Data Analytics and Machine Learning: From Node to Cluster

Understanding use cases to optimize on ARM Ecosystem
Vish Puttagunta
- Technical Program Manager, Linaro
- DSP (TI C64x) optimizations (Image / Signal Processing)
- ARM® NEON™ optimizations upstreamed to opus audio codec
- Data Analysis and Machine Learning: Why ARM

Ganesh Raju
- Tech Lead, Big Data, Linaro
- Brings in Enterprise experience in implementing Big Data solutions
Data Science: Big Picture

- Value Prediction
- Classification
- Transformation
- Correlations
- Causalities
- Wrangling...

- Data Science
- Machine Learning
- Big Data
- High Performance Computing (HPC)

- Data Analytics
- Deep Learning

- Statistics
  - Lin/Log Reg
  - KNN, Decision Trees, PCA...

- Deep Learning/NN
  - CNN
  - RNN...

- Software Tools
  - Python, R...
  - Pandas, scikit-learn, nltk...
  - Caffe, TensorFlow...
  - Spark, MLlib, Hadoop...
Overview

- Basic Data Science use cases
- Pandas Library (Python)
  - Time Series, Correlations, Risk Analysis
- Supervised Learning
  - scikit-learn Library (Python)
- Understand operations done repeatedly
- Open discussion for next steps:
  - Profile, Optimize (ARM Neon, OpenCL, OpenMP..) on a single machine.
  - Scale beyond a single machine
- Ref: https://github.com/viswanath-puttagunta/bkk16_MLPrimer
Goal

● Make the tools and operations work out of the box on ARM
  ○ ARM Neon, OpenCL, OpenMP.
  ○ ‘pip install’ should just work :)

● Why Now? (Hint: Spark)
Pandas Data Analysis (Pandas)

- Origins in Finance for Data Manipulation and Analysis (Python Library)
- DataFrame object
- Repeat Computations:
  - Means, Min, Max, Percentage Changes
  - Standard Deviation
  - Differentiation (Shift and subtract)
  - ...
Op: Rolling Mean (Pandas)
### Op: Percentage Change (Pandas)

```python
rets = df.pct_change()
rets.head()
```

<table>
<thead>
<tr>
<th>Date</th>
<th>AAPL</th>
<th>GE</th>
<th>GOOGL</th>
<th>IBM</th>
<th>KO</th>
<th>MSFT</th>
<th>PEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010-01-04</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>2010-01-05</td>
<td>0.0017309</td>
<td>0.005178</td>
<td>-0.004404</td>
<td>-0.012080</td>
<td>-0.012097</td>
<td>0.000323</td>
<td>0.012084</td>
</tr>
<tr>
<td>2010-01-06</td>
<td>-0.015906</td>
<td>-0.005151</td>
<td>-0.025209</td>
<td>-0.006496</td>
<td>-0.000355</td>
<td>-0.006137</td>
<td>-0.010003</td>
</tr>
<tr>
<td>2010-01-07</td>
<td>-0.001849</td>
<td>0.051780</td>
<td>-0.023279</td>
<td>-0.003462</td>
<td>-0.002485</td>
<td>-0.010400</td>
<td>-0.006356</td>
</tr>
<tr>
<td>2010-01-08</td>
<td>0.006648</td>
<td>0.021538</td>
<td>0.013331</td>
<td>0.010035</td>
<td>-0.018509</td>
<td>0.006897</td>
<td>-0.003280</td>
</tr>
</tbody>
</table>

**Operations:** Shift, Subtract, Divide
Op: Percentage Change Vs Risk (Variance)
Op: Correlation (Pandas)

Pearson Correlation Coef

\[
\frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}
\]

Operations: Mean, Variance, Square Root

Source: https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient
Correlation Heat Maps
Linear Regression (Predictor)

Objective: Fit line/curve to minimize a cost function
Linear Regression (Predictor)...

For Lnr Reg, directly reduces to:
$$\Theta = (X^TX)^{-1} X^T Y$$

Operations: Gradient Descent: Matrix Transforms/Inverse/Multiplications.

Source:
https://www.youtube.com/watch?v=SqA6TujbmWw&list=PLE6Wd9FR--Ecf_5nCbnSQMHqORpiChfJf&index=16
https://youtu.be/WnqQrPNYz5Q?list=PLaXDrXvwY-oDvedS3f4HW0b4KxqgPf_imw&t=284
Logistic Regression (Classification)

\[ \tilde{y}_i = \frac{1}{1 + e^{-x_i \Theta}} \]

Operations: Matrix Transforms/Inverse/Multiplications.

Source: https://www.youtube.com/watch?v=Zc7ouSD0DEQ&index=27&list=PLE6Wd9FR--Ec5nCbnSQMHoRpiChfJf
Artificial Neural Networks (Eg: Predictor)

Objective: Compute parameters to minimize a cost function
Operations: Matrix Transforms/Inverse/Multiplications, Dot Products...

Source: https://www.youtube.com/watch?v=bxk2T-V8XRs
Bayesian Classifiers

Operations (Training): Mean, Standard Deviations, Logs, binning/compare(/histograms)
Note: Some variations based on assumptions on variance (LDA, QDA)

FIGURE 4.4. Left: Two one-dimensional normal density functions are shown. The dashed vertical line represents the Bayes decision boundary. Right: 20 observations were drawn from each of the two classes, and are shown as histograms. The Bayes decision boundary is again shown as a dashed vertical line. The solid vertical line represents the LDA decision boundary estimated from the training data.

Source: An Introduction to Statistical Learning with Applications in R (Springer Texts)
K-Nearest Neighbors (Classifier)

Operations: Euclidean Distance Computations
Note: K is tunable. Typical to repeat computations for lot of k values
Decision Tree (Predictors & Classifiers)

**Operations:** Each split done so that
- Predictor: Minimize Root Mean Square Error
- Classifier: Minimize Gini Index  
  (Note: depth tunable)
Bagging / Random Forests

Operations: Each tree similar to Decision Tree
Note: Typical to see about 200 to 300 trees to stabilize.

Source: An Introduction to Statistical Learning with Applications in R (Springer Texts)
Segway to Spark

- So far, on a single machine with lots of RAM and compute.
  - How to scale to a cluster
- So far, data from simple csv files
  - How to acquire/clean data on large scale
Discussion

● **OpenCL (Deep Learning)**
  ○ Acceleration using GPU, DSPs
  ○ But what about Shamrock? OpenCL backend on CPU.
  ○ Port CUDA kernels into OpenCL kernels in various projects?

● **ARM NEON**
  ○ Highly scalable

● **OpenMP?**

● **Tools to profile end-to-end use cases?**
  ○ Perf..
Backup Slides

● Text Analytics
  ○ Sparse Matrices and operations
  ○ Word counts (Naive Bayes)

● Libraries (Python)
  ○ sklearn (No GPU Support!! FAQ)
  ○ gensim (Text Analytics)
  ○ TensorFlow™ (GPU backend available)
  ○ Caffe (GPU backend available)
  ○ Theano (GPU backend available)
Data science in Distributed Environment

Presented by
Ganesh Raju
Tech Lead, Big Data
Linaro

Date
Thursday 10 March 2016

Event
BKK16
Overview

1. Review of Data Science in Single Node
2. Data Science in Distributed Environment
   a. Hadoop and its limitations
3. Data Science using Apache Spark
   a. Spark Ecosystem
   b. Spark ML
   c. Spark Streaming
   d. Spark 2.0 and Roadmap
4. Q & A
Hadoop Vs Spark

80+ operators compared to 2 operators in Hadoop

map()  
reduce()  
filter()  
join()  
groupByKey()  
first()  
) count()
Hadoop and its limitations

Data loading is expensive

Mapper → (intermediates) → Partitioner

Mapper → (intermediates) → Partitioner

Mapper → (intermediates) → Partitioner

Mapper → (intermediates) → Partitioner

Shuffling is expensive

Reducer

Reducer

Reducer
Spark - Unified Platform

source: www.databricks.com
Why in Spark:

- Machine learning algorithms are
  - Complex, multi-stage
  - Iterative
- MapReduce/Hadoop unsuitable
Spark is In-Memory and Fast

source: http://www.slideshare.net/databricks/2015-0317-scala-days
Apache Spark Stack
Unified Engine across diverse workloads and Environments

- Spark SQL + DataFrames
- Streaming
- MLib / ML Machine Learning
- GraphX Graph Computation

Spark Core API
- R
- Python
- Scala
- Java
Spark Core
Internals

SparkContext connects to a cluster manager Obtains executors on cluster nodes Sends app code to them Sends task to the executors
PySpark enables developers to write driver programs in Python
- Application code is serialized and sent to the worker nodes
- Execution happens in native language (Python/R)
Apache Spark MLlib / ML Algorithms Supported:

- **Classification**: Logistic Regression, Linear SVM, Naive Bayes, classification tree
- **Regression**: Generalized Linear Models (GLMs), Regression Tree
- **Collaborative Filtering**: Alternating Least Squares (ALS), Non-Negative Matrix Factorization (NMF)
- **Clustering**: K-Means
- **Decomposition**: SVD, PCA
- **Optimization**: Stochastic Gradient Descent (SGD), L-BFGS
Spark Streaming

- Run a streaming computation as a series of very small, deterministic batch jobs
  - Live data streaming is converted into micro batches of input. Batch can be of ½ sec latency.
  - Each batch is processed in Spark
  - Output is returned as micro batches
  - Potential for combining batch and streaming processing.
- Linear models can be trained in streaming fashion
- Model weights can be updated via SGD, thus amenable to streaming
- Consistent API
- Scalable
Apache Spark
General-purpose cluster computing system

- Unified End-to-End data pipeline platform with support for Machine Learning, Streaming, SQL and Graph Pipelines
- Fast and Expressive Cluster Computing Engine. No Intermediate storage. In-memory Processing. Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.
- Rich higher level APIs in Scala, Python, R, Java. Typically less code (2-5x)
- REPL
- Interoperability with other ecosystem components
  - Mesos, YARN
  - EC2, EMR
  - HDFS, S3,
  - HBase, Cassandra, Parquet, Hive, JSON, ElasticSearch
Major Features in 2.0 - Project Tungsten

- Tungsten Phase 2 speedups of 5-10x
- Structured Streaming real-time engine on SQL / DataFrames
- Unifying Datasets and DataFrames

Focus on CPU and Memory Optimization

**Code Generation**
- Runtime Code Generation by dynamically generating bytecode during evaluation and eliminating Java boxing of primitive data types
- Faster serializer and compression on dataformat like Parquet.

**Manual Memory Management and Binary Processing**
- Off heap memory management. Avoiding non-transient Java objects (store them in binary format), which reduces GC overhead.
- Minimizing memory usage through denser in-memory data format, with less spill (I/O)
- Better memory accounting (size of bytes) rather than relying on heuristics
- For operators that understand data types (in the case of DataFrames and SQL), work directly against binary format in memory, i.e. have no serialization/deserialization

**Cache-aware Computation**
- Exploiting cache locality. Faster sorting and hashing for aggregations, joins, and shuffle
Unified API, One Engine, Automatically Optimized

Language Frontend
- SQL
- Python
- Java/Scala
- R
- ...

DataFrame Logical Plan

Tungsten Backend
- JVM
- LLVM
- OpenCL
- GPU
- NVRAM
- SSE/SIMD
- ...

Supports:
- LLVM
- JVM
- OpenCL
- GPU
- NVRAM
- SSE/SIMD
Apache Spark Roadmap

- Use LLVM for compilation
- Re-implement parallelizable code using OpenCL/SSE/SIMD to utilize underlying CPU / GPU advancements towards machine learning.

Spark slave nodes can achieve better performance and energy efficiency if with GPU acceleration by doing further data parallelization and algorithm acceleration.
De facto Analytics Platform

source: www.databricks.com
Q & A