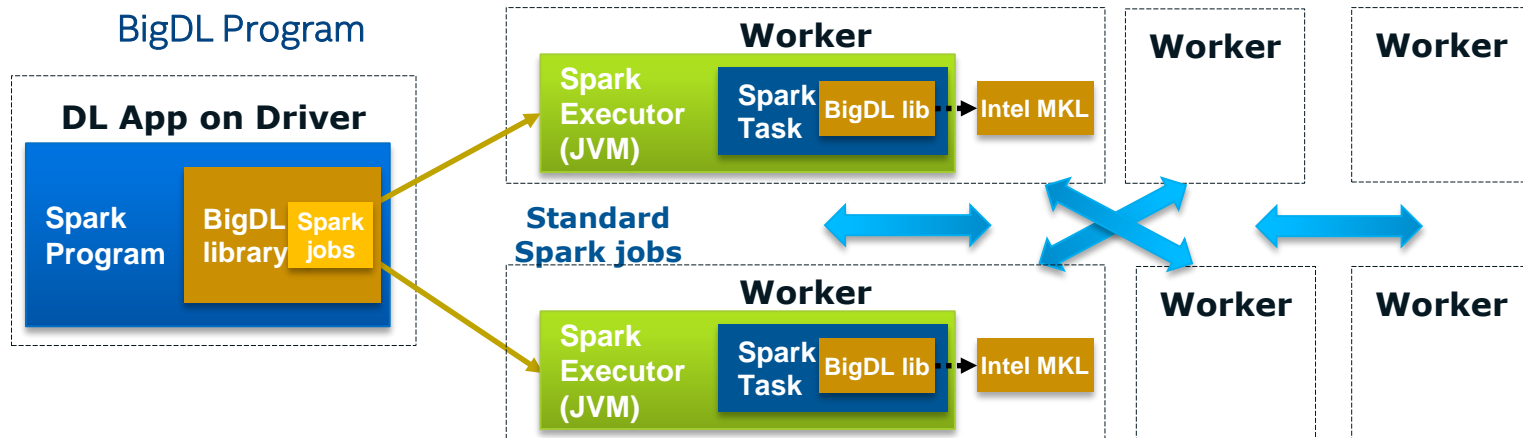
- No changes to the Spark or Hadoop clusters needed

Iterative

- Each iteration of the training runs as a Spark job

Data parallel

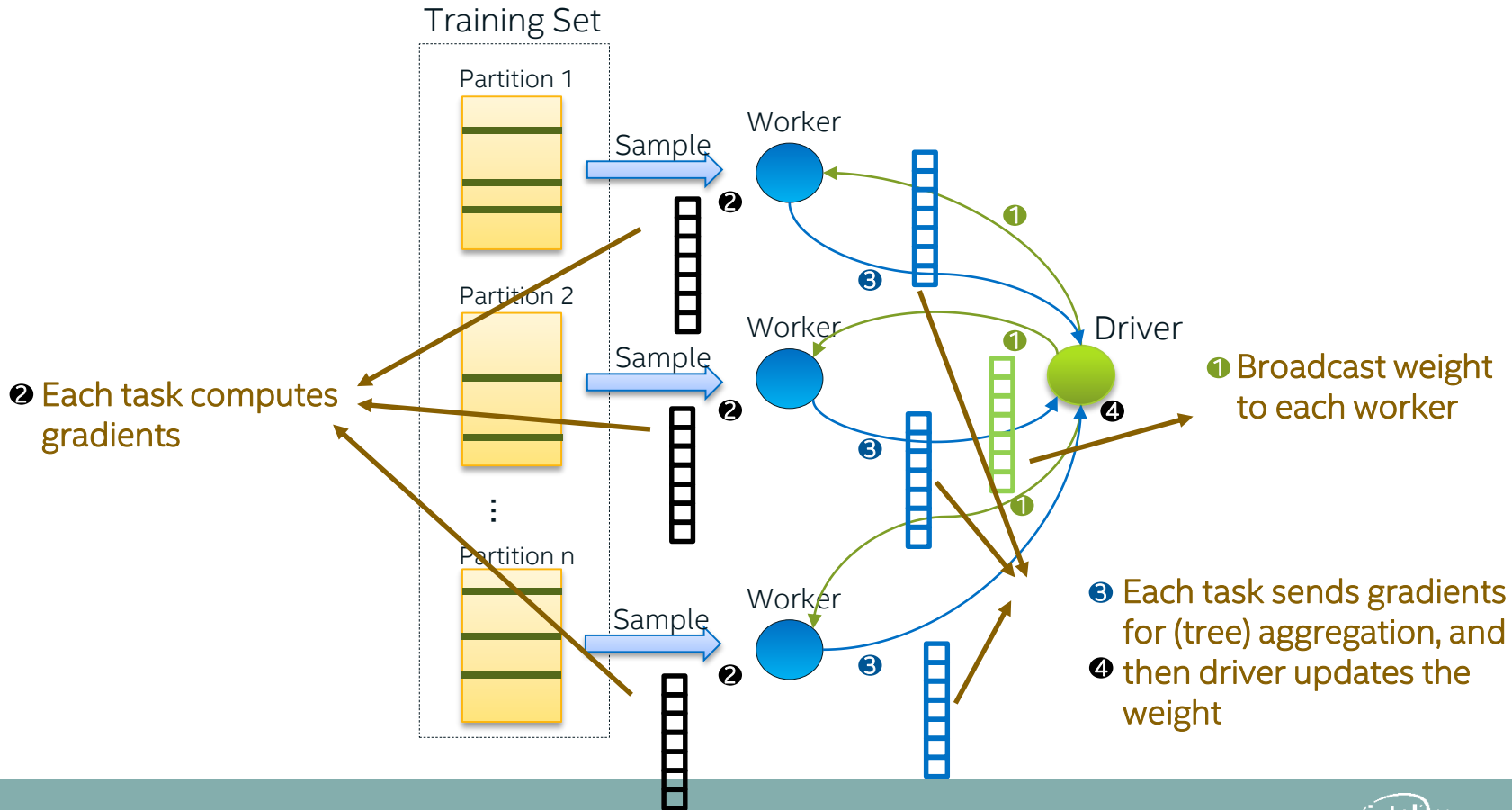
- Each Spark task runs the same model on a subset of the data (batch)



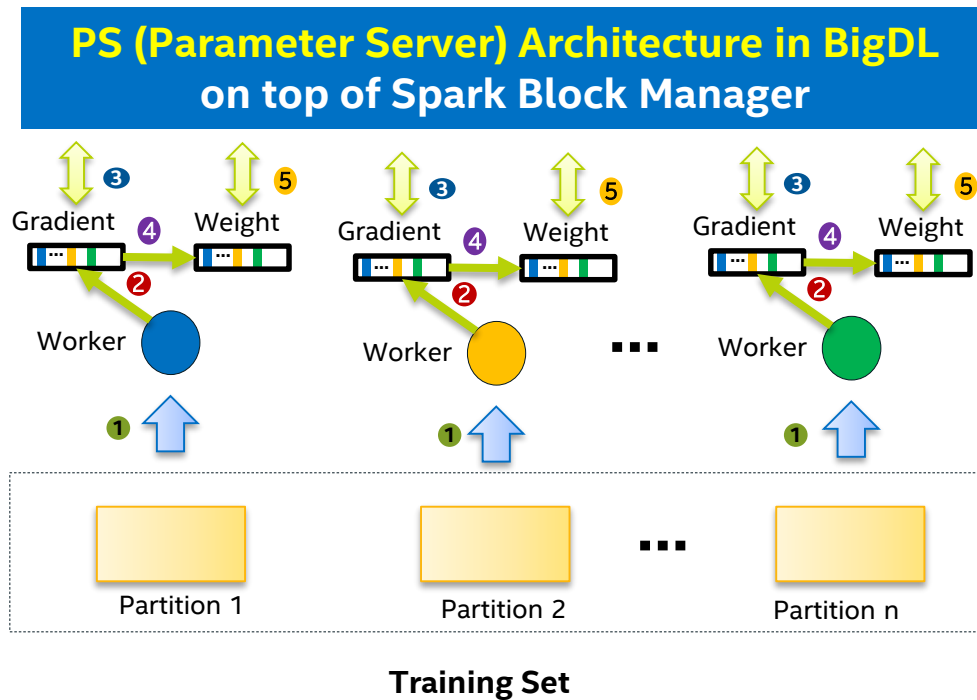
TECHNIQUES FOR LARGE-SCALE DISTRIBUTED TRAINING

- **Optimizing parameter synchronization and aggregation**
- **Optimizing task scheduling**
- **Scaling batch size**

PARAMETER SYNCHRONIZATION IN SPARK MLLIB

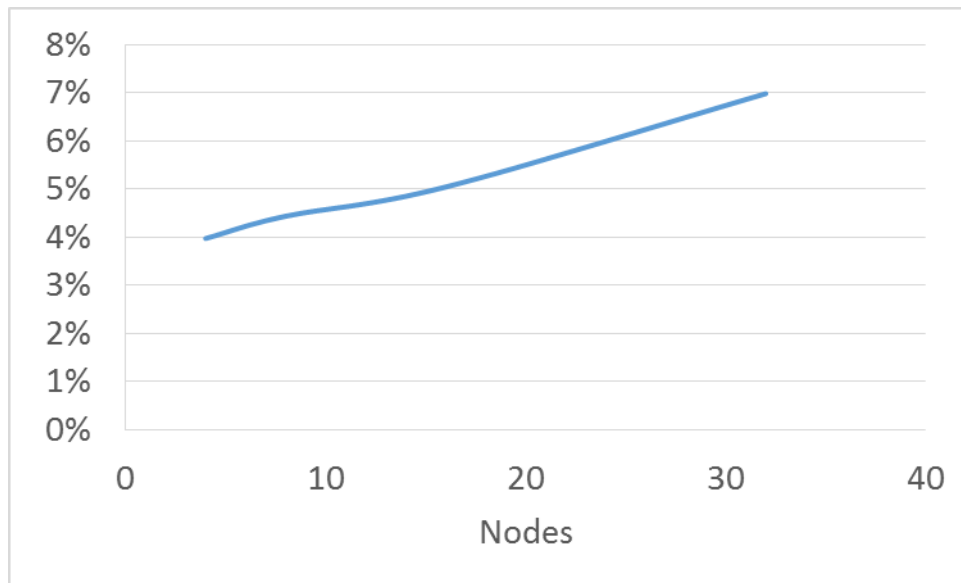


PARAMETER SYNCHRONIZATION IN BIGDL



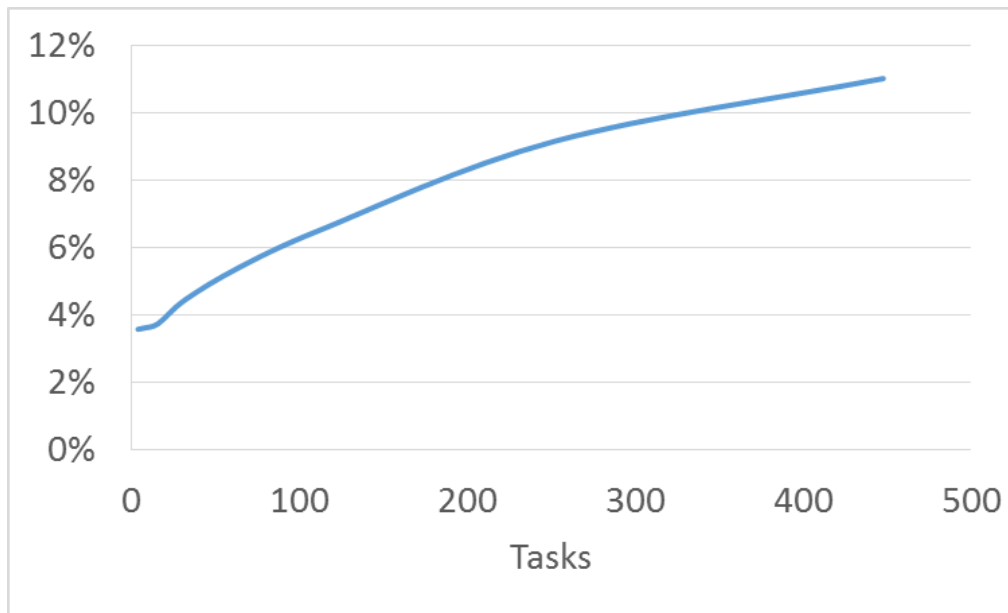
Peer-2-Peer **All-Reduce** synchronization

PARAMETER SYNCHRONIZATION IN BIGDL



Parameter synchronization time as a fraction of average compute time for Inception v1 training

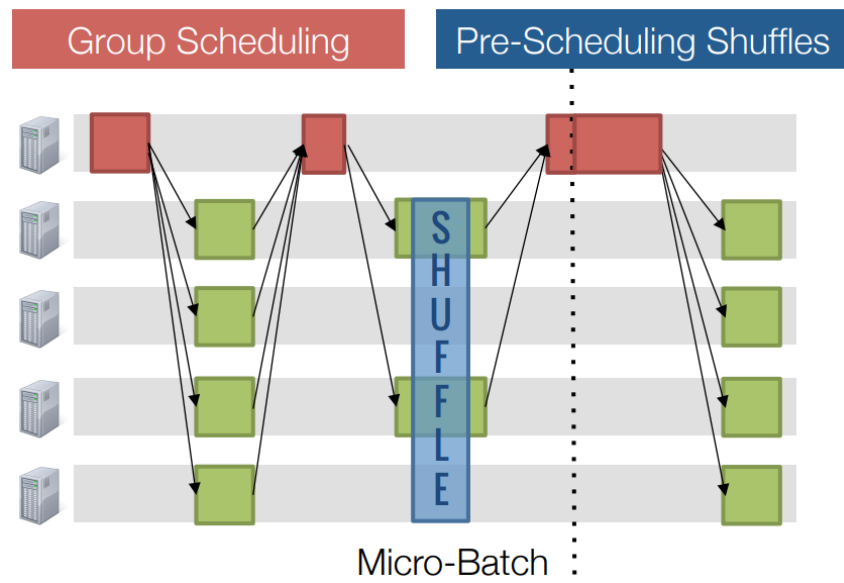
TASK SCHEDULING OVERHEADS



Spark overheads (task scheduling, task serde, task fetch) as a fraction of average compute time for Inception v1 training

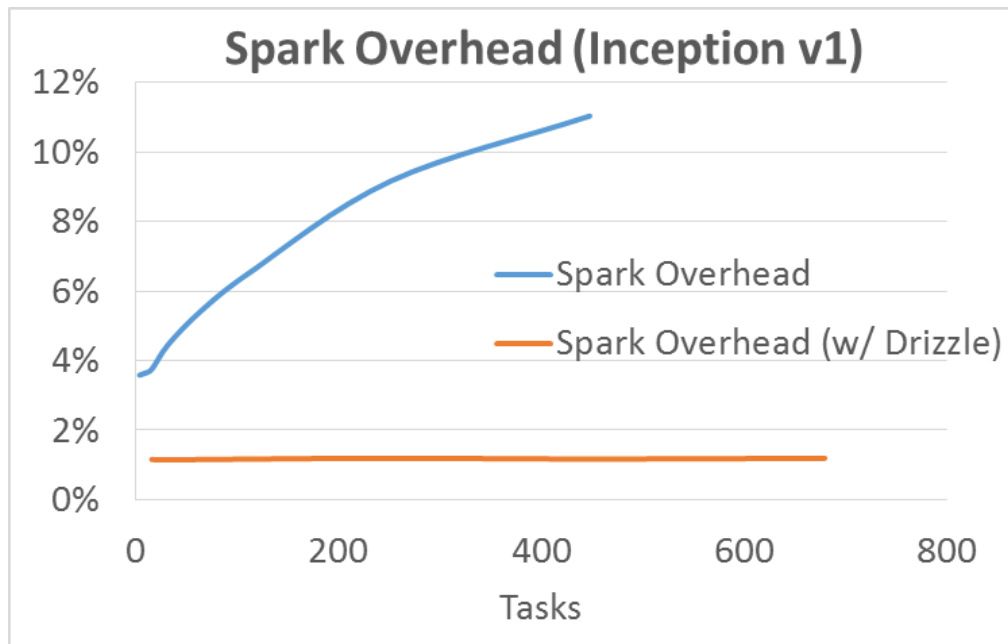
DRIZZLE

- Low latency execution engine for Apache Spark
- Fine-grained execution with coarse-grained scheduling
- Group scheduling
 - Schedule a group of iterations at once
 - Fault tolerance, scheduling at group boundaries
- Coordinating shuffles: PRE-SCHEDULING
 - Pre-schedule tasks on executors
 - Trigger tasks once dependencies are met



*Collaboration with Shivaram Venkataraman from UC Berkeley RICELab

REDUCING SCHEDULING OVERHEADS WITH DRIZZLE



INCREASED MINI-BATCH SIZE

- Distributed synchronous mini-batch SGD
 - Increased mini-batch size
$$total_batch_size = batch_size_per_worker * num_of_workers$$
 - Can lead to loss in test accuracy
- State-of-art method for scaling mini-batch size*
 - Linear scaling rule
 - Warm-up strategy
 - Layer-wise adaptive rate scaling
 - Adding batch normalization

* “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour”

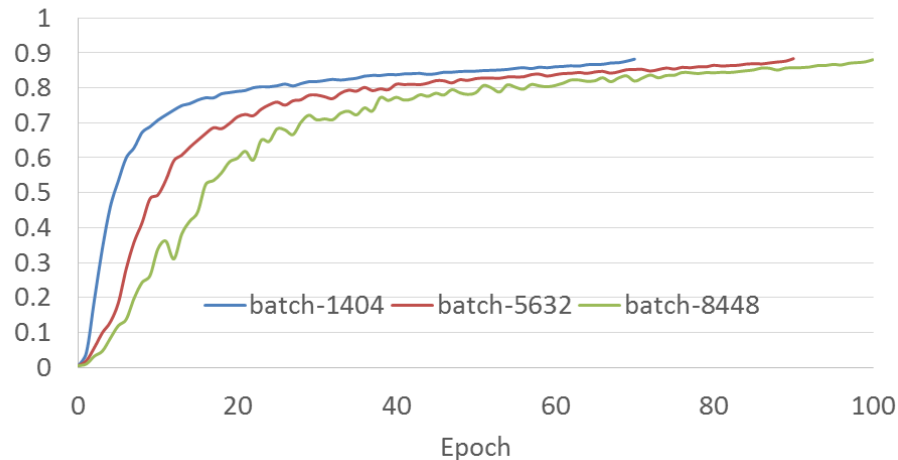
* “Scaling SGD Batch Size to 32K for ImageNet Training”

INCEPTION V1

Top1 accuracy



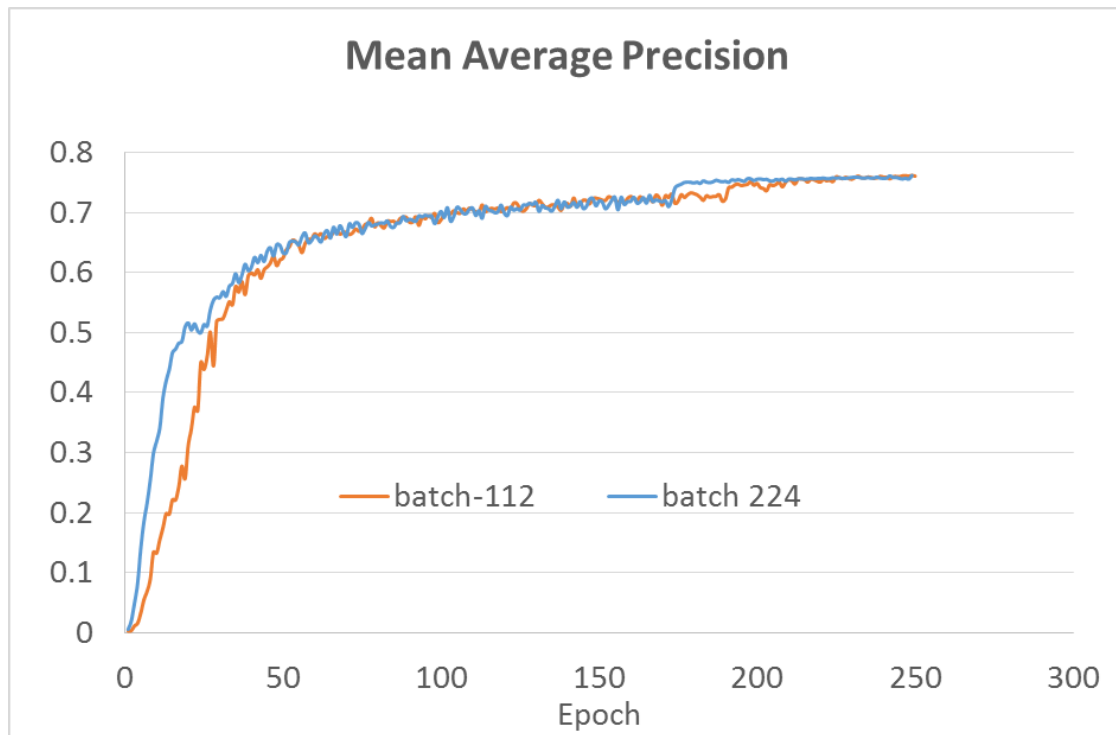
Top5 Accuracy



Strategies

- Gradual warm-up
- Linear scaling
- *Gradient clipping*
- TODO: adding batch normalization

SSD (SINGLE SHOT MULTI-BOX DETECTOR)



Strategies

- Warm-up w/ Adam
- Linear scaling
- *Gradient clipping*

DISTRIBUTED INFERENCE ON BIGDL

DISTRIBUTED INFERENCE ON BIGDL

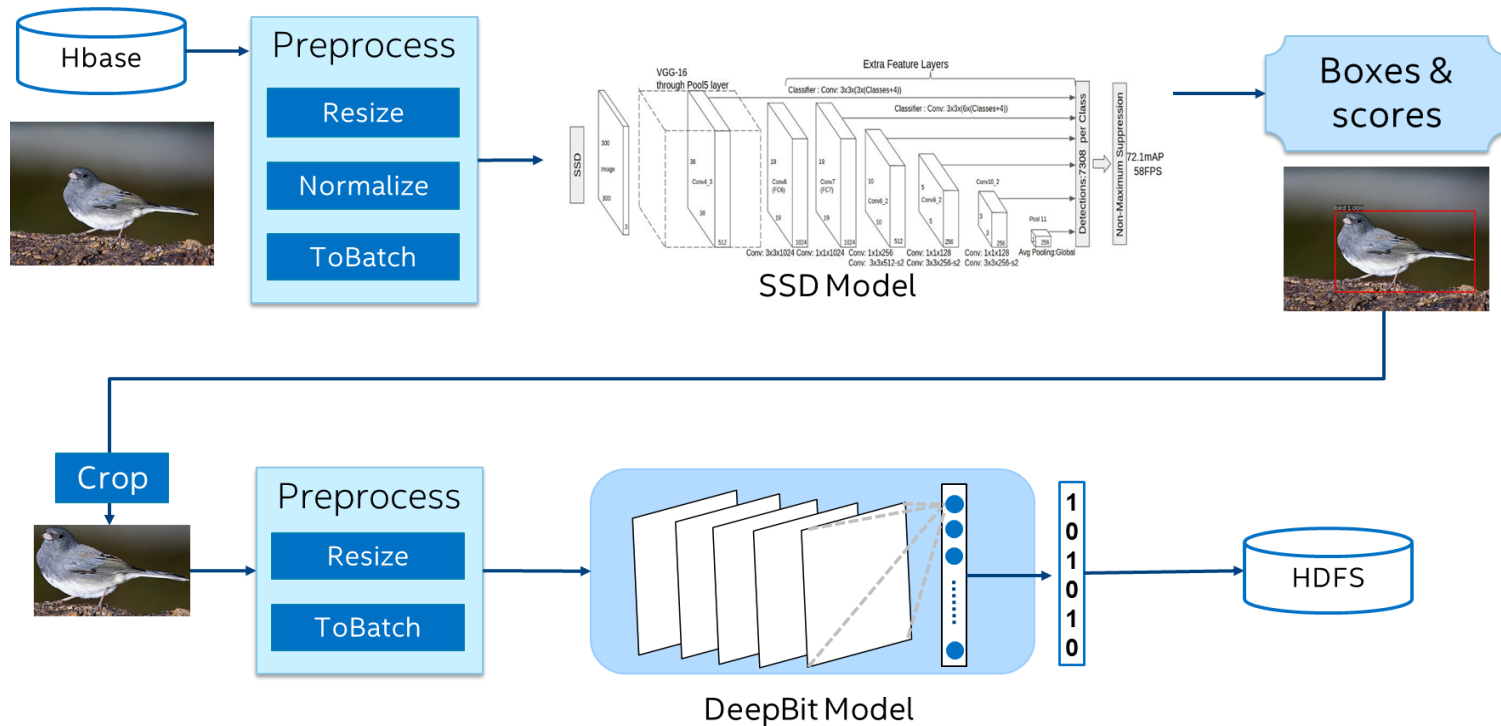
Inference

```
for (b <- 1 to D) {  
  input = next_data(i)  
  output = model.forward(input)  
}
```

Embarrassingly (data) parallel in nature

Efficiently scale out using BigDL on
Apache Spark

IMAGE DETECTION & EXTRACTION PIPELINE (USING SSD + DEEPBIT MODELS)



CHALLENGES OF LARGE-SCALE PROCESSING IN GPU SOLUTIONS

- Reading images out takes a very long time
- Image pre-processing on HBase is very complex
- No existing software frameworks can be leveraged
 - E.g., resource management, distributed data processing, fault tolerance, etc.
- Very challenging to scale out to massive amount of pictures
 - Due to SW and HW infrastructure constraints

UPGRADING TO **BIGDL** SOLUTIONS

- Reuse existing Hadoop/Spark clusters for deep learning with no changes
- Efficiently scale out on Spark with superior performance
 - Reading HBase data no longer a bottleneck
- Very easy to build the end-to-end pipeline in BigDL
 - Image transformation and augmentation based on OpenCV on Spark

```
val preProcessor = BytesToMat() -> Resize(300, 300) -> ...  
val transformed = preProcessor(dataRdd)
```
 - Directly Load pre-trained models (BigDL/Caffe/Torch/TensorFlow) into BigDL

```
val model = Module.loadCaffeModel(caffeDefPath, caffeModelPath)
```


MODEL QUANTIZATION FOR EFFICIENT INFERENCE IN BIGDL

- Local quantization scheme converting floats to integers
 - Faster compute and smaller models
 - Take advantage of SSE and AVX instructions on Xeon servers
 - Supports pre-trained models (BigDL/Caffe/Torch/TensorFlow)
val model = Module.loadCaffeModel(caffeDefPath, caffeModelPath)
val quantizedModel = model.quantize()
- Quantized SSD model
 - ~4x model size reduction
 - >2x inference speedup
 - ~0.001 mAP (mean average precision) loss

TRY BIGDL OUT

Running BigDL, Deep Learning
for Apache Spark, on AWS*
(Amazon* Web Service)

<https://aws.amazon.com/blogs/ai/running-bigdl-deep-learning-for-apache-spark-on-aws/>

Use BigDL on Microsoft*
Azure* HDInsight*

<https://azure.microsoft.com/en-us/blog/use-bigdl-on-hdinsight-spark-for-distributed-deep-learning/>

BigDL on Alibaba* Cloud
E-MapReduce*

<https://yq.aliyun.com/articles/73347>

BigDL on CDH* and Cloudera*
Data Science Workbench*

<http://blog.cloudera.com/blog/2017/04/bigdl-on-cdh-and-cloudera-data-science-workbench/>

Intel's BigDL on Databricks*

<https://databricks.com/blog/2017/02/09/intels-bigdl-databricks.html>

BigDL Distribution in Cray
Urika-XC Analytics Suite

<http://www.cray.com/products/analytics/urika-xc>

PARTNER WITH US



- **Use BigDL & Share your Experience**
- **Use Intel Optimized Libraries & Frameworks**
- **Leverage Intel Developer Zone Resources**

<https://github.com/intel-analytics/BigDL>

<http://software.intel.com/ai>

LEGAL NOTICES & DISCLAIMERS

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

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>.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, Pentium, Celeron, Atom, Core, Xeon and others are trademarks of Intel Corporation in the U.S. and/or other countries.

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

© 2016 Intel Corporation.