Best Practices for Deep Learning on Apache Spark

Tim Hunter (speaker)
Joseph K. Bradley

May 10th, 2017
GPU Technology Conference
About Me

- Tim Hunter
- Software engineer @ Databricks
- Ph.D. from UC Berkeley in Machine Learning
- Very early Spark user
- Contributor to MLlib
- Author of TensorFrames and GraphFrames
Founded by the creators of Apache Spark in 2013 to make big data simple

Provides hosted Spark platform in the cloud
Deep Learning and Apache Spark

Deep Learning frameworks w/ Spark bindings
- Caffe (CaffeOnSpark)
- Keras (Elephas)
- mxnet
- Paddle
- TensorFlow (TensorFlow on Spark, TensorFlow on Spark, TensorFlow on Spark, TensorFlow on Spark, TensorFlow on Spark)

Native Spark
- BigDL
- DeepDist
- DeepLearning4J
- MLlib
- SparkCL
- SparkNet

Extensions to Spark for specialized hardware
- Blaze (UCLA & Falcon Computing Solutions)
- IBM Conductor with Spark
Deep Learning and Apache Spark

2016: the year of emerging solutions for Spark + Deep Learning

No consensus

- Many approaches for libraries: integrate existing ones with Spark, build on top of Spark, modify Spark itself
- Official Spark MLlib support is limited (perceptron-like networks)
One Framework to Rule Them All?

Should we look for The One Deep Learning Framework?
Databricks’ perspective

- Databricks: hosted Spark platform on public cloud
- GPUs for compute-intensive workloads
- Customers use many Deep Learning frameworks: TensorFlow, MXNet, BigDL, Theano, Caffe, and more

This talk

- Lessons learned from supporting *many* Deep Learning frameworks
- Multiple ways to integrate Deep Learning & Spark
- Best practices for these integrations
Outline

• Deep Learning in data pipelines
• Recurring patterns in Spark + Deep Learning integrations
• Developer tips
• Monitoring
Outline

• Deep Learning in data pipelines
• Recurring patterns in Spark + Deep Learning integrations
• Developer tips
• Monitoring
ML is a small part of data 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., NIPS 2016
DL in a data pipeline: Training

- Data collection
- ETL
- Featurization
- Deep Learning
- Validation
- Export, Serving

- Large cluster: High memory/CPU ratio
- Small cluster: Low memory/CPU ratio
DL in a data pipeline: Transformation

Specialized data transforms: feature extraction & prediction

Input

Output

dog
dog
cat

Saulius Garalevicius - CC BY-SA 3.0
Outline

• Deep Learning in data pipelines
• Recurring patterns in Spark + Deep Learning integrations
• Developer tips
• Monitoring
Recurring patterns

Spark as a scheduler
  • Data-parallel tasks
  • Data stored outside Spark

Embedded Deep Learning transforms
  • Data-parallel tasks
  • Data stored in DataFrames/RDDs

Cooperative frameworks
  • Multiple passes over data
  • Heavy and/or specialized communication
Streaming data through DL

Primary storage choices:
- Cold layer (HDFS/S3/etc.)
- Local storage: files, Spark’s on-disk persistence layer
- In memory: Spark RDDs or Spark DataFrames

Find out if you are I/O constrained or processor-constrained
- How big is your dataset? MNIST or ImageNet?

If using PySpark:
- All frameworks heavily optimized for disk I/O
- Use Spark’s broadcast for small datasets that fit in memory
- Reading files is fast: use local files when it does not fit
Cooperative frameworks

- Use Spark for data input
- Examples:
  - IBM GPU efforts
  - Skymind’s DeepLearning4J
  - DistML and other Parameter Server efforts
Cooperative frameworks

• Bypass Spark for asynchronous / specific communication patterns across machines

• Lose benefit of RDDs and DataFrames and reproducibility/determinism

• But these guarantees are not requested anyway when doing deep learning (stochastic gradient)

• “reproducibility is worth a factor of 2” (Leon Bottou, quoted by John Langford)
Outline

• Deep Learning in data pipelines
• Recurring patterns in Spark + Deep Learning integrations
• Developer tips
• Monitoring
The GPU software stack

• Deep Learning commonly used with GPUs
• A lot of work on Spark dependencies:
  • Few dependencies on local machine when compiling Spark
  • The build process works well in a large number of configurations (just scala + maven)
• GPUs present challenges: CUDA, support libraries, drivers, etc.
  • Deep software stack, requires careful construction (hardware + drivers + CUDA + libraries)
  • All these are expected by the user
  • Turnkey stacks just starting to appear
The GPU software stack

- Provide a Docker image with all the GPU SDK
- Pre-install GPU drivers on the instance

<table>
<thead>
<tr>
<th>Python / JVM clients</th>
<th>Container: nvidia-docker, lxc, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep learning libraries (Tensorflow, etc.)</td>
<td>JCUDA</td>
</tr>
<tr>
<td>CuBLAS</td>
<td>CuDNN</td>
</tr>
<tr>
<td>CUDA</td>
<td></td>
</tr>
<tr>
<td>NV kernel driver (userspace interface)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linux kernel</th>
<th>NV Kernel driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU hardware</td>
<td></td>
</tr>
</tbody>
</table>
Using GPUs through PySpark

- Popular choice for many independent tasks
- Many DL packages have Python interfaces: TensorFlow, Theano, Caffe, MXNet, etc.

- Lifetime for python packages: the process
- Requires some configuration tweaks in Spark
PySpark recommendation

- `spark.executor.cores = 1`
  - Gives the DL framework full access over all the resources
  - Important for frameworks that optimize processor pipelines
Outline

• Deep Learning in data pipelines
• Recurring patterns in Spark + Deep Learning integrations
• Developer tips
• Monitoring
Monitoring

- How do you monitor the progress of your tasks?
- It depends on the granularity
  - Around tasks
  - Inside (long-running) tasks
Monitoring: Accumulators

- Good to check throughput or failure rate
- Works for Scala
- Limited use for Python (for now, SPARK-2868)
- No “real-time” update

```python
batchesAcc = sc.accumulator(1)

def processBatch(i):
    global acc
    acc += 1
    # Process image batch here

images = sc.parallelize(...)
images.map(processBatch).collect()
```
Monitoring: external system

• Plugs into an external system
• Existing solutions: Grafana, Graphite, Prometheus, etc.
• Most flexible, but more complex to deploy
Conclusion

• Distributed deep learning: exciting and fast-moving space
• Most insights are specific to a task, a dataset and an algorithm: nothing replaces experiments
• Get started with data-parallel jobs
  • Move to cooperative frameworks only when your data are too large.
Challenges to address

For Spark developers
- Monitoring long-running tasks
- Presenting and introspecting intermediate results

For DL developers
- What boundary to put between the algorithm and Spark?
- How to integrate with Spark at the low-level?
Resources

Recent blog posts — http://databricks.com/blog
• TensorFrames
• GPU acceleration
• Getting started with Deep Learning
• Intel’s BigDL

Docs for Deep Learning on Databricks — http://docs.databricks.com
• Getting started
• Spark integration
Thank you!

http://databricks.com/try