DLoBD: An Emerging Paradigm of Deep Learning over Big Data Stacks on RDMA-enabled Clusters

Talk at OSU Booth at SC 2018

by

Xiaoyi Lu
The Ohio State University
E-mail: luxi@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~luxi
Why Deep Learning is so hot?

- **Deep Learning** is a sub-set of Machine Learning
  - But, it is perhaps the most radical and revolutionary subset

- Deep Learning is going through a resurgence
  - **Model**: Excellent accuracy for deep/convolutional neural networks
  - **Data**: Public availability of versatile datasets like MNIST, CIFAR, and ImageNet
  - **Capability**: Unprecedented computing and communication capabilities: Multi-/Many-Core, GPGPUs, Xeon Phi, InfiniBand, RoCE, etc.

- **Big Data** has become one of the most important elements in business analytics
  - Increasing demand for getting **Big Value** out of Big Data to drive the revenue continuously growing
Application Example of DL: Flickr’s Magic View Photo Filtering

- Image recognition to divide pictures into surprisingly accurate categories
- Magic of AI/DL: Generate accurate tags for billions of pictures

Courtesy: https://thenextweb.com/opinion/2015/05/22/flickrs-new-magic-view-photo-filtering-feature-works-so-well-it-convinced-me-to-ditch-iphone/\#.tnw_RaZEaD6g

Network Based Computing Laboratory
OSU Booth at SC 2018
Deep Learning over Big Data (DLoBD)

- Deep Learning over Big Data (DLoBD) is one of the most efficient analyzing paradigms
- More and more deep learning tools or libraries (e.g., Caffe, TensorFlow) start running over big data stacks, such as Apache Hadoop and Spark
- **Benefits** of the DLoBD approach
  - Easily build a powerful data analytics pipeline
  - Better data locality
  - Efficient resource sharing and cost effective
Examples of DLoBD Stacks

- CaffeOnSpark
- SparkNet
- TensorFlowOnSpark
- TensorFlow
- DeepLearning4J
- BigDL
- mmlspark
  - CNTKOnSpark
- Many others...
Overview of DLoBD Stacks

- Layers of DLoBD Stacks
  - Deep learning application layer
  - Deep learning library layer
  - Big data analytics framework layer
  - Resource scheduler layer
  - Distributed file system layer
  - Hardware resource layer
- How much performance benefit we can achieve for end deep learning applications?

Deep Learning Models & Algorithms on Big Data Sets
- Deep Learning Libraries (e.g., Caffe, TensorFlow, BigDL, etc.)
- Big Data Frameworks (e.g., Hadoop, Spark, etc.)
- Resource Scheduler (e.g., YARN, Mesos, etc.)
- Distributed File Systems (e.g., HDFS, Ceph, OrangeFS, etc.)
- High-Performance Networks and Computing Units

Sub-optimal Performance?
Increasing Usage of HPC, Big Data and Deep Learning

Convergence of HPC, Big Data, and Deep Learning!!!
Highly-Optimized Underlying Libraries with HPC Technologies

• BLAS Libraries – the heart of math operations
  – Atlas/OpenBLAS
  – NVIDIA cuBlas
  – Intel Math Kernel Library (MKL)

• DNN Libraries – the heart of Convolutions!
  – NVIDIA cuDNN (already reached its 7th iteration – cudnn-v7)
  – Intel MKL-DNN (MKL 2017) – recent but a very promising development

• Communication Libraries – the heart of model parameter updating
  – RDMA
  – GPUDirect RDMA
Outline

• Accelerating Big Data Stacks

• Benchmarking and Characterizing DLoBD Stacks
  – CaffeOnSpark, TensorFlowOnSpark, MMLSpark, and BigDL

• Accelerating DLoBD Stacks
  – BigDL on RDMA-Spark
  – TensorFlow
Design Overview of Spark with RDMA

- **Design Features**
  - RDMA based shuffle plugin
  - SEDA-based architecture
  - Dynamic connection management and sharing
  - Non-blocking data transfer
  - Off-JVM-heap buffer management
  - InfiniBand/RoCE support

- Enables high performance RDMA communication, while supporting traditional socket interface
- JNI Layer bridges Scala based Spark with communication library written in native code


RDMA for Apache Spark Distribution

• High-Performance Design of Spark over RDMA-enabled Interconnects
  – High performance RDMA-enhanced design with native InfiniBand and RoCE support at the verbs-level for Spark
  – RDMA-based data shuffle and SEDA-based shuffle architecture
  – Non-blocking and chunk-based data transfer
  – Off-JVM-heap buffer management
  – Support for OpenPOWER
  – Easily configurable for different protocols (native InfiniBand, RoCE, and IPoIB)

• Current release: 0.9.5
  – Based on Apache Spark 2.1.0
  – Tested with
    • Mellanox InfiniBand adapters (DDR, QDR, FDR, and EDR)
    • RoCE support with Mellanox adapters
    • Various multi-core platforms (x86, POWER)
    • RAM disks, SSDs, and HDD
    – http://hibd.cse.ohio-state.edu
HiBD Packages on SDSC Comet and Chameleon Cloud

• RDMA for Apache Hadoop 2.x and RDMA for Apache Spark are installed and available on SDSC Comet.
  – Examples for various modes of usage are available in:
    • RDMA for Apache Hadoop 2.x: /share/apps/examples/HADOOP
    • RDMA for Apache Spark: /share/apps/examples/SPARK/
  – Please email help@xsede.org (reference Comet as the machine, and SDSC as the site) if you have any further questions about usage and configuration.

• RDMA for Apache Hadoop is also available on Chameleon Cloud as an appliance
  – https://www.chameleoncloud.org/appliances/17/

M. Tatineni, X. Lu, D. J. Choi, A. Majumdar, and D. K. Panda, Experiences and Benefits of Running RDMA Hadoop and Spark on SDSC Comet, XSEDE’16, July 2016
Network Based Computing Laboratory

OSU Booth at SC 2018

Performance Evaluation on SDSC Comet – HiBench PageRank

32 Worker Nodes, 768 cores, PageRank Total Time

64 Worker Nodes, 1536 cores, PageRank Total Time

- InfiniBand FDR, SSD, 32/64 Worker Nodes, 768/1536 Cores, (768/1536M 768/1536R)
- RDMA-based design for Spark 1.5.1
- RDMA vs. IPoIB with 768/1536 concurrent tasks, single SSD per node.
  - 32 nodes/768 cores: Total time reduced by 37% over IPoIB (56Gbps)
  - 64 nodes/1536 cores: Total time reduced by 43% over IPoIB (56Gbps)
Outline

• Accelerating Big Data Stacks

• Benchmarking and Characterizing DLoBD Stacks
  – CaffeOnSpark, TensorFlowOnSpark, MMLSpark, and BigDL

• Accelerating DLoBD Stacks
  – BigDL on RDMA-Spark
  – TensorFlow
Benchmarking and Characterization Methodology

- Choose proper DL workloads, models and datasets
  - Varied sizes to cover big and small models.
  - Small and large data sets
  - Cover different kinds of combinations
- Choose representative DLoBD stacks
  - CaffeOnSpark, TensorFlowOnSpark, and BigDL
  - Running over Spark, Yarn, HDFS

- Define characterization dimensions
  - Processor Type
  - Parameter updating approach (i.e., communication)
  - Network Protocol (IPoIB, RDMA)

- Generate evaluation reports
  - Performance (End-to-end training time; time to a certain accuracy; epoch execution time)
  - Accuracy, Scalability, Resource Utilization
  - Breakdown
Overview of Representative DLoBD Stacks - CaffeOnSpark

- **Spark Driver**: Job Launching and Job Control
- **Spark Executor**: For data feeding and task control
- **Model Synchronizer**: Communicates across nodes with RDMA / TCP, and output model file on HDFS
- **Scalable and Communication intensive**
  - Server-to-server direct communication (Ethernet or InfiniBand) achieves faster learning and eliminates scalability bottleneck
  - Out-of-band communication
Overview of Representative DLoBD Stacks - TensorFlowOnSpark

- Spark Executors acting as containers used to run TensorFlow code
- Two different modes to ingesting data
  - Read data directly from HDFS using built-in TensorFlow modules
  - Feeding data from Spark RDDs to Spark executors (TensorFlow core)
- Scalable and Communication intensive
  - Parameter Server-based approach
  - Embedded inside one Spark executor and talk to other workers over gRPC or gPRC with RDMA
  - Out-of-band communication
Overview of Representative DLoBD Stacks – CNTKOnSpark/MMLSpark

- Microsoft Cognitive Toolkit (CNTK) and OpenCV into Spark Machine Learning pipelines without data transfer overhead
- Feeding data for CNTK Core (e.g. images or texts) can be directly read from HDFS by Spark Executors
- Scalable and Communication intensive
  - Embedded inside one Spark executor and talk to other workers over MPI (RDMA, TCP)
  - Out-of-band communication
Overview of Representative DLoBD Stacks - BigDL

- Users can write deep learning applications as Spark programs
- Users can load pre-trained Caffe or Torch models into Spark programs using BigDL
- Feed data to BigDL core by Spark Executor which can directly load data from HDFS
- High performance
  - Support Intel MKL
  - Support both Xeon and Xeon Phi (e.g., KNL)
- Scalable and Communication intensive
  - Spark block manager as parameter server
  - Organically designed and integrated with Spark architecture
  - In-band Communication
- RDMA communication can be achieved through our RDMA-Spark package!
# Selected Various Datasets and Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Layers (Conv. / Full-connected)</th>
<th>Dataset</th>
<th>Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet</td>
<td>2 / 2</td>
<td>MNIST</td>
<td>CaffeOnSpark, TensorFlowOnSpark</td>
</tr>
<tr>
<td>SoftMax Regression</td>
<td>NA / NA</td>
<td>MNIST</td>
<td>TensorFlowOnSpark</td>
</tr>
<tr>
<td>CIFAR-10 Quick</td>
<td>3 / 1</td>
<td>CIFAR-10</td>
<td>CaffeOnSpark, TensorFlowOnSpark, MMLSpark</td>
</tr>
<tr>
<td>VGG-16</td>
<td>13 / 3</td>
<td>CIFAR-10</td>
<td>BigDL</td>
</tr>
<tr>
<td>AlexNet</td>
<td>5 / 3</td>
<td>ImageNet</td>
<td>CaffeOnSpark</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>22 / 0</td>
<td>ImageNet</td>
<td>CaffeOnSpark</td>
</tr>
<tr>
<td>Resnet-50</td>
<td>53 / 1</td>
<td>Synthetic</td>
<td>TensorFlow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MNIST</th>
<th>CIFAR-10</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Digit Classification</td>
<td>Object Classification</td>
</tr>
<tr>
<td>Resolution</td>
<td>28 × 28 B&amp;W</td>
<td>32 × 32 Color</td>
</tr>
<tr>
<td>Classes</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Training Images</td>
<td>60 K</td>
<td>50 K</td>
</tr>
<tr>
<td>Testing Images</td>
<td>10 K</td>
<td>10 K</td>
</tr>
</tbody>
</table>
Performance Characterization for CPU-/GPU-based Deep Learning with CaffeOnSpark

• DL workloads can benefit from the high performance of the DLoBD stacks.
• Network will become a bottleneck at some point if the sub-optimal IPoIB network protocol is used.
• GPU/GPU+cuDNN can get the best performance. GPU + cuDNN is degraded at a large scale (e.g., 16 nodes).
• For some models, solutions with CPU + MKL may outperform GPU-based solutions.
Performance Characterization for IPoIB and RDMA with CaffeOnSpark and TensorFlowOnSpark (IB EDR)

- CaffeOnSpark benefits from the high performance of RDMA compared to IPoIB once communication overhead becomes significant.
- Our experiments show that the default RDMA design in TensorFlowOnSpark is not fully optimized yet. For MNIST tests, RDMA is not showing obvious benefits.

<table>
<thead>
<tr>
<th></th>
<th>End-to-End Time (secs)</th>
<th>CaffeOnSpark</th>
<th>TensorFlowOnSpark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPoIB</td>
<td>GPU</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RDMA</td>
<td>GPU</td>
<td>14.2%</td>
<td>4.9%</td>
</tr>
<tr>
<td>IPoIB</td>
<td>GPU-cuDNN</td>
<td>13.3%</td>
<td></td>
</tr>
<tr>
<td>RDMA</td>
<td>GPU</td>
<td>51.2%</td>
<td></td>
</tr>
<tr>
<td>IPoIB</td>
<td>GPU-cuDNN</td>
<td>45.6%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CIFAR-10 Quick</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPoIB</td>
<td>LeNet on MNIST</td>
<td>33.01%</td>
<td></td>
</tr>
<tr>
<td>RDMA</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- CIFAR10 (2 GPUs/node)
- MNIST (1 GPU/node)
Performance Characterization with MMLSpark

- The solution of GPU + cuDNN performs best, up to 55x faster than CPU + OpenBLAS, and up to 15x than CPU + MKL.
- OpenMPI-based communication over IPoIB and RDMA; Similar performance; The latency and bandwidth of IPoIB in this cluster are sufficient for small models.
- Could not find other benchmarks with bigger models for MMLSpark
Characterization on Performance and Accuracy

• Performance Evaluation of **CaffeOnSpark** (training time to achieve a 70% accuracy)
  – RDMA reduces the overall time cost by **22%** in training AlexNet on ImageNet
  – RDMA reduces the overall time cost by **15%** in training GoogleNet on ImageNet

• Performance Evaluation of **BigDL** (training time to achieve a 70% accuracy)
  – RDMA reduces the overall time cost by **48%** in training VGG on CIFAR-10
- CIFAR-10 Quick Model and CIFAR-10 Dataset
- GPU-based solutions use less memory than CPU-based ones as they mostly use GPU memory.
- CPU + MKL solution uses host memory more efficiently and has better performance than CPU + OpenBLAS.
- RDMA utilizes the network resources more efficiently than the IPoIB in CaffeOnSpark.
- CaffeOnSpark still does not fully utilize the high throughput characteristic of RDMA and memory resource.
Performance Overhead across Layers in DLoBD Stacks

- SoftMax Regression model, over MNIST dataset
- Up to 15.5% time in Apache Hadoop YARN scheduler layer
- Up to 18.1% execution time in Spark job execution layer
- Data size is small, so we do not count the time spent on accessing HDFS layer.
- Need more effort to reduce the overhead across different layers of DLoBD stacks
- Maybe amortized in long-running deep learning jobs
Insights and Guidance

- RDMA can benefit DL workloads
  - Up to $2.7x$ speedup with RDMA compared to the IPoIB scheme for deep learning workloads.
  - RDMA can scale better and utilize resources more efficiently than IPoIB over InfiniBand clusters.
- GPU-based DL designs can outperform CPU-based designs, but not always
  - LeNet on MNIST, CPU + MKL achieved better performance than GPU and GPU + cuDNN on 8/16 nodes.
- Large rooms for further improvement in DLoBD stacks!!!
- **We need more benchmarks, public datasets, and analysis tools!!!**


Outline

- Accelerating Big Data Stacks
- Benchmarking and Characterizing DLoBD Stacks
  - CaffeOnSpark, TensorFlowOnSpark, MMLSpark, and BigDL
- Accelerating DLoBD Stacks
  - BigDL on RDMA-Spark
  - TensorFlow
Epoch-Level Evaluation with BigDL on SDSC Comet

- Epoch-level evaluation of training VGG model using BigDL on default Spark with IPoIB and our RDMA-based Spark.
- RDMA version takes constantly less time than the IPoIB version to finish every epoch.
  - RDMA finishes epoch 18 in 2.6x time faster than IPoIB
Scalability Evaluation with BigDL on SDSC Comet

- Using BigDL with IPoIB & RDMA Spark
- For VGG model trained with BigDL, RDMA-based Spark scales better than default IPoIB Spark
- For 384 CPU cores, 18 epochs and same batch size, RDMA takes about 870 seconds while IPoIB takes 2,372 seconds
- A speedup of 2.7x using RDMA for the epoch-level training time
Outline

- Accelerating Big Data Stacks
- Benchmarking and Characterizing DLoBD Stacks
  - CaffeOnSpark, TensorFlowOnSpark, MMLSpark, and BigDL
- Accelerating DLoBD Stacks
  - BigDL on RDMA-Spark
  - TensorFlow
Overview of gRPC with TensorFlow

Worker services communicate among each other using gRPC, or gRPC+X!
Performance Benefits for RDMA-gRPC with Micro-Benchmark

- **gRPC-RDMA Latency on SDSC-Comet-FDR**
  - **Up to 2.7x** performance speedup over IPoIB for Latency for small messages
  - **Up to 2.8x** performance speedup over IPoIB for Latency for medium messages
  - **Up to 2.5x** performance speedup over IPoIB for Latency for large messages

Performance Benefit for TensorFlow - Resnet50

Network Based Computing Laboratory

- TensorFlow Resnet50 performance evaluation on an IB EDR cluster
  - Up to 35% performance speedup over IPoIB for 4 nodes.
  - Up to 41% performance speedup over IPoIB for 8 nodes.
Performance Benefit for TensorFlow - Inception3

TensorFlow Inception3 performance evaluation on an IB EDR cluster
- Up to 27% performance speedup over IPoIB for 4 nodes
- Up to 36% performance speedup over IPoIB for 8 nodes.
Concluding Remarks

• Discussed challenges in benchmarking, characterizing, and accelerating Deep Learning over Big Data (DLoBD) stacks

• RDMA can benefit DL workloads as showed by our RDMA-Spark, AR-gRPC, and other RDMA designs

• Many other open issues need to be solved

• Will enable Big Data and Deep Learning community to take advantage of modern HPC technologies to carry out their analytics in a fast and scalable manner
Thank You!

luxi@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~luxi

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/